Bayes, BUGs and Beyond

Chapter 16, ARM

We have used `lmer` successfully to fit multilevel models.
Issue: uses point estimate of variance (if 0, drop the random effect).
A strategy:

- Start with `lm` or `glm`.
- Fit in `lmer` using multiple levels.
- Use BUGs or JAGs
- More R programming

Bayes Paradigm

Each parameter has a prior distribution. They use:

- Gaussian for group-level models or
- Uniform distributions (noninformative)

and caution (p348) that we should check robustness to prior specification.
Combine prior with likelihood to get a posterior which answers any question.

Regression

We've seen that classical regression estimates are posterior means given a $U(-\infty, \infty)$ prior on each component of β and $U(0, \infty)$ on $\log(\sigma)$.
Interpretation is different:

$$\beta|\mathbf{y}, \sigma \sim N(\widehat{\beta}, \sigma^2(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1})$$

versus

$$\widehat{\beta}|\sigma \sim N(\beta, \sigma^2(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1})$$

Varying Intercept:
$y_i \sim N(\alpha_{j[i]} + \beta x_i, \sigma_y^2)$ with prior $\alpha_j \sim N(\mu_\alpha, \sigma_\alpha^2)$

More Complexity

$$\mathbf{y}_i = \mathbf{X}_i\beta + \mathbf{Z}_i\mathbf{b}_i + \epsilon_i$$

Same uniform prior on β and $\log(\sigma)$.
$\mathbf{b}_i \sim N(\mathbf{0}, \mathbf{\Psi})$

Fit Radon data in R

```
srrs2 <- read.table("http://www.stat.columbia.edu/~gelman/arm/examples/radon/sr
    head = TRUE, sep = ",")
mn <- droplevels(subset(srrs2, state == "MN"))
names(mn)[19] <- "radon"
mn$log.radon <- y <- log(pmax(0.1, mn$radon))
n <- nrow(mn)
levels(mn$county) <- gsub("[[:space:]]$", "", levels(mn$county))
ybarbar <- mean(y)
sample.size <- as.vector(table(mn$county))
n.county <- length(sample.size)
sample.size.jittered <- sample.size * exp(runif(n.county, -0.1, 0.1))
cty.mns <- tapply(y, mn$county, mean)
cty.vars <- tapply(y, mn$county, var)
cty.sds <- mean(sqrt(cty.vars[!is.na(cty.vars)]))/sqrt(sample.size)
cty.sds.sep <- sqrt(cty.vars/sample.size)
```


lm fit

```
lm.pooled <- lm(log.radon ~ floor, data = mn)
display.xtable(lm.pooled)
```

	Estimate	Std. Error	t value
(Intercept)	1.33	0.03	44.64
floor	-0.61	0.07	-8.42

Table: n = 919 rank = 2 resid sd = 0.823 R-Squared = 0.072

```
lm.unpooled.1 <- lm(log.radon ~ 0 + floor + county, data =
display.xtable(lm.unpooled.1)
```

	Estimate	Std. Error	t value
floor	-0.72	0.07	-9.80
countyAITKIN	0.84	0.38	2.22
countyANOKA	0.87	0.10	8.33

JAGs setup

Build a model file containing:

```
model {
  for (i in 1:n){
    y[i] ~ dnorm (y.hat[i], tau.y)
    y.hat[i] <- a[county[i]] + b*x[i]
  }
  b ~ dnorm (0, .0001)
  tau.y <- pow(sigma.y, -2)
  sigma.y ~ dunif (0, 100)
  for (j in 1:J){
    a[j] ~ dnorm (mu.a, tau.a)
  }
  mu.a ~ dnorm (0, .0001)
  tau.a <- pow(sigma.a, -2)
  sigma.a ~ dunif (0, 100)
}
```

Same code works in BUGs or JAGs

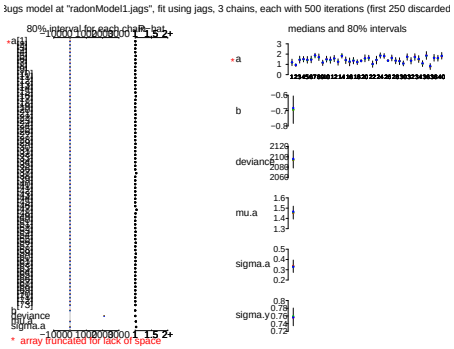
JAGs call

```
require(R2jags)
radon.data <- with(mn, list(n = n, J = n.county, y = log.radon, county = as.num
    x = floor))
radon.JAGs <- jags(model.file = "radonModel1.jags", data = radon.data,
    parameters.to.save = c("a", "b", "mu.a", "sigma.y", "sigma.a"), n.chains =
    n.iter = 500)

## module glm loaded
```

```
attach.jags(radon.JAGs)
ls(pos = 2)

## [1] "a"         "b"         "deviance" "mu.a"     "n.sims"   "sigma.a"
## [7] "sigma.y"

summary(gelman.diag(as.mcmc(radon.JAGs)))

##       Length Class  Mode
## psrf  180    -none- numeric
## mpsrf   1    -none- numeric
```

JAGs Plot

```
plot(radon.JAGs)
```



Setup Plots of Random Effects

```
display8 <- c(36, 1, 35, 21, 14, 71, 61, 70)  # counties to be displayed
x.jitter <- mn$floor + runif(n, -0.05, 0.05)
x.range <- range(x.jitter)
y.range <- range(mn$log.radon[!is.na(match(as.numeric(mn$county), display8))])

# pull out parameter estimates from classical fits
a.pooled <- coef(lm.pooled)[1]  # complete-pooling intercept
b.pooled <- coef(lm.pooled)[2]  # complete-pooling slope
a.nopooled <- coef(lm.unpooled.1)[2:(n.county + 1)]  # no-pooling vector of int
b.nopooled <- coef(lm.unpooled.1)[1]  # no-pooling slope
a.multilevel <- apply(a, 2, median)
b.multilevel <- median(b)
cnty <- as.numeric(mn$county)
```


Plots of Random Effects

```
par(mfrow = c(2, 4))
for (j in display8) {
    plot(x.jitter[cnty == j], mn$log.radon[cnty == j], xlim = c(-0.05,
        1.05), ylim = y.range, xlab = "floor", ylab = "log radon level",
        main = levels(mn$county)[j], cex.lab = 1.2, cex.axis = 1.1, cex.main =
        mgp = c(2, 0.7, 0), pch = 20)
    curve(a.pooled + b.pooled * x, lwd = 0.5, lty = 2, col = "gray10",
        add = TRUE)
    curve(a.nopooled[j] + b.nopooled * x, lwd = 0.5, col = "gray10", add = TRUE
    curve(a.multilevel[j] + b.multilevel * x, lwd = 1, col = "black", add = TRU
}
```

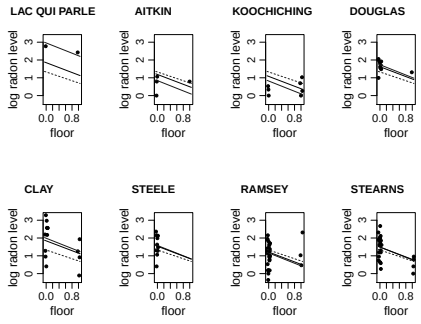
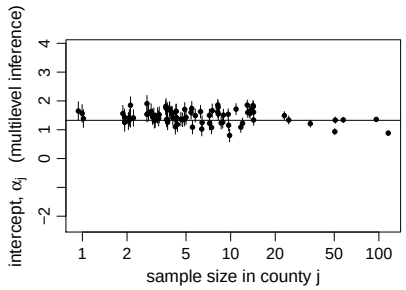


Figure 12.3b

```
sample.size <- as.vector(table(mn$county))
sample.size.jitter <- sample.size * exp(runif(85, -0.1, 0.1))
plot(sample.size.jitter, a.multilevel, xlab = "sample size in county j",
    ylim = range(mn$log.radon), ylab = expression(paste("intercept, ",
        alpha[j], "   (multilevel inference)")), cex.lab = 1.2, cex.axis = 1.1,
    cex.main = 1.1, mgp = c(2, 0.7, 0), pch = 20, log = "x")
for (j in 1:85) {
    lines(rep(sample.size.jitter[j], 2), median(a[, j]) + c(-1, 1) * sd(a[,
        j]))
}
abline(a.pooled, 0, lwd = 0.5)
```

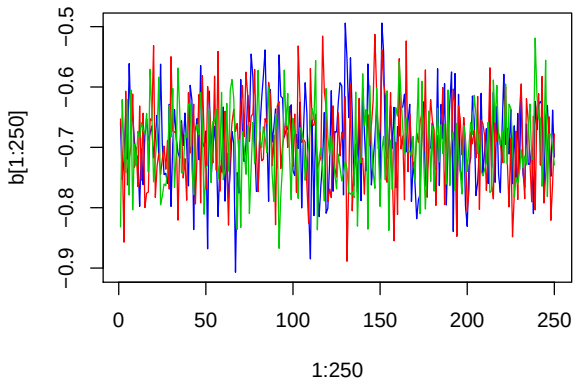


Convergence of MCMC runs

- Check to see that Gelman & Rubin's $\hat{R}$ is close to 1. $\hat{R} < 1.1$
- Check effective sample size $\approx n \times (1 - |\hat{\rho}|)$ where n is the number of samples saved and ρ is the autocorrelation of residuals. Correlation near 1 reduces effective sample size, near 0 has no effect. Should be over 100.
- Trace plots of multiple chains. See that they are sampling from the same general values of each parameter, not stuck in different areas.

Traceplot on Slope

```
plot(1:250, b[1:250], type = "l", col = 4)  ## chain 1 in 1:250
lines(1:250, b[1:250 + 250], type = "l", col = 2)  ## chain 2 251:500
lines(1:250, b[1:250 + 500], type = "l", col = 3)  ## chain 3 501:750
```

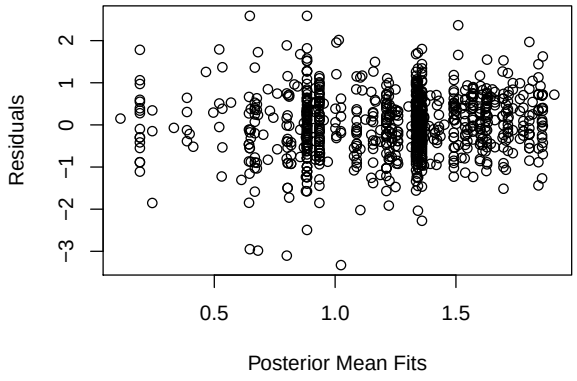


Three chains should "cover" the same space.

Fits and Residuals

Fitted values are based on posterior means:

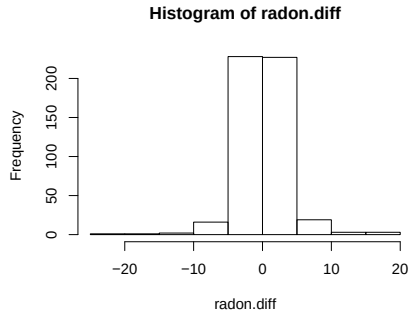
```
yHat <- a.multilevel[cnty] + b.multilevel * mn$floor  ## or monitor it
yResid <- mn$log.radon - yHat
plot(yHat, yResid, xlab = "Posterior Mean Fits", ylab = "Residuals")
```



More Posterior Effects

Compare unlogged radon for random house in Hennepin county (26) to random house in Lac Qui Parle county (36) at floor = 0.

```
lqp.radon <- exp(rnorm(500, a[, 26] + b, sigma.y))
hen.radon <- exp(rnorm(500, a[, 26] + b, sigma.y))
hist(radon.diff)
```




```
xtable(summary(radon.diff <- lqp.radon - hen.radon))
```

16.5 Add County Level Predictors

- Complete Pooling: one α, one β, Priors: $N(0, 100^2)$

```
model {
  for (i in 1:n){
    y[i] ~ dnorm (a + b*x[i], tau.y)
  }
  b ~ dnorm (0, .0001)
  tau.y <- pow(sigma.y, -2)
  sigma.y ~ dunif (0, 100)
  a ~ dnorm (0, .0001)
}
```

- No Pooling: One β as above, 84 α_j's, each $N(0, 100^2)$

```
model {
  for (i in 1:n){
    y[i] ~ dnorm (a[cnty[i]] + b*x[i], tau.y)
  }
  b ~ dnorm (0, .0001)
  tau.y <- pow(sigma.y, -2)
  sigma.y ~ dunif (0, 100)
  for(j in 1:J){
    a[j] ~ dnorm (mu.a, tau.a)
  }
  sigma.a ~ dunif (0, 100)
  tau.a <- pow(sigma.a, -2)
  mu.a ~ dnorm(0, .0001)
}
```


Add Predictors

Winter is associated with higher radon levels

```
y.hat[i] <- a + b_floor * floor[i] + b_winter * winter[i]
```

Interaction?

```
y.hat[i] <- a + b_floor * floor[i] + b_winter * winter[i] + b_fw * floor[i] *
    winter[i]
```

Each gets a `dnorm(0,.0001)` prior. Now pass matrix `X` with first column all ones

```
y.hat[i] <- inprod(b[], X[i, ])
```


County level Uranium fit

JAGs model

```
model {
  for (i in 1:n){
    y[i] ~ dnorm (a[cnty[i]] + b*x[i], tau.y)
  }
  b ~ dnorm (0, .0001)
  tau.y <- pow(sigma.y, -2)
  sigma.y ~ dunif (0, 100)
  for(j in 1:J){
    a[j] ~ dnorm (a.hat[j], tau.a)
    a.hat[j] <- g.o + g.1*u[j]
  }
  sigma.a ~ dunif (0, 100)
  tau.a <- pow(sigma.a, -2)
  g.0 ~ dnorm(0, .0001)
  g.1 ~ dnorm(0, .0001)
}
```


16.6 Predictions

Either in BUGs/JAGs
add more lines for missing responses, and a `y.tilde` flag.
or in R
attach the saved MCMC samples, generate data using each row of the simulation.

```
attach.jags(radon2.JAGs)
y.hepin <- rnorm(n.sims, a[, 26] + b * 1, sigma.y)  ## first floor
u.tilde <- mean(mn$u)
a.tilde <- rnorm(n.sims, g.0 + g.1 * u.tilde, sigma.a)
y.newcty <- rnorm(n.sims, a.tilde + b * 1, sigma.y)
```

16.7 Does it Work?

Simulate to Check, as in Chapter 8

1. Pick reasonable parameter values, θ^{true}.
   - From expert opinion
   - From preliminary fitted results.
2. Create y^{fake} using θ^{true} and the assumed model.
3. Fit the model, see if estimates are close to θ^{true}. Coverage rates for credible intervals.

16.8 Principles

Data, modeled responses or unmodeled: predictors or sizes

modeled: `y` used with $\sim$

unmodeled: `n, J, cnty, floor, u` used on RHS of <- or $\sim$

Parameters, modeled or not, are on LHS of $\sim$

modeled: `a` fit to uranium

unmodeled: `b, g.0, g.1, sigma.y, sigma.a`

Derived: `y.hat, tau.y, a.hat, tau.a` defined with <-

indices: `i, j`

Parameters and missing data can be given initial values.

16.9 Implementation

- Start simple, get it working, add complexity slowly.
- Use 3-4 chains
- Start with a few runs. `autojags` lets you update with more runs. Check $\hat{R} < 1.1$.
- Better to simplify or subset the data than to runn for thousands of iterations.

16.10 Open-ended

Models can get very complex.
Nonlinear is no harder to handle. `b * exp( -g * x[i])`
Ratio: `y[i] <- (a + b*x1[i])/(1 + g*x2[i])`
Unequal variances by group or as power of the mean.
T distribution instead of normal, even estimating the df.