Exercise 17 - Basics of linear mixed models

Zoltan Kekecs

24 november 2019

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 of the results of mixed models.

The latest version of this document and the code the document refers to can be found in the GitHub repository of the class at: https://github.com/kekecsz/PSYP13_Data_analysis_class-2019

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 bullying data

In this exercise we will work with simulated data about bullying. Let's say that we are interested in predicting how much body size can predict vulnerability to bullying among 4th grade primary school children. Vulnerability to bullying in this study is quantified by the number of lunch sandwitches taken from the child during the period of one month based on self report. The predictor of interest in this study is weight. The researchers hypothesize that the smaller the child, the more sandwitches will be taken from him or her.

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

```
# load data
data_bully_int = read_csv("https://raw.githubusercontent.com/kekecsz/PSYP13_Data_analysis_class-2018/ma
```

```
## Parsed with column specification:
## cols(
##   sandwich_taken = col_double(),
##   weight = col_double(),
##   class = col_character()
## )
```

```r
# asign class as a grouping factor
data_bully_int %>% mutate(class = factor(class))
```

```
## # A tibble: 80 x 3
##    sandwich_taken weight class
##             <dbl>  <dbl> <fct>
##  1              8     30 class_1
##  2             10     28 class_1
##  3             11     27 class_1
##  4             12     33 class_1
##  5              7     36 class_1
##  6             10     30 class_1
##  7              9     30 class_1
##  8             13     23 class_1
##  9              8     33 class_1
## 10              7     36 class_1
## # ... with 70 more rows
```

```r
data_bully_slope = read_csv("https://raw.githubusercontent.com/kekecsz/PSYP13_Data_analysis_class-2018/
```

```
## Parsed with column specification:
## cols(
##   sandwich_taken = col_double(),
##   weight = col_double(),
##   class = col_character()
## )
```

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

```
## # A tibble: 80 x 3
##    sandwich_taken weight class
##             <dbl>  <dbl> <fct>
##  1              8     44 class_1
##  2              8     38 class_1
##  3              7     40 class_1
##  4              8     42 class_1
##  5              8     36 class_1
##  6             10     29 class_1
##  7              6     43 class_1
##  8              8     35 class_1
##  9              8     31 class_1
## 10              7     36 class_1
## # ... with 70 more rows
```

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

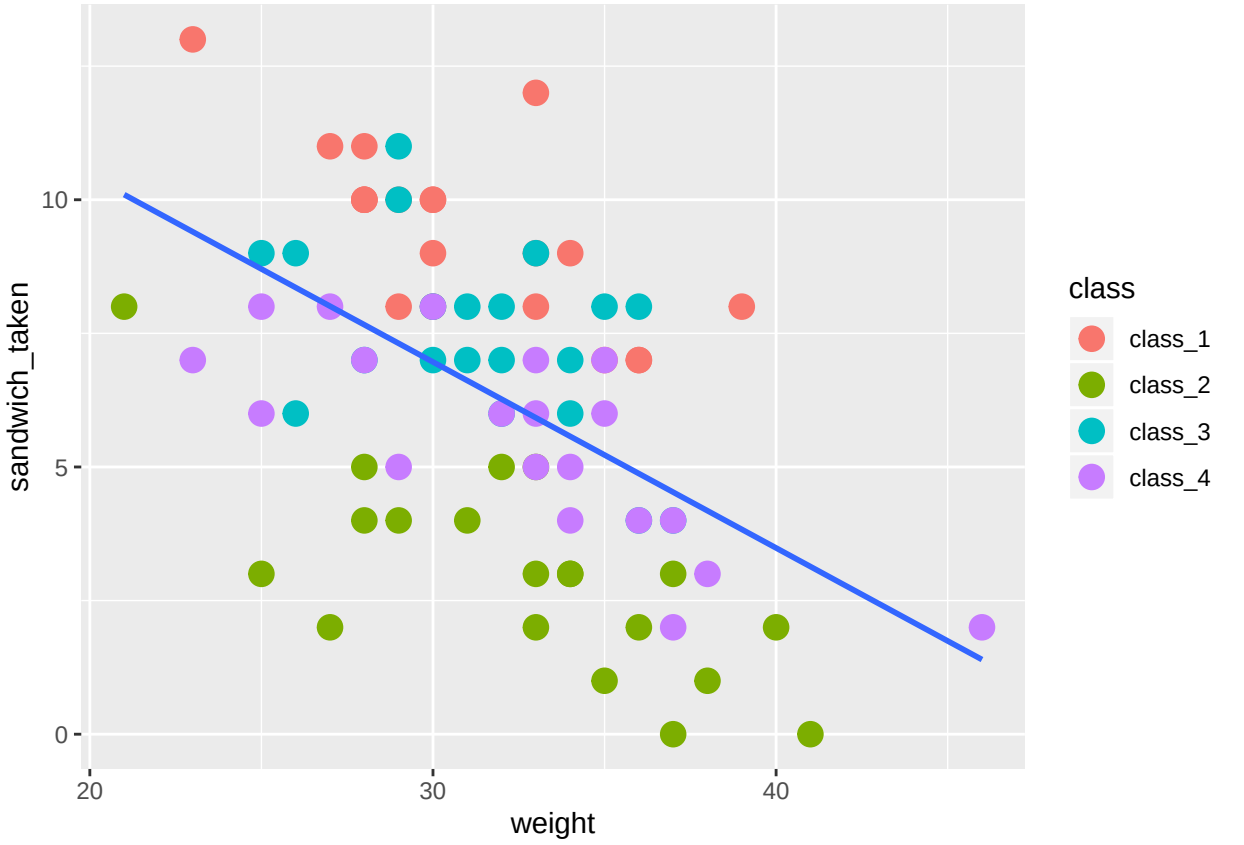
3.1 exploring clustering in the data

Let's plot the relationship for the simple regression model of sandwitch_taken ~ weight on a scatterplot. It seems that there is a clear negative relationship if weight and the number of sandwitches taken from the

children, howeverm 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.

```
data_bully_int %>% ggplot() + aes(y = sandwich_taken, x = weight) +
    geom_point(aes(color = class), size = 4) + geom_smooth(method = "lm",
    se = F)
```

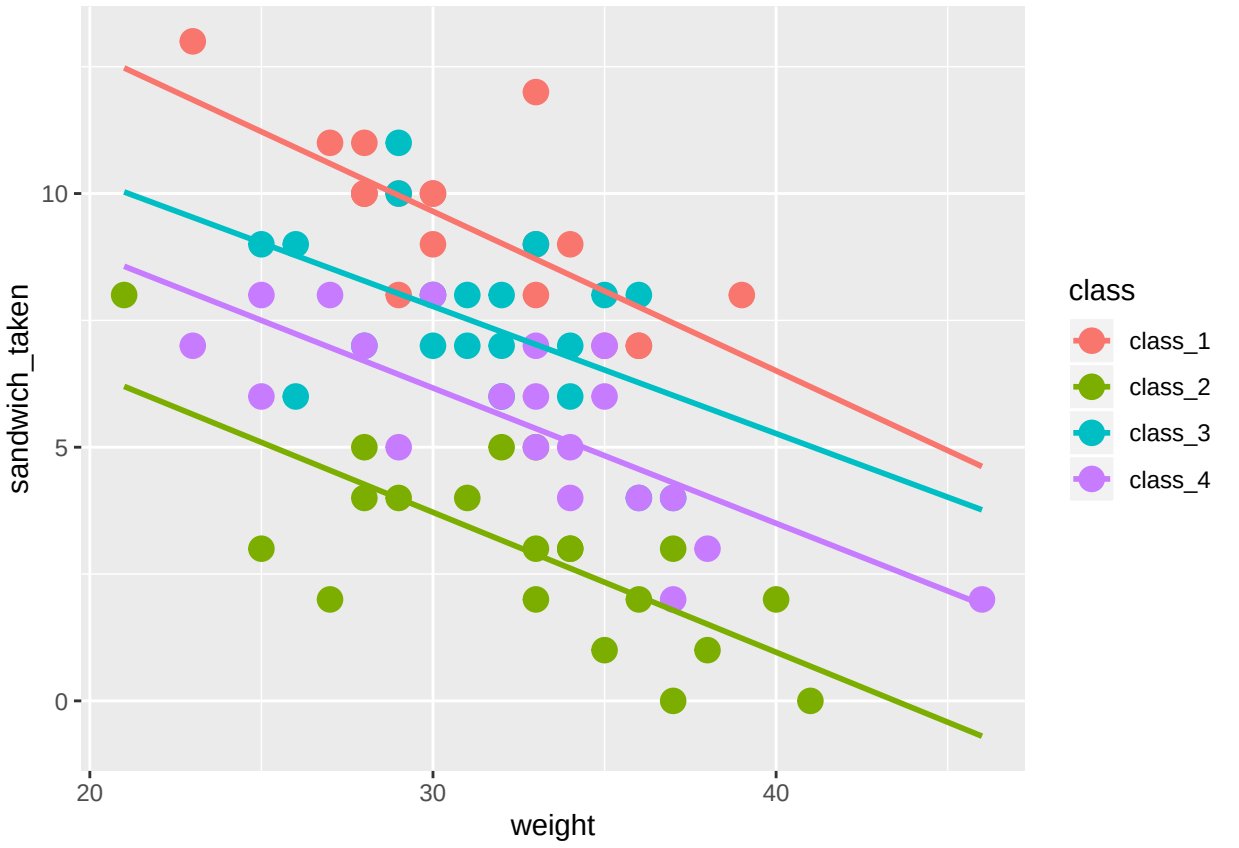


Let's see if class membership can explain some of this variability. We can plot the regression lines for each class separately. This seems to be able to explain some of the variability in the data, bringing the regression lines closer to the actual observations. So it seems that it would be worthwile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

(notice that we save the plot into an object called int_plot, so next time we can call the same plot just by referring to this object)

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

int_plot
```



3.2 Mixed models

Right now we are only interested in whether or not weight influences bullying vulnerability, and if so, to what extent. Class membership is secondary to our interests, and even if we would be able to establish the particular effect of any of the classes in this school, in other schools this information would be useless, since there would be different classes.

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 the random effect of class.

In this context, we call the effect of predictors on the outcome 'fixed effects'. The effect of clustering variables, for which we have too many levels to be meaningful to enter as predictors, 'random effects'.

Models that can take into account both fixed and random effects are called mixed effect models.

3.3 Two types of random effects

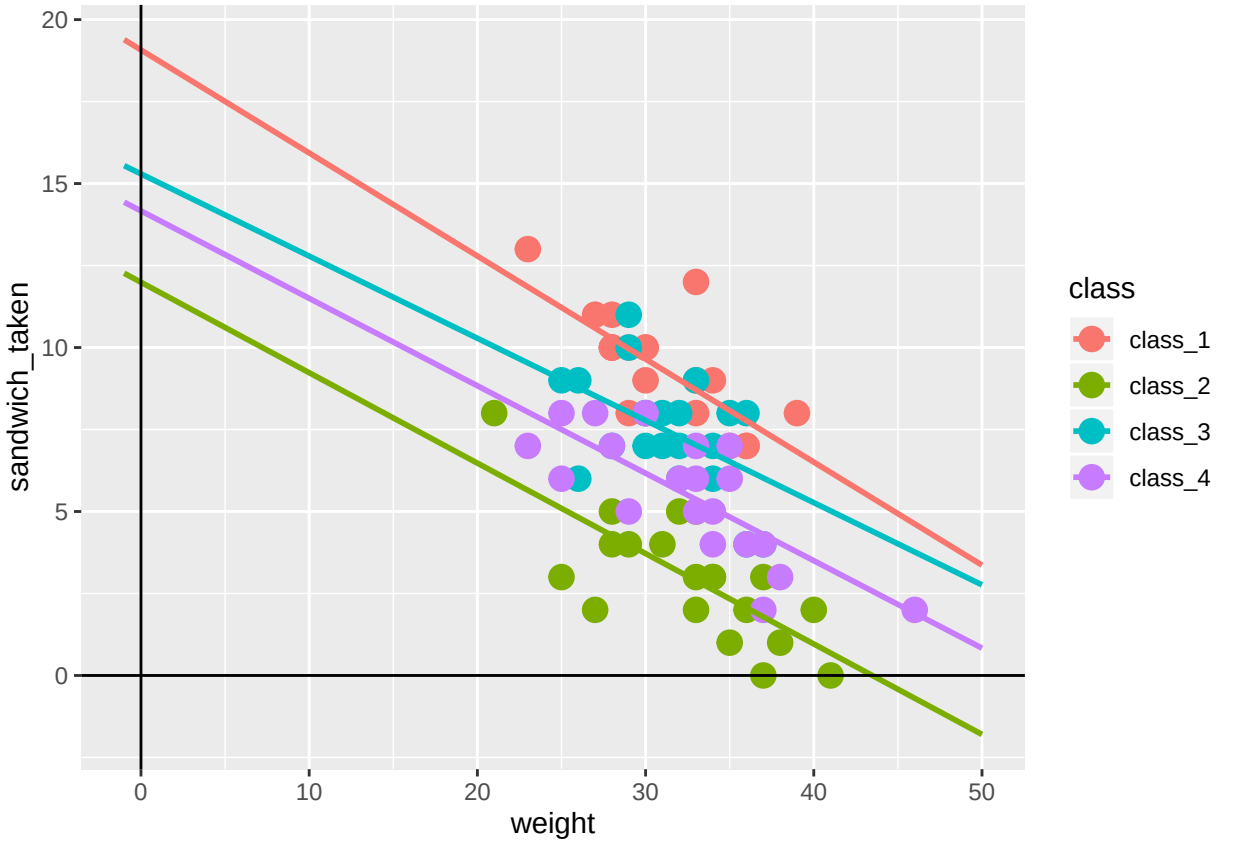
There is generally two ways in which clustering (or the random effect term) can influence the outcome variable:

random intercept, but no random slope: it is possible that clusters 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_bully_int. You can observe that weight affects number of sanwiches taken by rougly the same amount in all classes, but some classes just have fewer sandwiches taken from them in general than others.

On the scatterplot this manifests as the regression lines crossing the y axis at different places (different

intercepts), but being almost parallel to each other (similar slopes), indicating that the effect of weight is similar in the classes.

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



random intercept, AND random slope: Now let's look at the other dataset we loaded in the beggining of the exercise, saved in the data_bully_slope object. This dataset is also simulated, and it comes from the same scenario as the data_bully_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 stolen sandwitches is different across classes, but the effect of weight also seems to be different in the different classes.

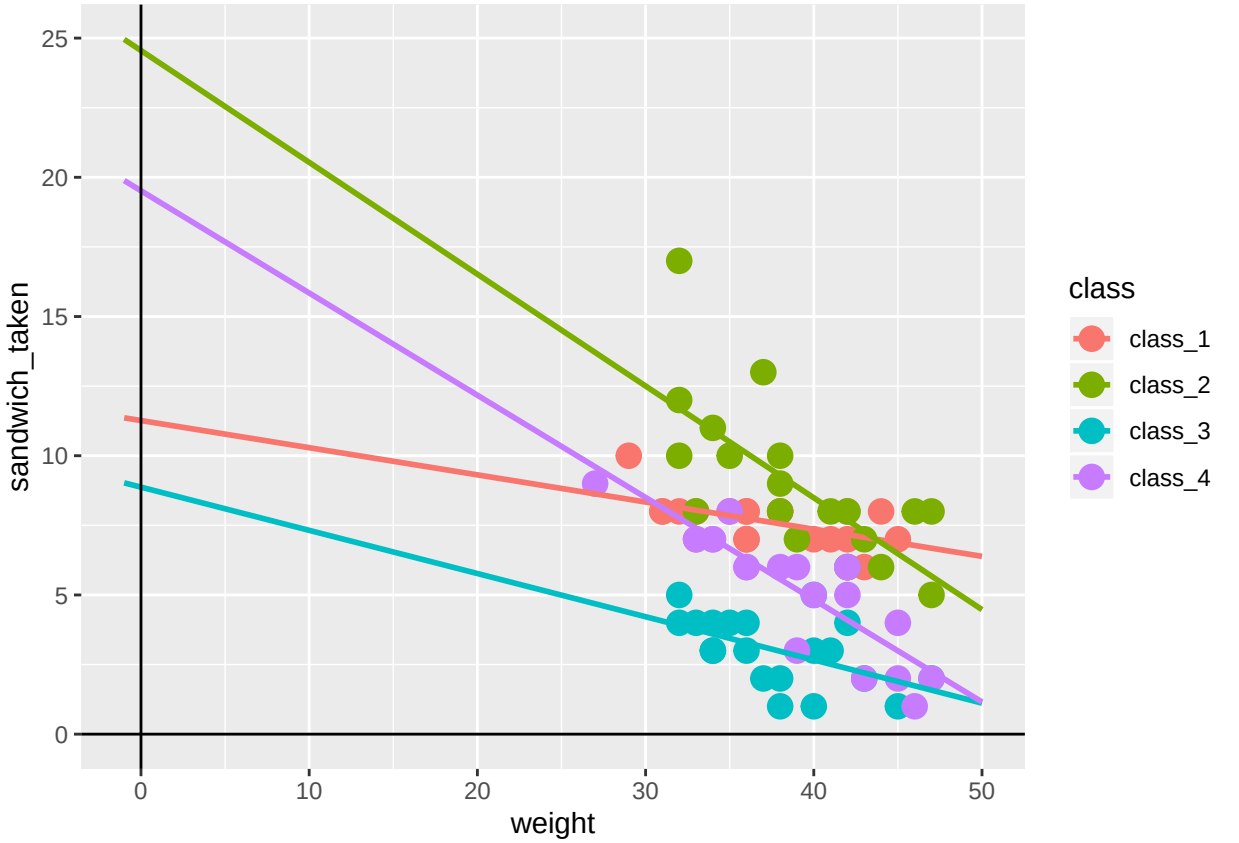
For example in Class 1 (the red dots), weigth seems to almost make no difference, everyone seems to lose roughly the same number of sanwiches each month regardless of weight. On the other hand in classes 2 and 4, smaller children seem to lose much more sandwitches than their heavy weight classmates.

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.

```
slope_plot = data_bully_slope %>% ggplot() + aes(y = sandwich_taken,
    x = weight, 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
```

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

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



Let's see how we can use this knowledge about clustering in the dataset to improve our prediction model. We will use the data_bully_slope dataset here.

Now we are going to fit three different models, a simple fixed effect model with no random effects, 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_bully_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, weight, 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(sandwich_taken ~ weight, data = data_bully_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) would have 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_bully_int dataset. But here we fit this to the data_bully_slope dataset, so we can compare the effectiveness of random slope and random intercept models.

```
mod_rnd_int = lmer(sandwich_taken ~ weight + (1 | class), data = data_bully_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 '+ (weight|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 predictors.

```
mod_rnd_slope = lmer(sandwich_taken ~ weight + (weight | class),
    data = data_bully_slope)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00317298 (tol = 0.002, component 1)
```

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] 581.6364
```

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

```
## [1] 159.5818
```

```
sum(residuals(mod_rnd_slope)^2)
```

```
## [1] 132.2322
```

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.

```
AIC(mod_fixed)
```

```
## [1] 391.7357
```

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

```
## [1] 294.4023
```

```
cAIC(mod_rnd_slope)$caic
```

```
## [1] 285.6445
```

3.4 Deciding whether to use random intercept or slope model

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 weight as well or not, or whether previous exploratory studies have shown that modeling a random slope roduces much better model fit than a simple random intercept model.

In some case however, we are exploring the topic without strong prior research or theoretical cues. In these cases it may be apropriate 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.

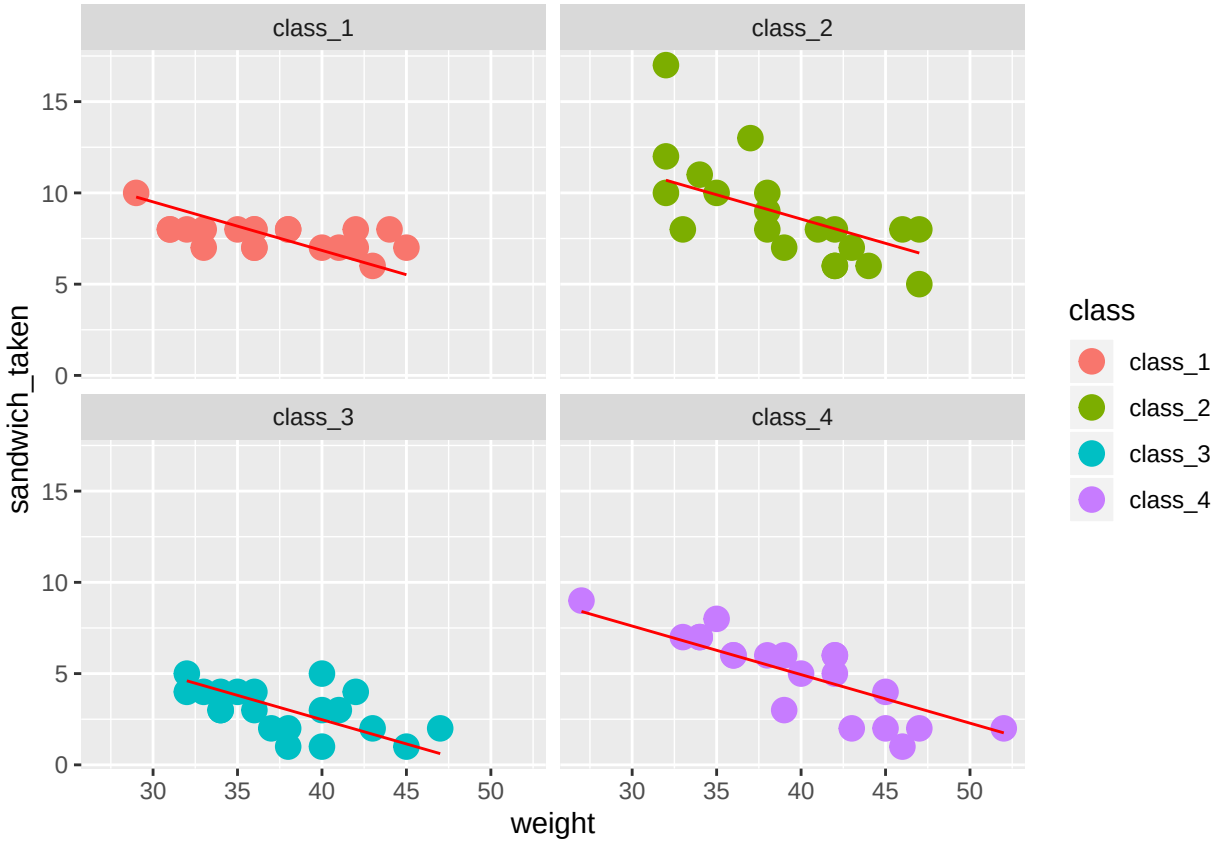
Visualization plays an important role in assessing the model fit of mixed effect models and potentially guiding decisions about whether to use random slope or random intercept models.

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

```
data_bully_slope = data_bully_slope %>% mutate(pred_int = predict(mod_rnd_int),
    pred_slope = predict(mod_rnd_slope))
```

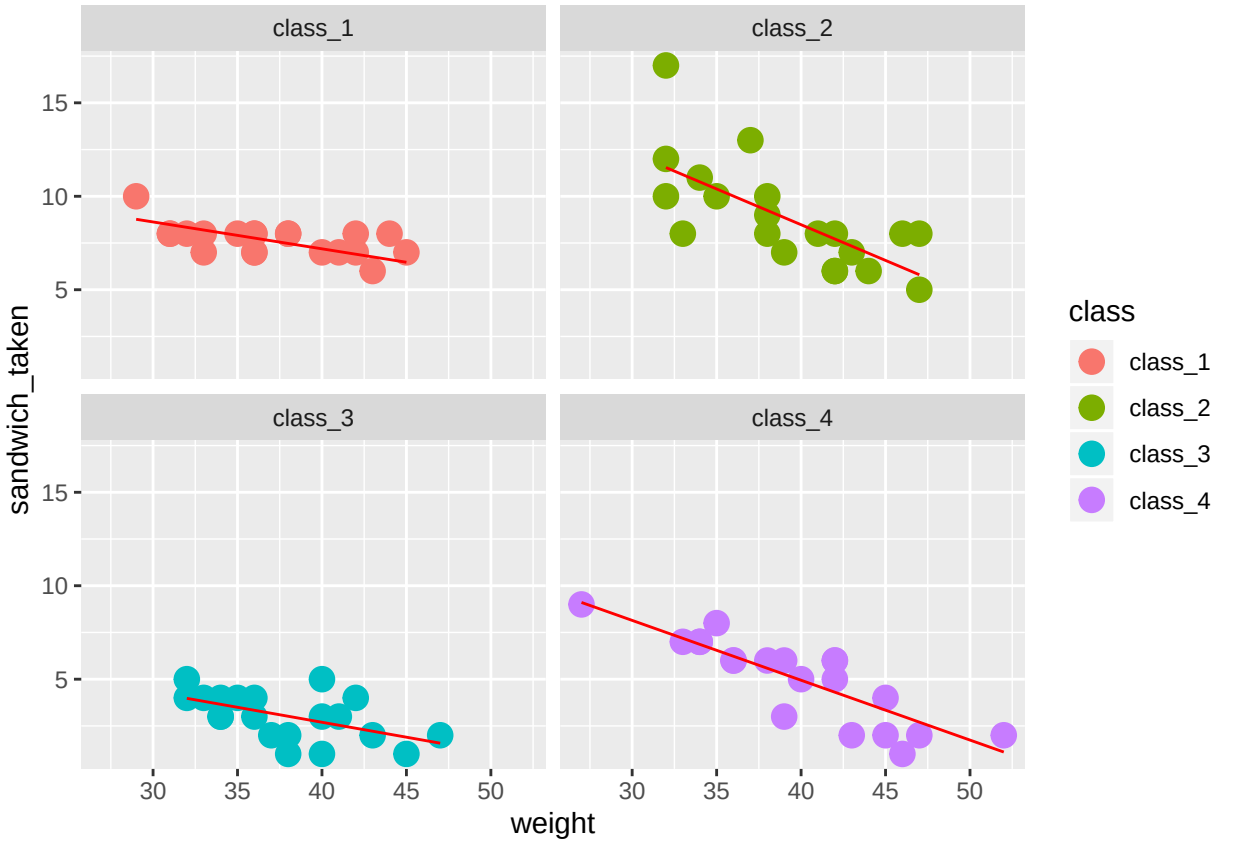
Regression line of the random intercept model

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



Regression line of the random slope model

```
data_bully_slope %>% ggplot() + aes(y = sandwich_taken, x = weight,
    group = class) + geom_point(aes(color = class), size = 4) +
    geom_line(color = "red", aes(y = pred_slope, x = weight)) +
    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.

Besides visual exploration of the clustering in the data, you could compare cAIC of the models with different random effect terms as seen before, or you can use a significance test of whether the error in the two models are significantly different via the anova() function.

The cAIC of the random slope model is smaller by more than 2, and the anova() also suggests that the two models are significantly different, indicating that the random slope model seems to be a slightly better fit.

(When you use the anova() function on mixed models, you can 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.)

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

```
## [1] 294.4023
```

```
cAIC(mod_rnd_slope)$caic
```

```
## [1] 285.6445
```

```
anova(mod_rnd_int, mod_rnd_slope)
```

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

```
## Data: data_bully_slope
## Models:
```

```
## mod_rnd_int: sandwich_taken ~ weight + (1 | class)
## mod_rnd_slope: sandwich_taken ~ weight + (weight | class)
##               Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## mod_rnd_int    4 310.00 319.53 -151.00   302.00
## mod_rnd_slope  6 307.94 322.23 -147.97   295.94 6.0595      2    0.04833 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.5 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.

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

"In order to determine the influence of weight on vulnerability to bullying, we used a linear mixed model. In the model the outcome variable was the number of sandwitches taken, and we used a single fixed effect predictor, weight. 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 weight 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 ue the follwing functions to get the reportable results:

cAIC:

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

```
## [1] 294.4023
```

```
cAIC(mod_rnd_slope)$caic
```

```
## [1] 285.6445
```

anova:

```
anova(mod_rnd_int, mod_rnd_slope)
```

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

```
## Data: data_bully_slope
## Models:
## mod_rnd_int: sandwich_taken ~ weight + (1 | class)
## mod_rnd_slope: sandwich_taken ~ weight + (weight | class)
##               Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## mod_rnd_int    4 310.00 319.53 -151.00   302.00
## mod_rnd_slope  6 307.94 322.23 -147.97   295.94 6.0595      2    0.04833 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model R squared and confidence interval:

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.

```
# marginal R squared with confidence intervals
r2beta(mod_rnd_slope, method = "nsj", data = data_bully_slope)
```

```
##   Effect   Rsq upper.CL lower.CL
## 1  Model 0.147    0.304    0.036
## 2 weight 0.147    0.304    0.036
```

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

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

```
##           R2m       R2c
## [1,] 0.147134 0.8331627
```

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 likelyhood ratio test ($\chi^2 = 6.06$, df = 2, p = .048) and the cAIC (cAIC intercept = 294.4, cAIC slope = 285.64). 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, weight, explained14.71% of the variance of sandwitches taken (R^2 = 0.15 [95% CI = 0.04, 0.3])."

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:

```
##                    2.5 %      97.5 %
## .sig01        2.43299289 14.98924444
## .sig02       -1.00000000 -0.36854937
## .sig03        0.02958071  0.30120346
## .sigma        1.15485234  1.60286560
## (Intercept)  8.01702852 23.72159964
## weight       -0.40919185 -0.09064192
```

```
##                 b 95%CI lb 95%CI ub Std.Beta p-value
## (Intercept) 15.89     8.02    23.72        0    .021
## weight      -0.25    -0.41    -0.09    -0.42     .04
```

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)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: sandwich_taken ~ weight + (weight | class)
##    Data: data_bully_slope
##
## REML criterion at convergence: 297.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.3403 -0.5630 -0.0076  0.3896  4.0511
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  class    (Intercept) 44.61736 6.6796
##           weight       0.01693 0.1301   -0.94
##  Residual              1.81799 1.3483
## Number of obs: 80, groups:  class, 4
##
## Fixed effects:
##             Estimate Std. Error       df t value Pr(>|t|)
## (Intercept) 15.88887    3.55290  2.95835   4.472   0.0215 *
## weight      -0.25155    0.07224  2.98048  -3.482   0.0404 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##        (Intr)
## weight -0.940
## convergence code: 0
## Model failed to converge with max|grad| = 0.00317298 (tol = 0.002, component 1)
```

Cofidence 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)
```

```
##                    2.5 %      97.5 %
## .sig01       2.43299289 14.98924444
## .sig02      -1.00000000 -0.36854937
## .sig03       0.02958071  0.30120346
## .sigma       1.15485234  1.60286560
## (Intercept)  8.01702852 23.72159964
## weight      -0.40919185 -0.09064192
```

Standardized beta for each predictor:

```
stdCoef.merMod(mod_rnd_slope)
```

```
##               stdcoef     stdse
## (Intercept)  0.000000 0.0000000
## weight      -0.422144 0.1212255
```

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.