Linear Mixed Models (LMMs) Notes

Daniel J. Eck

Contents

Many common statistical models can be expressed as linear models that incorporate both *fixed effects*, which are parameters associated with an entire population or with certain repeatable levels of experimental factors, and *random effects*, which are associated with individual units drawn at random from a population.

A model with both fixed effects and random effects is called a *mixed-effects model*. The random effects in a mixed-effects model can either be *nested* or *crossed*. When the levels of one factor vary only within the levels of another factor, that factor is said to be nested. For example, when measuring the performance of workers at several different job locations, if the workers only work at one location, the workers are nested within the locations. If the workers work at more than one location, then the workers are crossed with locations.

So far in this class we have discussed models with fixed effects only. In this sense, regression model parameters are population level parameters or sub population parameters. Inference for these parameters is at the population level. We will now introduce a simple example of a mixed-effects model.

We will consider a two-way analysis of variance (ANOVA) as a simple example of a mixed-effects model with both fixed and random effects:

$$y_{ijk} = \mu + \tau_i + v_j + \varepsilon_{ijk}$$

where μ a population mean fixed effect, τ_i is a sub population fixed-effect, v_j is a random effect, and ε_{ijk} is an error term. We will assume that $v_j \overset{iid}{\sim} N(0, \sigma_v^2)$ and $\varepsilon_{ijk} \overset{iid}{\sim} N(0, \sigma^2)$. In this model we want to estimate τ_i and test the hypothesis $H_0 : \tau_i = 0$ while we would estimate σ_v^2 and test $H_0 : \sigma_v^2 = 0$. Notice that we want to estimate and test all of the fixed-effects τ_i while we only need to estimate and test a single random effects parameter.

Basic linear mixed-effects model example

This is not as simple for fixed effects models. We will start with the simplest possible example random effects model: a one-way ANOVA design with a factor at a levels:

$$y_{ij} = \mu + \alpha_i + \varepsilon_{ij}; \quad i = 1, \ldots, a; \quad j = 1, \ldots, n,$$

where the αs and the εs have zero mean and respective variances σ_α^2 and σ_ε^2. These variances are known as the **variance components**. Note that this formulation induces an intraclass correlation between observations at

the same level that is formally expressed as

$$\rho = \frac{\sigma_\alpha^2}{\sigma_\alpha^2 + \sigma_\varepsilon^2}.$$

For simplicity, let there be an equal number of observations n per level. Recall that we can decompose the total variation as SST = SSE + SSA,

$$\sum_{i=1}^{a}\sum_{j=1}^{n}(y_{ij} - \bar{y}_{++})^2 = \sum_{i=1}^{a}\sum_{j=1}^{n}(y_{ij} - \bar{y}_{i+})^2 + \sum_{i=1}^{a}\sum_{j=1}^{n}(\bar{y}_{i+} - \bar{y}_{++})^2.$$

Dividing through by the respective degrees of freedom, we obtain the mean squares, MSE and MSA. Now we find that

$$E(SSE) = a(n-1)\sigma_\varepsilon^2, \qquad E(SSA) = (a-1)(n\sigma_\alpha^2 + \sigma_\varepsilon^2)$$

which use the estimators:

$$\hat{\sigma}_\varepsilon^2 = \frac{SSE}{a(n-1)} = MSE, \qquad \hat{\sigma}_\alpha^2 = \frac{SSA/(a-1) - \hat{\sigma}_\varepsilon^2}{n} = \frac{MSA - MSE}{n}.$$

These estimators are known as the ANOVA estimators, and this method of estimating the variance components can be used for more complicated designs. These estimators have the advantage that they take explicit form (a major advantage in the precomputing days), but they have a number of disadvantages including:

1. the possibility of negative variance estimates
2. not easily adapted to unbalanced data
3. the need for complex algebraic complications in more complex models

One technique for estimating parameters in mixed-effect regression models that alleviates these concerns is the familiar method of maximum likelihood estimation. Of course, this requires that we assume distributions for the random effects and the errors.

More general LMM setup

We will now consider models with normal modeling assumptions:

$$Y = X\beta + Zb + \varepsilon \quad \text{or} \quad Y|b \sim N(X\beta + Zb, \sigma^2 I),$$

where $Y \in \mathbb{R}^n$ is the response vector, $X \in \mathbb{R}^{n\times p}$ is a fixed model matrix, $\beta \in \mathbb{R}^p$ is a coefficient vector and is considered a fixed effect, $Z \in \mathbb{R}^{n\times q}$ is a fixed model matrix for the random effects, $b \in \mathbb{R}^q$ is a vector of random effects, and σ^2 is the variance of the error distribution. If we further assume that the random effects satisfy $b \sim N(0, \sigma^2 D)$, then the unconditional response Y has the distribution

$$Y \sim N(X\beta, \sigma^2(I + ZDZ^T)).$$

We can estimate β using generalized least squares if we know D. However, knowledge of D is usually lacking. Then, letting $V = I + ZDZ^T$, the log likelihood for this model is

$$l(\beta, \sigma, D|y) \propto \frac{\log|\sigma^2 V|}{2} + \frac{(Y - X\beta)^T V^{-1}(Y - X\beta)}{2}.$$

In principle the above can be estimated by maximum likelihood estimation, but it is difficult in practice. One difficulty is that the final unrestricted estimate may estimate some of the variances in D to be negative, and such estimates must be constrained to be zero. First order optimization conditions may therefore not be satisfied.

MLE also suffers bias when estimating random effect parameters. Recall a simple example where $x_1, \ldots, x_n \overset{iid}{\sim} N(\mu, \sigma^2)$, then the MLE of σ^2 is $\hat{\sigma}^2 = n^{-1}\sum_i (x_i - \bar{x})^2$. This estimate is biased, an unbiased estimator is

leading term should be $(n-1)^{-1}$ instead of n^{-1}. Conceptually, these problems are driven by estimation of μ. If μ were known then $n^{-1}\sum_i (x_i - \mu)^2$ is an unbiased estimator for σ^2. Similar problems occur with the estimation of variance components. Given, that the number of factors may not be large, the bias of the MLE for the variance component associated with that factor may be quite large.

Restricted maximum likelihood estimation (REML) estimators are an attempt to get around problems with standard unrestricted maximum likelihood estimation. The idea of REML is to find all linear combinations of the response, denoted k, such that $k^T X = 0$. Form a matrix K with columns k, so that

$$K^T Y \sim N(0, K^T V K).$$

This removes the effect of estimating the mean (fixed effects) from the problem. We can then proceed to maximize the likelihood based on $K^T Y$. Once the random effect parameters have been estimated, it is easy enough to estimate the fixed effect parameters via plug-in. Plug-in gives,

$$Y \approx N(X\beta, \sigma^2 \widehat{V}_{\text{REML}}),$$

and maximum likelihood then gives

$$\hat{\beta}_{\text{REML}} = \left(X^T \widehat{V}_{\text{REML}}^{-1} X\right)^{-1} X^T \widehat{V}_{\text{REML}}^{-1} Y.$$

REML generally produces less biased estimates than unconditional MLE. For balanced data, REML estimates are usually the same as ANOVA estimates.

Example: pulp data set

We illustrate the fitting methods using some data from an experiment which tests how the paper brightness is influenced by the shift operator on duty. See Section 8.1 in Faraway [2016] for more details. We start with a fixed effects one-way ANOVA.

```
library(ggplot2)
library(faraway)
data(pulp)
## note that we are changing the contrasts to sum contrasts
op = options(contrasts = c("contr.sum", "contr.poly"))
## aov is another wrapper for lm; results more appropriate for ANOVA models
lmod = aov(bright ~ operator, pulp)
summary(lmod)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## operator     3   1.34  0.4467   4.204 0.0226 *
## Residuals   16   1.70  0.1062
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coef(lmod)
```

```
## (Intercept)   operator1   operator2   operator3
##       60.40       -0.16       -0.34        0.22
```

We see that the p-value for the operator effect is roughly 0.023. We have $\hat{\sigma}^2 = .106$ and the overall mean is 60.40. For sum contrasts, we have that $\sum_i \alpha_i = 0$, so we can calculate the effect for the fourth operator as 0.16 + 0.34 - 0.22 = 0.28. Turing to the random effects model, we can calculate the variance of the operator effects $\hat{\sigma}_\alpha^2$ using the formula above as

```
## (MSA - MSE) / n
(0.4467 - 0.1062)/5
```

```
## [1] 0.0681
```

We now demonstrate the MLE approach using REML. We will use the `lme4` package to accomplish this task.

```
library(lme4)
```

```
## Loading required package: Matrix
```

```
mmod = lmer(bright ~ 1 + (1|operator), pulp)
summary(mmod)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: bright ~ 1 + (1 | operator)
##    Data: pulp
##
## REML criterion at convergence: 18.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.4666 -0.7595 -0.1244  0.6281  1.6012
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  operator (Intercept) 0.06808  0.2609
##  Residual             0.10625  0.3260
## Number of obs: 20, groups:  operator, 4
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  60.4000     0.1494   404.2
```

In the above, the model fitted has both fixed and random effects components. The fixed effect here is just the intercept represented by the first `1` in the model formula. The random effect is represented by the `(1|operator)` indicating that the data is grouped by the `operator` variable, and the `1` indicates that the random effect is constant within each group.

The default fitting method is the REML procedure mentioned above. We see that this method gives identical estimates to the ANOVA results above. For unbalanced designs, the REML and ANOVA estimators are not necessarily identical. **Analyze this data using unrestricted maximum likelihood estimation and comment on the differences that such analysis produces when compared to the REML and ANOVA estimates.**

Inference for nested models

Using standard likelihood theory, we may derive a test to compare two nested hypotheses, H_0 and H_1, by computing the likelihood ratio test statistic:

$$2\{l(\hat{\beta}_1, \hat{\sigma}_1^2, \hat{D}_1|Y) - l(\hat{\beta}_0, \hat{\sigma}_0^2, \hat{D}_0|Y)\}$$

where $\hat{\beta}_0, \hat{\sigma}_0^2, \hat{D}_0$ are the MLEs of the parameters under the null hypothesis and $\hat{\beta}_1, \hat{\sigma}_1^2, \hat{D}_1$ are the MLEs of the parameters under the alternative hypothesis. The distribution of the above test statistic

$$2\{l(\hat{\beta}_1, \hat{\sigma}_1^2, \hat{D}_1|Y) - l(\hat{\beta}_0, \hat{\sigma}_0^2, \hat{D}_0|Y)\} \approx \chi^2_{\text{df}}$$

where the degrees of freedom is equal to difference in the dimensions of the two parameter spaces (the difference in the number of parameters when both models are identifiable). Note that testing in this manner is approximate and that the parameters under the null hypothesis are on the boundary of the parameter space. This is a real concern when testing the hypothesis about the random effects that take the form $H_0 : \hat{\sigma}^2 = 0$. Furthermore, the χ^2 approximation can be poor even when the conditions are met. The p-values generated

by the LRT for fixed effects are approximate and unfortunately tend to be too small, thereby sometimes overstating the importance of some effects.

Note that we may use bootstrap methods to find more accurate p-values for the LRT. The most popular bootstrap approach is nonparametric in that no distributional assumptions are made. Since we are willing to assume normality for the errors and the random effects in our modeling formulation, we can use a technique called the **parametric bootstrap**. In this technique, we generate data under the null model using the fitted parameter estimates as guesses for the true parameter values. We compute the LRT test statistic for this generated data. We repeat this scheme many times and compare the observed LRT test statistic to the distribution of LRT test statistics obtained from repeatedly generation of samples. We will discuss this approach below. Here is a useful reference on bootstrap procedures.

An alternative approach is to condition on the estimated values of the random effect parameters and then use standard F- or t-tests. This assumes that the covariance of the random part of the model D is equal to its estimated value.

- **Testing random effects**: In most cases testing will be of the form $H_0 : \sigma^2 = 0$. This breaks the assumption that the parameter value is in the interior of its parameter space. The null distribution is unknown in general and precise methods therefore requires numerical methods. Use of the χ^2 distribution with the usual dfs tends to inflate the p-values. Thus you can be confident in significant findings. More time consuming bootstrapping methods are needed for small, not significant p-values. However, careful coding using parallel methods can greatly quicken these computations.
- **Expected mean squares**: These are more powerful than the LRT but they often require tedious algebra that is dependent on the model under study. They also require extensive adjustments for unbalanced designs.
- **Testing fixed effects**: For testing the fixed effects we must use unrestricted maximum likelihood estimation. One can use the usual LRT, but this test may be anti-conservative. The parametric bootstrap is recommended here.

Example: pulp data set (continued)

In this example the only fixed effect is the mean and there is not much interest in testing that. We first fit the null model

```
## null model fit
nullmod = lm(bright ~ 1, pulp)

## alternative model fit
smod = lmer(bright ~ 1 + (1|operator), pulp, REML = FALSE)
```

As there are no random effects in the null model we must use `lm`. For models of the same class, we could use built in software to perform a statistical test. Here we have to do the testing manually

```
as.numeric(2*(logLik(smod) - logLik(nullmod)))
```

```
## [1] 2.568371
```

```
pchisq(2.568371, df = 1, lower = FALSE)
```

```
## [1] 0.1090199
```

The p-value for this test is well above a nominal 5% significance level (assuming that testing is desired at this level). The use of the χ^2 test with its noted shortcomings allows for some doubt in this finding.

We will now try the parametric bootstrap procedure to obtain a more accurate p-value. We need to estimate the probability, given the null hypothesis is true, of observation an LRT of 2.568371 or greater. We have that

$$y \overset{H_0}{\sim} N(\mu, \sigma^2).$$

A simulation approach would generate data under this model, fit the null and alternative models and then compute the LRT. The process would be repeated a large number of times and the proportion of estimated LRTs exceeding the the observed LRT value of 2.568371 would be used to estimate the p-value. In practice we do not know the true values of μ and σ, but we can use plug-in.

Let's try it out. Fix the bootstrap sample size to be $B = 1000$ and then continue with the parametric bootstrap procedure

```
set.seed(13)
B = 1000
lrtstat = numeric(B)
system.time(for(b in 1:B){
  y = unlist(simulate(nullmod))
  bnull = lm(y ~ 1)
  balt = lmer(y ~ 1 + (1|operator), pulp, REML = FALSE)
  lrtstat[b] = as.numeric(2*(logLik(balt) - logLik(bnull)))
})
```

```
##    user  system elapsed
##   5.525   0.040   5.564
```

We may examine the distribution of the bootstrapped LRTs. We first compute the proportion that are close to zero

```
mean(lrtstat < 1e-5)
```

```
## [1] 0.69
```

The LRT clearly does not have a χ^2 distribution. See Section 8.2 in Faraway [2016] for discussion on this. An interesting refinement can be seen **here**. The parametric bootstrap may be the simplest approach, and the method we have used above is transparent and could be computed much more efficiently if needed. With all of this in mind the estimated p-value is

```
## p-value
pval = mean(lrtstat > 2.568371)
pval
```

```
## [1] 0.026
```

```
## simple standard error of the above
sqrt(pval*(1-pval)/B)
```

```
## [1] 0.005032296
```

We can be fairly sure that the estimated p-value is under a 0.05 nominal level. If in doubt, do some more replications to make sure; this only costs computer time (One can perform the above analysis with $B = 1e4$ and make it as fast as possible using parallel programming with either `mclapply` and/or `foreach` and whatever accompanying software packages are needed.) As it happens, this p-value is close to the fixed effects p-value.

In this example, the random and fixed effect tests gave similar outcomes. However, the hypotheses in random and fixed effects are intrinsically different. The conclusions from testing hypotheses involving fixed effects are only relevant to the levels under study. On the other hand, tests of hypotheses involving random effects are relevant of levels of the effect not considered. They are based upon a random generative mechanism that, in this case, are indexed by a single parameter. The conclusions of such test are more meaningful, more general, and should require more care than their fixed effects counterparts.

Predicting random effects

In a model with random effects, the *alpha*s are no longer parameters, but are realizations from a random generative process. Using the standard normal assumption we have that

$$\alpha_i \sim N(0, \sigma_\alpha^2).$$

It does not make much sense to estimate the αs because they are random variables from a common distribution under this setup. So instead we may think of interesting probabilities and expected values corresponding to the random effect distribution. However, $E(\alpha_i) = 0$ for all i which is clearly not very interesting.

If however, one approaches this problem from a Bayesian perspective then we have a prior density for the αs and $E(\alpha_i) = 0$ is just the prior mean. Let f represent a density function, then the posterior density for α is given by

$$f(\alpha_i|Y) \propto f(Y|\alpha_i)f(\alpha_i)$$

we can then find the posterior mean, denoted as $\hat{\alpha}$ as

$$E(\alpha_i|Y) = \int \alpha_i f(\alpha_i|Y) d\alpha_i.$$

For the general case, this works out to be

$$\hat{\alpha} = DZ^T V^{-1}(Y - X\beta).$$

A purely Bayesian approach would specify the parameters of the prior (or specify priors for these) and compute a posterior distribution for α. We can also take an empirical Bayesian point of view and substitute the MLEs into D, V, and β to obtain predicted random effects. These may be computed as

```
ranef(mmod)$operator
```

```
##   (Intercept)
## a  -0.1219403
## b  -0.2591231
## c   0.1676679
## d   0.2133955
```

The predicted random effects are related to the fixed effects. We can show these for all operators

```
(cc = model.tables(lmod))
```

```
## Tables of effects
##
##  operator
## operator
##     a     b     c     d
## -0.16 -0.34  0.22  0.28
```

and then compute the ratio to the random effects as

```
## estimated fixed effects divided by predicted random effects
cc[[1]]$operator / ranef(mmod)$operator
```

```
##   (Intercept)
## a    1.312117
## b    1.312117
## c    1.312117
## d    1.312117
```

We see that the predicted random effects are exactly in proportion to the fixed effects. Typically, the predicted random effects are smaller and could be viewed as a type of **shrinkage estimate**. The 95% confidence intervals for the random effects can be calculated and are displayed below:

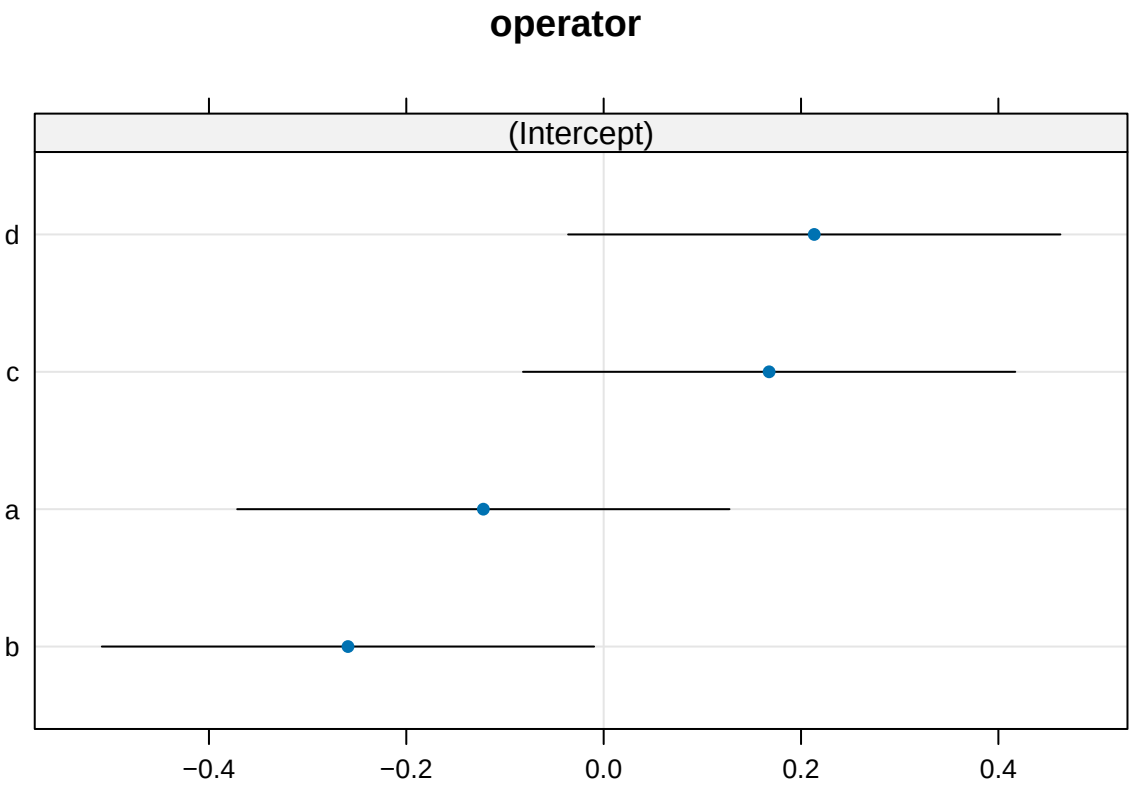
```
library(lattice)
```

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:faraway':
##
##     melanoma
```

```
dotplot(ranef(mmod, condVar=TRUE))
```

```
## $operator
```



Suppose we wish to predict a new value. If the prediction is to be made for a new operator or unknown operator, the best we can do is give $\hat{\mu} = 60.4$. If we know the operator, then we can combine this with our fixed effects to give the best linear unbiased predictors (BLUPs) as follows

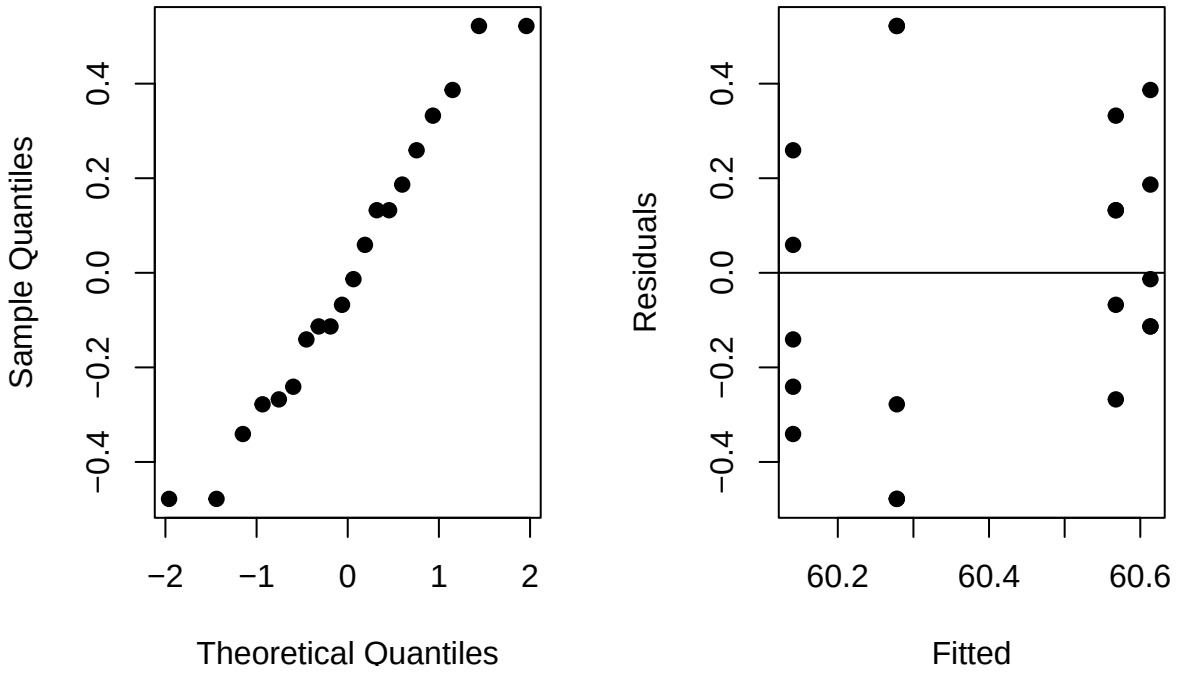
```
fixef(mmod) + ranef(mmod)$operator
```

```
##   (Intercept)
## a    60.27806
## b    60.14088
## c    60.56767
## d    60.61340
```

The implication of this is that we have more than one type of residual depending on what fitted values we use for inference. The default predicted values and residuals are from the outermost level of grouping. Thus, the usual diagnostic plots are still worthwhile.

```
par(mfrow = c(1,2))
qqnorm(resid(mmod), main="", pch = 19)
plot(fitted(mmod),resid(mmod), xlab="Fitted", ylab="Residuals", pch = 19)
```

```
abline(0,0)
```



Returning to the random effect predictions, we present a parametric bootstrap method for computing standard errors of the predicted random effects. As in previous bootstrap, the first step is to simulate from the fitted model. We refit the model with the simulated response and generate a predicted value. But there are two additional sources of variation. We have variation due to the new operator and also due to a new observation from that operator. For this reason, we add normal sample values with standard deviations equal to those estimated earlier. If you really want a confidence interval for the mean prediction, you should not add these extra error terms. We repeat this 1000 times and take the appropriate quantiles to get a 95% interval. We start with the unknown operator case:

```
group.sd = as.data.frame(VarCorr(mmod))$sdcor[1]
resid.sd = as.data.frame(VarCorr(mmod))$sdcor[2]
B = 1000
pv = numeric(B)
system.time(for(i in 1:B){
  y = unlist(simulate(mmod, use.u = TRUE))
  bmod = suppressWarnings(refit(mmod, y))
  pv[i] = predict(bmod, re.form=~0)[1] + rnorm(n=1,sd=group.sd) +  rnorm(n=1,sd=resid.sd)
})
```

```
##    user  system elapsed
##   6.344   0.030   6.374
```

```
quantile(pv, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 59.62780 61.24264
```

Some modification is necessary if we know the operator we are making the prediction interval for. We use the option `use.u=TRUE` in the simulate function indicating that we should simulate new values conditional on the estimated random effects. We need to do this because otherwise we would simulate an entirely new 'a' effect in each replication. Instead, we want to preserve the originally generated 'a' effect.

```
system.time(for(i in 1:B){
 y = unlist(simulate(mmod, use.u=TRUE))
  bmod = suppressWarnings(refit(mmod, y))
  pv[i] = predict(bmod, newdata=data.frame(operator="a")) + rnorm(n=1,sd=resid.sd)
})
```

```
##    user  system elapsed
##   9.136   0.056   9.192
```

```
quantile(pv, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 59.60897 60.97931
```

Example: Soybean analysis

The researchers are interested in determining which genotypes are associated with a different expression of apparent quantum efficiency (AqE) from the RC reference genotype. AqE is a part of the photosynthetic process in plants where improvements in AqE allow for improvements in yield. The researchers have collected data across multiple plots for multiple days over two years. The plot effects are not of interest.

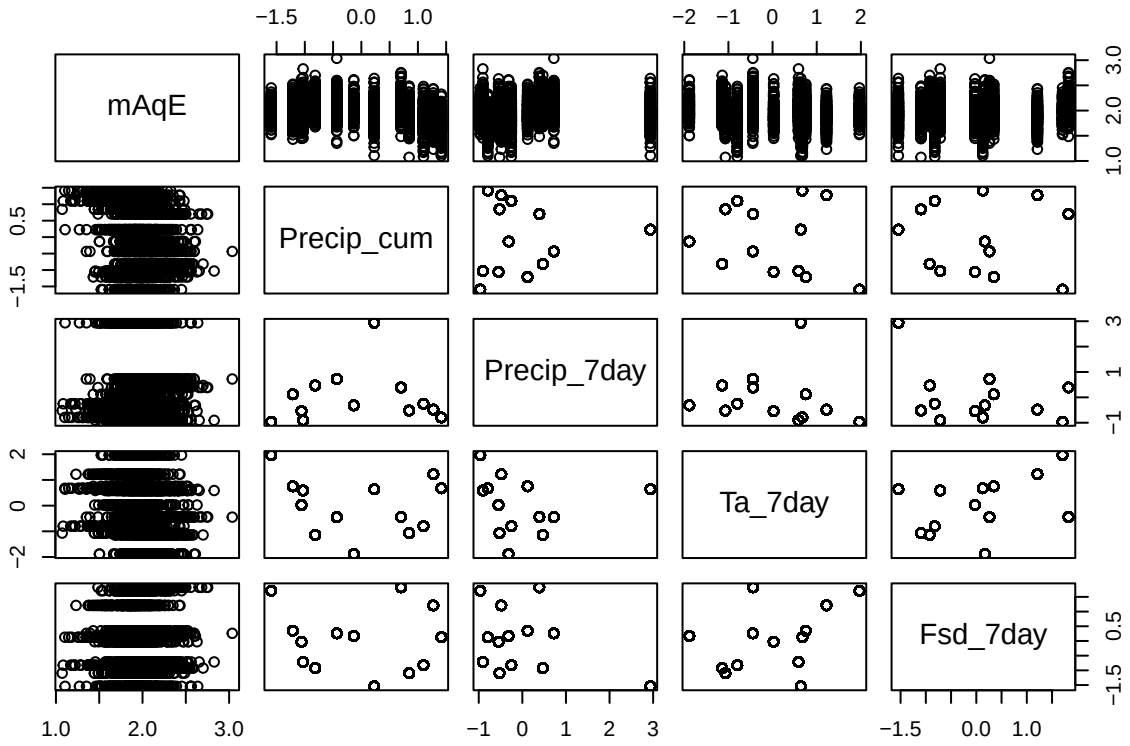
It is known that AqE will have a quadratic effect in date, and that AqE may depend on weather conditions. Some weather variables are also collected (Precip_cum, Precip_7day, Ta_7day, and Fsd_7day).

```
dat = read.csv("soybean.csv")
dat$ID = as.factor(dat$ID)
head(dat)
```

```
##         Date year    ID plot_number plot_number_year     mAqE Precip_cum
## 1 2021-06-24 2021 NAM46           1           1_2021 2.585728 -0.8163990
## 2 2021-06-30 2021 NAM46           1           1_2021 2.453331  0.2272382
## 3 2022-07-12 2022 NAM26           1           1_2022 2.295015 -1.2118618
## 4 2021-07-13 2021 NAM46           1           1_2021 2.011619  0.8453337
## 5 2022-07-19 2022 NAM26           1           1_2022 1.983389 -1.0587556
## 6 2021-07-20 2021 NAM46           1           1_2021 1.781536  1.0987849
##   Precip_7day     Ta_7day    Fsd_7day   Date_num
## 1   0.4695263 -1.14071540 -0.92257332 -1.0278192
## 2   2.9365607  0.63872728 -1.54297943 -0.9956941
## 3   0.1215929  0.75291580  0.34722996  1.0228330
## 4  -0.5227283 -1.06866861 -1.09565527 -0.9260897
## 5  -0.5442057  0.02483327 -0.02693322  1.0603123
## 6  -0.2528293 -0.79961846 -0.82287859 -0.8886104
```

This data consists of multiple measurements on the same day with only a few days of data collection. Here is a depiction of the relationships between the response and the weather variables.

```
pairs(dat[,6:10])
```



We now fit three models in increasing complexity. The first is a standard linear regression model with linear terms for precipitation variables and day, and a quadratic term for days. The next model includes a random effect for plots. The full model includes an additional random effect for disk.

```
m1_mAqE_lm = lm(mAqE ~ ID + Date_num + I(Date_num^2),
  data = dat)

m1_mAqE = lmer(mAqE ~ ID + Date_num + I(Date_num^2) +
  (1|plot_number_year),
  data = dat, REML = FALSE,
  control = lmerControl(optimizer ="Nelder_Mead"))

m1_mAqE_full = lmer(mAqE ~ ID + Date_num + I(Date_num^2) +
  Precip_cum + Precip_7day + Ta_7day + Fsd_7day + (1|plot_number_year),
  data = dat, REML = FALSE,
  control = lmerControl(optimizer ="Nelder_Mead"))

AIC(m1_mAqE_lm)
```

```
## [1] -190.9327
```

```
AIC(m1_mAqE)
```

```
## [1] -239.5221
```

```
AIC(m1_mAqE_full)
```

```
## [1] -415.3697
```

Just as before we can use a parametric bootstrap to compare nested models. In this case we compare the linear regression model to a mixed-effects model containing a random effect for plots. In this example we can see that the parametric bootstrap is fairly time-consuming.

```r
set.seed(13)
B = 1000
lrtstat = numeric(B)

# parametric bootstrap
system.time(for(b in 1:B){
  y = unlist(simulate(m1_mAqE_lm))
  bnull = lm(y ~ ID + Date_num + I(Date_num^2), data = dat)
  balt = lmer(y ~ ID + Date_num + I(Date_num^2) +
    (1|plot_number_year),
    data = dat, REML = FALSE,
    control = lmerControl(optimizer ="Nelder_Mead"))
  lrtstat[b] = as.numeric(2*(logLik(balt) - logLik(bnull)))
})
```

```
##    user  system elapsed
##  56.691   1.584  58.312
```

We examine the distribution of the bootstrapped LRTs. We observe a large mass at zero and only observe a few non-zero bootstrap observations.

```r
mean(lrtstat < 1e-5)
```

```
## [1] 0.9
```

The bootstrapped p-value is 0.

```r
## p-value
pval = mean(lrtstat >
  as.numeric(2*(logLik(m1_mAqE) - logLik(m1_mAqE_lm))))
pval
```

```
## [1] 0
```

```r
## simple standard error of the above
sqrt(pval*(1-pval)/B)
```

```
## [1] 0
```

We now compare the mixed-effects model containing a random effect for plots with a larger mixed-effects model containing random effects for both plots and disk. Again, the parametric bootstrap is fairly time-consuming.

```r
set.seed(13)
B = 1000
lrtstat = numeric(B)

# parametric bootstrap
system.time(for(b in 1:B){
  y = unlist(simulate(m1_mAqE))
  bnull = lmer(y ~ ID + Date_num + I(Date_num^2) +
    (1|plot_number_year),
    data = dat, REML = FALSE,
    control = lmerControl(optimizer ="Nelder_Mead"))
  balt = lmer(y ~ ID + Date_num + I(Date_num^2) +
    Precip_cum + Precip_7day + Ta_7day + Fsd_7day +
      (1|plot_number_year),
    data = dat, REML = FALSE,
    control = lmerControl(optimizer ="Nelder_Mead"))
```

```
  lrtstat[b] = as.numeric(2*(logLik(balt) - logLik(bnull)))
})
```

```
##    user  system elapsed
## 168.911   3.957 172.943
```

We examine the distribution of the bootstrapped LRTs. We observe a large mass at zero.

```
mean(lrtstat < 1e-5)
```

```
## [1] 0
```

The bootstrapped p-value is 0.

```
## p-value
pval = mean(lrtstat > as.numeric(2*(logLik(m1_mAqE_full) - logLik(m1_mAqE))))
pval
```

```
## [1] 0
```

```
## simple standard error of the above
sqrt(pval*(1-pval)/B)
```

```
## [1] 0
```

A quicker decision can be made using model selection criteria. Both AIC and BIC select the largest model with random effects for plots and disk effects.

```
## AIC
c(AIC(m1_mAqE_lm), AIC(m1_mAqE), AIC(m1_mAqE_full))
```

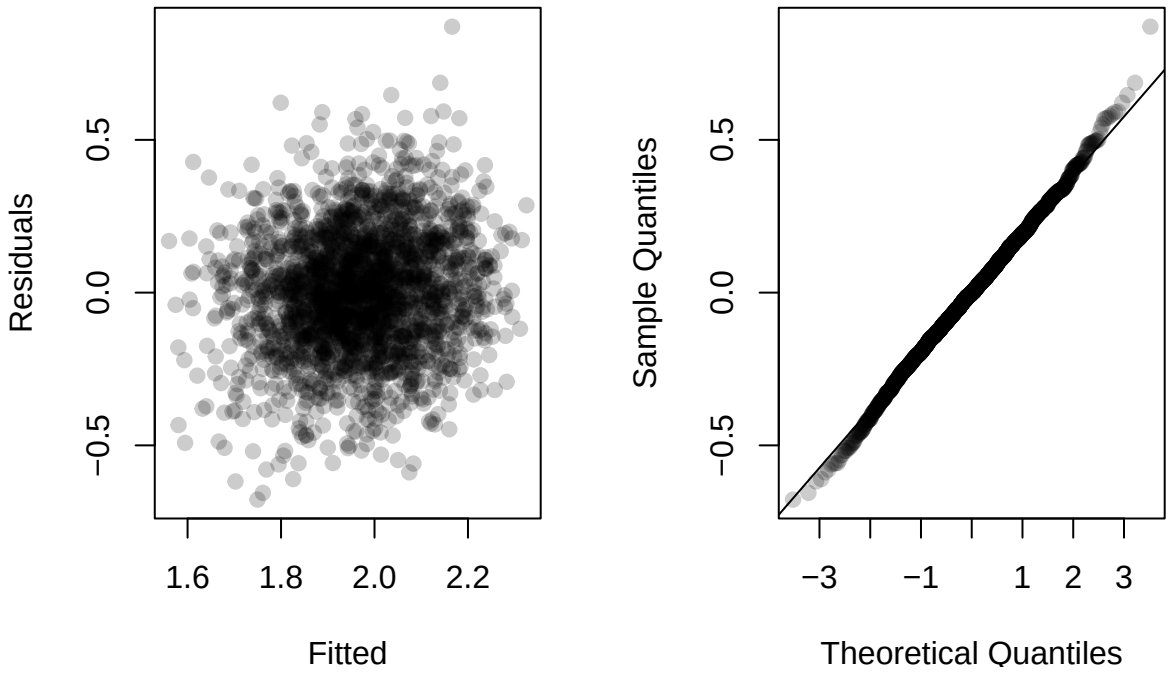
```
## [1] -190.9327 -239.5221 -415.3697
```

```
## BIC
c(BIC(m1_mAqE_lm), BIC(m1_mAqE), BIC(m1_mAqE_full))
```

```
## [1]   62.05648   19.21680 -133.63177
```

We see that the modeling assumptions of constant variance and normality of residuals are satisfied.

```
par(mfrow = c(1,2))
plot(fitted(m1_mAqE_full), residuals(m1_mAqE_full),
         xlab="Fitted", ylab="Residuals", pch = 19,
         col = rgb(0,0,0,alpha=0.2))
a = qqnorm(residuals(m1_mAqE_full), main="", pch = 19,
                        col = rgb(0,0,0,alpha=0.2))
qqline(residuals(m1_mAqE_full))
```



We now investigate which genotypes are associated with AqE values that are different than the RC reference level. To do this we will construct a model with a particular genotype level removed, and then compare the smaller model with the full model selected above. Model comparisons will be made using AIC. We will perform this procedure for each genotype. Results are displayed below, negative values indicate that a genotype is associated with AqE values that are different than the RC reference level.

```
## AIC for each ID variable from full AqE fixed-effects model
M = model.matrix(mAqE ~ ID + Date_num + I(Date_num^2) +
  Precip_cum + Precip_7day + Ta_7day + Fsd_7day,
                  data = dat)

# Note that likelihood ratios are asymptotic, i.e. don't account for
# uncertainty in the estimate of the residual variance
library(parallel)
ncores = detectCores() - 2
system.time({AIC_IDs = matrix(unlist(lapply(
    grep("ID", colnames(M)), function(j){
    M1 = M[, -j]
    foo = lmer(mAqE ~ -1 + M1 + (1|plot_number_year),
                                data = dat, REML = FALSE,
  control = lmerControl(optimizer ="Nelder_Mead"))
    AIC(m1_mAqE_full) - AIC(foo)
})), ncol = 1)})
```

```
##    user  system elapsed
##   3.247   0.110   3.357
```

```
rownames(AIC_IDs) = colnames(M)[grep("ID", colnames(M))]
colnames(AIC_IDs) = c("AqE")
cbind(round(AIC_IDs,2), ifelse(AIC_IDs < 0, 1, 0))
```

```
##         AqE AqE
## ID1   -0.32   1
## ID2   -0.13   1
```

```
## ID3    0.87   0
## ID4    1.17   0
## ID5    1.86   0
## ID6   -0.53   1
## ID7    0.69   0
## ID8   -2.70   1
## ID9   -0.70   1
## ID10   1.66   0
## ID11   1.98   0
## ID12   0.99   0
## ID13  -9.03   1
## ID14   1.73   0
## ID15  -1.44   1
## ID16   1.21   0
## ID17 -12.40   1
## ID18   1.80   0
## ID19  -0.52   1
## ID20   1.90   0
## ID21  -2.86   1
## ID22   1.61   0
## ID23   1.99   0
## ID24   0.86   0
## ID25   1.99   0
## ID26  -4.28   1
## ID27  -3.22   1
## ID28   0.93   0
## ID29  -1.48   1
## ID30  -0.30   1
## ID31   0.43   0
## ID32  -2.84   1
## ID33   1.11   0
## ID34   1.85   0
## ID35   1.90   0
## ID36   1.99   0
## ID37  -1.06   1
## ID38   0.76   0
## ID39   0.67   0
## ID40  -5.50   1
```

–>

–>

Acknowledgments

We borrow materials from Faraway [2016], Pinheiro and Bates [2006], Agresti [2013], the vignette for the `lem4` package, and Adam Rothman's notes on computing that are included along with these notes.

References

Alan Agresti. *Categorical Data Analysis*. John Wiley & Sons, 2013.

Julian J Faraway. *Extending the linear model with R: generalized linear, mixed effects and nonparametric regression models*. CRC press, 2016.

José Pinheiro and Douglas Bates. *Mixed-effects models in S and S-PLUS*. Springer Science & Business Media, 2006.