Exercise 03 - Basics of linear mixed models

Zoltan Kekecs

18 november 2021

Contents

1 Abstract

In this exercise you will learn important concepts related to linear mixed models. The exercise also describes how to formulate linear mixed models. You will also learn how to make a decision about which random effect term to use, and what to report about the mixed models.

2 Data management and descriptive statistics

2.1 Loading packages

You will need the following packages for this exercise.

```
library(psych)  # for describe\t
library(tidyverse)  # for tidy code and ggplot\t
library(cAIC4)  # for cAIC\t
library(r2glmm)  # for r2beta\t
library(lme4)  # for lmer
library(lmerTest)  # to get singificance test in lmer
library(MuMIn)  # for r.squaredGLMM
```

2.2 Custom function

This is a function to extract standardized beta coefficients from linear mixed models. This function was adapted from: https://stackoverflow.com/questions/25142901/standardized-coefficients-for-lmer-model

```
stdCoef.merMod <- function(object) {
    sdy <- sd(getME(object, "y"))
    sdx <- apply(getME(object, "X"), 2, sd)
    sc <- fixef(object) * sdx/sdy
    se.fixef <- coef(summary(object))[, "Std. Error"]
    se <- se.fixef * sdx/sdy
    return(data.frame(stdcoef = sc, stdse = se))
}
```

2.3 Load test performance data

In this exercise we will work with simulated data about school test performance and reading ability. Let's say that we are interested in predicting how much reading ability influences performance on a history test among 4th grade primary school children. Test performance is quantified by the test score achieved, and reading ability is measured by a reading time in seconds of a short text, which was measured separately from the history test at the beginning of the school year.

The researchers hypothesize that the reading time of the child will affect their test performance, the faster the reading ability, the higher the test performance.

Participants come from different classes in the same school which is denoted in the variable 'class'.

Variables:

- test_score - This is the test score on the history test.
- reading_time - Time (in seconds) it took to read a short text for the child.
- class - factor variable indicating the school class the child belongs to in primary school. Factor levels: class_1, class_2, class_3, class_4.

We will work with two datasets simulated for the same scenario. The effect of reading_time on the test_score is different in these two datasets. (we will see the difference later)

```r
# load data\t
data_test_performance_int = read_csv("https://raw.githubusercontent.com/kekecsz/SIMM61-Course-materials/
```

```
## Rows: 80 Columns: 3

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): class
## dbl (2): test_score, reading_time

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# asign class as a grouping factor\t
data_test_performance_int = data_test_performance_int %>%
    mutate(class = factor(class))

data_test_performance_slope = read_csv("https://raw.githubusercontent.com/kekecsz/SIMM61-Course-material
```

```
## Rows: 80 Columns: 3

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): class
## dbl (2): test_score, reading_time

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
data_test_performance_slope = data_test_performance_slope %>%
    mutate(class = factor(class))
```

2.4 Check the dataset

As always, you should start by checking the dataset for coding errors or data that does not make sense, by eyeballing the data through the data view tool, checking descriptive statistics and through data visualization.

3 Basics of linear mixed models

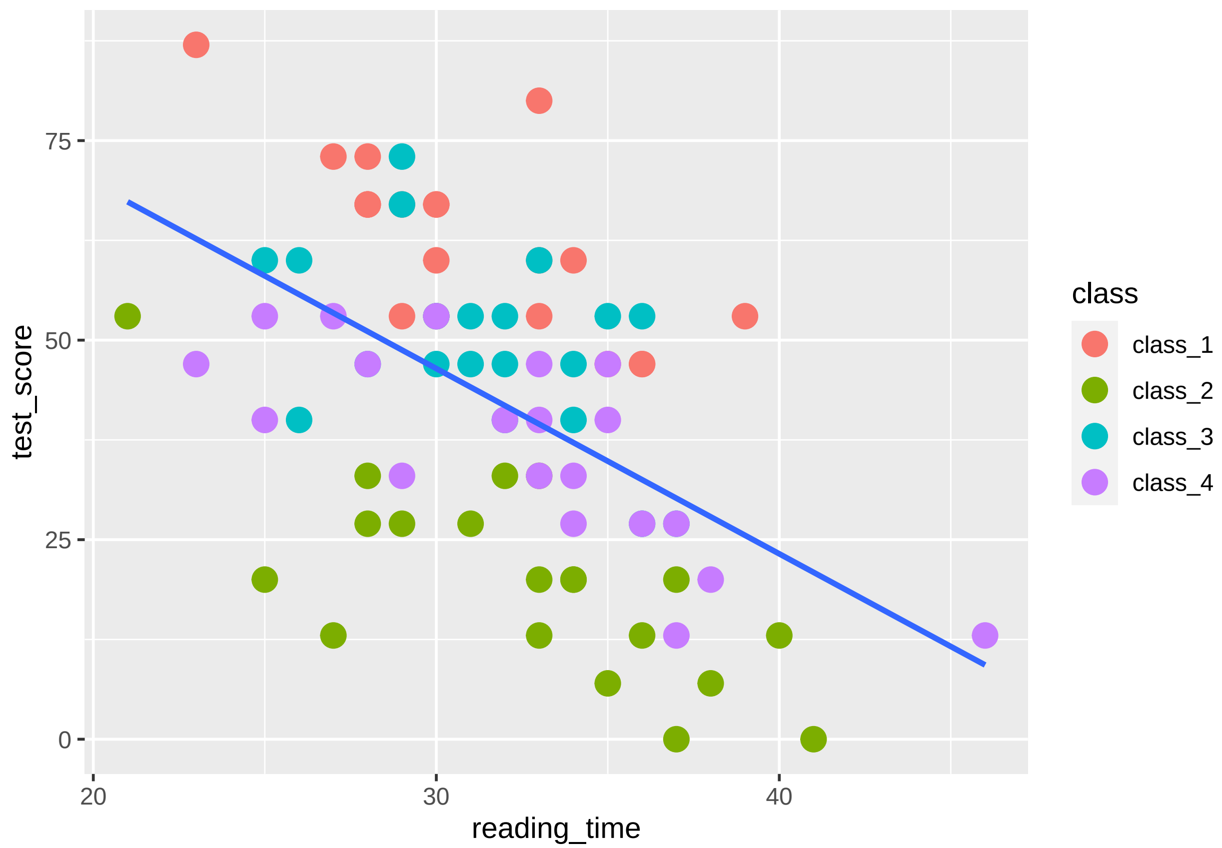
3.1 exploring clustering in the data

Let's plot the relationship for the simple regression model of test_score ~ reading_time on a scatterplot. It seems that there is a clear negative relationship between reading_time and the test_score from the children. However the variability seems pretty large.

If we look at the color of the dots showing class membership, we may notice that children coming from the same class are grouped together on the scatterplot. This indicates that there is some "clustering" in the data, so observations might not be completely independent from each other.

```r
data_test_performance_int %>%
    ggplot() + aes(y = test_score, x = reading_time) + geom_point(aes(color = class),
    size = 4) + geom_smooth(method = "lm", se = F)
```

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

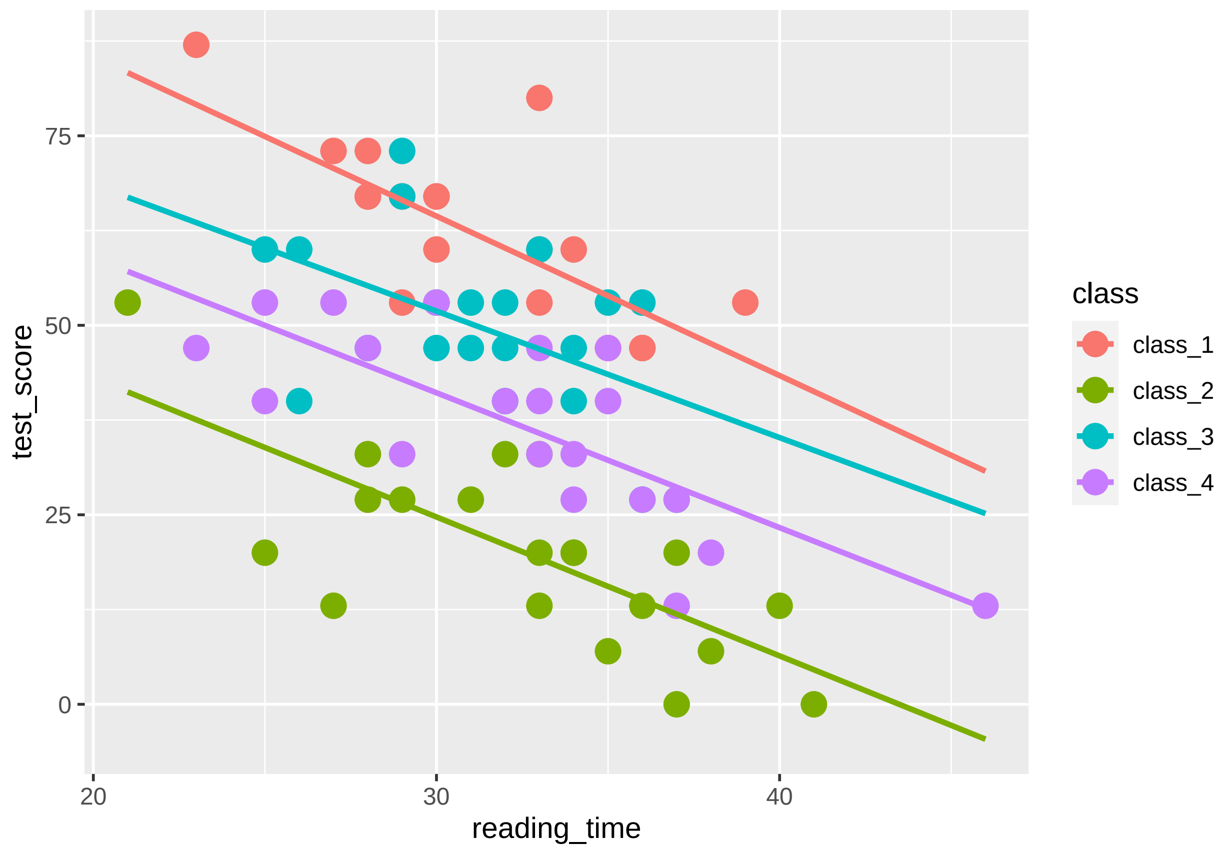


Let's see if class membership can explain some of this variability. We can plot the regression lines for each class separately. This seems to be able to explain some of the variability in the data, bringing the regression lines closer to the actual observations. So it seems that it would be worthwhile to take into account class membership of the participant when assessing their test performance to get good predictions.

```
int_plot = data_test_performance_int %>%
    ggplot() + aes(y = test_score, x = reading_time, color = class) +
    geom_point(size = 4) + geom_smooth(method = "lm", se = F,
    fullrange = TRUE)

int_plot
```

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



3.2 About the inadequacy of the fixed effect model solution

We cannot just ignore this clustering in the data, because it would make the our standard error, confidence interval, and p-value estimates imprecise. So we need to incorporate this clustering in our model somehow.

One solution could be to include class as a regular fixed effect predictor in our model.

```
mod_fixed_int = lm(test_score ~ reading_time + class, data = data_test_performance_int)

mod_fixed_int
```

```
##
## Call:
## lm(formula = test_score ~ reading_time + class, data = data_test_performance_int)
##
## Coefficients:
##  (Intercept)  reading_time  classclass_2  classclass_3  classclass_4
##      119.280        -1.838       -39.418       -12.057       -22.901
```

This model would be no doubt more precise at predicting the test_score in the same school´than a model without class as a predictor, and if the goal with your model is to only use the model in these four classes, this is a great solution.

However, we will not be able to use this regression equation outside of the four classes we observed in our data, since we only have the regression coefficients for these classes, and even the intercept is locked to one of the classes. So if we want to use the model at another school, the regression equation would be useless, or very imprecise.

3.3 Mixed models

We use mixed models when we have many possible levels for a categorical predictor in our model, but we only have information about a small subset of those levels in our dataset. As we have seen above, knowing the effects of these particular levels, like the effect of class 1, 2, 3, and 4 in school A, would not be very useful when we use the regression equation on new data class 7 in school B. So this is more of a nuisance variable for us, but we still have to incorporate it in our models to get accurate confidence intervals, p-values, and standard errors.

By using linear mixed models we can take into account that data is clustered according to class membership, without having to enter class as a classical predictor in our model. Instead, we can enter it into the model as a "random effect" predictor.

3.4 Two types of effects in mixed models (types of predictors)

Predictors can be entered into mixed effect models as having two types of effects: the predictor can either have a "fixed effect" on the outcome or a "random effect" on the outcome.

Fixed effects predictors - These are the good old predictors that we are familiar with from the regular linear models. We would like to estimate this effect so that we can use it in future predictions.

Random effect predictors - These are the "nuisance variables", the categorical variables that have many possible levels in the wild but we only have a few of these levels observed in our sample. We do not want to estimate the effects of these predictors on the outcome, and don't want these in our regression equation, because we will not be able to use the coefficients for the observed levels for prediction on new data. But we cannot simply disregard these variables, because they influence the outcome, and thus, because the observations are related to each other based on this variable, this affects how precise our estimation of the model's effectiveness is and how precise the model coefficient's estimates are. So we need to factor the effect of these variables into our models to get a clear picture about the precision of our estimates. The idea is to assume that the groups or clusters represented by this nuisance variables have a "random effect" on the outcome variable, and the effect of the groups are drawn from a normal distribution. This means that if we took enough of these groups, their effect would cancel out, since some would have a positive effect, while others would have a negative effect on the outcome, so it is not necessary to know exactly what the effect of each possible group is on the outcome, all we need to know is that there is this a variable that creates some "random noise" in our data, and the amount of noise it generates. If we know this, we can factor this into the precision of our other estimates.

Models that contain both fixed and random effect predictors are called **mixed models**. So this is where the name comes from.

3.5 Two types of random effects

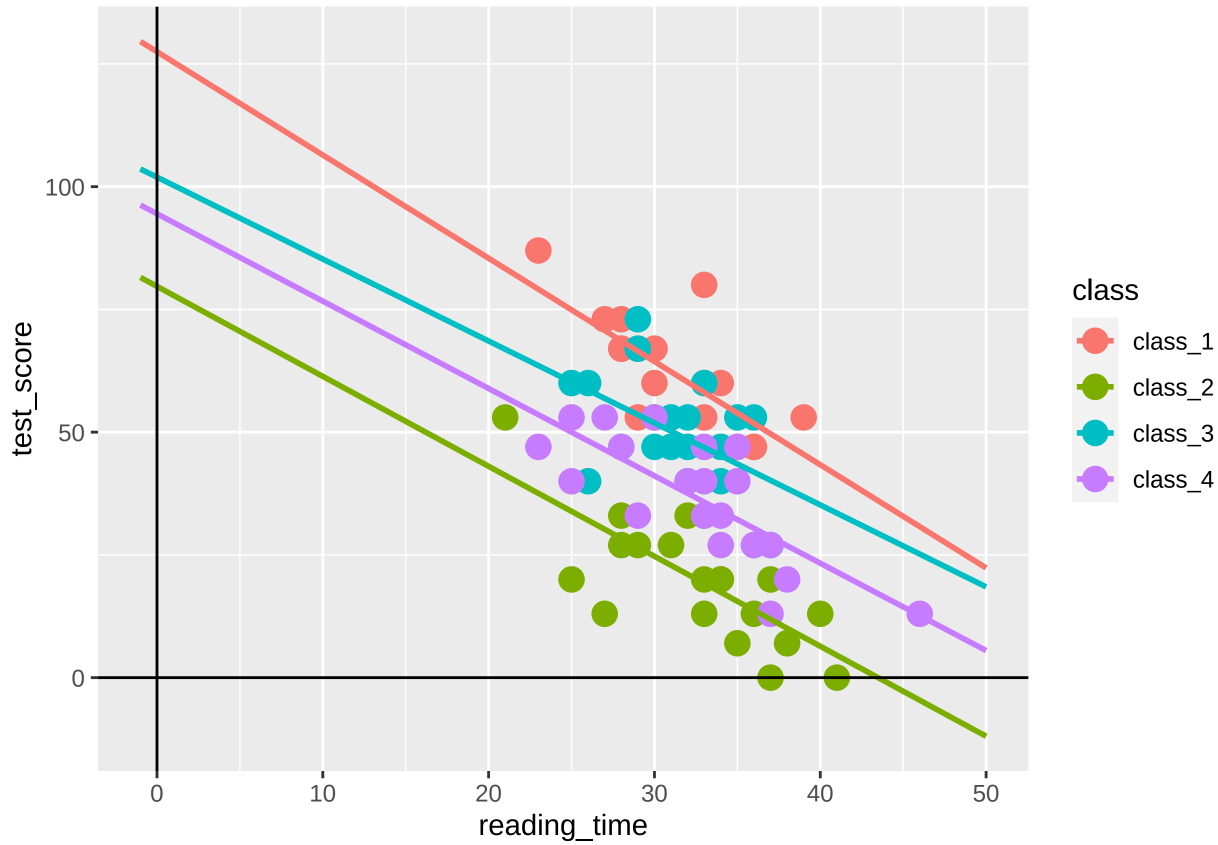
There is generally two ways in which a random effect predictor can influence the outcome variable:

random intercept, but no random slope: it is possible that predictor has a **direct effect** on the outcome, that is, the clusters within this predictor are only different in how high or low they are on the value of the outcome variable on average, but the effect of the fixed effect predictors are the same in all of the clusters. This is what we see in the dataset named data_test_performance_int. You can observe that in some classes children have higher test scores on average, while in other classes children have lower test scores. Also notice that reading time affects test score by roughly the same amount in all classes in data_test_performance_int.

If we build different regression models for each class and visualize them on a plot, the scatterplot shown that the regression lines cross the y axis at different places (different intercepts), but the regression lines are almost parallel to each other (similar slopes for the regression lines), indicating that the effect of reading_time is similar in the classes.

```
int_plot + xlim(-1, 50) + geom_hline(yintercept = 0) + geom_vline(xintercept = 0)
```

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



random intercept, AND random slope: Now let's look at the other dataset we loaded in the beginning of the exercise, saved in the data_test_performance_slope object. This dataset is also simulated, and it comes from the same scenario as the data_test_performance_int dataset, but the data looks a bit different. When we plot the regression lines for the classes separately, we find that not only the mean value of the test score is different across classes, but the effect of reading time also seems to be different in the different classes. So there is an **interaction** between reading time and class when it comes to determining test scores.

For example in Class 1 (the red dots), reading time seems to almost make no difference, everyone seems to have roughly the same test score regardless of reading time. On the other hand in classes 2 and 4, children with lower reading times seem to have much higher test score than their classmates with higher reading times.

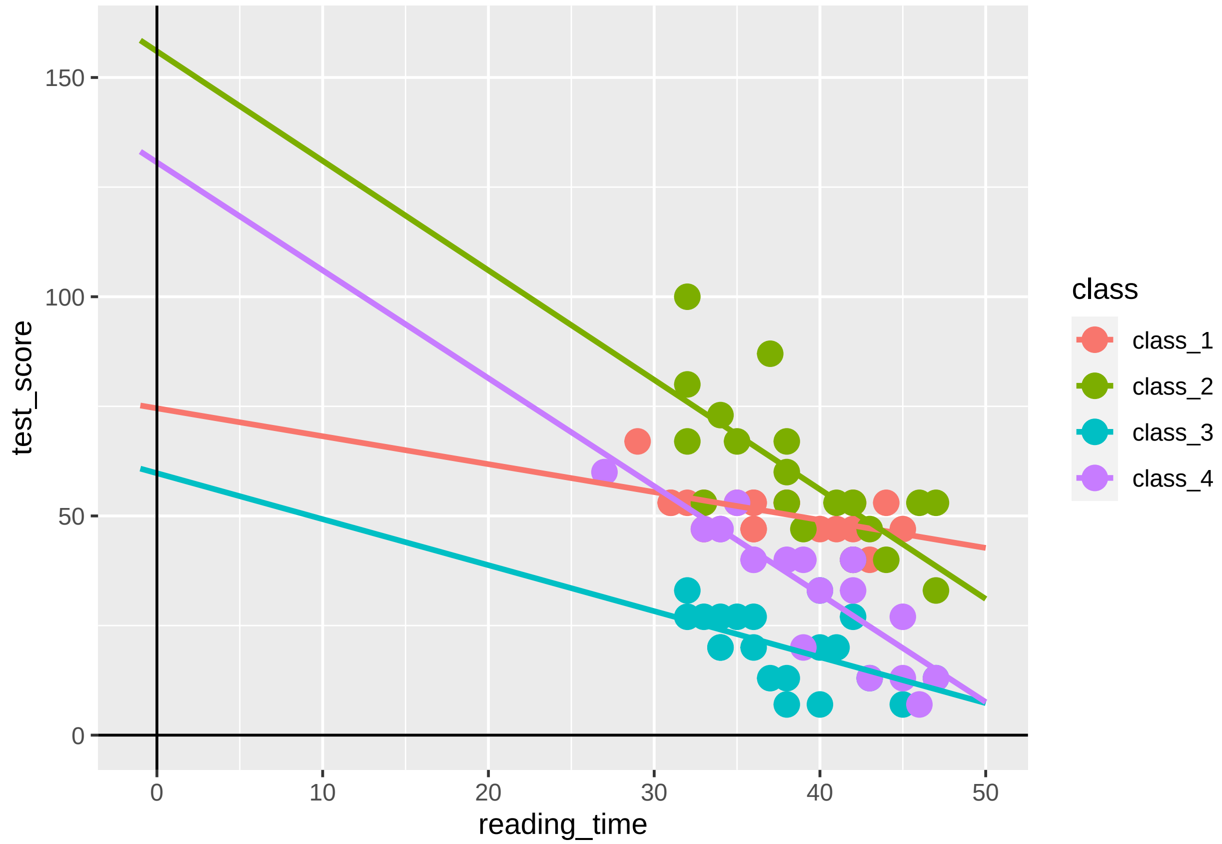
On the scatterplot, you can easily spot this by noticing that both the intercepts and the slope of the regression lines are different across classes.

```r
slope_plot = data_test_performance_slope %>%
    ggplot() + aes(y = test_score, x = reading_time, color = class) +
    geom_point(size = 4) + geom_smooth(method = "lm", se = F,
    fullrange = TRUE) + xlim(-1, 50) + geom_hline(yintercept = 0) +
    geom_vline(xintercept = 0)
slope_plot
```

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

## Warning: Removed 1 rows containing non-finite values (stat_smooth).

## Warning: Removed 1 rows containing missing values (geom_point).
```



3.6 Building mixed models in R

Let's see how we can build a model where we account for both the fixed and the random effects in a linear model.

We will fit three different models on the data_test_performance_slope dataset, a simple **fixed effect model** with no random effects just as a comparison, a **random intercept model**, and a **random slope model**.

Remember that based on the above plots, we suspect that the random effect predictor, class, has an effect on both the mean of the outcome (intercept), and the effect of the fixed effect predictor (slope) in the data_test_performance_slope dataset. So in reality we would probably only fit the random slope model. The other models are just here to show you how they differ in prediction effectiveness and how to formulate them.

First, let's fit the simple linear regression model as we have learned in the previous exercises. Notice that this model only contains a single *fixed effect* predictor, reading_time, so we will save this regression model to a model called mod_fixed.

simple regression (only fixed effect term, no random effect):

```
mod_fixed = lm(test_score ~ reading_time, data = data_test_performance_slope)
```

random intercept model:

The formula looks similar to the simple regression model, however, we use a different function here: lmer(), and we include a random intercept (but not random slope) in the model by adding '+ (1|class)' to the model formula.

This means that we allow the model to **fit a separate regression line** to each of the clustering levels (in this case, classes), but we **restrict** it so that all of these regression lines have to **have the same slope**.

We would do this if we suspected that the clustering variable (random effect predictor) has an effect on the outcome variable, and has no interaction with reading_time, that is, class membership has no influence on the effect of the fixed effect predictor. So based on what we saw on the plots above, this would be a good fit for the data_test_performance_int dataset. But here we fit this to the data_test_performance_slope dataset, so we can compare the effectiveness of random slope and random intercept models.

```
mod_rnd_int = lmer(test_score ~ reading_time + (1 | class), data = data_test_performance_slope)
```

random slope model (allowing BOTH random intercept and random slope):

The formula looks the same as for the random intercept model, with the difference that in the part of the formula describing the random effects, instead of '+ (1|class)' we now have '+ (reading_time|class)'.

This means that we allow the model to **fit a separate regression line** to each of the clustering levels (in this case, classes), and we **do not restrict the slope or the intercept** of this regression line (other than the line needs to be linear). We do this because we suspect that the clustering variable (random effect predictor) influences both the mean value of the outcome and the effect of the fixed effect predictor (reading_time). In other words we suspect that the random effect predictor has both a main effect on the outcome and an interaction with the fixed predictor.

Note that when I use the term "random slope" what I really mean is a model where we allow for a **random intercept AND a random slope effect**. It is very rare that we would like to model a random slope with restricting the intercept to be constant, because that is rarely the case in real life.

```
mod_rnd_slope = lmer(test_score ~ reading_time + (reading_time |
    class), data = data_test_performance_slope)
```

3.6.1 Common warning messages

Model failed to converge: It is common to get a "Model failed to converge" warning when using random slope models. The mixed model fitting technique uses an iterative process which tries to find the optimal solution for the regression problem. It tries to find this solution a given number of times, each time getting getting closer and closer to a "good enough" solution. The convergence warning indicates that the solution that the model arrived at by the end of the given number of iterations (tries) is "not good enough" according to some threshold. This is a pretty complex issue, with a number of possible solutions. You can read more about this here: https://biologyforfun.wordpress.com/2018/04/09/help-i-have-convergence-warnings/

In general the less complex the model, the less chance there is for a convergence failure. So one solution is to decrease the complexity of the model, by for example removing a few random slopes.

In some cases it is not possible to decrease the complexity of the model further, because that would be theoretically not varranted. In this case the approach we use in this example is to use a different optimizer which will find the right solution in another way. Generally speaking, the solution returned by the optimizer will be more trustworthy than the solution of a model with failed convergence.

```
mod_rnd_slope_opt = lmer(test_score ~ reading_time + (reading_time |
    class), control = lmerControl(optimizer = "Nelder_Mead"),
    data = data_test_performance_slope)
```

Singular fit: This warning tells you that one or more variances estimated for the random effect terms is (very close to) zero, meaning that that random effect predictor seems to be not useful in modeling the data. Here are some recommendations on how to deal with this warning from the leading statisticians on the field of mixed effect modeling:

- avoid fitting overly complex models, such that the variance-covariance matrices can be estimated precisely enough (Matuschek et al. 2017)
- use some form of model selection to choose a model that balances predictive accuracy and overfitting/type I error (Bates et al. 2015, Matuschek et al. 2017)

- “keep it maximal”, i.e. fit the most complex model consistent with the experimental design, removing only terms required to allow a non-singular fit (Barr et al. 2013)

(References:

Bates D, Kliegl R, Vasishth S, Baayen H. Parsimonious Mixed Models. arXiv:1506.04967, June 2015.

Barr DJ, Levy R, Scheepers C, Tily HJ. Random effects structure for confirmatory hypothesis testing: Keep it maximal. Journal of Memory and Language, 68(3):255–278, April 2013.

Matuschek H, Kliegl R, Vasishth S, Baayen H, Bates D. Balancing type I error and power in linear mixed models. Journal of Memory and Language, 94:305–315, 2017.)

3.7 Assessing model fit of mixed models and deciding which model to use

As always, when selecting model components, you should make decisions based on theoretical considerations and prior research results. So in this case, whether it makes sense theoretically for class membership to influence the slope of the effect of reading_time as well or not, or whether previous exploratory studies have shown that modeling a random slope produces much better model fit than a simple random intercept model.

In some cases, we are exploring the topic without strong prior research or theoretical clues. In these cases it may be appropriate to select between random slope and random intercept models based on model fit. But this decision needs to be clearly documented in the publication. The best if such decisions are pre-registered before data collection.

3.7.1 visualization

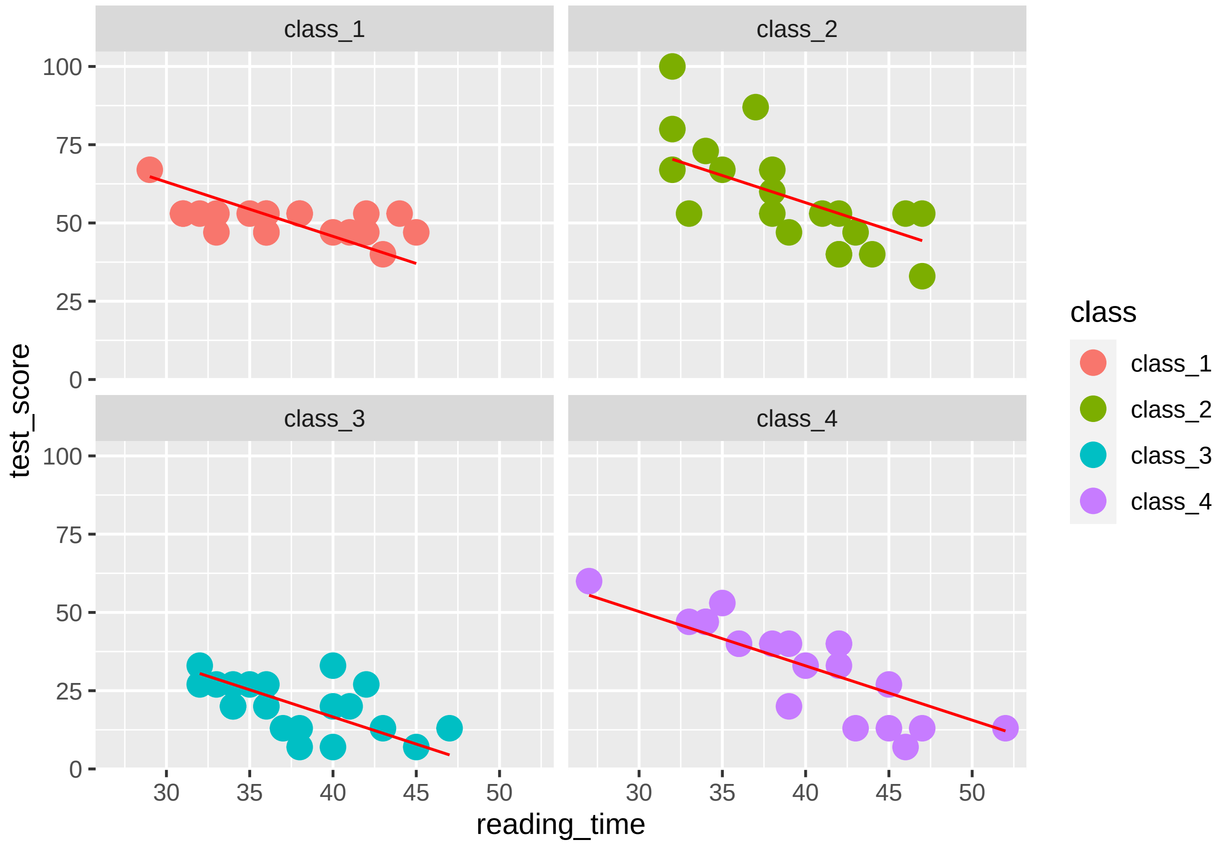
Visualization plays an important role in assessing the model fit of mixed effect models and potentially guiding decisions about whether to use random slope or random intercept models.

We can plot the regression lines of the two models by saving the predictions of the models into a variable.

```
data_test_performance_slope = data_test_performance_slope %>%
    mutate(pred_int = predict(mod_rnd_int), pred_slope = predict(mod_rnd_slope_opt))
```

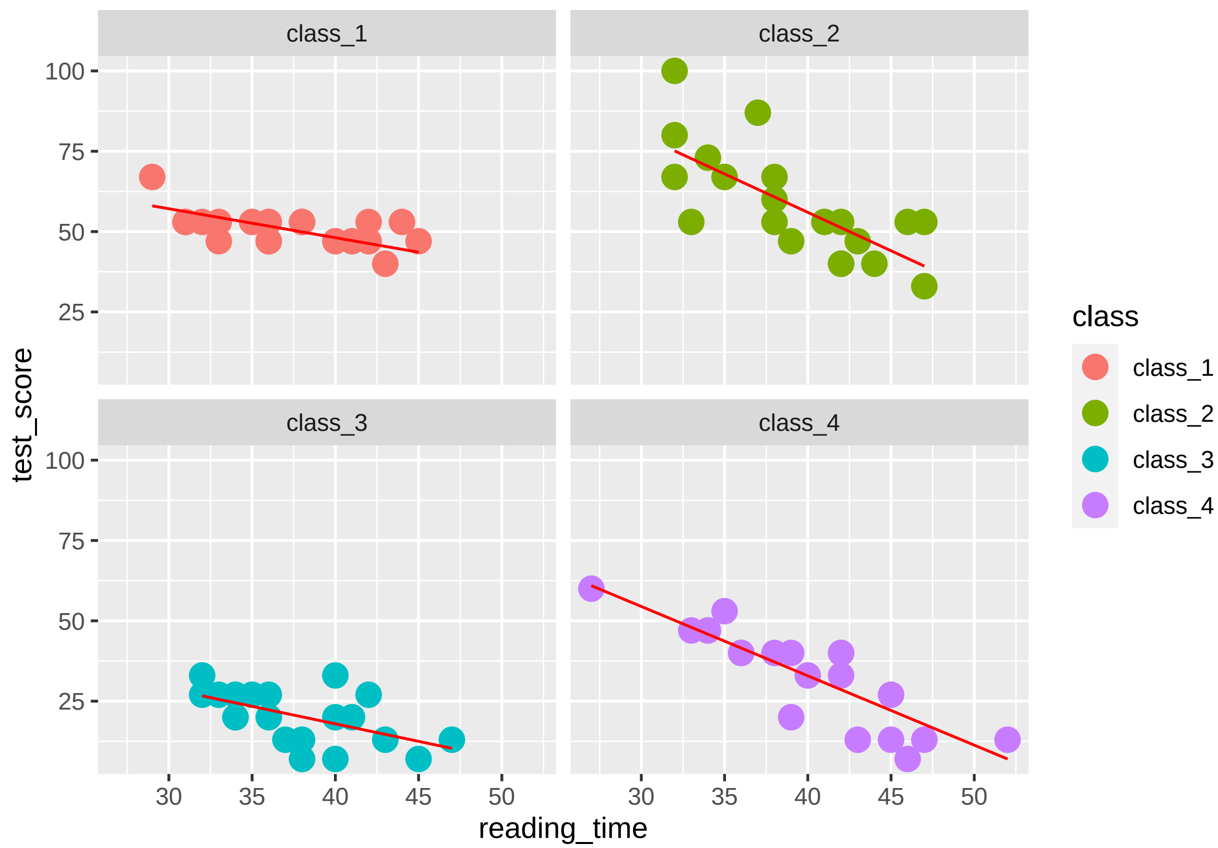
Regression line of the random intercept model

```
data_test_performance_slope %>%
    ggplot() + aes(y = test_score, x = reading_time, group = class) +
    geom_point(aes(color = class), size = 4) + geom_line(color = "red",
    aes(y = pred_int, x = reading_time)) + facet_wrap(~class,
    ncol = 2)
```



Regression line of the random slope model

```
data_test_performance_slope %>%
    ggplot() + aes(y = test_score, x = reading_time, group = class) +
    geom_point(aes(color = class), size = 4) + geom_line(color = "red",
    aes(y = pred_slope, x = reading_time)) + facet_wrap(~class,
    ncol = 2)
```



When comparing the plots, we can see a slight improvement in the model fit in the random slope model compared to the random intercept model, but the improvement is not too impressive.

3.7.2 Comparing model fit indices

If we compare the prediction error of the 3 models, we will see that the RSS is largest for the fixed effect model and the smallest for the random slope model.

```
sum(residuals(mod_fixed)^2)
```

```
## [1] 24310.52
```

```
sum(residuals(mod_rnd_int)^2)
```

```
## [1] 6148.282
```

```
sum(residuals(mod_rnd_slope_opt)^2)
```

```
## [1] 5045.968
```

But this is no surprise, since we allow for increasingly more flexibility in the model to fit our data in the random intercept, and the random slope models. So relying on RSS alone in our original dataset when comparing prediction efficiency would be misleading.

Instead, we can use model fit indices that are sensitive to the number of predictors, such as AIC.

Note that in the case of the random effect models, it is more appropriate to look at the conditional AIC (cAIC) than the simple AIC. We can get cAIC by using the cAIC() function from the cAIC4 package.

The cAIC of the random slope model is smaller by more than 2, indicating that the random slope model seems to be a slightly better fit.

```
cAIC(mod_rnd_int)$caic
```

```
## [1] 586.5155
```

```
cAIC(mod_rnd_slope_opt)$caic
```

```
## [1] 576.9777
```

You can also use the **Likelihood ratio test** to compare different mixed models. As we have learned with hiearchical regression, this test can only be used to compare **nested models**, where the fixed effect predictors in the simpler model are a subset of the fixed effect predictors of the more complex model.

(When you use the anova() function on mixed models, you will get the warning message: 'refitting model(s) with ML (instead of REML)'. This is normal and can be safely ignored. lmer() uses restricted maximum likelyhood estimator (REML) for the variance by default instead of the simple maximum likelyhood estimator (ML). But the anova() function can only compare models using the ML, so the models are refit with this specification.)

```
anova(mod_rnd_int, mod_rnd_slope_opt)
```

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

```
## Data: data_test_performance_slope
## Models:
## mod_rnd_int: test_score ~ reading_time + (1 | class)
## mod_rnd_slope_opt: test_score ~ reading_time + (reading_time | class)
##                   npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## mod_rnd_int          4 602.56 612.09 -297.28   594.56
## mod_rnd_slope_opt    6 600.24 614.53 -294.12   588.24 6.3207  2    0.04241 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.8 What to report

We have to report the same information about linear mixed models as for fixed-effect-only linear models seen in the previous exercises.

3.8.1 Describe the methods

You could describe the following in your methods section about the model formulation:

"In order to determine the influence of reading time on test performance, we used a linear mixed model. In the model the outcome variable was the test score, and we used a single fixed effect predictor, reading_time. Because data was clustered in school classes, we included the random effect of class in the model. No prior data or theory was available about how the random effect of class might manifest, so we built two separate models. In one model we only allowed for a random intercept of classes, while in the other model we allowed for both random intercept and the random slope of the effect of reading_time across different classes. As pre-registered in our experimental protocol, we compared the model fit of the random intercept and slope models using the anova() and cAIC() functions, and we made a choice between these two models based on cAIC."

You would use the follwing functions to get the reportable results:

cAIC:

```
cAIC(mod_rnd_int)$caic
```

```
## [1] 586.5155
```

```r
cAIC(mod_rnd_slope_opt)$caic
```

```
## [1] 576.9777
```

anova:

```r
anova(mod_rnd_int, mod_rnd_slope_opt)
```

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

```
## Data: data_test_performance_slope
## Models:
## mod_rnd_int: test_score ~ reading_time + (1 | class)
## mod_rnd_slope_opt: test_score ~ reading_time + (reading_time | class)
##                   npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## mod_rnd_int          4 602.56 612.09 -297.28   594.56
## mod_rnd_slope_opt    6 600.24 614.53 -294.12   588.24 6.3207  2    0.04241 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.8.2 Total model statistics: singificance and variance explained

The r2beta() function computes the marginal R squared based on Nakagawa, Johnson & Schielzeth (2017). This is a special type of the R squared statistic that shows the proportion of variance explained by the fixed factor(s) alone, when not taking into account the random effect terms. There is no classical F test p-value attached to this statistic, but significance can be interpreted from the confidence intervals. If the 95% CI does not contain 0, it means that the fixed effect term(s) explain a significant portion of the variation of the outcome compared to the mean (the null model).

Additionally, a point estimate of both the marginal and the conditional R squared value can be obtained via the r.squaredGLMM() function from the MuMIn package. This function also uses the formula published by Nakagawa, Johnson & Schielzeth (2017).

Reference:

Nakagawa, S., Johnson, P.C.D., Schielzeth, H. (2017) The coefficient of determination R2 and intraclass correlation coefficient from generalized linear mixed-effects models revisited and expanded. J. R. Soc. Interface 14: 20170213.

```r
# marginal R squared with confidence intervals
r2beta(mod_rnd_slope_opt, method = "nsj", data = data_test_performance_slope)
```

```
##         Effect   Rsq upper.CL lower.CL
## 1        Model 0.148    0.304    0.036
## 2 reading_time 0.148    0.304    0.036
```

```r
# marginal and conditional R squared values
r.squaredGLMM(mod_rnd_slope_opt)
```

```
## Warning: 'r.squaredGLMM' now calculates a revised statistic. See the help page.
```

```
##            R2m       R2c
## [1,] 0.1477794 0.8484016
```

In the results section, you should report the linear mixed models analysis with the following data:

"The random slope model produced a better model fit both according to the likelihood ratio test (Chi^2 = 6.3207, df = 2, p = 0.042) and the cAIC (cAIC intercept = 586.51, cAIC slope = 576.98). Thus, we present the results of the random slope model in the following. (The results of the random intercept model are listed in the supplement.)

The linear mixed model was significantly better than the null model, where the fixed effect predictor, reading_time, explained14.78% of the variance of test_score (R^2 = 0.15 [95% CI = 0.036, 0.304]).”

You will also have to report statistics related to the predictors, this is usually done in a table format, because most often we have multiple predictors (even though in this example we only have one). You can get the information about the important results related to the predictors from the following functions:

Model coefficients and p-values:

The final table would look something like this:

```
##                   b 95%CI lb 95%CI ub Std.Beta p-value
## (Intercept)  104.11    54.99   152.92        0    .019
## reading_time  -1.63    -2.63    -0.62    -0.42    .037
```

Specific parts of this table can be extracted using the following functions:

Model coefficients and p-values: (Note that the summary() function will only provide p-values if you also use the lmerTest package)

```
summary(mod_rnd_slope_opt)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: test_score ~ reading_time + (reading_time | class)
##    Data: data_test_performance_slope
## Control: lmerControl(optimizer = "Nelder_Mead")
##
## REML criterion at convergence: 582.2
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.36583 -0.56680 -0.00496  0.45815  2.98763
##
## Random effects:
##  Groups   Name         Variance  Std.Dev. Corr
##  class    (Intercept)  1744.6902 41.7695
##           reading_time    0.6792  0.8241  -0.92
##  Residual                69.4379  8.3329
## Number of obs: 80, groups:  class, 4
##
## Fixed effects:
##              Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)  104.1055    22.1889   2.9495   4.692   0.0190 *
## reading_time  -1.6345     0.4555   2.9855  -3.588   0.0374 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## reading_tim -0.929
```

Confidence intervals for the model coefficients:

(this can take a few minutes depending on your processor speed, requires a lot of computing power)

```
confint(mod_rnd_slope_opt)
```

```
##                   2.5 %      97.5 %
```

```
## .sig01       15.3522423  93.4648471
## .sig02       -0.9956117  -0.3008542
## .sig03        0.2085520   1.8968480
## .sigma        7.1373991   9.9049762
## (Intercept)  54.9914780 152.9242130
## reading_time -2.6264543  -0.6211491
```

Standardized beta for each predictor:

```
stdCoef.merMod(mod_rnd_slope_opt)
```

```
##                 stdcoef     stdse
## (Intercept)   0.0000000 0.0000000
## reading_time -0.4234211 0.1180017
```

Note that the use and interpretation of p-values for linear mixed models is controversial at the moment, so observe the trend in your particular sub-field and decide whether you want to use them or not. confidence intervals give you information about statistical significance, so it is not necessary to provide p-values.