Homework 12 Solution

Math 315, Fall 2019

Exercise 4.4

```
load(url("http://www4.stat.ncsu.edu/~reich/BSMdata/guns.RData"))
```

(a)

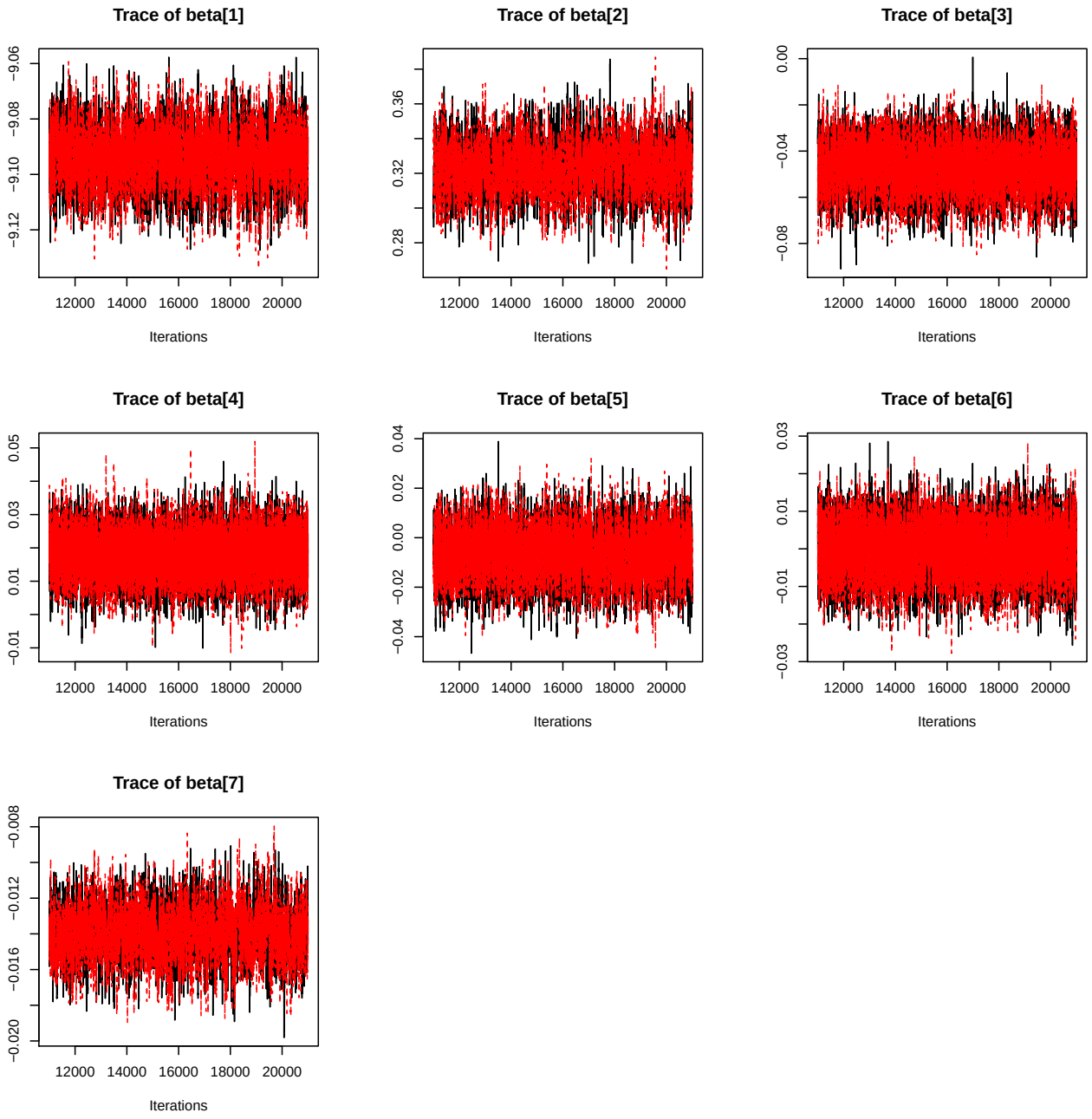
First, we fit the model $Y_i|\boldsymbol{\beta} \sim \text{Poisson}(N_i\lambda_i)$ where $\log(\lambda_i) = \boldsymbol{X_i}\boldsymbol{\beta}$, where we use our standard uninformative independent priors for the regression coefficients. I am running two chains below, but any number 1-5 seems reasonable depending on your convergence philosophy.

```
Z     <- scale(Z)
nlaws <- rowSums(X)
data4   <- list(n = 50, p = 7, Y = Y, N = N, X = nlaws, Z = Z)

model_string4a <- textConnection("model{
   for(i in 1:n){
     Y[i]        ~ dpois(lam[i])
     log(lam[i]) =  log(N[i]) + beta[1] + inprod(Z[i,],beta[2:6]) +
                    X[i]*beta[7]
   }
   for(j in 1:p){beta[j] ~  dnorm(0, 0.001)}
 }")

model4a <- jags.model(model_string4a, data = data4, n.chains = 2, quiet = TRUE)
update(model4a, 10000, progress.bar = "none")
samples4a <- coda.samples(model4a,
                          variable.names = "beta",
                          n.iter = 10000, progress.bar="none")
```

Next, we need to check convergence. This could be done graphically and/or numerically. Based on either type of check, the MCMC sampler appears to have converged to posterior of $\boldsymbol{\beta}$, and the chains are well mixed.

```
gelman.diag(samples4a)
```

```
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## beta[1]          1       1.00
## beta[2]          1       1.01
## beta[3]          1       1.00
## beta[4]          1       1.00
## beta[5]          1       1.01
## beta[6]          1       1.00
## beta[7]          1       1.01
##
## Multivariate psrf
##
## 1
```



Now that we know our sampler has converged, we can summarize the posterior. I recommend including the posterior mean, SD, and 95% interval for regression models. Be sure to make the row and column names readable (this is advice for your project).

From the posterior summary we can see that, after controlling for the confounders, increasing the number of gun laws is associated with a decrease in death rate.

```
library(broom)
sum4a <- tidyMCMC(samples4a, conf.int = TRUE)
colnames(sum4a) <- c("Term", "Mean", "SD", "2.5%", "97.5%")
sum4a$Term <- c("Intercept", paste0("Z", 1:5),"Num. laws")
knitr::kable(sum4a, digits = 3)
```

Term	Mean	SD	2.5%	97.5%
Intercept	-9.094	0.010	-9.114	-9.074
Z1	0.324	0.015	0.295	0.353
Z2	-0.047	0.011	-0.068	-0.026
Z3	0.018	0.007	0.004	0.032
Z4	-0.006	0.010	-0.026	0.014
Z5	0.000	0.007	-0.014	0.013
Num. laws	-0.014	0.001	-0.017	-0.011

(b)

You should still check convergence, but I'll omit that code here to save space.

You should find that the posterior distribution of the gun-laws coefficient is similar under both models.

```
model_string4b <- textConnection("model{
   for(i in 1:n){
     Y[i]          ~ dnegbin(q[i],m)
     q[i]         <- m/(m + lam[i])
     log(lam[i])   = log(N[i]) + beta[1] + inprod(Z[i,],beta[2:6]) +
                     X[i]*beta[7]
   }
   for(j in 1:p){beta[j] ~  dnorm(0, 0.001)}
   m ~ dgamma(0.1,0.1)
 }")

model4b <- jags.model(model_string4b, data = data4, n.chains = 2, quiet = TRUE)
update(model4b, 10000, progress.bar = "none")
samples4b <- coda.samples(model4b,
                          variable.names = "beta",
                          n.iter=10000, progress.bar = "none")
```

Term	Mean	SD	2.5%	97.5%
Intercept	-9.074	0.030	-9.132	-9.015
Z1	0.308	0.038	0.234	0.381
Z2	-0.044	0.026	-0.095	0.006
Z3	0.010	0.021	-0.032	0.051
Z4	0.008	0.025	-0.042	0.058
Z5	0.012	0.026	-0.038	0.063
Num. laws	-0.018	0.005	-0.027	-0.008

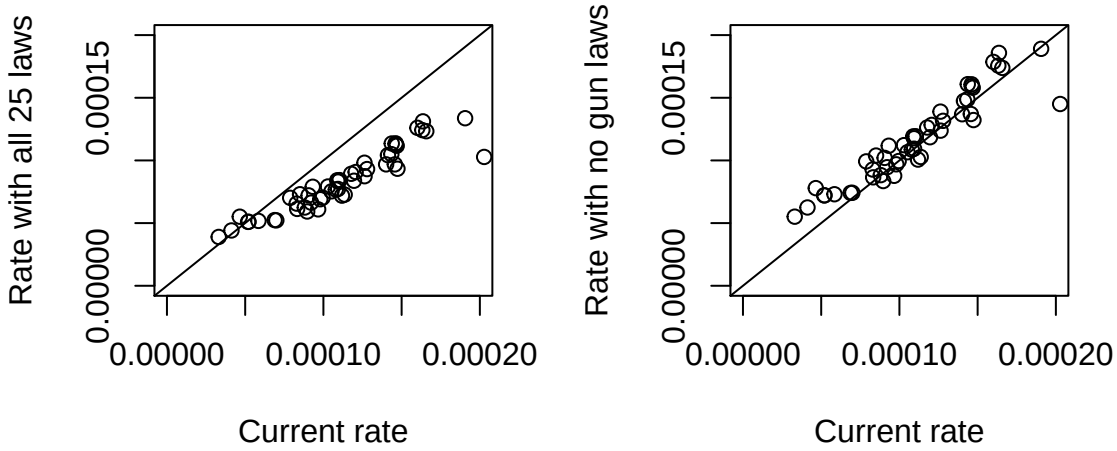
(c)

```
model_string4c <- textConnection("model{
   # Model
   for(i in 1:n){
     Y[i]          ~ dpois(lam[i])
     log(lam[i]) =  log(N[i]) + beta[1] + inprod(Z[i,], beta[2:6]) + X[i]*beta[7]
   }
   for(j in 1:p){beta[j] ~  dnorm(0, 0.001)}
```

```
  # PPD
  for(i in 1:n){
    log(lam0[i])  =  log(N[i]) + beta[1] + inprod(Z[i,], beta[2:6]) +
                     0*beta[7]
    log(lam25[i]) = log(N[i]) + beta[1] + inprod(Z[i,], beta[2:6]) +
                     25*beta[7]
    Y0[i]  ~ dpois(lam0[i])
    Y25[i] ~ dpois(lam25[i])
  }

 }")

model4c <- jags.model(model_string4c, data = data4, n.chains = 2, quiet=TRUE)
update(model4c, 10000, progress.bar="none")
samples4c <- coda.samples(model4c,
                          variable.names = c("Y0","Y25"),
                          n.iter = 10000, progress.bar = "none")

# Post pred checks
est <- summary(samples4c)$quantiles[,3]
R   <- Y/N
R0  <- est[1:50]/N
R25 <- est[1:50+50]/N
```

We can compare the predicted number of deaths to the current number of deaths using scatterplots:

```
par(mfrow = c(1, 2))
 plot(R, R25, xlim=c(0 , 0.0002), ylim=c(0, 0.0002),
      xlab = "Current rate", ylab = "Rate with all 25 laws")
 abline(0,1)

  plot(R, R0, xlim=c(0, 0.0002), ylim = c(0, 0.0002),
      xlab = "Current rate",ylab = "Rate with no gun laws")
 abline(0,1)
```



These results suggest a fairly dramatic effect of gun laws, but it is hard to trust them because they result from an observational study. The estimated effect could be much different if, say, they were derived from a randomized trial where causal inferences can be made.

Exercise 5.1

The below code implements 5-fold cross validation. One wrinkle to remember is that you need to make sure that you tune each sampler to be sure that it converges!

```
# Data wrangling
# Using model.frame will toss out NAs, otherwise you should use na.omit()
mod_data <- model.frame(lm(Ozone ~ Solar.R + Temp + Wind, data = airquality))
X <- cbind(1, mod_data[,-1])
colnames(X) <- c("int", "solar.r", "temp", "wind")

Y <- mod_data[,1]

# Specifying model
# Notice that I leave it vauge enough for use with both models
m <- "model{
     for(i in 1:n){
       Y[i] ~ dnorm(mu[i], tau)
       mu[i] <- inprod(X[i,], beta[])
     }
     for(j in 1:p){beta[j] ~ dnorm(0,0.001)}
     tau     ~ dgamma(0.1, 0.1)
   }"

# Randomly assign observations to k=5 folds
set.seed(11192019)
fold <- rep(1:5, length.out = nrow(X))
fold <- sample(fold)

# Storage
Y_mean   <- matrix(NA, nrow(X), 2)
Y_low    <- matrix(NA, nrow(X), 2)
Y_high   <- matrix(NA, nrow(X), 2)

# params to follow
params <- c("beta", "tau")

# Select training data with fold not equal to f
for(f in 1:5){
  d1 <- list(Y = Y[fold!=f], n = sum(fold != f), X = X[fold != f, 1:2], p = 2)
  d2 <- list(Y = Y[fold!=f], n = sum(fold != f), X = X[fold != f,], p = 4)

  # Model 1
  model1 <- jags.model(textConnection(m), data = d1, n.chains = 1, quiet = TRUE)
  update(model1, 10000, progress.bar="none")
  b1 <- coda.samples(model1, variable.names = params, thin = 5, n.iter = 20000, progress.bar="none")[[1]

  # Model 2
  model2 <- jags.model(textConnection(m), data = d2, n.chains = 1, quiet = TRUE)
  update(model2, 10000, progress.bar="none")
  b2 <- coda.samples(model2, variable.names = params, thin = 30, n.iter = 250000, progress.bar="none")[

  # Make predictions
  post1.est <- summary(b1)$statistics
```

```
  post2.est <- summary(b2)$statistics
  for(i in 1:nrow(X)) {
    if(fold[i] == f){
      Y1 <- rnorm(n = nrow(b1), mean = as.numeric(X[i,1:2]) %*% post1.est[1:2,1], sd = 1/sqrt(post1.est
      Y_mean[i, 1] <- mean(Y1)
      Y_low[i,1]   <- quantile(Y1, 0.025)
      Y_high[i,1]  <- quantile(Y1, 0.975)

      Y2 <- rnorm(n = nrow(b2), mean = as.numeric(X[i,]) %*% post2.est[1:4,1], sd = 1/sqrt(post2.est[5,
      Y_mean[i, 2] <- mean(Y2)
      Y_low[i, 2]  <- quantile(Y2, 0.025)
      Y_high[i, 2] <- quantile(Y2, 0.975)
    }
  }

  rm(model1, model2)

}

# Calculating metrics
cv_results <- rbind(
  bias  = colMeans(Y_mean - Y),
  mse   = colMeans((Y_mean-Y)^2),
  mad   = colMeans(abs(Y_mean-Y)),
  cov   = colMeans( (Y_low <= Y) & (Y <= Y_high)),
  width = colMeans(Y_high - Y_low)
)
colnames(cv_results) <- c("m1", "m2")
```

Model 2, has smaller MSE and MAD, and better coverage and narrower widths of the prediction intervals, indicating that it is preferred.

	m1	m2
bias	-0.196	-0.556
mse	987.990	486.005
mad	24.454	16.310
cov	0.937	0.955
width	122.405	82.740

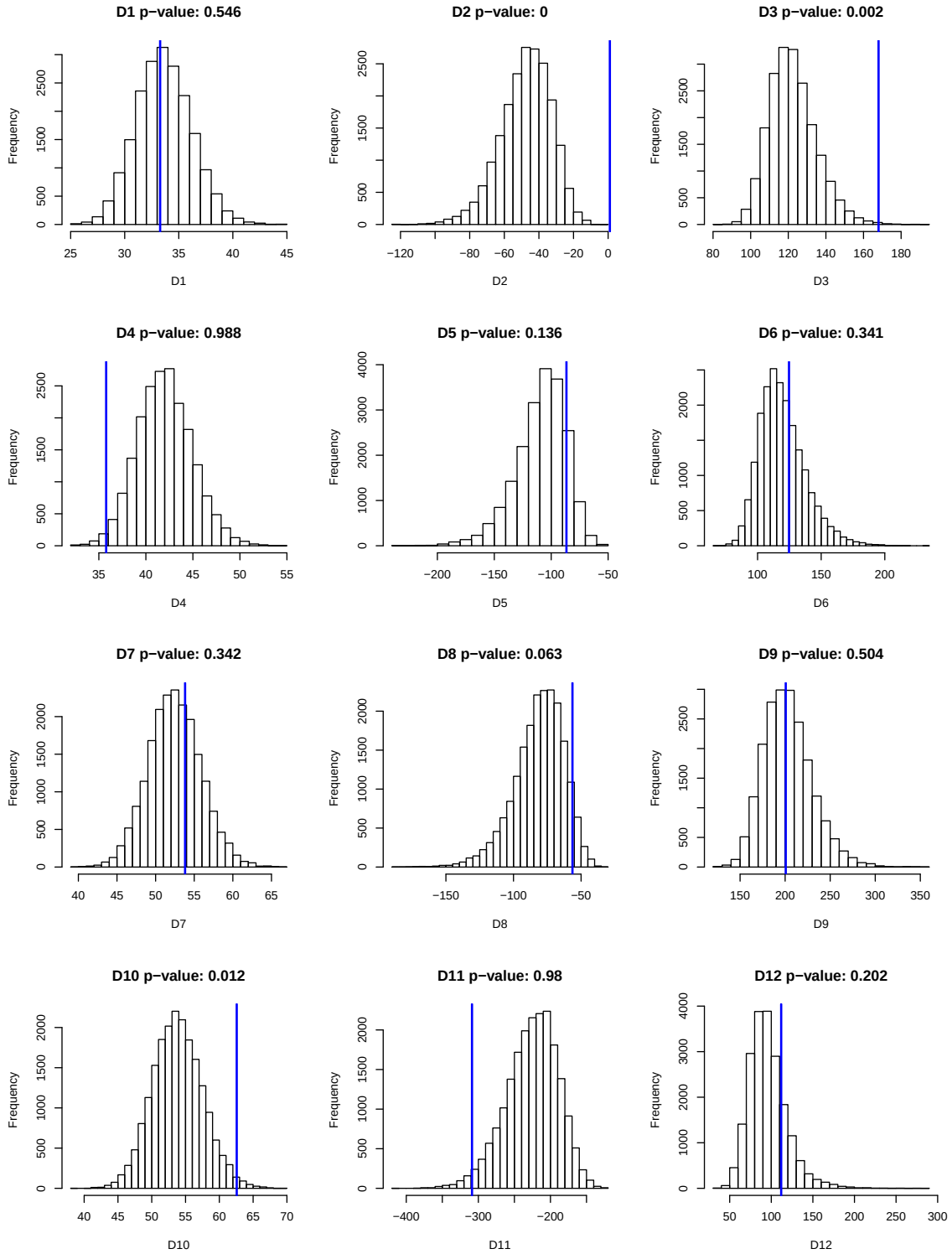
Exercise 5.2

The model does not fit the tails well. The PPD for the minimum is usually negative (D2) and the PPD for the maximum is less than the observed max with high probability (D3). A log transformation might help.

```
# Data wrangling
# Using model.frame will toss out NAs, otherwise you should use na.omit()
mod_data <- model.frame(lm(Ozone ~ Solar.R + Temp + Wind, data = airquality))
X <- scale(mod_data[,-1])
Y <- mod_data[,1]

# Specifying the model and PPDs
mod52 <- textConnection("model{
```

```
 for(i in 1:n){
   Y[i]   ~ dnorm(mu[i],tau)
   mu[i] <- beta[1] + X[i,1]*beta[2] +
            X[i,2]*beta[3] + X[i,3]*beta[4]
 }
 for(j in 1:4){beta[j] ~ dnorm(0,0.001)}
 tau ~ dgamma(0.1,0.1)

 #PPD checks
 for(i in 1:n){
  Yp[i]  ~ dnorm(mu[i],tau)
  X1[i] <- X[i,1]*Yp[i]
  X2[i] <- X[i,2]*Yp[i]
  X3[i] <- X[i,3]*Yp[i]
 }
 D[1]  <- sd(Yp[])
 D[2]  <- min(Yp[])
 D[3]  <- max(Yp[])
 D[4]  <- sd(X1[])
 D[5]  <- min(X1[])
 D[6]  <- max(X1[])
 D[7]  <- sd(X2[])
 D[8]  <- min(X2[])
 D[9]  <- max(X2[])
 D[10] <- sd(X3[])
 D[11] <- min(X3[])
 D[12] <- max(X3[])

}")

data52  <- list(Y = Y, X = X, n = nrow(X))
model52 <- jags.model(mod52, data = data52, n.chains=1,quiet=TRUE)
update(model52, 5000, progress.bar="none")
D     <- coda.samples(model52, variable.names=c("D"),
                      n.iter=20000, progress.bar="none")[[1]]

Do     <- rep(0,12)
Do[1]  <-  sd(Y)
Do[2]  <- min(Y)
Do[3]  <- max(Y)
Do[4]  <-  sd(Y*X[,1])
Do[5]  <- min(Y*X[,1])
Do[6]  <- max(Y*X[,1])
Do[7]  <-  sd(Y*X[,2])
Do[8]  <- min(Y*X[,2])
Do[9]  <- max(Y*X[,2])
Do[10] <-  sd(Y*X[,3])
Do[11] <- min(Y*X[,3])
Do[12] <- max(Y*X[,3])

par(mfrow = c(4, 3))
for(j in 1:12){
   pval <- mean(Do[j] < D[,j])
```

```
  hist(D[,j], breaks=25, xlab = paste0("D",j) , main = paste0("D", j," p-value: ", round(pval,3)))
  abline(v = Do[j], col = "blue", lwd = 2)
}
```



Exercise 5.5

```
# Load data and manipulate for JAGS
data("gambia", package = "geoR")
Y   <- gambia$pos
X   <- gambia[,4:8]
X   <- scale(X)

# Fit logistic model
logit_mod <- textConnection("model{
 for(i in 1:n){
  Y[i] ~ dbern(pi[i])
  logit(pi[i]) <- beta[1] + X[i,1]*beta[2] +
                  X[i,2]*beta[3] + X[i,3]*beta[4] +
                  X[i,4]*beta[5] + X[i,5]*beta[6]
  like[i] <- dbin(Y[i],pi[i],1) # For WAIC computation
 }
 for(j in 1:6){beta[j] ~ dnorm(0,0.01)}
}")


data.55  <- list(Y = Y, X = X, n = length(Y))
model.55.log <- jags.model(logit_mod, data = data.55, n.chains = 2, quiet=TRUE)
update(model.55.log, 5000, progress.bar="none")
samps.log <- coda.samples(model.55.log, variable.names = "like",
                          n.iter = 20000, progress.bar = "none")
```

```
# Compute DIC
DIC_logit  <- dic.samples(model.55.log, n.iter = 20000, progress.bar = "none")
```

```
# Compute WAIC
like.log   <- rbind(samps.log[[1]], samps.log[[2]]) # Combine the two chains
fbar.log   <- colMeans(like.log)
Pw.log     <- sum(apply(log(like.log), 2, var))
WAIC_logit <- -2*sum(log(fbar.log)) + 2*Pw.log
```

```
# Fit probit model
probit_mod <- textConnection("model{
 for(i in 1:n){
  Y[i] ~ dbern(pi[i])
  probit(pi[i]) <- beta[1] + X[i,1]*beta[2] +
                   X[i,2]*beta[3] + X[i,3]*beta[4] +
                   X[i,4]*beta[5] + X[i,5]*beta[6]
  like[i] <- dbin(Y[i],pi[i],1) # For WAIC computation
 }
 for(j in 1:6){beta[j] ~ dnorm(0,0.01)}
}")


model.55.probit <- jags.model(probit_mod, data = data.55, n.chains=2, quiet=TRUE)
update(model.55.probit, 5000, progress.bar="none")
samps <- coda.samples(model.55.probit, variable.names=c("like"),
                      n.iter=20000, progress.bar="none")
```

```
# Compute DIC
DIC_probit  <- dic.samples(model.55.probit,n.iter=20000,progress.bar="none")
```

```
# Compute WAIC
like        <- rbind(samps[[1]],samps[[2]]) # Combine the two chains
fbar        <- colMeans(like)
Pw          <- sum(apply(log(like),2,var))
WAIC_probit <- -2*sum(log(fbar)) + 2*Pw
```

Both criterion are very similar for both link functions, but slightly favor the logit link.

Model	DIC	WAIC
Logistic	2525.7	2525.7
Probit	2526.8	2527