Understanding and Summarizing the Fitted Models

JEFF GILL
Distinguished Professor
Departments of Government and Mathematics & Statistics
American University

Motivation

- We have seen ways to specify and fit various multilevel models, both linear and non-linear, and in two powerful computer languages.
- Summarizing can be a challenge since there are often many parameters to describe in their posterior sense and some may not be of primary interest.
- The number of parameters can also be a challenge in communicating with readers/reviewers/editors.
- Sometimes these are called "nuisance parameters," but they can matter.
- G&H advice: use more graphics.

Uncertainty and Variability

- Uncertainty: incomplete knowledge about unknown parameters.
- As sample size goes to infinity, uncertainty disappears.
- Variability: underlying differences between groups or individuals.
- Variability always exists even in asympotia.
- Start in R with the following command for the rest of the semester:

```
library("R2WinBUGS","lme4","rjags","R2jags","arm","coda","superdiag")
```

Back to Radon (of course), But Now With JAGS

- jags data file:

```
N <- 919
J <- 85
y <- c(0.78845736036427, 0.78845736036427, 1.06471073699243,.....)
county <- c(1, 1, 1, 1, 2, 2,.....)
x <- c(1, 0, 0, 0, 0, 0,.....)
```

- As opposed to bugs:

```
list(y=c(0.78845736036427, 0.78845736036427, 1.06471073699243,.....),
     county=c(1, 1, 1, 1, 2, 2,.....),
     x=c(1, 0, 0, 0, 0, 0,.....))
```

- Use the function:

```
bugs2jags("Class.Multilevel/examples/radon/radon.x.only.bugs.dat",
          "Class.Multilevel/examples/radon/radon.x.only.jags.dat")
```

to translate a bugs to jags data file.

Distinguishing Between Uncertainty and Variability in a Multilevel Model

- Recall from the radon example, the varying-intercept model:

$$y_i = N(\alpha_{j[i]} + \beta x_i, \sigma_y^2).$$

whose output is given on G&H page 351 from `radon.1.bug`.

- `BUGS` code:

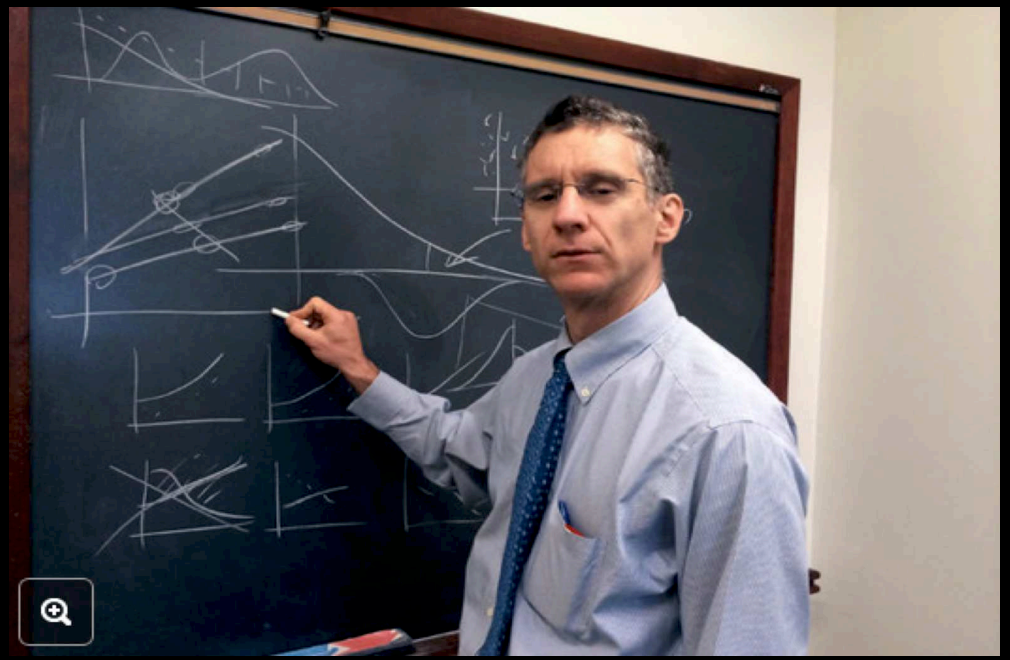
```
model {
  for (i in 1:N)  {
    y[i] ~ dnorm(y.hat[i], tau.y)
    y.hat[i] <- a[county[i]] + b*x[i]
  }
  b ~ dnorm(0, 0.0001)
  tau.y <- pow(sigma.y, -2)
  sigma.y ~ dunif(0,100)
  for (j in 1:J)  { a[j] ~ dnorm(mu.a, tau.a) }
  mu.a ~ dnorm(0, 0.0001)
  tau.a <- pow(sigma.a, -2)
  sigma.a ~ dunif(0, 100)
}
```

Comment on G&H Priors

- The book makes extensive use of this form of priors for variance coefficients:

```
tau.y <- pow(sigma.y, -2)
sigma.y   dunif(0,100)
```

- But this is a lot of density away from the high density region.
- And why uniform for a variance component to a power?
- Why have a default prior across all datasets in the second half of the book?



"Jeff, why don't you like my priors?"

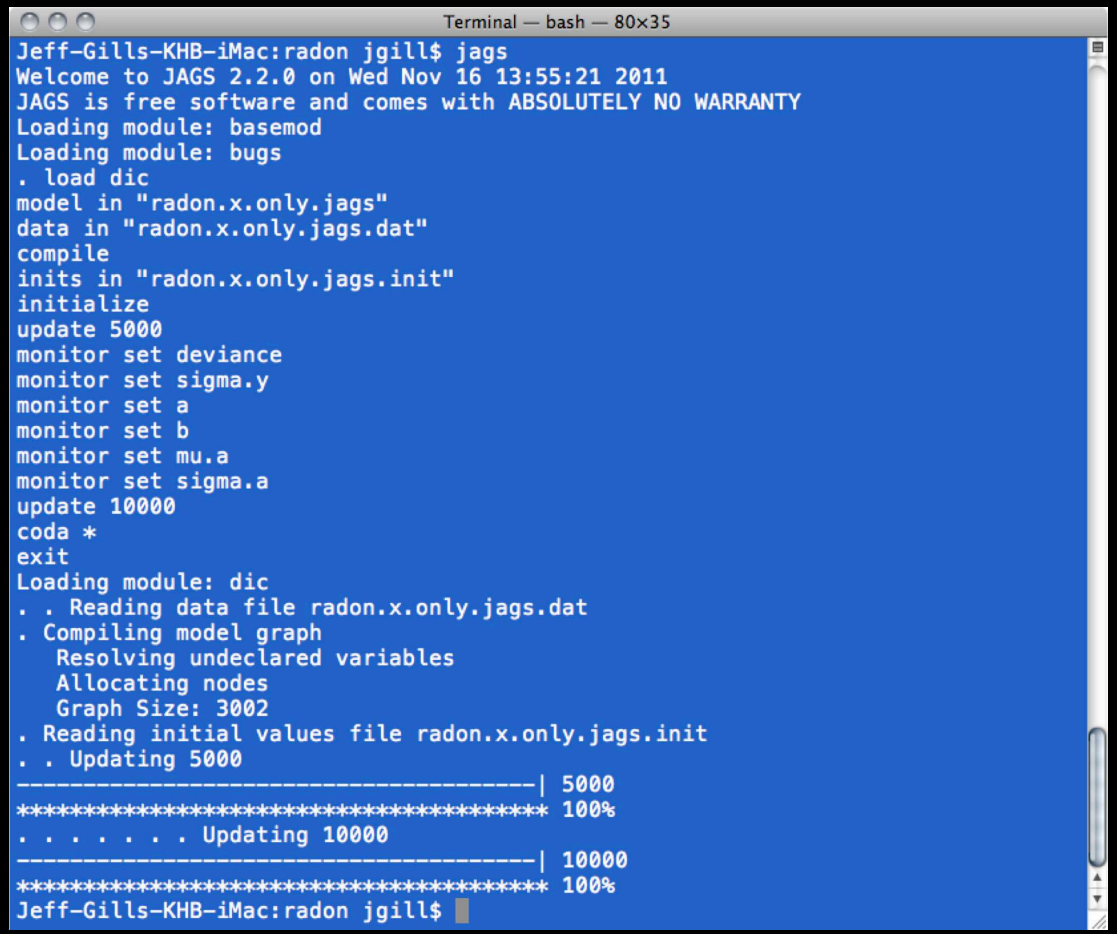
Initial Values in jags Form

```
sigma.y <- 50
a <- c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
       0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
       0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
b <- 0
mu.a <- 0
sigma.a <- 50
```

jags Commands

```
load dic
model in "radon.x.only.jags"
data in "radon.x.only.jags.dat"
compile, nchains(3)
parameters in radon1.inits # or inits in ...
parameters in radon2.inits
parameters in radon3.inits
initialize
update 5000
monitor set deviance
monitor set sigma.y
monitor set a
monitor set b
monitor set mu.a
monitor set sigma.a
update 10000
coda *
exit
```

JAGS Window


Terminal — bash — 80×35
Jeff-Gills-KHB-iMac:radon jgill$ jags
Welcome to JAGS 2.2.0 on Wed Nov 16 13:55:21 2011
JAGS is free software and comes with ABSOLUTELY NO WARRANTY
Loading module: basemod
Loading module: bugs
. load dic
model in "radon.x.only.jags"
data in "radon.x.only.jags.dat"
compile
inits in "radon.x.only.jags.init"
initialize
update 5000
monitor set deviance
monitor set sigma.y
monitor set a
monitor set b
monitor set mu.a
monitor set sigma.a
update 10000
coda *
exit
Loading module: dic
. . Reading data file radon.x.only.jags.dat
. Compiling model graph
Resolving undeclared variables
Allocating nodes
Graph Size: 3002
. Reading initial values file radon.x.only.jags.init
. . Updating 5000
--| 5000
** 100%
. . . . . Updating 10000
--| 10000
** 100%
Jeff-Gills-KHB-iMac:radon jgill$

CODA Loading Procedure

- Running through the CODA menus can get tiresome, so you can also use the underlying functions:

```
radon.x.only <- read.coda("Class.Multilevel/examples/radon/CODAchain1.txt",
                          "Class.Multilevel/examples/radon/CODAindex.txt")
Abstracting deviance ... 10000 valid values
Abstracting sigma.y ... 10000 valid values
Abstracting a[1] ... 10000 valid values
:
Abstracting a[85] ... 10000 valid values
Abstracting b ... 10000 valid values
Abstracting mu.a ... 10000 valid values
Abstracting sigma.a ... 10000 valid values

radon.x.only2 <- read.coda("Class.Multilevel/examples/radon/CODAchain2.txt",
                           "Class.Multilevel/examples/radon/CODAindex.txt")
radon.x.only3 <- read.coda("Class.Multilevel/examples/radon/CODAchain3.txt",
                           "Class.Multilevel/examples/radon/CODAindex.txt")
```

Convergence Analysis

```
radon.chain1 <- mcmc(radon.x.only)
radon.chain2 <- mcmc(radon.x.only2)
radon.chain3 <- mcmc(radon.x.only3)
radon.list <- as.mcmc.list(list(radon.chain1,radon.chain2,radon.chain3))
superdiag(radon.list,burnin=5000)

Number of chains = 3
Number of iterations = 10000 per chain before discarding the burn-in period
The burn-in period = 5000 per chain
Sample size in total = 15000

********** The Geweke diagnostic: **********
Z-scores:
                    chain1    chain 2     chain 3
deviance         0.39492203  0.2467381  0.60959145
sigma.y          0.44305842 -0.5349743  1.30338177
a[1]             2.08308553 -0.3863379 -0.02477471
a[2]             0.89312153 -0.7267228  1.06168888
a[3]             1.53253452 -0.6069151 -0.30797923
```

Convergence Analysis

```
********** The Gelman-Rubin diagnostic: **********
Potential scale reduction factors:

         Point est. Upper C.I.
deviance          1       1.00
sigma.y           1       1.00
a[1]              1       1.00
a[2]              1       1.00
a[3]              1       1.00
:
b                 1       1.00
mu.a              1       1.00
sigma.a           1       1.01

Multivariate psrf
1.01
```

Convergence Analysis

```
********** The Heidelberger-Welch diagnostic: **********

Chain 1, epsilon=0.1, alpha=0.05
         Stationarity start     p-value
         test         iteration
deviance passed           1     0.8450
sigma.y  passed           1     0.5657
a[1]     passed           1     0.0776
:
b        passed           1     0.7880
mu.a     passed           1     0.2062
sigma.a  passed           1     0.8664

         Halfwidth Mean     Halfwidth
         test
deviance passed    2094.498 0.624093
sigma.y  passed       0.757 0.000734
a[1]     passed       1.188 0.006895
:
```

Convergence Analysis

```
********** The Raftery-Lewis diagnostic: **********

Chain 1, converge.eps = 0.001
Quantile (q) = 0.025
Accuracy (r) = +/- 0.005
Probability (s) = 0.95

          Burn-in  Total Lower bound  Dependence
          (M)      (N)   (Nmin)       factor (I)
 deviance 3        4267  3746         1.140
 sigma.y  5        5771  3746         1.540
 a[1]     2        3803  3746         1.020
 :
 b        2        3866  3746         1.030
 mu.a     4        5299  3746         1.410
 sigma.a  14       15086 3746         4.030

You need a sample size of at least 243499 with these values of q, r and s
```

Look At Empirical Summary

```
radon.mean <- round(apply(radon.x.only[5001:10000,],2,mean),3)
radon.se <- round(apply(radon.x.only[5001:10000,],2,sd),3)
cbind(radon.mean,radon.se)
```

```
          radon.mean radon.se
deviance    2094.795   12.824
sigma.y        0.757    0.018
a[1]           1.192    0.252
a[2]           0.928    0.102
:
a[84]          1.590    0.179
a[85]          1.392    0.290
b             -0.694    0.069
mu.a           1.464    0.051
sigma.a        0.334    0.047
```

Distinguishing Between Uncertainty and Variability in a Multilevel Model

- So the unexplained variation of radon levels *within* a county is given by:

```
sigma.y        0.757    0.018
```

(assumed now to be the same for all counties).

- This means that `sigma.y` gives the variability among houses within a county, and `sigma.a` gives the variabilty between counties in total:

```
sigma.a        0.334    0.047
```

- But the explained variation *of* each county-level estimate is given by the:

```
a[1]           1.192    0.252
a[2]           0.928    0.102
:
a[84]          1.590    0.179
a[85]          1.392    0.290
```

Distinguishing Between Uncertainty and Variability in a Multilevel Model

- As we get more data *within* existing counties, uncertainty about the α_j declines *for that county*.
- If we were to get more counties (eg. add a state or something), uncertainty about aggregate group-level quantities, μ_α and σ_α, declines.
- Note that this is different than overall increases in data that affect σ_y.
- So unlike simple, non-hierarchical, models which "N" is increasing is important.
- As data size N for total cases increases, the group differences will not change unless these new samples are disproportionately applied to groups.

Radon Model With Uranium

```
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 (a.hat[j], tau.a)
    a.hat[j] <- g.0 + g.1*u[j]
  }
  g.0 ~ dnorm (0, .0001)
  g.1 ~ dnorm (0, .0001)
  tau.a <- pow(sigma.a, -2)
  sigma.a ~ dunif (0, 100)
}
```

jags Commands Adding Uranium

```
load dic
model in "radon.all.preds.jags"
data in "radon.all.preds.jags.dat"
compile
inits in "radon.all.preds.jags.init"
initialize
update 5000
monitor set deviance
monitor set sigma.y
monitor set a
monitor set b
monitor set mu.a
monitor set sigma.a
monitor set g.0
monitor set g.1
update 10000
coda *
exit
```

Bring In To R

```
radon.all.preds <- read.coda("Class.Multilevel/examples/radon/CODAchain1.txt",
                             "Class.Multilevel/examples/radon/CODAindex.txt")

Abstracting deviance ... 10000 valid values
Abstracting sigma.y ... 10000 valid values
Abstracting a[1] ... 10000 valid values
Abstracting a[2] ... 10000 valid values
:
Abstracting a[84] ... 10000 valid values
Abstracting a[85] ... 10000 valid values
Abstracting b ... 10000 valid values
Abstracting mu.a ... 10000 valid values
Abstracting sigma.a ... 10000 valid values
Abstracting g.0 ... 10000 valid values
Abstracting g.1 ... 10000 valid values
```

Look At Summaries

```
radon.mean <- round(apply(radon.all.preds[5001:10000,],2,mean),3)
radon.se <- round(apply(radon.all.preds[5001:10000,],2,sd),3)
cbind(radon.mean,radon.se)
```

```
deviance   2101.609   12.169
sigma.y       0.760    0.018
a[1]          0.942    0.160
a[2]          0.863    0.090
:
a[84]         1.483    0.132
a[85]         1.679    0.166
b            -0.670    0.069
sigma.a       0.159    0.049
g.0           1.466    0.039
g.1           0.727    0.097
```

Model That Allows the Variance to Vary Within Groups

```
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.sq.y ~ dunif (0, 100)

  for (j in 1:J){
    a[j] ~ dnorm (a.hat[j], tau.a[j]) # NOTE THE j ON tau.a HERE
    a.hat[j] <- g.0 + g.1*u[j]
    tau.a[j] ~ dgamma(1,0.001)
    sigma.sq.a[j] <- 1/tau.a[j]
  }
  g.0 ~ dnorm (0, .0001)
  g.1 ~ dnorm (0, .0001)
}
```

Running JAGS For a Distribution of Variances in y

```
load dic
model in "radon.var.y.jags"
data in "radon.all.preds.jags.dat"
compile
inits in "radon.var.y.jags.init"
initialize
update 5000
monitor set deviance
monitor set sigma.sq.y
monitor set a
monitor set b
monitor set mu.a
monitor set sigma.sq.a
monitor set g.0
monitor set g.1
update 10000
coda *
exit
```

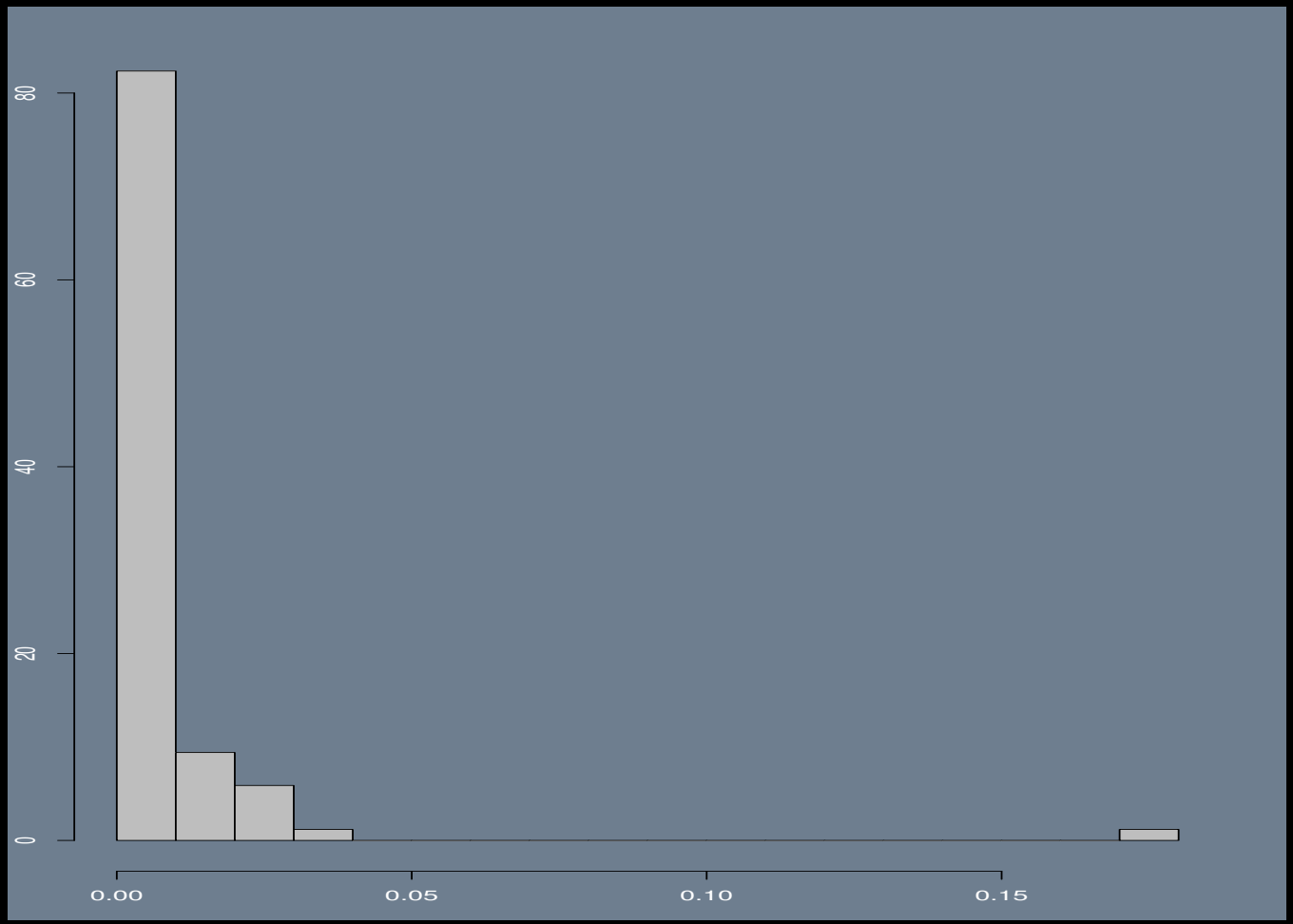
Looking at Variable Variance

```
radon.var.y <- read.coda("Class.Multilevel/examples/radon/CODAchain1.txt",
                         "Class.Multilevel/examples/radon/CODAindex.txt")

radon.mean <- round(apply(radon.var.y[5001:10000,],2,mean),3)
radon.se <- round(apply(radon.var.y[5001:10000,],2,sd),3)
cbind(radon.mean,radon.se)

postscript("Class.Multilevel/Images/var.y.hist.ps")
par(mfrow=c(1,1),mar=c(3,2,2,2),col.axis="white",col.lab="white",col.sub="white",
    col="white",bg="slategrey")
hist(radon.mean[89:173],breaks=20,col="gray",axes=TRUE,xlab="",ylab="",main="",
     freq=FALSE)
dev.off()
```

Variability of County-Level Standard Errors



A Varying-Intercept, Varying-Slope Model

- CD4 counts in a sample of children:

$$y_i \sim N(\alpha_{j[i]} + \beta_{j[i]} t_i, \sigma_y^2)$$

- With R code:

```
libary("foreign","lme4","rjags","R2jags","arm","coda","superdiag")

# READ IN THE DATA FROM AN EXCEL-FORMAT ".csv" COMMA SEPARATED FILE
hiv.data <- read.csv("https://jeffgill.org/wp-content/uploads/2021/04/allvar.csv.txt",
    colClasses=c("numeric","numeric","character",rep("numeric",6)),
    header=TRUE)
attach.all(hiv.data)                    # ATTACH A BUGS OBJECT

# JUST CONSIDER THE "control" PATIENTS (treatmnt==1) AND WITH
# INITIAL AGE BETWEEN 1 AND 5 YEARS, AND CASEWISE DELETED
ok <- treatmnt==1 & !is.na(CD4PCT) & (baseage>1 & baseage<5)
attach.all(hiv.data[ok,])
```

A Varying-Intercept, Varying-Slope Model

```
# SET UP SOME DATA STRUCTURES (GELMAN-STYLE)

y <- sqrt(CD4PCT)
age.baseline <- baseage          # kid's age (yrs) at the beginning of the study
age.measurement <- visage        # kids age (yrs) at the time of measurement
treatment <- treatmnt
time <- visage - baseage

# SET UP UNIQUE PATIENT ID NUMBERS FROM 1 TO J

unique.pid <- unique(newpid)
n <- length(y)
J <- length(unique.pid)
person <- rep(NA, n)
for (j in 1:J){ person[newpid==unique.pid[j]] <- j }
```

A Varying-Intercept, Varying-Slope Model

```
# DEALING WITH MISSING PERSON 26

sum(is.na(time))    [1] 3
sum(is.na(person))  [1] 3
cbind(person,time)[127:135,]

      person      time
[1,]     25 0.4708333
[2,]     25 0.7008333
[3,]     25 0.9308333
[4,]     NA        NA
[5,]     NA        NA
[6,]     NA        NA
[7,]     27 0.0000000
[8,]     27 0.2300000
[9,]     27 0.5175000

person <- person[-c(130,131,132)]; time <- time[-c(130,131,132)]
y <- y[-c(130,131,132)]
```

A Varying-Intercept, Varying-Slope Model

```
# FIT MULTILEVEL MODEL USING LMER
M1 <- lmer(y ~ time + (1 + time | person),na.action="na.omit")
summary(M1)
Linear mixed model fit by REML
Formula: y ~ time + (1 + time | person)
 AIC  BIC logLik deviance REMLdev
 1108 1132   -548     1092    1096
Random effects:
 Groups   Name        Variance Std.Dev. Corr
 person   (Intercept) 1.766    1.329
          time        0.462    0.680    0.149
 Residual             0.560    0.748
Number of obs: 369, groups: person, 83

Fixed effects:                          Correlation of Fixed Effects:
            Estimate Std. Error t value       (Intr)
(Intercept)    4.846      0.160   30.35     time -0.146
time          -0.468      0.128   -3.67
```

Interpreting These Results

- The mean of the α_j is 4.846, and the mean of the β_j is -0.468.
- So for the typical person: $\bar{y} = 4.846 - 0.468t$.
- `time -0.468 0.128 -3.67` tells us that the declining CD4 trend over time is statistically reliable (*uncertainty of the mean*).
- Conversely, `time 0.462 0.680` tells us that there is conserable variation among the j's (people) since it is on a comparable scale (*variation in the population*).

Running the Model in JAGS (which G&H do not do)

```
library("rjags","R2jags","arm","coda","superdiag","R2WinBUGS")

cd4.list <- list("y"=y[-c(130:132)],"time"=time[-c(130:132)],
    "person"=person[-c(130:132)], "COUNTS"=length(time)-3, "KIDS"=length(y)-3)

cd4.rjags  <- function()  {
    for (i in 1:COUNTS) {
        mu[i] <- alpha[person[i]] + beta[person[i]]*time[i]
        y[i] ~ dnorm(mu[i],tau.y)
    }
    for (j in 1:KIDS)  {
        alpha[j] ~ dnorm(0,tau.alpha)
        beta[j] ~ dnorm(0,tau.beta)
    }
    tau.alpha ~ dgamma(1,0.1)
    tau.beta ~ dgamma(1,0.1)
    tau.y ~ dgamma(1,0.1)
}
```

Notes On Initial Values

- If no initial (starting) values are supplied, then they will be generated automatically by `JAGS` .
- However, you do not then control overdispersion for the Gelman & Rubin diagnostic.
- One way to do this mechanically is to give a list of lists:

```
cd4.inits <- list(
    "inits1" <- list("alpha"=runif(366,-10,10),"beta"=runif(366,-10,10),
                     "tau.alpha"=runif(1,0,5),"tau.beta"=runif(1,0,5),
                     "tau.y"=runif(1,0,5)),
    "inits2" <- list("alpha"=runif(366,-10,10),"beta"=runif(366,-10,10),
                     "tau.alpha"=runif(1,0,5),"tau.beta"=runif(1,0,5),
                     "tau.y"=runif(1,0,5)),
    "inits3" <- list("alpha"=runif(366,-10,10),"beta"=runif(366,-10,10),
                     "tau.alpha"=runif(1,0,5),"tau.beta"=runif(1,0,5),
                     "tau.y"=runif(1,0,5))
)
```

- This is a good example for why users pick random initial values (with big dispersion) since there are a lot of points to determine: $366 + 366 + 3$.

Notes on Randomness

- Initial values file can also include `.RNG.seed <-`.
- The random number generator can also be changed in this file with `.RNG.name <-`, using "base::Wichmann-Hill", "base::Marsaglia-Multicarry", "base::Super-Duper", or "base::Mersenne-Twister".
- Reminder: `NaiveSE` $= \sqrt{\text{sample variance}}/\sqrt{n}$ and: `TIMEseriesSE` $= \sqrt{\text{spectral density var}}/\sqrt{n} =$ asymptotic SE.

Running the Model in JAGS (which G&H do not do)

- Run from R (jags.model from package rjags) with efficient use of datastructures.

```
# WRITE TO A FILE IF WANTED
write.model(cd4.rjags, "CLASSES/Class.Multilevel/cd4.rjags")

# DO A SHORT RUN JUST TO SEE IF WORKING OKAY
cd4.model <- jags.model(file="CLASSES/Class.Multilevel/Code/cd4.rjags",
                        data=cd4.list, n.chains=3, n.adapt=1000)

# ANOTHER WAY TO RUN THE CHAIN
update(cd4.model, n.iter=10000)

# FINAL RUN WITH MULTIPLE STARTING POINTS SAVING VALUES
cd4.mcmc <- coda.samples(model=cd4.model, n.iter=10000, inits=cd4.inits,
            variable.names=c("alpha","beta","tau.alpha","tau.beta","tau.y"))

# RESULTS ARE LARGE SO SAVE TO A FILE
sink("CLASSES/Class.Multilevel/cd4.mcmc.output")
cd4.mcmc
sink()
```

Running the Model in JAGS (which G&H do not do)

```
summary(cd4.mcmc)
                Mean      SD Naive SE Time-series SE
alpha[1]    4.62e+00 0.51576 2.98e-03      5.78e-03
alpha[2]    5.37e+00 0.44082 2.55e-03      4.50e-03
:
alpha[368] -4.66e-02 4.89979 2.83e-02      2.81e-02
alpha[369] -3.04e-02 4.94206 2.85e-02      2.86e-02
beta[1]    -4.70e-01 0.41540 2.40e-03      4.62e-03
beta[2]    -1.66e-02 0.52740 3.04e-03      5.36e-03
:
beta[368]  -3.10e-03 0.76476 4.42e-03      4.42e-03
beta[369]  -3.15e-03 0.76321 4.41e-03      4.37e-03
tau.alpha   4.25e-02 0.00662 3.82e-05      3.95e-05
tau.beta    1.96e+00 0.81989 4.73e-03      1.91e-02
tau.y       1.77e+00 0.16833 9.72e-04      1.99e-03
```

Running the Model in JAGS (which G&H do not do)

```
superdiag(as.mcmc.list(cd4.mcmc), burnin=0)
:

( cd4.dic <- dic.samples(cd4.model, n.iter=1000, type="pD") )
  |**************************************************| 100%
Mean deviance:  840              # \bar{D(\theta)}
penalty 116                      # p_D
Penalized deviance: 956          # DIC
```

Enhanced Model: Bivariate Prior

- Return to the Wishart distribution from Chapters 13 and 15, which is the multivariate generalization of the gamma distribution.
- The random variable is a $k \times k$ symmetric positive definite matrix $\mathbf{X}$.
- PDF: $\mathcal{W}(\mathbf{X}|\alpha, \mathbf{R}) = \frac{|\mathbf{X}|^{(\alpha-(k+1))/2}}{\Gamma_k(\alpha)|\mathbf{R}|^{\alpha/2}} \exp[-\text{tr}(\mathbf{R}^{-1}\mathbf{X})/2]$

  where: $\Gamma_k(\alpha) = 2^{\alpha k/2}\pi^{k(k-1)/4}\prod_{i=1}^{k}\Gamma\left(\frac{\alpha+1-i}{2}\right)$

  $2\alpha > k-1$, and $\mathbf{R}$ symmetric nonsingular
- Positive definite: $\mathbf{q}'\mathbf{X}\mathbf{q} > 0$ for any conformable, non-null $\mathbf{q}$. Nonsingular: invertible (non-zero determinant).
- Here $|\mathbf{X}|$ and $|\mathbf{R}|$are matrix determinants.
- $E[\mathbf{X}_{ij}] = \alpha\mathbf{R}_{ij}$
- $\text{Var}[X_{ij}] = \alpha(\mathbf{R}_{ij}^2 + \mathbf{R}_{ii}\mathbf{R}_{jj})$
- $\text{Cov}[X_{ij}, X_{kl}] = \alpha(\mathbf{R}_{ik}\mathbf{R}_{jl} + \mathbf{R}_{il}\mathbf{R}_{jk})$

Enhanced Model: Bivariate Prior

```
cd4.rjags  <- function()  {
    for (i in 1:COUNTS) {
        mu[i] <- beta[person[i],1] + beta[person[i],2]*time[i]
        y[i] ~ dnorm(mu[i],tau.y)
    }
    for (j in 1:KIDS)  {
        beta[j,1:2] ~ dmnorm(beta.mu,beta.W) # NOTICE THE "m"
    }
    beta.mu[1] ~ dt(0,1,4)  #(mu, tau, k)
    beta.mu[2] ~ dt(0,1,5)
    beta.W ~ dwish(R[,],2)
    R[1,1] <- 3.0
    R[2,2] <- 3.0
    R[1,2] <- 0.5
    R[2,1] <- 0.5
    tau.y ~ dgamma(1,0.1)
}
```

Enhanced Model: Bivariate Prior

- The same jags call except that the starting points have to reflect the vector/matrix structure:

```
write.model(cd4.rjags, "CLASSES/Class.Multilevel/cd4.rjags")
cd4.inits <- list(
    "inits1" <- list("beta"=runif(366,-10,10), "beta.mu"=rnorm(2,0,10),
                "beta.W"=matrix(rgamma(4,1,0.1),ncol=2), "tau.y"=runif(1,0,10)),
    "inits2" <- list("beta"=runif(366,-10,10), "beta.mu"=rnorm(2,0,10),
                "beta.W"=matrix(rgamma(4,1,0.1),ncol=2), "tau.y"=runif(1,0,10)),
    "inits3" <- list("beta"=runif(366,-10,10), "beta.mu"=rnorm(2,0,10),
                "beta.W"=matrix(rgamma(4,1,0.1),ncol=2), "tau.y"=runif(1,0,10))
)
cd4.model <- jags.model(file="CLASSES/Class.Multilevel/cd4.rjags", data=cd4.list,
                        n.chains=3,  n.adapt=1000)
update(cd4.model, n.iter=10000)
cd4.mcmc <- coda.samples(model=cd4.model, n.iter=10000, inits=cd4.inits,
            variable.names=c("beta","beta.mu","beta.W","tau.y"))
```

Enhanced Model: Bivariate Prior

```
summary(cd4.mcmc)
:
beta[82,2]  -2.23486 -0.55793  0.270646  1.11209  2.7425
beta[83,2]  -2.65922 -0.96624 -0.127047  0.70629  2.3559
beta[84,2]  -2.68186 -1.02856 -0.187254  0.64548  2.3013
beta.W[1,1]  0.47643  0.60932  0.691473  0.78362  0.9939
beta.W[2,1]  0.09466  0.23448  0.310863  0.39392  0.5783
beta.W[1,2]  0.09466  0.23448  0.310863  0.39392  0.5783
beta.W[2,2]  0.43387  0.60531  0.719445  0.85895  1.2192
beta.mu[1]   4.17303  4.39974  4.515727  4.63204  4.8582
beta.mu[2]  -0.30046 -0.03490  0.103303  0.23834  0.5142
tau.y        0.83436  0.94675  1.011231  1.07845  1.2142

superdiag(as.mcmc.list(cd4.mcmc), burnin=0)
:
( cd4.dic <- dic.samples(cd4.model, n.iter=1000, type="pD") )
Mean deviance:  1037
penalty 113.2
Penalized deviance: 1150
```

Superpopulation and Finite-Population Variances

- The variation among these can be summarized as:
  1. superpopulation standard deviation, σ_α, gives the variation of the population from which our J groups are drawn.
  2. finite-population standard deviation, s_α, gives the variation of the sampled J groups.
- This is the usual distinction, except that it exists *at the group level* rather than just at the data level.
- The reason that G&H make this distinction here is that this type of variance *never goes away* asymptotically (in any sense), unless the group distinction does not matter.

For a Group-Level Specification

- One level:

$$\alpha_j = U_j\gamma + \eta_j, \qquad \eta_j \sim N(0, \sigma_\alpha^2), \quad \text{for } j = 1, \ldots, J$$

- So the superpopulation standard deviation is σ_α, and the finite-population standard deviation is calculated from:

$$s_\alpha = \sqrt{\frac{1}{J-1}\sum_{j=1}^{J}(\eta_j - \bar{\eta})^2}$$

- We will still estimate both statistics.
- Meaning that these are not *different* estimates of the *same* unknown quantity, they are addressing different quantities:
  - σ_α addresses cases in the sample and all other possible cases.
  - s_α corresponds to the sample at hand.

Non-Nested Example: Flight Simulators

- See G&H pages 289, 380, and 424.
- A psychological experiment on pilots with $n = 40$ datapoints and $J = 5$ treatment conditions at $K = 8$ airports for response times y_i.
- The non-nested model is:

$$\begin{aligned} y_i &\sim N(\mu + \gamma_{j[i]} + \delta_{k[i]}, \sigma_y^2), \quad \text{for } i = 1, \ldots, n \\ \gamma_j &\sim N(0, \sigma_\gamma^2), \quad \text{for } j = 1, \ldots, J \\ \delta_k &\sim N(0, \sigma_\delta^2), \quad \text{for } k = 1, \ldots, K \end{aligned}$$

  where γ_j is the treatment effect and δ_k is the airport effect.
- The higher level distributions are centered at zero since the individual-level model has an intercept and the μ_γ, μ_δ would then just fall into the μ term at this individual-level. Conversely we could switch this strategy, but including all three would lead to non-indentifiability.

Non-Nested Example: Flight Simulators

- The data:

airport	treatment conditions				
1	0.38	0.25	0.50	0.14	0.43
2	0.00	0.00	0.67	0.00	0.00
3	0.38	0.50	0.33	0.71	0.29
4	0.00	0.12	0.00	0.00	0.86
5	0.33	0.50	0.14	0.29	0.86
6	1.00	1.00	1.00	1.00	0.86
7	0.12	0.12	0.00	0.14	0.14
8	1.00	0.86	1.00	1.00	0.75

Flight Simulators Data in List Form

```
y <-c(0.38,0.00,0.38,0.00,0.33,1.00,0.12,1.00,0.25,0.00,0.50,0.12,0.50,
      1.00,0.12,0.86,0.50,0.67,0.33,0.00,0.14,1.00,0.00,1.00,0.14,0.00,
      0.71,0.00,0.29,1.00,0.14,1.00,0.43,0.00,0.29,0.86,0.86,0.86,0.14,0.75)
airport <- rep(1:8,5)
treatment <- rep(1:5,each=8)
airport.jags.list <- list("y"=y, "airport"=airport, "treatment"=treatment,
    "n"=40, "n.airport"=8, "n.treatment"=5)
```

Flight Simulators Data in List Form

```
airport.jags.list
$y
 [1] 0.38 0.00 0.38 0.00 0.33 1.00 0.12 1.00 0.25 0.00 0.50 0.12 0.50 1.00 0.12 0.86
[17] 0.50 0.67 0.33 0.00 0.14 1.00 0.00 1.00 0.14 0.00 0.71 0.00 0.29 1.00 0.14 1.00
[33] 0.43 0.00 0.29 0.86 0.86 0.86 0.14 0.75

$airport
[1] 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8

$treatment
[1] 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5

$n
[1] 40

$n.airport
[1] 8

$n.treatment
[1] 5
```

Running From R

```
library("lme4","rjags","coda","superdiag")

# DEFINE THE MODEL
airport.model1.rjags  <- function()  {
    for (i in 1:n) {
        y[i] ~ dnorm(y.hat[i], tau.y)
        y.hat[i] <- mu + gamma[treatment[i]] + delta[airport[i]]
    }
    mu ~ dnorm(0, 0.0001)
    tau.y <- pow(sigma.y,-2)
    sigma.y ~ dunif(0, 100)
    for (j in 1:n.treatment) { gamma[j] ~ dnorm(0,tau.gamma) }
    tau.gamma <- pow(sigma.gamma,-2)
    sigma.gamma ~ dunif(0, 100)
    for (k in 1:n.airport) { delta[k] ~ dnorm(0, tau.delta) }
    tau.delta <- pow(sigma.delta,-2)
    sigma.delta~ dunif(0, 100)
}
write.model(airport.model1.rjags, "Class.Multilevel/airport.model1.rjags")
```

Running From R

```
# RUN THE SAMPLER AND COLLECT coda SAMPLES
airport1.model <- jags.model(file="Class.Multilevel/airport.model1.rjags",
              data=airport.jags.list, n.chains=3,  n.adapt=500)
update(airport1.model, n.iter=2500)
airport1.mcmc <- coda.samples(model=airport1.model,
    variable.names=names(airport.jags.list),n.iter=2500)
summary(airport1.mcmc)

Iterations = 3001:5500
Thinning interval = 1
Number of chains = 3
Sample size per chain = 2500

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

              Mean SD Naive SE Time-series SE
airport[1]    1.00  0        0              0
airport[2]    2.00  0        0              0
```

airport[3]	3.00	0	0	0
airport[4]	4.00	0	0	0
airport[5]	5.00	0	0	0
airport[6]	6.00	0	0	0
airport[7]	7.00	0	0	0
airport[8]	8.00	0	0	0
airport[9]	1.00	0	0	0
airport[10]	2.00	0	0	0
airport[11]	3.00	0	0	0
airport[12]	4.00	0	0	0
airport[13]	5.00	0	0	0
airport[14]	6.00	0	0	0
airport[15]	7.00	0	0	0
airport[16]	8.00	0	0	0
airport[17]	1.00	0	0	0
airport[18]	2.00	0	0	0
airport[19]	3.00	0	0	0
airport[20]	4.00	0	0	0
airport[21]	5.00	0	0	0
airport[22]	6.00	0	0	0
airport[23]	7.00	0	0	0

airport[24]	8.00	0	0	0
airport[25]	1.00	0	0	0
airport[26]	2.00	0	0	0
airport[27]	3.00	0	0	0
airport[28]	4.00	0	0	0
airport[29]	5.00	0	0	0
airport[30]	6.00	0	0	0
airport[31]	7.00	0	0	0
airport[32]	8.00	0	0	0
airport[33]	1.00	0	0	0
airport[34]	2.00	0	0	0
airport[35]	3.00	0	0	0
airport[36]	4.00	0	0	0
airport[37]	5.00	0	0	0
airport[38]	6.00	0	0	0
airport[39]	7.00	0	0	0
airport[40]	8.00	0	0	0
n	40.00	0	0	0
n.airport	8.00	0	0	0
n.treatment	5.00	0	0	0
treatment[1]	1.00	0	0	0

treatment[2]	1.00	0	0	0
treatment[3]	1.00	0	0	0
treatment[4]	1.00	0	0	0
treatment[5]	1.00	0	0	0
treatment[6]	1.00	0	0	0
treatment[7]	1.00	0	0	0
treatment[8]	1.00	0	0	0
treatment[9]	2.00	0	0	0
treatment[10]	2.00	0	0	0
treatment[11]	2.00	0	0	0
treatment[12]	2.00	0	0	0
treatment[13]	2.00	0	0	0
treatment[14]	2.00	0	0	0
treatment[15]	2.00	0	0	0
treatment[16]	2.00	0	0	0
treatment[17]	3.00	0	0	0
treatment[18]	3.00	0	0	0
treatment[19]	3.00	0	0	0
treatment[20]	3.00	0	0	0
treatment[21]	3.00	0	0	0
treatment[22]	3.00	0	0	0

treatment[23]	3.00	0	0	0
treatment[24]	3.00	0	0	0
treatment[25]	4.00	0	0	0
treatment[26]	4.00	0	0	0
treatment[27]	4.00	0	0	0
treatment[28]	4.00	0	0	0
treatment[29]	4.00	0	0	0
treatment[30]	4.00	0	0	0
treatment[31]	4.00	0	0	0
treatment[32]	4.00	0	0	0
treatment[33]	5.00	0	0	0
treatment[34]	5.00	0	0	0
treatment[35]	5.00	0	0	0
treatment[36]	5.00	0	0	0
treatment[37]	5.00	0	0	0
treatment[38]	5.00	0	0	0
treatment[39]	5.00	0	0	0
treatment[40]	5.00	0	0	0
y[1]	0.38	0	0	0
y[2]	0.00	0	0	0
y[3]	0.38	0	0	0

y[4]	0.00	0	0	0
y[5]	0.33	0	0	0
y[6]	1.00	0	0	0
y[7]	0.12	0	0	0
y[8]	1.00	0	0	0
y[9]	0.25	0	0	0
y[10]	0.00	0	0	0
y[11]	0.50	0	0	0
y[12]	0.12	0	0	0
y[13]	0.50	0	0	0
y[14]	1.00	0	0	0
y[15]	0.12	0	0	0
y[16]	0.86	0	0	0
y[17]	0.50	0	0	0
y[18]	0.67	0	0	0
y[19]	0.33	0	0	0
y[20]	0.00	0	0	0
y[21]	0.14	0	0	0
y[22]	1.00	0	0	0
y[23]	0.00	0	0	0
y[24]	1.00	0	0	0

```
y[25]          0.14  0        0               0
y[26]          0.00  0        0               0
y[27]          0.71  0        0               0
y[28]          0.00  0        0               0
y[29]          0.29  0        0               0
y[30]          1.00  0        0               0
y[31]          0.14  0        0               0
y[32]          1.00  0        0               0
y[33]          0.43  0        0               0
y[34]          0.00  0        0               0
y[35]          0.29  0        0               0
y[36]          0.86  0        0               0
y[37]          0.86  0        0               0
y[38]          0.86  0        0               0
y[39]          0.14  0        0               0
y[40]          0.75  0        0               0

2. Quantiles for each variable:

              2.5%   25%   50%   75% 97.5%
airport[1]    1.00  1.00  1.00  1.00  1.00
```

airport[2]	2.00	2.00	2.00	2.00	2.00
airport[3]	3.00	3.00	3.00	3.00	3.00
airport[4]	4.00	4.00	4.00	4.00	4.00
airport[5]	5.00	5.00	5.00	5.00	5.00
airport[6]	6.00	6.00	6.00	6.00	6.00
airport[7]	7.00	7.00	7.00	7.00	7.00
airport[8]	8.00	8.00	8.00	8.00	8.00
airport[9]	1.00	1.00	1.00	1.00	1.00
airport[10]	2.00	2.00	2.00	2.00	2.00
airport[11]	3.00	3.00	3.00	3.00	3.00
airport[12]	4.00	4.00	4.00	4.00	4.00
airport[13]	5.00	5.00	5.00	5.00	5.00
airport[14]	6.00	6.00	6.00	6.00	6.00
airport[15]	7.00	7.00	7.00	7.00	7.00
airport[16]	8.00	8.00	8.00	8.00	8.00
airport[17]	1.00	1.00	1.00	1.00	1.00
airport[18]	2.00	2.00	2.00	2.00	2.00
airport[19]	3.00	3.00	3.00	3.00	3.00
airport[20]	4.00	4.00	4.00	4.00	4.00
airport[21]	5.00	5.00	5.00	5.00	5.00
airport[22]	6.00	6.00	6.00	6.00	6.00

airport[23]	7.00	7.00	7.00	7.00	7.00
airport[24]	8.00	8.00	8.00	8.00	8.00
airport[25]	1.00	1.00	1.00	1.00	1.00
airport[26]	2.00	2.00	2.00	2.00	2.00
airport[27]	3.00	3.00	3.00	3.00	3.00
airport[28]	4.00	4.00	4.00	4.00	4.00
airport[29]	5.00	5.00	5.00	5.00	5.00
airport[30]	6.00	6.00	6.00	6.00	6.00
airport[31]	7.00	7.00	7.00	7.00	7.00
airport[32]	8.00	8.00	8.00	8.00	8.00
airport[33]	1.00	1.00	1.00	1.00	1.00
airport[34]	2.00	2.00	2.00	2.00	2.00
airport[35]	3.00	3.00	3.00	3.00	3.00
airport[36]	4.00	4.00	4.00	4.00	4.00
airport[37]	5.00	5.00	5.00	5.00	5.00
airport[38]	6.00	6.00	6.00	6.00	6.00
airport[39]	7.00	7.00	7.00	7.00	7.00
airport[40]	8.00	8.00	8.00	8.00	8.00
n	40.00	40.00	40.00	40.00	40.00
n.airport	8.00	8.00	8.00	8.00	8.00
n.treatment	5.00	5.00	5.00	5.00	5.00

treatment[1]	1.00	1.00	1.00	1.00	1.00
treatment[2]	1.00	1.00	1.00	1.00	1.00
treatment[3]	1.00	1.00	1.00	1.00	1.00
treatment[4]	1.00	1.00	1.00	1.00	1.00
treatment[5]	1.00	1.00	1.00	1.00	1.00
treatment[6]	1.00	1.00	1.00	1.00	1.00
treatment[7]	1.00	1.00	1.00	1.00	1.00
treatment[8]	1.00	1.00	1.00	1.00	1.00
treatment[9]	2.00	2.00	2.00	2.00	2.00
treatment[10]	2.00	2.00	2.00	2.00	2.00
treatment[11]	2.00	2.00	2.00	2.00	2.00
treatment[12]	2.00	2.00	2.00	2.00	2.00
treatment[13]	2.00	2.00	2.00	2.00	2.00
treatment[14]	2.00	2.00	2.00	2.00	2.00
treatment[15]	2.00	2.00	2.00	2.00	2.00
treatment[16]	2.00	2.00	2.00	2.00	2.00
treatment[17]	3.00	3.00	3.00	3.00	3.00
treatment[18]	3.00	3.00	3.00	3.00	3.00
treatment[19]	3.00	3.00	3.00	3.00	3.00
treatment[20]	3.00	3.00	3.00	3.00	3.00
treatment[21]	3.00	3.00	3.00	3.00	3.00

treatment[22]	3.00	3.00	3.00	3.00	3.00
treatment[23]	3.00	3.00	3.00	3.00	3.00
treatment[24]	3.00	3.00	3.00	3.00	3.00
treatment[25]	4.00	4.00	4.00	4.00	4.00
treatment[26]	4.00	4.00	4.00	4.00	4.00
treatment[27]	4.00	4.00	4.00	4.00	4.00
treatment[28]	4.00	4.00	4.00	4.00	4.00
treatment[29]	4.00	4.00	4.00	4.00	4.00
treatment[30]	4.00	4.00	4.00	4.00	4.00
treatment[31]	4.00	4.00	4.00	4.00	4.00
treatment[32]	4.00	4.00	4.00	4.00	4.00
treatment[33]	5.00	5.00	5.00	5.00	5.00
treatment[34]	5.00	5.00	5.00	5.00	5.00
treatment[35]	5.00	5.00	5.00	5.00	5.00
treatment[36]	5.00	5.00	5.00	5.00	5.00
treatment[37]	5.00	5.00	5.00	5.00	5.00
treatment[38]	5.00	5.00	5.00	5.00	5.00
treatment[39]	5.00	5.00	5.00	5.00	5.00
treatment[40]	5.00	5.00	5.00	5.00	5.00
y[1]	0.38	0.38	0.38	0.38	0.38
y[2]	0.00	0.00	0.00	0.00	0.00

y[3]	0.38	0.38	0.38	0.38	0.38
y[4]	0.00	0.00	0.00	0.00	0.00
y[5]	0.33	0.33	0.33	0.33	0.33
y[6]	1.00	1.00	1.00	1.00	1.00
y[7]	0.12	0.12	0.12	0.12	0.12
y[8]	1.00	1.00	1.00	1.00	1.00
y[9]	0.25	0.25	0.25	0.25	0.25
y[10]	0.00	0.00	0.00	0.00	0.00
y[11]	0.50	0.50	0.50	0.50	0.50
y[12]	0.12	0.12	0.12	0.12	0.12
y[13]	0.50	0.50	0.50	0.50	0.50
y[14]	1.00	1.00	1.00	1.00	1.00
y[15]	0.12	0.12	0.12	0.12	0.12
y[16]	0.86	0.86	0.86	0.86	0.86
y[17]	0.50	0.50	0.50	0.50	0.50
y[18]	0.67	0.67	0.67	0.67	0.67
y[19]	0.33	0.33	0.33	0.33	0.33
y[20]	0.00	0.00	0.00	0.00	0.00
y[21]	0.14	0.14	0.14	0.14	0.14
y[22]	1.00	1.00	1.00	1.00	1.00
y[23]	0.00	0.00	0.00	0.00	0.00

y[24]	1.00	1.00	1.00	1.00	1.00
y[25]	0.14	0.14	0.14	0.14	0.14
y[26]	0.00	0.00	0.00	0.00	0.00
y[27]	0.71	0.71	0.71	0.71	0.71
y[28]	0.00	0.00	0.00	0.00	0.00
y[29]	0.29	0.29	0.29	0.29	0.29
y[30]	1.00	1.00	1.00	1.00	1.00
y[31]	0.14	0.14	0.14	0.14	0.14
y[32]	1.00	1.00	1.00	1.00	1.00
y[33]	0.43	0.43	0.43	0.43	0.43
y[34]	0.00	0.00	0.00	0.00	0.00
y[35]	0.29	0.29	0.29	0.29	0.29
y[36]	0.86	0.86	0.86	0.86	0.86
y[37]	0.86	0.86	0.86	0.86	0.86
y[38]	0.86	0.86	0.86	0.86	0.86
y[39]	0.14	0.14	0.14	0.14	0.14
y[40]	0.75	0.75	0.75	0.75	0.75

Non-Nested Example: Flight Simulators

- These results are summarized in G&H.
- $\hat{\sigma}_y = 0.23$, thus variation among individuals is pretty big,

$$y_i \sim N(\mu + \gamma_{j[i]} + \delta_{k[i]}, \sigma_y^2), \quad \text{for } i = 1, \ldots, n$$

- $\hat{\sigma}_\gamma = 0.04$, which means that variation among treatments is very small,

$$\gamma_j \sim N(0, \sigma_\gamma^2), \quad \text{for } j = 1, \ldots, J$$

- $\hat{\sigma}_\delta = 0.32$, so variation among airports is very large,

$$\delta_k \sim N(0, \sigma_\delta^2), \quad \text{for } k = 1, \ldots, K$$

BUGS Code

- To draw the finite-population standard deviations, add the following code:

```
for (i in 1:n) { e.y[i] = y[i] - y.hat[i] }
s.y <- sd(e.y[])          # FINITE-POPULATION STANDARD DEVIATION
s.g <- sd(gamma[])        # TREATMENT COEFFICIENT STANDARD DEVIATION
s.d <- sd(delta[])        # AIRPORT COEFFICIENT STANDARD DEVIATION
```

- Hypothetically suppose there were group-level predictors/covariates at one hierarchy,

```
for (j in 1:n.treatment){
    gamma[j] ~ dnorm(gamma.hat[j],tau.gamma)
    gamma.hat[j] <- a.0 + a.1*u.1[j] + a.2*u.2[j]
    e.g[j] <- gamma[j] - gamma.hat[j]
}
tau.gamma <- pow(sigma.gamma,-2)
s.gamma <- sd(e.g[])
```

then `sigma.gamma` is the estimate of the superpopulation standard deviation, and `s.gamma` is the finite-population standard deviation.

Average Predictive Comparisons

- In a standard multilevel model, $y = f(x, \theta)$, differentiate between:
  - u: the input of interest
  - v: all other inputs

  so $x = (u, v)$.
- We are interested in the expected predictive difference in y per unit difference in u, holding v constant:

$$b_u(u^{(lo)}, u^{(hi)}, v, \theta) = \frac{E(y|u^{(hi)}, v, \theta) - E(y|u^{(low)}, v, \theta)}{u^{(hi)} - u^{(lo)}}$$

  where know the form of $E(y|x, \theta)$
- This is a standard "first difference" scaled by $u^{(hi)} - u^{(lo)}$.

Averaging Across Units

- This quantity averages across all the units to get an average predictive difference per unit:

$$B_u(u^{(lo)}, u^{(hi)}, v, \theta) = \sum_{1}^{n} b_u(u^{(lo)}, u^{(hi)}, v_i, \theta)$$

  (note the v_i indexing).

- First differences can actually be explained more clearly than the language used in the book...

First Differences for Non-Linear Models

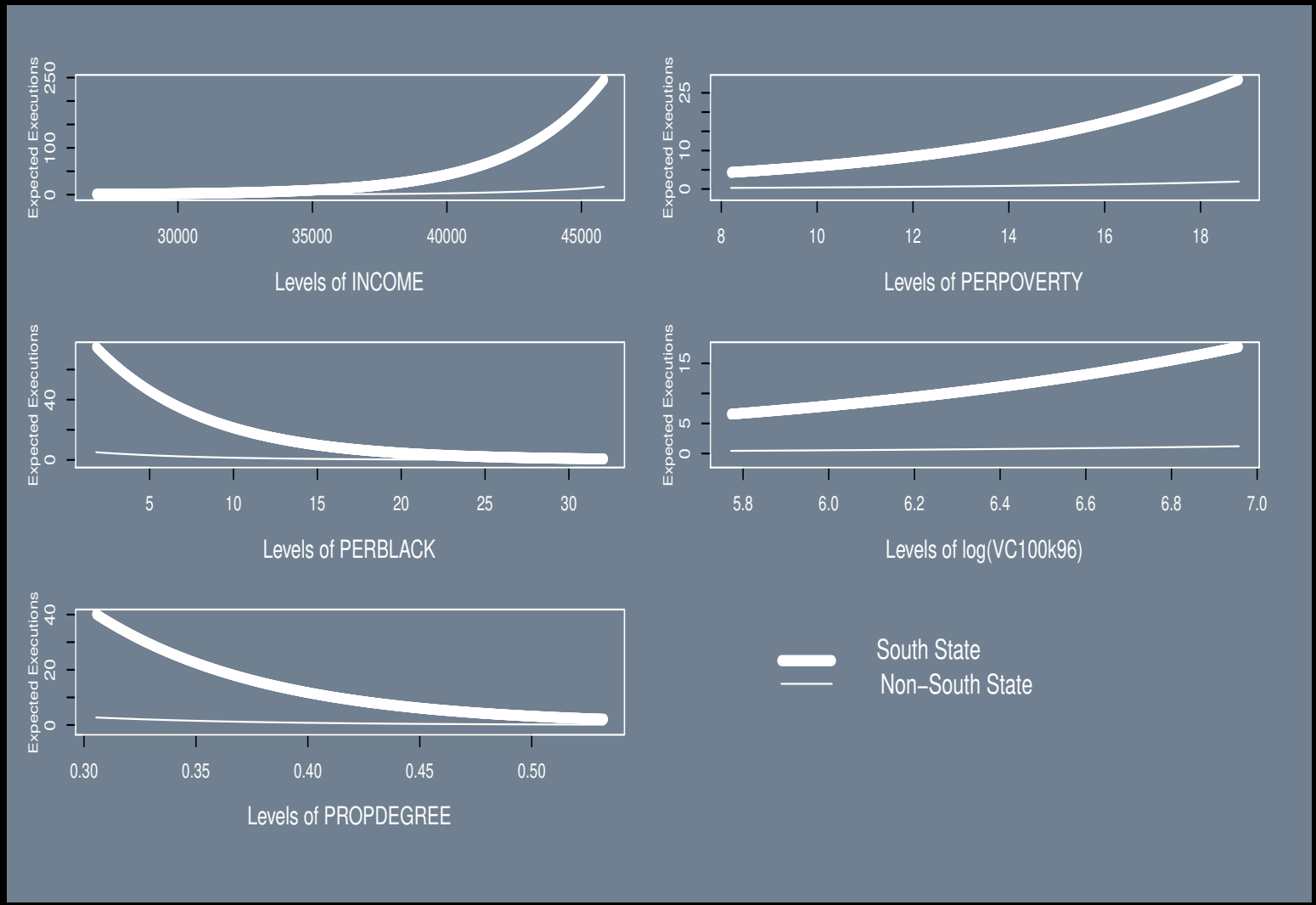
- We can no longer use "a one unit change in X gives a β change in Y."
- Main idea:
  - pick one covariate of interest, $\mathbf{X}_q$
  - choose 2 levels of this variable, $\mathbf{X}_{1,q}$, $\mathbf{X}_{2,q}$
  - set all other covariates at their mean, $\bar{\mathbf{X}}_{\neg q}$
  - create two predictions by running these values through the link function:

$$\hat{Y}_1 = g^{-1}(\bar{\mathbf{X}}_{-q}\hat{\boldsymbol{\beta}}_{\neg q} + \mathbf{X}_{1,q}\hat{\boldsymbol{\beta}}_q)$$

$$\hat{Y}_2 = g^{-1}(\bar{\mathbf{X}}_{q}\hat{\boldsymbol{\beta}}_{\neg q} + \mathbf{X}_{2,q}\hat{\boldsymbol{\beta}}_q)$$

  - Look at $\hat{Y}_2 - \hat{Y}_1$ for the outcome difference.
- For example:

```
dp.1 <- dp.2 <- c(1,apply(dp.97[,c(3,4,5,6,7,15)],2,mean))
dp.1[6] <- 0; dp.2[6] <- 1
y.1 <- exp(dp.1 %*% dp.out$coef); y.2 <- exp(dp.2 %*% dp.out$coef)
y.2 - y.1
```


Expected Executions
0
100
250
30000
35000
40000
45000
Levels of INCOME
Expected Executions
0
10
25
8
10
12
14
16
18
Levels of PERPOVERTY
Expected Executions
0
40
5
10
15
20
25
30
Levels of PERBLACK
Expected Executions
0
5
15
5.8
6.0
6.2
6.4
6.6
6.8
7.0
Levels of log(VC100k96)
Expected Executions
0
20
40
0.30
0.35
0.40
0.45
0.50
Levels of PROPDEGREE
South State
Non–South State

Poisson GLM of Capital Punishment, First Difference Code

```
X <- cbind(rep(1,nrow(dp.97)), as.matrix(dp.97[,3:5]), as.matrix(log(dp.97[,6])),
        as.matrix(dp.97[,7]), as.matrix(dp.97[,15]))
X.0 <- cbind(X[,1:5],rep(0,length=nrow(X)),X[,7])
dimnames(X.0)[[2]] <- names(dp.out$coefficients)
X.1 <- cbind(X[,1:5],rep(1,length=nrow(X)),X[,7])
dimnames(X.1)[[2]] <- names(dp.out$coefficients)

postscript("/Users/jgill/Class.MLE/glm.fig2.ps")
par(mfrow=c(3,2),mar=c(4,3,2,2),oma=c(3,1,1,1),col.axis="white",col.lab="white",
    col.sub="white",col="white",bg="slategray")
```

Poisson GLM of Capital Punishment, First Difference Code

```
for (i in 2:(ncol(X.0)-1))  {
  if (i==6) i <- i+1
  ruler  <- seq(min(X.0[,i]),max(X.0[,i]),length=1000)
  xbeta0 <- exp(dp.out$coefficients[-i]%*%apply(X.0[,-i],2,mean)
                + dp.out$coefficients[i]*ruler)
  xbeta1 <- exp(dp.out$coefficients[-i]%*%apply(X.1[,-i],2,mean)
                + dp.out$coefficients[i]*ruler)
  plot(ruler,xbeta0,type="l",xlab="",ylab="",
      ylim=c(min(xbeta0,xbeta1)-2,max(xbeta0,xbeta1)) )
  lines(ruler,xbeta1,type="b")
  mtext(outer=F,side=1,paste("Levels of",dimnames(X.0)[[2]][i]),cex=0.8,line=3)
  mtext(outer=F,side=2,"Expected Executions",cex=0.6,line=2)
}
plot(ruler[100:200],rep(ruler[400],101),bty="n",xaxt="n",yaxt="n",xlab="",ylab="",
       type="l",xlim=range(ruler),ylim=range(ruler))
lines(ruler[100:200],rep(ruler[600],101),type="b")
text(ruler[445],ruler[400],"Non-South State",cex=1.4)
text(ruler[390],ruler[700],"South State",cex=1.4)
dev.off()
```