Using JAGS for concussions data

Chapter 3.3: Introduction to JAGS

The response is the total number of concussions (summing across teams and games) in each year from 2012-2015. We fit the model

$$Y_i \sim \text{Poisson}(N\lambda_i) \text{ where } \lambda_i \sim \text{Gamma}(1, \gamma)$$

and $\gamma \sim \text{Gamma}(0.1, 0.1)$. We have previously coded Gibbs sampling for this problem, and here we verify that we obtain the same results using *JAGS*.

Load concussions data

```
library(rjags)

# Number of concussions in 2012-2015
Y <- c(171, 152, 123, 199)
n <- 4
N <- 256
```

(1) Define the model as a string

```
model_string <- textConnection("model{
  # Likelihood
   for(i in 1:n){
     Y[i] ~ dpois(N*lambda[i])
   }
  # Priors
   for(i in 1:n){
     lambda[i] ~ dgamma(1,gamma)
   }
   gamma   ~  dgamma(a, b)
}")
```

(2) Load the data and compile the MCMC code

```
inits <- list(lambda=rgamma(n,1,1),gamma=1)
data  <- list(Y=Y,N=N,n=n,a=0.1,b=0.1)
model <- jags.model(model_string,data = data, inits=inits, n.chains=2)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 4
##    Unobserved stochastic nodes: 5
##    Total graph size: 21
##
## Initializing model
```

(3) Burn-in for 10000 samples

```
update(model, 10000, progress.bar="none")
```

(4) Generate 20000 post-burn-in samples

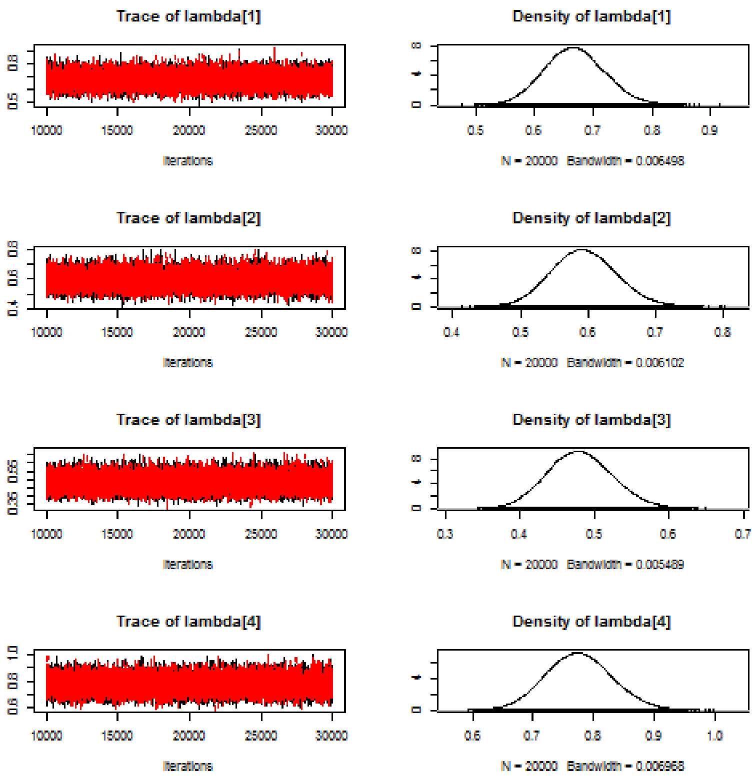
```
params  <- c("lambda")
samples <- coda.samples(model,
           variable.names=params,
           n.iter=20000, progress.bar="none")
```

(5) Summarize the output

```r
summary(samples)
```

```
##
## Iterations = 10001:30000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 20000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean      SD  Naive SE Time-series SE
## lambda[1] 0.6679 0.05120 0.0002560      0.0002653
## lambda[2] 0.5943 0.04793 0.0002396      0.0002405
## lambda[3] 0.4811 0.04311 0.0002155      0.0002155
## lambda[4] 0.7762 0.05473 0.0002736      0.0002736
##
## 2. Quantiles for each variable:
##
##             2.5%    25%    50%    75%  97.5%
## lambda[1] 0.5714 0.6329 0.6666 0.7013 0.7725
## lambda[2] 0.5040 0.5614 0.5931 0.6258 0.6919
## lambda[3] 0.4007 0.4512 0.4798 0.5095 0.5686
## lambda[4] 0.6727 0.7388 0.7750 0.8122 0.8871
```

```r
plot(samples)
```



Loading [MathJax]/jax/output/HTML-CSS/jax.js