**http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm**

**Lecture 28—Monday, December 6, 2010**

**Topics**

* [Choice of estimates: REML or maximum likelihood](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm#choice)
* [Centering predictors](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm#centering)
  + [Centering in separate slopes and intercepts models](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm#separate)
  + [Centering in random effects models](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm#random)
* [Adding level-2 predictors to the centered model](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm#adding)
* [Graphing a random slopes and intercepts model](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm#graphing)
* [Bayesian approach to the random slopes and intercepts model](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm#Bayesian)[not done in class]
  + [Modeling the elements of the random effects variance-covariance matrix individually](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm#modeling)
  + [Generalizing the Bayesian model to when there are two or more sets of random effects](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm#generalizing)
* [Multilevel models with three hierarchical levels](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm#multilevel)
  + [The 3-level model without predictors](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm#without)
  + [Fitting 3-level models with the nlme package](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm#nlme)
* [Cited references](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm#cited)
* [R code](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm#Rcode)

**Choice of estimates: REML or maximum likelihood**

REML, which stands for restricted or residual maximum likelihood, is an alternative to full maximum likelihood estimation and is typically the default method in most statistical packages. Rather than maximizing the likelihood of the data, it maximizes the likelihood of the observed residuals. REML obtains initial estimates of the fixed effects using ordinary least squares and then using these estimates it maximizes the likelihood of the residuals (in which the fixed effects are subtracted off) to obtain estimates of the variance parameters. Finally the estimated variance parameters are used to obtain generalized least squares estimates of the fixed effect parameters.

REML is a good alternative to ML when the sole focus is on estimation of the variance components. The variance components obtained via ML are biased when the samples are small while REML estimates are more nearly unbiased. In [lecture 24](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture24.htm#bias) we saw an example of this bias in the maximum likelihood estimate of the residual variance. The unbiased estimate of the variance divides the sum of the squared residuals by *n* – *p*, where *p* is the number of estimated regression parameters, whereas the maximum likelihood estimator divides by *n* yielding an estimate that is too small on average. The removal of the fixed effects from the likelihood means that REML does not "lose" any degrees of freedom by having to estimate the regression parameters and so its estimates of the variance terms are closer to being unbiased.

But there is a down side to REML. Because of the way the regression parameters are handled the "likelihoods" obtained for models with different fixed effects are not comparable. Hence it is not valid to compare models with different fixed effects using a likelihood ratio test or AIC when REML is used to estimate the model. For this reason we have not used REML in this course.

You must be aware of the distinction between ML and REML because REML is the default for most software packages. Thus you will generally need to explicitly specify maximum likelihood estimation if that's what you desire. As we've seen REML is the default for both **lme** and **lmer** and hence we've been using method='ML' and REML=FALSE, respectively. For the models we've been considering the differences between the REML and ML estimates are not large.

inverts <- read.table( 'http://www.unc.edu/courses/2010fall/ecol/563/001/data/lectures/74species.csv', header=T, sep=',')

library(nlme)

**#ML estimates**

out2a.lme <- lme(log(PLD)~log(temp), random=~log(temp)|species, data=inverts, method='ML')

**#REML estimates**

out2a.rlme <- lme(log(PLD)~log(temp), random=~log(temp)|species, data=inverts)

fixef(out2.lme)

(Intercept) log(temp)   
7.088667 -1.454417

fixef(out2a.rlme)

(Intercept) log(temp)   
7.095608 -1.456586

VarCorr(out2a.lme)

species = pdLogChol(log(temp))   
Variance StdDev Corr   
(Intercept) 4.22070369 2.0544351 (Intr)  
log(temp) 0.28109629 0.5301851 -0.92   
Residual 0.02310586 0.1520061

VarCorr(out2.rlme)

species = pdLogChol(log(temp))   
Variance StdDev Corr   
(Intercept) 4.32012538 2.0784911 (Intr)  
log(temp) 0.28947431 0.5380282 -0.92   
Residual 0.02305284 0.1518316

**Centering predictors**

Last time we introduced the concept of centering the level-1 predictor in a multilevel model. By centering I mean subtracting a constant from that predictor. Centering has three beneficial effects in fitting multilevel models.

1. A judicious choice of centering constant can improve model interpretation. After centering the intercept now represents the value of the response at the centering constant.
2. Centering can dramatically reduce the correlation between the slope and intercept random effects. This makes it possible to interpret the magnitudes of one set of random effects separate from the other.
3. Without centering the high amount of correlation between the random effects can lead to numerical instability. As the multilevel model becomes more complex, correlated random effects can prevent the numerical optimization routine from converging to a global solution.

Except for those instances when the uncentered model fails to converge to the global solution, centering has no effect on model fit. The log-lilkelihood is the same whether the model is centered or not. Furthermore there is a one-to-one mapping between the parameter estimates of centered and uncentered models.

**Centering in separate slopes and intercepts models**

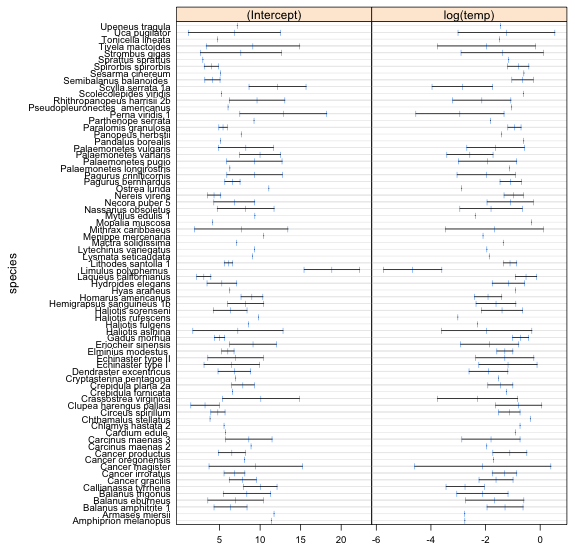
To get a better understanding of what centering does we examine the effect that centering can have on the correlation of slopes and intercepts for the separate slopes and intercepts model. The **nlme** package contains the function **lmList** that can be used to fit separate regression models to observations from different groups. To use it we give the regression formula for the linear model followed by a vertical bar and then the grouping variable as in y ~ x|group. Next we apply the **intervals** function to the **lmList** object that is created to obtain confidence intervals for the estimated coefficients. This can then be fed to the **plot** function which calls the hidden plot method **plot.intervals.lmList** which in turn uses **dotplot** of the **lattice** library to graphically display the confidence intervals of the intercept and slope for each species. Fig. 1 shows the result.

**#fit a separate slopes and intercepts model**

out.lmlist <- lmList(log(PLD)~log(temp)|species, data=inverts)

**#display confidence intervals for individual slopes and intercepts**

plot(intervals(out.lmlist))



**Fig. 1** Confidence intervals for individual intercepts and slopes

There is something rather peculiar about the graph we obtained. Observe that the left and right panels appear to be mirror images of each other. This happens when the parameters in the model are highly correlated. In Fig. 1 the intercepts and slopes are almost perfectly negatively correlated. We can demonstrate this fact by using the **coef** function to extract the coefficients (as the columns of a matrix) and then using the **cor** function to calculate their correlation.

cor(coef(out.lmlist)[,1], coef(out.lmlist)[,2])

[1] -0.936915

Because of the correlation we are not obtaining independent information about the two sets of parameters. For instance to determine whether intercepts vary more than slopes (or vice versa) we first need to make them as uncorrelated as possible. We can do that by choosing an appropriate centering constant for the predictor. By centering I mean subtracting a constant from the value of the predictor, in this case log(temp). Our choice for the constant should be a value that makes the correlation between the intercepts and slopes as close to zero as is possible.

The function center.data defined below takes as its argument a number k, subtracts log(k) from each value of log(temp) thus centering the predictor, and then fits separate straight line models of log(PLD) versus centered log(temp) for each species. The function then calculates the correlation between the intercepts and slopes for the individual models and returns the value of the correlation along with the value of k that was used.

center.data<-function(**k**) {

inverts$k <- k

**#fit separate regression models with centered predictor**

sep.models <- lmList(log(PLD)~ I(log(temp)**-log(k)**)|species, data=inverts)

**#correlation of slopes and intercepts**

corrs <- cor(coef(sep.models)[,1], coef(sep.models)[,2])

c(k,corrs)

}

There is a "bug" of sorts in the **lmList** function. To get the function to work I had to explicitly add the centering constant as a variable to the data frame that is specified in the data argument. (See the first line of the function.) Apparently **lmList** needs all referenced variables to be in the same environment or in the global environment.

To determine what values of k to try, I examine the temperature values that appear in the data set. For k to be interpretable it probably should lie within the range of temperatures in the data set.

range(inverts$temp)

[1] 2.5 35.0

mean(inverts.temp)

18.53922

Because these are preliminary models in which interpretability is not really a goal, I choose a slightly larger range of values for k. I set things up as a for loop. Because log 1 = 0, I start with k = 1 (no centering) and continue to k = 40, incrementing k by 1 each time. Each iteration of the loop runs the center.data function and saves the output from the function.

There are two basic strategies for saving the output from a loop. It can be done incrementally, creating space as needed by appending the new results to the old results, or by creating all the storage space initially before any results are generated and assigning the new values to specified slots in the storage space. For a problem this small, the choice is immaterial, but for larger problems in R the second choice is always the better one and can make a huge difference in speed.

**#Method 1: slower method--create storage space as needed**

**#initialize storage variable**

my.results <- NULL

for(k in 1:40) {

cur.results <- center.data(k)

**#append current result to a matrix of previous results**

my.results <- rbind(my.results, cur.results)

}

**#Method 2: faster method--allocate storage space initially**

**#create matrix of desired dimensions for storage**

my.results<-matrix(NA, ncol=2, nrow=40)

for(k in 1:40) {

**#assign current result to a new row in the storage matrix**

my.results[k,] <- center.data(k)

}

The reason why the first method can be prohibitively slow in big problems is that rbinding, cbinding, or even cing objects during a loop forces R to copy all of the data each time, including the previous rows that have remained unchanged. As the loop progresses the number of rows being copied keeps increasing and becomes the rate-limiting step. In the second method only a single row is copied at each iteration and so there is no cumulative effect on speed.

Checking the results, we see the large negative correlation between slopes and intercepts is gradually decreased (in absolute value) as the centering constant is increased. The correlation first turns positive just after k = 25 and then continues to increase. Fig. 2 displays the results.

plot(my.results[,2]~my.results[,1], type='l', xlab='centering constant, k', ylab='correlation')

abline(h=0,lty=2,col=4)

my.results[abs(my.results[,2])== min(abs(my.results[,2])),]

[1] 25.000000000 -0.008691895

|  |
| --- |
| fig 2 |
| **Fig. 2** The effect of centering on the correlation of slopes and intercepts |

The value of k that comes closest to yielding a correlation of zero is k = 25. Using this as a centering constant, I refit the separate regressions using the **lmList** function and replot the confidence intervals for the estimated intercepts and slopes (Fig. 3). Unlike Fig. 1, the two panels of Fig. 3 no longer resemble each other because the two sets of estimates are uncorrelated.

out.lmlist <- lmList(log(PLD)~I(log(temp)-log(25))|species, data=inverts)

plot(intervals(out.lmlist))



**Fig. 3** Confidence intervals for individual intercepts and slopes after centering

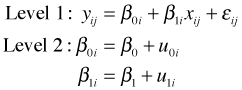
Fig. 3 can be used to suggest a strategy for building a random effects model. In particular we can use it to suggest which level-1 parameters should be allowed to be random at level 2. Because random effects account for unexplained heterogeneity, parameters that are prime candidates for random effects are those that vary greatly across individual species.

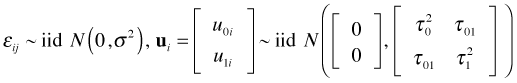
* From Fig. 3 we can see that both intercepts and slopes vary considerably, but intercepts vary more than slopes. I base this statement on the fact that the confidence intervals for the intercepts are much narrower than the confidence intervals for the slopes and overlap each other less. A formal statistical test would find fewer species with significantly different slopes than intercepts.
* Keep in mind Fig. 3 is guilty of some serious overfitting of the data. As we saw previously about two thirds of the depicted linear relationships in Fig. 3 are based on only three data points. For that reason we should only use this graph as a rough guideline on how to proceed.

So, if we had to choose between modeling intercepts or modeling slopes we should probably choose to model intercepts because there appears to be more variability to explain there. If there are no species-level predictors available then we should at least include species-level random effects in the level-2 intercept equation. Of course, this is what we would normally have done. In a linear model the natural progression is to allow intercepts to vary randomly at level 2 and then, if necessary, allow slopes to vary randomly too.

**Centering in random effects models**

It's worth noting that the choice of centering constant that causes the slopes and intercepts to be uncorrelated in individual regression models is not necessarily the same choice of centering constant that makes slope and intercept random effects uncorrelated. One of the reasons for this is that in the separate slopes and intercepts model all species are treated equally, whereas in the random effects version of the model the correlation is a weighted estimate. The basic random slopes and intercepts model is shown below.





The term tau01is the covariance of the two sets of random effects. While it's generally not necessary to have the random effects uncorrelated, it can happen that models in which the random effects are highly correlated will fail to converge, or worse, converge to a false solution. Thus it is advantageous to center the predictors in mixed effects models too. Typically we choose a centering constant that both reduces the correlation (although not necessarily to zero) and makes the intercept interpretable.

The code below modifies the centering function we used above to now fit a linear mixed effects model to the invertebrates data and then extract the correlation of the random effects.

**#lme centering function**

center.data2 <- function(k) {

inverts$k <- k

lme.models <- lme(log(PLD)~I(log(temp)-log(k)), random=~I(log(temp)-log(k))|species, data=inverts)

c(k,as.numeric(VarCorr(lme.models)[2,3]))

}

Here are a few comments on the modifications I made to the centering function.

* I use the **VarCorr** function to extract the correlation, not the covariance, of the random effects. The correlation occupies the (2,3) position of a character matrix so I use the **as.numeric** function to convert it to a number.
* The same bug that plagues the **lmList** function is present in the **lme** function. To get the model to run for different *k* I had to explicitly add the centering constant as a variable to the data frame that is specified in the data argument. (See the first line of the function.)

**#repeat for a range of centering constants**

my.results2 <- matrix(NA, ncol=2, nrow=40)

for(k in 1:40) {

my.results2[k,] <- center.data2(k)

}

**#display results graphically**

plot(my.results2[,2]~ my.results2[,1], type='l', xlab='centering constant, k', ylab='correlation')

abline(h=0, lty=2, col=4)

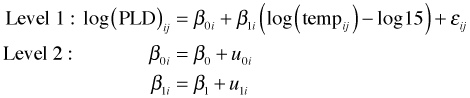
**#locate smallest correlation**

my.results2[abs(my.results2[,2])== min(abs(my.results2[,2])),]

[1] 35 0

|  |
| --- |
| fig 4 |
| **Fig. 4** The effect of centering on the correlation of random slopes and random intercepts |

So *k* = 35 can be used to make the slope and intercept random effects uncorrelated. O'Connor *et al.* (2007) elected to use *k* = 15 as their centering constant because it reduced the correlation enough to prevent convergence problems while still being a value within the range of the data.



Recall that the intercept in a linear model corresponds to the value of the response variable when all the predictors are set to zero. Setting the single predictor in their model, log(temp) – log15, to zero yields

http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/centered%20intercept.gif

Thus in their published model the intercept can be interpreted as the value of log(PLD) when the temperature is 15°C.

out2.lme<-lme(log(PLD)~I(log(temp)-log(15)), random=~I(log(temp)-log(15))|species, data=inverts, method='ML')

VarCorr(out2.lme)

species = pdLogChol(I(log(temp) - log(15)))   
Variance StdDev Corr   
(Intercept) 0.85672249 0.9255931 (Intr)  
I(log(temp) - log(15)) 0.28109622 0.5301851 -0.49   
Residual 0.02310587 0.1520062

sapply(list(out2a.lme,out2.lme),logLik)

[1] -99.21859 -99.21859

fixef(out2a.lme)

(Intercept) log(temp)   
7.088667 -1.454417

fixef(out2.lme)

(Intercept) I(log(temp) - log(15))   
3.150033 -1.454417

As advertised, the model fit has not changed. Observe also that the correlation between the random effects has been cut in half to –0.49. The estimated slopes are the same in the centered and uncentered models.

**Adding level-2 predictors to the centered model**

With this background I refit the three climate models but this time using the centered log(temp) variable.

**#add climate.3 as a predictor of the intercepts**

out3.lme <- lme(log(PLD)~I(log(temp)-log(15)) + climate.3, random=~I(log(temp)-log(15))|species, data=inverts, method='ML')

**#test significance of climate.3**

anova(out3.lme,out2.lme)

Model df AIC BIC logLik Test L.Ratio p-value  
out3.lme 1 8 213.8831 240.9590 -98.94154   
out2.lme 2 6 210.4372 230.7441 -99.21859 1 vs 2 0.554096 0.758

So we conclude the intercepts do not vary significantly by **climate.3**.

**#add climate.3 as a predictor of the slopes**

out4.lme <- lme(log(PLD)~I(log(temp)-log(15)) + I(log(temp)-log(15)):climate.3, random=~I(log(temp)-log(15))|species, data=inverts, method='ML')

**#test significance of climate.3**

anova(out4.lme,out2.lme)

Model df AIC BIC logLik Test L.Ratio p-value  
out4.lme 1 8 206.7896 233.8655 -95.39479   
out2.lme 2 6 210.4372 230.7441 -99.21859 1 vs 2 7.647598 0.0218

So the slopes do vary significantly by **climate.3**. Finally I let **climate.3** be a predictor of both slopes and intercepts.

**#add climate.3 as a predictor of slopes and intercepts**

out5.lme <- lme(log(PLD)~I(log(temp)-log(15)) + I(log(temp)-log(15))\*climate.3, random=~I(log(temp)-log(15))|species, data=inverts, method='ML')

At this point there are a number of tests possible. We can do a global test of whether we need **climate.3** in the model in some capacity, either as a predictor of the slopes or as a predictor of the intercepts.

anova(out5.lme,out2.lme)

Model df AIC BIC logLik Test L.Ratio p-value  
out5.lme 1 10 206.4266 240.2715 -93.21330   
out2.lme 2 6 210.4372 230.7441 -99.21859 1 vs 2 12.01058 0.0173

The answer is yes. If **climate.3** is a predictor of the intercepts do we also need it to be a predictor of the slopes?

anova(out5.lme,out3.lme)

Model df AIC BIC logLik Test L.Ratio p-value  
out5.lme 1 10 206.4266 240.2715 -93.21330   
out3.lme 2 8 213.8831 240.9590 -98.94154 1 vs 2 11.45648 0.0033

The answer is again yes. Finally if **climate.3** is a predictor of the slopes do we also need it as a predictor of the intercepts?

anova(out5.lme,out4.lme)

Model df AIC BIC logLik Test L.Ratio p-value  
out5.lme 1 10 206.4266 240.2715 -93.21330   
out4.lme 2 8 206.7896 233.8655 -95.39479 1 vs 2 4.362981 0.1129

The answer is no. So, the results are fairly clear cut; **climate.3** modifies the slopes but not the intercepts.

sapply(list(out2.lme, out3.lme, out4.lme, out5.lme), AIC)

[1] 210.4372 213.8831 206.7896 206.4266

Based on AIC a model in which **climate.3** affects both the slopes and intercepts (out5.lme) is not distinguishable from a model in which **climate.3** only affects the slope (out4.lme), but both models are an improvement over one in which **climate.3** affects only the intercepts (out3.lme) or a model in which **climate.3** is completely absent (out2.lme).

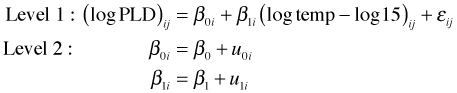
summary(out4.lme)$tTable

Value Std.Error DF t-value p-value  
(Intercept) 3.1724449 0.1122258 141 28.268402 5.921156e-60  
I(log(temp) - log(15)) -1.0277108 0.1672332 141 -6.145376 7.715916e-09  
I(log(temp) - log(15)):climate.3temperate -0.4736086 0.1865685 141 -2.538524 1.221795e-02  
I(log(temp) - log(15)):climate.3tropical -0.7131552 0.2704806 141 -2.636622 9.312393e-03

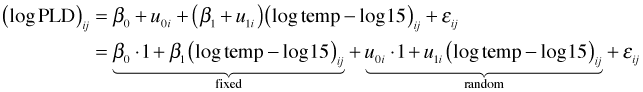
Slopes are negative in all three geographic regions but are significantly more negative in temperate and tropical regions than they are in polar regions.

**Graphing a random slopes and intercepts model**

In this section we work with the basic random slopes & intercepts model without any level-2 predictors, the model out2.lme above. Formally, this is the following model written in multilevel form.



or in composite form



A multilevel model returns estimates at a number of different levels.

* It yields what's called the population-average (marginal) model, a model that describes the general level-1 pattern of all the level-2 units. This is the model constructed from the fixed effects alone.
* It also yields the subject-specific model. This is a model that describes the behavior of individual level-2 units. It incorporates both fixed effects and random effects in the parameter estimates.

The **nlme** package has three functions that are useful in extracting the quantities needed to construct these models: **fixef**, **ranef**, and **coef**.

* **fixef** returns the fixed effect parameter estimates that define the population-average model. For model 2 above these are β0 and β1.

fixef(out2.lme)

(Intercept) I(log(temp)-log(15))   
7.088667 -1.454417

* **ranef** returns a matrix of random effect predictions for each level 2 unit. These are http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/u0i.gifand http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/u1i.gifwith each row of the matrix returned by **ranef** corresponding to the random effect predictions for a different level-2 unit. For our model there are 74 of these, one for each species.

dim(ranef(out2.lme))

[1] 74 2

ranef(out2.lme)[1:4,]

(Intercept) I(log(temp)-log(15))  
Amphiprion melanopus 0.3383047 -0.09451435  
Armases miersii 1.2984135 -0.28956246  
Balanus amphitrite 1 -0.6182210 0.13257574  
Balanus eburneus -0.6271444 -0.04302652

* **coef** returns a matrix of subject-specific parameters. For model 2 above these are http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/subjectspecific1.gifand http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/subjectspecific2.gif, each row in the matrix corresponding to a different level-2 unit. Each entry shown below is just the random effect from the output table above plus the corresponding fixed effect.

coef(out2.lme)[1:4,]

(Intercept) I(log(temp)-log(15))  
Amphiprion melanopus 7.426971 -1.548931  
Armases miersii 8.387080 -1.743979  
Balanus amphitrite 1 6.470446 -1.321841  
Balanus eburneus 6.461522 -1.497443

To understand the relationship between the population-average and subject-specific models it is worthwhile to also consider the individual ordinary least squares (OLS) regression models, what we've previously called the separate regressions model. The separate regressions model is what we would obtain if we fit the level-1 model separately to each level-2 unit.

To compare these three models individually for each species I customize a lattice graph for this purpose. I begin by collecting the various parameter estimates from the multilevel model output and assembling them along with the raw data in a new data frame.

subj.specific.parms <- coef(out2.lme)

pop.average.parms <- fixef(out2.lme)

**#create data frame of subject specific estimates and species names**

ran.data <- data.frame(rownames(subj.specific.parms), subj.specific.parms)

colnames(ran.data) <- c('species', 'int', 'slope')

**#add model estimates to the raw data**

inverts2 <- merge(inverts, ran.data)

Here's what the first part of the new data frame looks like after the merge.

inverts2[1:6,]

species meantemp temp PLD taxon climate.3 feeding.type lntemp lnPLD  
1 Amphiprion melanopus 26.5 25.0 12.3 Chordata tropical P 3.218876 2.509599  
2 Amphiprion melanopus 26.5 28.0 9.0 Chordata tropical P 3.332205 2.197225  
3 Armases miersii 26.6 23.9 19.3 Arthropoda tropical L 3.173878 2.960105  
4 Armases miersii 26.6 29.3 11.0 Arthropoda tropical L 3.377588 2.397895  
5 Balanus amphitrite 1 22 15.0 17.0 Arthropoda temperate P 2.708050 2.833213  
6 Balanus amphitrite 1 22 20.0 14.0 Arthropoda temperate P 2.995732 2.639057  
n PLD.15c lnPLD.15c X1000.k lograte int slope  
1 2 50.28975 3.917801 3.355705 -0.8943831 3.232388 -1.548931  
2 2 50.28975 3.917801 3.322259 -0.5820084 3.232388 -1.548931  
3 2 69.80030 4.245638 3.368137 -1.0170519 3.664297 -1.743980  
4 2 69.80030 4.245638 3.307972 -0.4548421 3.664297 -1.743980  
5 5 17.73829 2.875725 3.472222 -2.2600730 2.890834 -1.321841  
6 5 17.73829 2.875725 3.412969 -2.0659170 2.890834 -1.321841

Notice that the subject-specific coefficient estimates, int and slope, appear as new columns. When a species has more than one observation (level-1 unit) the same values of the subject-specific estimates are repeated in each of the rows. When we produce the panel graphs we will need to pull off only one of these values.

The **xyplot** call shown below plots the OLS line, the subject-specific line, and the population-average line for each level 2 unit. Because there are too many species (74) to display all at once, I plot nine at a time and use the **ask** argument of **par** to pause after each graph of nine panels before drawing the next graph. To get the subject-specific lines to plot we need to use the **subscripts** argument in the panel function. When the panel function is called upon to draw a panel the variable **subscripts** contains the appropriate value of species (the variable that defines the panels) for the current panel. As a result the expressions inverts2$int[**subscripts**]and inverts2$slope[**subscripts**] extract a vector of intercepts and a vector of slopes for the species that is currently being plotted.

For instance when **subscripts** references the first species, *Amphiprion melanopus*, the expression inverts2$int[**subscripts**] will yield the vector c(7.426791, 7.426791). (Verify this from the display of the first six rows of the inverts2 data frame above.) As a by-product of the merge, the intercept value was duplicated so that it appears in every level-1 observation coming from the same level-2 unit. Because we need only one intercept value to plot a line with the **abline** function I discard the duplicated values and retain only the first one by specifying inverts2$int[**subscripts**][1].

The code for the function is shown below.

library(lattice)

par(ask=T)

xyplot(log(PLD)~I(log(temp)-log(15))|species, data=inverts2, layout=c(3,3), xlab=expression(log(temperature/15)), ylab=expression(log(PLD)), subscripts=T, panel=function(x, y, **subscripts**) {

**#raw data**

panel.xyplot(x, y, pch=16)

**#separate regressions OLS line**

panel.lmline(x, y, lty=1, col=4)

**#population-average line**

panel.abline(c(pop.average.parms[1], pop.average.parms[2]), lty=1, lwd=2, col=2)

**#subject-specific line**

panel.abline(c(inverts2$int[**subscripts**][1], inverts2$slope[**subscripts**][1]), lty=2)

},

strip = strip.custom(par.strip.text = list(cex = 0.75)), par.settings = list(axis.text = list(cex = 0.7)), key=list(lines=list(lty=c(1,1,2), col=c(4,2,1)), text=list(c("OLS fitted trajectory", "population average trajectory", "model-based empirical Bayes trajectory"), cex=rep(.85,3)), border=0, space='top'))

par(ask=F)

Fig. 1 shows the first two sets of nine species.

|  |  |
| --- | --- |
| fig 5a | fig 5b |
| **Fig. 5** Fitted trajectories for the first 18 species | |

The empirical Bayes trajectory is the subject-specific model, i.e., the model that includes both fixed and random effects. The empirical Bayes trajectory is a weighted sum of the population-average trajectory and the individual OLS trajectory. The weights are determined by the amount of data available for this subject and the reliability of the OLS model. If there are few data values and/or they are very close together, the empirical Bayes line will more closely resemble the population-average line. If there is a lot of data and/or the data values are spread out, then the empirical Bayes estimate will more closely resemble the OLS line. This is how the random effects model avoids the problem of overfitting that plagues the separate regressions model.

For example, in the right graph of Fig. 5 we see that the subject-specific model for *Circeus spirillum* is nearly indistinguishable from the separate regressions OLS line. For this species there are three widely-spaced observations. Thus the individual regression line we obtain is a reliable estimate of that species' trend and is a better estimate for that species than is the population-average line.

On the other hand consider *Chthamalus stellatus* in the right graph where we see that the subject-specific line is nearly identical to the population-average line and very different from the OLS line. Here we have only two observations with nearly identical values of the predictor. A line constructed from two closely-spaced points does not provide a reliable estimate of trend over the full range of the data. As a result here the population-average model should be preferred over the separate regressions (OLS) model.

This also explains why a multilevel model can provide subject-specific estimates even for subjects with so few observations that an individual OLS model cannot be fit. For these subjects the subject-specific model will be very close to the population-average model. In fact a multilevel model can estimate a regression line for a level-2 unit that has only a single level-1 observation.

What follows is a brief line-by-line description of the code that produces the individual panel graphs.

|  |  |
| --- | --- |
| log(PLD)~I(log(temp)-log(15))|species | This defines the plotting variables and the grouping variable (species) |

|  |  |
| --- | --- |
| layout=c(3,3) | Sets up a tableau of 3 rows and 3 columns of plots |

|  |  |
| --- | --- |
| data=inverts2 | Data set containing variables to be plotted |

|  |  |
| --- | --- |
| xlab=expression(log(temperature/15)), ylab=expression(log(PLD)) | Labels for the *x*- and *y*-axes. Here **expression** is used to display mathematical notation. Note: http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/model1.gif |

|  |  |
| --- | --- |
| subscripts=T | Makes subscripts available to subsequent functions. This is actually not needed here. |

|  |  |
| --- | --- |
| panel=function(x,y,subscripts){ | Begins definition of panel function for defining what is displayed in each panel. The **subscripts** argument references the identifier of each panel and is needed here in order to add the individual empirical Bayes trajectories |

|  |  |
| --- | --- |
| panel.xyplot(x,y,pch=16) | Plots individual points |

|  |  |
| --- | --- |
| panel.lmline(x,y,lty=1,col=4) | Adds OLS line |

|  |  |
| --- | --- |
| panel.abline(c(pop.average.parms[1], pop.average.parms[2]), lty=1, lwd=2, col=2) | Adds population-average line |

|  |  |
| --- | --- |
| panel.abline(c(inverts2$int [subscripts][1], inverts2$slope [subscripts][1]),lty=2) | Adds subject-specific lines. **subscripts** extracts a vector of int and slope values correct for the observations being plotted in the current panel and **[1]** selects only the first element of each of these vectors (because **panel.abline** needs only a single intercept and slope value). |

|  |  |
| --- | --- |
| strip = strip.custom(par.strip.text = list(cex = 0.75)) | Adjusts size of text in the strips above each plot (the species names) |

|  |  |
| --- | --- |
| par.settings = list(axis.text = list(cex = 0.7)) | Adjusts the size of numbers on axes |

|  |  |
| --- | --- |
| key=list(lines=list(lty=c(1,1,2), col=c(4,2,1)), text=list(c("OLS fitted trajectory", "population average trajectory", "model-based empirical Bayes trajectory"), cex=rep(.85,3)), border=0, space='top') | Creates legend to label lines appearing in plots |

**Bayesian approach to the random slopes and intercepts model**

**Modeling the elements of the random effects variance-covariance matrix individually**

The major change required in moving to a random intercepts and slopes model is that the probability model for these parameters is a multivariate normal distribution. This necessitates making the intercepts and slopes the columns of a matrix. In the code below this matrix is denoted by **B**.

* The code in **blue** with the yellow background indicates the change in going from a constant slope model to one in which the slopes vary by species.
* The code in **red** sets up the multivariate normal probability model for the matrix of random effects. The intercepts occupy column 1 of this matrix while the slopes occupy column 2.
* The code in **blue** formulates uninformative priors for the variances and covariances of the random effects. Observe that to go from a precision matrix (as required for the multivariate normal distribution in WinBUGS) to a variance matrix we have to calculate a matrix inverse (first blue line). The covariance matrix for the random effects is denoted **Sigma.B** in the code below. (Note: I use lower case letters to denote entries of this matrix. So **sigma.b** is a scalar, while **Sigma.B** is a matrix.)

Because the response and predictor are on a log scale the priors do not need to be as diffuse as we would normally use for raw data.

|  |  |
| --- | --- |
| **Random intercepts** | **Random slopes and intercepts (randomslopeint.txt)** |
| model{ for(i in 1:n) {  y[i]~dnorm(y.hat[i],tau.y)  y.hat[i]<-a[species[i]]+ **b**\*x[i]  } tau.y<-pow(sigma.y,-2) sigma.y~dunif(0,100) for(j in 1:J){ **a[j]~dnorm(a.hat[j],tau.a)** a.hat[j]<-mu.a} **b~dnorm(0,.0001)** mu.a~dnorm(0,.0001) **tau.a<-pow(sigma.a,-2) sigma.a~dunif(0,100)** } | model{ for(i in 1:n) {  y[i]~dnorm(y.hat[i],tau.y)  y.hat[i]<-a[species[i]]+ **b[species[i]]**\*x[i]  }  tau.y<-pow(sigma.y,-2) sigma.y~dunif(0,100) **#matrix of random slopes and intercepts** for(j in 1:J){  **a[j]<-B[j,1]**  **b[j]<-B[j,2]**  **B[j,1:2]~dmnorm(B.hat[j,], Tau.B[,])**  **B.hat[j,1]<-mu.a**  **B.hat[j,2]<-mu.b**  } mu.a~dnorm(0,.0001) mu.b~dnorm(0,.0001)  **Tau.B[1:2,1:2]<- inverse(Sigma.B[,])**  **#variances of the random effects** **Sigma.B[1,1]<-pow(sigma.a,2) sigma.a~dunif(0,100) Sigma.B[2,2]<-pow(sigma.b,2) sigma.b~dunif(0,100) #correlation**  **Sigma.B[1,2]<-rho\*sigma.a\*sigma.b**  **Sigma.B[2,1]<-Sigma.B[1,2] rho~dunif(-1,1)** } |

Here's how we would run this model. The biggest change in addition to the need for initial values for **sigma.b** and **rho** is that the initial values for the intercepts and slopes must be organized in a matrix. The declaration B=array(rnorm(2\*J),c(J,2)) below sets up a matrix with J rows and 2 columns populated with random normal initial values. We track the additional parameters **mu.b**, **sigma.b**, and **rho** by placing them in the parameters list.

x <- log(inverts$temp)-log(15)

y <- log(inverts$PLD)

n <- length(y)

J <- length(unique(inverts$species))

species <- as.numeric(inverts$species)

library(arm) **#for WinBUGS**

library(R2jags) **#for JAGS**

invert.data <- list("n", "J", "y", "species", "x")

invert.inits <- function() { list(**B=array(rnorm(2\*J),c(J,2))**, sigma.y=runif(1), mu.a=rnorm(1), sigma.a=runif(1), mu.b=rnorm(1), sigma.b=runif(1), **rho=runif(1)**)}

invert.parameters <- c("B", "sigma.y", "mu.a", "sigma.a", "mu.b", "sigma.b", "rho")

invert.5a <- bugs(invert.data, invert.inits, invert.parameters, "randomslopeint.txt", n.chains=3, n.iter=10, debug=T)

invert.5a <- bugs(invert.data, invert.inits, invert.parameters, "randomslopeint.txt", n.chains=3, n.iter=5000, debug=T)

max(invert.5a$summary[,"Rhat"])

[1] 1.014

min((invert.5a$summary[,"n.eff"])

[1] 140

The diagnostics look good so I carry out the final run of the model. Recall that in this model the random intercepts and slopes needed to be organized in a 74 × 2 matrix. So far we've been returning the entire **B** matrix so that we could then use the **last.values** of the Markov chains as the initial values for the next run. This organization turns out to be rather awkward for final use as we can see below. The intercepts and slopes are interdigitated and need to be differentiated by which column of the **B** matrix is being displayed

invert.5a$summary[1:4,]

mean sd 2.5% 25% 50% 75% 97.5% Rhat n.eff  
B[1,1] 3.233 0.3093 2.615 3.032 3.229 3.448 3.8330 1.003 580  
B[1,2] -1.548 0.5116 -2.548 -1.873 -1.530 -1.224 -0.5228 1.003 580  
B[2,1] 3.660 0.3013 3.078 3.449 3.659 3.858 4.2587 1.001 1100  
B[2,2] -1.738 0.4957 -2.700 -2.075 -1.736 -1.400 -0.8024 1.002 1000

A more useful presentation is to have the intercepts and slopes grouped separately. This can easily be arranged because in the BUGS model definition file I've pulled off the intercepts and slopes from the matrix as the vectors **a** and **b**. So in the final run I include "a" and "b" in the list of parameters to be returned, rather than the matrix **B**.

**#In final run return intercepts and slopes separately**

invert.parameters2 <- c("a", "b", "sigma.y", "mu.a", "sigma.a", "mu.b", "sigma.b", "rho")

invert.5b <- bugs(invert.data, invert.5a$last.values, invert.parameters2, "randomslopeint.txt", n.chains=3, n.iter=5100, n.burnin=100, debug=T)

invert.5b$summary[149:155,]

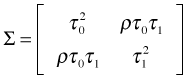
mean sd 2.5% 25% 50% 75% 97.5% Rhat n.eff  
sigma.y 0.155 0.0133 0.132 0.146 0.154 0.164 0.183 1.00 410  
mu.a 3.148 0.1128 2.926 3.071 3.148 3.222 3.375 1.00 650  
sigma.a 0.949 0.0845 0.805 0.889 0.939 1.005 1.127 1.00 680  
mu.b -1.452 0.0903 -1.635 -1.510 -1.451 -1.387 -1.281 1.01 1100  
sigma.b 0.540 0.0851 0.382 0.485 0.539 0.597 0.707 1.01 320  
rho -0.468 0.1208 -0.688 -0.556 -0.474 -0.392 -0.217 1.00 1100  
deviance -195.027 29.9842 -251.130 -215.700 -195.900 -174.200 -133.310 1.00 600

The following table compares the Bayesian estimates with what was obtained with **lme**.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Frequentist estimate** | **Bayesian estimate  (mean of posterior distribution)** |
| http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/beta0.gif | 3.150 | 3.1475 |
| http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/beta1.gif | –1.454 | –1.4517 |
| http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/tau0.gif | 0.9256 | 0.9491 |
| http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/tau1.gif | 0.5302 | 0.5404 |
| http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/sigmasmall.gif | 0.1520 | 0.1555 |
| http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/rho.gif | –0.49 | –0.4675 |

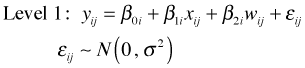
**Generalizing the Bayesian model to when there are two or more sets of random effects**

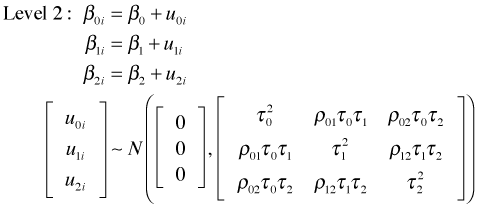
The random slopes & intercepts model is a model in which there are two sets of random effects, random slopes and intercepts. The covariance matrix for the random intercepts and slopes is the following.



The BUGS code we used to fit this model is somewhat involved because of the matrix notation, but otherwise is a fairly straight-forward extension of the simpler models we've considered. What made things easy was that we were able to specify priors for all the components of Σ individually and independently of the remaining components.

When we switch to a model with three or more random effects, the situation becomes far more complicated and the approach we used with two random effects does not work at all. To illustrate the problems that arise I consider a generic model with three sets of random effects. Naturally for there to be three sets of random effects we must have at least three regression parameters in the level-1 equation.





Much of the BUGS code needed to fit this model would be fairly standard. Where things get sticky is in setting the non-informative priors for the six parameters of the 3 × 3 covariance matrix of the random effects. The reason this is difficult is that it is no longer possible to do it one parameter at a time. Covariance matrices are not just random collections of numbers. In particular, a covariance matrix must be positive semi-definite something that is a property of the matrix as a whole rather than a property of its individual elements.

Consequently it is necessary to set a non-informative prior on the matrix itself rather than on its individual elements. A probability distribution that is often used as a prior for covariance matrices is the inverse Wishart distribution. The difficulty in using this distribution is in choosing values for its parameters that yield truly non-informative priors for the individual covariance elements. For instance the settings that essentially place uniform priors on the correlations tend to place priors on the variances that are too constraining. One proposed remedy is to use a slight modification of the inverse Wishart distribution called the scaled inverse Wishart distribution (Gelman and Hill, 2007).

Using an inverse Wishart prior for a covariance matrix is equivalent to using a Wishart prior for the precision matrix. The Wishart distribution has two parameters called the scale Δ and the degrees of freedom ν.

1. The degrees of freedom ν corresponds roughly to the prior sample size. Thus an inverse Wishart prior with degrees of freedom ν weights a prior mean for the covariance matrix as strongly as the information contained in a sample of ν additional observations. So to guarantee an uninformative prior we need to make this value as small as possible, but choosing the value too small will cause numerical problems in matrix inversion. For a *K* × *K* precision matrix the smallest legal value for the ν is *K*. The general recommendation in the literature is to use ν *= K* + 1 as this leads to a less restrictive prior on the correlations.
2. The scale parameter Δ of the Wishart distribution is a matrix **W**. To obtain a non-informative prior the usual recommendation is to choose **W** to be diagonal where the magnitude of the diagonal entries reflects the expected magnitudes of the variance components. For scaled data **W** can be taken to be the identity matrix but for other cases we need prior estimates of the intercept and slope variances. One option is to use the variances calculated using the slopes and intercepts from the separate slopes and intercepts model.

The table below gives the matrix version of the BUGS code for the random slopes and intercepts model and compares it to the previous version of this model used above. As we can see, the only difference is in the formulation of the prior for the variance components. The scale parameter of the Wishart distribution (**dwish**) is labeled W and has to be included as data. Because the precision matrix is 2 × 2 I use three for the degrees of freedom parameter.

|  |  |
| --- | --- |
| **Random slopes and intercepts** | **Random slopes and intercepts—Wishart prior (invertsvar2.txt)** |
| model{ for(i in 1:n) {  y[i]~dnorm(y.hat[i],tau.y)  y.hat[i]<-a[species[i]]+ b[species[i]]\*x[i]  }  tau.y<-pow(sigma.y,-2) sigma.y~dunif(0,100)  **#matrix of random slopes and intercepts**  for(j in 1:J){  a[j]<-B[j,1]  b[j]<-B[j,2]  B[j,1:2]~dmnorm(B.hat[j,], Tau.B[,])  B.hat[j,1]<-mu.a  B.hat[j,2]<-mu.b  } mu.a~dnorm(0,.0001) mu.b~dnorm(0,.0001)  **Tau.B[1:2,1:2]<- inverse(Sigma.B[,])**  **#variances of the random effects** **Sigma.B[1,1]<-pow(sigma.a,2) sigma.a~dunif(0,100) Sigma.B[2,2]<-pow(sigma.b,2) sigma.b~dunif(0,100) #correlation**  **Sigma.B[1,2]<-rho\*sigma.a\*sigma.b**  **Sigma.B[2,1]<-Sigma.B[1,2] rho~dunif(-1,1)** } | model{ for(i in 1:n) {  y[i]~dnorm(y.hat[i],tau.y)  y.hat[i]<-a[species[i]]+ b[species[i]]\*x[i]  }  tau.y<-pow(sigma.y,-2) sigma.y~dunif(0,100)  **#matrix of random slopes and intercepts**  for(j in 1:J){  a[j]<-B[j,1]  b[j]<-B[j,2]  B[j,1:2]~dmnorm(B.hat[j,], Tau.B[,])  B.hat[j,1]<-mu.a  B.hat[j,2]<-mu.b  } mu.a~dnorm(0,.0001) mu.b~dnorm(0,.0001) **Tau.B[1:2,1:2] ~ dwish(W[,],3)**  **sigma.raw[1:2,1:2] <- inverse(Tau.B[,])**  **#for comparison with model with separate priors**  **sigma.a <- sqrt(sigma.raw[1,1]) sigma.b <- sqrt(sigma.raw[2,2])**  **rho <- sigma.raw[2,1]/ (sigma.a\*sigma.b)**  } |

To obtain values for the diagonal entries of **W** I use the **lmList** function to fit a separate slopes and intercepts model to the data using a linear model with log(temp) centered at log(15). I calculate the individual variances of the intercepts and slopes and use them as the diagonal values of **W**.

out.lmL <- lmList(log(PLD)~I(log(temp)-log(15))|species, data=inverts)

varvec <- apply(coef(out.lmL),2,var)

W <- diag(2)\*varvec

When we modeled the elements of random effects covariance matrix separately we provided initial values for the elements of this matrix separately. In the matrix version of the random slopes and intercepts model we model the precision matrix as a unit. Hence we need to provide an initial estimate of the precision matrix. Because the precision matrix has a Wishart distribution a logical way of choosing an initial estimate is to generate a random matrix from a Wishart distribution. The **MCMCpack** package contains probability functions for the Wishart distribution. The **rwish** function generates a single matrix from a Wishart distribution with a specified degrees of freedom v and scale matrix S: rwish(v, S). In the code below I use v = 3 for the degrees of freedom and a diagonal 2 × 2 identity matrixfor the scale.

**#for rwish**

library(MCMCpack)

invert.data <- list("n","J","y","species","x","W")

invert.inits <- function() {list(B=array(rnorm(2\*J), c(J,2)), Tau.B=rwish(3,diag(2)), mu.a=rnorm(1), mu.b=rnorm(1))}

invert.parameters <- c("B", "sigma.y", "Tau.B", "mu.a", "mu.b")

invert.6a <- bugs(invert.data, invert.inits, invert.parameters, "invertsvar2.txt", n.chains=3, n.iter=100, debug=T)

invert.6a <- bugs(invert.data, invert.inits, invert.parameters, "invertsvar2.txt", n.chains=3, n.iter=5000, debug=T)

max(invert.6a$summary[,"Rhat"])

[1] 1.00992974217

invert.parameters2 <- c("a", "b", "sigma.y", "mu.a", "sigma.a", "mu.b", "sigma.b", "rho")

invert.6b <- bugs(invert.data, invert.6a$last.values, invert.parameters2, "invertsvar2.txt", n.chains=3, n.iter=5100, n.burnin=100)

round(invert.6b$summary[149:154,],4)

mean sd 2.5% 25% 50% 75% 97.5% Rhat n.eff  
sigma.y 0.1569 0.0131 0.1331 0.1479 0.1562 0.1648 0.1857 1.0003 1000  
mu.a 3.1432 0.1135 2.9230 3.0660 3.1450 3.2177 3.3540 1.0017 1000  
sigma.a 0.9344 0.0805 0.7938 0.8791 0.9262 0.9838 1.1010 1.0010 1000  
mu.b -1.4398 0.0924 -1.6279 -1.4998 -1.4410 -1.3742 -1.2591 0.9995 1000  
sigma.b 0.5241 0.0811 0.3771 0.4708 0.5215 0.5759 0.6935 1.0009 1000  
rho -0.4739 0.1285 -0.6969 -0.5634 -0.4840 -0.3885 -0.2083 1.0043 420

**Multilevel models with three hierarchical levels**

Thus far all of the multilevel models we've considered have had only two levels of structure. We now consider the problem of analyzing a multilevel model with three levels. The data I've selected track the growth of apples on trees in an orchard and are described in Schabenberger and Pierce (2002), p. 466, as follows.

At the Winchester Agricultural Experiment Station of Virginia Polytechnic Institute and State University ten apple trees were randomly selected and twenty five apples were randomly chosen on each tree. We concentrate on the analysis of the apples in the largest size class, those whose initial diameter exceeded 2.75 inches. In total there were eighty apples in that size class. Diameters of the apples were recorded in two-week intervals over a twelve-week period.

The goal is to develop a model of apple growth. Observe that the data have structure. We have repeated measurements on individual apples (up to six) and the apples themselves came in groups from individual trees. These are hierarchical data with three levels to the hierarchy: individual measurements nested in apples nested in trees.

apples<- read.table( 'http://www.unc.edu/courses/2010fall/ecol/563/001/data/lectures/AppleData.csv', sep=',', header=TRUE)

names(apples)

[1] "TREE" "APPLE" "size" "appleid" "time" "diam"

apples[1:15,]

TREE APPLE size appleid time diam  
1 1 1 7 1 1 2.90  
2 1 1 7 1 2 2.90  
3 1 1 7 1 3 2.90  
4 1 1 7 1 4 2.93  
5 1 1 7 1 5 2.94  
6 1 1 7 1 6 2.94  
7 1 4 7 4 1 2.86  
8 1 4 7 4 2 2.90  
9 1 4 7 4 3 2.93  
10 1 4 7 4 4 2.96  
11 1 4 7 4 5 2.99  
12 1 4 7 4 6 3.01  
13 1 5 7 5 1 2.75  
14 1 5 7 5 2 2.78  
15 1 5 7 5 3 2.80

The variable diam is the response and time is the predictor. It indicates the different times that measurements were made on the same apple. To understand the meaning of some of the variables a little bit better, I focus on a section of the data set where there is a transition from one tree to another.

apples[115:130,]

TREE APPLE size appleid time diam  
115 2 25 7 50 1 2.75  
116 2 25 7 50 2 2.80  
117 2 25 7 50 3 2.83  
118 2 25 7 50 4 2.85  
119 2 25 7 50 5 2.86  
120 2 25 7 50 6 2.88  
121 3 1 7 51 1 2.91  
122 3 1 7 51 2 3.00  
123 3 1 7 51 3 3.02  
124 3 1 7 51 4 3.03  
125 3 1 7 51 5 NA  
126 3 1 7 51 6 NA  
127 3 10 7 60 1 2.81  
128 3 10 7 60 2 2.89  
129 3 10 7 60 3 2.87  
130 3 10 7 60 4 2.93

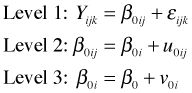
TREE identifies the tree from which the apple came and APPLE identifies the specific apple from that tree. Notice that APPLE is not a unique identifier because it starts over from 1 once we switch to a new tree. The variable appleid on the other hand uniquely identifies the different apples in the study. So, if we treat this as a two-level data set we will need to use appleid as the grouping variable. (If APPLE were used then the data for apples from different trees but with the same value of APPLE would be combined.) Observe also there are some missing data.

**The 3-level model without predictors**

There are three levels of information.

* Level 1 is the repeated measures on individual apples.
* Level 2 consists of the apples indicated by the variable appleid (or APPLE).
* Level 3 consists of the trees from which multiple apples were selected and is indicated by the variable TREE.

Just as with 2-level models, the proper place to start is the unconditional means model, a model with no predictors but with a random intercept at each level. The notation for a 3-level model is slightly more complicated because we now need three subscripts to reference the levels. So in the equations that follow, *i* represents the tree, *j* the apple, and *k* the individual observation on that apple.



or in composite form

level 3 composite

where variance level 1 effect, http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/var%20u0ij.gif, and level 3 effect.

**Fitting 3-level models with the nlme package**

To fit this model using the **lme** function of the **nlme** package, the only change in notation from 2-level models occurs in the random argument where the structural hierarchy is described. There are two ways the three-level model can be specified.

* When the same coefficients are random at all levels a shortcut notation allows the hierarchy to be specified as TREE/appleid where the variable identifying the highest level of the hierarchy, TREE, is listed first. So in this case the **random** argument is written as random=~1|TREE/appleid.
* The second way of specifying the random component is by listing the structural units first followed by a formula. The two units are linked together with a **list** function: random=list(TREE=~1,appleid=~1). As with the shortcut notation the required syntax in the random argument here lists the top level first followed by the level that is nested inside the top level. When different coefficients are made random at the two levels this second notation is the only one that is possible.

I fit the unconditional means (variance components) model accounting for all three levels. Note the use of the option **na.action=na.omit**. This instructs **lme** to drop observations that have missing data for any of the variables currently appearing in the model (only diam here). The default action with missing data is **na.fail** which causes the program to abort and not produce any output.

**#shortcut notation**

model0<-lme(diam~1, random=~1|TREE/appleid, data=apples, method='ML', na.action=na.omit)

**#equivalent notation**

model0a<-lme(diam~1, random=list(TREE=~1,appleid=~1), data=apples, method='ML', na.action=na.omit)

sapply(list(model0,model0a),AIC)

[1] -1092.074 -1092.074

VarCorr(model0)

Variance StdDev   
TREE = pdLogChol(1)   
(Intercept) 0.0001511123 0.01229278  
appleid = pdLogChol(1)   
(Intercept) 0.0093915920 0.09691023  
Residual 0.0030494623 0.05522194

The variance components tell us how much variance is available to be explained at each level. From the output we see that http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/sigma2.gif= 0.003 http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/tau02.gif= 0.00939, and http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/psi02.gif= 0.00015 . From this we can calculate the fraction of the total variance that is attributable to variation between trees.

0.0001511123/(0.0001511123+0.0093915920+0.0030494623)

[1] 0.0120005

The fraction of the total variance that is attributable to variation between apples on the same tree is

0.0093915920/(0.0001511123+0.0093915920+0.0030494623)

[1] 0.7458281

Finally the fraction of the total variance that is due to variation between different observations on the same apple is

0.0030494623/(0.0001511123+0.0093915920+0.0030494623)

[1] 0.2421714

These variance fractions can be useful to us in modeling. Levels that account for a lot of the total variance are natural places to include predictors to help explain that variance. What we see from the above fractions is that there is only a relatively trivial amount of variation between trees. Practically speaking it doesn't seem to matter whether apples were selected from the same tree or different trees.

There are also quantities analogous to the intra-class correlation coefficients, except that now there is more than one. The correlation between observations made on the same apple is calculated as follows.

rho1

The correlation between observations made on different apples on the same tree is given by

http://www.unc.edu/courses/2010fall/ecol/563/001/images/lectures/lecture28/rho2.gif

With so little of the variance occurring between trees there really is no need to continue with the 3-level model in this example. A 2-level model would suffice. Of course with only ten trees to work with we probably don't have enough data to estimate the tree variance component very accurately. So the absence of a substantial TREE effect may say more about the experimental design than it does about tree to tree variation.

If we include time as a level-1 predictor then we have the option of making time random at both the apple and the tree level. For this we need to use the alternative notation for the **random** argument. The following model fits a random intercept and a random slope (coefficient of time) at level-2, but only a random intercept at level-3.

model1a<-lme(diam~time, random=list(TREE=~1, appleid=~time), data=apples, method='ML', na.action=na.omit)

**Cited References**

* Gelman, A. and J. Hill. 2007. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, Cambridge, UK.
* Pinheiro, J. C. and Bates, D. M. 2000. *Mixed-Effects Models in S and S-Plus* . Springer-Verlag, New York.
* Schabenberger, Oliver and Francis J. Pierce. 2002. *Contemporary Statistical Models for the Plant and Soil Sciences*. CRC Press, Boca Raton, FL.

**R Code**

A compact collection of most of the R code displayed in this document appears [here](http://www.unc.edu/courses/2010fall/ecol/563/001/notes/lecture28%20Rcode.html).

[Course Home Page](http://www.unc.edu/courses/2010fall/ecol/563/001/index.html)

|  |
| --- |
| Jack Weiss *Phone:* (919) 962-5930 *E-Mail:* jack\_weiss@unc.edu *Address:* Curriculum in Ecology, Box 3275, University of North Carolina, Chapel Hill, 27599 Copyright © 2010 Last Revised--December 15, 2010 URL: [http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture28.htm#lecture28) |