# Bayesian statistics with R
# 8. Heterogeneity and multilevel models (aka mixed models)

Olivier Gimenez

November-December 2024

# 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.

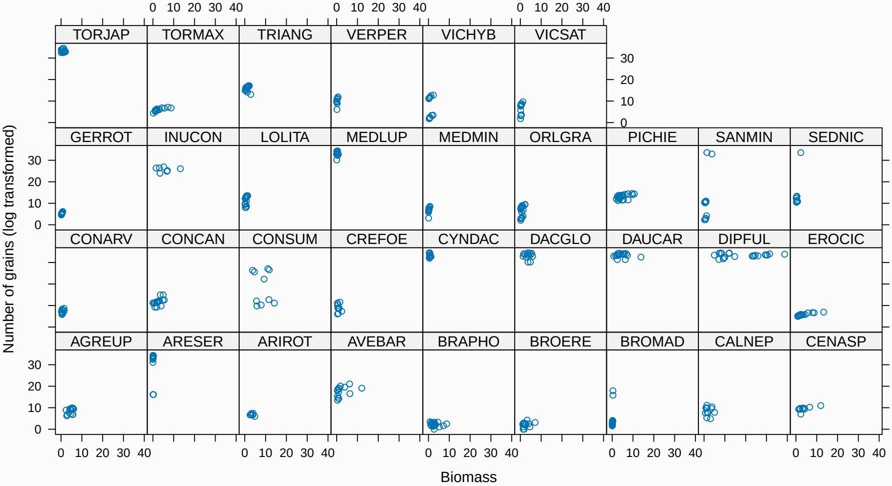
# McElreath's explanation of multilevel models

- 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

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

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

# GLM with no pooling

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

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

# GLMM or GLM with partial pooling

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

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

# Back to the plant example

# Model with complete pooling (all species are the same)

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

# 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

```
# 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
```

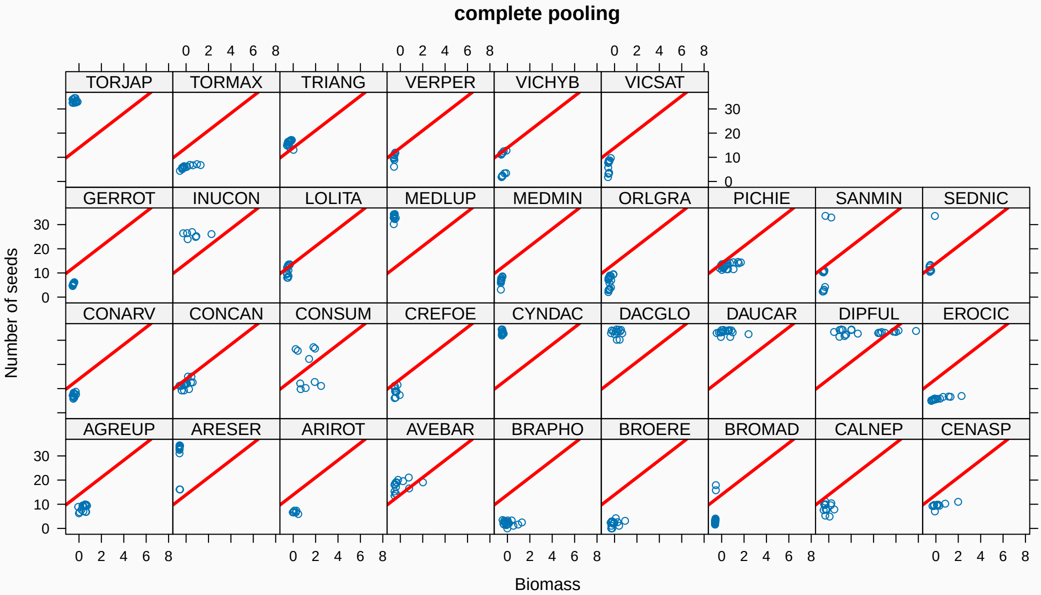
# Display results

```
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. Running time = 0.503 secs
#>           mu.vect sd.vect     2.5%      25%      50%      75%    97.5%  Rhat
#> a          13.916   0.469   12.958   13.613   13.929   14.230   14.829 1.001
#> b           3.570   0.479    2.634    3.246    3.576    3.895    4.496 1.002
#> sigma.y    10.429   0.334    9.811   10.195   10.417   10.652   11.076 1.001
#> deviance 3672.032   2.496 3669.232 3670.253 3671.394 3673.124 3678.600 1.001
#>          n.eff
#> a         3000
#> b         1000
#> sigma.y   2400
#> deviance  3000
#>
#> 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).
```

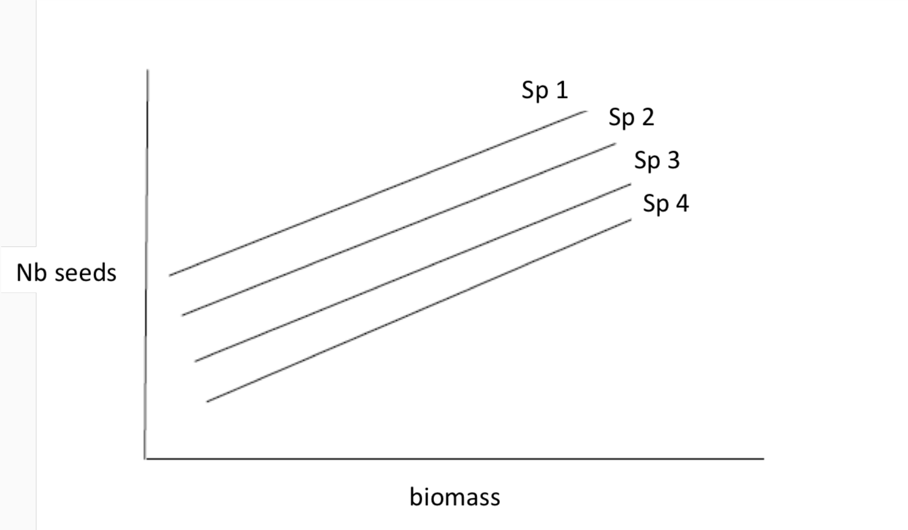
# Compare with Frequentist approach

```r
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


complete pooling
TORJAP
TORMAX
TRIANG
VERPER
VICHYB
VICSAT
GERROT
INUCON
LOLITA
MEDLUP
MEDMIN
ORLGRA
PICHIE
SANMIN
SEDNIC
CONARV
CONCAN
CONSUM
CREFOE
CYNDAC
DACGLO
DAUCAR
DIPFUL
EROCIC
AGREUP
ARESER
ARIROT
AVEBAR
BRAPHO
BROERE
BROMAD
CALNEP
CENASP
Number of seeds
0
10
20
30
Biomass
0
2
4
6
8

# Model with partial pooling (species random effect)



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

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

# 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
```

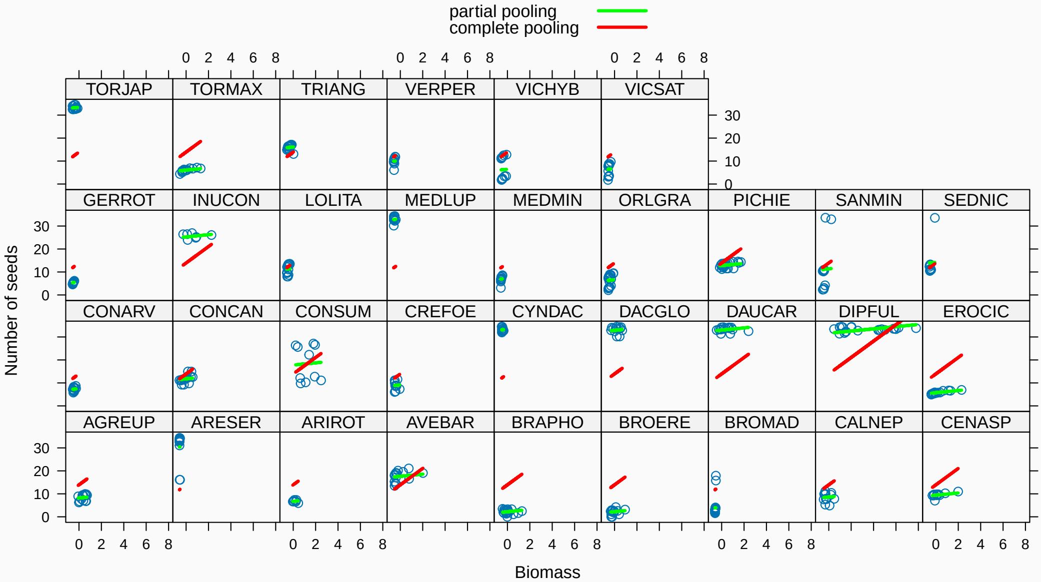
# Display results

```
allom.2
#> Inference for Bugs model at "/Users/oliviergimenez/Dropbox/OG/GITHUB/bayesian-stats-wit
#>  2 chains, each with 2500 iterations (first 1000 discarded)
#>  n.sims = 3000 iterations saved. Running time = 0.507 secs
#>            mu.vect sd.vect    2.5%      25%      50%      75%    97.5%  Rhat
#> b            0.478   0.240   -0.009    0.318    0.480    0.644    0.938 1.001
#> mu.a        14.447   1.954   10.652   13.160   14.437   15.700   18.219 1.001
#> sigma.a     11.126   1.503    8.684   10.103   10.952   11.979   14.592 1.001
#> sigma.y      3.075   0.104    2.881    3.004    3.074    3.147    3.274 1.003
#> deviance  2478.387   8.681 2463.208 2472.329 2477.759 2483.696 2498.167 1.001
#>          n.eff
#> b         3000
#> mu.a      3000
#> sigma.a   3000
#> sigma.y    670
#> 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)

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

# Implementation in Jags

```
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

```r
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
```

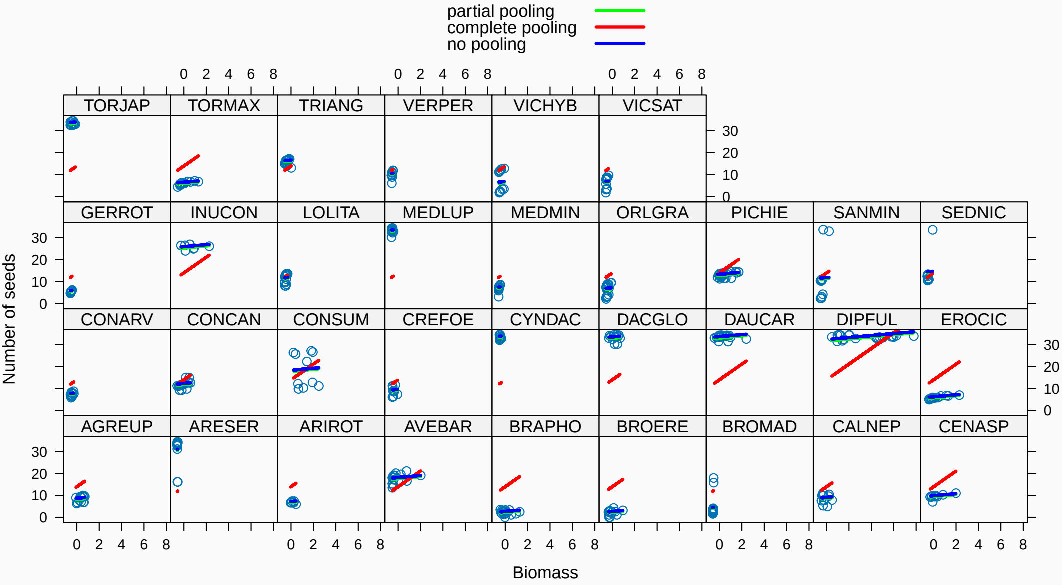
# Display results

```
allom.3$BUGSoutput$summary[c(1:4, 32:33, 34), -c(4,6)]
#>              mean        sd        2.5%        50%      97.5%     Rhat n.eff
#> a[1]   8.1886261 0.8185522  6.62396963  8.1864755  9.8189046 1.000750  3000
#> a[2]  30.7316032 0.8899398 29.05420239 30.7338143 32.4747557 1.002763   720
#> a[3]   6.6552896 1.1669822  4.43517720  6.6533508  9.0051123 1.001328  3000
#> a[4]  17.6224399 0.7857028 16.02560200 17.6325943 19.1593966 1.000878  3000
#> a[32]  6.3811281 0.8173090  4.77163480  6.3820607  8.0252336 1.001332  2200
#> a[33]  6.5981116 0.8124098  5.02335729  6.5979696  8.1669467 1.000697  3000
#> b      0.4516762 0.2398144 -0.03511515  0.4544029  0.9215743 1.001444  3000
```

# 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)

}
")
```

# 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  1.28   5.23  7.79  10.23 1.00   530
#> a[2]       24.41  6.16  12.74 24.74  35.49 1.01  3000
#> a[32]       8.30  2.00   4.54  8.31  12.13 1.01   460
#> a[33]      13.05  4.16   4.91 12.86  21.13 1.00  3000
#> b[1]        1.64  2.73  -3.59  1.63   7.16 1.00   800
#> b[2]      -10.53 10.68 -31.06 -9.87   8.55 1.01  1300
#> b[32]       5.10  4.57  -3.48  5.01  14.23 1.01  2100
#> b[33]      12.95  8.02  -2.65 12.69  28.87 1.00  3000
#> mu.a       16.68  1.95  12.79 16.70  20.60 1.00   420
#> mu.b        5.04  2.21   0.78  5.00   9.64 1.01   130
#> sigma.a    10.88  1.53   8.40 10.71  14.30 1.00   430
#> sigma.b    11.44  2.16   7.86 11.18  16.09 1.02   170
#> sigma.y     2.67  0.10   2.49  2.66   2.86 1.00  2500
```

# 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

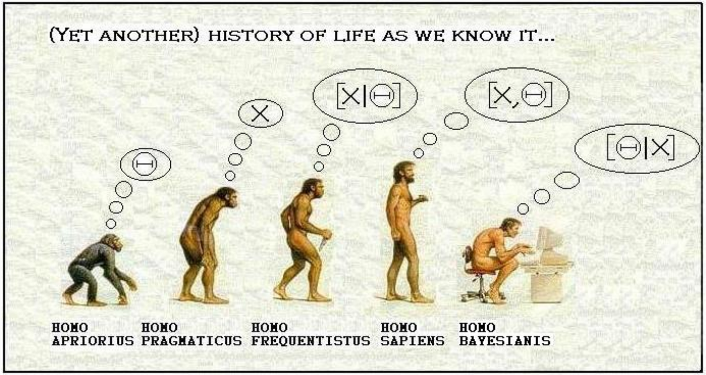
- Frees the modeler in you (M. Kéry)
  - Uses probability to quantify uncertainty for everything (propagation of uncertainty).
  - Allows use of prior information (‘better’ estimates).
  - Can fit complex (hierarchical) models with same MCMC algorithms.

# 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 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.


(YET ANOTHER) HISTORY OF LIFE AS WE KNOW IT...
[Θ]
[X]
[X|Θ]
[X,Θ]
[Θ|X]
HOMO APRIORIUS
HOMO PRAGMATICUS
HOMO FREQUENTISTUS
HOMO SAPIENS
HOMO 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