ADVANCED BAYESIAN MODELING

Flint Data Example: JAGS Analysis

```
> flint <- read.csv("Flintdata.csv", header=TRUE, row.names=1)

> head(flint)
  SampleID ZipCode Ward FirstDraw After45Sec After2Min Notes
1        1   48504    6     0.344      0.226     0.145
2        2   48507    9     8.133     10.770     2.761
3        4   48504    1     1.111      0.110     0.123
4        5   48507    8     8.007      7.446     3.384
5        6   48505    3     1.951      0.048     0.035
6        7   48507    9     7.200      1.400     0.200
```

In file `flint1.bug`:

```
data {
  dimY <- dim(loglead)
}
model {
  for (i in 1:dimY[1]) {
    for (j in 1:dimY[2]) {
      loglead[i,j] ~ dnorm(betadraw[j] + betahouse[i], sigmasqyinv)
    }
    betahouse[i] ~ dnorm(0, 1/sigmahouse^2)
  }

  betadraw ~ dmnorm(betadraw0, Sigmabetadrawinv)
  sigmahouse ~ dunif(0, 1000)
  sigmasqyinv ~ dgamma(0.0001, 0.0001)

  sigmasqhouse <- sigmahouse^2
  sigmasqy <- 1/sigmasqyinv
}
```

Notes:

- The JAGS `data` block is meant for computations that should take place prior to running the Markov chains.

  In this case, it pre-computes dimensions of a data array.

- `for` structures can be nested.

- `betadraw0` and `Sigmabetadrawinv` need to be specified with the data.

```
d1 <- list(loglead = log(flint[,c("FirstDraw","After45Sec","After2Min")]),
           betadraw0 = c(0,0,0),
           Sigmabetadrawinv = rbind(c(0.0001, 0, 0),
                                    c(0, 0.0001, 0),
                                    c(0, 0, 0.0001)))
```

Note: `loglead` can be treated as if it is a 271×3 array.

Set up initializations for 4 chains:

```
inits1 <- list(list(betadraw=c(100, 100, 100),
                    sigmasqyinv=0.0001, sigmahouse=100),
               list(betadraw=c(-100, -100, 100),
                    sigmasqyinv=1000, sigmahouse=0.01),
               list(betadraw=c(-100, 100, -100),
                    sigmasqyinv=0.0001, sigmahouse=0.01),
               list(betadraw=c(100, -100, -100),
                    sigmasqyinv=1000, sigmahouse=100))
```

These values are meant to be overdispersed (relative to the ranges of values we would expect in the posterior).

`betahouse` is not a top-level parameter, so we will let it be auto-initialized.

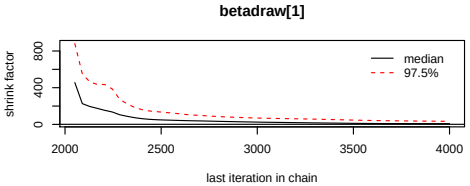
```
> library(rjags)
...

> m1 <- jags.model("flint1.bug", d1, inits1, n.chains=4, n.adapt=1000)
...

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

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

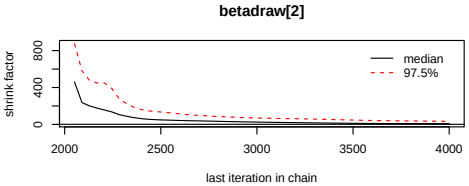
```
> gelman.diag(x1, autoburnin=FALSE)
Potential scale reduction factors:

             Point est. Upper C.I.
betadraw[1]        7.89       33.7
betadraw[2]        7.89       33.7
betadraw[3]        7.89       33.7
sigmasqhouse       7.50       24.7
sigmasqy           1.00        1.0

Multivariate psrf

9.52
```

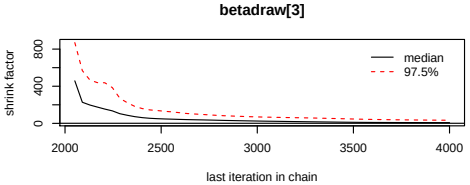
Convergence problems are strongly apparent.

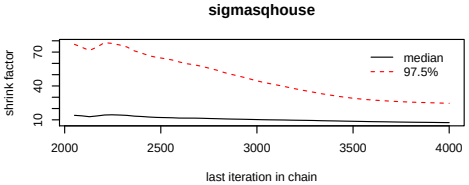
We can check whether the chains at least appear to approach convergence:

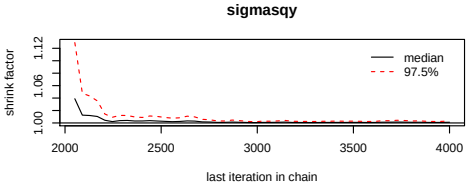
```
> gelman.plot(x1, autoburnin=FALSE)
```


betadraw[1]
shrink factor
0
400
800
median
97.5%
2000
2500
3000
3500
4000
last iteration in chain


betadraw[2]
shrink factor
0
400
800
median
97.5%
2000
2500
3000
3500
4000
last iteration in chain

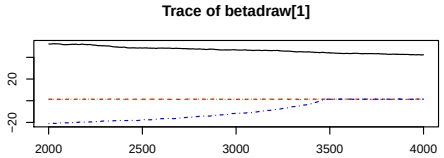

betadraw[3]
shrink factor
0
400
800
median
97.5%
2000
2500
3000
3500
4000
last iteration in chain

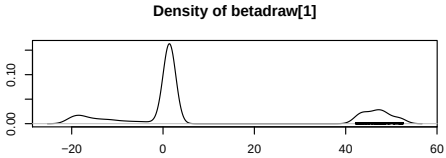

sigmasqhouse
shrink factor
10
40
70
median
97.5%
2000
2500
3000
3500
4000
last iteration in chain

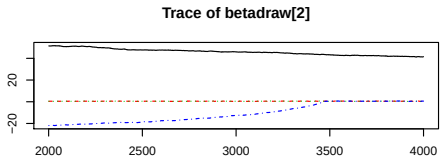

sigmasqy
shrink factor
1.00
1.06
1.12
median
97.5%
2000
2500
3000
3500
4000
last iteration in chain

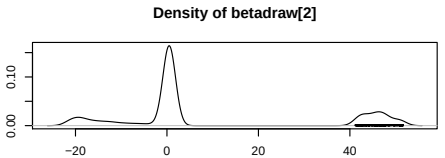
Examine trace plots for more information:

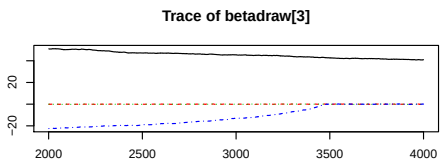
```
> plot(x1, smooth=FALSE)
```

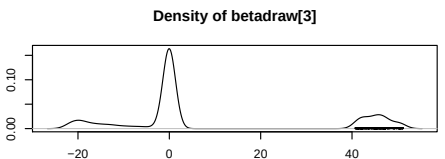

Trace of betadraw[1]
20
−20
2000
2500
3000
3500
4000
Iterations

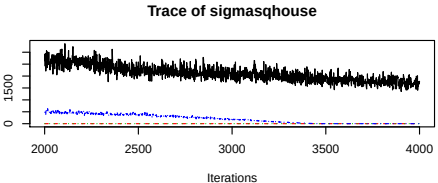

Density of betadraw[1]
0.00
0.10
−20
0
20
40
60
N = 2000 Bandwidth = 1.383

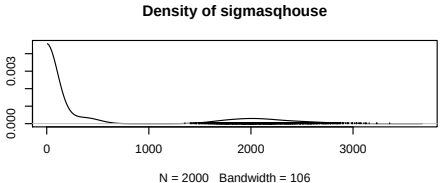

Trace of betadraw[2]
20
−20
2000
2500
3000
3500
4000
Iterations

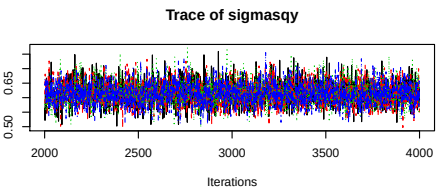

Density of betadraw[2]
0.00
0.10
−20
0
20
40
N = 2000 Bandwidth = 1.374

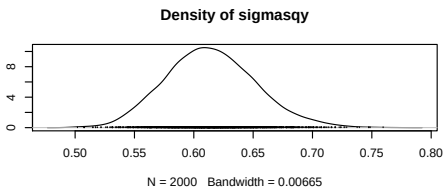

Trace of betadraw[3]
20
−20
2000
2500
3000
3500
4000
Iterations


Density of betadraw[3]
0.00
0.10
−20
0
20
40
N = 2000 Bandwidth = 1.379


Trace of sigmasqhouse
0
1500
2000
2500
3000
3500
4000
Iterations


Density of sigmasqhouse
0.000
0.003
0
1000
2000
3000
N = 2000 Bandwidth = 106


Trace of sigmasqy
0.50
0.65
2000
2500
3000
3500
4000
Iterations


Density of sigmasqy
0
4
8
0.50
0.55
0.60
0.65
0.70
0.75
0.80
N = 2000 Bandwidth = 0.00665

Convergence may eventually occur, but much more burn-in is needed.

Suggestion: Continue to simulate from the chains, doubling the number of iterations each time before checking convergence. Stop when convergence criteria are met.

So we simulate an additional 4000 (still not converged), then an additional 8000 (still not converged), then an additional 16000 (finally converged, according to Gelman-Rubin statistics and plots).

Now let's discard the iterations from before as burn-in, and run additional iterations to be used for inference. Let's also start to monitor `betahouse`:

```
> x1 <- coda.samples(m1, c("betadraw","betahouse","sigmasqy","sigmasqhouse"),
+                    n.iter=2000)
  |************************************************| 100%

> effectiveSize(x1[,c("betadraw[1]","betadraw[2]","betadraw[3]",
+                     "sigmasqy","sigmasqhouse")])
 betadraw[1]  betadraw[2]  betadraw[3]     sigmasqy sigmasqhouse
    641.3857     640.2259     664.5990    3996.9173    3486.6495
```

Effective sample sizes seem adequate.

```
> summary(x1[,c("betadraw[1]","betadraw[2]","betadraw[3]", "sigmasqy",
+               "sigmasqhouse")])

Iterations = 32001:34000
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
betadraw[1]   1.40353 0.09174 0.0010257      0.0036519
betadraw[2]   0.49578 0.09133 0.0010211      0.0036878
betadraw[3]  -0.02447 0.09095 0.0010169      0.0036063
sigmasqy      0.61426 0.03743 0.0004185      0.0005923
sigmasqhouse  1.55114 0.15164 0.0016954      0.0062496
```

2. Quantiles for each variable:

	2.5%	25%	50%	75%	97.5%
betadraw[1]	1.2260	1.34226	1.40168	1.46472	1.5888
betadraw[2]	0.3161	0.43643	0.49462	0.55712	0.6764
betadraw[3]	-0.2045	-0.08372	-0.02643	0.03617	0.1559
sigmasqy	0.5456	0.58811	0.61291	0.63879	0.6907
sigmasqhouse	1.2764	1.44456	1.54623	1.65046	1.8633

Now combine the separate samples into a single matrix:

```
> post.samp <- as.matrix(x1)
```

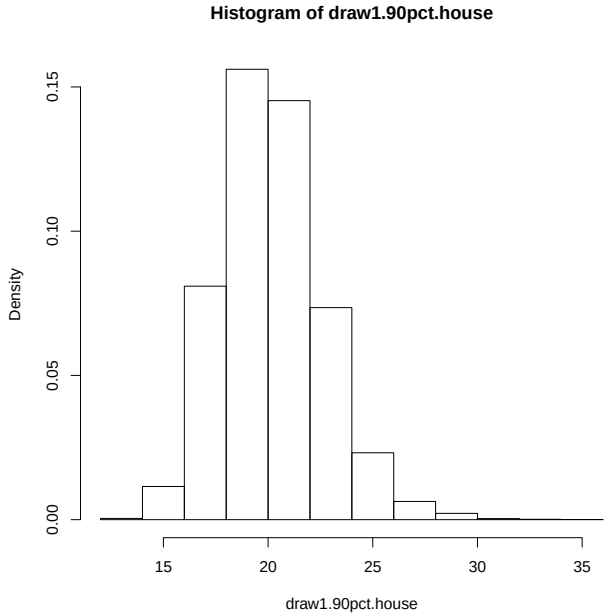
For a newly-sampled household, the "true" first-draw log lead level is $\mathrm{N}(\beta_1^d, \sigma_h^2)$.

The 90th percentile of this distribution is the 90th percentile of all household log lead levels.

We create a sample from its distribution (and transform back to the original scale):

```
> draw1.90pct.house <- exp(qnorm(0.9, post.samp[,"betadraw[1]"],
+                                     sqrt(post.samp[,"sigmasqhouse"])))

> hist(draw1.90pct.house, freq=FALSE)
```

We see it is highly probable that the 90th percentile of "true" first-draw lead levels exceeds 15 ppb ...


Histogram of draw1.90pct.house
Density
0.00
0.05
0.10
0.15
15
20
25
30
35
draw1.90pct.house

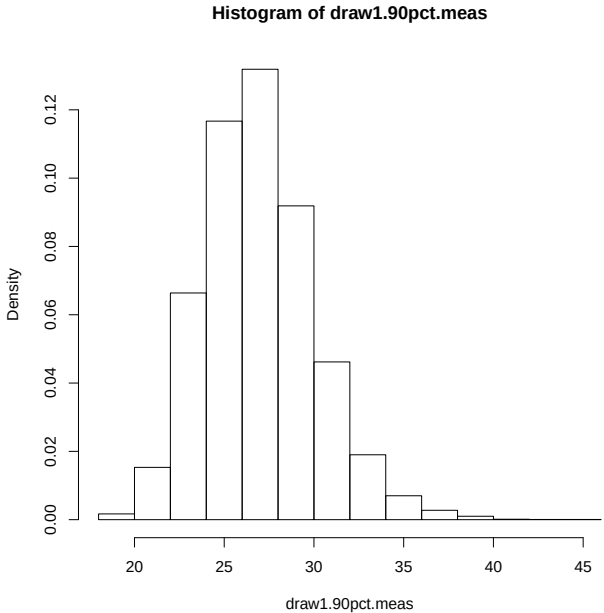
What if we include error in measurement, i.e., consider the measured log lead level of first-draw sample from the random household?

Its distribution will be $\mathrm{N}(\beta_1^d, \sigma_h^2 + \sigma_y^2)$.

Now sample from the 90th percentile of its distribution (and transform back to the original scale):

```
> draw1.90pct.meas <- exp(qnorm(0.9, post.samp[,"betadraw[1]"],
+                                    sqrt(post.samp[,"sigmasqhouse"] +
+                                         post.samp[,"sigmasqy"])))

> hist(draw1.90pct.meas, freq=FALSE)
```

We see it is even more highly probable that the 90th percentile of measured first-draw lead levels exceeds 15 ppb ...


Histogram of draw1.90pct.meas
Density
0.00
0.02
0.04
0.06
0.08
0.10
0.12
20
25
30
35
40
45
draw1.90pct.meas

The Lead and Copper Rule requires that the 90th percentile of measured first-draw lead levels not exceed 15 ppb.

This indicates that, if a large enough resample of households were taken, the results would very likely require further action.

(The second and third draws have lower 90th percentiles – results not shown.)

Do mean lead levels decrease as flushing time increases?

```
> mean(post.samp[,"betadraw[1]"] > post.samp[,"betadraw[2]"])
[1] 1

> mean(post.samp[,"betadraw[2]"] > post.samp[,"betadraw[3]"])
[1] 1
```

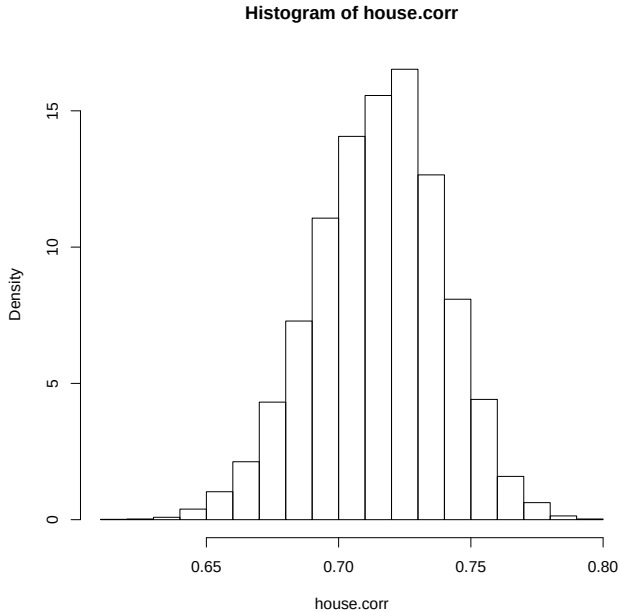
Answer is highly likely yes, both from 0 s to 45 s flushing, and from 45 s to 120 s flushing.

What about the correlation

$$\frac{\sigma_h^2}{\sigma_h^2 + \sigma_y^2}$$

between measurements from the same household?

```
> house.corr <- post.samp[,"sigmasqhouse"] /
+               (post.samp[,"sigmasqhouse"] + post.samp[,"sigmasqy"])

> hist(house.corr, freq=FALSE)  # within-household correlation
```


Histogram of house.corr
Density
0
5
10
15
0.65
0.70
0.75
0.80
house.corr

Could perform posterior predictive checks of assumptions, such as

- Distribution of household effects β_i^h is normal.
  (Result: Possible evidence of right skew.)
- Sampling variance σ_y^2 does not depend on household.
  (Result: Insufficient evidence that it does.)