ADVANCED BAYESIAN MODELING

EXAMPLE OF PRACTICAL MCMC:
RAT TUMOR DATA: MODEL 2

Consider alternative model specified in JAGS code file `rattumor2.bug`:

```
model {

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

  alpha <- phi1 / phi2^2
  beta <- (1-phi1) / phi2^2

  phi1 ~ dunif(0,1)
  phi2 ~ dunif(0,1000)

}
```

(Also analyzed earlier)

Nodes `alpha` and `beta` are deterministic, so JAGS does not allow them to be initialized.

Instead initialize `phi1` and `phi2`, which are

$$\phi_1 \;=\; \frac{\alpha}{\alpha+\beta} \;\in\; (0,1) \qquad\qquad \phi_2 \;=\; (\alpha+\beta)^{-1/2} \;\in\; (0,\infty)$$

Helpful preliminary values can be obtained using the preliminary α and β values computed previously ...

```
> ( phi1hat <- alphahat / (alphahat + betahat) )
[1] 0.1381151

> ( phi2hat <- (alphahat + betahat)^(-1/2) )
[1] 0.3170556
```

Again, let's use 4 chains, initialized in a manner overdispersed relative to the preliminary values:

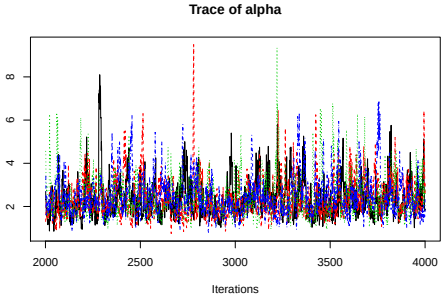
	Initial ϕ_1	Initial ϕ_2
Chain 1:	0.001	0.003
Chain 2:	0.9	0.003
Chain 3:	0.001	30
Chain 4:	0.9	30

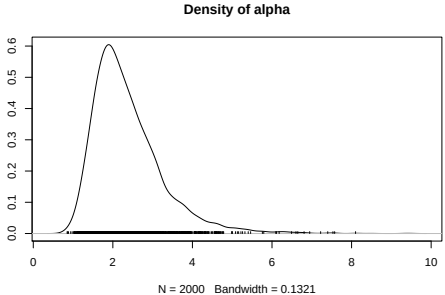
Set up JAGS model in R with 4 chains initialized, and perform 1000 iterations of adaptation:

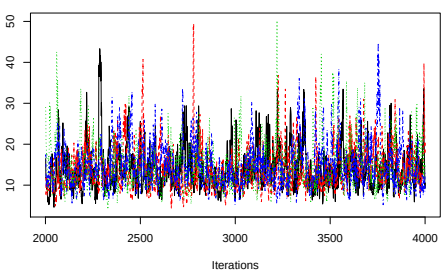
```
> library(rjags)
...

> initial.vals <- list(list(phi1=0.001, phi2=0.003),
+                      list(phi1=0.9,   phi2=0.003),
+                      list(phi1=0.001, phi2=30),
+                      list(phi1=0.9,   phi2=30))

> m2 <- jags.model("rattumor2.bug", d, initial.vals, n.chains=4, n.adapt=1000)
...
```

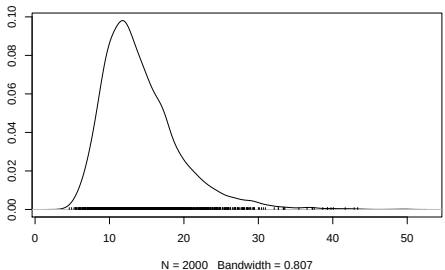
Perform 1000 iterations of burn-in, collect α and β for 2000 more iterations, and plot results:

```
> update(m2, 1000)  # burn-in
  |**************************************************| 100%

> x2 <- coda.samples(m2, c("alpha","beta"), n.iter=2000)
  |**************************************************| 100%

> plot(x2, smooth=FALSE)
```


Trace of alpha
2
4
6
8
2000
2500
3000
3500
4000
Iterations

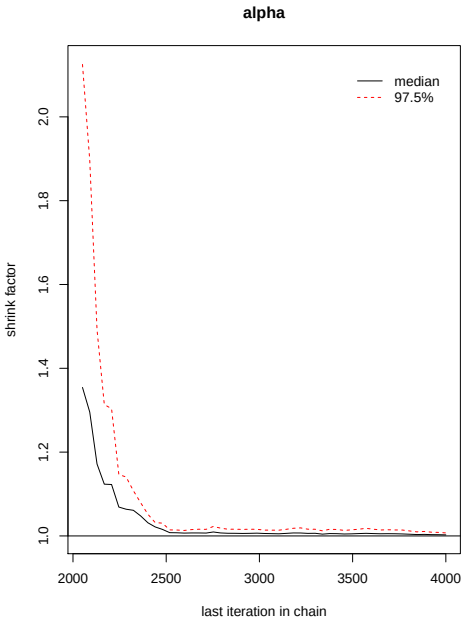

Density of alpha
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
2
4
6
8
10
N = 2000 Bandwidth = 0.1321

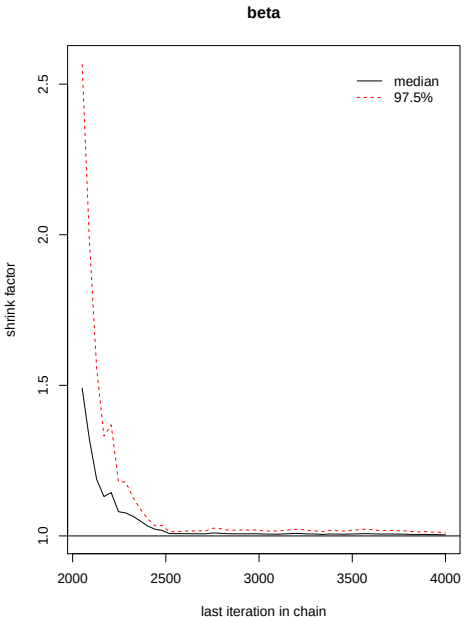

Trace of beta
10
20
30
40
50
2000
2500
3000
3500
4000
Iterations


Density of beta
0.00
0.02
0.04
0.06
0.08
0.10
0
10
20
30
40
50
N = 2000 Bandwidth = 0.807

Trace plots look good. Check a version of the Gelman-Rubin statistic, also called a **potential scale reduction factor**:

```
> gelman.diag(x2, autoburnin=FALSE)
Potential scale reduction factors:

      Point est. Upper C.I.
alpha          1       1.01
beta           1       1.01

Multivariate psrf

1

> gelman.plot(x2, autoburnin=FALSE)
```

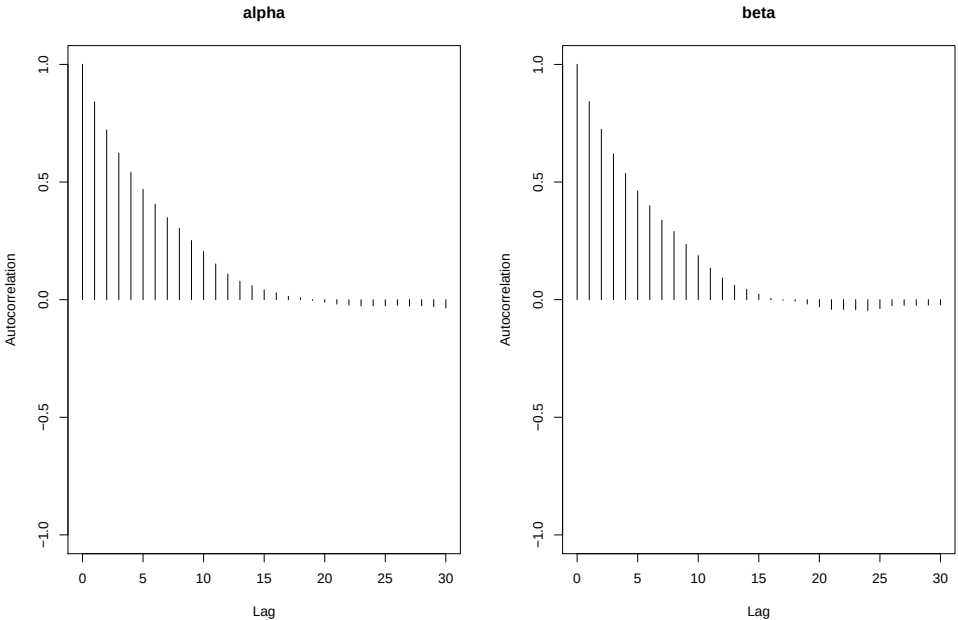

alpha
median
97.5%
shrink factor
1.0
1.2
1.4
1.6
1.8
2.0
2000
2500
3000
3500
4000
last iteration in chain


beta
median
97.5%
shrink factor
1.0
1.5
2.0
2.5
2000
2500
3000
3500
4000
last iteration in chain

Gelman-Rubin statistics appear legitimately near 1 (not just there by chance).

Check autocorrelation plot (pooled over all chains):

```
> autocorr.plot(x2)
```


alpha
beta
Autocorrelation
Lag
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30

Some high autocorrelations, but essentially zero by lag 15.

Check effective sample sizes (summed over all chains):

```
> effectiveSize(x2)
   alpha     beta
708.8611 723.9213
```

Both exceed the suggested minimum value of 400.

These diagnostics all suggest adequate convergence and sampling, but could have run further diagnostics on other parameters (θs) had they been saved.