Robust Bayesian Regression & Gibbs Sampling

October 6, 2008

Readings: GIII 4

Latent Variable Model

$$Y_i \mid \alpha, \beta, \phi, \lambda \overset{ind}{\sim} N(\alpha + \beta x_i, \frac{1}{\phi \lambda_i})$$

$$\lambda_i \overset{iid}{\sim} G(\nu/2, \nu/2)$$

$$p(\alpha, \beta, \phi) \propto 1/\phi$$

Joint Posterior Distribution:

$$p(\alpha, \beta, \phi, \lambda_1, \ldots, \lambda_n \mid Y) \propto \phi^{n/2-1} \exp\left\{ -\frac{\phi}{2} \sum \lambda_i (y_i - \alpha - \beta x_i)^2 \right\} \times \prod_{i=1}^{n} \lambda_i^{1/2} \prod_{i=1}^{n} \lambda_i^{\nu/2-1} \exp(-\lambda_i \frac{\nu}{2})$$

Single Component Gibbs Sampler

Start with $(\alpha^{(0)}, \beta^{(0)}, \phi^{(0)}, \lambda_1^{(0)}, \ldots, \lambda_n^{(0)})$
For $t = 1, \ldots, T$, generate from the following sequence of Full Conditional distributions:

- $p(\alpha \mid \beta^{(t-1)}, \phi^{(t-1)}, \lambda_1^{(t-1)}, \ldots, \lambda_n^{(t-1)}, Y)$
- $p(\beta \mid \alpha^{(t)}, \phi^{(t-1)}, \lambda_1^{(t-1)}, \ldots, \lambda_n^{(t-1)}, Y)$
- $p(\phi \mid \alpha^{(t)}, \beta^{(t)}, \phi^{(t-1)}, \lambda_1^{(t-1)}, \ldots, \lambda_n^{(t-1)}, Y)$
- $p(\lambda_j \mid \alpha^{(t)}, \beta^{(t)}, \phi^{(t)}, \lambda_{(-j)}^{(t-1)}, Y)$ for $j = 1, \ldots, n$

$\lambda_{(-j)}$ is the vector of λs excluding the jth component

Easy to find and sample!

Programs

BUGS: Bayesian inference Using Gibbs Sampling

- WinBUGS is the Windows implementation
  - can be called from R with `R2WinBUGS` package
  - can be run on any intel-based computer using VMware, wine
- OpenBUGS open source version of WinBUGS
- LinBUGS is the Linux implementation of OpenBUGS.
- JAGS: Just Another Gibbs Sampler is an alternative program that uses the same model description as BUGS (Linux, MAC OS X, Windows)

Include more than just Gibbs Sampling

BUGS

Need to specify

- Model
- Data
- Initial values

May do this through ordinary text files or use the functions in `R2WinBUGS` to specify model, data, and initial values then call WinBUGS.

Model Specification via R2WinBUGS

```
rr.model = function() {
  for (i in 1:n) {
    mu[i] <- alpha0 + alpha1*(X[i] - Xbar)
    lambda[i] ~ dgamma(4.5, 4.5)
    prec[i] <- phi*lambda[i]
    Y[i] ~ dnorm(mu[i], prec[i])
  }
  phi ~ dgamma(1.0E-6, 1.0E-6)
  alpha0 ~ dnorm(0, 1.0E-6)
  alpha1 ~ dnorm(0,1.0E-6)
}
```

Notes on Models

- Distributions of stochastic "nodes" are specified using $\sim$
- Assignment of deterministic "nodes" uses `<-` (NOT $=$)
- Cannot put expressions as arguments in distributions
- Normal distributions are parameterized using precisions, so `dnorm(0, 1.0E-6)` is a $N(0, 1.0 \times 10^6)$
- uses `for` loop structure as in R

Initial Values

Function to calculate initial values for parameters as a list

```
rr.inits = function() {
    bf.lm = lm(bf.data$Y ~ bf.data$X)
    coefs = coef(bf.lm)
    alpha1=coefs[2]
    alpha0 = coefs[1] - alpha1*bf.data$Xbar
    phi = (1/summary(bf.lm)$sigma)^2
   lambda = rep(1, bf.data$n)
return(list(alpha0=alpha0, alpha1 = alpha1,
            phi=phi, lambda=lambda))
}
```

Data

A list or rectangular data structure for all data and summaries of data used in the model

```
bf.data = list(Y = bodyfat$Bodyfat,
               X=bodyfat$Abdomen)
bf.data$n = length(bf.data$Y)
bf.data$Xbar = mean(bf.data$X)
```

Specifying which Parameters to Save

The parameters to be monitored and returned to R are specified with the variable `parameters`

```
parameters = c("beta0", "beta1", "sigma",
               "mu34", "y34", "lambda[39]")
```

- All of the above (except lambda) are calculated from the other parameters. (See R-code for definitions of these parameters.)
- `lambda[39]` saves only the 39th case of λ
- To save a whole vector (for example all lambdas, just give the vector name)

Running WinBUGS from R

Write the model out as a text file, then call `bugs()`

```
write.model(rr.model, "rr-model.txt")
model.file = "rr-model.txt"

bf.sim = bugs(bf.data, rr.inits, parameters,
              model.file,
              n.chains=2, n.iter=5000,
              debug=T, DIC=F)
```

`debug=T` keeps WinBUGS open – very useful for debugging BUGS!

Other arguments necessary for running under Linux or MAC OSX using wine.

Output

	mean	sd	2.5%	50%	97.5%
beta0	-41.70	2.75	-46.91	-41.67	-36.40
beta1	0.66	0.03	0.60	0.66	0.71
sigma	4.48	0.23	4.05	4.46	4.96
mu34	15.10	0.35	14.43	15.10	15.82
y34	14.94	5.15	4.37	15.21	24.65
lambda[39]	0.33	0.16	0.11	0.30	0.72

95% HPD interval for expected bodyfat $(14.5, 15.8)$
95% HPD interval for bodyfat $(5.1, 25.3)$

Comparison

- 95% Probability Interval for β is $(0.60, 0.71)$ with t_9 errors
- 95% Confidence Interval for β is $(0.58, 0.69)$ (all data normal model)
- 95% Confidence Interval for β is $(0.61, 0.73)$ (normal model without case 39)

Results intermediate without having to remove any observations

Case 39 down weighted by λ_{39}

Full Conditional for λ_j

$$p(\lambda_j \mid \text{rest}, Y) \propto p(\alpha, \beta, \phi, \lambda_1, \ldots, \lambda_n \mid Y)$$

$$\propto \phi^{n/2-1} \prod_{i=1}^{n} \exp\left\{-\frac{\phi}{2}\lambda_i(y_i - \alpha - \beta x_i)^2\right\} \times$$

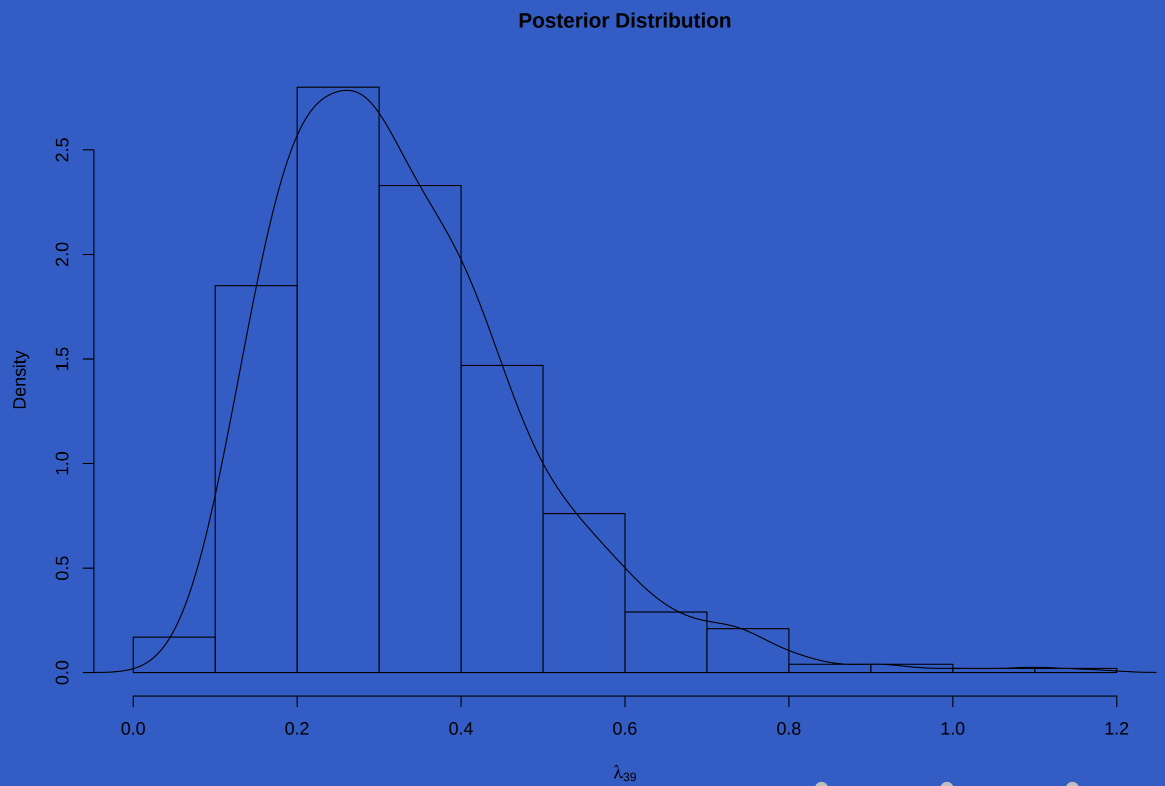
$$\prod_{i=1}^{n} \lambda_i^{\frac{\nu+1}{2}-1} \exp(-\lambda_i \frac{\nu}{2})$$

Ignore all terms except those that involve λ_j

$$\lambda_j \mid \text{rest}, Y \sim G(\frac{\nu+1}{2}, \frac{\phi(y_j - \alpha - \beta x_j)^2 + \nu}{2}$$

Weights

Under prior $E[\lambda_i] = 1$
Under posterior, large residuals are down-weighted (approximately those bigger than $\sqrt{\nu}$)



Full Conditional for ϕ

Full Conditional for α

Full Conditional for β