ADVANCED BAYESIAN MODELING

Rat Growth Data Example: JAGS Analysis

$$y_{ij} \quad = \quad \text{weight of rat } j \text{ at age } x_i \text{ (days)} \qquad i = 1, \ldots, 5 \quad j = 1, \ldots, 30$$

$$x_1 = 8 \qquad x_2 = 15 \qquad x_3 = 22 \qquad x_4 = 29 \qquad x_5 = 36$$

To aid computation, use centered ages:

$$x_i^{\text{cent}} \quad = \quad x_i \; - \; \bar{x}$$

```
> ratgrowth <- read.csv("ratgrowth.csv", header=TRUE)

> head(ratgrowth)
  day8 day15 day22 day29 day36
1  151   199   246   283   320
2  145   199   249   293   354
3  147   214   263   312   328
4  155   200   237   272   297
5  135   188   230   280   323
6  159   210   252   298   331
```

Note: Index j corresponds to rows, index i to columns.

We fit normal-theory model

$$y_{ij} \mid \beta^{(j)}, \sigma_y^2, x_i \quad \sim \quad \text{indep. } \mathrm{N}\big(\beta_1^{(j)} + \beta_2^{(j)} x_i^{\mathsf{cent}},\, \sigma_y^2\big)$$

$$\beta^{(j)} \mid \mu_\beta, \Sigma_\beta \quad \sim \quad \text{iid } \mathrm{N}(\mu_\beta,\, \Sigma_\beta)$$

$$\mu_\beta \;=\; \begin{pmatrix} \mu_{\beta_1} \\ \mu_{\beta_2} \end{pmatrix} \qquad \Sigma_\beta \;=\; \begin{pmatrix} \sigma^2_{\beta_1} & \rho\,\sigma_{\beta_1}\sigma_{\beta_2} \\ \rho\,\sigma_{\beta_1}\sigma_{\beta_2} & \sigma^2_{\beta_2} \end{pmatrix}$$

Parameters of interest include σ_y^2, μ_{β_1}, μ_{β_2}, $\sigma^2_{\beta_1}$, $\sigma^2_{\beta_2}$, and ρ.

Using centered predictors implies that the intercept $\beta_1^{(j)}$ represents predicted weight at the average age ($\bar{x} = 22$).

Simple least squares analyses can guide choice of priors and initialization:

```
> betahat <- matrix(NA, nrow(ratgrowth), 2)
> for(j in 1:nrow(ratgrowth))
+   betahat[j,] <- lsfit(rbind(8, 15, 22, 29, 36) - 22, t(ratgrowth[j,]))$coef

> apply(betahat, 2, mean)
[1] 242.653333   6.185714

> var(betahat)
           [,1]      [,2]
[1,] 210.399816 4.8615764
[2,]   4.861576 0.3344968
```

We need to choose proper priors:

$$\begin{aligned}
\sigma_y^2 &\sim \text{Inv-gamma}(0.0001, 0.0001) \\
\mu_\beta &\sim \text{N}(0, 1000^2 I) \\
\Sigma_\beta^{-1} &\sim \text{Wishart}_2\left(\Sigma_0^{-1}/2\right)
\end{aligned}$$

To ensure scaling is not grossly incorrect, let

$$\Sigma_0 = \begin{pmatrix} 100 & 0 \\ 0 & 0.1 \end{pmatrix}$$

based on preliminary least squares results.

```
data {
  dimY <- dim(weight)
  agecent <- age - mean(age)
}
model {
  for (j in 1:dimY[1]) {
    for (i in 1:dimY[2]) {
      weight[j,i] ~ dnorm(beta[1,j] + beta[2,j]*agecent[i], sigmasqyinv)
    }
    beta[1:2,j] ~ dmnorm(mubeta, Sigmabetainv)
  }

  mubeta ~ dmnorm(mubeta0, Sigmamubetainv)
  Sigmabetainv ~ dwish(2*Sigma0, 2)
  sigmasqyinv ~ dgamma(0.0001, 0.0001)

  Sigmabeta <- inverse(Sigmabetainv)
  rho <- Sigmabeta[1,2] / sqrt(Sigmabeta[1,1] * Sigmabeta[2,2])
  sigmasqy <- 1/sigmasqyinv
}
```

Note:

- Age centering performed in `data` block
- Explicit row range in `beta[1:2,j]` needed for JAGS to determine dimension
- `inverse` computes matrix inverse

Set up data and constants:

```
d1 <- list(weight = ratgrowth,
           age = c(8, 15, 22, 29, 36),
           mubeta0 = c(0, 0),
           Sigmamubetainv = rbind(c(0.000001, 0),
                                  c(0, 0.000001)),
           Sigma0 = rbind(c(100, 0),
                          c(0, 0.1)))
```

Set up initializations (extreme relative to data) for 4 chains:

```
inits1 <- list(list(sigmasqyinv = 10, mubeta = c(1000, 1000),
                    Sigmabetainv = rbind(c(100, 0),
                                         c(0, 100))),
               list(sigmasqyinv = 0.001, mubeta = c(-1000, 1000),
                    Sigmabetainv = rbind(c(100, 0),
                                         c(0, 100))),
               list(sigmasqyinv = 10, mubeta = c(1000, -1000),
                    Sigmabetainv = rbind(c(0.001, 0),
                                         c(0, 0.001))),
               list(sigmasqyinv = 0.001, mubeta = c(-1000, -1000),
                    Sigmabetainv = rbind(c(0.001, 0),
                                         c(0, 0.001))))
```

Note: Only top-level nodes are initialized.

```
> library(rjags)

> m1 <- jags.model("ratgrowth1.bug", d1, inits1, n.chains=4, n.adapt=1000)
...

> update(m1, 1000)  # burn-in
  |**************************************************| 100%

> x1 <- coda.samples(m1, c("mubeta","Sigmabeta","sigmasqy"), n.iter=2000)
  |**************************************************| 100%
```

Note: Monitoring the full matrix `Sigmabeta` (including both off-diagonal elements)

```
> gelman.diag(x1, autoburnin=FALSE, multivariate=FALSE)
Potential scale reduction factors:

               Point est. Upper C.I.
Sigmabeta[1,1]       1.31       2.03
Sigmabeta[2,1]       1.30       1.67
Sigmabeta[1,2]       1.30       1.67
Sigmabeta[2,2]       1.31       2.09
mubeta[1]            2.39      12.94
mubeta[2]           31.95     199.76
sigmasqy             7.92      48.96
```

Clearly requires more burn-in.

Remark: `multivariate=FALSE` because of a singularity issue

Chains eventually appear to converge by 31000 iterations (not shown).

Now compute the iterations to be used for inference:

```
> x1 <- coda.samples(m1, c("beta","mubeta","Sigmabeta","sigmasqy","rho"),
+                    n.iter=2000)
  |**************************************************| 100%

> effectiveSize(x1[,c("mubeta[1]","mubeta[2]","Sigmabeta[1,1]","Sigmabeta[1,2]",
+                     "Sigmabeta[2,2]","sigmasqy","rho")])
     mubeta[1]      mubeta[2] Sigmabeta[1,1] Sigmabeta[1,2] Sigmabeta[2,2]
      7528.876       5263.405       6756.783       4983.577       3174.161
      sigmasqy            rho
      2809.634       3059.025
```

```
> summary(x1[,c("mubeta[1]","mubeta[2]","Sigmabeta[1,1]","Sigmabeta[1,2]",
+               "Sigmabeta[2,2]","sigmasqy","rho")])

Iterations = 31001:33000
Thinning interval = 1
Number of chains = 4
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

                    Mean       SD Naive SE Time-series SE
mubeta[1]       242.6663  2.74148 0.030651       0.031593
mubeta[2]         6.1867  0.10715 0.001198       0.001478
Sigmabeta[1,1] 217.3919 61.70650 0.689900       0.751189
Sigmabeta[1,2]   4.9700  1.96815 0.022005       0.027941
Sigmabeta[2,2]   0.2694  0.09612 0.001075       0.001707
sigmasqy        37.6433  5.87822 0.065720       0.111669
rho              0.6478  0.13426 0.001501       0.002431
```

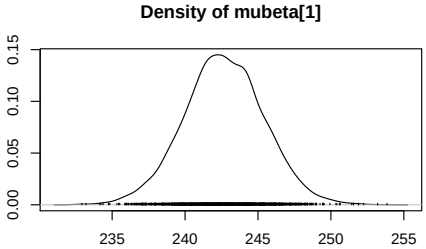
2. Quantiles for each variable:

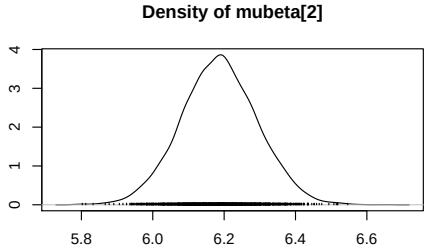
	2.5%	25%	50%	75%	97.5%
mubeta[1]	237.2535	240.8789	242.6421	244.4544	248.0520
mubeta[2]	5.9758	6.1160	6.1865	6.2576	6.3993
Sigmabeta[1,1]	127.9961	173.6677	207.4945	250.1384	365.0139
Sigmabeta[1,2]	1.9920	3.6102	4.6657	6.0064	9.6293
Sigmabeta[2,2]	0.1276	0.2025	0.2550	0.3186	0.5031
sigmasqy	28.0428	33.4059	37.0914	41.1130	50.5589
rho	0.3369	0.5699	0.6658	0.7464	0.8567

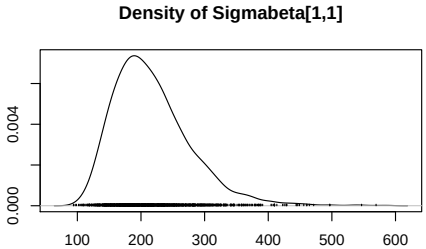
Note: ρ is almost certainly positive.

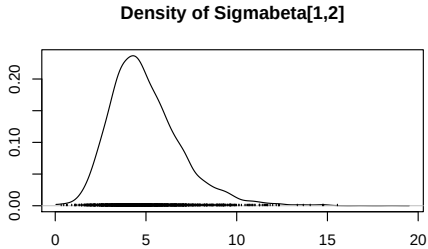
Plot estimates of marginal posterior densities:

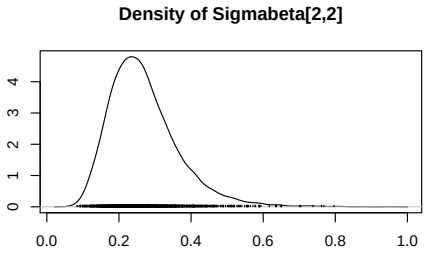
```
> par(mfrow=c(2,2))
> densplot(x1[,c("mubeta[1]","mubeta[2]","Sigmabeta[1,1]","Sigmabeta[1,2]",
+                "Sigmabeta[2,2]","sigmasqy","rho")])
```

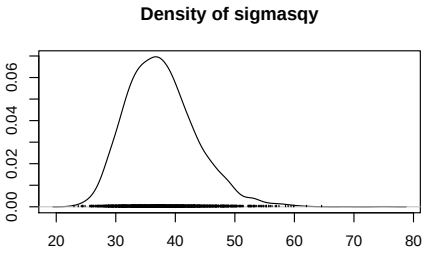

Density of mubeta[1]
0.00
0.05
0.10
0.15
235
240
245
250
255
N = 2000 Bandwidth = 0.4687

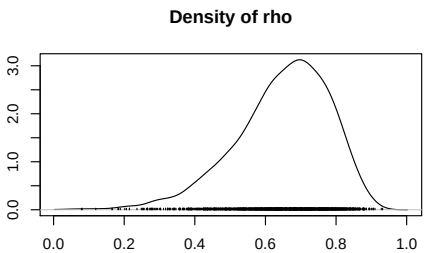

Density of mubeta[2]
0
1
2
3
4
5.8
6.0
6.2
6.4
6.6
N = 2000 Bandwidth = 0.01857


Density of Sigmabeta[1,1]
0.000
0.004
100
200
300
400
500
600
N = 2000 Bandwidth = 10.02


Density of Sigmabeta[1,2]
0.00
0.10
0.20
0
5
10
15
20
N = 2000 Bandwidth = 0.3141


Density of Sigmabeta[2,2]
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
N = 2000 Bandwidth = 0.01521


Density of sigmasqy
0.00
0.02
0.04
0.06
20
30
40
50
60
70
80
N = 2000 Bandwidth = 1.01


Density of rho
0.0
1.0
2.0
3.0
0.0
0.2
0.4
0.6
0.8
1.0
N = 2000 Bandwidth = 0.02315

Compute JAGS version of DIC:

```
> dic.samples(m1,2000)
  |**************************************************| 100%
Mean deviance:  967.7
penalty 51.68
Penalized deviance: 1019
```

Effective number of parameters: about 52

Actual number of parameters: 60 ($\beta^{(j)}$s) + 1 (σ_y^2) + 2 (μ_β) + 3 (Σ_β)

Alternative JAGS model using a scaled inverse Wishart:

```
data {
  dimY <- dim(weight)
  agecent <- age - mean(age)
}
model {
  for (j in 1:dimY[1]) {
    for (i in 1:dimY[2]) {
      weight[j,i] ~ dnorm(beta[1,j] + beta[2,j]*agecent[i], sigmasqyinv)
    }
    beta[1:2,j] <- mubeta + xi * eta[,j]
    eta[1:2,j] ~ dmnorm(zeros2, Sigmaetainv)
  }
  xi[1] ~ dunif(0,1000)
  xi[2] ~ dunif(0,1000)
```

(continued...)

```
  mubeta ~ dmnorm(mubeta0, Sigmamubetainv)
  Sigmaetainv ~ dwish(identity2x2, 3)
  sigmasqyinv ~ dgamma(0.0001, 0.0001)

  Sigmaeta <- inverse(Sigmaetainv)
  sigmasqbeta[1] <- xi[1]^2 * Sigmaeta[1,1]
  sigmasqbeta[2] <- xi[2]^2 * Sigmaeta[2,2]
  rho <- Sigmaeta[1,2] / sqrt(Sigmaeta[1,1] * Sigmaeta[2,2])
  sigmasqy <- 1/sigmasqyinv
}
```

Constants `zeros2` and `identity2x2` are specified with the data.

Remark: Convergence requires more iterations.