Nesting interaction terms

We're all familiar with how to make regression model formulae in R. We all know that if we want to test the interaction of Var1 and Var2, we use the formula `Var1 * Var2`, which is a shorthand for `Var1 + Var2 + Var1:Var2`. In other words, it's the effect of Variable 1, the effect of Variable 2, and the Variable1-by-Variable2 interaction. This is called *factor crossing*. But did you know there's another way you can put in these interactions, and it's super useful?

Instead of using `*` to *cross* the factors, you can use `/` to *nest* them. This trick is little-known (it's not even documented in the R help page for `?formula`!). But it is very useful in some contexts, which we'll demonstrate below. It has two main consequences:

- First of all, whereas `*` gives you both main effects and the interaction, `/` suppresses one of the main effects. So `Var1 / Var2` is equivalent to `Var1 + Var1:Var2`. Notice that there's no main effect of Var2 anymore.
- Secondly, when used with a regression function like `lm()` or `lmer()`, `/` changes the way interaction coefficients are shown. Imagine that you have four conditions: Aa, Ab, Ba, and Bb; and imagine that your factors are all dummy-coded. When you use `*`, you'll get one coefficient showing how much Aa-Ab differs from zero, and another coefficient showing how much Ba-Bb differs from Aa-Ab. On the other hand,

when you use `/`, you still get that coefficient showing how much Aa-Ab differs from zero, and another coefficient showing how much Ba-Bb differs from *zero*. In other words, factor crossing (`*`) will give you one simple-effect term and one interaction term, whereas factor nesting (`/`) will give you both simple-effect terms.

Why might this be useful, and why should you be careful about it? Let's go through an example.[1]

```
rm(list=ls())

# Load the libraries we will need
library(lme4)
```

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

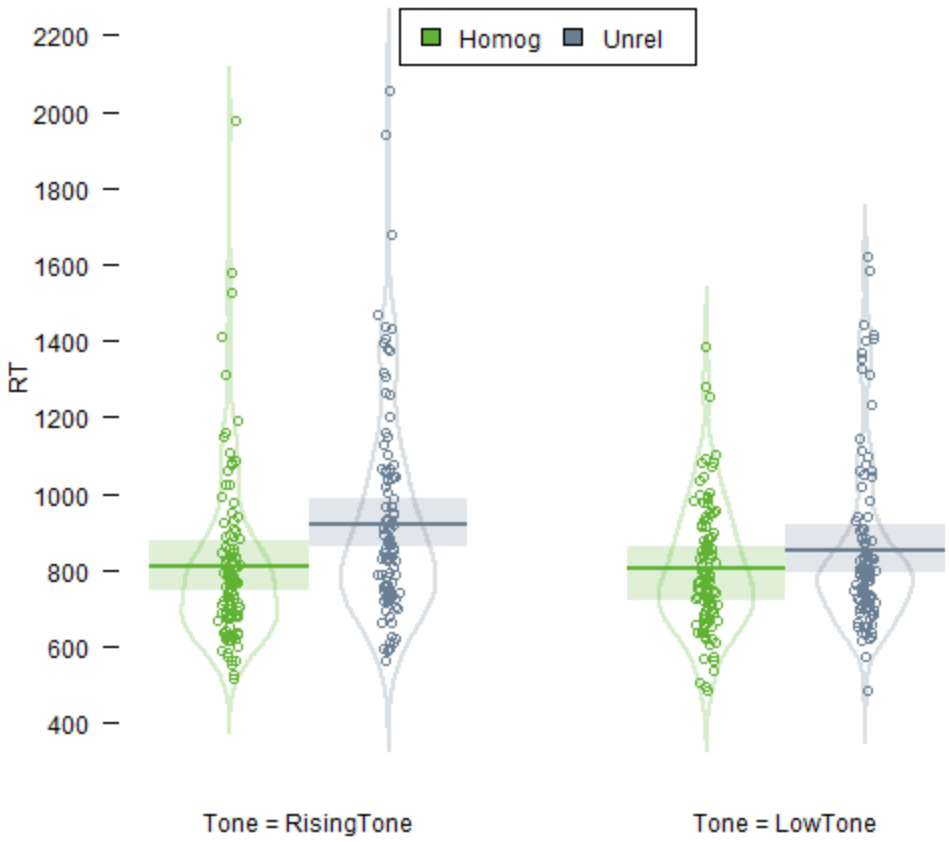
```
# Read the data
load( url("http://users.ox.ac.uk/~cpgl0080/data.RData") )

# Load my custom function for pirate plot with confidence int
source( url("http://users.ox.ac.uk/~cpgl0080/Steve_functions.

# Plot the data
CIpirateplot( RT ~ Condition+Tone, data, grouping.var=c("Subj
```

```
## Warning: closing unused connection 6 (http://users.ox.ac.u
## Steve_functions.txt)
```

```
## Loading required package: boot
```



We see that there's a big simple effect of Condition (i.e., a big Unrelated-Homogeneous difference) for Rising Tone, and a small simple effect of condition for Low Tone. How does this translate into model coefficients? First let's try the usual model, with dummy-coding and a crossed interaction term. (Notice that, unlike what I've shown here, in real applications you should always be using random slopes. Here I just stick with random intercepts for expository purposes, because I want the code to run fast and not have to fight with convergence errors.)

```
summary( crossed_model <- lmer( RT ~ Tone*Condition + (1|Subj
```

```
##                                Estimate Std. Error   t val
## (Intercept)                   859.49323   33.14934 25.9279
## ToneRisingTone                 68.07547   35.01643  1.9441
## ConditionHomog                -61.12232   26.39940 -2.3152
## ToneRisingTone:ConditionHomog -50.30979   36.79587 -1.3672
```

What does this mean? Well, as we went over here, you're not seeing main effects (because the factors are dummy coded). Rather, the coefficient for `ConditionHomog` is telling you that, *for Low Tone* (the baseline level of Tone), Homogeneous trials were responded to 61 ms faster than Unrelated trials, and the *t* statistic for this difference is -2.31. The interaction coefficient (the next row down) is telling you that the Homogeneous-Unrelated difference is even 50 ms faster for Rising Tone than it was for Low Tone, although the *t* statistic for this difference of differences is only -1.37. All in all, these are entirely reasonable: they nicely reflect what we could see in that plot. But what if you care about both simple effects? What if you want to know whether the Homogeneous-Unrelated difference is significant for Low Tones (which we see in this model summary already), *and* if it's significant for Rising Tones (which we do not directly see)? This is where a nested interaction comes in handy.

The benefit of nested interactions: Seeing simple effects

How's about we re-run that model using a nested interaction rather than a crossed interaction, and see what happens:

```
summary( nested_model <- lmer( RT ~ Tone/Condition + (1|Subje
```

```
##                                 Estimate Std. Error   t va
## (Intercept)                    859.49323   33.14934 25.927
## ToneRisingTone                  68.07547   35.01643  1.944
## ToneLowTone:ConditionHomog     -61.12232   26.39940 -2.315
## ToneRisingTone:ConditionHomog -111.43211   25.65605 -4.343
```

Well well well, what have we here! The coefficient for `ToneLowTone:ConditionHomog` is, once again, showing us the simple effect of Condition (Homogeneous minus Unrelated) for Low Tone: notice that it's the exact same value as what we saw in the previous model, which also showed this simple effect. But now in the

row below that, instead of seeing an interaction term, we see the term `ToneRisingTone:ConditionHomog`, which is that coveted simple effect of Condition for *Rising Tone*.

Is a factor nesting valid?

How can we be sure that simple effect we estimated with a nested interaction model is correct? Well, for starters, we can use the t-test formula and directly compare this simple effect to the other coefficient, i.e., test the difference of differences, which is the same as testing the interaction; if the coefficient from this nested model is trustworthy, then comparing these coefficients should give us the exact same *t* statistic as what we got for the interaction coefficient in the crossed model. And you're darn tootin' it does:

```
# A cool function that compares two coefficients. You just pu
#       the coefficients to compare
comparecoef <- function(model, refcoef, comparecoef ){
        varcov <- vcov(model)
        pooled <- sqrt( varcov[comparecoef,comparecoef] + var
        t <- as.numeric( (fixef(model)[comparecoef] - fixef(m
        return( t )
}

# Compare the coefficients
comparecoef( nested_model, 3, 4 )
```

```
## [1] -1.367267
```

Another thing you can try is redo the crossed model with Rising Tone, rather than Low Tone, as the reference level. Then it will show us the simple effect of Condition for Rising Tone (remember that our other crossed model did indeed show the simple effect of Condition for Low Tone). Again, we see that it's the same as the simple effect we saw in the fixed model:

```
# Make a copy of the data and set RisingTone as the reference
data2 <- data
data2$Tone <- relevel( data2$Tone, ref="RisingTone" )

# Make a crossed lmer model again and check out the coefficie
```

```
summary( crossed_model2 <- lmer( RT ~ Tone*Condition + (1|Sub
```

```
##                               Estimate Std. Error   t value
## (Intercept)                  927.56870   33.24076 27.904554
## ToneLowTone                  -68.07547   35.01643 -1.944101
## ConditionHomog              -111.43211   25.65605 -4.343307
## ToneLowTone:ConditionHomog   50.30979   36.79587  1.367267
```

Finally, we can do model comparison and see that the nested and crossed models have the same fit (in terms of log-likelihood, deviance, whatever), indicating that they really are the same model, just showing different comparisons within their coefficients:

```
anova( nested_model, crossed_model )
```

```
## refitting model(s) with ML (instead of REML)
```

```
## Data: data
## Models:
## nested_model: RT ~ Tone/Condition + (1 | Subject) + (1 | I
## crossed_model: RT ~ Tone * Condition + (1 | Subject) + (1
##               Df    AIC    BIC  logLik deviance Chisq Chi
## nested_model   8 5951.6 5984.3 -2967.8   5935.6
## crossed_model  8 5951.6 5984.3 -2967.8   5935.6     0
```

So all that is to say, I think using a nested model in this way is kosher. But why's it useful?

Is factor nesting useful?

In this particular example, looking at the simple effect of Rising Tone doesn't tell us much that we didn't already know from looking at the interaction term. But these things can get much more complicated when a model is more complex. For example, in the very experiment these data came from (Politzer-Ahles & Zhang, in press), we were actually testing a $4 \times 2 \times 2$ design and had specific predictions about certain simple effects, so we used nested interactions (you can see the model coefficients in the tables within that paper). Another example will show how helpful this can be.

Let's look at the self-paced reading data from

Politzer-Ahles & Fiorentino (2013). These data follow a 10×2×2 design. The critical sentences were 10 regions long, and we manipulated the Quantifier appearing at a certain part of the sentence (it could be either *only some* or *some*) and the Boundedness of the sentence context (I won't go into detail, but essentially, the sentence could either be responding to a question about whether "all" of something was there, or whether "any" of something was there). This sounds like a lot, but we had a pretty specific and limited prediction: we expected that for *some* sentences, there would be a Boundedness effect (*any*>*all*) at Region 8, whereas for *only some* sentences there would not be this Boundedness effect at Region 8; likewise, we expected there not to be Boundedness effects at most other regions. So essentially, while this is a kind of three-way interaction, we are most interested in two particular simple effects: the simple effect of Boundedness at Region 8 for *some* sentences, and simple effect of Boundedness at Region 8 for *only some* sentences. Here's what the model looks like with a nasty crossed interaction:

```
# Clear the workspace
rm(list=ls())

# Load self-paced reading data
load( url("http://users.ox.ac.uk/~cpgl0080/UCL_Rworkshop/File

# Pull out just the critical (non-filler) conditions from the
crit <- RTdata[ RTdata$RegionNum>2 & RTdata$Quantifier!="fill

# Clean up factors (remove empty levels, shorten level names)
crit$Quant <- factor( crit$Quantifier )
levels(crit$Quant) <- c("onlysome", "some")
crit$Bound <- factor( crit$Boundedness )
levels(crit$Bound) <- c("all", "any")
crit$Item <- factor( crit$Item )
crit$Region <- factor( crit$RegionNum )

# Do the crossed model
summary( lmer( log(RT) ~ Region*Quant*Bound + (1|Subject) + (
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(RT) ~ Region * Quant * Bound + (1 | Subject)
##    Data: crit
##
```

```
## REML criterion at convergence: 7612.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.6066 -0.6181 -0.1024  0.4722 13.3972
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  Item     (Intercept) 0.001829 0.04276
##  Subject  (Intercept) 0.029264 0.17107
##  Residual             0.105039 0.32410
## Number of obs: 12320, groups:  Item, 48; Subject, 28
##
## Fixed effects:
##                               Estimate Std. Error t value
## (Intercept)                  6.2937974  0.0373612  168.46
## Region4                     -0.3309111  0.0250047  -13.23
## Region5                     -0.4089905  0.0250047  -16.36
## Region6                     -0.4992256  0.0250047  -19.97
## Region7                     -0.5652513  0.0250047  -22.61
## Region8                     -0.5822271  0.0250047  -23.28
## Region9                     -0.6010977  0.0250047  -24.04
## Region10                    -0.4764281  0.0250047  -19.05
## Region11                    -0.2867366  0.0250047  -11.47
## Region12                     0.0121693  0.0471896    0.26
## Quantsome                   -0.0185443  0.0250047   -0.74
## Boundany                    -0.0070758  0.0250047   -0.28
## Region4:Quantsome           -0.1089560  0.0353619   -3.08
## Region5:Quantsome           -0.0941612  0.0353619   -2.66
## Region6:Quantsome            0.0061239  0.0353619    0.17
## Region7:Quantsome           -0.0236284  0.0353619   -0.67
## Region8:Quantsome           -0.0391738  0.0353619   -1.11
## Region9:Quantsome            0.0007828  0.0353619    0.02
## Region10:Quantsome          -0.0002883  0.0353619   -0.01
## Region11:Quantsome           0.0534974  0.0353619    1.51
## Region12:Quantsome          -0.1839578  0.0662362   -2.78
## Region4:Boundany            -0.0046397  0.0353619   -0.13
## Region5:Boundany            -0.0198805  0.0353619   -0.56
## Region6:Boundany             0.0344590  0.0353619    0.97
## Region7:Boundany            -0.0233765  0.0353619   -0.66
## Region8:Boundany            -0.0312178  0.0353619   -0.88
## Region9:Boundany             0.0122040  0.0353619    0.35
## Region10:Boundany            0.0016170  0.0353619    0.05
## Region11:Boundany            0.0438373  0.0353619    1.24
## Region12:Boundany           -0.1599300  0.0662121   -2.42
## Quantsome:Boundany           0.0431226  0.0353619    1.22
## Region4:Quantsome:Boundany  -0.0076233  0.0500093   -0.15
## Region5:Quantsome:Boundany   0.0138621  0.0500093    0.28
## Region6:Quantsome:Boundany  -0.0581377  0.0500093   -1.16
## Region7:Quantsome:Boundany  -0.0014766  0.0500093   -0.03
## Region8:Quantsome:Boundany   0.0562667  0.0500093    1.13
## Region9:Quantsome:Boundany  -0.0162222  0.0500093   -0.32
## Region10:Quantsome:Boundany -0.0254729  0.0500093   -0.51
## Region11:Quantsome:Boundany -0.0999498  0.0500093   -2.00
## Region12:Quantsome:Boundany  0.2243597  0.0936721    2.40
```

```
##
## Correlation matrix not shown by default, as p = 40 > 12.
## Use print(x, correlation=TRUE)  or
##       vcov(x)         if you need it
```

Now where is the effect that I care about in all this? Well, the baseline region is Region 3 (the first "critical" region in the sentence), so we have to keep in mind that every coefficient we see for Region 8 is relative to the effects at Region 3; that's dummy coding for ya. And the baseline Quantifier is *only some*, so the coefficient for *some* is relative to that. Now, there is a coefficient `Region8:Quantsome:Boundany`, and its interpretation is... oh I dunno, I don't want to do all that math, just shoot me now. Let's look at it with the nested coefficients instead:

```
summary( lmer( log(RT) ~ Region/(Quant/Bound) + (1|Subject) +
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(RT) ~ Region/(Quant/Bound) + (1 | Subject) +
##    Data: crit
##
## REML criterion at convergence: 7612.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.6066 -0.6181 -0.1024  0.4722 13.3972
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  Item     (Intercept) 0.001829 0.04276
##  Subject  (Intercept) 0.029264 0.17107
##  Residual             0.105039 0.32410
## Number of obs: 12320, groups:  Item, 48; Subject, 28
##
## Fixed effects:
##                                Estimate Std. Error t val
## (Intercept)                    6.293797   0.037361  168.
## Region4                       -0.330911   0.025005  -13.
## Region5                       -0.408991   0.025005  -16.
## Region6                       -0.499226   0.025005  -19.
## Region7                       -0.565251   0.025005  -22.
## Region8                       -0.582227   0.025005  -23.
## Region9                       -0.601098   0.025005  -24.
## Region10                      -0.476428   0.025005  -19.
## Region11                      -0.286737   0.025005  -11.
## Region12                       0.012169   0.047190    0.
## Region3:Quantsome             -0.018544   0.025005   -0.
## Region4:Quantsome             -0.127500   0.025005   -5.
## Region5:Quantsome             -0.112706   0.025005   -4.
## Region6:Quantsome             -0.012420   0.025005   -0.
## Region7:Quantsome             -0.042173   0.025005   -1.
## Region8:Quantsome             -0.057718   0.025005   -2.
## Region9:Quantsome             -0.017761   0.025005   -0.
## Region10:Quantsome            -0.018833   0.025005   -0.
## Region11:Quantsome             0.034953   0.025005    1.
## Region12:Quantsome            -0.202502   0.061335   -3.
## Region3:Quantonlysome:Boundany -0.007076   0.025005   -0.
```

```
## Region4:Quantonlysome:Boundany  -0.011715   0.025005   -0.
## Region5:Quantonlysome:Boundany  -0.026956   0.025005   -1.
## Region6:Quantonlysome:Boundany   0.027383   0.025005    1.
## Region7:Quantonlysome:Boundany  -0.030452   0.025005   -1.
## Region8:Quantonlysome:Boundany  -0.038294   0.025005   -1.
## Region9:Quantonlysome:Boundany   0.005128   0.025005    0.
## Region10:Quantonlysome:Boundany -0.005459   0.025005   -0.
## Region11:Quantonlysome:Boundany  0.036761   0.025005    1.
## Region12:Quantonlysome:Boundany -0.167006   0.061309   -2.
## Region3:Quantsome:Boundany       0.036047   0.025005    1.
## Region4:Quantsome:Boundany       0.023784   0.025005    0.
## Region5:Quantsome:Boundany       0.030028   0.025005    1.
## Region6:Quantsome:Boundany       0.012368   0.025005    0.
## Region7:Quantsome:Boundany       0.011194   0.025005    0.
## Region8:Quantsome:Boundany       0.061096   0.025005    2.
## Region9:Quantsome:Boundany       0.032029   0.025005    1.
## Region10:Quantsome:Boundany      0.012191   0.025005    0.
## Region11:Quantsome:Boundany     -0.020066   0.025005   -0.
## Region12:Quantsome:Boundany      0.100477   0.061309    1.
```

```
##
## Correlation matrix not shown by default, as p = 40 > 12.
## Use print(x, correlation=TRUE)  or
##       vcov(x)         if you need it
```

Now we have very clearly interpretable coefficients. `Region8:Quantsome:Boundany` is the simple effect of Boundedness for *some* at Region 8, and as expected, it is significant and positive. Now we also have `Region8:Quantonlysome:Boundany`, the simple effect of Boundedness for *only some* at Region 8, which is non-significant, as expected. Not only are these coefficients in line with our predictions (that alone would be a pretty underhanded way of judging which coefficients to like more!), but they also accurately reflect the patterns we see in a plot of the data, which you can find at the paper, or can make yourself as a super fun exercise. The reason these coefficients are much more interpretable is because they're showing things within Region 8 itself, rather than showing them compared to Region 3, which is arbitrary and meaningless.

You can even mix factor nesting and factor crossing. Say I really like this whole nesting thing to look just at Region 8, but I also want to test the difference-of-differences between the *only some* Boundedness effect and the *some* Boundedness effect, rather than testing

each simple effect. That can be done, no sweat:

```
summary( lmer( log(RT) ~ Region/(Quant*Bound) + (1|Subject) +
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(RT) ~ Region/(Quant * Bound) + (1 | Subject)
##    Data: crit
##
## REML criterion at convergence: 7612.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.6066 -0.6181 -0.1024  0.4722 13.3972
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  Item     (Intercept) 0.001829 0.04276
##  Subject  (Intercept) 0.029264 0.17107
##  Residual             0.105039 0.32410
## Number of obs: 12320, groups:  Item, 48; Subject, 28
##
## Fixed effects:
##                              Estimate Std. Error t value
## (Intercept)                  6.293797   0.037361  168.46
## Region4                     -0.330911   0.025005  -13.23
## Region5                     -0.408991   0.025005  -16.36
## Region6                     -0.499226   0.025005  -19.97
## Region7                     -0.565251   0.025005  -22.61
## Region8                     -0.582227   0.025005  -23.28
## Region9                     -0.601098   0.025005  -24.04
## Region10                    -0.476428   0.025005  -19.05
## Region11                    -0.286737   0.025005  -11.47
## Region12                     0.012169   0.047190    0.26
## Region3:Quantsome           -0.018544   0.025005   -0.74
## Region4:Quantsome           -0.127500   0.025005   -5.10
## Region5:Quantsome           -0.112706   0.025005   -4.51
## Region6:Quantsome           -0.012420   0.025005   -0.50
## Region7:Quantsome           -0.042173   0.025005   -1.69
## Region8:Quantsome           -0.057718   0.025005   -2.31
## Region9:Quantsome           -0.017761   0.025005   -0.71
## Region10:Quantsome          -0.018833   0.025005   -0.75
## Region11:Quantsome           0.034953   0.025005    1.40
## Region12:Quantsome          -0.202502   0.061335   -3.30
## Region3:Boundany            -0.007076   0.025005   -0.28
## Region4:Boundany            -0.011715   0.025005   -0.47
## Region5:Boundany            -0.026956   0.025005   -1.08
## Region6:Boundany             0.027383   0.025005    1.10
## Region7:Boundany            -0.030452   0.025005   -1.22
## Region8:Boundany            -0.038294   0.025005   -1.53
## Region9:Boundany             0.005128   0.025005    0.21
## Region10:Boundany           -0.005459   0.025005   -0.22
## Region11:Boundany            0.036761   0.025005    1.47
## Region12:Boundany           -0.167006   0.061309   -2.72
## Region3:Quantsome:Boundany   0.043123   0.035362    1.22
## Region4:Quantsome:Boundany   0.035499   0.035362    1.00
## Region5:Quantsome:Boundany   0.056985   0.035362    1.61
## Region6:Quantsome:Boundany  -0.015015   0.035362   -0.42
## Region7:Quantsome:Boundany   0.041646   0.035362    1.18
## Region8:Quantsome:Boundany   0.099389   0.035362    2.81
```

```
## Region9:Quantsome:Boundany   0.026900   0.035362    0.76
## Region10:Quantsome:Boundany  0.017650   0.035362    0.50
## Region11:Quantsome:Boundany -0.056827   0.035362   -1.61
## Region12:Quantsome:Boundany  0.267482   0.086741    3.08
```

```
##
## Correlation matrix not shown by default, as p = 40 > 12.
## Use print(x, correlation=TRUE)  or
##       vcov(x)         if you need it
```

So there you have it. Factor nesting, it's great, write home to your mom about it.

The drawbacks of factor nesting

It's not all lollipops and rainbow bunnies, though. There are some things you need to keep in mind when you want to use factor nesting.

First of all, looking at simple effects presupposes that there was an interaction. When you cross factors, you have a test of that interaction: you have the interaction coefficient which shows the difference-of-differences (or, for an Nth-order interaction, that many "differences of"; if you're brave/foolhardy enough to test a four-way interaction with crossed factors, you will be the proud owner of a difference-of-differences-of-differences-of-differences test, and also probably sorely in need of a drink). But when you nest factors you don't have that. So you need to do something to test the interaction before you go on to look at your nested terms; otherwise, you would be committing the same sin as if you did ANOVA post-hoc tests without running an omnibus ANOVA first. Just finding that one simple effect was significant and the other not is not enough to prove that you had a significant interaction: the difference between a significant effect and a non-significant effect might not be significant in of itself. So what can you do? You can run the model with crossed factors to directly test the interaction, and then show it again with nested factors to zero in on the simple effects you care about. Or you can use model

comparison (i.e. log-likelihood tests) to show that the interaction is significant, and then go on to running the model with nested factors to show the simple effects once you've earned that privilege through showing a significant interaction. This latter strategy is exactly what I did for the studies we talked about here. In practice, a combination of both may be necessary. When you have factors with more than 2 levels, you always need log-likelihood tests, because there's no one coefficient that shows you the interaction; nevertheless, an omnibus interaction with the log-likelihood test can be coming from anywhere, so you still might need a crossed interaction coefficient to show that your two simple effects are actually different from one another. In fact, when I look back on that self-paced reading example we just discussed, if I had more statistically astute reviewers they might have demanded that I show a crossed interaction to prove that not only was the *some* simple effect significant and the *only some* effect not, but they also were different from one another, *and* that this difference-of-differences is significantly bigger than the corresponding difference-of-differences at the preceding region.

Another serious problem comes from the fact that the nested model is, technically, kicking out some of the main effects. Therefore, if you're not careful, it will screw up some model comparisons you might try. To see this in action, let's go back to our first data example, and let's use `anova()` with the package `{lmerTest}` to get ANOVA-style summaries of the main effects and interactions. I don't use lmerTest, but I know many of you crazy kids do.

```
rm(list=ls())

# Load the libraries we will need
library(lmerTest)
```

```
##
## Attaching package: 'lmerTest'
```

```
## The following object is masked from 'package:lme4':
##
##     lmer
```

```
## The following object is masked from 'package:stats':
##
##     step
```

```
# Read the data
load( url("http://users.ox.ac.uk/~cpgl0080/data.RData") )

# Nested and crossed models
anova( crossed_model <- lmer( RT ~ Tone*Condition + (1|Subjec
```

```
## Analysis of Variance Table of type III  with  Satterthwait
## approximation for degrees of freedom
##                Sum Sq Mean Sq NumDF  DenDF F.value    Pr(>
## Tone            73826   73826     1  20.05  2.0896    0.16
## Condition      775545  775545     1 412.77 21.9517 3.802e-
## Tone:Condition  66046   66046     1 413.53  1.8694    0.17
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
```

```
anova( nested_model <- lmer( RT ~ Tone/Condition + (1|Subject
```

```
## Analysis of Variance Table of type III  with  Satterthwait
## approximation for degrees of freedom
##                Sum Sq Mean Sq NumDF  DenDF F.value    Pr(>
## Tone            73826   73826     1  20.05  2.0896    0.16
## Tone:Condition 855213  427606     2 411.02 12.1034 7.808e-
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
```

See anything wrong with this picture? In the crossed model, the interaction looks non-significant, but in the nested model it looks significant! What gives? Well, you need to remember how model comparison works: a model with something is being compared to a model without it. With the crossed model, you're being shown what happens when you compare a model with the interaction and both main effects to a model with just the main effects; it's not significant because the interaction doesn't improve the model enough. With the nested model, on the other hand, one of the main effects is missing, but the

interaction still has to handle that factor. So what you're actually seeing is what happens with you compare the model with the interaction against a model with *only* Tone, and not Condition. You can tell this is going on because the interaction has 2 degrees of freedom here, whereas it should only have one degrere of freedom, like it does in the crossed model. So essentially, explainable variance that's really due to the main effect of Condition, is instead getting erroneously credited to the interaction. Thus you get what looks like a huge interaction effect, which is completely spurious. How can you avoid this? Well, one option is to just not use factor nesting when you do log-likelihood tests with `anova()`. Alternatively, my personal recommendation is not to use lmerTest at all; I do all my log-likelihood tests by hand (i.e., I manually create both models that I want to compare), so I always know what's going on, instead of trusting it to lmerTest and potentially missing important stuff like this.

A final drawback about factor nesting is that it's little-known. I've had at least two experienced R-using statisticians (who have published papers *about* mixed-effects models) looking at my code be like, "Whoa, what's that `/`?" and then, after I explained it to them, they would be like "Wow, cool, I didn't know you could do that!" In one case the person expressed that they were not sure if it was ok. While I've never been challenged on this during peer review, a colleague of mine has. So, while I have presented some arguments above for why I'm pretty sure this method is ok, I can't guarantee that you won't experience pushback. Of course, if anyone reading this knows something more about factor nesting—either something I haven't thought of that proves it's actually terrible, or some knock-down argument I haven't thought of that proves it's completely ok—I would love to hear it!

Anyway, long story short, I find this a very

useful tool, as long as I keep in mind the [sometimes serious] caveats discussed above. I hope you do too. So go nest away!

[1]This example will use our old friend: a subset of the data from Politzer-Ahles & Zhang (in press). Several conditions have been removed to end up with a long-format data frame consisting of repeated-measures data from 25 subjects in a 2×2 factorial design comparing reaction times in a paradigm including the factors Condition (Unrelated vs. Homogeneous word sets) and Tone (Rising Tone vs. Low Tone sets).

---

by Stephen Politzer-Ahles. Last modified on 2016-07-25.