ADVANCED BAYESIAN MODELING

2016 Polls Data: Model Checking

Data Set

In file `polls2016.txt`:

```
# 2016 U.S. presidential election race between H. Clinton and D. Trump
# National poll results for two-way race, conducted November 3 and later
# y = percentage of Clinton lead, with margin of error

poll        y   ME
YouGov      4   1.7
Bloomberg   3   3.5
ABCWaPo     3   2.5
Fox         4   2.5
IBD         1   3.1
Monmouth    6   3.6
NBCWSJ      5   2.73
```

We read in the data and create a variable for `sigma` separately:

```
> d <- read.table("polls2016.txt", header=TRUE)

> d$sigma <- d$ME/2  # standard dev = half margin of error
```

JAGS Analysis

We use an approximating JAGS model (`polls2016ppc.bug`) having node array `yrep` for replicate data sets:

```
model {

  for (j in 1:length(y)) {
    y[j] ~ dnorm(theta[j], 1/sigma[j]^2)
    theta[j] ~ dnorm(mu, 1/tau^2)

    yrep[j] ~ dnorm(theta[j], 1/sigma[j]^2)
  }

  mu ~ dunif(-1000,1000)
  tau ~ dunif(0,1000)

}
```

Remark: JAGS will recognize that `yrep` is not linked to any observed values, and therefore simulate it separately (rather than within the Gibbs sampler).

```
> library(rjags)
...

> m <- jags.model("polls2016ppc.bug", d)
...

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

> x <- coda.samples(m, c("mu","tau","theta","yrep"), n.iter=10000)
  |**************************************************| 100%
```

(Check convergence for yourself.)

For convenience, we make matrices of posterior simulated θ and y^{rep}:

```
> theta <- as.matrix(x)[, paste("theta[",1:nrow(d),"]", sep="")]

> yrep <- as.matrix(x)[, paste("yrep[",1:nrow(d),"]", sep="")]
```

For example, consider posterior predictive p-values for test statistics $\max_j y_j$, $\min_j y_j$, the average of the y_j, and the sample standard deviation of the y_j:

```
> mean(apply(yrep, 1, max) >= max(d$y))
[1] 0.4762

> mean(apply(yrep, 1, min) >= min(d$y))
[1] 0.6514

> mean(apply(yrep, 1, mean) >= mean(d$y))
[1] 0.5126

> mean(apply(yrep, 1, sd) >= sd(d$y))
[1] 0.4832
```

No evidence of problems

One type of test quantity has the *chi-square* form

$$\sum_{j=1}^{J} \frac{(y_j - \mu_j)^2}{\sigma_j^2}$$

where μ_j and σ_j^2 are supposed to be the mean and variance of y_j.

If the means are mis-specified, the value tends to be larger. If the variances are mis-specified, the value could be larger or smaller (but is often larger).

Classically, this is converted to a *chi-square statistic* by replacing μ_j and σ_j^2 by estimates or null values (if necessary).

In Bayesian posterior predictive analysis, it need not be a statistic ...

For the 2016 polls model, consider

$$T(y, \theta) \;=\; \sum_{j=1}^{J} \frac{(y_j - \theta_j)^2}{\sigma_j^2}$$

Tends to be larger than it should if the θ_js have a prior that is too concentrated (underdispersed).

Might also be larger if some y_j is an outlier.

Calculate

$$\Pr\Big(T(y^{\mathsf{rep}}, \theta) \;\geq\; T(y, \theta) \mid y\Big)$$

```
> Tchi <- numeric(nrow(yrep))
> Tchirep <- numeric(nrow(yrep))
> for(s in 1:nrow(yrep)){
+   Tchi[s] <- sum((d$y - theta[s,])^2 / d$sigma^2)
+   Tchirep[s] <- sum((yrep[s,] - theta[s,])^2 / d$sigma^2)
+ }

> mean(Tchirep >= Tchi)
[1] 0.5084
```

No evidence of problems

Overall, we found:

- No evidence against normality of θ_js

  (but more specific tools exist – see later)

- No evidence that hyperpriors unduly constrain or bias μ and τ