1. Is a mixed model right for your needs?

A mixed model is similar in many ways to a linear model. It estimates the effects of one or more explanatory variables on a response variable. The output of a mixed model will give you a list of explanatory values, estimates and confidence intervals of their effect sizes, p-values for each effect, and at least one measure of how well the model fits. You should use a mixed model instead of a simple linear model when you have a variable that describes your data sample as a subset of the data you could have collected.

What do I mean by that? Let's take a look at some preliminary data from a paper wasp kin recognition project I'm working on.

```
str(recog)
```

```
## 'data.frame':        84 obs. of  6 variables:
##  $ Test.ID   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Observer  : Factor w/ 4 levels "Charles","Michelle",..: 1 4 2 4 1 3 2 2 1 2 ...
##  $ Relation  : Factor w/ 2 levels "Same","Stranger": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Aggression: int  4 1 15 2 1 0 2 0 3 10 ...
##  $ Tolerance : int  4 34 14 31 4 13 7 6 13 15 ...
##  $ Season    : Factor w/ 2 levels "Early","Late": 1 1 1 1 1 1 1 1 1 1 ...
```

My response variables of interest are Aggression and Tolerance. Aggression is the number of aggressive behaviors in a sixty minute period. Tolerance is the number of tolerant behaviors in a sixty minute period. I am interested in the effects of relation (whether the wasps came from the same or different colonies) and season (early or late in the colony cycle) on these response variables. These effects are "fixed" because no matter where, how, or how many wasps I had sampled, I would have still have had the same levels in the same variables: same colony vs. different colony, and early season vs. late season.

There are two other variables, though, that would not remain fixed between samples. Observer represents the person who scored the interaction and Test.ID represents the group of three wasps observed for sixty minutes. The levels of Observer would be different if I had sampled in a different year, because different undergraduate volunteers would be available to observe behavior. The levels of Test ID would also vary between samples, because I could always rearrange which wasps participate in each experimental trial. Each trial is a unique sub-sample of the wasps I collected at that time. If I had been able to test the wasps individually, and if all observers had scored all interactions, I wouldn't have any random effects. But instead, my data are inherently "lumpy," and the random effects describe that lumpiness. Mixed models allow us to account for the lumpiness of data.

Before you proceed, you will also want to think about the structure of your random effects. Are your random effects nested or crossed? In the case of my study, the random effects are *nested*, because each observer recorded a certain number of trials, and no two observers recorded the same trial, so here Test.ID is nested within Observer. But say I had collected wasps that clustered into five different genetic lineages. The 'genetics' random effect would have nothing to do with observer or arena; it would be orthogonal to these other two random effects. Therefore this random effect would be *crossed* to the others.

2. What probability distribution best fits your data?

Say you've decided you want to run a mixed model. The next thing you have to do is find a probability distribution that best fits your data. There are many ways to test this, but I'll show you one. (There are also graphical methods such as QQ plots you can use to achieve the same result.) Note that the negative binomial and gamma distributions can only handle positive numbers, and the Poisson distribution can only handle positive whole numbers. Now let's find a fitting distribution for my Aggression variable.

```
# This is so that distributions that must be non-zero can make sense of my
# data
recog$Aggression.t <- recog$Aggression + 1
normal <- fitdistr(recog$Aggression.t, "normal")
logLik(normal)
```

```
## 'log Lik.' -307.8 (df=2)
```

```
lognormal <- fitdistr(recog$Aggression.t, "lognormal")
logLik(lognormal)
```

```
## 'log Lik.' -205 (df=2)
```

```
nbinom <- fitdistr(recog$Aggression.t, "Negative Binomial")
```

```
logLik(nbinom)
```

```
## 'log Lik.' -229.3 (df=2)
```

```
poisson <- fitdistr(recog$Aggression.t, "Poisson")
logLik(poisson)
```

```
## 'log Lik.' -427.3 (df=1)
```

```
gamma <- fitdistr(recog$Aggression.t, "gamma")
logLik(gamma)
```

```
## 'log Lik.' -221.8 (df=2)
```

The fit for the lognormal distribution has the lowest log likelihood, which means it has the best fit of the probability distributions I tested. Now I'm armed with the probability distributions I need to run my mixed model.

3. How to fit a mixed model to your data

3a. If your data are normally distributed

First, a note: if your data best fit the lognormal distribution, *do not transform them*. This is true for *any* type of transformation you might apply to your data to make them normal. If you can transform your data to normality, common wisdom says you should use the transformed data. More recent statistics literature has entirely changed stance on this matter, however, because transformation makes interpretation of model results more difficult, and it makes mischief with the variance of the transformed variable. Even if your data are transformable to normality, they are still *not normal*, and you should move on to the next section.

If your data are normally distributed, your life will be a little easier, because you can use a linear mixed model (LMM). You will want to load the lme4 package and make a call to the function lmer. The first argument to the function is a formula that takes the form y ~ x1 + x2 ... etc., where y is the response variable and x1, x2, etc. are explanatory variables. Random effects are added in with the explanatory variables. Crossed random effects take the form (1 | r1) + (1 | r2) ... while nested random effects take the form (1 | r1 / r2).

The next argument is where you designate the data frame your variables come from. The argument after that is an important one. This is where you can designate whether the mixed model will estimate the parameters using maximum likelihood or restricted maximum likelihood. If your random effects are nested, or you have only one random effect, and if your data are balanced (i.e., similar sample sizes in each factor group) set REML to FALSE, because you can use maximum likelihood. If your random effects are crossed, don't set the REML argument because it defaults to TRUE anyway.

Lest this all seem too abstract, let's try this with some data. We will use some data from a paper I recently published with some colleagues about immunocompetence in paper wasps. In this study, we were interested in the effects of sex, season, and ploidy on encapsulation response (ER), a measure of immune system activity. Our random effect was colony of origin. Encapsulation response fits a normal probability distribution.

```
str(immunity)
```

```
## 'data.frame':        103 obs. of  6 variables:
##  $ sex   : Factor w/ 2 levels "Female","Male": 1 1 2 1 1 1 1 1 1 1 ...
##  $ season: Factor w/ 2 levels "Spring","Fall": 1 1 1 1 1 1 1 1 1 1 ...
##  $ year  : int  2008 2008 2008 2008 2008 2008 2008 2008 2008 2008 ...
##  $ colony: Factor w/ 107 levels "        ","ad1     ",..: 7 7 7 8 8 9 10 11 11 12 ...
##  $ ploidy: Factor w/ 3 levels "Haploid","Diploid",..: 2 2 1 2 2 2 2 2 2 2 ...
##  $ ER    : num  82.3 46.6 55.1 110.2 59.9 ...
```

```
lmm <- lmer(ER ~ sex + season + ploidy + (1 | colony), data = immunity, REML = FALSE)
summary(lmm)
```

```
## Linear mixed model fit by maximum likelihood
## Formula: ER ~ sex + season + ploidy + (1 | colony)
##    Data: immunity
##  AIC BIC logLik deviance REMLdev
##  954 969   -471      942     920
```

```
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  colony   (Intercept) 293      17.1
##  Residual             355      18.8
## Number of obs: 103, groups: colony, 45
##
## Fixed effects:
##               Estimate Std. Error t value
## (Intercept)     71.496     10.209    7.00
## sexMale         -2.139      7.589   -0.28
## seasonFall     -17.479      7.951   -2.20
## ploidyDiploid    0.024      9.264    0.00
##
## Correlation of Fixed Effects:
##             (Intr) sexMal ssnFll
## sexMale     -0.658
## seasonFall  -0.370 -0.094
## ploidyDipld -0.932  0.661  0.267
```

Let's look at these results. First we get some measures of model fit, including AIC, BIC, log likelihood, and deviance. Then we get an estimate of the variance explained by the random effect. This number is important, because if it's indistinguishable from zero, then your random effect probably doesn't matter and you can go ahead and do a regular linear model instead. Next we have estimates of the fixed effects, with standard errors. This information may be enough for you. Some journals like you to report the results of these models as effect sizes with confidence intervals. Certainly when I look at these fixed effects estimates, I can see that season is the only fixed effect that changes the encapsulation response. But some journals want you to report p-values.

The creators of the lme4 package are philosophically opposed to p-values, for reasons I'll not go into here, so if you want some p-values you'll have to turn to another package, LMERConvenienceFunctions.

```
library(LMERConvenienceFunctions)
pamer.fnc(lmm)
```

```
##        Df    Sum Sq   Mean Sq F value upper.den.df upper.p.val
## sex     1  623.3178  623.3178   1.756           99      0.1882
## season  1 1847.9859 1847.9859   5.205           99      0.0247
## ploidy  1    0.0024    0.0024   0.000           99      0.9980
##        lower.den.df lower.p.val expl.dev.(%)
## sex              54      0.1908        0.727
## season           54      0.0265        2.155
## ploidy           54      0.9980        0.000
```

There's a lot going on in this output, but I recommend that you report the F value, the "lower.den.df" for degrees of freedom, and "lower.p.val," a more conservative estimate of p value. My philosophical stance is that I'd rather make a Type II error than a Type I error.

There is one complication you might face when fitting a linear mixed model. R may throw you a "failure to converge" error, which usually is phrased "iteration limit reached without convergence." That means your model has too many factors and not a big enough sample size, and cannot be fit. Unfortunately, I don't have any data that actually fail to converge on a model that I can show you, but let's pretend that last model didn't converge. What you should then do is drop fixed effects and random effects from the model and compare to see which fits the best.

```
noploidylmm <- lmer(ER ~ sex + season + (1 | colony), data = immunity, REML = FALSE)
nosexlmm <- lmer(ER ~ season + ploidy + (1 | colony), data = immunity, REML = FALSE)
noseasonlmm <- lmer(ER ~ sex + ploidy + (1 | colony), data = immunity, REML = FALSE)
anova(noploidylmm, nosexlmm, noseasonlmm)
```

```
## Data: immunity
## Models:
## noploidylmm: ER ~ sex + season + (1 | colony)
## nosexlmm: ER ~ season + ploidy + (1 | colony)
## noseasonlmm: ER ~ sex + ploidy + (1 | colony)
##             Df AIC BIC logLik Chisq Chi Df Pr(>Chisq)
## noploidylmm  5 952 965   -471
## nosexlmm     5 952 965   -471     0      0          1
## noseasonlmm  5 956 969   -473     0      0          1
```

The p value less than 0.05 and the asterisk indicate that the model that includes only season and ploidy as fixed effects fits the data best. Another approach is to use summary() on the three models and compare the AIC values and pick whichever one is lowest. This will

come up with the same result in almost all cases. Whatever you choose, be sure to report in your manuscript the p values or AIC values you used to pick the best model.

3b. If your data are not normally distributed

This is where life gets interesting. You see, the REML and maximum likelihood methods for estimating the effect sizes in the model make assumptions of normality that don't apply to your data, so you have to use a different method for parameter estimation. The problem is that there are many alternative estimation methods, each run from a different R package, and it can be hard to decide which one suits. I will guide you through this decision process with examples.

First, we need to test whether we can use penalized quasilikelihood (PQL) or not. PQL is a flexible technique that can deal with non-normal data, unbalanced design, and crossed random effects. However, it produces biased estimates if your response variable fits a discrete count distribution, like Poisson or binomial, and the mean is less than 5 - or if your response variable is binary.

The Aggression variable fits the log-normal distribution, which is not a discretized distribution. That means we can proceed with the PQL method. But before we proceed, let's return to the matter of transformation to normality. The reason we want to use a GLMM for this is that if we imagine a stastical method as E(x), E(ln(x)) is not the same as ln(E(x)). The former is performing a LMM on a transformed variable, while the latter is performing a GLMM on an untransformed variable. The latter is better because it better captures the variance of x.

Make sure you have the MASS package loaded. Note that instead of taking all the fixed and random effects as one formula, the random effects get their own argument in the glmmPQL function. To set the distribution to log-normal, we set the family to gaussian (another word for normal) and the link to log. The link can be anything, though if you want to use something besides log or inverse then you'll have to research how to customize the link function yourself.

```
PQL <- glmmPQL(Aggression.t ~ Relation + Season, ~1 | Observer/Test.ID, family = gaussian(link = "log"),
    data = recog, verbose = FALSE)
```

```
## Loading required package: nlme
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:lme4':
##
##     fixef, lmList, ranef, VarCorr
```

```
summary(PQL)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: recog
##    AIC BIC logLik
##     NA  NA     NA
##
## Random effects:
##  Formula: ~1 | Observer
##         (Intercept)
## StdDev:      0.3312
##
##  Formula: ~1 | Test.ID %in% Observer
##         (Intercept) Residual
## StdDev:      0.5295    7.128
##
## Variance function:
##  Structure: fixed weights
##  Formula: ~invwt
## Fixed effects: Aggression.t ~ Relation + Season
##                   Value Std.Error DF t-value p-value
## (Intercept)       1.033    0.5233 55   1.974  0.0535
## RelationStranger  1.210    0.4674 55   2.589  0.0123
## SeasonLate       -1.333    0.5983 23  -2.228  0.0359
##  Correlation:
##                  (Intr) RltnSt
## RelationStranger -0.855
## SeasonLate       -0.123  0.000
##
## Standardized Within-Group Residuals:
##      Min       Q1      Med       Q3      Max
## -4.86916 -0.29958 -0.08012  0.14280  5.93336
```

```
##
## Number of Observations: 84
## Number of Groups:
##              Observer Test.ID %in% Observer
##                     4                    28
```

This model suggests that season has an effect on Aggression, that is, wasps collected late in the colony cycle were less aggressive than those collected early. It also suggests that the relationship between the wasps has an effect; they are more likely to be aggressive toward strangers than nestmates. I would report these statistics in a paper with the estimate, standard error, t-value, and p-value.

So what if the mean of your response variable is less than 5, or you have a binary response variable, and you can't use PQL? Here you're going to have to make another decision, because there are two alternatives you can use: the Laplace approximation and Markov chain Monte Carlo algorithms (MCMC). The Laplace approximation can handle up to 3 random effects. Any more than that, and you'll have to use MCMC, which is a Bayesian method that can be somewhat confusing.

Let's start with an example where we can use the Laplace approximation. We will use a sample dataset from the package mlmRev, which gives us data about how well some Dutch students did in school. For the purposes of this example, I will subset the data to only a few variables of interest, and simplify the "repeatgr" variable to a binary response.

```
library("mlmRev")
```

```
##
## Attaching package: 'mlmRev'
##
## The following object is masked from 'package:nlme':
##
##     bdf, Oxboys
```

```
data(bdf, package = "mlmRev")
bdf <- subset(bdf, select = c(schoolNR, Minority, ses, repeatgr))
bdf$repeatgr[bdf$repeatgr == 2] <- 1
str(bdf)
```

```
## 'data.frame':        2287 obs. of  4 variables:
##  $ schoolNR: Factor w/ 131 levels "1","2","10","12",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Minority: Factor w/ 2 levels "N","Y": 1 2 1 1 1 2 2 1 2 2 ...
##  $ ses     : num  23 10 15 23 10 10 23 10 13 15 ...
##  $ repeatgr: Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 2 1 ...
```

Let's say we want to find out whether membership in a racial minority and socioeconomic status affect students' likelihood to repeat a grade. Our response variable is 'repeatgr', a binary response indicating whether the student repeated a grade or not. Racial minority status is a binary Y/N category and socioeconomic status is represented by 'ses', a numeric scale that ranges from 10 to 50, where 50 is the richest. Our random factor is 'schoolNR', which represents the schools from which the students were sampled. Because the response variable is binary, we will need a generalized linear mixed model with a binomial distribution, and because we have fewer than five random effects, we can use the Laplace approximation.

Strictly speaking, the Laplace approximation is a special case of a parameter estimation method called Gauss-Hermite quadrature (GHQ), with one iteration. GHQ is more accurate than Laplace due to repeated iterations, but becomes less flexible after the first iteration, so you can only use it for one random effect. We can use it in this example because our only random effect is 'schoolNR.' To go ahead with this method, we use the lme4 package again.

```
GHQ <- glmer(repeatgr ~ Minority + ses + ses * Minority + (1 | schoolNR), data = bdf,
    family = binomial(link = "logit"), nAGQ = 50)  # Set nAGQ to # of desired iterations
summary(GHQ)
```

```
## Generalized linear mixed model fit by the adaptive Gaussian Hermite approximation
## Formula: repeatgr ~ Minority + ses + ses * Minority + (1 | schoolNR)
##    Data: bdf
##   AIC  BIC logLik deviance
##  1673 1701   -831     1663
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  schoolNR (Intercept) 0.264    0.514
## Number of obs: 2287, groups: schoolNR, 131
##
## Fixed effects:
```

```
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.45194    0.20816   -2.17     0.03 *
## MinorityY      0.47958    0.46984    1.02     0.31
## ses           -0.06205    0.00794   -7.81  5.6e-15 ***
## MinorityY:ses  0.01196    0.02288    0.52     0.60
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) MnrtyY ses
## MinorityY   -0.406
## ses         -0.915  0.387
## MinortyY:ss  0.303 -0.867 -0.335
```

```
Laplace <- glmer(repeatgr ~ Minority + ses + ses * Minority + (1 | schoolNR),
    data = bdf, family = binomial(link = "logit"))  # Contrast to the Laplace approximation
summary(Laplace)
```

```
## Generalized linear mixed model fit by the Laplace approximation
## Formula: repeatgr ~ Minority + ses + ses * Minority + (1 | schoolNR)
##    Data: bdf
##   AIC  BIC logLik deviance
##  1673 1701   -831     1663
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  schoolNR (Intercept) 0.259    0.509
## Number of obs: 2287, groups: schoolNR, 131
##
## Fixed effects:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.45456    0.20793   -2.19    0.029 *
## MinorityY      0.48035    0.46937    1.02    0.306
## ses           -0.06194    0.00794   -7.80    6e-15 ***
## MinorityY:ses  0.01193    0.02285    0.52    0.602
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) MnrtyY ses
## MinorityY   -0.406
## ses         -0.915  0.387
## MinortyY:ss  0.304 -0.867 -0.335
```

You can see there's no important difference between Laplace and GHQ in this case. Both show that socioeconomic class has a highly significant effect on students' likelihood to repeat a grade, though even with the logit transformation we can see that the effect size is small. There is one more consideration, though, when using this method. It becomes inaccurate when used on overdispersed data - that is, when the combined residuals are much larger than the residual degrees of freedom. When you use this method, you should check the model to make sure the data are not overdispersed. Here is some code that will help you with that.

```
overdisp_fun <- function(model) {
    ## number of variance parameters in an n-by-n variance-covariance matrix
    vpars <- function(m) {
        nrow(m) * (nrow(m) + 1)/2
    }
    # The next two lines calculate the residual degrees of freedom
    model.df <- sum(sapply(VarCorr(model), vpars)) + length(fixef(model))
    rdf <- nrow(model.frame(model)) - model.df
    # extracts the Pearson residuals
    rp <- residuals(model, type = "pearson")
    Pearson.chisq <- sum(rp^2)
    prat <- Pearson.chisq/rdf
    # Generates a p-value. If less than 0.05, the data are overdispersed.
    pval <- pchisq(Pearson.chisq, df = rdf, lower.tail = FALSE)
    c(chisq = Pearson.chisq, ratio = prat, rdf = rdf, p = pval)
}
```

Save this code as an R script and source it. Use the function on the model you've created. These school data are not overdispersed, if yours are, don't panic. You can use a quasi distribution instead. Quasi distributions can handle overdispersion. Just run the model again with "quasibinomial" as the distribution instead.

Let's move on to the case where we can't use glmmPQL (i.e., because the mean of Poisson data is too small or because the response

variable is categorical) and we have five or more random effects. I couldn't find a sample dataset for this, so instead we will turn the variable of interest in a sample dataset into a binary response. Consider these data about barley farmers. Imagine that for a barley harvest to yield a profit, the income from that barley harvest must be greater than 140.

```
library(agridat)
```

```
## Loading required package: grid
## Loading required package: reshape2
```

```
data(student.barley, package = "agridat")
profit <- as.numeric(student.barley$income > 140)
farmers <- cbind(student.barley, profit)
farmers <- farmers[c(1:5, 8)]
str(farmers)
```

```
## 'data.frame':        102 obs. of  6 variables:
##  $ year    : int  1901 1901 1901 1901 1902 1902 1902 1902 1902 1902 ...
##  $ farmer  : Factor w/ 18 levels "Allardyce","Dooley",..: 10 5 3 18 10 5 18 17 4 12 ...
##  $ place   : Factor w/ 16 levels "Arnestown","Bagnalstown",..: 3 16 14 11 3 16 11 4 8 6 ...
##  $ district: Factor w/ 4 levels "CentralPlain",..: 2 2 1 1 2 2 1 1 4 4 ...
##  $ gen     : Factor w/ 2 levels "Archer","Goldthorpe": 1 1 1 1 1 1 1 1 1 1 ...
##  $ profit  : num  1 1 1 1 1 1 1 1 1 1 ...
```

In this case, let's say we have no explanatory variables at all. We don't have any ideas about what fixed effects might influence whether a barley harvest turns a profit or not. We're interested in which random effects contribute to the variability of profit. After all, random effects are factors that change the variance of a response variable; sometimes we're trying to account for that variance to make the fixed effects clearer, but sometimes we're interested in the variances of fixed effects for their own sake. To do this, we will use MCMCglmm, which can not only handle many random effects, but provides confidence intervals for the random effects, which none of the other packages we've used here provide in their summary().

The confusing part about MCMCglmm is that it is a Bayesian statistical method. All models make assumptions about the distribution of the variance in your data, but in a Bayesian method these assumptions are explicit, and we need to specify these assumed distributions. In Bayesian statistics, we call these *priors*. The priors I've used here are bog-standard and will work for most data. Feel free to use them yourself. Just keep in mind that one R structure needs to be specified for each fixed effect and one G structure needs to be specified for each random effect.

```
library(MCMCglmm)
```

```
## Loading required package: tensorA
##
## Attaching package: 'tensorA'
##
## The following object is masked from 'package:Matrix':
##
##     norm
##
## The following object is masked from 'package:base':
##
##     norm
##
## Loading required package: coda
##
## Attaching package: 'coda'
##
## The following object is masked from 'package:lme4':
##
##     HPDinterval
##
## Loading required package: ape
## Loading required package: corpcor
```

```
prior = list(R = list(V = 1, n = 0, fix = 1), G = list(G1 = list(V = 1, n = 1),
    G2 = list(V = 1, n = 1), G3 = list(V = 1, n = 1), G4 = list(V = 1, n = 1),
    G5 = list(V = 1, n = 1)))
set.seed(45)
MCMC <- MCMCglmm(profit ~ 1, random = ~year + farmer + place + gen + district,
    data = farmers, family = "categorical", prior = prior, verbose = FALSE)
```

```
summary(MCMC)
```

```
##
##  Iterations = 3001:12991
##  Thinning interval  = 10
##  Sample size  = 1000
##
##  DIC: 76.61
##
##  G-structure:  ~year
##
##      post.mean l-95% CI u-95% CI eff.samp
## year      18.2    0.266     68.6     18.1
##
##                ~farmer
##
##        post.mean l-95% CI u-95% CI eff.samp
## farmer      1.93   0.0789     6.68     99.7
##
##                ~place
##
##       post.mean l-95% CI u-95% CI eff.samp
## place      1.54   0.0711     4.84      173
##
##                ~gen
##
##     post.mean l-95% CI u-95% CI eff.samp
## gen      12.1   0.0874     28.6     1000
##
##                ~district
##
##          post.mean l-95% CI u-95% CI eff.samp
## district      1.78   0.0639     5.63     1000
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units         1        1        1        0
##
##  Location effects: profit ~ 1
##
##             post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)      3.47    -1.16    10.75      220  0.14
```

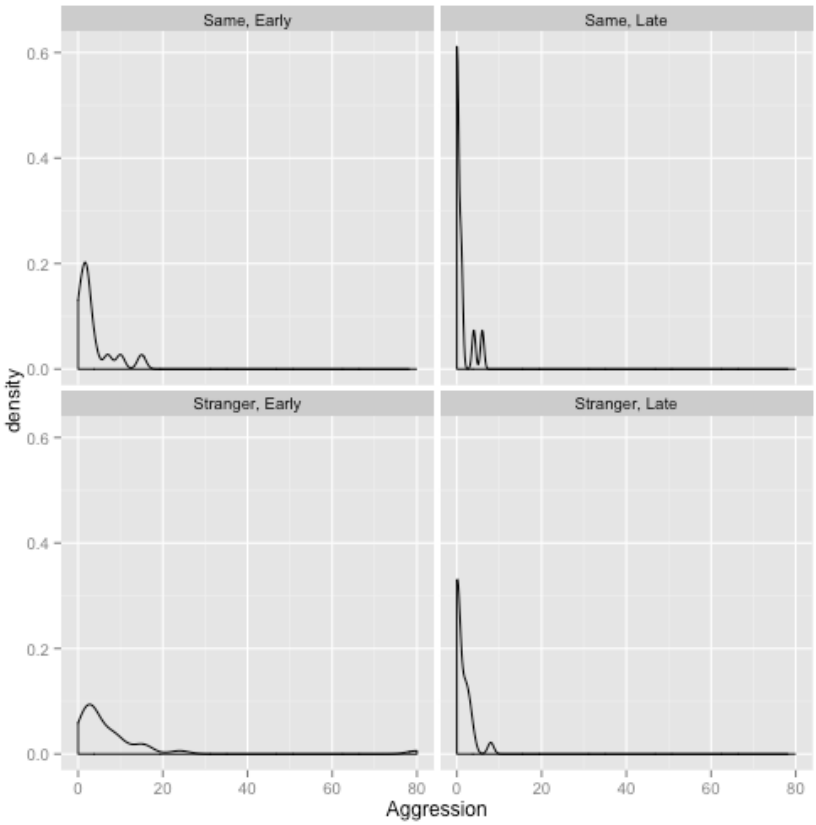
If we look at the means and confidence intervals, we can see that the only random effects that really contributes to the variability of profit are year and genotype. That is to say, the probability of a barley harvest turning a profit varies a lot from year to year, and between genotypes, but it doesn't vary much between districts, farmers, or places.

Closing thoughts: know your data

I've taken you on a whirlwind tour of mixed models. I emphasize this because a lot more goes into data analysis than I've shown you. There are some important steps that go before and after that I don't have space to cover in detail here, but that I'd like to touch upon. Those steps are graphing.

You can't really know which analyses are right for your data until you get familiar with them, and the best way to get familiar with them is to plot them. Usually my first step is to do density plots of my variable of interest, broken down by the explanatory variable I'm most curious about. Density plots are like histograms, except they aren't dependent on how large you make the bins along the x axis. There are a few ways to do make them, but here's how I do it in ggplot2, which makes very pretty graphs.

```
library(ggplot2)
ggplot(recog, aes(x = Aggression)) + geom_density() + facet_wrap(Relation ~
    Season)
```



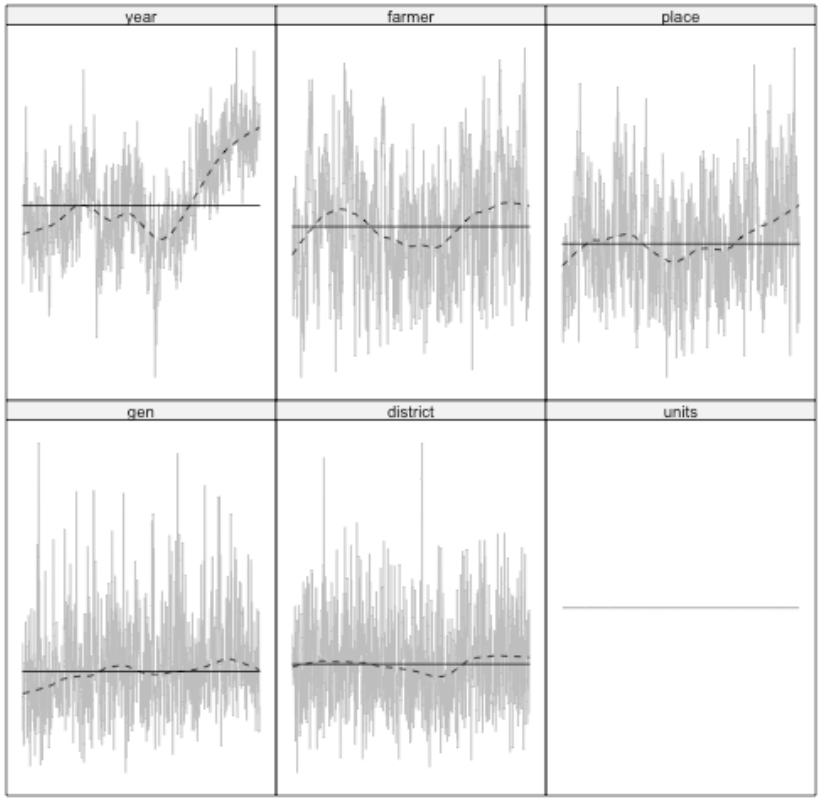
Here I split up my data by season and relation, my two fixed effects. We can see right away that the dataset contains an extreme positive outlier; by far most of the observations fall between 0 and 20 and there's one outlier throwing it off. This is good to know. We can also see that a high proportion of the late season observations are equal to zero.

Plotting is also important for assessing model fit. You can tell which model you fit does the best job describing the data by plotting the fitted values in various ways. Let's try a technique for graphically comparing MCMC models. We'll go back to our barley farmers example and use the scapeMCMC package for a diagnostic plot of the model. We're looking for graphs that resemble white noise around a line.

```
library(scapeMCMC)
```

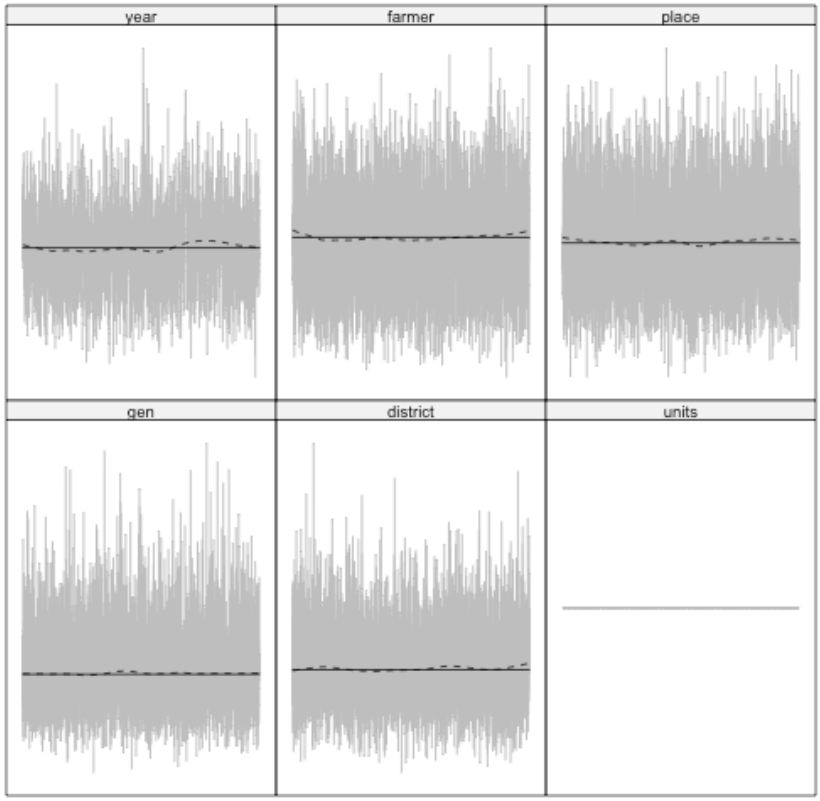
```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

```
plotTrace(MCMC$VCV, log = TRUE)
```



These random effects are looking pretty spiky, not like white noise. So let's try refitting the model with more iterations. This is more computationally intensive, but produces more accurate results.

```
set.seed(55)
MCMC2 <- MCMCglmm(profit ~ 1, random = ~year + farmer + place + gen + district,
    data = farmers, prior = prior, family = "categorical", verbose = FALSE,
    nitt = 5e+05, burnin = 5000, thin = 100)
plotTrace(MCMC2$VCV, log = TRUE)
```

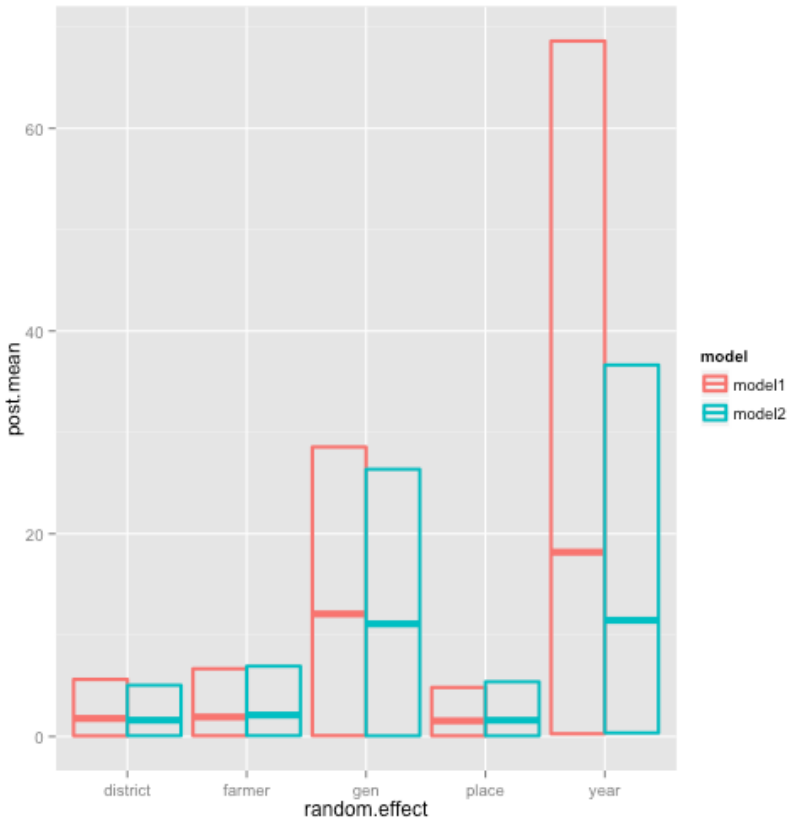


Now everything looks much closer to white noise around a line, which suggests a better model. Now let's compare the confidence intervals for the variance of the random effects between the two models.

```
# glue together the estimates and confidence intervals for the two models
conf.int <- rbind(summary(MCMC)$Gcovariances, summary(MCMC2)$Gcovariances)
# create a data frame with factors for which model and which random effect
conf.int <- data.frame(conf.int, model = factor(rep(c("model1", "model2"), each = 5)),
    random.effect = dimnames(conf.int)[[1]])
```

```
## Warning: some row.names duplicated: 6,7,8,9,10 --> row.names NOT used
```

```
# plot estimates and confidence intervals as box plots grouped by model
ggplot(conf.int, aes(x = random.effect, y = post.mean), group = model) + geom_crossbar(aes(ymax = u.95..CI,
    ymin = l.95..CI, color = model), size = 1, position = "dodge")
```



This is a reassuring plot because the estimates are very similar between the two models (though the estimate for year is a little lower in the second) but the confidence interval for year is markedly smaller in the second model, which means we can be more confident about this estimate. We can feel pretty good about our inferences on the second model, i.e., that genotype and year are the main contributors to variability.

Our brains are best at detecting patterns when they are presented visually, so plot your data and your models whenever you can. Learn to use the base, lattice, or ggplot2 package and it will serve you well for years to come.

Resources

- Bates, D. M. (2010). lme4: Mixed-effects modeling with R. URL http://lme4.r-forge.r-project.org/book.
- The r-sig-mixed-models FAQ, by Ben Bolker (Prof. of Botany and Zoology, University of Florida.)
- Worked examples of GLMMs in R by Ben Bolker
- Bolker, B. M., Brooks, M. E., Clark, C. J., Geange, S. W., Poulsen, J. R., Stevens, M. H. H., & White, J.-S. S. (2009). Generalized linear mixed models: a practical guide for ecology and evolution. Trends in ecology & evolution, 24(3), 127â€“135.
- Lindsey, J. K., & Jones, B. (1998). Choosing among generalized linear models applied to medical data. Statistics in medicine, 17(1), 59â€“68.