Generalized linear models using JAGS and R

Andrew O. Finley

January 25, 2013

Generalized linear models

We make use of several libraries in the following example session, including:

- `library(coda)`
- `library(rjags)`
- `library(fields)`
- `library(MBA)`
- `library(geoR)`

Let's consider the *logistic* model where the outcome of the i-th sample unit, y_i is either 1 or 0. In addition to the outcome, we often have a set of covariates observed at the sample units. Given these data, we can write the model as

$$\pi(y_i \,|\, \eta_i) \sim Ber(p(\eta_i)), \text{ for } i = 1, \ldots, n \tag{1}$$

where $p(\eta_i) = \exp(\eta_i)/(1 + \exp(\eta_i))$ is the probability of $y_i = 1$ and

$$\eta_i = \boldsymbol{x}_i'\boldsymbol{\beta}. \tag{2}$$

As in the ordinary linear regression model, $\boldsymbol{x}_i = (x_{i,0}, x_{i,1}, \ldots, x_{i,p})'$ is a $p \times 1$ vector (with x_{i0} set to one, i.e., the intercept) and $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)'$ is a $p \times 1$ vector of regression coefficients.

1 Simulated data analysis

Let's begin by simulating some data from model (1).

```
> set.seed(1)
> n <- 100
> X <- cbind(1, rnorm(n))
> beta <- c(0.1, -0.5)
> eta <- X %*% beta
> p <- 1/(1 + exp(-eta))
> y <- rbinom(n, size = 1, prob = p)
```

Note, $\exp(\eta)/(1 + \exp(\eta))$ and $1/(1 + \exp(-\eta))$ are mathematically equivalent forms of the inverse logit function.

We would like to generate samples from the posterior distributions of β_0 and β_1. Again, it is useful to provide JAGS with some reasonable starting values for these parameters. Note, the call to `glm` includes a `-1` that indicates the intercept is already included in $\boldsymbol{x}$.

```
> glm.m <- glm(y ~ X - 1, family = "binomial")
> summary(glm.m)
```

```
Call:
glm(formula = y ~ X - 1, family = "binomial")

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1663  -1.0622   0.5172   1.0552   1.7453

Coefficients:
   Estimate Std. Error z value Pr(>|z|)
X1   0.1521     0.2179   0.698 0.485090
X2  -0.9454     0.2779  -3.401 0.000671 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 138.63  on 100  degrees of freedom
Residual deviance: 124.33  on  98  degrees of freedom
AIC: 128.33

Number of Fisher Scoring iterations: 4
```

Now we formulate the JAGS model file *ex-3a.jag*.

```
model{

 for(i in 1:n){
  eta[i] <- beta.0 + beta.1*x[i]
  pi[i] <- 1/(1+exp(-eta[i]))
  y[i] ~ dbern(pi[i])
 }

 beta.0 ~ dnorm(0, 0.000001)
 beta.1 ~ dnorm(0, 0.000001)

}
```

As in the previous exercises, we define the data objects needed in the JAGS model along with parameter starting values. Our call to `jags.model` requests three MCMC chains of 5000 iterations.

```
> x <- X[, 2]
> n <- length(y)
> data <- list(y = y, x = x, n = n)
> inits <- list(beta.0 = coefficients(glm.m)[1], beta.1 = coefficients(glm.m)[2])
> jags.m <- jags.model(file = "ex-3a.jag", data = data,
+     inits = inits, n.chains = 3, n.adapt = 1000)

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 806

Initializing model
```

Now we collect the posterior samples.

```
> params <- c("beta.0", "beta.1")
> samps <- coda.samples(jags.m, params, n.iter = 5000)
> plot(samps, density = FALSE)
```

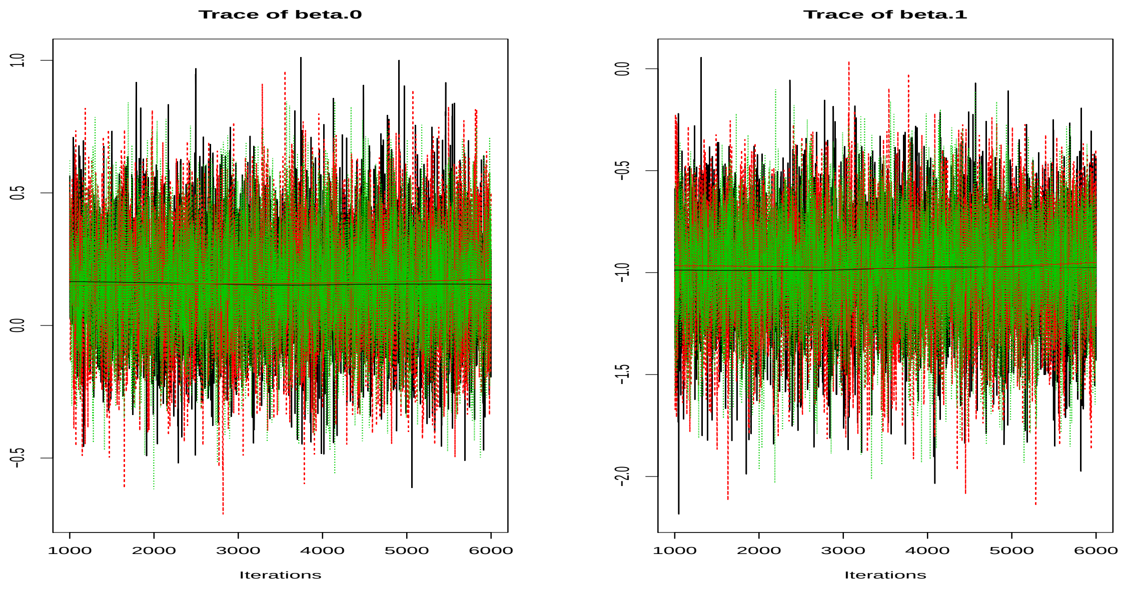

Figure 1: Posterior sample chain and density plots for $\boldsymbol{\beta}$.

We will also check the Gelman-Rubin Diagnostic to assess convergence, Figure 2. Given the *potential scale reduction factors* are all less then ~1.1 and visual inspection of the chains suggests they are mixing well, we can concluded the chains have converged and we can turn our attention to summarizing the posterior samples.

```
> gelman.diag(samps)

Potential scale reduction factors:

       Point est. Upper C.I.
beta.0          1          1
beta.1          1          1

Multivariate psrf

1

> gelman.plot(samps)

> burn.in <- 2000
> round(summary(window(samps, start = burn.in))$quantiles[,
+     c(3, 1, 5)], 2)

         50%  2.5% 97.5%
beta.0  0.16 -0.27  0.60
beta.1 -0.97 -1.57 -0.45
```

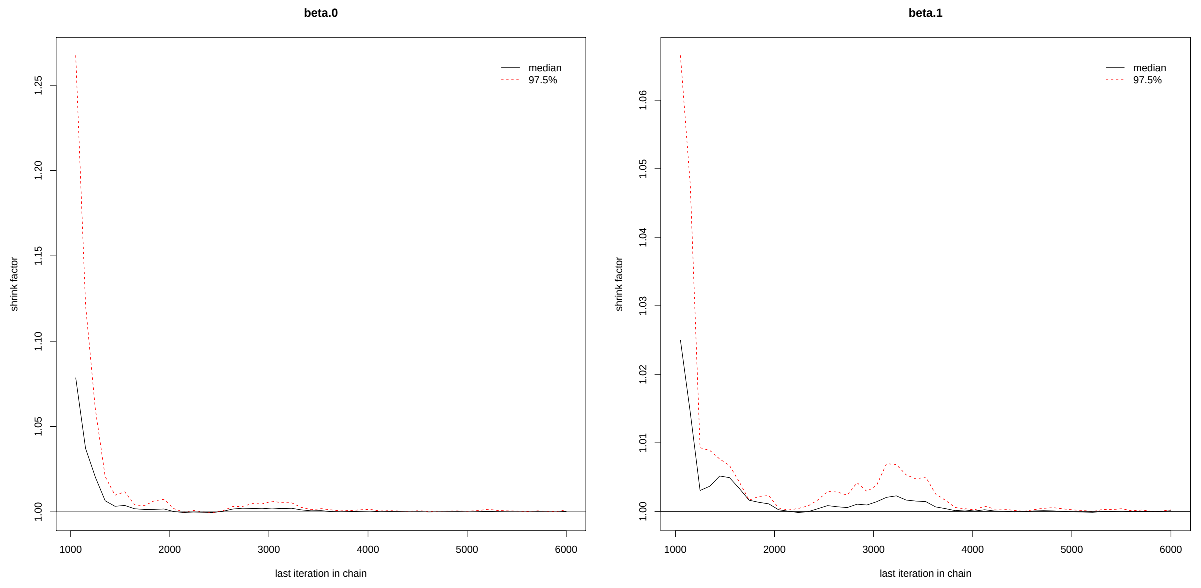

Figure 2: Gelman-Rubin diagnostic plots for $\boldsymbol{\beta}$.

Considering the addition of a random intercept

Say our data fall within three groups and we want to explore $\boldsymbol{y}$ differs with regard to group membership. We could set this model up using *dummy* variables or use a multilevel model. When the number of groups is large or we expect the model to become more complicated, then the multilevel approach is desirable. For the multilevel model specification, we modify model (1) by adding an additional term to η. Specifically,

$$\eta_{ij} = \boldsymbol{x}_i'\boldsymbol{\beta} + \alpha_j, \text{ for } i = 1 \ldots n \text{ and } j = 1 \ldots 3, \tag{3}$$

where α_j is the group specific intercept offset. We assume $\alpha_j \sim N(0, \sigma^2_\alpha)$.

Let's simulate some data to try this model out.

```
> set.seed(1)
> n <- 200
> X <- cbind(1, rnorm(n))
> beta <- c(0.1, -0.5)
> alpha.groups <- sample(1:3, size = n, replace = TRUE)
> Z.alpha <- matrix(0, n, 3)
> Z.alpha[alpha.groups == 1, 1] <- 1
> Z.alpha[alpha.groups == 2, 2] <- 1
> Z.alpha[alpha.groups == 3, 3] <- 1
> head(Z.alpha)
```

```
     [,1] [,2] [,3]
[1,]    0    1    0
[2,]    1    0    0
[3,]    0    0    1
[4,]    0    0    1
[5,]    0    0    1
[6,]    0    0    1
```

```
> sigma.sq.alpha <- 1
> alpha <- rnorm(3, 0, sqrt(sigma.sq.alpha))
> p <- 1/(1 + exp(-(X %*% beta + Z.alpha %*% alpha)))
> y <- rbinom(n, size = 1, prob = p)
```

Now we formulate the JAGS model file *ex-3a.jag*.

```
model{

 for(i in 1:n){
  eta[i] <- beta.0 + beta.1*x[i] + alpha[alpha.indx[i]]
  pi[i] <- 1/(1+exp(-eta[i]))
  y[i] ~ dbern(pi[i])
 }

 beta.0 ~ dnorm(0, 0.000001)
 beta.1 ~ dnorm(0, 0.000001)

 for(j in 1:q){
  alpha[j] ~ dnorm(0, tau.sq.alpha)
 }

 tau.sq.alpha ~ dgamma(0.01,0.01)
 sigma.sq.alpha <- 1/tau.sq.alpha

}
```

As in the previous exercises, we define the data objects needed in the JAGS model along with parameter starting values. Our call to `jags.model` requests three MCMC chains of 10000 iterations.

```
> x <- X[, 2]
> n <- length(y)
> data <- list(y = y, x = x, n = n, q = 3, alpha.indx = alpha.groups)
> inits <- list(beta.0 = coefficients(glm.m)[1], beta.1 = coefficients(glm.m)[2])
> jags.m <- jags.model(file = "ex-3b.jag", data = data,
+     n.chains = 3, n.adapt = 1000)

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 1813

Initializing model

> params <- c("beta.0", "beta.1", "alpha", "sigma.sq.alpha")
> samps <- coda.samples(jags.m, params, n.iter = 10000)
> plot(samps, density = FALSE)
> burn.in <- 2000
> round(summary(window(samps, start = burn.in))$quantiles[,
+     c(3, 1, 5)], 2)

                 50%  2.5% 97.5%
alpha[1]        0.19 -3.43  3.80
alpha[2]       -1.69 -5.34  1.88
```

```
alpha[3]        1.40 -2.13  5.16
beta.0          0.67 -2.94  4.26
beta.1         -0.37 -0.77  0.02
sigma.sq.alpha  3.50  0.60 82.77
```

So, are these parameter values correct? How is chain convergence?

Try your hand at writing some more complex models by, e.g., 1) adding another random offset to the intercept, and 2) allowing the covariate to vary by group, i.e., an intercept and slope varying model.