

# Bayesian statistics with R

## 8. Heterogeneity and multilevel models (aka mixed models)

---

Olivier Gimenez

November 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

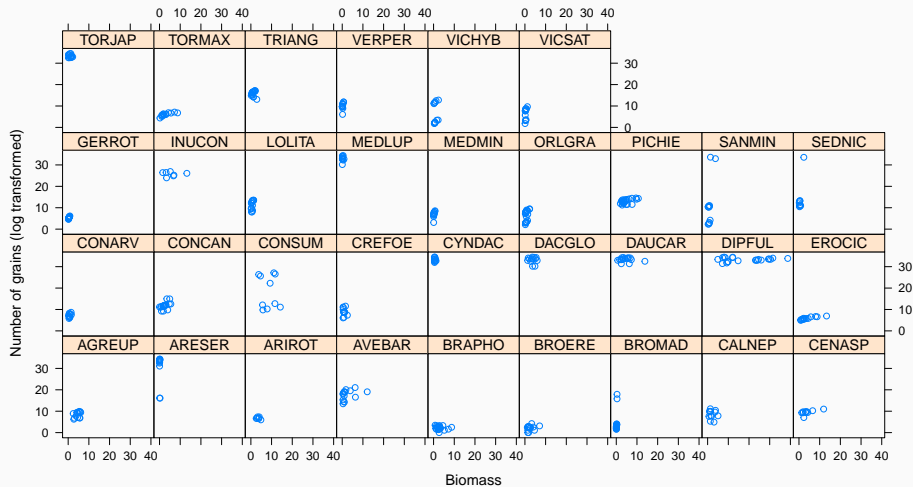
- Fixed-effect models have amnesia.
- Every new cluster (individual, species, classroom) is a new world.
- No information passed among clusters.
- Multilevel models remember and pool information. They have memory.
- Properties of clusters come from a population.
- If previous clusters improve your guess about a new cluster, you want to use pooling.

# Plant experiment in the field at CEFE



Courtesy of Pr Eleni Kazakou

# Number of grains per species (cluster) as a function of biomass



## GLM with complete pooling

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

$$\text{link}(\text{mean})_i = \alpha + \beta x_i \quad [\text{linear model}]$$

$$\alpha \sim \text{to be determined} \quad [\text{prior for intercept}]$$

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

**Model with complete pooling. All clusters the same.**

## GLM with no pooling

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

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

$$\alpha_j \sim \text{to be determined} \quad [\text{prior for intercept}]$$

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

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

## GLMM or GLM with partial pooling

$Y_i \sim \text{Distribution}(\text{mean}_i)$	[likelihood]
$\text{link}(\text{mean})_i = \alpha_{\text{CLUSTER}[i]} + \beta x_i$	[linear model]
$\alpha_j \sim \text{Normal}(\bar{\alpha}, \sigma)$	[prior for varying intercepts]
$\bar{\alpha} \sim \text{to be determined}$	[prior for population mean]
$\sigma \sim \text{to be determined}$	[prior for standard deviation]
$\beta \sim \text{to be determined}$	[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)

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

$\mu_i = \alpha + \beta \text{ biomass}_i$  [linear model]

$\alpha \sim \text{Normal}(0, 1000)$  [prior for intercept]

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

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



## Read in and manipulate data

```
# 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","completepoolsing.bug"))
```

## Prepare ingredients for running Jags

*# 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/oliviorgimenez/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	13.923	0.463	13.015	13.611	13.920	14.239	14.811	1.001
#> b	3.573	0.470	2.662	3.255	3.575	3.891	4.505	1.001
#> sigma.y	10.437	0.337	9.819	10.203	10.428	10.663	11.105	1.002
#> deviance	3671.986	2.350	3669.245	3670.248	3671.398	3673.070	3677.759	1.004
#>	n.eff							
#> a	3000							
#> b	3000							
#> sigma.y	3000							
#> deviance	890							
#>								

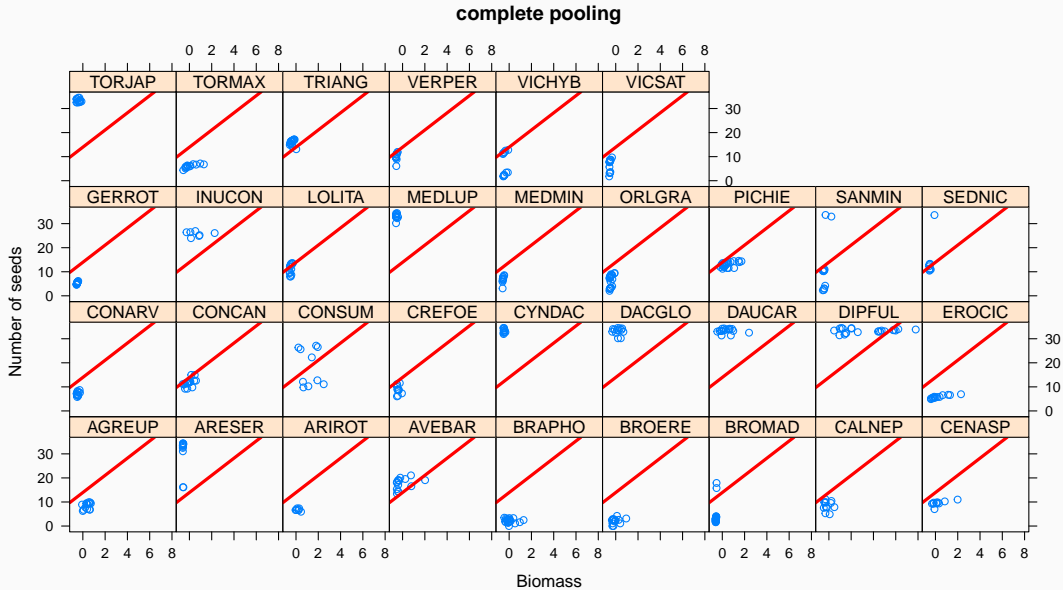
```
#> For each parameter, n.eff is a crude measure of effective sample size,
```

```
#> and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
```

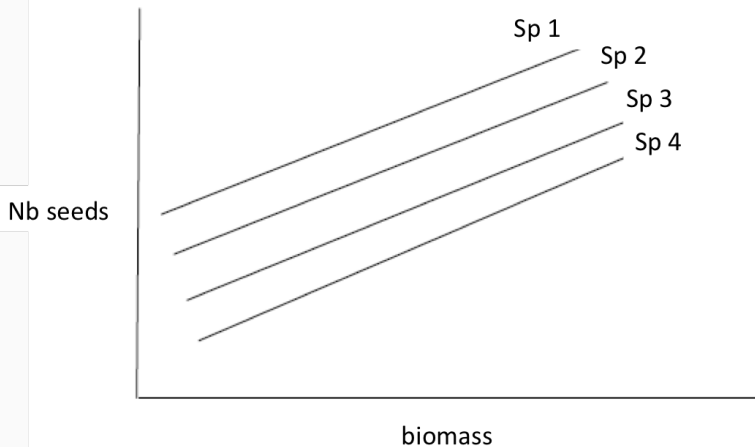
## Compare with Frequentist approach

```
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
```

## Output



## Model with partial pooling (species random effect)





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

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

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

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

$\bar{\alpha} \sim \text{Normal}(0, 1000)$  [prior for population mean]

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

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

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

# Implementation in Jags

```
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/oliviorgimenez/Dropbox/OG/GITHUB/bayesian-stats-wit

#> 2 chains, each with 2500 iterations (first 1000 discarded)

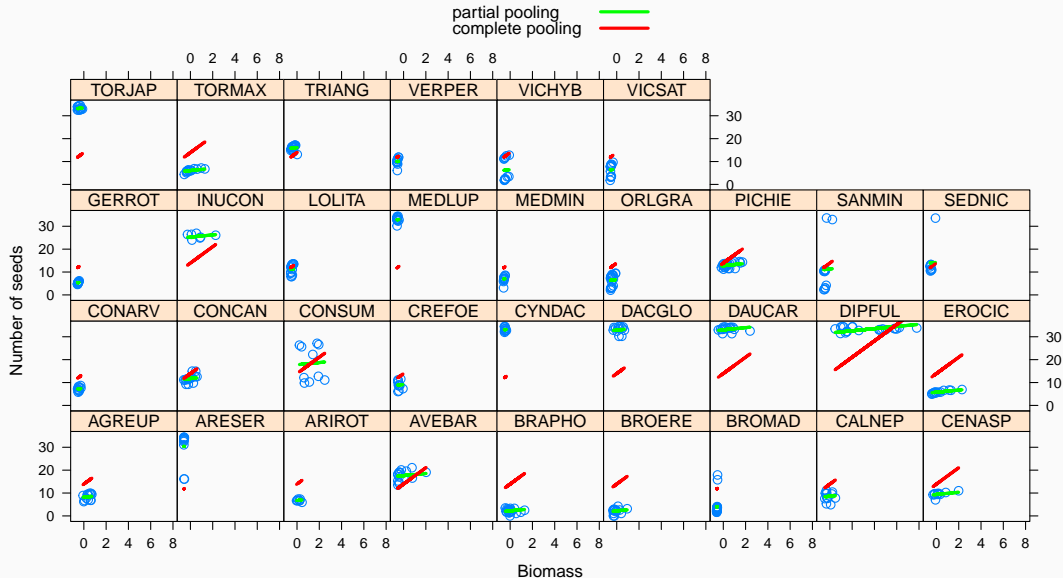
#> n.sims = 3000 iterations saved

#>	mu.vect	sd.vect	2.5%	25%	50%	75%	97.5%	Rhat
#> b	0.475	0.241	-0.006	0.317	0.476	0.634	0.953	1.003
#> mu.a	14.434	1.927	10.428	13.192	14.434	15.691	18.270	1.001
#> sigma.a	11.027	1.456	8.552	10.017	10.859	11.882	14.272	1.001
#> sigma.y	3.067	0.103	2.881	2.995	3.065	3.134	3.279	1.003
#> deviance	2478.040	8.824	2462.956	2471.806	2477.068	2483.649	2497.662	1.001
#>	n.eff							
#> b	760							
#> mu.a	3000							
#> sigma.a	3000							
#> sigma.y	620							
#> deviance	3000							
#>								

## 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
```

# Compare complete pooling vs partial pooling



## Model with no pooling (all species unrelated)

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

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

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

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

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



# Implementation in Jags

```
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

```
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

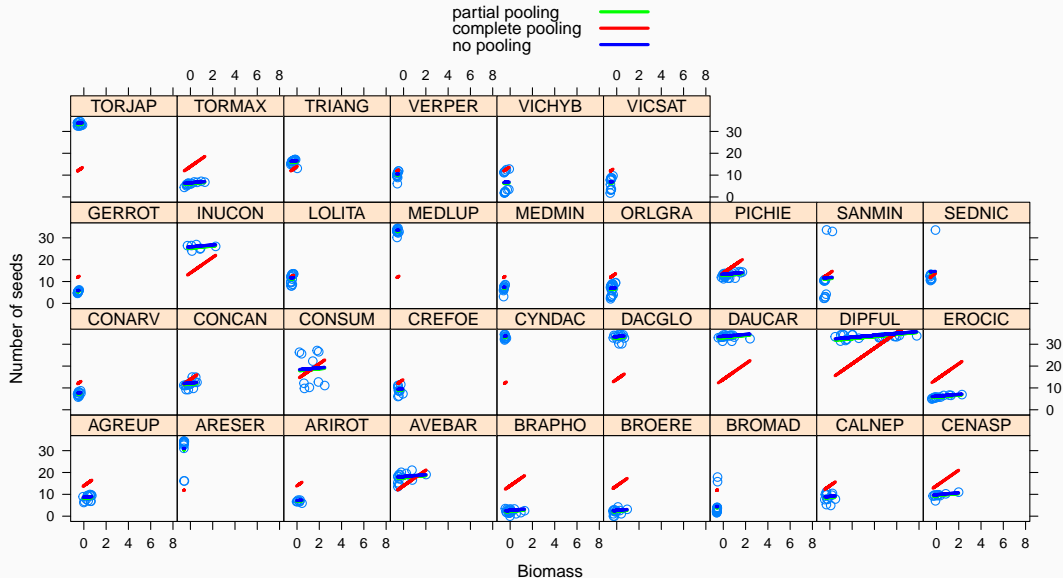
```
allom.3$BUGSoutput$summary[c(1:4, 32:33, 34), -c(4,6)]
```

#>	<i>mean</i>	<i>sd</i>	<i>2.5%</i>	<i>50%</i>	<i>97.5%</i>	<i>Rhat</i>	<i>n.eff</i>
#> a[1]	8.1587806	0.8255539	6.47206297	8.1678198	9.7808819	1.000887	3000
#> a[2]	30.7519970	0.9108058	28.97078788	30.7427963	32.5351465	1.001619	1600
#> a[3]	6.6382393	1.1652806	4.38579525	6.6203551	8.9331740	1.002049	3000
#> a[4]	17.6329823	0.7949344	16.11289584	17.6351150	19.2215977	1.000665	3000
#> a[32]	6.3892192	0.8145411	4.82091923	6.3912129	7.9894134	1.000677	3000
#> a[33]	6.6306778	0.8080154	5.01324244	6.6403791	8.2066792	1.000669	3000
#> b	0.4441261	0.2410698	-0.03120993	0.4428079	0.9115254	1.003689	500

## Compare with Frequentist approach

```
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
```

# Compare complete pooling vs partial pooling vs no pooling



## **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"))
```

## 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]         7.80   1.30    5.18   7.80 10.28 1.01    340
#> a[2]        25.06   6.34   13.46  24.96 37.59 1.01   3000
#> a[32]         8.30   2.11    4.05   8.27 12.68 1.01   1500
#> a[33]        12.68   4.11    3.95  12.96 20.15 1.01    300
#> b[1]          1.59   2.83   -3.78   1.56   7.13 1.01    290
#> b[2]         -9.39  11.01  -29.81  -9.51 12.78 1.00   3000
#> b[32]         5.07   4.85   -4.67   4.98 14.81 1.01   3000
#> b[33]        12.19   7.90   -4.71  12.81 26.43 1.01    310
#> mu.a         16.68   2.01   12.72  16.67 20.60 1.00   1300
#> mu.b          4.88   2.31    0.47   4.85   9.50 1.00    830
#> sigma.a      10.89   1.50    8.39  10.75 14.28 1.01    290
#> sigma.b      11.95   2.45    7.62  11.78 17.35 1.04     44
#> sigma.y       2.66   0.09    2.49   2.66   2.86 1.00   3000
```

## Compare with Frequentist approach

```
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

```
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.

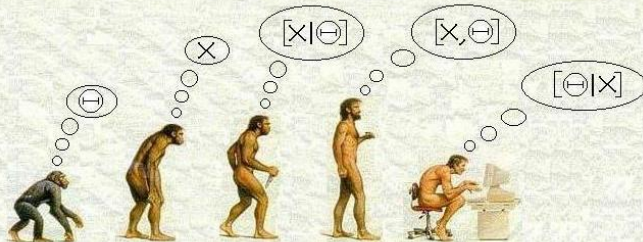
## Take-home messages about Bayesian statistics

- 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 responsibilities
  - 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 responsibilities
  - 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.

(YET ANOTHER) HISTORY OF LIFE AS WE KNOW IT...



HOMO HOMO HOMO HOMO HOMO  
APRIORIUS PRAGMATICUS FREQUENTISTUS SAPIENS BAYESIANIS

## Why become a bayesian? Ask twitter!



**Chelsea Parlett-Pelleriti**  
@ChelseaParlett

Why did you become a Bayesian, wrong answers only

[Traduire le Tweet](#)



## **Your turn: Practical 9**

---