ADVANCED BAYESIAN MODELING

BINOMIAL HIERARCHICAL MODEL IN R/JAGS:
RUNNING JAGS IN R

Rat Tumor Example

$$
\begin{aligned}
n_j &= \text{total number of rats in control group of experiment } j \\
y_j &= \text{number in control group of experiment } j \text{ that develop a tumor} \\
\theta_j &= \text{control group tumor probability in experiment } j \\
j &= 1, \ldots, 71
\end{aligned}
$$

Data in file `rattumor.txt`:

```
# Rat tumor data from Tarone (1982).  Data from Table 5.1 of Bayesian
# Data Analysis.
# From:  http://www.stat.columbia.edu/~gelman/book/data/rats.asc

y N
0 20
0 20
0 20
0 20
0 20

...

9 24
4 14
```

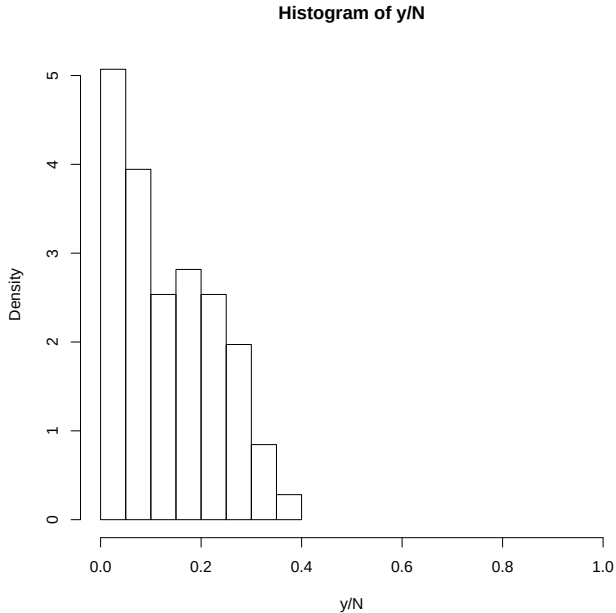
```
> d <- read.table("rattumor.txt", header=TRUE)

> head(d)
  y  N
1 0 20
2 0 20
3 0 20
4 0 20
5 0 20
6 0 20

> summary(d)
       y                N
 Min.   : 0.000   Min.   :10.00
 1st Qu.: 1.000   1st Qu.:19.00
 Median : 3.000   Median :20.00
 Mean   : 3.761   Mean   :24.49
 3rd Qu.: 5.000   3rd Qu.:22.50
 Max.   :16.000   Max.   :52.00
```

A naive estimate of θ_j:

$$\hat{\theta}_j \;=\; y_j/n_j$$

Histogram of $\hat{\theta}_j$s:

```
> with(d, hist(y/N, freq=FALSE, xlim=c(0,1)))
```


Histogram of y/N
Density
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0
y/N

Model

First try this hierarchical model:

$$\begin{aligned} y_j \mid \theta_j &\sim \text{Bin}(n_j, \theta_j) \\ \theta_j \mid \alpha, \beta &\sim \text{Beta}(\alpha, \beta) \\ \alpha, \beta &\sim \text{indep. Expon}(\lambda) \end{aligned}$$

Choosing λ near zero makes hyperprior flatter (more diffuse).

Later: Choosing λ too small can lead to problems.

JAGS model specification in file `rattumor1.bug`:

```
model {

  for (j in 1:length(y)) {
    y[j] ~ dbin(theta[j], N[j])
    theta[j] ~ dbeta(alpha, beta)
  }

  alpha ~ dexp(0.001)
  beta ~ dexp(0.001)

}
```

- Using $\lambda = 0.001$
- JAGS allows use of `length` function.

Using rjags

First install JAGS (Windows, Mac OS X, Linux):

http://mcmc-jags.sourceforge.net

We will use R package rjags to access JAGS within R:

```
> install.packages("rjags")
```

Now make it available to use:

```
> library(rjags)  # automatically loads coda package
Loading required package: coda
Linked to JAGS 4.3.0
Loaded modules: basemod,bugs
```

Now create a JAGS model object from model in file `rattumor1.bug` and data in data frame `d`:

```
> m <- jags.model("rattumor1.bug", d)
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 71
   Unobserved stochastic nodes: 73
   Total graph size: 216

Initializing model

  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100%
```

Object `m` defines an iterative random sampling scheme (more later).

First run it for many iterations until it becomes reliable:

```
> update(m, 2500)  # burn-in
  |**************************************************| 100%
```

Then run it for many more iterations, storing samples from selected nodes:

```
> x <- coda.samples(m, c("alpha","beta"), n.iter=10000)
  |**************************************************| 100%
```

Now `x` is a special object containing posterior samples of α and β.

```
> head(x)
[[1]]
Markov Chain Monte Carlo (MCMC) output:
Start = 3501
End = 3507
Thinning interval = 1
        alpha     beta
[1,] 5.204525 33.83398
[2,] 5.092650 32.32364
[3,] 4.945052 32.34294
[4,] 5.384586 29.77359
[5,] 5.831973 29.91007
[6,] 5.660368 29.43142
[7,] 5.809329 31.59240

attr(,"class")
[1] "mcmc.list"
```

Converting `x` to a matrix makes the variates easier to access:

```
> head(as.matrix(x))
        alpha     beta
[1,] 5.204525 33.83398
[2,] 5.092650 32.32364
[3,] 4.945052 32.34294
[4,] 5.384586 29.77359
[5,] 5.831973 29.91007
[6,] 5.660368 29.43142
```