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.

Your turn

Question

- Come up with examples of clusters or groups

Solution

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.

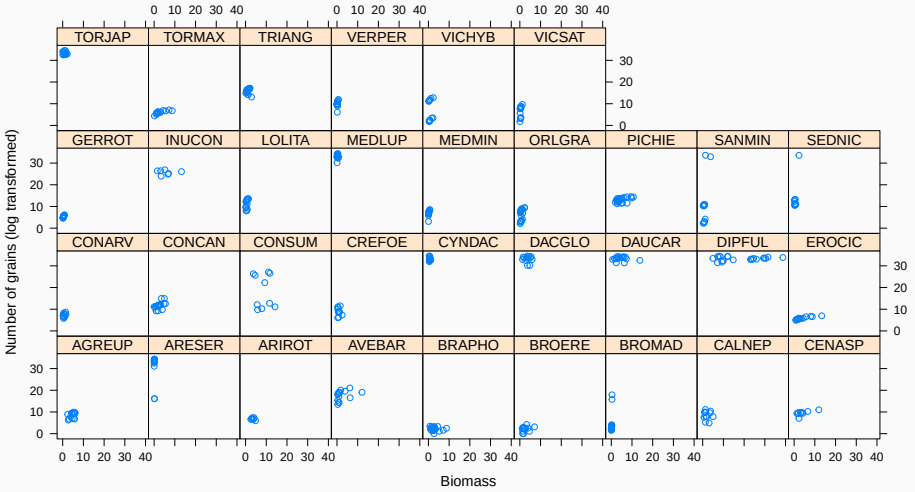
McElreath's explanation of multilevel models

- 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

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

Model with complete pooling. All clusters the same.

GLM with no pooling

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

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

GLMM or GLM with partial pooling

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

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

Back to the plant example

Model with complete pooling (all species are the same)

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

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

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

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

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

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
```

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")
```

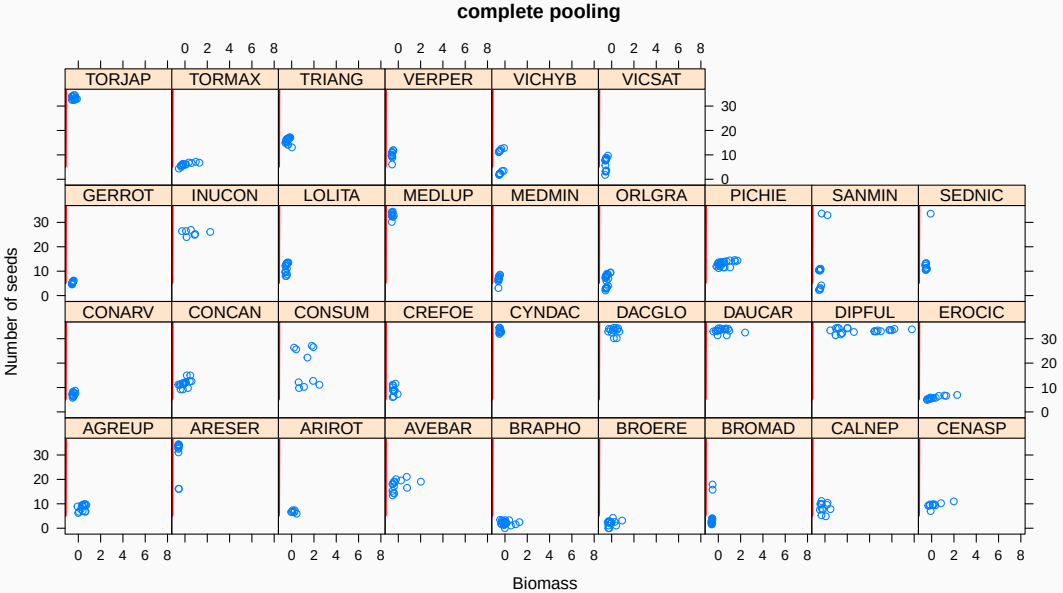
Run Jags

```
allom.1 <- jags(allom.data,
                inits,
                allom.parameters,
                n.iter = 2500,
                model.file = here::here("slides","code","completepooling.bu
                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
```

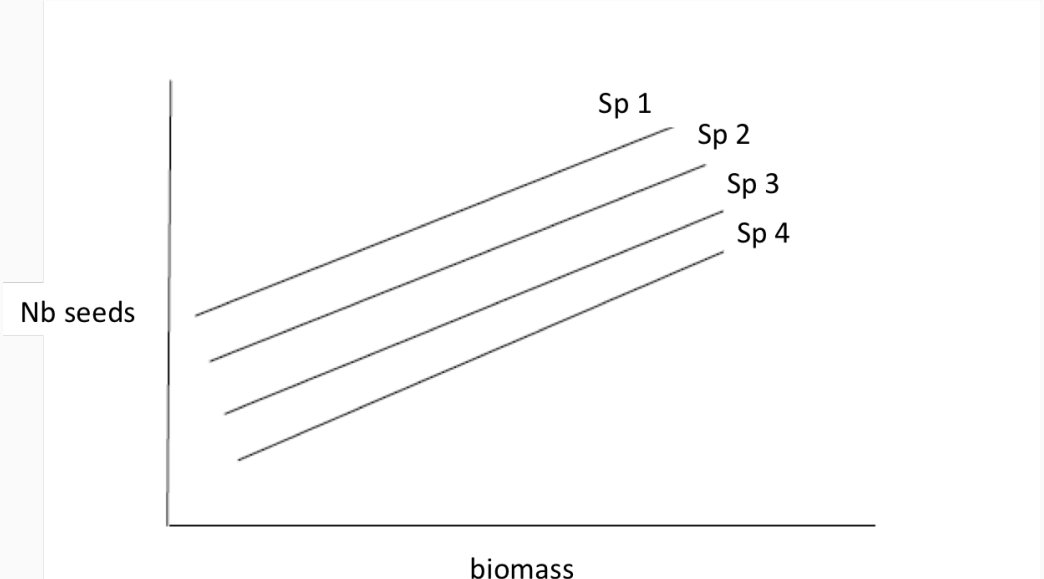
Display results

```
allom.1
#> Inference for Bugs model at "/Users/oliviergimenez/Dropbox/OG/GITHUB/ba
#>  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%
#> a          13.916   0.483   12.980   13.590   13.907   14.240   14.869
#> b           3.584   0.481    2.635    3.252    3.585    3.901    4.553
#> sigma.y    10.437   0.346    9.810   10.192   10.423   10.663   11.179
#> deviance 3672.164   2.480 3669.263 3670.324 3671.527 3673.338 3678.690
#>          n.eff
#> a         2200
#> b         3000
#> sigma.y   3000
#> deviance  3000
```

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
Biomass

Model with partial pooling (species random effect)



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

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

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

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

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

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

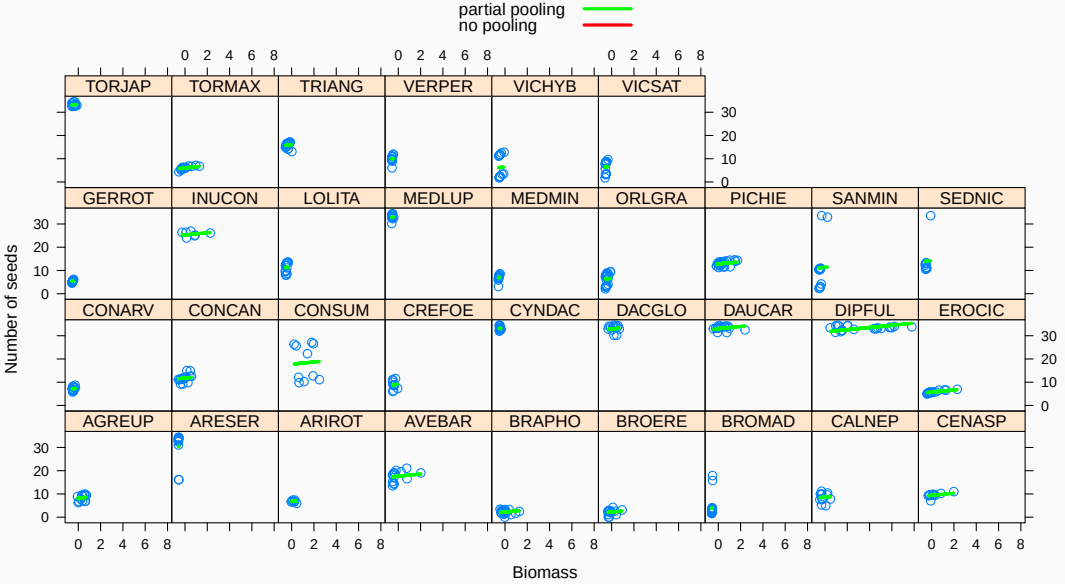
$$\beta \sim \text{Normal}(0, 1000) \quad \text{[prior for slope]}$$

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

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 <- pow(sigma.y, -2)
  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 <- pow(sigma.a, -2)
  sigma.a ~ dunif (0, 100)
  b ~ dnorm (0, 0.001)    }")
writeLines(model,here::here("slides", "code", "varint_bug"))
```

Compare complete pooling vs partial pooling



Model with no pooling (all species unrelated)

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

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

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

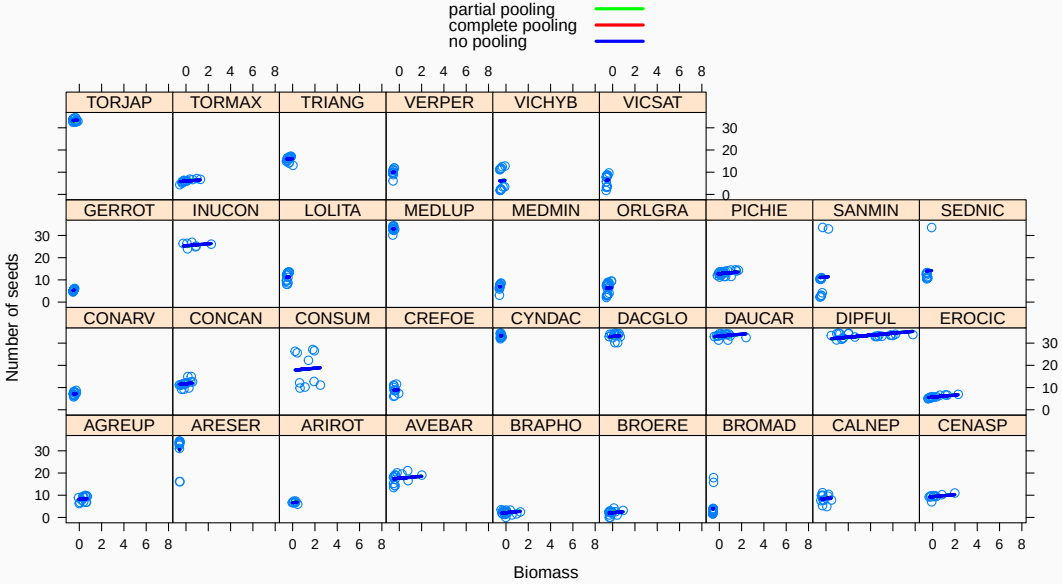
$$\beta \sim \text{Normal}(0, 1000) \quad \text{[prior for slope]}$$

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

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 <- pow(sigma.y, -2)
  sigma.y ~ dunif(0, 100)
  for (j in 1:nbspecies){ a[j] ~ dnorm (0, 0.001)}
  b ~ dnorm (0,0.001)    }")
writeLines(model,here::here("slides","code","nopooling.bug"))
```

Compare complete pooling vs partial pooling vs no pooling



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

Model selection with WAIC

- Consider the plant example. Compare the three models (no, partial and complete pooling) with WAIC.

Solution

R code

WAIC of model with no pooling

```
samples <- jags.samples(model = allom.1$model,
                        variable.names = c("WAIC", "deviance"),
                        type = "mean",
                        n.iter = 2000,
                        n.burnin = 1000,
                        n.thin = 1)
samples$p_waic <- samples$WAIC
samples$waic <- samples$deviance + samples$p_waic
tmp <- sapply(samples, sum)
waic_completepooling <- round(c(waic = tmp[["waic"]], p_waic = tmp[["p_wai
```

WAIC of model with partial pooling

```
samples <- jags.samples(model = allom.2$model,
                        variable.names = c("WAIC","deviance"),
                        type = "mean",
                        n.iter = 2000,
                        n.burnin = 1000,
                        n.thin = 1)
samples$p_waic <- samples$WAIC
samples$waic <- samples$deviance + samples$p_waic
tmp <- sapply(samples, sum)
waic_partialpooling <- round(c(waic = tmp[["waic"]], p_waic = tmp[["p_waic
```

WAIC of model with complete pooling

```
samples <- jags.samples(model = allom.3$model,
                        variable.names = c("WAIC","deviance"),
                        type = "mean",
                        n.iter = 2000,
                        n.burnin = 1000,
                        n.thin = 1)
samples$p_waic <- samples$WAIC
samples$waic <- samples$deviance + samples$p_waic
tmp <- sapply(samples, sum)
waic_nopooling <- round(c(waic = tmp[["waic"]], p_waic = tmp[["p_waic"]]),
```

Model ranking

```
data.frame(model = c('no pooling', 'partial pooling', 'complete pooling'),
           waic = c(waic_nopooling[1],
                    waic_partialpooling[1],
                    waic_completepooling[1]),
           p_waic = c(waic_nopooling[2],
                      waic_partialpooling[2],
                      waic_completepooling[2])) %>%
  arrange(waic)
```

Model ranking

```
#>              model   waic p_waic
#> 1       no pooling 2405.8   65.8
#> 2  partial pooling 2524.4   45.9
#> 3 complete pooling 3674.4    2.4
```

Conclusions

Take-home messages about Bayesian statistics

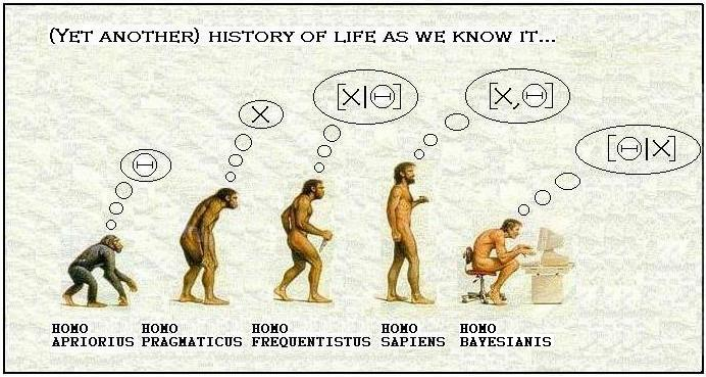
- Frees the modeler in you (M. Kéry)
  - Uses probability to quantify uncertainty for everything (propagation of uncertainty).
  - Allows use of prior information ('better' estimates).
  - Can fit complex (hierarchical) models with same MCMC algorithms.

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...
HOMO APRIORIUS
HOMO PRAGMATICUS
HOMO FREQUENTISTUS
HOMO SAPIENS
HOMO BAYESIANIS

Why become a bayesian? Ask twitter!

Bonus

Longitudinal study on coral reef and GLMM (Poisson)

A survey of a coral reef uses 10 predefined linear transects covered by divers once every week. The response variable of interest is the abundance of a particular species of anemone as a function of water temperature. Counts of anemones are recorded at 20 regular line segments along the transect. The following piece of code will generate a data set with realistic properties according to the above design. Make sure you understand what it is doing. You might want to explain the script to the colleague next to you. Also, to try and make sense of the code of others, it is always good to plot and/or run small sections of the code.

From Jason Matthiopoulos' book.

```r
transects <- 10
data <- NULL
for (tr in 1:transects){
  ref <- rnorm(1,0,.5) # random effect (intercept)
  t <- runif(1, 18,22) + runif(1,-.2,0.2)*1:20 # water temperature gradien
  ans <- exp(ref -14 + 1.8 * t - 0.045 * t^2) # Anemone gradient (expected
  an <- rpois(20, ans) # actual counts on 20 segments of the current trans
  data <- rbind(data,cbind(rep(tr, 20), t, an))
}
```

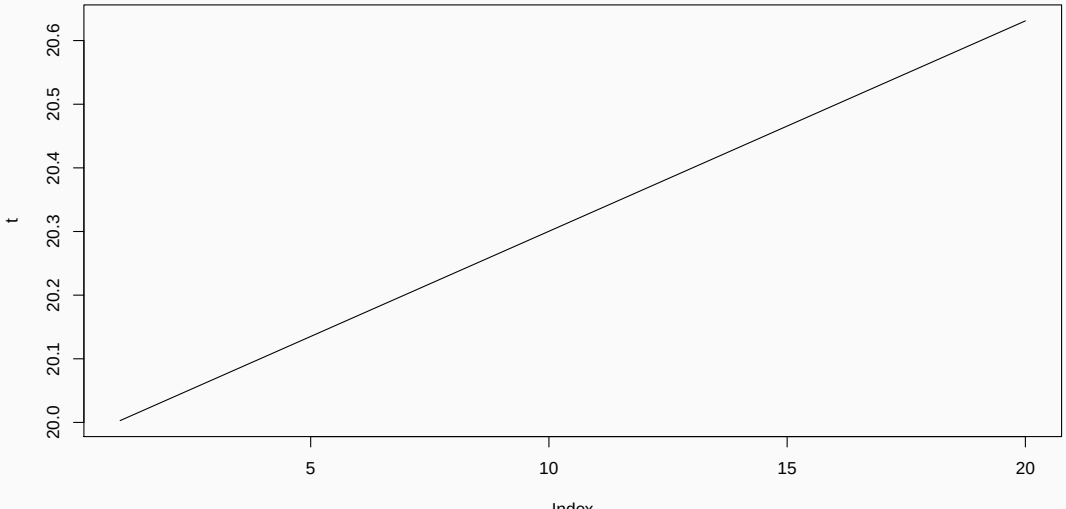
- Generate a data set using the anemone code and fit a GLMM with quadratic effect of temperature and a random intercept.
- Fit the same model to the same data in a Frequentist framework using function `lme4::glmer()`.
- Compare the estimates.

Solution

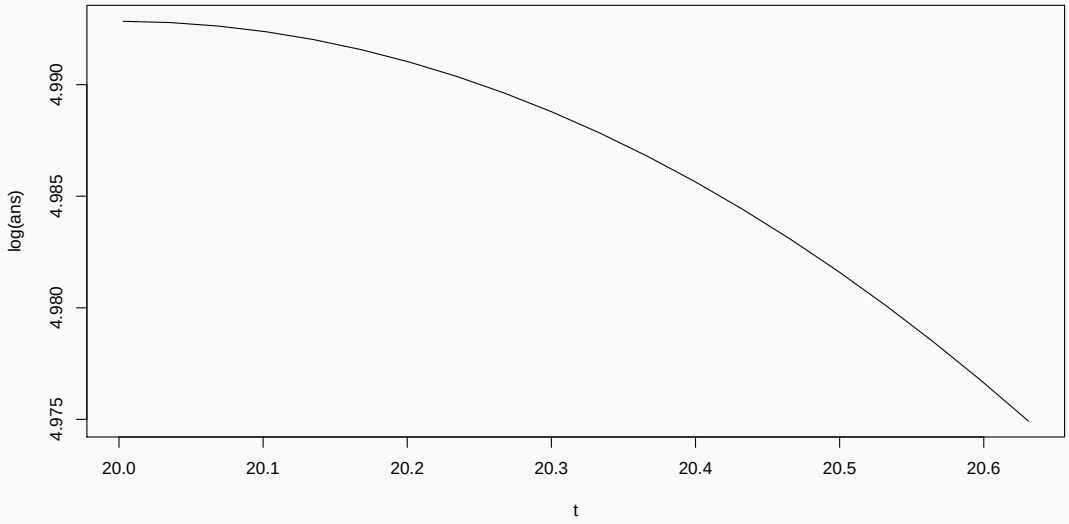
Make sense of the code

- Always difficult to make sense of the code of others.
- Good to plot and/or run small sections of the code.

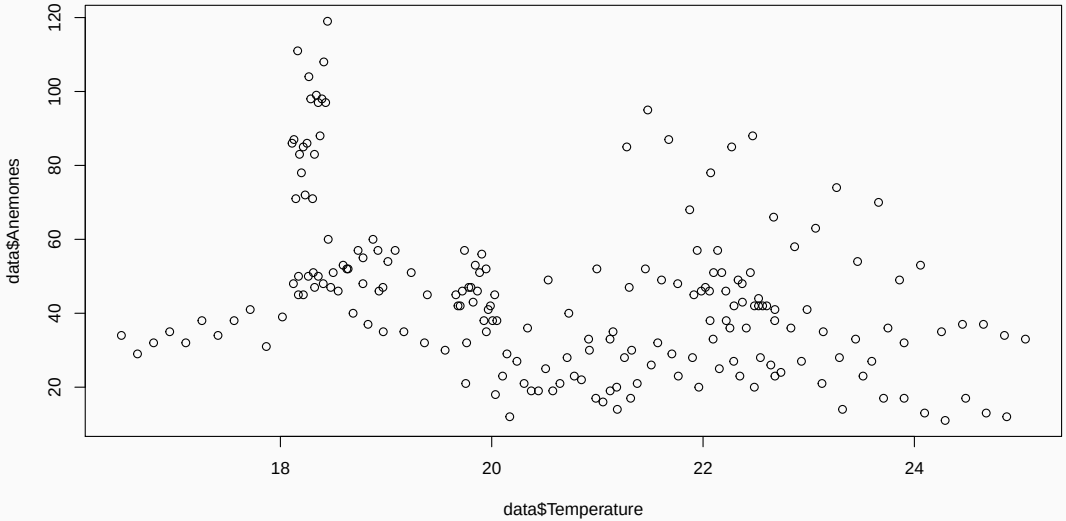
```r
ref <- rnorm(1,0,.5) # random effect (intercept)
t <- runif(1, 18,22) + runif(1,-.2,0.2)*1:20 # water temperature gradient
plot(t,type='l')
```



```r
ans <- exp(ref -14 + 1.8 * t - 0.045 * t^2) # Anemone gradient (expected r
plot(t,log(ans),type='l')
```



```
data <- data.frame(Transect=data[,1],Temperature=data[,2],Anemones=data[,3]
plot(data$Temperature, data$Anemones)
```



Write down model

$$\text{Count}_i \sim \text{Poisson}(\lambda_i) \quad \text{[likelihood]}$$

$$\log(\lambda_i) = a_{\text{TRANSECT[i]}} + b_1\,\text{temp}_i + b_2\,\text{temp}_i^2 \quad \text{[linear model]}$$

$$a_j \sim \text{Normal}(\bar{a}, \sigma) \quad \text{[prior for varying intercepts]}$$

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

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

$$b_1, b_2 \sim \text{Normal}(0, 1000) \quad \text{[prior for slopes]}$$

Standardize Temperature covariate.

```
data$Temp <- (data$Temperature - mean(data$Temperature))/sd(data$Temperatu
head(data)
#>   Transect Temperature Anemones      Temp
#> 1        1    18.44608      119 -1.107961
#> 2        1    18.42843       97 -1.116599
#> 3        1    18.41077      108 -1.125238
#> 4        1    18.39311       98 -1.133877
#> 5        1    18.37545       88 -1.142516
#> 6        1    18.35780       97 -1.151154
```

```
model <-
paste("
model {
  for (i in 1:n){
    count[i] ~ dpois(lambda[i])
    log(lambda[i]) <- a[transect[i]] + b[1] * x[i] + b[2] * pow(x[i],2)
  }
  for (j in 1:nbtransects){
    a[j] ~ dnorm (mu.a, tau.a)
  }
  mu.a ~ dnorm (0, 0.001)
  tau.a <- pow(sigma.a, -2)
  sigma.a ~ dunif (0, 100)
  b[1] ~ dnorm (0, 0.001)
  b[2] ~ dnorm (0, 0.001)
}
```

```
dat <- list(n = nrow(data),
            nbtransects = transects,
            x = data$Temp,
            count = data$Anemones,
            transect = data$Transect)
init1 <- list(a=rnorm(transects), b=rnorm(2), mu.a=rnorm(1), sigma.a=runif
init2 <- list(a=rnorm(transects), b=rnorm(2), mu.a=rnorm(1), sigma.a=runif
inits <- list(init1,init2)
par <- c ("a", "b", "mu.a", "sigma.a")
```

```
fit <- jags(dat, inits, par, n.iter = 2500, model.file=here::here("slides"
#> Compiling model graph
#>    Resolving undeclared variables
#>    Allocating nodes
#> Graph information:
#>    Observed stochastic nodes: 200
#>    Unobserved stochastic nodes: 14
#>    Total graph size: 1622
#>
#> Initializing model
```

```
fit
#> Inference for Bugs model at "/Users/oliviergimenez/Dropbox/OG/GITHUB/ba
#>  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%
#> a[1]        4.659   0.060    4.540    4.619    4.659    4.700    4.774
#> a[2]        3.355   0.059    3.239    3.315    3.356    3.394    3.471
#> a[3]        4.023   0.057    3.910    3.984    4.024    4.062    4.132
#> a[4]        3.089   0.048    2.993    3.055    3.090    3.122    3.181
#> a[5]        3.949   0.072    3.808    3.901    3.951    3.997    4.090
#> a[6]        4.519   0.045    4.433    4.489    4.519    4.549    4.607
#> a[7]        3.987   0.041    3.908    3.959    3.987    4.015    4.066
#> a[8]        3.912   0.043    3.827    3.883    3.913    3.941    3.994
#> a[9]        3.802   0.038    3.725    3.777    3.802    3.827    3.873
#> a[10]       3.472   0.041    3.393    3.445    3.472    3.500    3.551
#> b[1]       -0.092   0.035   -0.161   -0.114   -0.092   -0.068   -0.025
```

```r
library(lme4)
fit_lme4 <- glmer(Anemones ~ Temp + I(Temp^2) + (1 | Transect), data=data,
fit_lme4
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#>   Approximation) [glmerMod]
#>  Family: poisson  ( log )
#> Formula: Anemones ~ Temp + I(Temp^2) + (1 | Transect)
#>    Data: data
#>       AIC       BIC    logLik  deviance  df.resid
#> 1364.4337 1377.6269 -678.2168 1356.4337       196
#> Random effects:
#>  Groups   Name        Std.Dev.
#>  Transect (Intercept) 0.4612
#> Number of obs: 200, groups:  Transect, 10
#> Fixed Effects:
#> (Intercept)        Temp    I(Temp^2)
```

```
visreg::visreg(fit_lme4,xvar='Temp')
```