PSC 204B Lab 6 - Multilevel Modeling

Simran Johal

February 16, 2024

```
library(tidyverse) # for data manipulation
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.4.3     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(foreign) # to read in our datasets
library(lme4) # for running multilevel models
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
library(lmerTest) # for inference on the multilevel models
```

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

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

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

```
library(performance) # to calculate ICC
library(ggplot2) # for plotting
library(ggpubr) # for arranging plots
```

Today we will be learning how to run multilevel regression models in R.

We will be using two different datasets, both of which I obtained from this website. This is the corresponding website for the book *Multilevel Analysis: Techniques and Applications, Third Edition* (Hox, Moerbeek, & Schoot, 2018), and thus contains some datasets that can be used to illustrate multilevel modeling.

The first dataset we will be using is the *popularity* dataset, which is a **simulated** dataset with information on 2000 students in 100 schools. This will help us get our feet wet in terms of multilevel modeling!

The second dataset we are using is also a simulated dataset, one that I created while working on a research project on detecting cohort effects in multilevel models. This is a repeated measures dataset: each individual was "assessed" 3 times on some cognitive test (or whatever you'd like), with each assessment occurring 2 years apart.

```
popular = read.spss("popular2.sav", to.data.frame = TRUE)
```

```
## re-encoding from CP1252
```

```
head(popular)
```

```
##   pupil class extrav  sex texp popular popteach    Zextrav       Zsex    Ztexp
## 1     1     1      5 girl   24     6.3        6 -0.1703149  0.9888125 1.486153
## 2     2     1      7  boy   24     4.9        5  1.4140098 -1.0108084 1.486153
## 3     3     1      4 girl   24     5.3        6 -0.9624772  0.9888125 1.486153
## 4     4     1      3 girl   24     4.7        5 -1.7546396  0.9888125 1.486153
## 5     5     1      5 girl   24     6.0        6 -0.1703149  0.9888125 1.486153
## 6     6     1      4  boy   24     4.7        5 -0.9624772 -1.0108084 1.486153
##     Zpopular   Zpopteach Cextrav Ctexp Csex
## 1  0.8850133  0.66905609  -0.215 9.737  0.5
## 2 -0.1276291 -0.04308451   1.785 9.737 -0.5
## 3  0.1616973  0.66905609  -1.215 9.737  0.5
## 4 -0.2722923 -0.04308451  -2.215 9.737  0.5
## 5  0.6680185  0.66905609  -0.215 9.737  0.5
## 6 -0.2722923 -0.04308451  -1.215 9.737 -0.5
```

The important variables in this dataset are:

- Pupil: id number for each individual student
- Class: indicates which class each student was in
- extrav: student's level of extraversion
- popular: student's level of popularity, based on asking other students in the class
- sex: student's sex

```
cognitive = read.csv("CognitiveExample.csv", header = TRUE)

head(cognitive)
```

```
##   ids      age1      age2      age3 cognitivescore1 cognitivescore2
## 1   1 17.994012 19.000000        NA       2.1592282        2.394423
## 2   2 16.485030 18.497006        NA       1.4676461        1.601049
## 3   3 16.904192 18.245509        NA       2.8402458        3.173507
## 4   4  5.251497  7.766467  9.946108       0.8900078        1.516589
## 5   5 13.215569 15.143713 17.658683       2.4915783        2.970501
## 6   6 12.544910 14.892216 16.401198       0.8785296        1.214507
##   cognitivescore3
## 1              NA
## 2              NA
## 3              NA
## 4        2.059625
## 5        3.595183
## 6        1.430492
```

The variables in this dataset are:

- ids: id variable for each individual

- age: age at each assessment occasion
- cognitivescore: individual's cognitive score at each assessment

This dataset is also currently in *wide* format, which means that each individual has one row, with repeated measures across different columns. However, the packages we will be using today require that the dataset is in *long* format (each individual has multiple rows, one for each timepoint) so I will also go over how to transform between the two.

Multilevel Modeling: Clustered within Schools

Suppose we were interested in predicting student's popularity, which is based on an average rating from all other students in the class, using their level of extraversion (and later gender) as predictors. I use the variable Cextrav, which is grand-mean centered (which means it was centered using the average extraversion score across all classes and all students).

```
regular_regression = lm(popular ~ Cextrav, data = popular)

summary(regular_regression)
```

```
##
## Call:
## lm(formula = popular ~ Cextrav, data = popular)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0021 -0.9021  0.0062  0.8896  3.8896
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.07645    0.02934  173.02   <2e-16 ***
## Cextrav      0.34585    0.02325   14.88   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.312 on 1998 degrees of freedom
## Multiple R-squared:  0.09973,    Adjusted R-squared:  0.09927
## F-statistic: 221.3 on 1 and 1998 DF,  p-value: < 2.2e-16
```

Let's calculate an intra-class correlation coefficient, to see how much variability there is across clusters. To do this, we need to calculate an intercept-only model, with a random effect on the intercept.

In `lme4`, you can write your regression equation like you normally would in `lm`. To indicate which variables you want random effects for, you put them in parentheses (1 indicates the intercept, and you can just use + to include other variables, such as your predictors if you want random effects on the slope), and then separate them from your grouping variable using the |

```
interceptonly = lmer(popular ~ 1 + (1 | class),
                     data = popular)

summary(interceptonly)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: popular ~ 1 + (1 | class)
##    Data: popular
##
```

```
## REML criterion at convergence: 6330.5
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.5655 -0.6975  0.0020  0.6758  3.3175
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  class    (Intercept) 0.7021   0.8379
##  Residual             1.2218   1.1053
## Number of obs: 2000, groups:  class, 100
##
## Fixed effects:
##             Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)  5.07786    0.08739 98.90973    58.1   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
performance::icc(interceptonly)
```

```
## # Intraclass Correlation Coefficient
##
##     Adjusted ICC: 0.365
##   Unadjusted ICC: 0.365
```

The intra-class correlation coefficient is 0.37, which indicates that there is a decent amount of variability due to the different classes! So we should prefer to use a multilevel model.

Just to illustrate the different models, we will start off with a model that only allows random effects on the intercept. That is, different classes are allowed to have different baseline levels of popularity, but the effect of extraversion on popularity is the same within each class.

```
mlm_popularity_intercept = lmer(popular ~ Cextrav + (1 | class),
                                data = popular)

summary(mlm_popularity_intercept)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: popular ~ Cextrav + (1 | class)
##    Data: popular
##
## REML criterion at convergence: 5832.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.0644 -0.7267  0.0165  0.7088  3.3587
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  class    (Intercept) 0.8406   0.9168
##  Residual             0.9304   0.9646
## Number of obs: 2000, groups:  class, 100
##
## Fixed effects:
##              Estimate Std. Error       df t value Pr(>|t|)
## (Intercept) 5.078e+00  9.421e-02 9.830e+01   53.90   <2e-16 ***
## Cextrav     4.863e-01  2.015e-02 1.965e+03   24.13   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##         (Intr)
## Cextrav 0.000
```

So in this model, we have two fixed effects: the intercept (which represents the level of popularity when extraversion is at the grand-mean), and the effect of extraversion on popularity. Since these are fixed effects, they represent the average effect across the different classrooms.

We also have a random effect on the intercept: this indicates how much variability there is in that expected level of popularity when extraversion is at its grand mean, across the different classrooms.

We can also expand our model to include a random effect on the slope - so now, different classes are allowed to have different intercepts, and the effect of extraversion on popularity is also allowed to differ.

```
mlm_popularity = lmer(popular ~ Cextrav + (1 + Cextrav | class),
                      data = popular)

summary(mlm_popularity)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: popular ~ Cextrav + (1 + Cextrav | class)
##    Data: popular
##
## REML criterion at convergence: 5779.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.1961 -0.7291  0.0146  0.6816  3.2217
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  class    (Intercept) 0.89178  0.9443
##           Cextrav     0.02599  0.1612   -0.88
##  Residual             0.89492  0.9460
## Number of obs: 2000, groups:  class, 100
##
## Fixed effects:
##             Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)  5.03127    0.09702 97.07723   51.86   <2e-16 ***
## Cextrav      0.49286    0.02546 89.69832   19.36   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##         (Intr)
## Cextrav -0.552
```

As before, we have fixed effects for the intercept and slope, and the interpretation of these parameters are going to mostly be the same as before. Only now, since we had a random effect on the slope, the fixed effect of extraversion is the *average* slope acrosss clusters, not just the effect that every classroom follows.

We also have a random effect for the intercept and slope: as before, we can see a decent amount of variability in the intercept, and a smaller amount of variance in the slope. We also get a correlation between the random effect, which is negative: this indicates that classrooms that have larger intercepts (higher values for students in this classroom with average levels of extraversion) tend to have lower slopes (a 1-unit change in extraversion leads to a smaller change in popularity).

How do our models compare?

```
summary(regular_regression)
```

```
##
## Call:
## lm(formula = popular ~ Cextrav, data = popular)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0021 -0.9021  0.0062  0.8896  3.8896
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.07645    0.02934  173.02   <2e-16 ***
## Cextrav      0.34585    0.02325   14.88   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.312 on 1998 degrees of freedom
## Multiple R-squared:  0.09973,    Adjusted R-squared:  0.09927
## F-statistic: 221.3 on 1 and 1998 DF,  p-value: < 2.2e-16
```

```
summary(mlm_popularity_intercept)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: popular ~ Cextrav + (1 | class)
##    Data: popular
##
## REML criterion at convergence: 5832.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.0644 -0.7267  0.0165  0.7088  3.3587
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  class    (Intercept) 0.8406   0.9168
##  Residual             0.9304   0.9646
## Number of obs: 2000, groups:  class, 100
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept) 5.078e+00  9.421e-02 9.830e+01   53.90   <2e-16 ***
## Cextrav     4.863e-01  2.015e-02 1.965e+03   24.13   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
```

```
##         (Intr)
## Cextrav 0.000
```

```r
summary(mlm_popularity)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: popular ~ Cextrav + (1 + Cextrav | class)
##    Data: popular
##
## REML criterion at convergence: 5779.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.1961 -0.7291  0.0146  0.6816  3.2217
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  class    (Intercept) 0.89178  0.9443
##           Cextrav     0.02599  0.1612   -0.88
##  Residual             0.89492  0.9460
## Number of obs: 2000, groups:  class, 100
##
## Fixed effects:
##             Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)  5.03127    0.09702 97.07723   51.86   <2e-16 ***
## Cextrav      0.49286    0.02546 89.69832   19.36   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##         (Intr)
## Cextrav -0.552
```

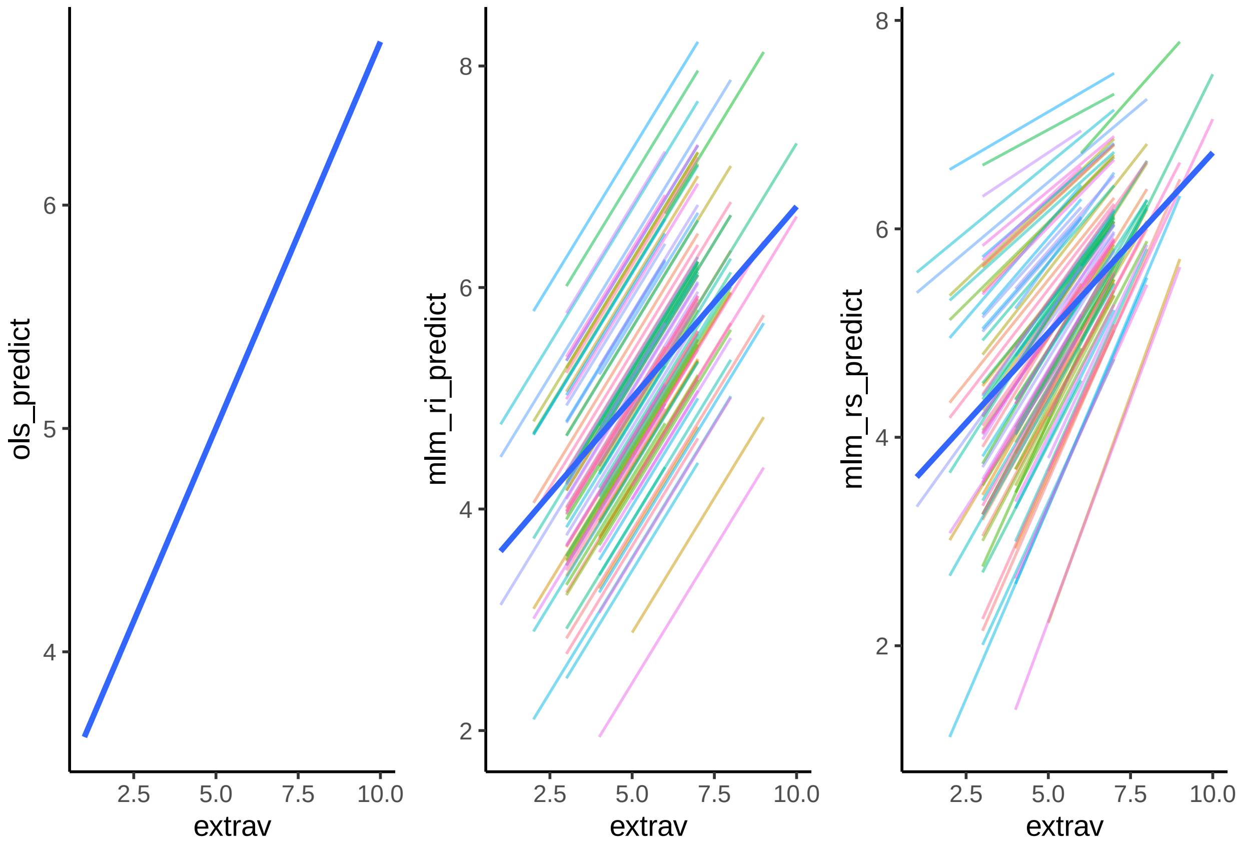
Across all models, we can see that the fixed effects estimates are very similar. The only difference is in the standard errors of the estimates (and in this case, only for the intercept, although the standard errors can also change for the slope). In particular, the standard error of the intercept is much larger in the multilevel model. Although this didn't affect our inference in this case, you can see how it might!

Also, this visualization is just to see how each of the models differs in terms of random effects.

```r
popular$ols_predict = predict(regular_regression)
popular$mlm_ri_predict = predict(mlm_popularity_intercept)
popular$mlm_rs_predict = predict(mlm_popularity)

g1 = ggplot(data = popular, aes(x = extrav, y = ols_predict))+
  geom_line(aes(color = as.factor(class)), alpha = 0.5)+
  geom_smooth(method = "lm")+
  theme_classic()+
  theme(legend.position = "none")

g2 = ggplot(data = popular, aes(x = extrav, y = mlm_ri_predict))+
  geom_line(aes(color = as.factor(class)), alpha = 0.5)+
  geom_smooth(method = "lm", se = FALSE)+
  theme_classic()+
  theme(legend.position = "none")
```

```r
g3 = ggplot(data = popular, aes(x = extrav, y = mlm_rs_predict))+
  geom_line(aes(color = as.factor(class)), alpha = 0.5)+
  geom_smooth(method = "lm", se = FALSE)+
  theme_classic()+
  theme(legend.position = "none")

ggarrange(g1, g2, g3, ncol = 3)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



In the first graph, clustering is not taken into account, and we are also assuming that each classroom has the same intercept and slope (that is, every classroom follows the plotted regression line). In the second image, different classrooms are allowed to have different intercepts, but their slopes are the same (which we can see based on the different colored lines all being parallel to each other). Finally, in the third model, different classrooms are allowed to have different intercepts and different slopes - visually, although the relation between extraversion and popularity is always positive, some classrooms have much stronger relations.

Multilevel Modeling: Repeated Measures

Converting Wide to Long

`lme4` requires that your data be in long format when you run the model, which means that each individual (or cluster) will have multiple rows. However, your data is not always ready-to-go in long format, so you'd have to convert between them. We can do this using the `pivot_longer` function from the tidyverse package.

```
# this becomes a little bit complicated because we have three variables that we need to collapse: age,
# this requires a little bit of regular expressions, which can differ based on the column names of your

cognitive_long = pivot_longer(data = cognitive,
                              cols = c("age1", "age2", "age3",
                                       "cognitivescore1", "cognitivescore2", "cognitivescore3"),
                              names_to = c(".value", "timepoint"),
                              names_pattern = "([A-Za-z]+)(\\d+)")

head(cognitive_long)
```

```
## # A tibble: 6 x 4
##     ids timepoint   age cognitivescore
##   <int> <chr>     <dbl>          <dbl>
## 1     1 1          18.0           2.16
## 2     1 2          19             2.39
## 3     1 3          NA            NA
## 4     2 1          16.5           1.47
## 5     2 2          18.5           1.60
## 6     2 3          NA            NA
```

```
# the created timepoint column is treated as a character - we want it to be numeric
cognitive_long$timepoint = as.numeric(cognitive_long$timepoint)
```

Using Timepoint as a Predictor

If you use a time-invariant variable as a predictor, running a multilevel model with repeated measures is pretty much exactly the same as what we did before with students clustered into classrooms. We simply include the variable representing the timepoint as our predictor - we might want to center this variable at the first occasion, however (e.g., have timepoints 0, 1, 2 instead of 1, 2, 3) just to make our intercept more interpretable.

```
# centering the timepoint variable at the first value, so that intercept represents baseline measuremen
cognitive_long$timepoint_centered = cognitive_long$timepoint - 1

head(cognitive_long)
```

```
## # A tibble: 6 x 5
##     ids timepoint   age cognitivescore timepoint_centered
##   <int>     <dbl> <dbl>          <dbl>              <dbl>
## 1     1         1  18.0           2.16                  0
## 2     1         2  19             2.39                  1
## 3     1         3  NA            NA                     2
## 4     2         1  16.5           1.47                  0
## 5     2         2  18.5           1.60                  1
## 6     2         3  NA            NA                     2
```

Just as before, we can compare what happens to our estimates and standard errors when we do or don't take clustering into account. Therefore, we can run a regression model looking at the change in cognitive scores across timepoints.

```
cogols = lm(cognitivescore ~ timepoint_centered, data = cognitive_long)

summary(cogols)
```

```
##
```

```
## Call:
## lm(formula = cognitivescore ~ timepoint_centered, data = cognitive_long)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.30274 -0.62191 -0.08739  0.56930  2.97432
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1.46862    0.03632  40.432  < 2e-16 ***
## timepoint_centered  0.24191    0.02928   8.261 3.26e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8871 on 1420 degrees of freedom
##   (78 observations deleted due to missingness)
## Multiple R-squared:  0.04586,    Adjusted R-squared:  0.04519
## F-statistic: 68.25 on 1 and 1420 DF,  p-value: 3.264e-16
```

We can calculate the intra-class correlation coefficient to see how much variability there is across individuals.

```
interceptonly_cognitive = lmer(cognitivescore ~ 1 + (1 | ids),
                               data = cognitive_long)

performance::icc(interceptonly_cognitive)
```

```
## # Intraclass Correlation Coefficient
##
##     Adjusted ICC: 0.869
##   Unadjusted ICC: 0.869
```

This is super high!! Probably because I generated this data so that there was a good amount of variability in cognitive scores across individuals :)

Now we can run a multilevel model to see how cognitive scores change as a function of timepoint. I will allow there to be random effects on both the intercept (individuals can differ in their initial cognitive scores) and slopes (individuals can differ in their rate of change across timepoints).

```
cogmlm_timepoint = lmer(cognitivescore ~ timepoint_centered + (1 + timepoint_centered | ids), data = cog

summary(cogmlm_timepoint)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: cognitivescore ~ timepoint_centered + (1 + timepoint_centered |
##     ids)
##    Data: cognitive_long
##
## REML criterion at convergence: 147.5
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -3.15530 -0.31629 -0.00509  0.33719  2.50562
##
## Random effects:
##  Groups   Name               Variance Std.Dev. Corr
##  ids      (Intercept)        0.756848 0.86997
##           timepoint_centered 0.017853 0.13362  0.18
##  Residual                    0.002441 0.04941
## Number of obs: 1422, groups:  ids, 500
##
## Fixed effects:
##                     Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)        1.448e+00  3.896e-02 4.989e+02   37.17   <2e-16 ***
## timepoint_centered 3.035e-01  6.255e-03 4.971e+02   48.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## tmpnt_cntrd 0.158
```

The fixed effects parameters can still be interpreted as we've done before:

- The intercept represents the expected cognitive score when timepoint = 0, which is the first timepoint. Therefore, on average, individuals have an initial cognitive score of 1.47.
- The slope represents the average change in cognitive score per timepoint. So, on average, as individuals move from one timepoint to th next, their cognitive scores are expected to increase by 0.15 points.

We can also look at the random effects, where there appears to be a decent amount of variability in both the intercept and slope, and that the intercept and slope are positively correlated (which means that individuals who tend to have higher cognitive scores at the initial timepoint tend to have higher rates of change).

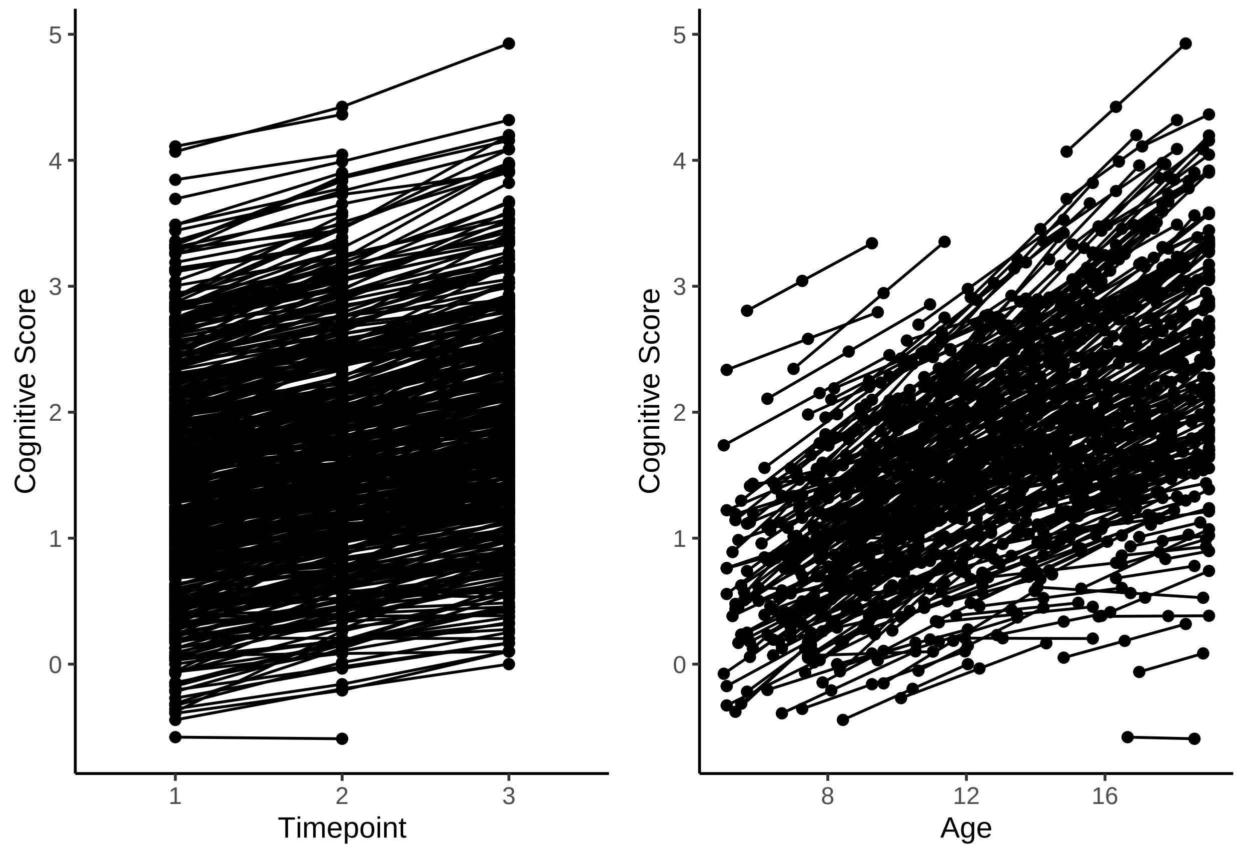
We also see that, like in our school example, the standard error of the intercept has increased between the OLS regression and the multilevel model, although the standard error of the slope has decreased.

Using Age as a Predictor

However, although we were able to use timepoint as a predictor in our previous model, that might not be very interesting to us. We might be more interested in how cognition develops with *age*, but this is not synonymous with timepoint in this example because we had a wide range of ages at our initial timepoint. Therefore, we might want to include age itself as a predictor in our model.

```r
g1 = ggplot(data = cognitive_long, aes(x = as.factor(timepoint), y = cognitivescore, group = ids))+
  geom_point()+
  geom_line()+
  theme_classic()+
  labs(x = "Timepoint", y = "Cognitive Score")

g2 = ggplot(data = cognitive_long, aes(x = age, y = cognitivescore, group = ids))+
  geom_point()+
  geom_line()+
  theme_classic()+
  labs(x = "Age", y = "Cognitive Score")

ggarrange(g1, g2, ncol = 2)
```

```
## Warning: Removed 78 rows containing missing values (`geom_point()`).

## Warning: Removed 78 rows containing missing values (`geom_line()`).

## Warning: Removed 78 rows containing missing values (`geom_point()`).

## Warning: Removed 78 rows containing missing values (`geom_line()`).
```



The problem with simply including age is that it is what we call a time-varying predictor: at each timepoint, people's value on this variable also changes. If we simply include time-varying predictors in our model "as is", the resulting fixed slope represents both between-person effects (differences across individuals) and within-person effects (differences across time for a single individual).

The way to separate out these effects is to include 2 predictors: one is age, centered at some person-specific value (e.g., average age or age at the initial timepoint), and the second is that person-specific value itself. Therefore, the regression coefficient for person-centered age represents the within-person effect of age, and the person-specific mean (which can also be centered at some grand value).

```
# create the person-specific value: I will use age at the initial timepoint

person_initage = as.data.frame(cognitive_long %>%
  group_by(ids) %>%
  filter(timepoint == 1) %>%
  ungroup() %>%
  select(age))

cognitive_long$person_initage = rep(person_initage$age, each = 3)

# then subtract each person's age from the initial age

cognitive_long = cognitive_long %>%
  mutate(personcenteredage = age - person_initage)

head(cognitive_long %>% dplyr::select(ids, age, person_initage, personcenteredage), 12)
```

```
## # A tibble: 12 x 4
##      ids   age person_initage personcenteredage
##    <int> <dbl>          <dbl>             <dbl>
##  1     1 18.0           18.0               0
##  2     1 19             18.0               1.01
##  3     1 NA             18.0              NA
##  4     2 16.5           16.5               0
##  5     2 18.5           16.5               2.01
##  6     2 NA             16.5              NA
##  7     3 16.9           16.9               0
##  8     3 18.2           16.9               1.34
##  9     3 NA             16.9              NA
## 10     4  5.25           5.25              0
## 11     4  7.77           5.25              2.51
## 12     4  9.95           5.25              4.69
```

```
# Now we run the model
# only personcenteredage gets a random slope

cogmlm_age = lmer(cognitivescore ~ person_initage + personcenteredage + (1 + personcenteredage | ids),
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.730202 (tol = 0.002, component 1)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly uniden
##  - Rescale variables?

## Warning in as_lmerModLT(model, devfun): Model may not have converged with 1
## eigenvalue close to zero: 6.2e-09
```

```
summary(cogmlm_age)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula:
## cognitivescore ~ person_initage + personcenteredage + (1 + personcenteredage |
##     ids)
##    Data: cognitive_long
##
## REML criterion at convergence: -10915.1
##
## Scaled residuals:
##        Min         1Q     Median         3Q        Max
## -7.586e-06 -6.075e-07 -1.400e-08  5.724e-07  7.044e-06
##
## Random effects:
##  Groups   Name              Variance  Std.Dev.  Corr
##  ids      (Intercept)       3.184e-01 5.643e-01
##           personcenteredage 3.858e-03 6.211e-02 0.48
##  Residual                   5.307e-14 2.304e-07
## Number of obs: 1422, groups:  ids, 500
##
## Fixed effects:
##                    Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)       -0.180920   0.074260  0.479253  -2.436 0.416433
## person_initage     0.143048   0.006098  0.450729  23.459 0.157062
## personcenteredage  0.155084   0.002776  2.193195  55.869 0.000168 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) prsn_n
## person_intg -0.941
## persncntrdg  0.169 -0.006
## optimizer (nloptwrap) convergence code: 0 (OK)
## Model failed to converge with max|grad| = 0.730202 (tol = 0.002, component 1)
## Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
```

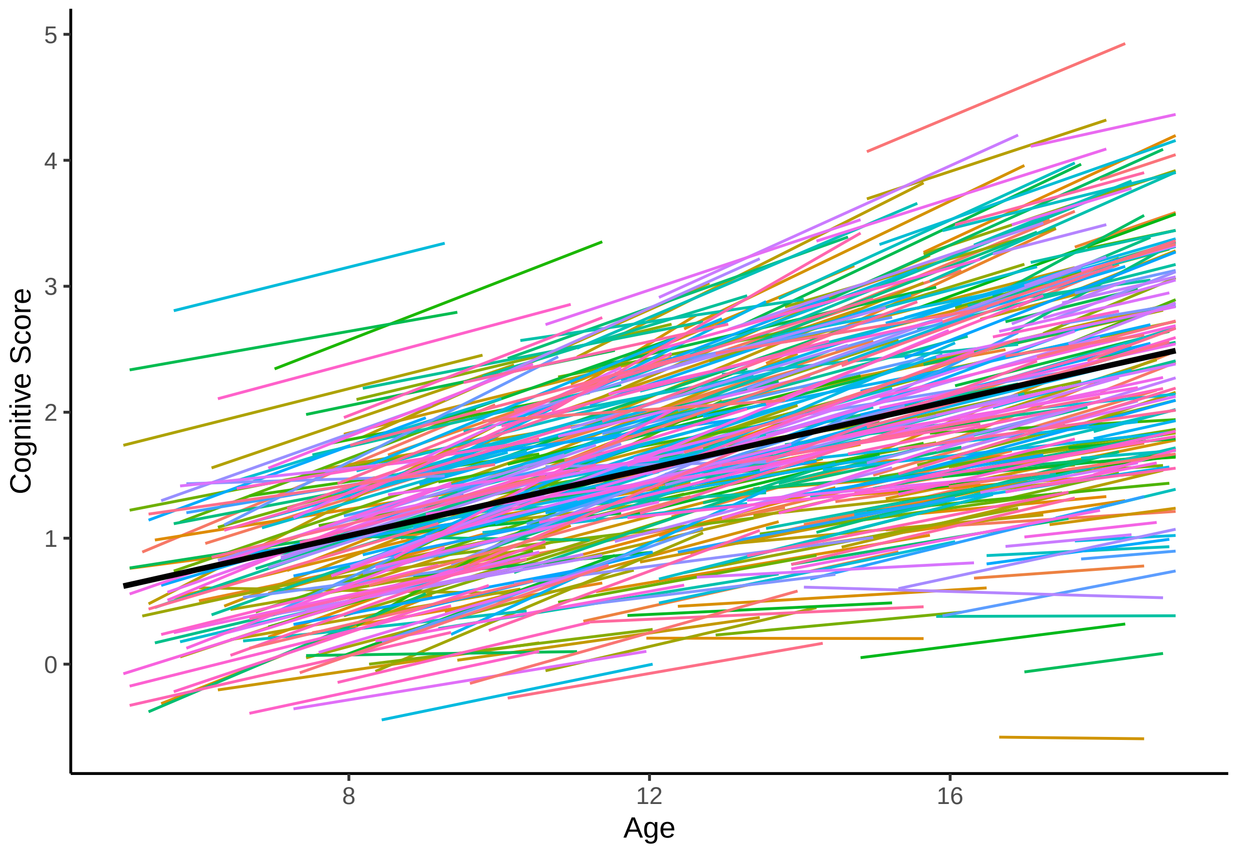
So now we have three parameters:

- The intercept, which represents the expected cognitive score for a person who is at the average initial age
- The slope of initial age, which represents between-person differences in cognitive scores due to age. This slope is positive, so people who are older at the initial timepoint tend to have higher cognitive scores.
- The slope of person-centered age, which represents the effect of age on cognitive scores. For every additional year a person gets older, their cognitive scores are expected to increase by 0.15 points.

And to visualize:

```
# the reason I'm creating a new dataframe instead of adding on a new column is because not every indivi
cognitive_predict = data.frame(id = cognitive_long$ids[complete.cases(cognitive_long$age)],
                               age = cognitive_long$age[complete.cases(cognitive_long$age)],
                               mlm_agepredict = predict(cogmlm_age))

ggplot(data = cognitive_predict, aes(x = age, y = mlm_agepredict))+
  geom_line(aes(color = as.factor(id)))+
  geom_smooth(method = "lm", se = FALSE, color = "black")+
  theme_classic()+
  theme(legend.position = "none")+
  labs(x = "Age", y = "Cognitive Score")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Additional Resources

- I have taken longitudinal data classes in the Human Development Department before, which covered briefly multilevel modeling for repeated measures data
- Book: Data Analysis Using Regression and Multilevel/Hierarchical Models by Andrew Gelman and Jennifer Hill
- Lesa Hoffman (Professor at University of Iowa) also has materials from her courses on multilevel modeling: Link to her website