

Mixed Models Part 1

Dr Andrew J. Stewart

E: drandrewjstewart@gmail.com

T: [@ajstewart_lang](#)

G: [ajstewartlang](#)



In this workshop...

- We will take our first look at (generalised) linear mixed models - (G)LMMs.
- (G)LMMs allow for models with a combination of fixed and random effects (intercepts and slopes).
- We'll focus on designs with one factor of several levels, and 2 x 2 designs for continuous data.
- We'll also examine measures of model fit, and using {emmeans} to interpret interactions in factorial designs.

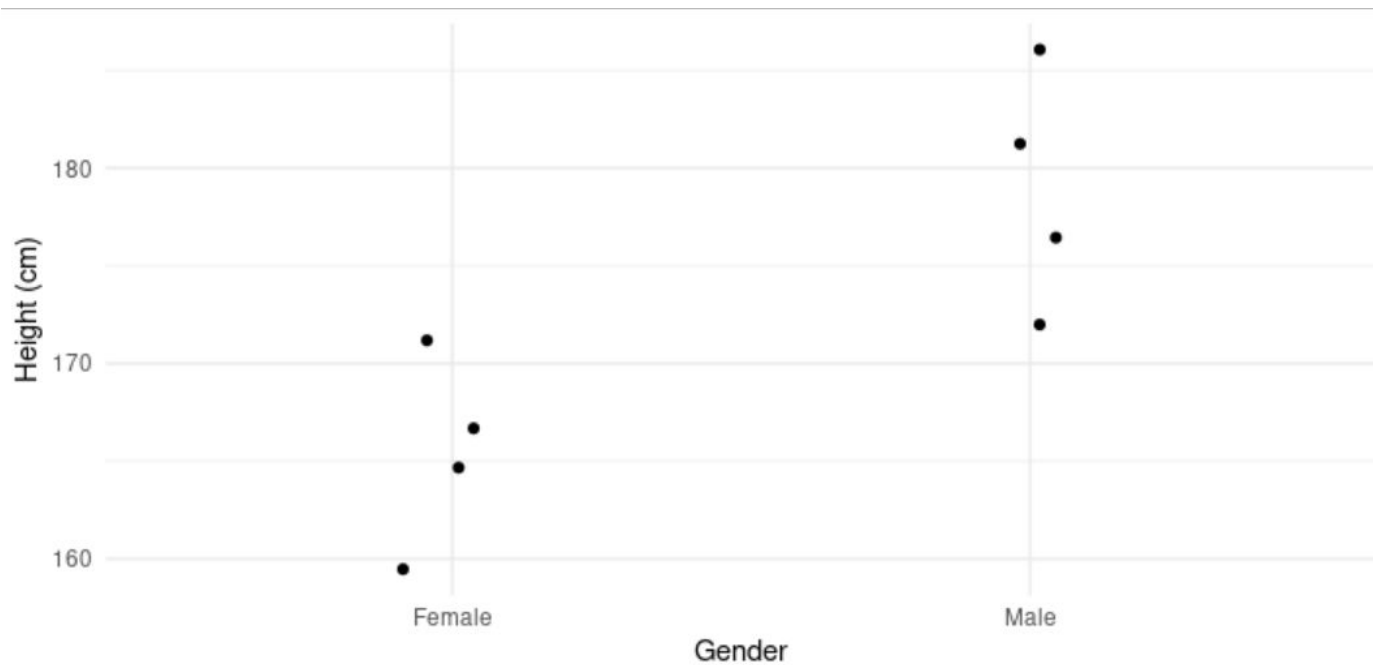
Why linear mixed models?

(G)LMMs are more flexible than ANOVA, allow for multiple simultaneous random effects (e.g., subjects and items), subject and item covariates, nesting, unbalanced designs, cope with missing data, allow you to model both continuous and categorical IVs and DVs, operate over trial-level data, provide a good balance of Type I and Type II error, and allow you to determine the best statistical models to fit to your data that make the most theoretical sense...

Recap – linear modelling

Here we have a measure of height for 4 males and 4 females. Can gender be used to predict height?

```
> gender_height_data
# A tibble: 8 x 3
  subject gender height
  <int>   <fct>   <dbl>
1       1   male    170
2       2   male    180
3       3   male    175
4       4   male    185
5       5 female    160
6       6 female    170
7       7 female    165
8       8 female    165
```



It certainly looks like Males (on average) are taller than Females (on average).

Let's fit a linear model using the `lm()` function.

```
height_model <- lm(height ~ gender, data = gender_height_data)
```

```
> summary(height_model)
```

Call:

```
lm(formula = height ~ gender, data =  
gender_height_data)
```

Residuals:

Min	1Q	Median	3Q	Max
-7.500	-3.125	0.000	3.125	7.500

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	165.000	2.700	61.104	1.29e-09 ***
gendermale	12.500	3.819	3.273	0.017 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.'
0.1 ' ' 1

Residual standard error: 5.401 on 6 degrees of freedom

Multiple R-squared: 0.641, Adjusted R-squared:
0.5812

F-statistic: 10.71 on 1 and 6 DF, p-value: 0.01696

We can see here that
Gender is a significant
predictor ($p = 0.017$)

```
> summary(height_model)
```

```
Call:
```

```
lm(formula = height ~ gender, data =  
gender_height_data)
```

```
Residuals:
```

```
      Min       1Q   Median       3Q      Max  
-7.500 -3.125  0.000   3.125   7.500
```

```
Coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)  165.000      2.700   61.104 1.29e-09 ***  
gendermale    12.500      3.819    3.273  0.017 *
```

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  
0.1 ' ' 1
```

```
Residual standard error: 5.401 on 6 degrees of freedom
```

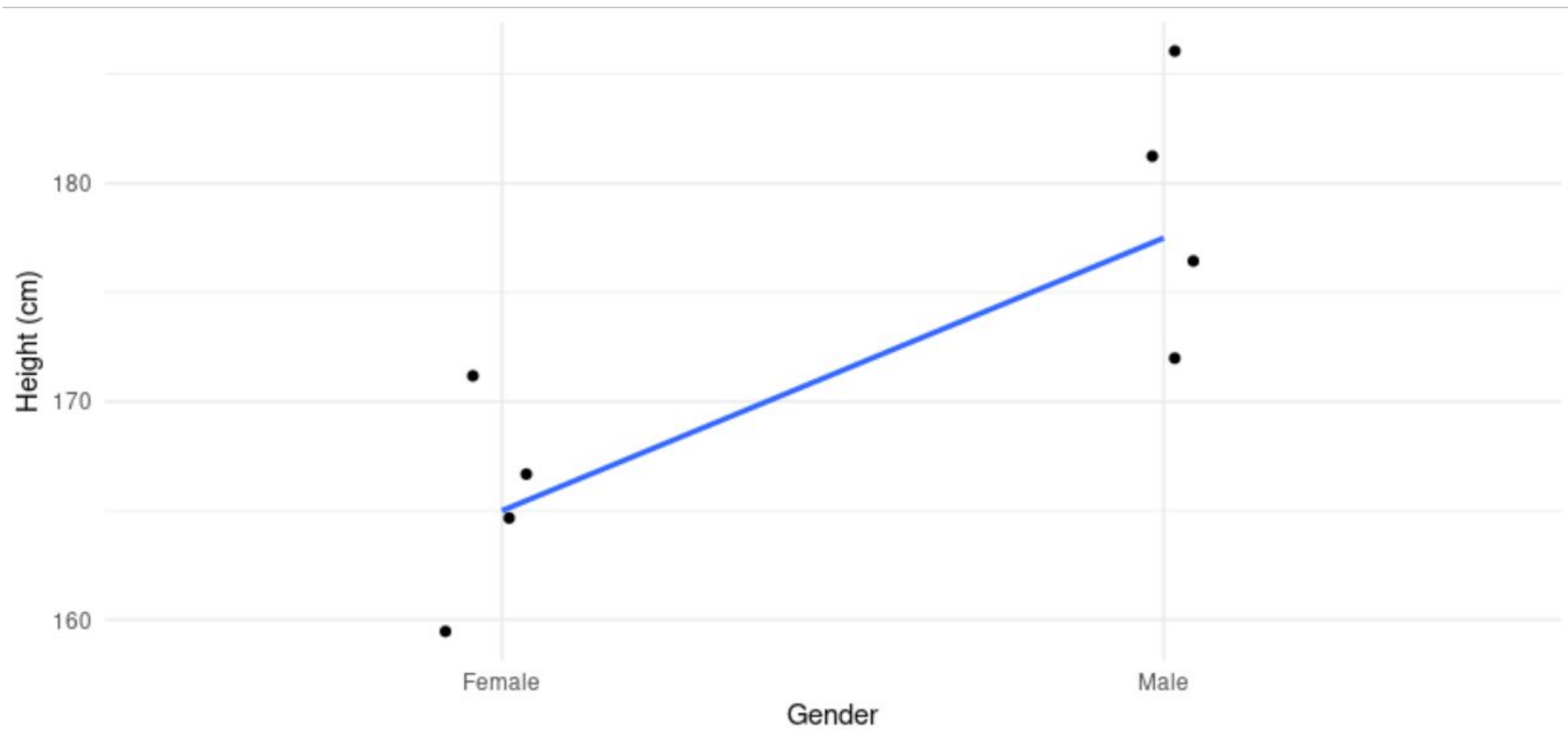
```
Multiple R-squared:  0.641,    Adjusted R-squared:  
0.5812
```

```
F-statistic: 10.71 on 1 and 6 DF,  p-value: 0.01696
```

The Intercept (165)
corresponds to the mean
height of our reference
category (Female).

The `gendermale`
coefficient (12.5) is the
difference between our
reference category
(Intercept) and our Males.

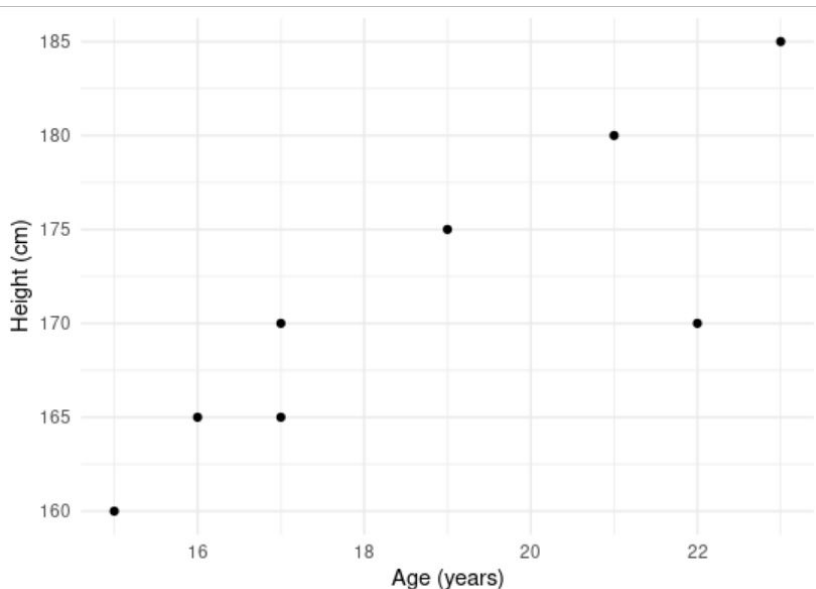
Females were taken as the
reference category (i.e.,
the intercept) simply
because R chooses this
on an alphabetical basis.



How about with a non-categorical predictor?

Here we have age and height data - can age predict someone's height in our dataset?

```
> age_height_data
# A tibble: 8 x 3
  subject age height
  <int> <dbl> <dbl>
1       1  22  170
2       2  21  180
3       3  19  175
4       4  23  185
5       5  15  160
6       6  17  170
7       7  16  165
8       8  17  165
```



Let's build a model...

```
> age_model <- lm(height ~ age, data = age_height_data)
> summary(age_model)
```

```
Call:
lm(formula = height ~ age, data = age_height_data)
```

```
Residuals:
```

```
      Min       1Q   Median       3Q      Max
-9.045  -2.104   1.646   3.201   3.557
```

```
Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	126.281	11.411	11.067	3.24e-05	***
age	2.398	0.602	3.984	0.00725	**

```
---
```

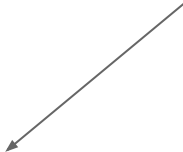
```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 4.721 on 6 degrees of freedom
```

```
Multiple R-squared:  0.7257,    Adjusted R-squared:  0.6799
```

```
F-statistic: 15.87 on 1 and 6 DF,  p-value: 0.007252
```

For every increase in Age by 1, Height increases by 2.398. But of course, we know this relationship likely breaks down at a certain age - but for the data we have, we can fit a linear function.



Linear Mixed Models in R

For mixed effects linear modelling in R, we need to install the package `{lme4}`. We also want the `{lmerTest}` package and the `{emmeans}` package.

```
> library(lme4)
> library(lmerTest)
> library(emmeans)
```

The `{lme4}` package is for model building, the `{lmerTest}` package is for p-value estimates of our model parameters, and the `{emmeans}` package will allow us to run follow-up tests on our models.

Linear mixed models

- What happens when we have many observations per person that we want to model?
- Imagine we are interested in how a person's reaction time varies whether they're responding to Large or Small target items.
- We observe the same 10 people each responding to 5 Large and 5 Small target items.
- We have 10 observations per person. These observations are not independent of each other as (which is an assumption of a linear model).

Linear mixed models

We can get around the lack of independence by treating participants as a *random effect* such that each participant has their own individual reaction time baseline.

This gives us a separate random intercept value for each participant - in other words, our model can account for individual variation.

This is a mixed effects model:

```
rt ~ condition + (1 | subject) + error
```

↑
This is our random effect term and models each subject having a different random intercept.

Linear mixed models


Imagine also that we have different Target Items (e.g., 10 different items that were presented in either in Large or Small format).

Each Target Item might have been a little different. One particular Target might just be responded to more quickly (regardless of what condition it was in) - in other words, the Target Items will also have different baselines.

Linear mixed models

We can capture the random effect of Item in the same way that we did for participants:

`rt ~ condition + (1 | subject) + (1 | item) + error`



We now have two random effects and are modelling each Subject **and** each Item having their own individual intercept.

Fixed vs. Random Effects

Fixed Effect

Data has been gathered from all the levels of the factor that are of interest. (Typically your experimental factors and maybe factors like age group - young vs. old for example).

Random Effect

The factor has many possible levels, interest is in all possible levels, but only a random sample of levels is included in the data. (Typically participants and items). Typically need > 5 levels in order to estimate effects.

Linear mixed models

```
> mixed_model_data
# A tibble: 400 x 4
  subject item condition  rt
  <fct>   <fct> <fct>   <int>
1 1      1     small    908
2 1      2     small    884
3 1      3     small    849
4 1      4     small    722
5 1      5     small   1090
6 2      1     small    890
7 2      2     small    703
8 2      3     small    781
9 2      4     small    942
10 2     5     small    898
# ... with 390 more rows
```

```
mixed_model_data %>%
  group_by(condition) %>%
  summarise(mean_rt = mean(rt), sd_rt
            = sd(rt))

# A tibble: 2 x 3
  condition mean_rt sd_rt
  <fct>      <dbl> <dbl>
1 large      854.   87.8
2 small      804.   97.5
```

```
mixed_model <- lmer(rt ~ condition + (1 | subject) + (1 | item),
                    data = mixed_model_data)
summary(mixed_model)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method ['lmerModLmerTest']
Formula: rt ~ condition + (1 | subject) + (1 | item)
Data: mixed_model_data
```

```
REML criterion at convergence: 4696.8
```

```
Scaled residuals:
```

```
      Min      1Q   Median      3Q      Max
-2.5385 -0.6366 -0.1475  0.6054  2.6043
```

```
Random effects:
```

Groups	Name	Variance	Std.Dev.
subject	(Intercept)	1240.3	35.22
item	(Intercept)	442.8	21.04
Residual		7126.7	84.42

```
Number of obs: 400, groups:  subject, 10; item, 5
```

```
Fixed effects:
```


	Estimate	Std. Error	df	t value	Pr(> t)
(Intercept)	854.140	15.755	12.166	54.213	6.99e-16 ***
conditionsmall	-49.780	8.442	385.000	-5.897	8.12e-09 ***

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

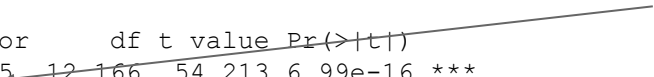
```
Correlation of Fixed Effects:
```

```
      (Intr)
conditnsml1 -0.268
```

More variability in subjects than in items.



The intercept corresponds to the RT to the Large Condition (854 ms) - going from Large to Small contexts decreases RT by around 50 ms.



Fixed effects:

	Estimate	Std. Error	df	t value	Pr(> t)	
(Intercept)	854.140	15.755	12.166	54.213	6.99e-16	***
conditionsmall	-49.780	8.442	385.000	-5.897	8.12e-09	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

We can see that the experimental condition is significant as we see the p -value associated with the t-test on the parameter is significant ($p < .001$).

We can also compare this model with our fixed effect of condition with a model which doesn't have this fixed effect. If the difference between the two models is significant, then we can conclude that the fixed effect is significant.

```
mixed_model_null <- lmer(rt ~ (1 | subject) + (1 | item),  
                          data = mixed_model_data)
```

Comparing Models Using LRT

We can use the Likelihood Ratio Test (LRT) to compare our model with the fixed effect of Condition to the model without.

```
anova(mixed_model, mixed_model_null)
```

```
Data: mixed_model_data
```

```
Models:
```

```
mixed_model_null: rt ~ (1 | subject) + (1 | item)
```

```
mixed_model: rt ~ condition + (1 | subject) + (1 | item)
```

		npar	AIC	BIC	logLik	deviance	Chisq	Df	Pr(>Chisq)
mixed_model_null	4	4751.5	4767.5	-2371.8	4743.5				
mixed_model	5	4720.1	4740.1	-2355.1	4710.1	33.368	1	7.626e-09	***

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```

Data: mixed_model_data
Models:
mixed_model_null: rt ~ (1 | subject) + (1 | item)
mixed_model: rt ~ condition + (1 | subject) + (1 | item)
      npar AIC BIC      logLik deviance  Chisq Df Pr(>Chisq)
mixed_model_null  4 4751.5 4767.5 -2371.8    4743.5
mixed_model      5 4720.1 4740.1 -2355.1    4710.1 33.368   1 7.626e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

We see the Likelihood Ratio Test is significant - note the AIC, BIC, and deviance values are all lower for the model with our fixed effect. Deviance is the same as the residual sum of squares in linear models.

LRTs should only be conducted with nested models - i.e., when one model is a subset of the other.

Modelling differences in the magnitude of our effect

So far we have accounted for the possibility that our subjects and items might have different reaction time baselines - that some people are faster at responding than others (which is why we introduced the separate random intercepts).

But what if the *magnitude* of the effect of Condition is different for different subjects, and also what if the *magnitude* of the effect of Condition is different for different items?

So far we are assuming constant magnitudes of the effect across subjects and items

```
coef(mixed_model)
$subject
  (Intercept) conditionsmall
1      896.2420         -49.78
2      872.6334         -49.78
3      858.2934         -49.78
4      901.0512         -49.78
5      874.9943         -49.78
6      839.7562         -49.78
7      842.9915         -49.78
8      801.1955         -49.78
9      841.2427         -49.78
10     812.9998         -49.78


$item
  (Intercept) conditionsmall
1      868.8340         -49.78
2      833.9098         -49.78
3      852.9328         -49.78
4      837.1982         -49.78
5      877.8252         -49.78
```

The different intercepts for each subjects and for each item take into account individual baseline differences.

However, this doesn't take into account the fact our effect might be bigger for some subjects than for others (and for some items than for others). In other words, the slopes are all currently the same (-49.78).

Let's model the variability in the magnitude of the effect

```
mixed_model_slopes <- lmer(rt ~ condition +  
  (1 + condition | subject) +  
  (1 + condition | item),  
  data = mixed_model_data)
```



These modified terms tell the model to expect different intercepts for Condition (which we had before) as well as differing slopes as a function of the factor Condition. Remember, these are our random effects.

Now let's look at the coefficients in our model

```
coef(mixed_model_slopes)
$subject
  (Intercept) conditionsmall
1    881.8123   -11.07787
2    866.2407   -32.46840
3    858.8051   -57.03899
4    888.3678   -26.08780
5    869.7440   -41.46898
6    843.8967   -58.48443
7    847.1946   -62.21594
8    814.3650   -69.96477
9    846.9626   -69.07433
10   824.0112   -69.91850
```

The slopes between the two levels of our Condition differ for each subject...

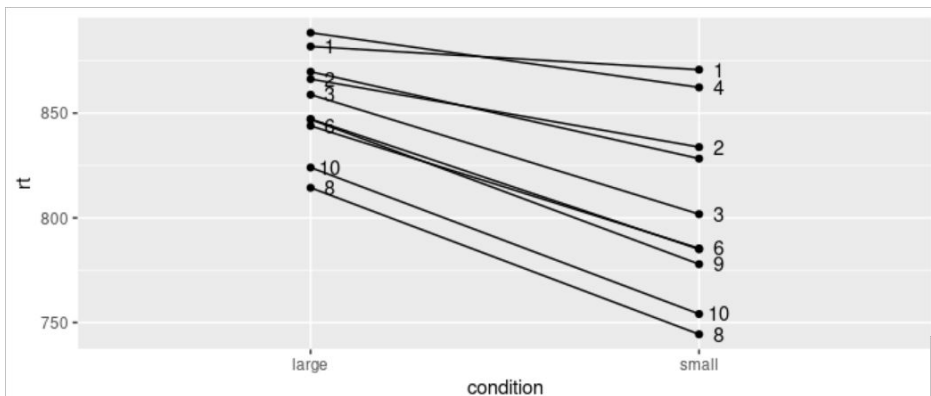


```
$item
  (Intercept) conditionsmall
1    868.9277   -51.97957
2    838.7420   -59.38390
3    844.5847   -28.36271
4    846.8623   -72.12582
5    871.5834   -37.04801
```

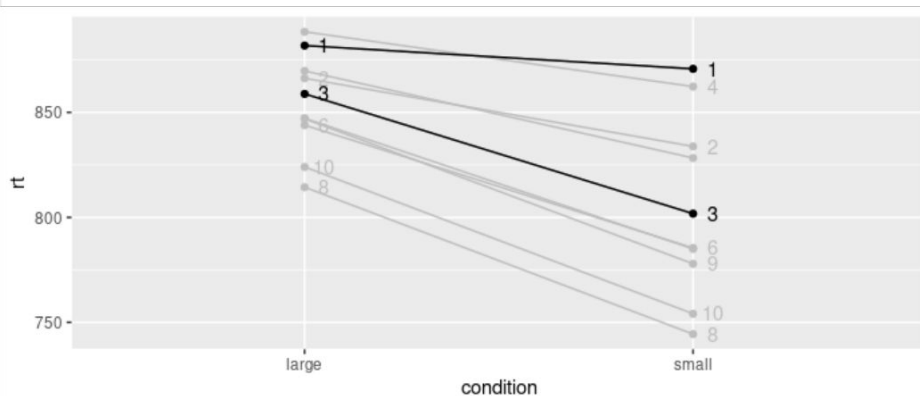
...and also for each item.



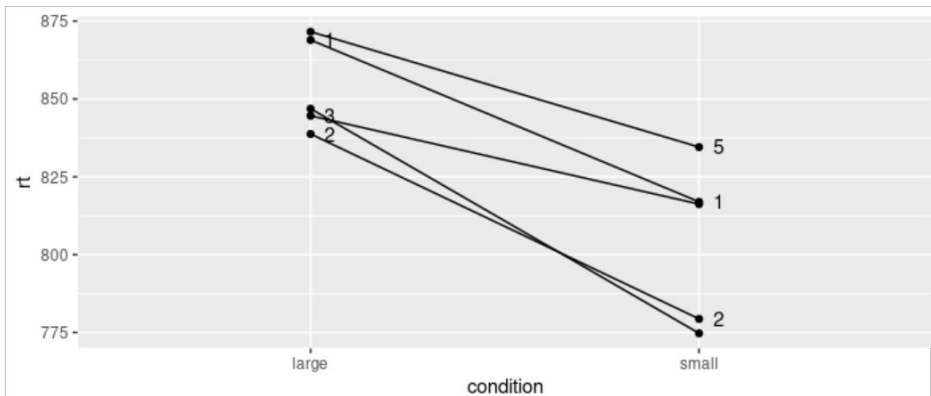
Plotting the individual intercepts and slopes for subjects



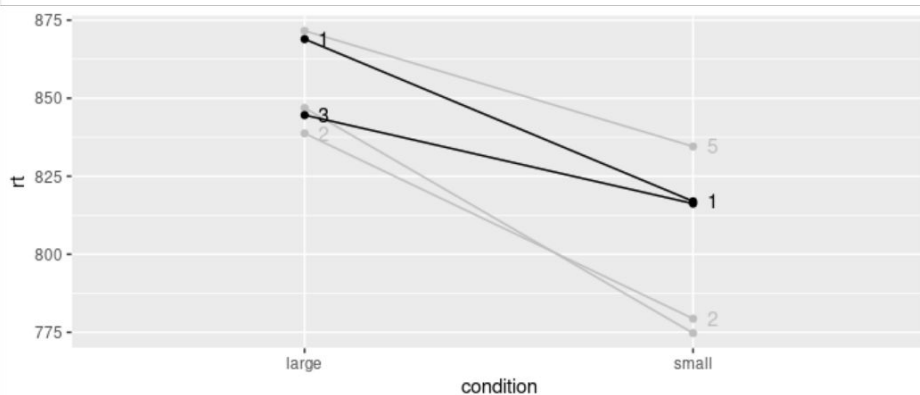
Let's just look at subjects 1 and 3 - for subject 1 we can see the difference between conditions looks pretty small compared to the difference for subject 3.



Plotting the individual intercepts and slopes for items



Let's just look at items 1 and 3 - for these items, the reaction times for the `small` condition are pretty much the same - but for the `large` condition item 1 is about 25 ms. slower than for item 3.



Partial Pooling in LMMs

LMMs use partial pooling to estimate the parameters of the model coefficients.

Partial pooling takes account of the individual slopes and intercepts for each level of the random effect structure, but also the slope and intercept of the overall model (which ignores how things vary from one participant to the next).

The use of partial pooling is one reason why LMMs are so powerful - they can cope with missing data (by being sensitive to properties of the overall dataset) and are not too affected by extreme data points (because they know these are quite unlikely in the context of the larger dataset - shrinkage reduces the influence of these extreme values on your parameter estimates).

Have a look at this great blogpost by Tristan Mahr:

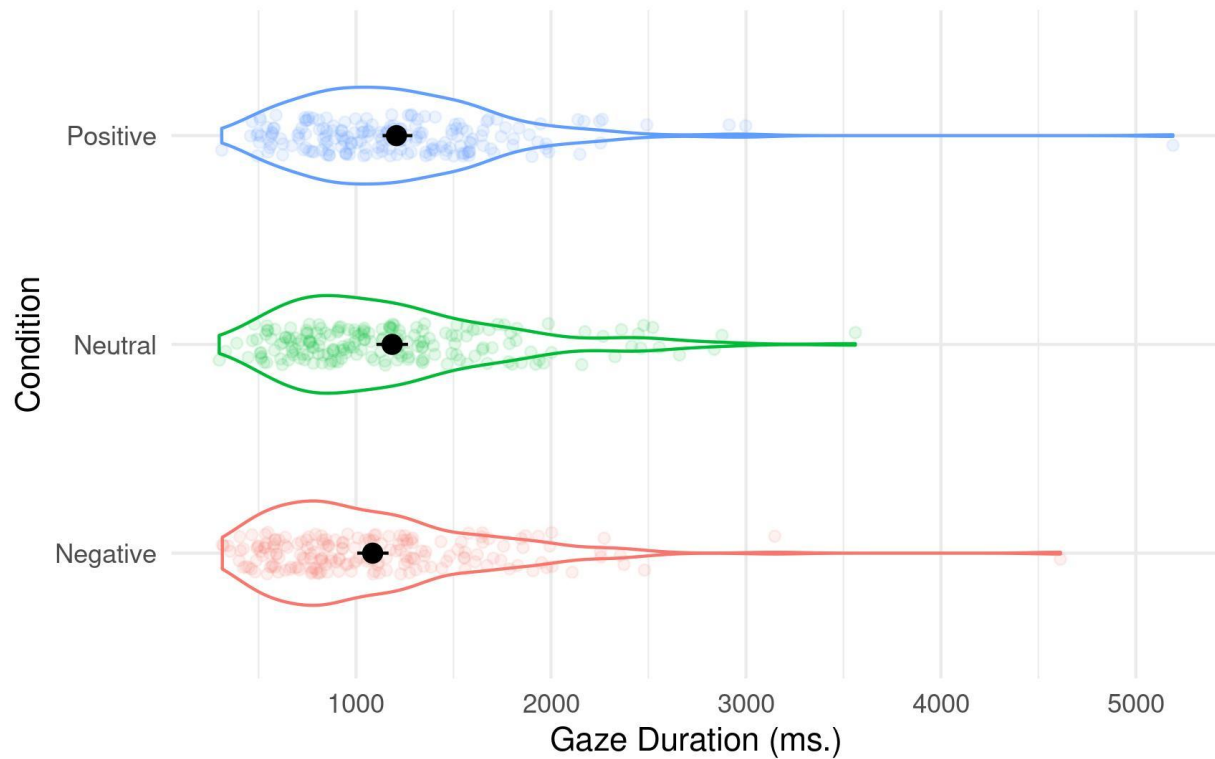
<https://www.tjmahr.com/plotting-partial-pooling-in-mixed-effects-models/>

One Factor Design Example

Imagine we measured 24 subjects' gaze duration while reading a sentence that appeared in one of three conditions - Negative, Neutral, and Positive. We had 24 items. The study is repeated measures (so everyone saw every condition).

```
head(tidied_factor_1_data)
# A tibble: 6 x 4
  subject item condition gaze
  <fct>    <fct> <fct>    <dbl>
1 S1      I1    Neutral    867
2 S1      I2    Positive  1061
3 S1      I3    Negative   771
4 S1      I4    Neutral    626
5 S1      I5    Positive  1283
6 S1      I6    Negative   846
```

One Factor Design Example



Building our model – attempt 1

```
factor_1_model <- lmer(gaze ~ condition + (1 + condition | subject) +  
                      (1 + condition | item), data = tidied_factor_1_data)
```

```
boundary (singular) fit: see ?isSingular
```

The warning we receive suggests we might be trying to estimate more parameters that can be estimated using the data set of the size we have. In other words, our model may be too complex. One solution would be to ignore the warning - especially if there is a strong theoretical reason to model all of the terms. Another solution would be to simplify the random effects structure until the warning goes away.

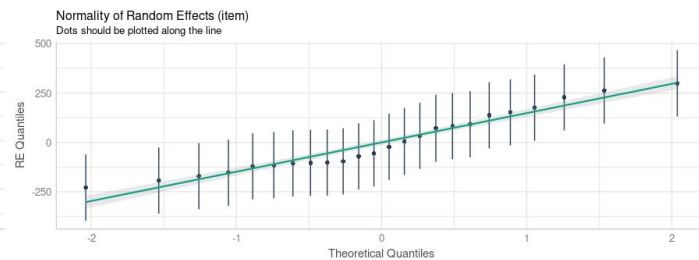
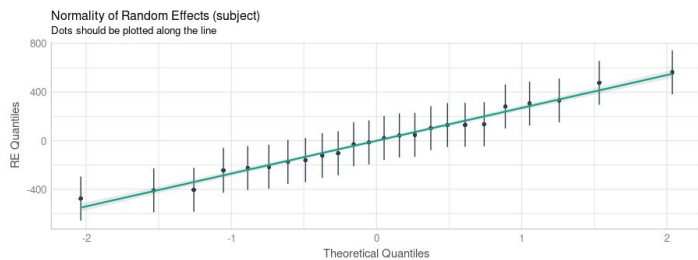
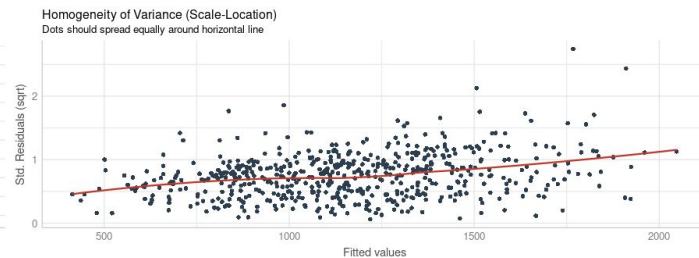
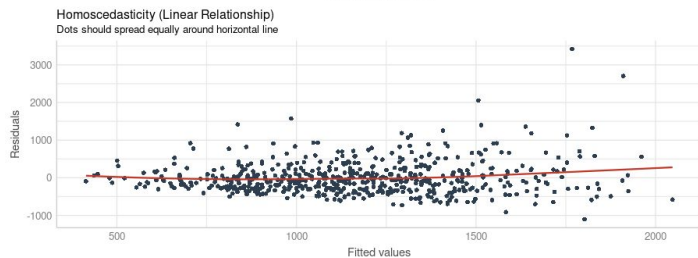
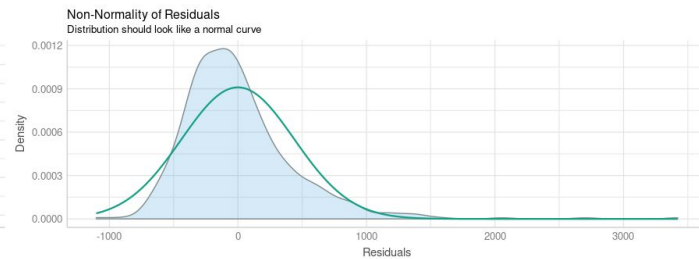
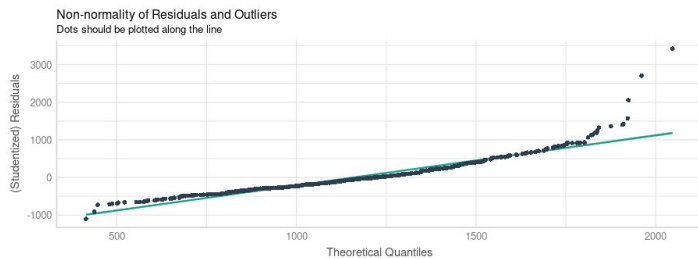
Building our model – attempt 2

```
factor_1_model <- lmer(gaze ~ condition + (1 | subject) + (1 | item),  
  data = tidied_factor_1_data)
```

If we drop both random slopes, we end up with a model that doesn't generate the warning. We can then use the `check_model()` function from the `{performance}` package to check our model assumptions.

```
check_model(factor_1_model)
```

Checking our assumptions



Interpreting our model

```
summary(factor_1_model)
Linear mixed model fit by REML. t-tests use Satterthwaite's method ['lmerModLmerTest']
Formula: gaze ~ condition + (1 | subject) + (1 | item)
Data: tidied_factor_1_data
```

REML criterion at convergence: 8713.1

Scaled residuals:

	Min	1Q	Median	3Q	Max
	-2.4240	-0.6246	-0.1505	0.4069	7.5250

Random effects:

Groups	Name	Variance	Std.Dev.
subject	(Intercept)	81340	285.2
item	(Intercept)	29586	172.0
Residual		206838	454.8

Number of obs: 574, groups: subject, 24; item, 24

Fixed effects:

	Estimate	Std. Error	df	t value	Pr(> t)
(Intercept)	1083.90	75.53	45.12	14.350	< 2e-16 ***
conditionNeutral	100.84	46.55	525.13	2.166	0.03073 *
conditionPositive	123.40	46.48	525.09	2.655	0.00818 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:

	(Intr)	cndtnN
conditnNtrl	-0.308	
conditnPstv	-0.309	0.501

These tests compare the Intercept to first the Neural condition, then to the Positive.

LRT

So the summary on the previous slide tells us only part of the story. Let's compare the model with the fixed effect of condition to a model without.

```
factor_1_model_null <- lmer(gaze ~ (1 | subject) + (1 | item),  
                             data = tidied_factor_1_data)
```

```
anova(factor_1_model, factor_1_model_null)
```

```
Data: tidied_factor_1_data
```

```
Models:
```

```
factor_1_model_null: gaze ~ (1 | subject) + (1 | item)
```

```
factor_1_model: gaze ~ condition + (1 | subject) + (1 | item)
```

	npar	AIC	BIC	logLik	deviance	Chisq	Df	Pr(>Chisq)
factor_1_model_null	4	8758.1	8775.6	-4375.1	8750.1			
factor_1_model	6	8754.2	8780.3	-4371.1	8742.2	7.9515	2	0.01877 *

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The models differ -
factor_1_model has lower
deviance (lower residual
sum of squares).



Which level(s) of our factor differ(s) from which other level(s)?

We're going to use the `emmeans()` function from the `{emmeans}` package.

```
emmeans(factor_1_model, pairwise ~ condition)
```

```
$emmeans
```

condition	emmean	SE	df	lower.CL	upper.CL
Negative	1084	75.5	45.1	932	1236
Neutral	1185	75.5	45.1	1033	1337
Positive	1207	75.5	45.0	1055	1359

```
Degrees-of-freedom method: kenward-roger
```

```
Confidence level used: 0.95
```


```
$contrasts
```

contrast	estimate	SE	df	t.ratio	p.value
Negative - Neutral	-100.8	46.5	525	-2.166	0.0780
Negative - Positive	-123.4	46.5	525	-2.655	0.0222
Neutral - Positive	-22.6	46.5	525	-0.485	0.8783

```
Degrees-of-freedom method: kenward-roger
```

```
P value adjustment: tukey method for comparing a family of 3 estimates
```

Note, with Tukey corrected multiple comparisons only the Negative vs. Positive comparison is significant.



A few points to note...

Models can only be compared to each other using the LRT if they are nested - in other words, if one model is a subset of the other. Models with different fixed and random effects structures cannot be compared in this way - use AIC or BIC comparisons.

AIC is the Akaike Information Criterion and measures how much 'information' is not captured by our model (values that are relatively lower are better).

Absolute AIC values cannot be interpreted - they have to be compared with the AIC value of another model. AIC penalises the addition of new parameters in a model - but not as much as BIC.