Bayesian statistics with R

8. Heterogeneity and multilevel models (aka mixed models)

Olivier Gimenez

March 2021

Multilevel (aka mixed-effect) models

What are multilevel models?

- Multilevel models include both fixed and random effects.

What are multilevel models?

- Multilevel models include both fixed and random effects.

- Random effects are statistical parameters that attempt to **explain noise caused by clusters** of the population you are trying to model.

What are multilevel models?

- Multilevel models include both fixed and random effects.

- Random effects are statistical parameters that attempt to **explain noise caused by clusters** of the population you are trying to model.

- A multilevel model assumes that the dataset being analysed consists of **a hierarchy of different populations** whose differences relate to that hierarchy.

What are multilevel models?

- Multilevel models include both fixed and random effects.

- Random effects are statistical parameters that attempt to **explain noise caused by clusters** of the population you are trying to model.

- A multilevel model assumes that the dataset being analysed consists of **a hierarchy of different populations** whose differences relate to that hierarchy.

- Measurement that come **in clusters** or groups.

What are multilevel models?

- Multilevel models include both fixed and random effects.

- Random effects are statistical parameters that attempt to **explain noise caused by clusters** of the population you are trying to model.

- A multilevel model assumes that the dataset being analysed consists of **a hierarchy of different populations** whose differences relate to that hierarchy.

- Measurement that come **in clusters** or groups.

- Come up with examples of clusters or groups.

Clusters might be:

- Classrooms within schools
- Students within classrooms
- Chapters within books
- Individuals within populations
- Populations within species
- Trajectories within individuals
- Fishes within tanks
- Frogs within ponds
- PhD applicants in doctoral schools
- Nations in continents
- Sex or age are not clusters per se (if we were to sample again, we would take the same levels, e.g. male/female and young/old)

Why do we need multilevel models?

- Model the clustering itself.

Why do we need multilevel models?

- Model the clustering itself.

- Interested in variance components (environmental vs. genetic variance).

Why do we need multilevel models?

- Model the clustering itself.

- Interested in variance components (environmental vs. genetic variance).

- Control for bias due to pseudoreplication (time, space, individual).

McElreath's explanation of multilevel models

- Fixed-effect models have amnesia.

McElreath's explanation of multilevel models

- Fixed-effect models have amnesia.

- Every new cluster (individual, species, classroom) is a new world.

McElreath's explanation of multilevel models

- Fixed-effect models have amnesia.

- Every new cluster (individual, species, classroom) is a new world.

- No information passed among clusters.

McElreath's explanation of multilevel models

- Fixed-effect models have amnesia.

- Every new cluster (individual, species, classroom) is a new world.

- No information passed among clusters.

- Multilevel models remember and pool information. They have memory.

McElreath's explanation of multilevel models

- Fixed-effect models have amnesia.

- Every new cluster (individual, species, classroom) is a new world.

- No information passed among clusters.

- Multilevel models remember and pool information. They have memory.

- Properties of clusters come from a population.

## McElreath's explanation of multilevel models

- Fixed-effect models have amnesia.

- Every new cluster (individual, species, classroom) is a new world.

- No information passed among clusters.

- Multilevel models remember and pool information. They have memory.

- Properties of clusters come from a population.

- If previous clusters improve your guess about a new cluster, you want to use pooling.

Courtesy of Pr Eleni Kazakou
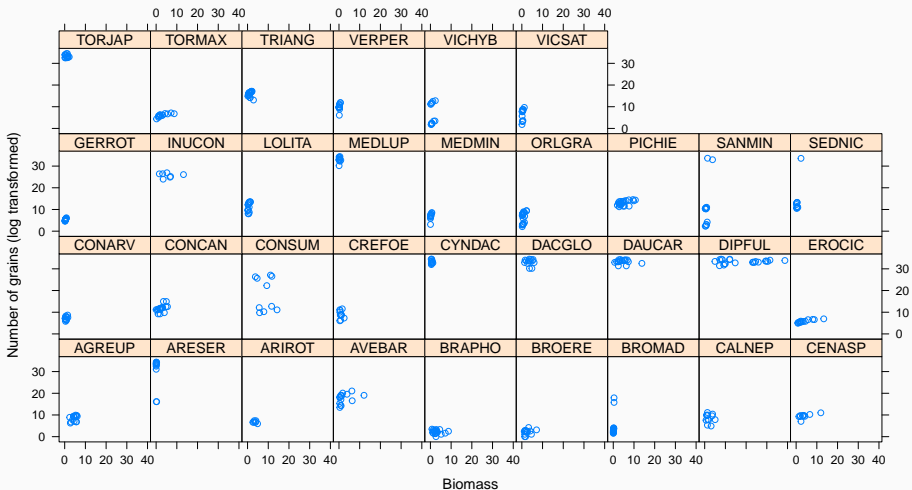
GLM with complete pooling

$$Y_i \sim \text{Distribution}(\text{mean}_i) \qquad \text{[likelihood]}$$
$$\text{link}(\text{mean})_i = \alpha + \beta\, x_i \qquad \text{[linear model]}$$
$$\alpha \sim \text{to be determined} \qquad \text{[prior for intercept]}$$
$$\beta \sim \text{to be determined} \qquad \text{[prior for slope]}$$

Model with complete pooling. All clusters the same.

GLM with no pooling

$$Y_i \sim \text{Distribution(mean}_i) \qquad \text{[likelihood]}$$
$$\text{link(mean)}_i = \alpha_{\text{CLUSTER}[i]} + \beta\, x_i \qquad \text{[linear model]}$$
$$\alpha_j \sim \text{to be determined} \qquad \text{[prior for intercept]}$$
$$\beta \sim \text{to be determined} \qquad \text{[prior for slope]}$$

Model with no pooling. All clusters unrelated (fixed effect).

GLMM or GLM with partial pooling

$$Y_i \sim \text{Distribution}(\text{mean}_i) \qquad \text{[likelihood]}$$

$$\text{link(mean)}_i = \alpha_{\text{CLUSTER}[i]} + \beta\, x_i \qquad \text{[linear model]}$$

$$\alpha_j \sim \text{Normal}(\bar{\alpha}, \sigma) \qquad \text{[prior for varying intercepts]}$$

$$\bar{\alpha} \sim \text{to be determined} \qquad \text{[prior for population mean]}$$

$$\sigma \sim \text{to be determined} \qquad \text{[prior for standard deviation]}$$

$$\beta \sim \text{to be determined} \qquad \text{[prior for slope]}$$

Model with partial pooling. Clusters are somehow related (random effect).

Back to the plant example

Model with complete pooling (all species are the same)

$$\text{nseeds}_i \sim \text{Normal}(\mu_i, \sigma^2) \qquad \text{[likelihood]}$$
$$\mu_i = \alpha + \beta \, \text{biomass}_i \qquad \text{[linear model]}$$
$$\alpha \sim \text{Normal}(0, 1000) \qquad \text{[prior for intercept]}$$
$$\beta \sim \text{Normal}(0, 1000) \qquad \text{[prior for slope]}$$
$$\sigma \sim \text{Uniform}(0, 100) \qquad \text{[prior for standard deviation]}$$

Read in and manipulate data

```r
# read in data
VMG <- read_csv2(here::here("slides","dat","VMG.csv")) %>%
  mutate(Sp = as_factor(Sp), Vm = as.numeric(Vm))
# nb of seeds
y <- log(VMG$NGrTotest)
# biomass
x <- VMG$Vm
x <- (x - mean(x))/sd(x)
# species name
Sp <- VMG$Sp
# species label
species <- as.numeric(Sp)
# species name
nbspecies <- length(levels(Sp))
# total nb of measurements
n <- length(y)
```

Specify the model in Jags

```
model <-
paste("
model{
for(i in 1:n){
    y[i] ~ dnorm(mu[i], tau.y)
    mu[i] <- a + b * x[i]
    }
tau.y <- 1 / (sigma.y * sigma.y)
sigma.y ~ dunif(0,100)
a ~ dnorm(0,0.001)
b ~ dnorm(0,0.001)
}
")
writeLines(model,here::here("slides","code","completepooling.bug"))
```

Prepare ingredients for running Jags

```r
# data
allom.data <- list(y = y, n = n, x = x)

# initial values
init1 <- list(a=rnorm(1), b=rnorm(1),sigma.y=runif(1))
init2 <- list(a=rnorm(1), b=rnorm(1),sigma.y=runif(1))
inits <- list(init1,init2)

# parameters to be estimated
allom.parameters <- c("a", "b", "sigma.y")
```

Run Jags

```
allom.1 <- jags(allom.data,
                inits,
                allom.parameters,
                n.iter = 2500,
                model.file = here::here("slides","code","completepooling.bug"),
                n.chains = 2,
                n.burn = 1000)
#> Compiling model graph
#>    Resolving undeclared variables
#>    Allocating nodes
#> Graph information:
#>    Observed stochastic nodes: 488
#>    Unobserved stochastic nodes: 3
#>    Total graph size: 1956
#>
#> Initializing model
```

Display results

```
allom.1
#> Inference for Bugs model at "/Users/oliviergimenez/Dropbox/OG/GITHUB/bayesian-stats-wit
#>  2 chains, each with 2500 iterations (first 1000 discarded)
#>  n.sims = 3000 iterations saved
#>          mu.vect sd.vect     2.5%      25%      50%      75%     97.5%  Rhat
#> a         13.926   0.477   12.980   13.612   13.925   14.251   14.884 1.001
#> b          3.569   0.471    2.648    3.238    3.577    3.897    4.483 1.001
#> sigma.y   10.430   0.336    9.805   10.202   10.420   10.644   11.126 1.001
#> deviance 3672.036  2.439 3669.273 3670.253 3671.403 3673.181 3678.269 1.002
#>          n.eff
#> a         3000
#> b         3000
#> sigma.y   3000
#> deviance   860
#>
#> For each parameter, n.eff is a crude measure of effective sample size,
#> and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
```
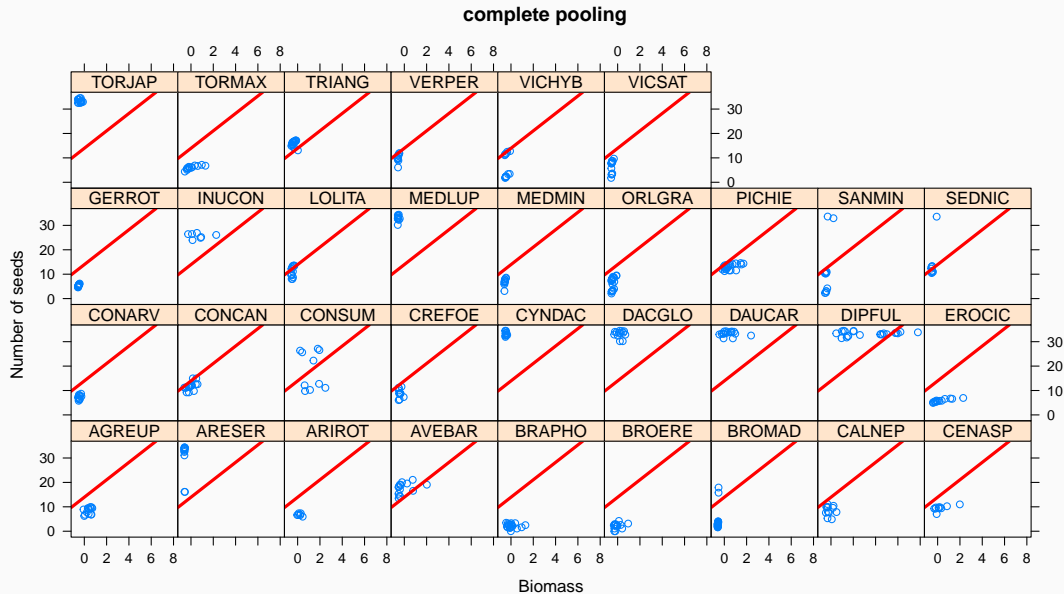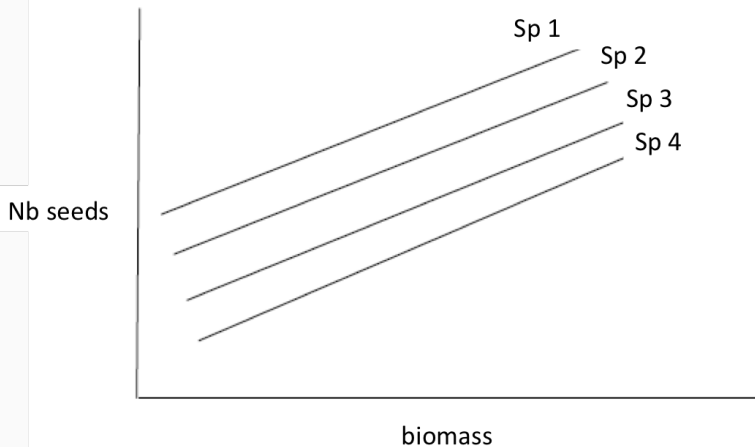
Compare with Frequentist approach

```
freq_lm <- lm(y ~ x, data = allom.data)
freq_lm
#>
#> Call:
#> lm(formula = y ~ x, data = allom.data)
#>
#> Coefficients:
#> (Intercept)            x
#>      13.927        3.578
```

complete pooling

Model with partial pooling (species random effect)

Model with partial pooling (all species related in some way)

$$
\begin{aligned}
\text{nseeds}_i &\sim \text{Normal}(\mu_i, \sigma^2) && \text{[likelihood]} \\
\mu_i &= \alpha_{\text{species}[i]} + \beta \, \text{biomass}_i && \text{[linear model]} \\
\alpha_j &\sim \text{Normal}(\bar{\alpha}, \sigma_\alpha) && \text{[prior for varying intercepts]} \\
\bar{\alpha} &\sim \text{Normal}(0, 1000) && \text{[prior for population mean]} \\
\sigma_\alpha &\sim \text{Uniform}(0, 100) && \text{[prior for } \sigma_\alpha\text{]} \\
\beta &\sim \text{Normal}(0, 1000) && \text{[prior for slope]} \\
\sigma &\sim \text{Uniform}(0, 100) && \text{[prior for } \sigma\text{]}
\end{aligned}
$$

Implementation in Jags

```r
model <- paste("
model {
  for (i in 1:n){
    y[i] ~ dnorm(mu[i], tau.y)
    mu[i] <- a[species[i]] + b * x[i]
  }
  tau.y <- 1/ (sigma.y * sigma.y)
  sigma.y ~ dunif(0, 100)
  for (j in 1:nbspecies){
    a[j] ~ dnorm(mu.a, tau.a)
  }
  mu.a ~ dnorm(0, 0.001)
  tau.a <- 1/(sigma.a * sigma.a)
  sigma.a ~ dunif(0, 100)
  b ~ dnorm (0, 0.001)
}")
writeLines(model,here::here("slides","code","varint.bug"))
```

Prepare ingredients for running Jags

```
allom.data <- list(n = n,
                   nbspecies = nbspecies,
                   x = x,
                   y = y,
                   species = species)
init1 <- list(a = rnorm(nbspecies), b = rnorm(1), mu.a = rnorm(1),
              sigma.y = runif(1), sigma.a=runif(1))
init2 <- list(a = rnorm(nbspecies), b = rnorm(1), mu.a = rnorm(1),
              sigma.y = runif(1), sigma.a = runif(1))
inits <- list(init1,init2)
allom.parameters <- c("b", "mu.a","sigma.y", "sigma.a")
```

Run Jags

```
allom.2 <- jags(allom.data,
                inits,
                allom.parameters,
                n.iter = 2500,
                model.file = here::here("slides","code","varint.bug"),
                n.chains = 2,
                n.burn = 1000)
#> Compiling model graph
#>    Resolving undeclared variables
#>    Allocating nodes
#> Graph information:
#>    Observed stochastic nodes: 488
#>    Unobserved stochastic nodes: 37
#>    Total graph size: 2484
#>
#> Initializing model
```

Display results

```
allom.2
#> Inference for Bugs model at "/Users/oliviergimenez/Dropbox/OG/GITHUB/bayesian-stats-wit
#>  2 chains, each with 2500 iterations (first 1000 discarded)
#>  n.sims = 3000 iterations saved
#>           mu.vect sd.vect     2.5%      25%      50%      75%    97.5%  Rhat
#> b           0.472   0.239   -0.004    0.309    0.473    0.638    0.935 1.002
#> mu.a       14.452   1.933   10.642   13.164   14.440   15.772   18.200 1.001
#> sigma.a    11.026   1.440    8.634   10.027   10.870   11.824   14.390 1.001
#> sigma.y     3.069   0.101    2.876    3.000    3.068    3.134    3.284 1.002
#> deviance 2478.124   8.699 2463.295 2471.940 2477.327 2483.570 2496.793 1.001
#>           n.eff
#> b          1500
#> mu.a       2000
#> sigma.a    3000
#> sigma.y    1600
#> deviance   3000
#>
```

24

Compare with Frequentist approach

```
library(lme4)
freq_lmm <- lmer(y ~ x + (1 | species), allom.data, REML = FALSE)
freq_lmm
#> Linear mixed model fit by maximum likelihood  ['lmerMod']
#> Formula: y ~ x + (1 | species)
#>    Data: allom.data
#>       AIC       BIC    logLik   deviance   df.resid
#>  2652.606  2669.368 -1322.303   2644.606        484
#> Random effects:
#>  Groups    Name        Std.Dev.
#>  species   (Intercept) 10.472
#>  Residual              3.058
#> Number of obs: 488, groups:  species, 33
#> Fixed Effects:
#> (Intercept)            x
#>      14.526        0.479
```
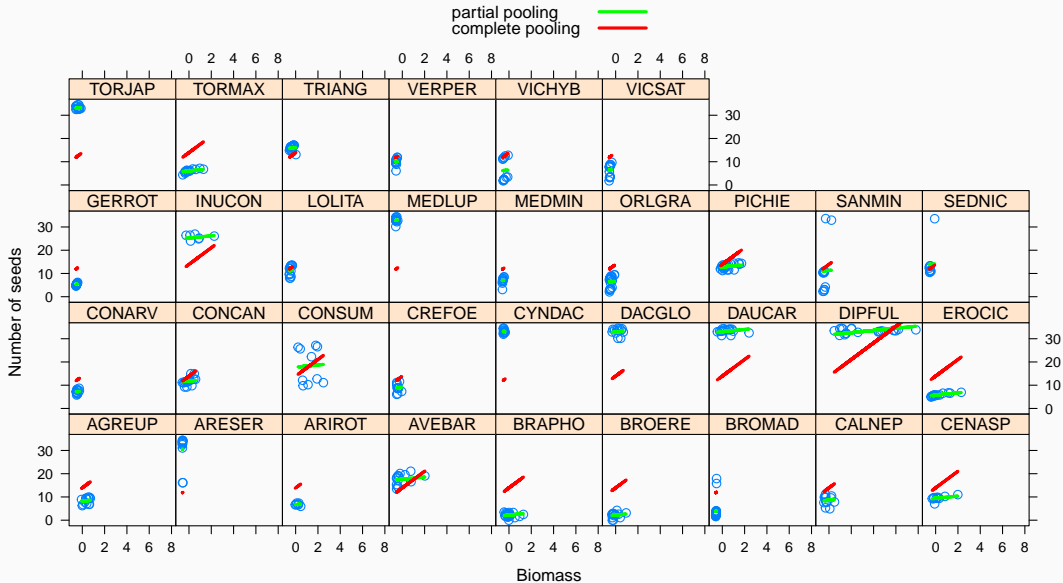
Model with no pooling (all species unrelated)

$$\text{nseeds}_i \sim \text{Normal}(\mu_i, \sigma^2) \qquad \text{[likelihood]}$$

$$\mu_i = \alpha_{\text{species}[i]} + \beta \, \text{biomass}_i \qquad \text{[linear model]}$$

$$\alpha_j \sim \text{Normal}(0, 1000) \qquad \text{[prior for intercepts]}$$

$$\beta \sim \text{Normal}(0, 1000) \qquad \text{[prior for slope]}$$

$$\sigma \sim \text{Uniform}(0, 100) \qquad \text{[prior for} \sigma\text{]}$$

```r
model <- paste("
model {
  for (i in 1:n){
    y[i] ~ dnorm (mu[i], tau.y)
    mu[i] <- a[species[i]] + b * x[i]
  }
  tau.y <- 1 / (sigma.y * sigma.y)
  sigma.y ~ dunif(0, 100)
  for (j in 1:nbspecies){
    a[j] ~ dnorm(0, 0.001)
  }
  b ~ dnorm(0,0.1)
}")
writeLines(model,here::here("slides","code","nopooling.bug"))
```

Prepare ingredients

```
allom.data <- list(n = n, nbspecies = nbspecies, x = x, y = y, species = species)
init1 <- list(a = rnorm(nbspecies), b = rnorm(1), sigma.y = runif(1))
init2 <- list(a = rnorm(nbspecies), b = rnorm(1), sigma.y = runif(1))
inits<-list(init1, init2)
allom.parameters <- c("a","b","sigma.y")
```

Run JAGS

```r
allom.3 <- jags(data = allom.data,
                inits = inits,
                parameters.to.save = allom.parameters,
                n.iter = 2500,
                model.file = here::here("slides","code","nopooling.bug"),
                n.chains = 2,
                n.burn = 1000)
#> Compiling model graph
#>    Resolving undeclared variables
#>    Allocating nodes
#> Graph information:
#>    Observed stochastic nodes: 488
#>    Unobserved stochastic nodes: 35
#>    Total graph size: 2481
#>
#> Initializing model
```

## Display results

```
allom.3$BUGSoutput$summary[c(1:4, 32:33, 34), -c(4,6)]
#>          mean        sd        2.5%        50%       97.5%     Rhat n.eff
#> a[1]   8.1561653 0.8235623  6.50839374  8.1605490  9.777330 1.001131  3000
#> a[2]  30.7454216 0.8831116 28.97873409 30.7478042 32.552915 1.003321   620
#> a[3]   6.6656759 1.1760048  4.36379853  6.6583313  8.911928 1.001422  2000
#> a[4]  17.6434425 0.7804799 16.13317397 17.6378330 19.210157 1.000671  3000
#> a[32]  6.3750576 0.8032132  4.80493250  6.3910267  7.953414 1.002133  1400
#> a[33]  6.6323701 0.7856907  5.11286680  6.6330881  8.136548 1.000667  3000
#> b      0.4428339 0.2387568 -0.03042633  0.4428899  0.899377 1.001561  1700
```

Compare with Frequentist approach

```r
lm(y ~ -1 + as.factor(species) + x, data = allom.data) %>%
  broom::tidy() %>%
  slice(c(1:4, 32:33, 34))
#> # A tibble: 7 x 5
#>   term                estimate std.error statistic   p.value
#>   <chr>                  <dbl>     <dbl>     <dbl>     <dbl>
#> 1 as.factor(species)1     8.17     0.824      9.92 3.92e- 21
#> 2 as.factor(species)2    30.8      0.895     34.4  1.67e-128
#> 3 as.factor(species)3     6.67     1.16       5.76 1.56e-  8
#> 4 as.factor(species)4    17.6      0.791     22.3  5.32e- 75
#> 5 as.factor(species)32    6.38     0.797      8.01 9.95e- 15
#> 6 as.factor(species)33    6.63     0.800      8.29 1.33e- 15
#> 7 x                       0.441    0.243      1.81 7.06e-  2
```
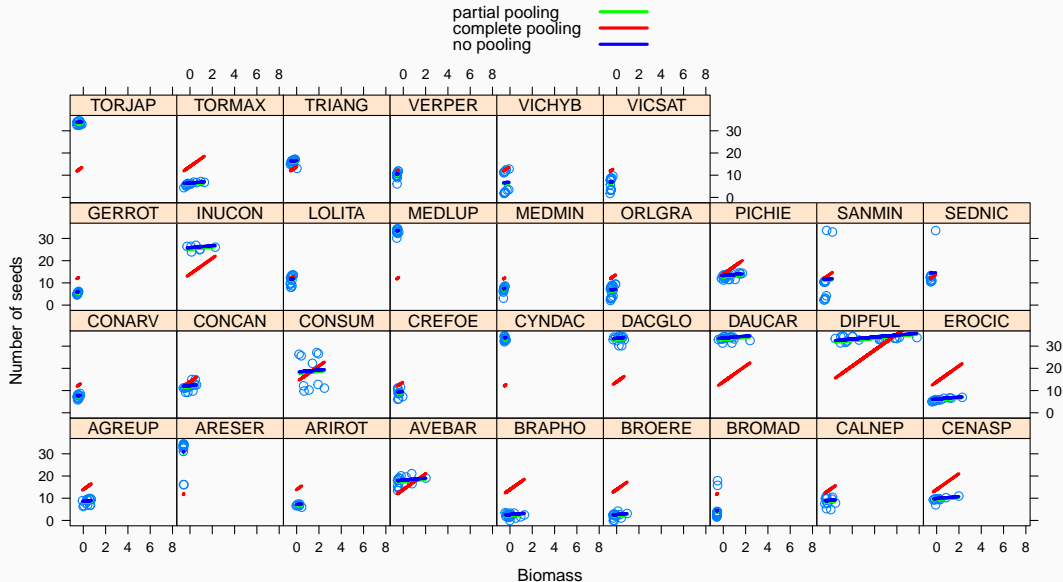
Bonus: Model with varying intercept and varying slope

Code: part 1

```
model <-
paste("
# varying-intercept, varying-slope allometry model
# with Vm as a species predictor

model {
  for (i in 1:n){
    y[i] ~ dnorm (mu[i], tau.y)
    mu[i] <- a[species[i]] + b[species[i]] * x[i]
  }

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

...
```

Code: part 2

```
  for (j in 1:nbspecies){
    a[j] ~ dnorm (mu.a, tau.a)
    b[j] ~ dnorm (mu.b, tau.b)
  }

  mu.a ~ dnorm (0, .001)
  tau.a <- pow(sigma.a, -2)
  sigma.a ~ dunif (0, 100)

  mu.b ~ dnorm (0, .001)
  tau.b <- pow(sigma.b, -2)
  sigma.b ~ dunif (0, 100)

}
")
writeLines(model,here::here("slides","code","varintvarslope.bug"))
```

35

Prepare ingredients

```
init1 <- list(a = rnorm(nbspecies), b = rnorm(nbspecies),
              mu.a = rnorm(1), mu.b = rnorm(1),
              sigma.y = runif(1), sigma.a = runif(1), sigma.b = runif(1))
init2 <- list(a = rnorm(nbspecies), b = rnorm(nbspecies),
              mu.a = rnorm(1), mu.b = rnorm(1),
              sigma.y = runif(1), sigma.a = runif(1), sigma.b = runif(1))
inits <- list(init1, init2)
allom.parameters <- c ("a", "b", "mu.a", "mu.b", "sigma.y", "sigma.a", "sigma.b")
```

Run Jags

```
allom.4 <- jags(data = allom.data,
                inits = inits,
                parameters.to.save = allom.parameters,
                n.iter = 2500,
                model.file = here::here("slides","code","varintvarslope.bug"),
                n.chains = 2,
                n.burn = 1000)
#> Compiling model graph
#>    Resolving undeclared variables
#>    Allocating nodes
#> Graph information:
#>    Observed stochastic nodes: 488
#>    Unobserved stochastic nodes: 71
#>    Total graph size: 2521
#>
#> Initializing model
```

Display results

```
round(allom.4$BUGSoutput$summary[c(1:2, 32:33, 34:35, 65:66, 68:72), -c(4,6)],2)
#>            mean     sd    2.5%    50%  97.5%  Rhat n.eff
#> a[1]       7.77   1.31    5.29   7.77  10.43  1.00  3000
#> a[2]      24.63   6.67    8.81  25.10  36.74  1.11    19
#> a[32]      8.34   1.94    4.63   8.32  12.27  1.00  3000
#> a[33]     13.32   3.93    6.20  13.43  20.84  1.02  2600
#> b[1]       1.65   2.85   -3.89   1.66   7.07  1.00  3000
#> b[2]     -10.13  11.56  -37.68  -9.22  10.75  1.15    14
#> b[32]      5.15   4.50   -3.60   5.15  13.94  1.00  1100
#> b[33]     13.45   7.59   -0.23  13.68  27.88  1.01   320
#> mu.a      16.72   1.97   12.79  16.73  20.52  1.00   740
#> mu.b       5.05   2.39    0.57   5.02   9.88  1.01   470
#> sigma.a   10.83   1.51    8.24  10.68  14.10  1.00   910
#> sigma.b   11.74   2.25    8.00  11.50  16.65  1.04    58
#> sigma.y    2.66   0.09    2.49   2.66   2.85  1.00   640
```

Compare with Frequentist approach

```r
freq_lmm2 <- lmer (y ~ x + (1 + x | species), allom.data, REML = FALSE)
freq_lmm2
#> Linear mixed model fit by maximum likelihood  ['lmerMod']
#> Formula: y ~ x + (1 + x | species)
#>    Data: allom.data
#>       AIC       BIC    logLik   deviance   df.resid
#>  2609.941  2635.083  -1298.971  2597.941        482
#> Random effects:
#>  Groups   Name        Std.Dev. Corr
#>  species  (Intercept) 10.409
#>           x           11.325   0.22
#>  Residual             2.652
#> Number of obs: 488, groups:  species, 33
#> Fixed Effects:
#> (Intercept)            x
#>      16.866        5.244
```

Compare with Frequentist approach - with no correlation

```r
freq_lmm_wocorr <- lmer(y ~ x + (1 | species) +
                                (0 + x | species), allom.data, REML = FALSE)
freq_lmm_wocorr
#> Linear mixed model fit by maximum likelihood  ['lmerMod']
#> Formula: y ~ x + (1 | species) + (0 + x | species)
#>    Data: allom.data
#>      AIC       BIC     logLik   deviance   df.resid
#>  2609.086  2630.037  -1299.543  2599.086        483
#> Random effects:
#>  Groups    Name        Std.Dev.
#>  species   (Intercept) 10.203
#>  species.1 x           10.632
#>  Residual              2.661
#> Number of obs: 488, groups:  species, 33
#> Fixed Effects:
#> (Intercept)            x
#>      16.688        4.929
```

Shrinkage results from pooling of information

- Varying effect estimates shrink towards mean ($\bar{\alpha}$).

Shrinkage results from pooling of information

- Varying effect estimates shrink towards mean $(\bar{\alpha})$.

- Avoids underfitting as in complete pooling model (null variance) or overfitting as in no pooling model (infinite variance).

Shrinkage results from pooling of information

- Varying effect estimates shrink towards mean ($\bar{\alpha}$).

- Avoids underfitting as in complete pooling model (null variance) or overfitting as in no pooling model (infinite variance).

- Varying effects: adaptive regularization through cluster variance estimation.

Shrinkage results from pooling of information

- Varying effect estimates shrink towards mean ($\bar{\alpha}$).

- Avoids underfitting as in complete pooling model (null variance) or overfitting as in no pooling model (infinite variance).

- Varying effects: adaptive regularization through cluster variance estimation.

- Further from mean, more shrinkage.

Shrinkage results from pooling of information

- Varying effect estimates shrink towards mean ($\bar{\alpha}$).

- Avoids underfitting as in complete pooling model (null variance) or overfitting as in no pooling model (infinite variance).

- Varying effects: adaptive regularization through cluster variance estimation.

- Further from mean, more shrinkage.

- Fewer data in cluster, more shrinkage.

Multilevel models are awesome!

Multilevel models in a nutshell

- **Shrinkage via pooling is desirable**. The no-pooling model overstates variation among clusters and makes the individual clusters look more different than they are (overfitting). The complete-pooling model simply ignores the variation among clusters (underfitting).

Multilevel models in a nutshell

- **Shrinkage via pooling is desirable**. The no-pooling model overstates variation among clusters and makes the individual clusters look more different than they are (overfitting). The complete-pooling model simply ignores the variation among clusters (underfitting).

- We can **generalize to a wider population**. Is there an allometry relationship between number of seeds and biomass?

Multilevel models in a nutshell

- **Shrinkage via pooling is desirable**. The no-pooling model overstates variation among clusters and makes the individual clusters look more different than they are (overfitting). The complete-pooling model simply ignores the variation among clusters (underfitting).

- We can **generalize to a wider population**. Is there an allometry relationship between number of seeds and biomass?

- We may consider **varying slopes**. We'd need to deal with correlations between intercept and slope random effects. Open a whole new world with spatial (or time) autocorrelation, phylogenetic regressions, quantitative genetics, network models.

Multilevel models in a nutshell

- **Shrinkage via pooling is desirable**. The no-pooling model overstates variation among clusters and makes the individual clusters look more different than they are (overfitting). The complete-pooling model simply ignores the variation among clusters (underfitting).

- We can **generalize to a wider population**. Is there an allometry relationship between number of seeds and biomass?

- We may consider **varying slopes**. We'd need to deal with correlations between intercept and slope random effects. Open a whole new world with spatial (or time) autocorrelation, phylogenetic regressions, quantitative genetics, network models.

- We may **include predictors at the cluster level**. Imagine we know something about functional traits, and wish to determine whether some species-to-species variation in the allometry relationship is explained by these traits.

Your turn: Practical 8
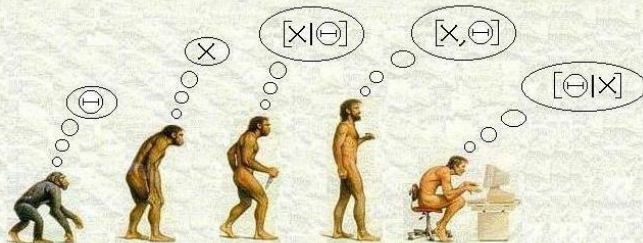
Conclusions

Take-home messages about Bayesian statistics

- Frees the modeler in you (M. Kéry)
    - Uses probability to quantify uncertainty for everything (propagation of uncertainty).
    - Allows use of prior information ('better' estimates).
    - Can fit complex (hierarchical) models with same MCMC algorithms.

Take-home messages about Bayesian statistics

- Frees the modeler in you (M. Kéry)
    - Uses probability to quantify uncertainty for everything (propagation of uncertainty).
    - Allows use of prior information ('better' estimates).
    - Can fit complex (hierarchical) models with same MCMC algorithms.

- With great tools come great responsabilities
    - Checking convergence is painful.
    - Specifying priors might be tricky.
    - Model adequacy should be checked (posterior predictive checks - not covered).
    - Computational burden can be high (see function `R2jags::jags.parallel()` and package 'jagsUI'.

Take-home messages about Bayesian statistics

- Frees the modeler in you (M. Kéry)
    - Uses probability to quantify uncertainty for everything (propagation of uncertainty).
    - Allows use of prior information ('better' estimates).
    - Can fit complex (hierarchical) models with same MCMC algorithms.

- With great tools come great responsabilities
    - Checking convergence is painful.
    - Specifying priors might be tricky.
    - Model adequacy should be checked (posterior predictive checks - not covered).
    - Computational burden can be high (see function `R2jags::jags.parallel()` and package 'jagsUI').

- So what?
    - Make an informed and pragmatic choice.
    - Are you after complexity, speed, uncertainties, etc?
    - Talk to colleagues.

43

Your turn: Practical 9