Applied Bayesian Analysis : NCSU ST 540

Homework 7

Bruce Campbell

In this assignment we will performing random slopes logistic regression in JAGS using the Gambia data described in `http://www4.stat.ncsu.edu/~reich/ABA/code/GLM` Let Y_i be the binary response for individual i , and let $\nu_i \in 1 \cdots 65$ denote the village of individual i Let $X_i = 1$ if individual i regularly sleeps under a bed-net and $X_i = 0$ otherwise. Fit the model

$$logit(P(Y_i = 1)) = \alpha_{\nu_i} + X_i \beta_{\nu_i}$$

where α_{ν_i} and β_{ν_i} are the intercept and slope for village j The priors (independent over village and with each other) are

$$\alpha_{\nu_i} \sim Normal(\mu_a, \sigma_a^2)$$

and

$$\beta_{\nu_i} \sim Normal(\mu_b, \sigma_b^2)$$

We choose uninformative priors for $\mu_a, \sigma_a^2, \mu_b, \sigma_b^2$

Before we perform the analysis we address the question of why might the effect of bed-net vary by village? The effect of bed-net use on malaria status could vary based on many factors. We list a few below;

- proximality to wet area with mosquito larve
- the amount of time childeren play outside or in areas near mosquito infested areas
- variation in the level of malaria parasites in the mosquito population

Model definition and MCMC sampling

```
library(rjags)
library(coda)
library(modeest)

DEBUG <- FALSE
if(DEBUG)
{
nSamples <- 10000
n.chains <- 1
} else
{
nSamples <- 20000
n.chains <- 8
}
```

```
load("gambia.RData")

X.net <- as.numeric((X$netuse==1) | (X$treated==1))
Y <- pos
n <- length(X.net)

df <- data.frame(cbind(X.net,village,pos))
names(df) <- c("net","village","pos")
numVillages <- length(unique(df$village))
villages <- df$village

model_string.logistic_random_slopes <- "model{
   # Likelihood
   for(i in 1:n){
      Y[i]    ~ dbern(q[i])
      logit(q[i]) <- beta[villages[i],1] + beta[villages[i],2]*X.net[i]
   }

   # Random effects
   for(j in 1:numVillages){
    beta[j,1] ~dnorm(mu1,precision1)
    beta[j,2] ~dnorm(mu2,precision1)
   }

  #Priors - uninformative.
  mu1 ~ dnorm(0, 0.01)
  mu2 ~ dnorm(0, 0.01)
  precision1 ~ dgamma(0.01, 0.01)
  precision2 ~ dgamma(0.01, 0.01)

  for(j in 1:numVillages){
    pred[j] <- beta[j,1] + beta[j,2]
  }
  }"


model.logistic_random_slopes <- jags.model(textConnection(model_string.logistic_random_slopes)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2035
##    Unobserved stochastic nodes: 134
##    Total graph size: 6629
##
## Initializing model
```

```
update(model.logistic_random_slopes, nSamples, progress.bar="none"); # Burnin
samp.coeff.logistic_random_slopes <- coda.samples(model.logistic_random_slopes, variable.names=
```

MCMC Chain diagnostics

Chain diagnostics are optionally produced with the deubg flag. These were run and inspected by hand for chain convergence. The debug flag is set to false for rendering the final markdown in the report. For the final version we calculate the gelman scale reduction factors for each parameter. The scalre reduction factor near 1 means that between and within chain variance are nearly equal. We report the number of coefficients with a gelman scale reduction factor greater than one.

```
gelman.srf <- gelman.diag(samp.coeff.logistic_random_slopes)

count.coeff.gt <- sum(gelman.srf$psrf >
    1.1)

pander(data.frame(count.coeff.gt = count.coeff.gt),
    caption = "Number of coefficients with a a gelman scale reduction factor greater than 1.1")
```

Table 1: Number of coefficients with a a gelman scale reduction factor greater than 1.1

count.coeff.gt
0

```
if (DEBUG) {
    autocorr.plot(samp.coeff.logistic_random_slopes)

    plot(samp.coeff.logistic_random_slopes)

    # Sample again and estimate posterior
    # means and MAP posterior modes.
    samp.coeff.logistic_random_slopes.jags <- jags.samples(model.logistic_random_slopes,
        variable.names = c("beta"), n.iter = nSamples,
        progress.bar = "none")
    posterior_means.logistic_random_slopes <- lapply(samp.coeff.logistic_random_slopes.jags,
        apply, 1, "mean")
    pander(posterior_means.logistic_random_slopes,
        caption = "posterior means second sample")

    posterior_modes.logistic_random_slopes <- lapply(samp.coeff.logistic_random_slopes.jags,
        apply, 1, "mlv")
    posterior_modes.logistic_random_slopes$beta[[1]]$M

    if (n.chains > 1) {
        gelman.plot(samp.coeff.logistic_random_slopes)
```

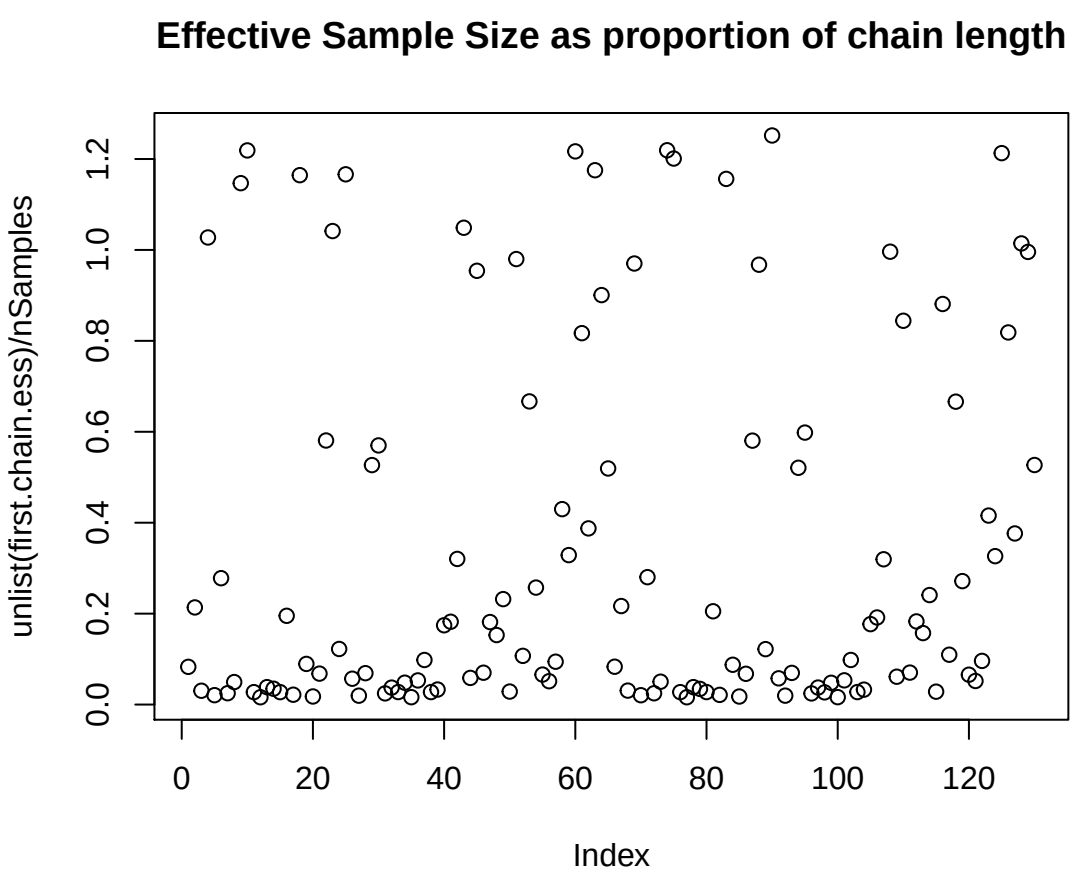
```
    }
}
```

Calculate the effective samples size of the chains

We calculate the efective sample size (ESS) below and make a plot for the first chain. We noted sigificant autocorrelation in some of the afc plots for the intercepts and slope parameters. As a follow up we might run the MCMC chains for a lot longer on more compute enabled hardware. We'd also like to investigate the relationship between coefficients with low ESS and those which are not deemed significant by the 95% credible intervals.

```
chains.ess <- lapply(samp.coeff.logistic_random_slopes,
    effectiveSize)

first.chain.ess <- chains.ess[1]
plot(unlist(first.chain.ess)/nSamples,
    main = "Effective Sample Size as proportion of chain length")
```



Calculate posterior means and MAP estimates for interceps and slopes

```
# Allocate matrix for means and modes
# of the intercepts
intercepts.means.modes <- data.frame(mean = rep(NaN,
    65), mode = rep(NaN, 65))

# First get a chain
chain <- samp.coeff.logistic_random_slopes[[1]]
# Alpha first
for (i in 1:65) {
    colname <- colnames(chain)[i]

    samples <- chain[, i]

    sample.mode <- mlv(samples)

    sample.mean <- mean(samples)

    intercepts.means.modes[i, 1] <- sample.mean

    intercepts.means.modes[i, 2] <- sample.mode$M

}

intercept.max.mean <- which.max(intercepts.means.modes$mean)
intercept.max.mode <- which.max(intercepts.means.modes$mode)

pander(data.frame(intercept.max.mean.index = villages[intercept.max.mean],
    intercept.max.mode.index = villages[intercept.max.mode]),
    caption = "Village index of max mean and mode for intercept posteriors")
```

Table 2: Village index of max mean and mode for intercept posteriors

intercept.max.mean.index	intercept.max.mode.index
2	2

```
# Allocate matrix for means and modes
# of the slopes
slopes.means.modes <- data.frame(mean = rep(NaN,
    65), mode = rep(NaN, 65))

# First get a chain
chain <- samp.coeff.logistic_random_slopes[[1]]
# Alpha first
```

```
for (i in 66:130) {
    colname <- colnames(chain)[i]

    samples <- chain[, i]

    sample.mode <- mlv(samples)

    sample.mean <- mean(samples)

    slopes.means.modes[i - 65, 1] <- sample.mean

    slopes.means.modes[i - 65, 2] <- sample.mode$M

}
rownames(slopes.means.modes) <- c()

slope.max.mean <- which.max(slopes.means.modes$mean)
slope.max.mode <- which.max(slopes.means.modes$mode)

pander(data.frame(slope.max.mean.index = villages[slope.max.mean],
    intercept.max.mode.index = villages[slope.max.mode]),
    caption = "Village index of max mean and mode for slope posteriors")
```

Table 3: Village index of max mean and mode for slope posteriors

slope.max.mean.index	intercept.max.mode.index
2	2

Conclusions

(2)MCMC algorithm converge? (3) Do you see evidence that the slopes and/or intercepts vary by village? (4) Which village has the largest intercept? Slope? Does this agree with the data in these villages?

The MCMC algorithm converged for all the parameters. We saw evidence that the slopes and intercepts vary by village. Below we plot the table for the villages max intercept and slope. We see large values for $(net, pos) = (0, 0)$ and $(net, pos) = (1, 1)$ for these villages.

```
tables <- table(df$net, df$pos, df$village)

pander(tables[, , slope.max.mean], caption = "freq for max posterior mean for slope coefficien
```

	0	1

Table 4: freq for max posterior mean for slope coefficient

	0	1
0	18	11
1	0	2

```
pander(tables[, , intercept.max.mean],
    caption = "freq for max posterior mean for intercept coefficient")
```

Table 5: freq for max posterior mean for intercept coefficient

	0	1
0	0	2
1	1	12