

The goal today is to get familiar with how to handle random effects in linear models in R. We will do this by analyzing a data set from Eliza. She writes:

This data set is from a greenhouse experiment investigating the effects of water treatments on seedling growth. High water treatment plants were watered every other day; low water treatment plants were watered every 5 days. The experiment followed a randomized complete blocks design with 8 blocks and 4 replicates of each treatment in a block. The experiment lasted 10 weeks before plants were dried and weighed to get root/shoot ratios.

Lets start by getting the data into R and taking a look at it:

```
> water=read.table("water.txt", header=1)
> summary(water)
  treatment      block      Root.Shoot
high:32  Min.    :1.000  Min.    :0.1038
low :30  1st Qu.:2.250  1st Qu.:0.2007
        Median :4.000  Median :0.2367
        Mean   :4.435  Mean   :0.2374
        3rd Qu.:6.000  3rd Qu.:0.2745
        Max.   :8.000  Max.   :0.3671
> water$block=factor(water$block)
> summary(water)
  treatment      block      Root.Shoot
high:32  1      : 8  Min.    :0.1038
low :30  2      : 8  1st Qu.:0.2007
        3      : 8  Median :0.2367
        4      : 8  Mean   :0.2374
        5      : 8  3rd Qu.:0.2745
        8      : 8  Max.   :0.3671
        (Other):14
```

(Note that R has automatically converted the column name “Root/Shoot” to “Root.Shoot”, because “/” is not allowed for the name of an object.)

There are a number of graphical summaries we might like to do:

```
> boxplot(Root.Shoot~treatment, data=water)
```

This shows that there are differences between the treatments, but also quite a bit of scatter. We can also look at the blocking factor:

```
> boxplot(Root.Shoot~block, data=water)
```

Of course, we are also going to want to pay attention to the possibilities of any interactions. We can do this with an interaction plot:

```
> interaction.plot(treatment, block, Root.Shoot)
```

Another way to draw the plot:

```
> interaction.plot(treatment, block, Root.Shoot, col=1:8, lty=1)
```

(In the above command `1:8` in the `col` is interpreted as a vector of the integers from one to eight. Read the help file for mor information on these arguments)

This plot suggests that there is some interaction (since the lines cross), but of course, we don't have any sense of the scatter on these points, so we had better go for the formal analysis.

You have a couple options for dealing with random effects:

1) Using `lm`:

When you use `lm` to do you anova modeling, you can call the `anova` function on the fit and get the mean squares, F values and p values. The problem is that `anova` calculates F statistics as though everything is a fixed factor. Thus, if you have a random factor, the statistical tests will be wrong, but you can use the provided mean squares to construct your own test.

```
> water.lm=lm(Root.Shoot~treatment*block, data=water)
```

```
> anova(water.lm)
```

Analysis of Variance Table

Response: Root.Shoot

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
treatment	1	0.029605	0.029605	9.0029	0.004343 **
block	7	0.007727	0.001104	0.3357	0.933454
treatment:block	7	0.009072	0.001296	0.3941	0.900989
Residuals	46	0.151266	0.003288		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

If these were all both fixed factors, then we could say that there was a significant treatment effect, and no block or block x treatment effect. But, block is a random factor, which means that the F ratios are wrong. The treatment x block MS is properly tested over the residual MS, so we are safe there. The block effect is tested over the residual, so we are safe in saying that there is no block effect. But the problem is that the treatment MS is tested over the residual MS, when it should be tested over the interaction. (Check Sokal and Rohlf, p. 334) But we can do the test by hand:

The F ratio is:

```
> treat.F = .029605/.001296
> treat.F
[1] 22.84336
```

So now we need to know just how extreme this value is. R can be used as a large statistical table. *An Introduction to R* (a free online document with a link on the course website) lists many of the available distributions. For each distribution, we can use the cumulative density function (i.e. the probability of a value this extreme or less) for statistical tests. (We can also draw from the density function, quantiles, or get random deviates. Look at the manual for more info.)

In this case, we want to see the probability of getting a F value this large or larger. We can get this with:

```
> pf(treat.F, 1, 7)
[1] 0.9979865
```

To get the p-value as typically expressed:

```
> 1-pf(treat.F, 1, 7)
[1] 0.002013506
```

So we've had to do a little work to get the right tests for the random effects, but we got there. As long as you know the right F-ratios for a given model, this way of doing things works, and will give you kinds of results we are used to seeing.

2) Doing the mixed model directly:

Mixed models are handled directly with the function `lme`. See Pinherio and Bates (2000) for lots more information.

The package `nlme` is used for mixed-effects models in R. `nlme` is a user contributed package, which means that you need to load the package before you can use any of the functions:

```
> library(nlme)
```

To fit a mixed effects model, you use the function `lme`:

```
> water.lme=lme(Root.Shoot~treatment, data=water, random=~1|block)
> summary(water.lme)
```

Linear mixed-effects model fit by REML

```
Data: water
      AIC      BIC   logLik
-167.5239 -159.1466 87.76197
```

Random effects:

```
Formula: ~1 | block
      (Intercept)  Residual
StdDev: 0.0003681551 0.05292475
```

Fixed effects: Root.Shoot ~ treatment

	Value	Std.Error	DF	t-value	p-value
(Intercept)	0.25852322	0.009356767	53	27.629545	0.000
treatmentlow	-0.04372653	0.013449907	53	-3.251065	0.002

Correlation:

	(Intr)
treatmentlow	-0.696

Standardized Within-Group Residuals:

Min	Q1	Med	Q3	Max
-2.09723524	-0.74071778	-0.08777217	0.75510641	2.72068065

Number of Observations: 62

Number of Groups: 8

So we have a bunch of output here which is pretty new. `lme` fits the model using maximum likelihood or restricted maximum likelihood (REML). We can assess the significance of the fixed effects (i.e., the treatment) in the output labeled fixed effects. We see that the intercept of linear model is significantly different from zero (not that we really care), and we also see that treatment effect is significant. If you want to see this in terms of F-ratios, use `anova`:

```
> anova(water.lme)
```

	numDF	denDF	F-value	p-value
(Intercept)	1	53	1246.6515	<.0001
treatment	1	53	10.5694	0.002

But what about the block effect? For reasons that I don't quite understand, `lme` doesn't give you the block effect directly. To test for the block effect, we will fit another model without the block effect, and then compare the two models.

```
> water2.lm=lm(Root.Shoot~treatment, data=water)
> anova(water.lme, water2.lm)
      Model df      AIC      BIC  logLik  Test  L.Ratio p-value
water.lme   1  4 -167.5239 -159.1465  87.76197
water2.lm   2  3 -169.5255 -163.2425  87.76277 1 vs 2 0.001608493 0.968
```

We have used the `anova` function to compare the two models. When you compute an anova table in a traditional analysis, the F-ratio compares two models which differ in a single term. A significant F-ratio indicates that that term is significantly greater than zero. In this case, we compare two models by the likelihood ratio test. When one model is a subset of the other model, then the difference in their likelihoods is distributed as χ^2 with $k_2 - k_1$ degrees of freedom, where k_2 and k_1 are the degrees of freedom of the two models. So in this case, we can see that the p-value associated with the likelihood ratio test is very large, indicating that there was no block effect.

In the above analysis, we did not include a block x treatment interaction, but we can certainly do so. We can do this one of two ways. We can either fit an entirely new model:

```
water2.lme=lme(Root.Shoot~treatment, data=water,
random=~1|block/treatment)
```

or we can use the update function

```
> water2.lme=update(water.lme, random=~1|block/treatment)
```

The update function takes a model which has already been fit, and changes those arguments which are specified. In this case, we only change the random effects, so we do not need to specify the fixed effects or the data frame. This method should give the same results as fitting an entirely new model.

So now we have three separate models: `water2.lme`, which has both a random block effect and a random interaction effect, `water.lme`, which has a random block effect (but no interaction), and `water2.lm` which has only the treatment effect. Just as we would compare these models with F-ratios in a traditional anova, we can compare them with likelihood ratio test here:

```
> anova(water2.lme, water.lme, water2.lm)
      Model df      AIC      BIC  logLik  Test  L.Ratio  p-value
water2.lme   1  5 -165.5243 -155.0526  87.7621
water.lme    2  4 -167.5239 -159.1465  87.7619 1 vs 2 0.0003819 0.9844
water2.lm    3  3 -169.5255 -163.2425  87.7627 2 vs 3 0.0016084 0.9680
```

Notice that these p-values are slightly different from those calculated by the traditional F-ratio method, but in this case, it hardly matters- both random effects are highly insignificant. Before Eliza breathes a complete sigh of relief, we should probably like to examine some of our assumptions using plots. Unlike `lm` fits, calling `plot` on a `lme` object only produces one plot (try it to see). If we want to see the q-q plot, we have to do a little work. The `resid` function returns the residuals of a fit model, and we can produce the q-q plot with `qqnorm`, which produces a q-q plot, assuming that the residuals were sampled from a normal distribution. (See `qqplot` for a more general function which could use other sampling distributions.) We can get our plot with

```
> qqnorm(resid(water.lme))
```

and add a line of slope 1 with

```
> qqline(resid(water.lme))
```

We could also get this plot with

```
> qqnorm(water.lme, abline=c(0,1))
```

Look at the help for `qqnorm.gls` for more info on this latter way of doing things.

These data don't look wickedly non-normal, which should make us all happy. To my knowledge, there is no function to calculate Cook's distance or other influence measures for `lme` models.

A data set to play with:

The file `lacY.txt` is a data set of mine, which you might want to try analyzing. It is a randomized complete block design, where I measured the activity of an enzyme (lactose permease) and the concentration of total protein in cultures five species of bacteria. I measured these two things on three separate days, and you can try analyzing these data to see if enzyme activity is influenced by strain (a fixed effect) and day (a random effect). There is no replication within days. Note that because there is no replication within days, you can't test for an interaction. (Try fitting a model with an interaction and see what R does.)