Mixed Effect Models

Stacie Chadwick and Scott Nelson

This handout is based off of Chapters 14 and 15 from Bodo Winter's *Statistics For Linguists* book. It provides a basic introduction to mixed effect models. Mixed effects models are widely used in experimental analysis within the language sciences. The primary (but not exclusive!) reason for this is due to testing multiple participants as will be explained below.

Assumptions

There are certain assumptions that we want to make sure are met for our regression models. We have already discussed two of these:

1. **Normality** - the residuals of the model are approximately normally distributed.
2. **Constant Variance/Homoscedasticity** - the spread of the residuals should be about equal while moving along the regression line.

There is a third assumption that we have not discussed:

3. **Independence** - data points must be independent.

This assumption is the most important assumption to not violate. One of the reasons for this is because violations of the independence assumption lead to "massive effects" on the Type I error rate.

	Null Hypothesis is True	**Null Hypothesis is False**
Null Hypothesis Rejected	Type 1 Error	Correct
Null Hypothesis is Accepted	Correct	Type 2 Error

Imagine the null hypothesis was that a person has cancer. A Type I error is when it is true that a person has cancer, but some statistical test rejects this hypothesis thus implying that the person does not have cancer. This false negative is bad! Obviously language experiments aren't on the same level as a person having cancer, but hopefully it's clear why we should try to avoid Type I errors.

Dealing with Non-Independence

Imagine you were doing a speech production experiment and had the same speaker repeat the same item multiple times, then you would be introducing a dependency into the data. If you do not explicitly tell you're model about this then all you are doing is falsely increasing the sample size (i.e. - there is a difference between 20 data points from 1 speaker vs. 1 data point from 20 speakers.). So how do we avoid this? One option is to design experiments that minimize dependence between data points:

- Between-participant experiments where each participant only contributes one data point
- Aggregate all the data points for each participant so that they only provide one average data point
  - Be careful with this as you lose information!
  - The model will underestimate the amount of variation.

Mixed effect models

Another option is to used mixed models with varying intercepts and varying slopes. You may recall from earlier in the class that interactions were also just varying intercepts and varying slopes. So what is the difference in this case? It has to do with "random effects" vs. "fixed effects".

- Random effects: vary across experiments and introduce a dependency structure (participants, items). They must be categorical.
- Fixed effects: constant across experiments and independent. They are effects that are repeatable given a new set of participants. These can be either categorical or continuous.

The term **mixed effects** comes from the fact that these types of models *mix* both random effects and fixed effects.

- **Varying intercept model ('random intercept model'):** $y = \beta_{0j} + \beta_1 * trial + \epsilon$
- **Varying intercept, varying slope model ('random slope model'):** $y = \beta_{0j} + \beta_{1j} * trial + \epsilon$

One vary important point is that mixed-effect models do not estimate unique slopes and/or intercepts for each random effect. Instead, they estimate the *variation around the specified random effect*. So each random effect only adds one new term to the model. Imagine there was a random intercept for each participant. The model calculates a deviation score for each participant against the average intercept and uses this variation as a term in the model. Random effects don't directly predict the dependent variable, but instead provide a range around the fixed effect prediction for which we should expect to find a certain amount of the data.

Modeling Mixed-Effects

In this section we are going to create a random data set. The goal of this exercise is to highlight what exactly a mixed-effect model is doing. That is, what information it is trying to model. We will do this by simulating vowel duration data based on the log frequency of a word.

The basic structure of our data is:

- 6 participants
- 20 items (words)
- A unique logfreq for each word
- 120 duration values based on participant/item/logfreq/random noise

---

Let's start off by creating three vectors. One for participant numbers, one for word ids, and one for the log frequency of each word. These are the predictors/independent variables.

```
library(tidyverse) # load tidyverse
set.seed(666) # this ensures the same values each time you run the script
ppt_ids <- gl(6,20) #gl = generate levels; 6 levels, 20 times each
ppt_ids
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [36] 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4
##  [71] 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6
## [106] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
## Levels: 1 2 3 4 5 6
```

```
it_ids <- gl(20,1)
it_ids <- rep(it_ids,6) # repeat it_ids 6 times (once per participant)
it_ids
```

```
##   [1] 1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 1  2  3
##  [24] 4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 1  2  3  4  5  6
##  [47] 7  8  9  10 11 12 13 14 15 16 17 18 19 20 1  2  3  4  5  6  7  8  9
##  [70] 10 11 12 13 14 15 16 17 18 19 20 1  2  3  4  5  6  7  8  9  10 11 12
##  [93] 13 14 15 16 17 18 19 20 1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
## [116] 16 17 18 19 20
```

```
## Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

```
logfreqs <- round(rexp(20)*5,2) # rexp = random exponent; 20 of these rounded to two digits
logfreqs <- rep(logfreqs,6) # repeat logfreqs to match it_ids
logfreqs
```

```
##   [1]  2.74  9.82  0.70  0.05  2.76 17.57 13.06  7.62 10.38  3.11 14.71
##  [12]  1.62  0.13  3.06  3.80 15.70  1.12  0.53  1.38  3.05  2.74  9.82
##  [23]  0.70  0.05  2.76 17.57 13.06  7.62 10.38  3.11 14.71  1.62  0.13
##  [34]  3.06  3.80 15.70  1.12  0.53  1.38  3.05  2.74  9.82  0.70  0.05
##  [45]  2.76 17.57 13.06  7.62 10.38  3.11 14.71  1.62  0.13  3.06  3.80
##  [56] 15.70  1.12  0.53  1.38  3.05  2.74  9.82  0.70  0.05  2.76 17.57
##  [67] 13.06  7.62 10.38  3.11 14.71  1.62  0.13  3.06  3.80 15.70  1.12
##  [78]  0.53  1.38  3.05  2.74  9.82  0.70  0.05  2.76 17.57 13.06  7.62
##  [89] 10.38  3.11 14.71  1.62  0.13  3.06  3.80 15.70  1.12  0.53  1.38
## [100]  3.05  2.74  9.82  0.70  0.05  2.76 17.57 13.06  7.62 10.38  3.11
## [111] 14.71  1.62  0.13  3.06  3.80 15.70  1.12  0.53  1.38  3.05
```

Now that we have these three vectors, let's create a tibble that we can add further information to.

```
xdata <- tibble(ppt = ppt_ids, item = it_ids, freq = logfreqs)
xdata
```

```
## # A tibble: 120 x 3
##    ppt   item   freq
##    <fct> <fct> <dbl>
##  1 1     1      2.74
##  2 1     2      9.82
##  3 1     3      0.7
##  4 1     4      0.05
##  5 1     5      2.76
##  6 1     6     17.6
##  7 1     7     13.1
##  8 1     8      7.62
##  9 1     9     10.4
## 10 1     10     3.11
## # ... with 110 more rows
```

We can now add a column to the tibble for the average intercept value. But remember, it is unlikely that each participant will have the same intercept value, so we can add a second new column with a **random intercept** adjustment value for each participant.

```
xdata$int <- 300 # add a column for intercept where every value is 300
ppt_ints <- rnorm(6,sd=40) #rnorm is a random value from a normal distribution;
                           # 6 values with mean=0 and sd=40.
xdata$ppt_ints <- rep(ppt_ints,each=20) # repeat each value in ppt_ints 20 times
xdata
```

```
## # A tibble: 120 x 5
##    ppt   item   freq   int ppt_ints
##    <fct> <fct> <dbl> <dbl>    <dbl>
```

```
##  1 1     1      2.74   300    0.892
##  2 1     2      9.82   300    0.892
##  3 1     3      0.7    300    0.892
##  4 1     4      0.05   300    0.892
##  5 1     5      2.76   300    0.892
##  6 1     6     17.6    300    0.892
##  7 1     7     13.1    300    0.892
##  8 1     8      7.62   300    0.892
##  9 1     9     10.4    300    0.892
## 10 1     10     3.11   300    0.892
## # ... with 110 more rows
```

We can also add a column for **random intercepts** by item. We also want to add some trial-by-trial noise since the real world is messy.

```
item_ints <- rnorm(20, sd = 20) # smaller sd than participant sd
xdata$item_ints <- rep(item_ints, times = 6)
xdata$error <- rnorm(120, sd = 20) # no relation to item or participant
xdata
```

```
## # A tibble: 120 x 7
##    ppt   item   freq   int ppt_ints item_ints  error
##    <fct> <fct> <dbl> <dbl>    <dbl>     <dbl>  <dbl>
##  1 1     1      2.74   300    0.892     12.8   13.9
##  2 1     2      9.82   300    0.892      5.39 -19.1
##  3 1     3      0.7    300    0.892     46.0   22.8
##  4 1     4      0.05   300    0.892    -27.5  -28.3
##  5 1     5      2.76   300    0.892     13.2    4.02
##  6 1     6     17.6    300    0.892      9.67 -24.7
##  7 1     7     13.1    300    0.892     24.6   -2.42
##  8 1     8      7.62   300    0.892    -35.6   13.4
##  9 1     9     10.4    300    0.892     17.7   -6.67
## 10 1     10     3.11   300    0.892      7.79   5.03
## # ... with 110 more rows
```

Everything we've done so far has to do with random effects, but there is also the fixed effect of frequency that we need to take into account. Let's assume that the vowel duration decreases by 5ms for each increase of 1 in log frequency. Once we add this, we can add a final column that gives the duration based on both the random and fixed effects.

```
xdata$effect <- (-5) * xdata$freq
xdata <- mutate(xdata, dur = int + ppt_ints + item_ints + error + effect)
xdata
```

```
## # A tibble: 120 x 9
##    ppt   item   freq   int ppt_ints item_ints  error effect   dur
##    <fct> <fct> <dbl> <dbl>    <dbl>     <dbl>  <dbl>  <dbl> <dbl>
##  1 1     1      2.74   300    0.892     12.8   13.9  -13.7   314.
##  2 1     2      9.82   300    0.892      5.39 -19.1  -49.1   238.
##  3 1     3      0.7    300    0.892     46.0   22.8   -3.5   366.
##  4 1     4      0.05   300    0.892    -27.5  -28.3   -0.25  245.
```

```
##  5 1     5      2.76   300    0.892     13.2    4.02 -13.8   304.
##  6 1     6     17.6    300    0.892      9.67 -24.7  -87.8   198.
##  7 1     7     13.1    300    0.892     24.6   -2.42 -65.3   258.
##  8 1     8      7.62   300    0.892    -35.6   13.4  -38.1   241.
##  9 1     9     10.4    300    0.892     17.7   -6.67 -51.9   260.
## 10 1     10     3.11   300    0.892      7.79   5.03 -15.6   298.
## # ... with 110 more rows
```

In an actual experiment we would only have the predictors (participant, item, logfreq) and the duration values. So let's make a second tibble that only contains these items.

```
xreal <- select(xdata, -(int:effect))
xreal
```

```
## # A tibble: 120 x 4
##    ppt   item   freq   dur
##    <fct> <fct> <dbl> <dbl>
##  1 1     1      2.74  314.
##  2 1     2      9.82  238.
##  3 1     3      0.7   366.
##  4 1     4      0.05  245.
##  5 1     5      2.76  304.
##  6 1     6     17.6   198.
##  7 1     7     13.1   258.
##  8 1     8      7.62  241.
##  9 1     9     10.4   260.
## 10 1     10     3.11  298.
## # ... with 110 more rows
```

Now that we have our data set, let's use the *lmer()* function from the `lme4` package to run a mixed effect model. The syntax for fixed effects is the same we have been using: dependent~independent. For the random effects, you must use parentheses. Recall that for modeling, 1 is the stand in for the intercept. So if we wanted a random intercept for participants we would write (1|ppt). If we also wanted a random slope for one of the fixed effects, we would include that on the left side of the input: (1 + freq|ppt). We did not specify any random slopes while building our data set, but it's good to know how to do it if need be.

There is also an argument to *lmer()* called REML. Setting this to false ensures that the model uses 'maximum likelihood' for estimation. This is more important when comparing models against one another, but we are not going to explicitly do that here.

```
library(lme4)
xmdl <- lmer(dur ~ freq + (1|ppt) + (1|item), data = xreal, REML = FALSE)
summary(xmdl)
```

```
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula: dur ~ freq + (1 | ppt) + (1 | item)
##    Data: xreal
##
##      AIC      BIC   logLik deviance df.resid
##   1105.1   1119.0   -547.5   1095.1      115
##
```

```
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.09700 -0.60946  0.06483  0.60761  2.39754
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  item     (Intercept)  589.2   24.27
##  ppt      (Intercept) 1296.6   36.01
##  Residual              284.0   16.85
## Number of obs: 120, groups:  item, 20; ppt, 6
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  337.973     16.735  20.196
## freq          -5.460      1.004  -5.438
##
## Correlation of Fixed Effects:
##      (Intr)
## freq -0.339
```

So let's compare how the model did compared to the actual values that we used. I would say it does pretty good at estimating the random effect structure!

	Model Prediction	**Actual Value**
Intercept	337.973	300
Slope	-5.46	-5
Item Intercept	sd = 24.28	sd = 20
Participant Intercept	sd = 36.01	sd = 40
Error Term	sd = 20	sd = 16.85

Remember, mixed effect models only add 1 parameter for each random effect (a standard deviation and NOT a coefficient). If you're still wondering how this is different from a normal linear model with multiple effects with or without interactions, I suggest using *lm()* to make a model that accounts for the item/ppt variation in this way. You'll find that the models are doing something drastically different and require many more parameters!