Lecture 20 (lab 7)—Friday, February 25, 2012

Topics

- Revisiting the coral cores data set
- Mixed effects models in R
- The random intercepts model
  - Extracting parameter estimates from an lme object
  - Obtaining predictions for individual observations
  - Graphing the random intercepts model
- Adding a residual correlation structure to a mixed effects model
- Random slopes and intercepts model
- Random slopes without random intercepts
- Adding level-2 predictors to the model
  - Reef type as a predictor of the population intercept
  - Reef type as a predictor of the population slope
  - Reef type as a predictor of both the population intercept and slope
- The variance components model
- Fitting mixed effects models with the lme4 package
- Cited references
- Appendix: code for Figs. 2, 6, and 8

R functions and commands demonstrated

- ACF (from **nlme**) calculates the autocorrelation function for the residuals from an **lme** or **gls** model while at the same time accounting for the hierarchical structure of the data. It's advantage over **acf** is that it is designed to automatically work with grouped data.
- as.numeric converts a variable's class to numeric. It's useful for obtaining numerical values for factors and character data.
- detach is used to remove a previously loaded package from memory.
- fixef (from **nlme**) extracts the fixed effects estimates from an **lme** object.
- lme (from **nlme**) sets up and fits linear mixed effects models.
- lmer (from **lme4**) sets up and fits generalized linear mixed effects models.
- ranef extracts predictions of the random effects from an **lme** model.
- VarCorr (from **nlme**) extracts variance components from **lme** models.

R function options

- alpha= (option to **plot** when plotting ACF objects) is used to specify the α-level for confidence bands in plotting an **ACF** object.

- fixed= (argument to **lme**) defines the fixed effects portion of a linear mixed effects model.
- key= (argument to **xyplot** and other **lattice** functions) defines characteristics of the legend of a lattice graph.
- level= (argument to **predict**) specifies the level at which to compute predictions. The choice **level=0** corresponds to the population level.
- method= (argument to **lme**) sets the estimation method used to obtain parameter estimates. The default setting is "REML"; maximum likelihood estimates are obtained with **method="ML"**.
- random= (argument to **lme**) defines the random effects portion of a linear mixed effects model.
- REML= (argument to **lmer**) is used to specify maximum likelihood estimation, **REML=FALSE**, or restricted maximum likelihood estimation, the default.
- resType= (option to **ACF**) is used to specify the type of residuals to extract for use in calculating the ACF.
- subscripts (argument to the panel function of **xyplot** and other lattice functions) passes to panel functions the value of the index that identifies the panel that is currently being drawn.

R packages used

- lattice for panel graph displays of mixed effects models.
- lme4 was used for the function **lmer** for fitting generalized linear mixed effects models.
- nlme was used for the functions **lme** and **VarCorr** for mixed effects models with a normal random component.

Revisiting the coral cores data set

The data for today's lecture are the coral extension rates that we previously examined in lecture 17. They consist of the annual growth rings from coral cores extracted from 13 *Siderastrea siderea* colonies inhabiting the outer (forereef, backreef) and nearshore reefs of the Mesoamerican Barrier Reef System in southern Belize. The R commands from lecture 17 that we used to read in and process the data are listed below.

```
#load and manipulate data
ext.temp <- read.table( 'ecol 562/coral cores.txt', header=T, sep=',')
ext.rates.temp <- data.frame(rep(ext.temp$Year,14), unlist(ext.temp[,2:15]), rep(names(ext.temp)
   [2:15], rep(nrow(ext.temp),14)))
names(ext.rates.temp) <- c('Year','Ext.rate','Core')
ext.rates.temp$reef.type <- substr(ext.rates.temp$Core,1,2)
ext.rates <- ext.rates.temp[!is.na(ext.rates.temp$Ext.rate),]
ext.rates2 <- ext.rates[order(ext.rates$Core, ext.rates$Year),]
```

The data frame that these commands generate is called ext.rates2.

```
names(ext.rates2)
[1] "Core" "Year" "Ext.rate" "reef.type" "core.means"
ext.rate2[1:4,]
   Core Year Ext.rate reef.type core.means
1 BR.06 2008    0.809        BR  0.5960875
2 BR.06 2007    0.740        BR  0.5960875
```

```
3 BR.06 2006    0.705       BR  0.5960875
4 BR.06 2005    0.628       BR  0.5960875
```

This is a structured data set because although the observational units are the annual growth rings, the rings were obtained from a random sample of coral cores rather than a random sample of annual rings. Once a coral core was obtained it automatically brought with it all of the annual rings associated with that coral core. Such data are likely to exhibit observational heterogeneity because we have observations from different corals as well as multiple observations from the same coral core. Observations from the same core are likely to be more similar to each other than they are to observations from different cores.

In lecture 17 we fit a separate slopes and intercepts model using dummy variables to indicate the separate cores. We then accounted for temporal correlation in the separate time series of individual coral cores with generalized least squares. Today we replace the separate slopes and intercepts model with a random intercepts model and a random slopes and intercepts model. Because we are dealing with temporal data we will need to investigate whether an additional correlation model for the residuals is needed beyond the one that is contributed by the random effects.

Mixed effects models in R

There are two packages in R for fitting multilevel models. The older and more comprehensive package is **nlme**, an acronym for **n**on**l**inear **m**ixed **e**ffects models. Its workhorse function is **lme** whose limitation is that it only fits normal-based models and is not designed to fit mixed effects models to data that are nonhierarchical. The newer R package for fitting mixed models is **lme4** with its workhorse function **lmer**. The primary syntax differences between the **lme** and **lmer** functions lie in how the random effects component of the model is specified. The **lme4** package fits generalized linear mixed effects regression models such as logistic and Poisson models and permits the inclusion of both hierarchical and crossed random effects. It currently lacks some of the nonlinear features and correlation functions of **nlme**. Because the **lme4** package is under active development this situation may change.

Because the example we consider today has a normally distributed response, we will focus on the **nlme** package. The **lme4** package will be considered again when we discuss generalized linear mixed effects models. The primary reference for the **nlme** package is Pinheiro and Bates (2000). A book is currently being written to describe the **lme4** package. A preprint of the first five chapters is available here.

The random intercepts model

From a multilevel perspective, the coral cores data set consists of two levels. There is the annual ring level (level 1) and the core level (level 2). Any regression model that can be fit to a single level-2 unit, the individual core, is called a level-1 regression equation. The level-1 equation is our starting point for fitting a random intercepts model.

The random intercepts model is the simplest mixed effects model one can fit that still properly accounts for a data set's structure. For these data the variables that vary within a core are Ext.rate (the width of an individual ring) and Year (the time period during which that growth occurred). Thus the following regression equation could be fit separately to each individual coral core.

$$\text{Ext. rate}_{ij} = \beta_{0i} + \beta_{1i}\text{Year}_{ij} + \varepsilon_{ij}$$

where $\varepsilon_{ij} \sim \text{iid } N\left(0, \sigma^2\right)$. Here *i* denotes the coral core and *j* denotes a specific annual ring within that coral core. This is a level-1 equation. The equation says the width of the rings varies linearly with year. Because both β_{0i} and β_{1i} have subscripts *i*, each can potentially have different values for different coral cores.

In the random intercepts model we allow the intercept β_{0i} to vary, but β_{1i} is treated as a constant that is common to all of the cores.

$$\begin{aligned} \beta_{0i} &= \beta_0 + u_{0i} \\ \beta_{1i} &= \beta_1 \end{aligned}$$

where $u_{0i} \sim \text{iid } N\left(0, \tau^2\right)$. Here u_{0i} denotes how much coral core *i* deviates from the population average intercept β_0. The random intercepts model formulated as a multilevel model can be written as follows.

$$\begin{aligned} \text{Level 1:} \quad & \text{Ext. rate}_{ij} = \beta_{0i} + \beta_1 \text{Year}_{ij} + \varepsilon_{ij} \\ \text{Level 2:} \quad & \beta_{0i} = \beta_0 + u_{0i} \end{aligned}$$

where $\varepsilon_{ij} \sim \text{iid } N\left(0, \sigma^2\right), u_{0i} \sim \text{iid } N\left(0, \tau^2\right)$. To understand how the R specification of this model works we need to substitute the level-2 equation into the level-1 equation to obtain the composite equation.

$$\begin{aligned} \text{composite equation:} \quad \text{Ext. rate}_{ij} &= \beta_0 + \beta_1 \text{Year}_{ij} + u_{0i} + \varepsilon_{ij} \\ &= \underbrace{\beta_0 \cdot 1 + \beta_1 \text{Year}_{ij}}_{\text{fixed}} + \underbrace{u_{0i} \cdot 1}_{\text{random}} + \varepsilon_{ij} \end{aligned}$$

From the composite equation we can directly construct the corresponding R code for fitting this model. The function in the **nlme** package for fitting linear mixed effect models is **lme**.

```
model1 <- lme(fixed = Ext.rate~1+Year, random=~1|Core, data=ext.rates2, method="ML")
```

- The first argument defines the fixed part of the model. It specifies the response and the predictor, in this case an intercept, 1, and Year. As is the case with all regression models in R it is unnecessary to specify an intercept explicitly because R automatically includes it. So we could write this as `fixed = Ext.rate ~ Year`. Because we will always list the fixed part of the model first, it is unnecessary to include the argument name **fixed=**, and I will omit it from now on.
- The second argument **random=** defines the random part of the model. The required syntax is

```
random=~"random variable"|"structural variable"
```

Observe that the response is not specified in the random part so the model statement begins with just a ~. Because the level-2 equations (one in our case) only include a random effect in the intercept equation, the intercept is identified as the only predictor on the right hand side of the model expression in the **random=** argument, `random=~1`. Another way to understand the origin of 1 in the **random** argument is that in the composite model we can think of u_{0i} as being a coefficient multiplying the intercept, which is denoted by 1. Finally the vertical bar is used to separate the model specification from the structural specification. The variable Core is listed as the variable that defines the level-2 units.

- The third argument `data = ext.rates2` defines the data set that contains the variables.
- The last argument `method="ML"` requests that estimates be obtained using full maximum likelihood rather than the default estimation method of REML, restricted (residual) maximum likelihood. Choosing `method="ML"` is required for carrying out likelihood ratio tests and is necessary if we wish to use AIC to compare models with different predictors. REML provides better estimates of the variance components and standard errors than does ML. Having used `method="ML"` to compare different models it is sometimes recommended that the final model be refit using `method="REML"` and that the estimates and standard errors from the REML fit be used for the final inference.

The summary output from the random intercepts model is displayed below.

```
summary(model1)
Linear mixed-effects model fit by maximum likelihood
Data: ext.rates2
      AIC       BIC   logLik
-1020.799 -1002.230 514.3997

Random effects:
Formula: ~1 | Core
        (Intercept)  Residual
StdDev:  0.08384704 0.1203458

Fixed effects: Ext.rate ~ Year
                 Value Std.Error  DF   t-value p-value
(Intercept)  2.0263400 0.4010847 753  5.052150   0e+00
Year        -0.0008155 0.0002023 753 -4.030118   1e-04
 Correlation:
     (Intr)
Year -0.998

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3       Max
-3.5820921 -0.6203362 -0.0555163 0.5814467 3.2817378

Number of Observations: 767
Number of Groups: 13
```

I've highlighted the information of interest to us.

- In the first line we're told that maximum likelihood is the estimation method that was used. Thus the AIC and log-likelihood that appear on lines 3 and 4 are correct and can be used to make comparisons between models with different fixed effects (or random effects).
- The next section lists the estimates from the random effects part of the model. Recall that we don't actually estimate the random effects, we instead estimate the parameters of their distribution. For the random intercepts model these parameters are τ^2 and σ^2. Notice that in the line where the numerical estimates appears the label is StdDev, so in fact what's displayed here are τ (Intercept) and σ (Residual).
- In the Fixed effects section are the reported values of the intercept and slope, their estimated standard errors, and Wald tests that test whether each is significantly different from zero.

Extracting parameter estimates from an lme object

Variance components are extracted from the model with the **VarCorr** function.

```
VarCorr(model1)
Core = pdLogChol(1)
            Variance    StdDev
(Intercept) 0.007030326 0.08384704
Residual    0.014483115 0.12034581
```

What's reported in the first column are the squares of the values reported in the summary table, so these are variances. From the output we determine that $\hat{\tau}^2$ = 0.0070 and $\hat{\sigma}^2$ = 0.0145. It turns out that output from **VarCorr** is a matrix of character strings.

```
VarCorr(model1)[1,1]
[1] "0.007030326"
```

So if we want to use the values from **VarCorr** directly in calculations we need to convert them first to numbers with the **as.numeric** function.

```
as.numeric(VarCorr(model1)[1,1])
[1] 0.007030326
```

Previously we've used the **coef** function to extract the parameter estimates from regression objects. With an **lme** object there are three functions available for this purpose: **coef**, **fixef**, and **ranef**. The **fixef** function returns the fixed effect parameter estimates from an **lme** model. In model1 these are the estimates of the parameters we labeled $\beta_0$and β_1.

```
fixef(model1)
 (Intercept)         Year
 2.026340003 -0.000815468
```

The **ranef** function returns predictions of the random effects for each of the structural units in our model. The structural units are the individual cores. The random effects are the terms we've been labeling u_{0i} . As was explained in lecture 19, the random effects are not parameters that are estimated when fitting the model, instead they are obtained after the fact using the estimates of the regression parameters and the variance components that were obtained. These estimates are referred to as empirical Bayes estimates, also called BLUPs (best linear unbiased predictors).

```
ranef(model1)
      (Intercept)
BR.06  0.17060294
BR.07 -0.07687632
BR.08 -0.06710188
FR.02  0.01554065
FR.04  0.13069457
FR.05 -0.09064458
FR.09 -0.03437241
FR.11 -0.03181603
FR.12  0.03395007
FR.13 -0.04377088
NS.14  0.10160275
NS.15 -0.07635586
NS.16 -0.03145303
```

The **coef** function combines the fixed effects with the random effects. It returns the random intercepts $\beta_{0i} = \beta_0 + u_{0i}$ in column 1 as well as the fixed effect coefficient β_1 in column 2. Notice that the random intercepts vary by core while the fixed coefficient is constant across all cores.

```
coef(model1)
      (Intercept)         Year
BR.06    2.196943 -0.000815468
BR.07    1.949464 -0.000815468
BR.08    1.959238 -0.000815468
FR.02    2.041881 -0.000815468
FR.04    2.157035 -0.000815468
FR.05    1.935695 -0.000815468
FR.09    1.991968 -0.000815468
FR.11    1.994524 -0.000815468
FR.12    2.060290 -0.000815468
FR.13    1.982569 -0.000815468
NS.14    2.127943 -0.000815468
NS.15    1.949984 -0.000815468
NS.16    1.994887 -0.000815468
```

Obtaining predictions for individual observations

We obtain predictions of the mean for individual observations in an ordinary regression model by using the values of the predictors for those observations in the regression equation. With a random intercepts model there are two different predictions for the mean that we could calculate. There is the prediction that includes only the fixed effect estimates, referred to as the population average value, and a prediction that includes both the fixed effects and the random effects, called the subject-specific value.

$$\text{Population average value:} \quad \hat{\mu}_{ij} = \beta_0 + \beta_1 \text{Year}_{ij}$$

$$\text{Subject-specific value:} \quad \hat{\mu}_{ij} = \left(\beta_0 + u_{0i}\right) + \beta_1 \text{Year}_{ij}$$

We obtain these predictions by including the **level** argument in the **predict** function and assigning it the value 0 (population average) or 1 (subject-specific).

```
#population average
predict(model1, level=0)[1:12]
    BR.06     BR.06     BR.06     BR.06     BR.06     BR.06     BR.06     BR.06     BR.06 
0.4533023 0.4524869 0.4516714 0.4508559 0.4500404 0.4492250 0.4484095 0.4475940 0.4467786 
    BR.06     BR.06     BR.06 
0.4459631 0.4451476 0.4443322 
#subject specific
predict(model1, level=1)[1:12]
    BR.06     BR.06     BR.06     BR.06     BR.06     BR.06     BR.06     BR.06     BR.06 
0.6239053 0.6230898 0.6222743 0.6214589 0.6206434 0.6198279 0.6190125 0.6181970 0.6173815 
    BR.06     BR.06     BR.06 
0.6165661 0.6157506 0.6149351 
```

Observe that the difference between the population average and subject-specific predictions is equal to the random intercept for that core.

```
ranef(model1)[1,1]
[1] 0.1706029
predict(model1, level=1)[1:12] - predict(model1, level=0)[1:12]
    BR.06     BR.06     BR.06     BR.06     BR.06     BR.06     BR.06     BR.06     BR.06 
0.1706029 0.1706029 0.1706029 0.1706029 0.1706029 0.1706029 0.1706029 0.1706029 0.1706029 
    BR.06     BR.06     BR.06 
0.1706029 0.1706029 0.1706029 
```

Graphing the random intercepts model

Because the random intercepts model yields a different subject-specific model for each core, an obvious way to display the model graphically is as a lattice graph. Because the observations were previously sorted in date order within each core we can graph the models by connecting the predicted values with line segments. In order to obtain the correct subject-specific predictions for each core we need to include the **subscripts** key word as an argument to the panel function and then use it in specific panel function calls to select the appropriate observations. In the code below the **key** argument is used to create a legend.

```
library(lattice)
xyplot(Ext.rate~Year|Core, layout=c(4,4), data=ext.rates2, panel=function(x, y, subscripts) {
   panel.xyplot(x, y, type='l', col='grey70')
   panel.lines(x, predict(model1, level=0)[subscripts], col=1, lwd=2)
   panel.lines(x, predict(model1, level=1)[subscripts], col=2, lty=2)
}, key=list(x=.6, y=.95, origin=c(0,0), text=list(c('raw data', 'population average', 'subject-
   specific'), cex=.9), lines=list(col=c('grey70',1,2), lwd=c(1,2,1), lty=c(1,1,2))))
```

The **key** argument is mostly self-explanatory although a bit convoluted.

- The key itself is an R object called a list that is here created with the **list** function. A list object in R is one whose components can be of various types and of different lengths.
- The **text=** and **lines=** arguments shown in the key are specified separately also using individual **list** functions. The elements of each list are the characteristics that we want the displayed text and symbols in the legend to have. For the **text** list I specify the vector of text that should appear in the legend and the size I want it to have. For the **lines** list I specify the line types, line widths, and colors.
- To position the legend inside the panels I give relative coordinates with the **x=** and **y=** arguments. Because I specified **corner=c(0,0)**, the **x=** and **y=** arguments are measured with respect to this corner, the lower left corner. The bottom of the figure is $y = 0$ and the top of the figure corresponds to $y = 1$. The left edge of the figure is $x = 0$ and the right edge of the figure corresponds to $x = 1$. By specifying `x=.6, y=.95` the legend is placed near the top right corner.

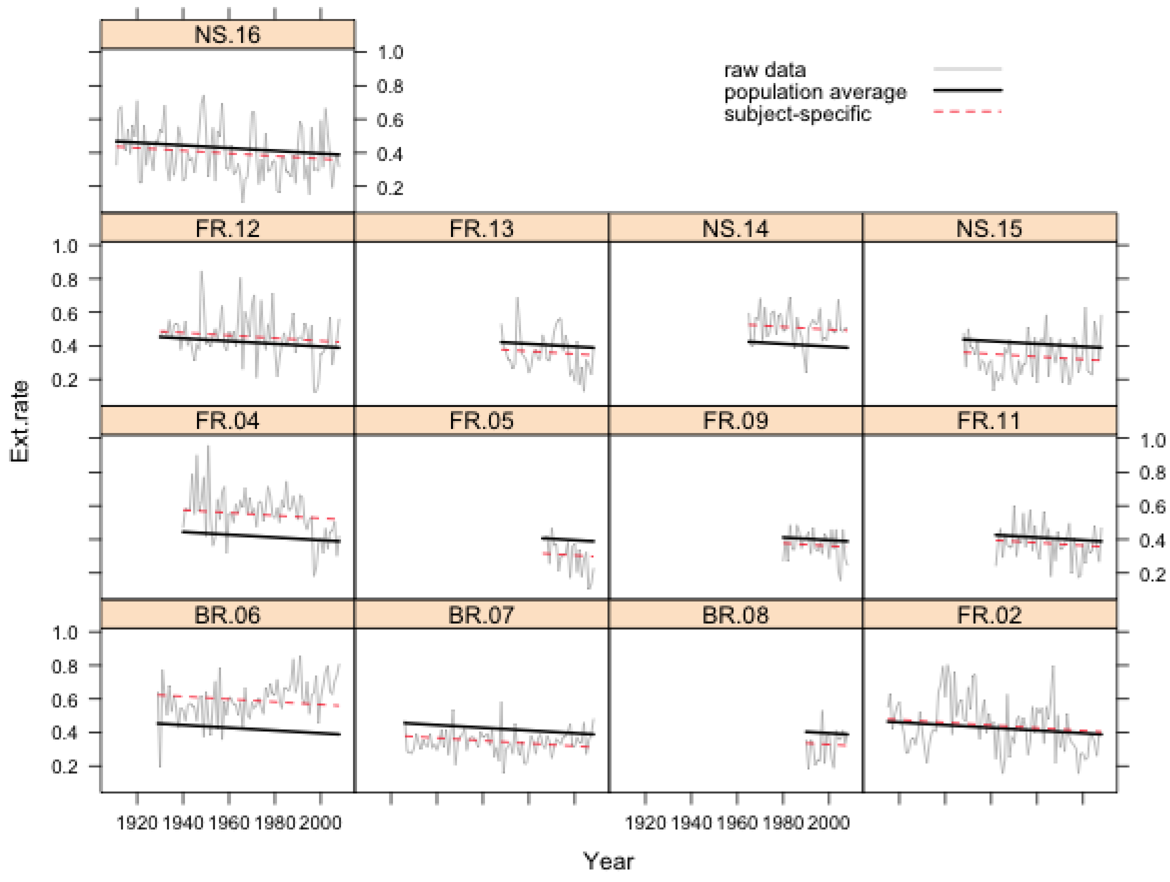

Fig. 1 Random intercepts model in separate panels

What we see in Fig. 1 are the raw data for each core connected by line segments, the subject-specific model, and the population average model. The population average model is exactly the same for each core but the subject-specific model varies from core to core. Because this is a random intercepts model, the subject-specific model differs from the population average model only in the value of its intercept. Thus all the displayed regression lines are parallel.

The variation of the actual observations about their individual subject-specific lines is the level-1 variance σ^2. The variation of the individual subject-specific lines about the population average line is the level-2 variance τ^2. Fig. 2 shows all the data and regression lines in a single figure and tries to distinguish the different roles that σ^2 and τ^2 have in the model.

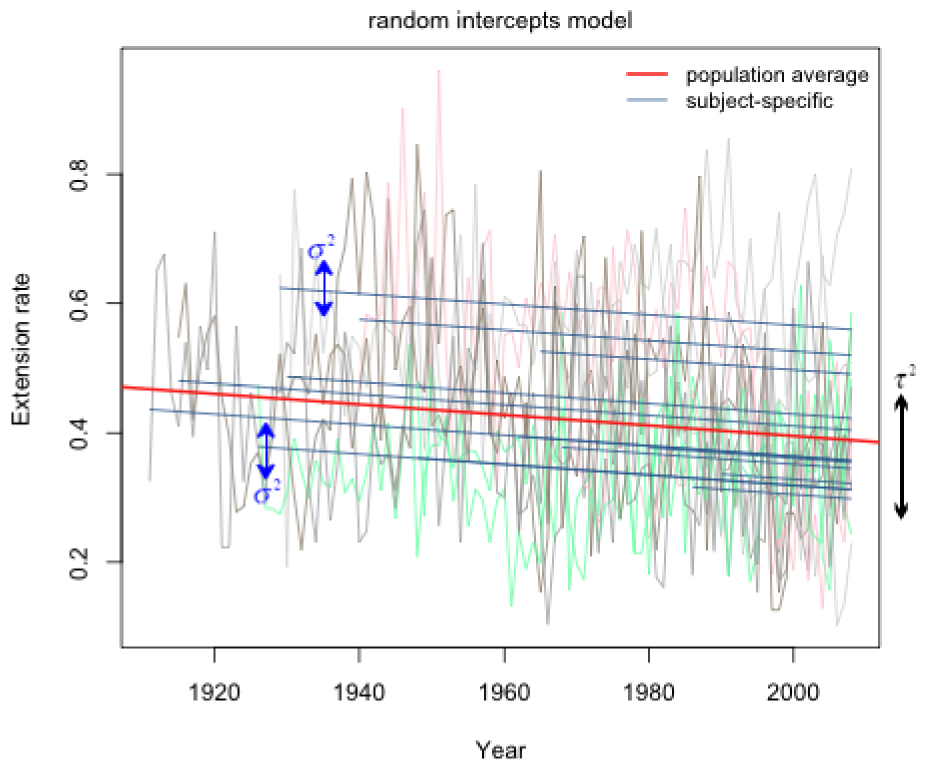

Fig. 2 Random intercepts model (R code)

Adding a residual correlation structure to a mixed effects model

While including random effects in a model induces a correlation structure in the response (see lecture 19), with long time series, like we have here, the induced correlation structure is typically inadequate. So, we should investigate whether the random intercepts model is properly accounting for the correlation. The method for checking this is to plot the ACF of the level-1 residuals.

Last week we used the **acf** function for this purpose, but to use it with grouped data we had to insert missing values between the residuals from the different groups to prevent observations from two different groups contributing to the correlations at the various lags. The **nlme** package has its own ACF function, **ACF**, that automatically accounts for the grouping that is specified when fitting an **lme** model. The **ACF** function assumes that the observations within a group are in their correct temporal order in the original data frame and that the time spacing of the observations is the same. If this is not the case, the results returned by **ACF** will not be sensible. The **ACF** function will also get the wrong answer if there are gaps in the time sequence. Because the observations in the coral cores data set are ordered correctly and there are no missing values we can use the **ACF** function here.

The **ACF** function acts on an **lme** model from which it extracts the level-1 residuals. To get a plot we use the output from **ACF** as the argument of the **plot** function. Confidence bands are displayed if a value for the **alpha** argument, the significance level for the bands, is given.

```
plot(ACF(model1), alpha=.05)
```

Because 20 lags are displayed we can modify the confidence bands with a Bonferroni correction that assumes that 20 tests have been carried out.

```
plot(ACF(model1),alpha=.05/20)
```

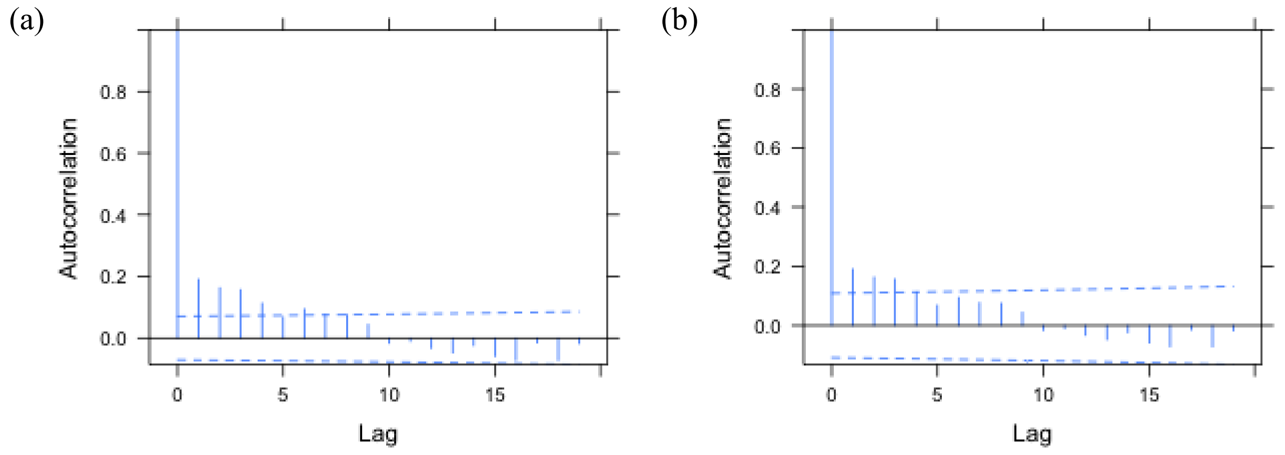

Fig. 3 ACF for the random intercepts model using (a) alpha = .05 and (b) alpha = .05/20, a Bonferroni correction to account for testing at 20 different lags.

The displayed ACF has the signature of an autoregressive process. There is no PACF function in **nlme** so to produce one here we'd have to use the **pacf** function and go through the same protocol that we used in lecture 17. Because we already know a low-order ARMA process worked there, I just try fitting an AR(1) and ARMA(1,1) process and compare them using AIC. The syntax for including a correlation structure in an **lme** model is the same as it was for the **gls** function. We need to include a **form** argument in which we indicate the "time" variable and optionally a grouping term. Specifying the grouping structure is unnecessary here because we're updating a model and the group structure will be inherited from the model we're updating

```
#AR(1)
model1a <- update(model1, correlation=corARMA(p=1,q=0, form=~Year|Core))
#explicitly indicate the groups
model1a1 <- update(model1, correlation=corARMA(p=1,q=0, form=~Year|Core))
#ARMA(1,1)
model1b <- update(model1, correlation=corARMA(p=1,q=1, form=~Year|Core))
AIC(model1, model1a, model1b)
        df       AIC
model1   4 -1020.799
model1a  5 -1051.789
model1b  6 -1085.985
```

The AIC indicates that the ARMA(1,1) model beats the AR(1) model and both beat a model with no correlation structure. To see if the correlation has been properly accounted for we can use the **resType** argument of **ACF**. Just like the **type** argument of residuals, **resType** can be used to tell **ACF** to extract normalized residuals. When we use the Bonferroni correction of Fig. 3b, both the AR(1) and ARMA(1,1) models appear to remove all the statistically significant correlations, but the correlation shows a much greater reduction with the ARMA(1,1) model.

```
plot(ACF(model1b, resType='normalized'), alpha=.05/20)
plot(ACF(model1a, resType='normalized'), alpha=.05/20)
```

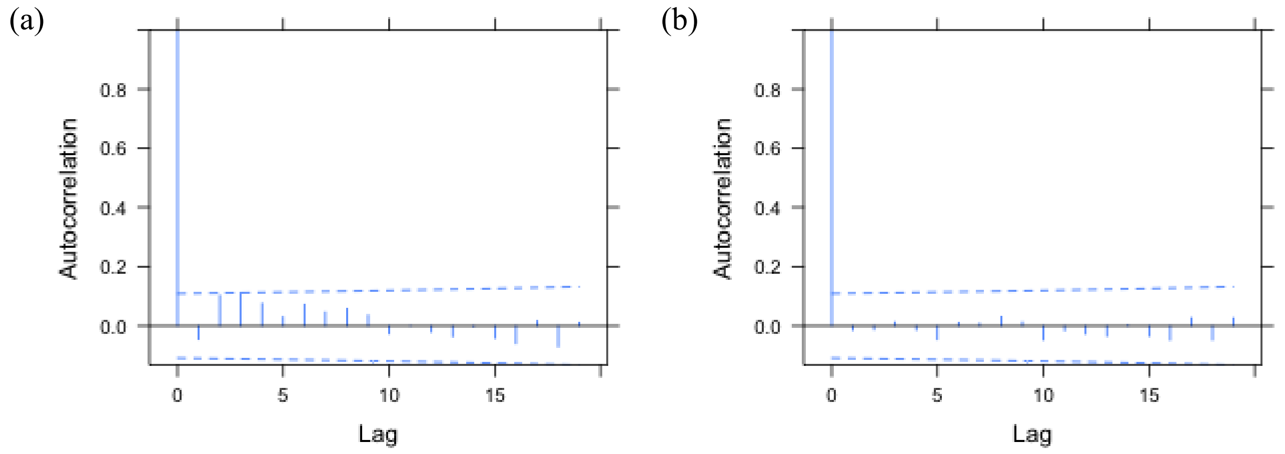

Fig. 4 ACF for the normalized residuals of a random intercepts model with (a) an AR(1) model for the residuals and (b) an ARMA(1,1) model for the residuals. An experimentwise α = .05 rate was obtained using a Bonferroni correction under the assumption that 20 lags were examined.

If we compare the summary table of the model without a correlation structure to the model with an ARMA(1,1) model for the residuals we observe that there is a big difference in the standard errors and *p*-values.

```
printCoefmat(summary(model1)$tTable)
                  Value    Std.Error          DF     t-value p-value
(Intercept)  2.0263e+00  4.0108e-01  7.5300e+02  5.0522e+00   0e+00
Year        -8.1547e-04  2.0234e-04  7.5300e+02 -4.0301e+00   1e-04
printCoefmat(summary(model1b)$tTable)
                  Value    Std.Error          DF     t-value p-value
(Intercept)  2.1305e+00  8.2271e-01  7.5300e+02  2.5896e+00  0.0098
Year        -8.6679e-04  4.1578e-04  7.5300e+02 -2.0847e+00  0.0374
```

Notice that the coefficient of Year has gone from being highly significant to being barely significant. The reason for that is simple. In model1 we treated the observations as being independent when they weren't. Because the observations are positively correlated the response variable exhibits much less variability than it should. When we account for this correlation we get a more honest estimate of the true variability of the response. Put another way, when we treat the observations as being independent we are exaggerating our sample size and hence our precision. When we properly account for the correlation the effective sample is much less than the naive sample size obtained assuming independence.

The random slopes and intercepts model

The random slopes and intercepts model adds a second equation at level 2 to the random intercepts model by allowing slopes to vary across cores. The multilevel model formulation is shown below.

$$\text{Level 1}: \quad \text{Ext. rate}_{ij} = \beta_{0i} + \beta_{1i}\text{Year}_{ij} + \varepsilon_{ij}$$

$$\text{Level 2}: \quad \begin{aligned} \beta_{0i} &= \beta_0 + u_{0i} \\ \beta_{1i} &= \beta_1 + u_{1i} \end{aligned}$$

where

$$\varepsilon_{ij} \sim \text{iid } N\left(0,\sigma^2\right), \mathbf{u}_i = \begin{bmatrix} u_{0i} \\ u_{1i} \end{bmatrix} \sim \text{iid } N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \tau_0^2 & \rho\tau_0\tau_1 \\ \rho\tau_0\tau_1 & \tau_1^2 \end{bmatrix}\right)$$

The equivalent composite equation formulation, obtained by plugging the two level-2 equations into the level-1 equation, is shown next.

$$\begin{aligned} \text{composite equation}: \quad \text{Ext. rate}_{ij} &= \beta_0 + \beta_1\text{Year}_{ij} + u_{0i} + u_{1i}\text{Year}_{ij} + \varepsilon_{ij} \\ &= \underbrace{\beta_0 \cdot 1 + \beta_1\text{Year}_{ij}}_{\text{fixed}} + \underbrace{u_{0i} \cdot 1 + u_{1i}\text{Year}_{ij}}_{\text{random}} + \varepsilon_{ij} \end{aligned}$$

From this we can write down the R formulation of the model.

```
model2 <- lme(Ext.rate~1+Year, random=~1+Year|Core, data=ext.rates2, method="ML")
```

or equivalently

```
model2 <- lme(Ext.rate~Year, random=~Year|Core, data=ext.rates2, method="ML")
```

The summary output from this model does not look unusual but in fact there is a problem that manifests itself when we examine the variance components.

```
VarCorr(model2)
Core = pdLogChol(1 + Year)
                Variance       StdDev    Corr
(Intercept) 6.189482e-15 7.867326e-08 (Intr)
Year        1.799186e-09 4.241681e-05 0.419
Residual    1.447072e-02 1.202943e-01
```

Notice that the variance of the random intercepts is reported to be 10^{-15}, essentially zero, whereas in model1, the model without random slopes, it was estimated to be 0.007. What happened is that the **lme** function was not able to estimate both a random slope and a random intercept variance in this model. Usually the symptoms of this are more obvious than they are here, but the solution is the same—center the predictor. In multilevel models it is crucial that predictors in the model be roughly on the same scale. The magnitude of Year is currently too big. As we remarked in lecture 17 it's also the case that using a raw scale for Year is crazy because then the intercept corresponds to the extension rate in the year 0 A.D.

In a multilevel model centering is almost always necessary in order to obtain valid solutions. I refit the model centering year at 1967. Notice that centered year gets used in the random portion of the model too. This should be obvious because it is the coefficient of centered year, not year, that is being made random.

```
model2.uncentered <- lme(Ext.rate~Year, random=~Year|Core, data=ext.rates2, method="ML")
model2 <- lme(Ext.rate~I(Year-1967), random=~I(Year-1967)|Core, data=ext.rates2, method="ML")
AIC(model1, model2.uncentered, model2)
                  df       AIC
model1             4 -1020.799
model2.uncentered  6 -1017.420
model2             6 -1046.547
```

We now have obtained sensible results. The addition of random slopes has led to a large decrease in AIC. Notice that the variance components are also dramatically changed from the uncentered model.

```
VarCorr(model2)
Core = pdLogChol(I(Year - 1967))
                   Variance      StdDev    Corr
(Intercept)    6.769404e-03 0.082276386 (Intr)
```

```
I(Year - 1967) 2.131837e-06 0.001460081 0.009
Residual       1.360657e-02 0.116647202
```

From the output we see that $\hat{\tau}_0^2$ = 0.00677, $\hat{\tau}_1^2$ = 0.000002, and $\hat{\sigma}^2$ = 0.0136. The reported correlation is the correlation between the random slopes and intercepts, which is now near zero.

The **ranef** function displays BLUPs of both the intercept and slope random effects.

```
ranef(model2)
      (Intercept) I(Year - 1967)
BR.06  0.15941823   2.940924e-03
BR.07 -0.08429038   1.072531e-03
BR.08 -0.06470105  -8.377937e-05
FR.02  0.00383491  -5.583785e-04
FR.04  0.13538494  -1.518369e-03
FR.05 -0.04938103  -1.515433e-03
FR.09 -0.02255652  -5.097505e-04
FR.11 -0.02862262  -3.662648e-04
FR.12  0.02754750  -2.475086e-04
FR.13 -0.02870880  -8.749352e-04
NS.14  0.09128868   4.128806e-04
NS.15 -0.09687663   1.477382e-03
NS.16 -0.04233724  -2.292991e-04
```

The **coef** function adds these values to the corresponding fixed effects to obtain estimates of the random slopes and intercepts.

```
fixef(model2)
   (Intercept) I(Year - 1967)
   0.429875436   -0.001056583
coef(model2)
      (Intercept) I(Year - 1967)
BR.06   0.5892937   1.884341e-03
BR.07   0.3455851   1.594844e-05
BR.08   0.3651744  -1.140362e-03
FR.02   0.4337103  -1.614961e-03
FR.04   0.5652604  -2.574952e-03
FR.05   0.3804944  -2.572015e-03
FR.09   0.4073189  -1.566333e-03
FR.11   0.4012528  -1.422847e-03
FR.12   0.4574229  -1.304091e-03
FR.13   0.4011666  -1.931518e-03
NS.14   0.5211641  -6.437019e-04
NS.15   0.3329988   4.208000e-04
NS.16   0.3875382  -1.285882e-03
```

Fig. 5 displays the population average and subject-specific random slopes and intercepts models. As was the case in Fig. 1, the population-average line is the same in each panel but the subject-specific model varies. This time both the slopes and intercepts of these lines vary across panels.

```
xyplot(Ext.rate~Year|Core,data=ext.rates2, layout=c(4,4), panel=function(x,y,subscripts) {
   panel.xyplot(x, y, type='l', col='grey70')
   panel.lines(x, predict(model2, level=0)[subscripts], col=1, lwd=2)
   panel.lines(x, predict(model2, level=1)[subscripts], col=2, lty=2)
}, key=list(x=.6, y=.95, origin=c(0,0), text=list(c('raw data', 'population average', 'subject-
   specific'), cex=.9), lines=list(col=c('grey70',1,2), lwd=c(1,2,1), lty=c(1,1,2))))
```

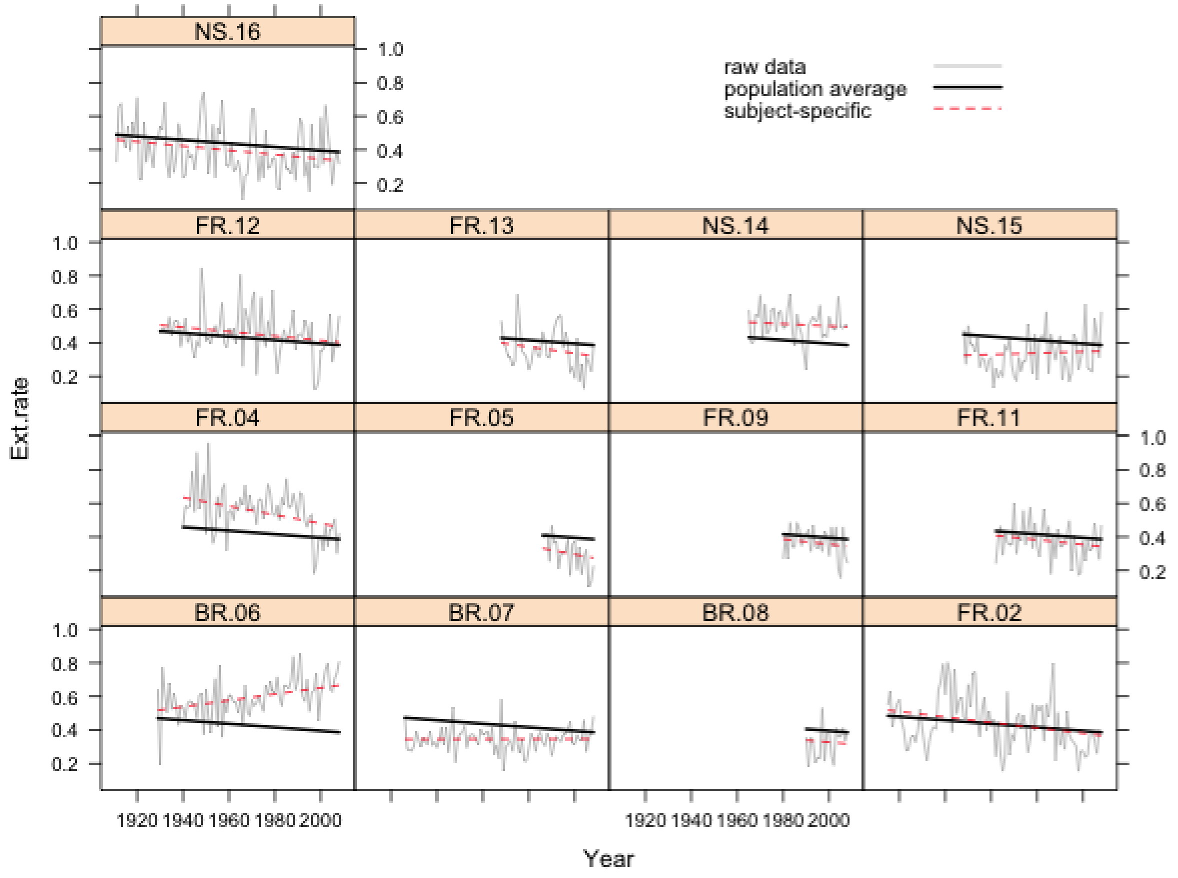

Fig. 5 Random slopes and intercepts model in separate panels

Fig. 6 shows all the data and regression lines in a single figure and tries to distinguish the different roles the three variance components have in the model.

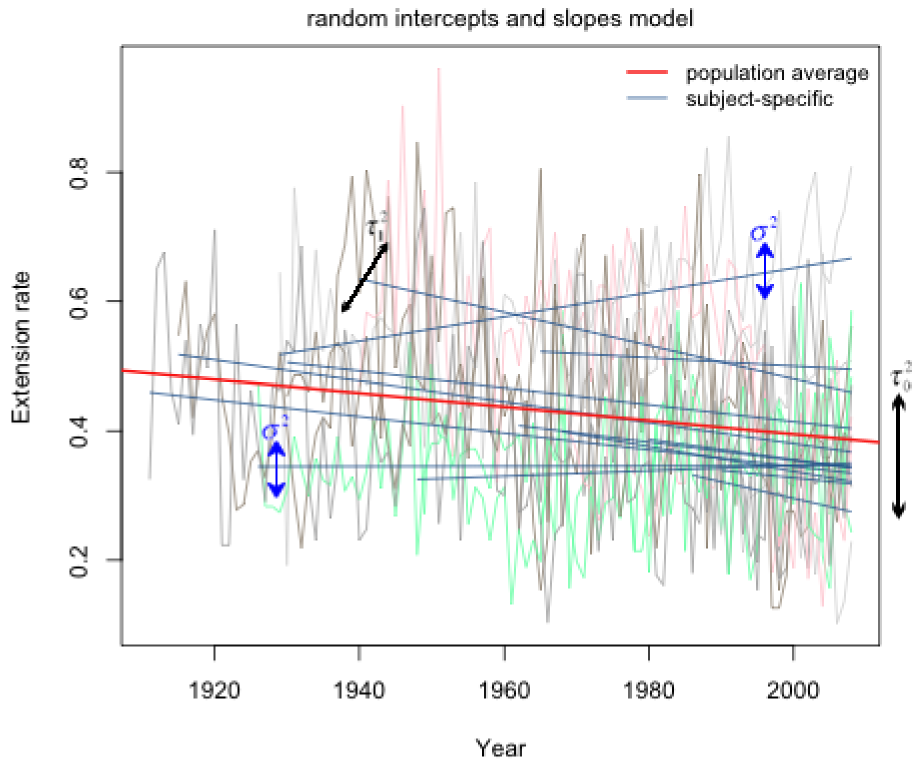

Fig. 6 Random slopes and intercepts model (R code)

- τ_0^2 is the variability in the intercepts of the subject-specific regression lines about the intercept of the population-average line (the line determined by the estimated fixed effects). This describes the spread of the lines in the vertical direction (keeping their slopes the same).
- τ_1^2 is the variability of the individual slopes about the slope of the population-average line. It can be thought of as describing the extent to which individual lines are rotated around the population-average line.
- σ^2 is the average variability of the observations about their own individual core-level regression lines. It has the same interpretation as in ordinary regression.

Random slopes without random intercepts

It is possible to fit a model in which the slopes are random (each core gets its own slope) but the intercepts are fixed. To accomplish this we have to remove the intercept from the random formula with the -1 notation we've used previously. Thus we would need to write: `random = ~I(Year-1967)-1|Core`. When we compare the random slopes, random intercepts, and random slopes and intercepts models for these data, the random slopes and intercepts model has the lowest AIC.

```
#fit a model with only random slopes
```

```
model2.5 <- lme(Ext.rate~I(Year-1967), data=ext.rates2, random=~I(Year-1967)-1|Core, method="ML")
AIC(model2, model1, model2.5)
        df       AIC
model2    6 -1046.5471
model1    4 -1020.7995
model2.5  4  -801.8022
```

Adding level-2 predictors to the model

The reef.type variable in the data set is a level-2 predictor because it varies only at the core level. It is a categorical variable with three levels so it needs to be converted to a factor when included in a regression model. By default R constructs the dummy variables alphabetically so that forereef and nearshore coral colonies are contrasted with backreef coral colonies.

```
contrasts(factor(ext.rates2$reef.type))
   FR IR
BR  0  0
FR  1  0
IR  0  1
```

Thus R has created two dummy variables of the following form.

$$Z_1 = \begin{cases} 1, & \text{if reef type = 'FR'} \\ 0, & \text{otherwise} \end{cases} \qquad Z_2 = \begin{cases} 1, & \text{if reef type = 'NS'} \\ 0, & \text{otherwise} \end{cases}$$

To include reef.type in the model we have to decide in what way we think it alters the basic level-1 relationship between Ext.rate and centered Year. Does it modify the intercept, the slope, or both? The researchers hypothesized that it should modify the slope. We'll try all three possibilities.

Reef type as a predictor of the population intercept

If reef.type affects the intercept it should be entered into the level-2 equation for the intercept.

$$\text{Level 1:}\quad \text{Ext. rate}_{ij} = \beta_{0i} + \beta_{1i}\left(\text{Year}_{ij} - 1967\right) + \varepsilon_{ij}$$

$$\text{Level 2:}\quad \begin{aligned} \beta_{0i} &= \beta_0 + \gamma_1 Z_{1i} + \gamma_2 Z_{2i} + u_{0i} \\ \beta_{1i} &= \beta_1 + u_{1i} \end{aligned}$$

The corresponding composite equation is shown below. In it we see that reef type becomes part of the fixed effects portion of the model.

$$\begin{aligned}\text{Ext. rate}_{ij} &= \beta_0 + \beta_1\left(\text{Year}_{ij} - 1967\right) + \gamma_1 Z_{1i} + \gamma_2 Z_{2i} + u_{0i} + u_{1i}\left(\text{Year}_{ij} - 1967\right) + \varepsilon_{ij} \\ &= \underbrace{\beta_0 \cdot 1 + \beta_1\left(\text{Year}_{ij} - 1967\right) + \gamma_1 Z_{1i} + \gamma_2 Z_{2i}}_{\text{fixed}} + \underbrace{u_{0i} \cdot 1 + u_{1i}\left(\text{Year}_{ij} - 1967\right)}_{\text{random}}\end{aligned}$$

To estimate this model in R we start with the random slopes and intercepts formulation and just add reef.type to the fixed part of the model.

```
model3<- lme(Ext.rate~I(Year-1967) + factor(reef.type), random=~I(Year-1967)|Core, data=ext.rates2,
   method='ML')
```

Reef type as a predictor of the population slope

To see if reef.type affects the slope we enter it into the level-2 equation for the slope.

$$\begin{aligned}\text{Level 1:}\quad & \text{Ext. rate}_{ij} = \beta_{0i} + \beta_{1i}\left(\text{Year}_{ij} - 1967\right) + \varepsilon_{ij} \\ \text{Level 2:}\quad & \beta_{0i} = \beta_0 + u_{0i} \\ & \beta_{1i} = \beta_1 + \gamma_3 Z_{1i} + \gamma_4 Z_{2i} + u_{1i}\end{aligned}$$

or in composite form

$$\begin{aligned}\text{Ext. rate}_{ij} &= \beta_0 + \beta_1\left(\text{Year}_{ij} - 1967\right) + \gamma_3 Z_{1i} \times \left(\text{Year}_{ij} - 1967\right) + \gamma_4 Z_{2i} \times \left(\text{Year}_{ij} - 1967\right) \\ &\quad + u_{0i} + u_{1i}\left(\text{Year}_{ij} - 1967\right) + \varepsilon_{ij} \\ &= \underbrace{\beta_0 \cdot 1 + \beta_1\left(\text{Year}_{ij} - 1967\right) + \gamma_3 Z_{1i} \times \left(\text{Year}_{ij} - 1967\right) + \gamma_4 Z_{2i} \times \left(\text{Year}_{ij} - 1967\right)}_{\text{fixed}} \\ &\quad + \underbrace{u_{0i} \cdot 1 + u_{1i}\left(\text{Year}_{ij} - 1967\right)}_{\text{random}}\end{aligned}$$

In R we specify this model by entering a term for the interaction between reef.type and I(Year-1967). Thus we need to include the term `I(Year-1967):factor(reef.type)` in the fixed part of the model expression.

```
model4 <- lme(Ext.rate~I(Year-1967) + I(Year-1967):factor(reef.type), random=~I(Year-1967)|Core,
   data=ext.rates2, method='ML')
```

Reef type as a predictor of both the population slope and intercept

If reef.type affects both the slope and intercept it should be entered into both of the level-2 equations.

$$\begin{aligned}
\text{Level 1:} \quad & \text{Ext. rate}_{ij} = \beta_{0i} + \beta_{1i}\left(\text{Year}_{ij} - 1967\right) + \varepsilon_{ij} \\
\text{Level 2:} \quad & \beta_{0i} = \beta_0 + \gamma_1 Z_{1i} + \gamma_2 Z_{2i} + u_{0i} \\
& \beta_{1i} = \beta_1 + \gamma_3 Z_{1i} + \gamma_4 Z_{2i} + u_{1i}
\end{aligned}$$

or in composite form

$$\begin{aligned}
\text{Ext. rate}_{ij} &= \beta_0 + \gamma_1 Z_{1i} + \gamma_2 Z_{2i} + \beta_1\left(\text{Year}_{ij} - 1967\right) + \gamma_3 Z_{1i} \times \left(\text{Year}_{ij} - 1967\right) + \gamma_4 Z_{2i} \times \left(\text{Year}_{ij} - 1967\right) \\
&\quad + u_{0i} + u_{1i}\left(\text{Year}_{ij} - 1967\right) + \varepsilon_{ij} \\
&= \underbrace{\beta_0 \cdot 1 + \gamma_1 Z_{1i} + \gamma_2 Z_{2i} + \beta_1\left(\text{Year}_{ij} - 1967\right) + \gamma_3 Z_{1i} \times \left(\text{Year}_{ij} - 1967\right) + \gamma_4 Z_{2i} \times \left(\text{Year}_{ij} - 1967\right)}_{\text{fixed}} \\
&\quad + \underbrace{u_{0i} \cdot 1 + u_{1i}\left(\text{Year}_{ij} - 1967\right)}_{\text{random}}
\end{aligned}$$

From the composite form the fixed part of the model should include I(Year-19667), factor(reef.type) and the interaction between the two. We can obtain this in R as follows.

```
fixed=Ext.rate~I(Year-1967)+factor(reef.type)+I(Year-1967):factor(reef.type)
```

or using R's shortcut notation

```
fixed=Ext.rate~I(Year-1967)*factor(reef.type)
```

where the * notation automatically generates all three terms (plus the intercept).

```
model5 <- lme(Ext.rate~I(Year-1967)*factor(reef.type), random=~I(Year-1967)|Core, data=ext.rates2,
    method='ML')
AIC(model5, model4, model3, model2)
       df       AIC
model5 10 -1051.221
model4  8 -1054.929
```

```
model3   8 -1042.783
model2   6 -1046.547
```

Based on AIC we should include reef.type as a predictor of the slopes but not the intercepts. Notice that in all three models the random portion of the model has remained the same. The random portion is determined by which coefficients in the level-1 model were declared to be random, not by predictors in the level-2 equations.

It may seem a little strange to consider a model with an interaction but without the corresponding main effect. Usually we would test the terms in model5 as follows.

```
anova(model5)
                                numDF denDF  F-value p-value
(Intercept)                         1   751 344.1075  <.0001
I(Year - 1967)                      1   751   9.5658  0.0021
factor(reef.type)                   2    10   0.5458  0.5957
I(Year - 1967):factor(reef.type)    2   751  10.8099  <.0001
```

These are variables added in order tests. The basic level-1 model appears on line 2. Line 3 tests whether the single population model should instead be three population models (based on reef type) that are parallel lines but with different intercepts. Based on the *p*-value the answer is no. When we add an interaction term so that the three population lines also have different slopes, we obtain a significant improvement (line 4).

In an ordinary regression model this is where we'd stop, but in a multilevel model removing the main effect of reef.type can make sense. Because this is a random slopes and intercepts model, individual cores already have different slopes and intercepts. So removing the reef type main effect does not lead to the silly constraint of forcing all the lines to pass through a single point. From the multilevel formulation it is clear that we can choose to add reef.type to the slope equation and not the intercept equation.

We can formally compare these models with a partial *F*-test because they're nested.

```
anova(model4,model5)
       Model df       AIC       BIC   logLik   Test   L.Ratio p-value
model4     1  8 -1054.928 -1017.789 535.4643                         
model5     2 10 -1051.221 -1004.796 535.6105 1 vs 2 0.2925490  0.8639
```

The non-significant result is consistent with our conclusions based on AIC. Allowing the population value of the intercepts to vary by reef type is not supported by the data. Fig. 7 displays the final reef type model, model4, as a panel graph. Unlike Figs. 1 and 5, now the population average model is not the same in each panel. Instead it varies by reef type.

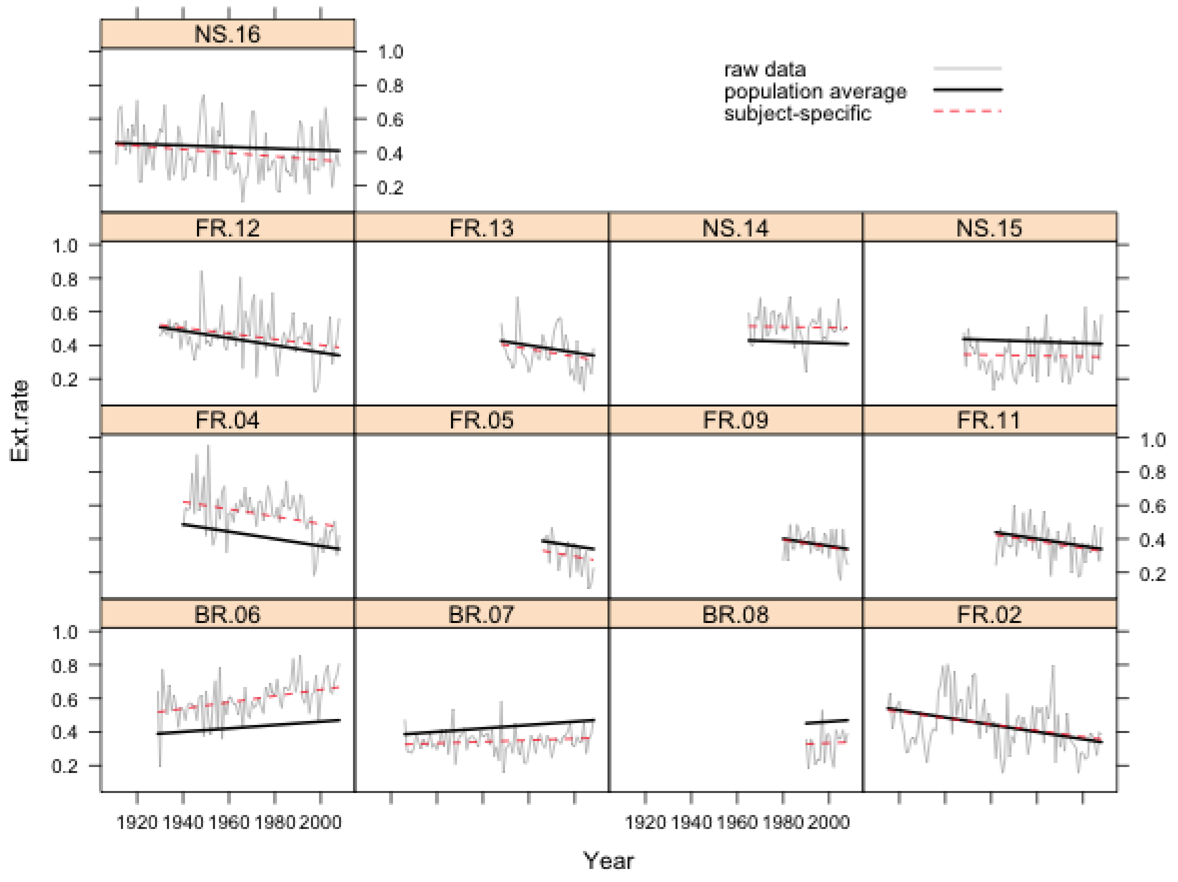

Fig. 7 Random slopes and intercepts model with separate population slopes based on reef type

Fig. 8 displays this same model now divided into only three panels corresponding to the three different reef types. In each case the set of subject-specific lines in that panel varies randomly (i.e. the slopes and intercepts vary randomly) about the displayed population line.

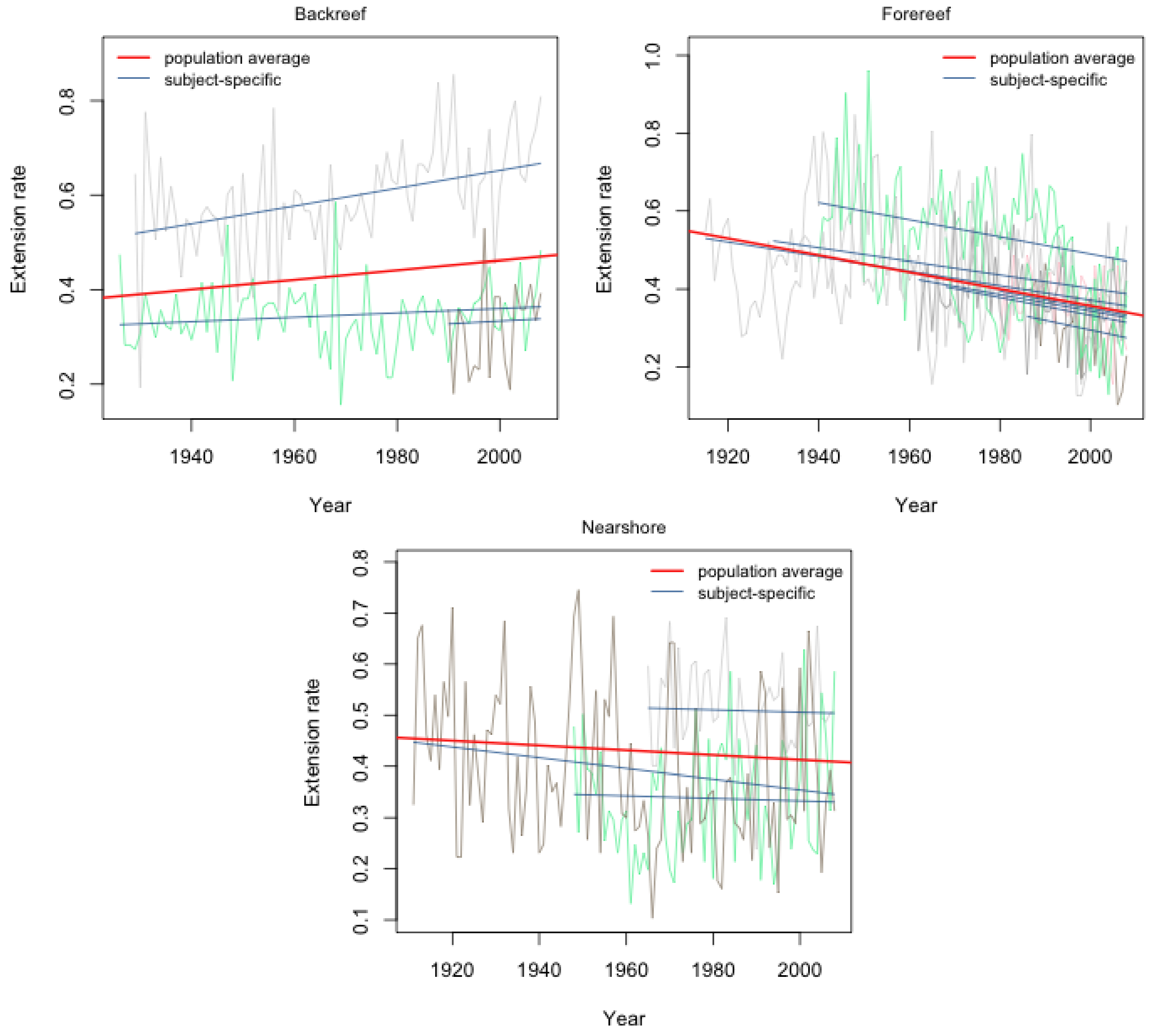

Fig. 8 Random slopes and intercepts model with separate population slopes based on reef type (R code)

The variance components model

A starting place when modeling hierarchical data is to first fit a variance components model, also called the unconditional means model. The variance components model is substantively uninteresting because it contains no predictors. It does however account for the structure in the data. Its value statistically is that it allows us to estimate variance components. In particular it allows us to determine how much variance lies within subjects versus how much variance lies between subjects. To allow for heterogeneity across level 2 units, it includes a random effect for each level 2 unit (core). The variance components model is the following.

$$\begin{aligned} \text{Level 1:} &\quad \text{Ext. rate}_{ij} = \beta_{0i} + \varepsilon_{ij} \\ \text{Level 2:} &\quad \beta_{0i} = \beta_0 + u_{0i} \end{aligned}$$

where $\varepsilon_{ij} \sim \text{iid } N(0, \sigma^2), u_{0i} \sim \text{iid } N(0, \tau^2)$. To understand how the R specification of this model works, I substitute the level-2 equation into the level-1 equation to obtain the composite equation.

$$\begin{aligned} \text{composite equation}: \text{Ext. rate}_{ij} &= \beta_0 + u_{0i} + \varepsilon_{ij} \\ &= \underbrace{\beta_0 \cdot 1}_{\text{fixed}} + \underbrace{u_{0i} \cdot 1}_{\text{random}} + \varepsilon_{ij} \end{aligned}$$

This last equation maps directly into R code as shown next.

```
model0 <- lme(fixed=Ext.rate~1, random=~1|Core, data=ext.rates2, method="ML")
```

Variance components are extracted from the model with the **VarCorr** function.

```
VarCorr(model0)
Core = pdLogChol(1)
            Variance    StdDev
(Intercept) 0.007738093 0.08796643
Residual    0.014771638 0.12153863
```

What's reported in the first column are the squares of the values reported in the summary table, so these are variances. From the output we determine that $\hat{\tau}^2 = 0.0077$ and $\hat{\sigma}^2 = 0.0148$. From this we learn a couple of things.

1. Because $\hat{\tau}^2 > 0$, heterogeneity is present. Put another way we have evidence of a correlation structure (as opposed to independent observations).
2. Furthermore because the variability between cores is roughly one half the variability within cores and hence is not negligible, level-2 modeling in which predictors are included to explain differences between cores is likely to be productive.

We can use the output from **VarCorr** to calculate the intraclass correlation coefficient that was derived in lecture 19.

$$\rho = \frac{\tau^2}{\tau^2 + \sigma^2}$$

VarCorr has not produced a matrix, but a table, and the elements of the table are actually character values.

```
VarCorr(model0)[1,1]
[1] "0.007738093"
```

As a result our calculations of the correlation are rather awkward. I need to convert each character value to a numeric value before I can do arithmetic on it. (It's probably easier to just copy and paste the numbers.)

```
tau.sq <- as.numeric(VarCorr(model0)[1,1])
sigma.sq <- as.numeric(VarCorr(model0)[2,1])
tau.sq/(tau.sq+sigma.sq)
[1] 0.3437666
```

So the correlation between observations coming from the same core is 0.34, a modest value and far from 0. The implication is that the independence assumption of ordinary linear regression is not valid for these data.

Fitting mixed effects models with the lme4 package

The **lme4** package is slated to eventually replace the **nlme** package. It extends normal mixed effects models to other members of the exponential family of distributions, thus permitting the fitting of generalized linear mixed effects models. Not all features are fully implemented so at the moment the **nlme** package has functionality not present in **lme4**. The **lme4** package is not part of the standard installation of R, so it must first be downloaded from the CRAN site.

In **lme4** the **lmer** function replaces **lme**. The major change in syntax is that there is no longer a random argument; the random effects are now written as part of the model formula. A second change is that to obtain maximum likelihood estimates one must specify the argument **REML=FALSE**.

Here are how some of the earlier **lme** models would be fit using **lmer**. The functions in the **nlme** package conflict with those inside **lme4**, so it's a good idea to first **detach** the **nlme** package before loading **lme4**.

```
detach(package:nlme)
library(lme4)
#random intercepts model, model1 from before
out1.lmer <- lmer(Ext.rate ~ Year + (1|Core), data=ext.rates2, REML=F)
#random slopes and intercepts, model2 from before
out2.lmer <- lmer(Ext.rate ~ I(Year-1967) + (I(Year-1967)|Core), data=ext.rates2, REML=F)
#reef.type as a predictor of random slopes, model4 from before
```

```
out4.lmer <- lmer(Ext.rate ~ I(Year-1967) + I(Year-1967):factor(reef.type) + (I(Year-1967)|Core),
   data=ext.rates2, REML=F)
```

A number of additional models are readily fit with the **lmer** function. For example, we can fit a random slopes and intercepts model in which the correlation of the two sets of random effects is constrained to be zero. Fitting such a model is rarely a good idea.

```
#uncorrelated random slopes and intercepts model using centered year
out5.lmer <- lmer(Ext.rate~I(Year-1967) + (1|Core)+(I(Year-1967)-1|Core), data=ext.rates2,
   REML=FALSE)
```

Cited References

- Pinheiro, J. C. and Bates, D. M. 2000. *Mixed-Effects Models in S and S-Plus*. Springer-Verlag, New York.

Appendix: Code for Figs. 2, 6, and 8

Figure 2

```
library(nlme)
#random intercepts
model1 <- lme(Ext.rate~Year, random=~1|Core, data=ext.rates2, method='ML')
plot(Ext.rate~ Year, data=ext.rates2, type='n', ylab='Extension rate')
mycols <- rep(c('grey80', 'seagreen1', 'grey60', 'bisque4', 'pink'), 5)
#vector of unique cores
core.list <- unique(ext.rates2$Core)
#matrix of Year starts and stops
year.limits <- matrix(unlist(tapply(ext.rates2$Year, ext.rates2$Core,range)), ncol=2, byrow=T)
#draw observed trajectories
for(i in 1:length(core.list)) {
  cur.core <- ext.rates2[ext.rates2$Core==core.list[i],]
  lines(cur.core$Year, cur.core$Ext.rate, lty=1, col=mycols[i])
}
#matrix of random intercepts
new.ran1 <- cbind(year.limits, ranef(model1)+ fixef(model1)[1])
#subject-specific curves
apply(new.ran1, 1, function(y) curve(y[3]+ fixef(model1)[2]*x, col='dodgerblue4', add=T, from=y[1],
   to=y[2])) -> my.curves
#population averaged curve
abline(c(fixef(model1)[1], fixef(model1)[2]), col='red', lwd=2)
```

```
mtext('random intercepts model', side=3, line=.5)
legend('topright', c('population average', 'subject specific'), col=c(2, 'dodgerblue4'), lty=1,
   cex=.9, bty='n', lwd=c(2,1))
#add text and arrows
new.ran1[,1] -> starts
new.ran1[,3] -> ran.ints
y2.location <- ran.ints[2]+ fixef(model1)[2]* starts[2]
arrows(starts[2]+1, y2.location-.05, starts[2]+1, y2.location+.05, code=3, angle=45, length=.05,
   col=4, lwd=2)
text(starts[2], y2.location-.05, expression(sigma^2), pos=4, cex=.9, col=4)
y1.location <- ran.ints[1]+ fixef(model1)[2]* starts[1]
arrows(starts[1]+1, y1.location-.05, starts[1]+1, y1.location+.05, code=3, angle=45, length=.05,
   col=4, lwd=2)
text(starts[1]-1, y1.location, expression(sigma^2), pos=3, cex=.9, col=4)
par(xpd=T)
y.location <- fixef(model1)[1]+ fixef(model1)[2]* par("usr")[2]
arrows(par("usr")[2]+2, y.location-.1, par("usr")[2]+2, y.location+.1, code=3, angle=45, length=.05,
   lwd=2)
text(par("usr")[2]+2, y.location+.1, expression(tau^2), pos=3, cex=.9)
par(xpd=F)
```

Figure 6

```
library(nlme)
#random slopes and intercepts
model2 <- lme(Ext.rate~I(Year-1967), random=~I(Year-1967)| Core, data=ext.rates2, method='ML')
plot(Ext.rate~ Year, data=ext.rates2, type='n', ylab='Extension rate')
mycols <- rep(c('grey80', 'seagreen1', 'grey60', 'bisque4', 'pink'), 5)
#vector of unique cores
core.list <- unique(ext.rates2$Core)
#matrix of Year starts and stops
year.limits <- matrix(unlist(tapply(ext.rates2$Year, ext.rates2$Core,range)), ncol=2, byrow=T)
#draw observed trajectories
for(i in 1:length(core.list)) {
  cur.core<-ext.rates2[ext.rates2$Core==core.list[i],]
  lines(cur.core$Year, cur.core$Ext.rate, lty=1, col=mycols[i])
}
#matrix of random slopes and intercepts
new.ran2<-cbind(year.limits, ranef(model2)[,1]+ fixef(model2)[1]-1967* (ranef(model2)[,2]+
   fixef(model2)[2]), ranef(model2)[,2]+ fixef(model2)[2])
```

```
#subject-specific curves
apply(new.ran2, 1, function(y) curve(y[3]+y[4]*x, col='dodgerblue4', add=T, from=y[1], to=y[2])) ->
   my.curve
#population averaged curve
abline(c(fixef(model1)[1]-1967* fixef(model1)[2], fixef(model1)[2]), col='red', lwd=2)
mtext('random intercepts and slopes model', side=3, line=.5)
legend('topright', c('population average', 'subject specific'), col=c(2, 'dodgerblue4'), lty=1,
   cex=.9, bty='n', lwd=c(2,1))
#add text and arrows
new.ran2[,1] -> starts
new.ran2[,3] -> ran.ints2
new.ran2[,4] -> ran.slopes
#random intercept variances
y1.location <- ran.ints2[1]+ran.slopes[1]*1998
arrows(1998, y1.location-.05, 1998, y1.location+.05, code=3, angle=45, length=.05, col=4, lwd=2)
text(1999.2, y1.location+.05, expression(sigma^2), pos=2, cex=.9, col=4)
y2.location <- ran.ints2[2]+ ran.slopes[2]* starts[2]
arrows(starts[2]+1, y2.location-.05, starts[2]+1, y2.location+.05, code=3, angle=45, length=.05,
   col=4, lwd=2)
text(starts[2]-1, y2.location, expression(sigma^2), pos=3, cex=.9, col=4)
#random slope variances
y3.location<-ran.ints2[5]+ran.slopes[5]*(starts[5]+3)
arrows(starts[5]+2, y3.location-.07, starts[5]+4, y3.location+.07, code=3, angle=45, length=.05,
   lwd=2)
text(starts[5]+4, y3.location+.07, expression(tau[1]^2), pos=2, cex=.9)
par(xpd=T)
y.location <- fixef(model1)[1]-1967* fixef(model1)[2]+ fixef(model1)[2]* par("usr")[2]
arrows(par("usr")[2]+2, y.location-.1, par("usr")[2]+2, y.location+.1, code=3, angle=45, length=.05,
   lwd=2)
text(par("usr")[2]+2.2, y.location+.1, expression(tau[0]^2), pos=3, cex=.9)
par(xpd=F)
```

Figure 8

```
library(nlme)
#slopes vary by reef type: random slopes and intercepts
model4<-lme(Ext.rate~I(Year-1967) + I(Year-1967):factor(reef.type), random=~I(Year-1967)|Core,
   data=ext.rates2, method='ML')
#vector of unique cores
core.list <- unique(ext.rates2$Core)
```

```
#matrix of Year starts and stops
year.limits <- matrix(unlist(tapply(ext.rates2$Year, ext.rates2$Core,range)), ncol=2, byrow=T)
#define function to draw graph for a given reef type
graph.reeftype<-function(k) {
  reef.vals <- tapply(as.numeric(factor(ext.rates2$reef.type)), ext.rates2$Core, function(x) x[1])
  reef.vals <- reef.vals[!is.na(reef.vals)]
  reef.names <- c('Backreef', 'Forereef', 'Nearshore')
  reeftype.list <- c('BR', 'FR', 'NS')
  #extract data for requested reef type
  ext.rates4 <- ext.rates2[ext.rates2$reef.type==reeftype.list[k],]
  new.core.list <- core.list[reef.vals==k]
  plot(Ext.rate~ Year, data=ext.rates4, type='n', ylab='Extension rate', ylim=c(min(Ext.rate),
     max(Ext.rate)+.05))
  mycols <- rep(c('grey80', 'seagreen2', 'bisque4', 'pink', 'grey60'), 5)
  #draw observed trajectories
  for(i in 1:length(new.core.list)) {
    cur.core <- ext.rates4[ext.rates4$Core==new.core.list[i],]
    lines(cur.core$Year, cur.core$Ext.rate, lty=1, col=mycols[i])
  }
  #matrix of random slopes and intercepts
  new.ran2 <- cbind(year.limits, ranef(model4)[,1]+ fixef(model4)[1]-1967* (ranef(model4)[,2]
     +fixef(model4)[2]+ (reef.vals==2)*fixef(model4)[3]+ (reef.vals==3)*fixef(model4)[4]),
     ranef(model4)[,2]+ fixef(model4)[2]+ (reef.vals==2)*fixef(model4)[3]+
     (reef.vals==3)*fixef(model4)[4], reef.vals)
  new.ran2a <- new.ran2[new.ran2[,"reef.vals"]==k,]
  sub.cols <- c('grey40', 'dodgerblue4', 'tomato')
  #subject-specific curves
  apply(new.ran2a, 1, function(y) curve(y[3]+y[4]*x, add=T, from=y[1], to=y[2], col='dodgerblue4'))
  #population averaged line
  sapply(k,function(x) abline(c(fixef(model4)[1]-1967* (fixef(model4)[2]+ (x==2)*fixef(model4)[3]+
     (x==3)*fixef(model4)[4]), fixef(model4)[2]+ (x==2)*fixef(model4)[3]+ (x==3)*fixef(model4)[4]),
     col='red', lwd=2))
  mtext(reef.names[k], side=3, line=.5, cex=.9)
  legend('topright', c('population average', 'subject-specific'), col=c(2, 'dodgerblue4'), lty=1,
     cex=.85, bty='n', lwd=c(2,1))
}
graph.reeftype(1)
graph.reeftype(2)
graph.reeftype(3)
```

Jack Weiss
Phone: (919) 962-5930
E-Mail: jack_weiss@unc.edu
Address: Curriculum for the Environment and Ecology, Box 3275, University of North Carolina, Chapel Hill, 27599
Copyright © 2012
Last Revised--February 26, 2012
URL: https://sakai.unc.edu/access/content/group/2842013b-58f5-4453-aa8d-3e01bacbfc3d/public/Ecol562_Spring2012/docs/lectures/lecture20.htm