Mixed Models Part 2

Unit Coordinator: Dr Danna R. Gifford
(danna.gifford@manchester.ac.uk)

Original author: Prof Andrew J. Stewart
(drandrewjstewart@gmail.com)

Recap (and overview of this session)

- Previously we looked at mixed models and examined how they were related to (but more flexible) than models built using the general linear model.
- We built mixed models for designs involving one factor, and looked at how we decide what kind of random effects structure to build.
- We also spent some time looking at how best to interpret the results of mixed models, and how to test whether the assumptions underlying our mixed models have been met.
- In this session we're going to look at mixed models for factorial designs, generalised mixed models for where our outcome (dependent) variable is not continuous, and models for cases where our outcome variable is ordinal (e.g., as you might get from using a Likert scale).

LMMs for factorial designs

We have a 2 x 2 repeated measures design. The first factor is Context (Negative vs. Positive) and the second is Sentence Type (Negative vs. Positive). The DV is reading time duration to a Target Sentence (measured in ms.). We have 60 subjects, and 28 items.

```
head(tidied_factorial_data)
# A tibble: 6 x 5
  subject item    RT     context  sentence
  <fct>   <fct> <dbl>    <fct>    <fct>
1 1           3  1270    Negative Positive
2 1           7   739    Negative Positive
3 1          11   982    Negative Positive
4 1          15  1291    Negative Positive
5 1          19  1734    Negative Positive
6 1          23  1757    Negative Positive
```

```
str(tidied_factorial_data)
tibble [1,680 × 5] (S3: tbl_df/tbl/data.frame)
 $ subject : Factor w/ 60 levels "1","2","3","4",..: 1 1 1 1 1 1 1 2 2 2 ...
 $ item  : Factor w/ 28 levels "1","2","3","4",..: 3 7 11 15 19 23 27 4 8 12
...
 $ RT    : num [1:1680] 1270 739 982 1291 1734 ...
 $ context : Factor w/ 2 levels "Negative","Positive": 1 1 1 1 1 1 1 1 1 1 ...
 $ sentence: Factor w/ 2 levels "Negative","Positive": 2 2 2 2 2 2 2 2 2 2 …

tidied_factorial_data %>%
  group_by(context, sentence) %>%
  summarise(mean_rt = mean(RT), sd_rt = sd(RT))

# A tibble: 4 x 4
# Groups:   context [2]
  context   sentence     mean_rt  sd_rt
  <fct>      <fct>        <dbl>   <dbl>
1 Negative Negative      1474.    729.
2 Negative Positive      NA       NA
3 Positive Negative      NA       NA
4 Positive Positive      1579.    841.
```
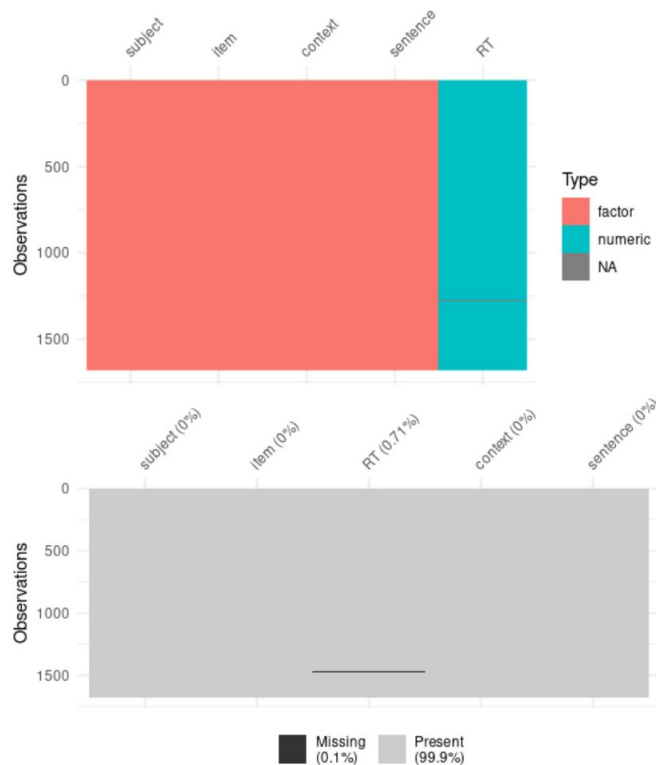
So what's going on here?

Using {visdat}

`vis_dat(tidied_factorial_data)`

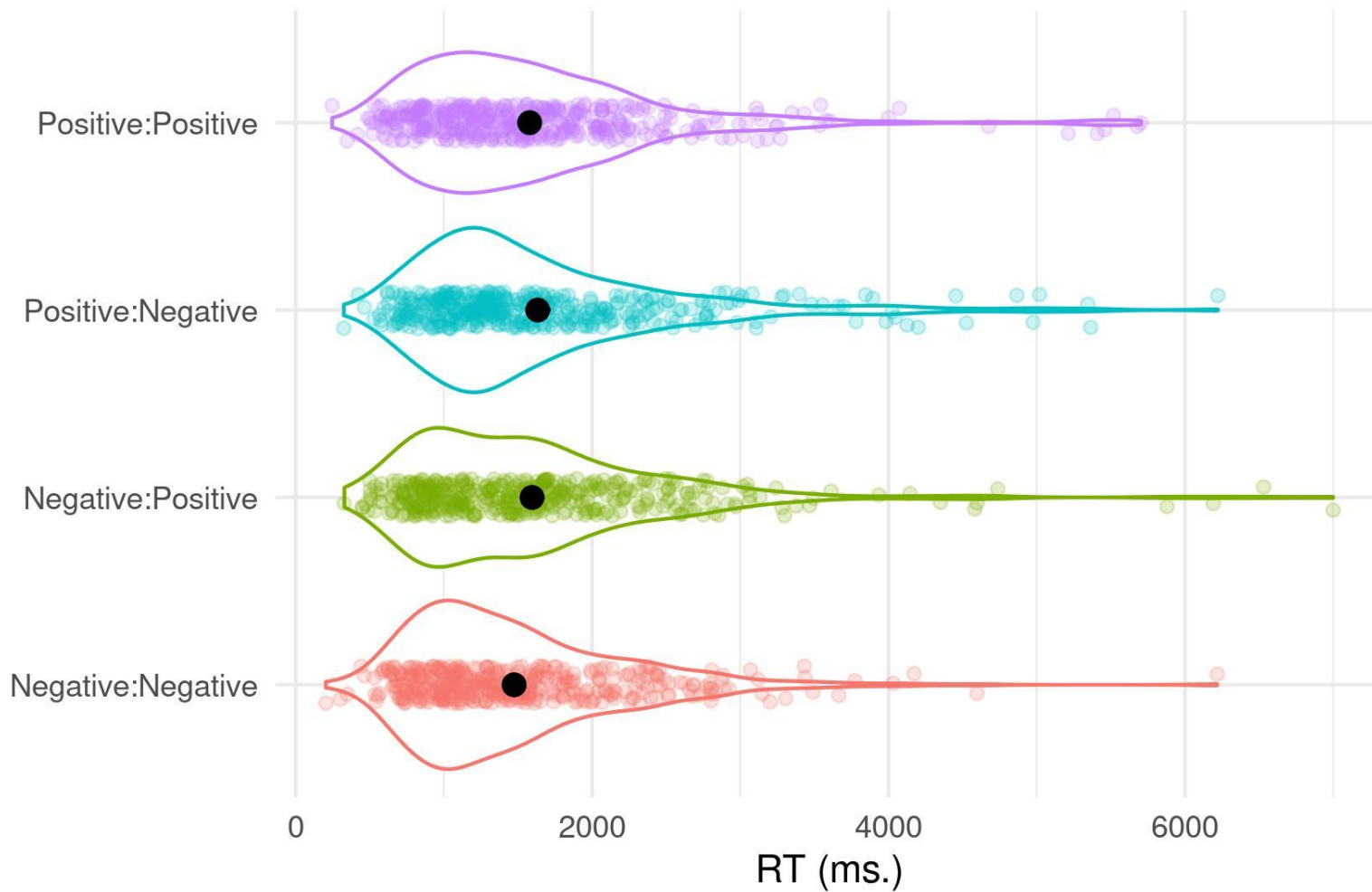`vis_miss(tidied_factorial_data)`

Let's filter out missing data (NAs)

```
tidied_factorial_data %>%
  filter(!is.na(RT)) %>%
  group_by(context, sentence) %>%
  summarise(mean_rt = mean(RT), sd_rt = sd(RT))

# A tibble: 4 x 4
# Groups:   context [2]
  context   sentence    mean_rt     sd_rt
  <fct>     <fct>       <dbl>       <dbl>
1 Negative  Negative    1474.       729.
2 Negative  Positive    1595.       887.
3 Positive  Negative    1633.       877.
4 Positive  Positive    1579.       841.
```

Setting up our contrasts

By default, R uses dummy or treatment coding for the different experimental factors. Recall how we used this type of coding when we examined ANOVA as a special case of the linear model. However, for mixed models this default coding can make parameter estimates harder to understand (and can result in misinterpretation of main vs. simple effects). For factors with 2 levels, we can use sum/deviation coding - this means that the intercept for the overall model will be equal to the grand mean (i.e., the mean of all our conditions).

```
contrasts(tidied_factorial_data$context) <- matrix(c(.5, -.5))
contrasts(tidied_factorial_data$sentence) <- matrix(c(.5, -.5))
```

Building our model – attempt 1

```
factorial_model <- lmer(RT ~ context * sentence +
                            (1 + context * sentence | subject) +
                            (1 + context * sentence | item),
                        data = tidied_factorial_data)
```

This is a maximal model in which we try to model the most complex random effects structure we can. The fixed effect `context * sentence` corresponds to an effect of `context`, an effect of `sentence`, and the interaction between the two. It is a more succinct way of writing `context + sentence + context : sentence`.

Building our model – attempt 2

```
Warning message:
In checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv,  :
  Model failed to converge with max|grad| = 0.005999 (tol = 0.002,
component 1)
```

Let's simplify the random effects structure by dropping the interaction term for the subjects random effect.

```
factorial_model <- lmer(RT ~ context * sentence +
                    (1 + context + sentence | subject) +
                    (1 + context * sentence | item),
                   data = tidied_factorial_data)
```

Checking our assumptions
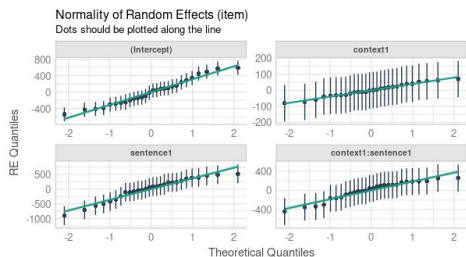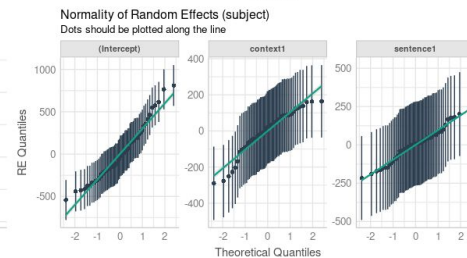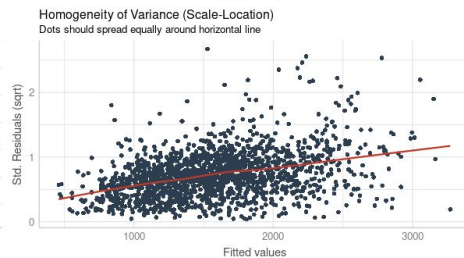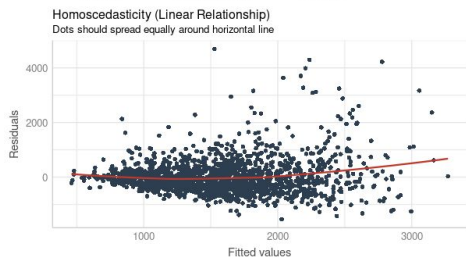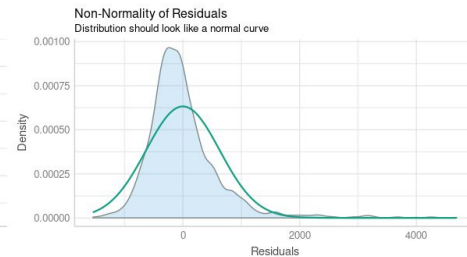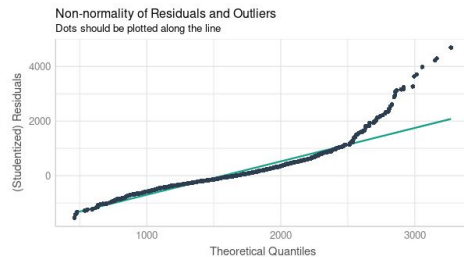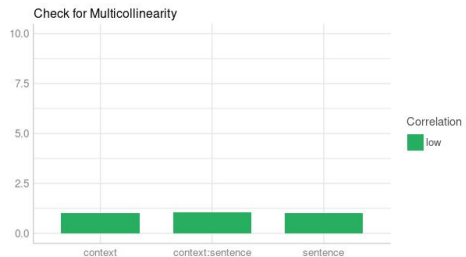
Interpreting the interaction

We can use the `emmeans()` function to run pairwise comparisons in order to determine what is driving the interaction:

```
emmeans(factorial_model, pairwise ~ context*sentence, adjust = "none")
$emmeans
 context  sentence  emmean   SE   df lower.CL upper.CL
 Negative Negative    1474 80.9 38.9     1310     1638
 Positive Negative    1627 99.0 43.9     1428     1827
 Negative Positive    1595 96.5 37.1     1399     1790
 Positive Positive    1579 90.2 48.4     1398     1761

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95


$contrasts
 contrast                             estimate   SE   df  t.ratio p.value
 Negative Negative - Positive Negative  -153.1  51.7 28.4  -2.960  0.0061
 Negative Negative - Negative Positive  -120.6  90.3 29.1  -1.335  0.1922
 Negative Negative - Positive Positive  -105.2  92.0 29.0  -1.144  0.2621
 Positive Negative - Negative Positive    32.5  97.1 31.4   0.335  0.7399
 Positive Negative - Positive Positive    47.9  98.3 28.4   0.487  0.6301
 Negative Positive - Positive Positive    15.4  59.9 28.7   0.256  0.7996

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

We can see the interaction is being driven by reading times to Negative sentences in Negative vs. Positive contexts. Let's look again at the plot and descriptives...
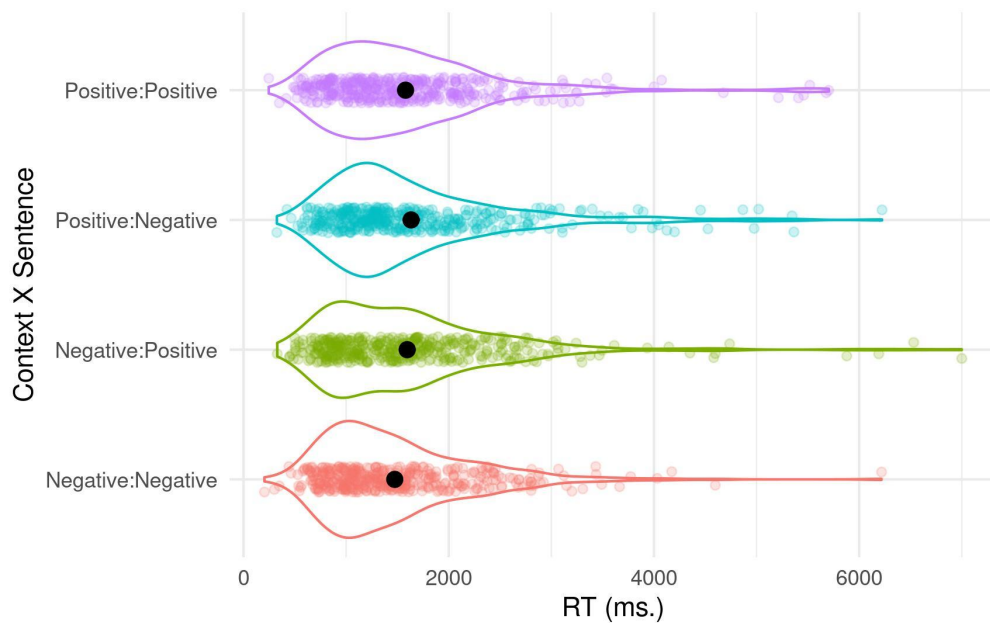
context	sentence	mean_rt	sd_rt
<fct>	<fct>	<dbl>	<dbl>
1 Negative	Negative	1474.	729.
2 Negative	Positive	1595.	887.
3 Positive	Negative	1633.	877.
4 Positive	Positive	1579.	841.

Interpreting our model

```
summary(factorial_model)
Linear mixed model fit by REML. t-tests use Satterthwaite's method ['lmerModLmerTest']
Formula: RT ~ context * sentence + (1 + context + sentence | subject) +
            (1 + context * sentence | item)
   Data: tidied_factorial_data

REML criterion at convergence: 26628.6

Scaled residuals:
            Min      1Q  Median        3Q      Max
        -2.3460 -0.5439 -0.1563  0.3202  7.1297

Random effects:
 Groups   Name                    Variance Std.Dev. Corr
 subject  (Intercept)             106850   326.88
          context1                21848   147.81   -0.68
          sentence1               28914   170.04   -0.08 -0.28
 item     (Intercept)             105902   325.43
          context1                4919     70.14    0.27
          sentence1               163485   404.33   -0.02 -0.11
          context1:sentence1      56612   237.93   -0.71 -0.81 -0.21
 Residual                         432879   657.94
Number of obs: 1668, groups:  subject, 60; item, 28

Fixed effects:
                    Estimate Std. Error       df     t value Pr(>|t|)
(Intercept)         1568.68    76.31        50.04    20.556   <2e-16 ***
context1            -68.87     39.77        31.62    -1.732   0.0931 .
sentence1           -36.35     85.81        29.71    -0.424   0.6749
context1:sentence1  -168.45    78.68        26.32    -2.141   0.0417 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) cntxt1 sntnc1
context1    -0.109
sentence1   -0.025 -0.070
cntxt1:snt1 -0.327 -0.147 -0.112
```

Our fixed effect parameter estimates. The interaction is significant (p = .0417)

Writing up the results

The analyses were carried out using the lme4 package (Bates, Maechler, Bolker, & Walker, 2015) to fit the linear mixed models for the reading time measure in R version 3.6.3 (R Development Core Team, 2020). Deviation coding was used for each of the two experimental factors (Barr et al., 2013). Pairwise comparisons conducted with the emmeans package (Lenth, 2018) were used to investigate the interaction for the reading time measure. Below we report regression coefficient estimates, standard errors, dfs, *t*-values, and *p*-values for the intercept and fixed effects.Degrees of freedom were approximated using the Kenward-Roger method. Restricted maximum likelihood estimation was used for the reporting of parameters. For pairwise comparisons we report the *t*-values and *p*-values.

	Estimate	SE	df	*t*-value	*p*-value
Intercept	1568.68	76.31	50.04	20.556	<2e-16
Context	-68.87	39.77	31.62	-1.732	0.0931
Sentence	-36.35	85.81	29.71	-0.424	0.6749
Context x Sentence	-168.45	78.68	26.32	-2.141	0.0417

Writing up the results

When reporting the results of LMMs, it is important to provide all the information that someone would need to reproduce your analysis exactly. It's important to provide versions for R and the packages you're using so that exactly the same version of R and associated packages can be used by someone else. You can use `sessionInfo()` to generate a list of the packages (plus their version numbers) loaded in your R environment.

We're in a world where many journals now ask for your analysis code and data to be uploaded as supplementary material.

One of the best ways to ensure reproducibility is to use Binder - and link to your Binderised script in your journal submission.

Addressing convergence errors

If you receive convergence errors, it means your are likely trying to estimate too many parameters than your data will allow. You need to simplify the random effects structure until you are able to estimate the parameters you want to. One way to do this is to simplify the random effects terms that explain the least amount of variance - assessed using the `summary()` function.

You could also systematically, one by one, drop interaction terms, then individual terms from your random effects structure until you are able to fit a model to your data. But you want to avoid random effects with just random intercepts (i.e., no slopes) as that can inflate the Type 1 error rate (Barr et al., 2013).

A few other things…

You can add participant and item covariates as fixed effects, and you can have a variety of continuous and categorical variables in your LMM. LMMs are *very* flexible.

You'll find that sometimes several models fit your data - you can run likelihood comparison test on nested models to determine which is the best fit. If you have a selection where not one is statistically better than the others, choose the model that makes most **theoretical** sense.
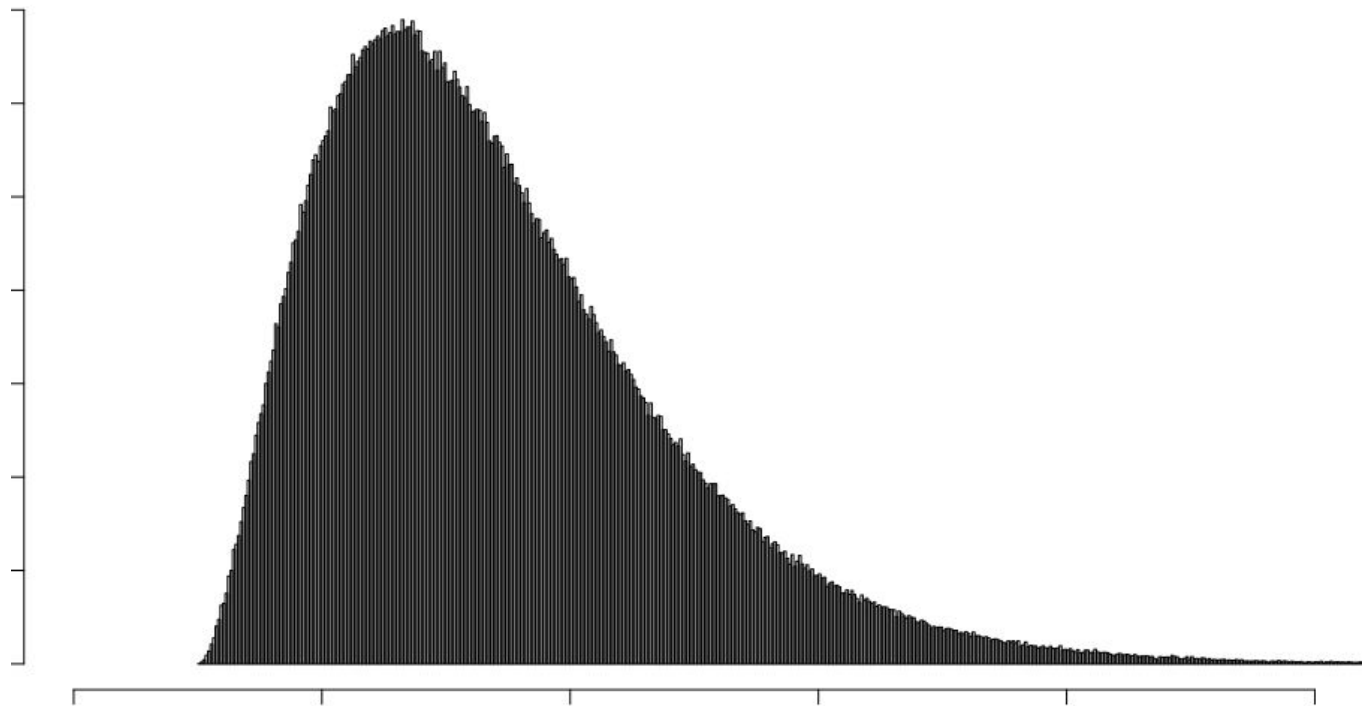
Other distributions

Looking at the 2 x 2 factorial data again, we can plot the response variable on a Cullen and Frey graph using the `{fitdistrplus}` package and the `descdist()` function.

```
descdist(tidied_factorial_data$RT)
```
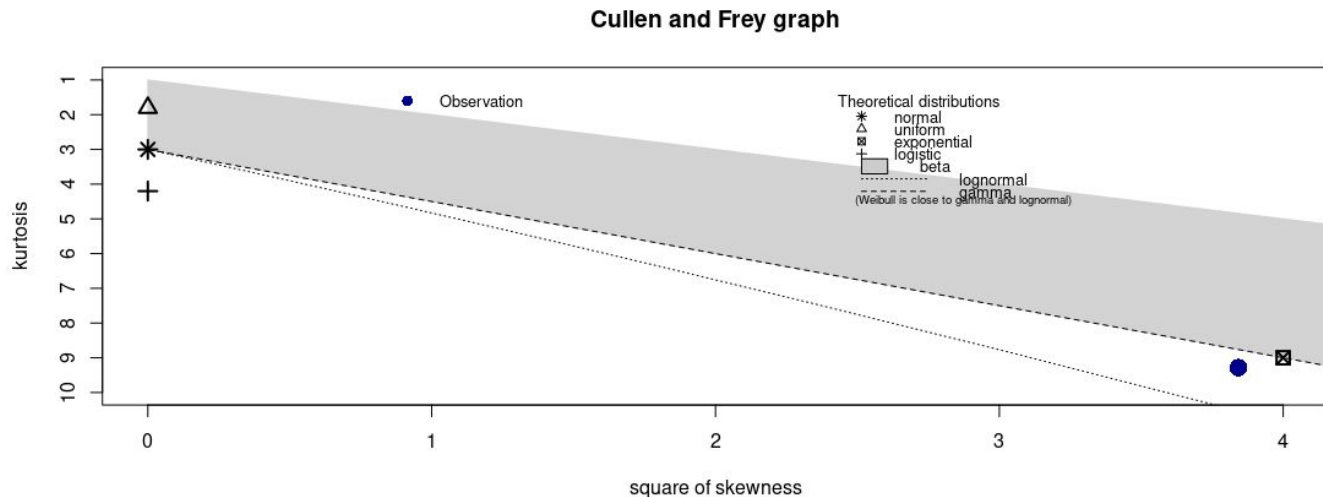
An Example of The Gamma Distribution

Maybe we should have built a model assuming sampling from the Gamma distribution?

Reaction time data very often follow the Gamma distribution (see Kliegl et al. 2010, Lo & Andrews, 2015, for discussion).

Gamma mixed models are often harder to fit and frequently require a simpler random effects structure. You can build them using the `glmer()` function and specify `family = Gamma)`

Kliegl, R., Masson, M. E. J., and Richter, E. M. (2010). A linear mixed model analysis of masked repetition priming. *Visual Cognition,18,* 655−681.

Lo, S., and Andrews, S. (2015). To transform or not to transform: using generalized linear mixed models to analyse reaction time data. *Frontiers in Psychology,* 6:1171.

A few other things...

Sometimes you'll find yourself trying to fit an over-parameterised model - this is one whether you are trying to estimate more components of the model than your data/design supports.

In `{lme4}`, you'll receive a "singular fit" error if your model appears over-parameterised - one solution is simplify the random effects structure (usually by removing random slopes) in a way that makes theoretical sense until you arrive at a model that fits (but doesn't overfit) your data.

Having said that, if the random effects structure makes complete theoretical sense then you might not want to simplify it. Often it's a judgement call.

Read more in "Parsimonious mixed models" by Bates et al.: https://arxiv.org/abs/1506.04967

What if your DV isn't a continuous variable?

In this section and the next we'll take a look at how you might build mixed models for variables that are binomial (e.g., zeros and ones) and ordinal (as might be found with Likert-scale data).

In eye movement work, in addition to reading time we also the number of times people re-read a region of text - these types of eye-movements are called "regressions". For any one person reading a region of text, they either re-read it, or they don't. Thus, the data are binomial/binary (not continuous). In our data sets containing a measure of eye-movement regressions, 1 corresponds to a region of text being re-read, 0 to it not being re-read.

Binomial data

In an eye-movement study we measured readers' regressions as they read sentences. We had sentences conveying three different types of meaning (Negative vs. Neutral vs. Positive). For each, we measured whether people did or did not make a regressive eye-movement. This is our DV and is coded as 0 or 1. We had 24 subjects and 24 items.

```
str(tidied_regressions_data)
tibble [553 × 4] (S3: tbl_df/tbl/data.frame)
 $ Subject  : Factor w/ 24 levels "S1","S10","S11",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ Item   : Factor w/ 24 levels "I1","I10","I11",..: 12 18 19 20 21 22 23 24 2 3 ...
 $ Condition: Factor w/ 3 levels "Negative","Neutral",..: 3 1 2 3 1 2 3 1 2 3 ...
 $ DV     : num [1:553] 0 0 0 0 0 1 0 0 0 0 …
```

Binomial data

```
head(tidied_regressions_data)
# A tibble: 6 x 4
  Subject Item  Condition DV
  <fct>   <fct> <fct>     <dbl>
1 S1      I2    Positive      0
2 S1      I3    Negative      0
3 S1      I4    Neutral       0
4 S1      I5    Positive      0
5 S1      I6    Negative      0
6 S1      I7    Neutral       1
```

```
tidied_regressions_data %>%
  group_by(Condition) %>%
  summarise(mean_DV = mean(DV), sd_DV =
sd(DV))

# A tibble: 3 x 3
  Condition mean_DV sd_DV
  <fct>       <dbl> <dbl>
1 Negative    0.247 0.433
2 Neutral     0.265 0.443
3 Positive    0.269 0.445
```

Building our model – attempt 1

Rather than build a linear mixed model, we build a generalised linear mixed model using the `glmer()` function in the {lme4} package. Instead of parameters being tested with a t-test, they are compared against the z-distribution. With `glmer()` we need to specify the distribution that our data come from. In this case, our data are binomial so we specify our model as follows:

```
binomial_model <- glmer(DV ~ Condition +
                        (1 + Condition | Subject) +
                        (1 + Condition | Item),
                        data = tidied_regressions_data, family = binomial)
```

In this case, the maximal model (i.e., with random slopes for both our random effects) fails to converge so we simplify it.

Building our model – attempt 2

So we end up having to drop random slopes from the subject random effect, and the item random effect entirely to be able to build a model that converges (and isn't overfitted).

```
binomial_model <- glmer(DV ~ Condition + (1 | Subject),
                        data = tidied_regressions_data,
                        family = binomial)
```

Building our model – attempt 2

```
summary(binomial_model)

Generalized linear mixed model fit by maximum likelihood (Laplace Approximation) [glmerMod]
 Family: binomial  ( logit )
Formula: DV ~ condition + (1 | subject)
   Data: tidied_regressions_data

      AIC      BIC   logLik deviance df.resid
   603.7 621.0   -297.9   595.7     549

Scaled residuals:
      Min      1Q  Median      3Q      Max
-1.3096 -0.6118 -0.4044  0.7636  3.1872

Random effects:
 Groups  Name    Variance Std.Dev.
 subject (Intercept) 0.8363   0.9145
Number of obs: 553, groups:  subject, 24

Fixed effects:
                  Estimate Std. Error   z value Pr(>|z|)
(Intercept)        -1.3096  0.2647      -4.947 7.55e-07 ***
ConditionNeutral    0.1093  0.2539       0.431    0.667
ConditionPositive   0.1162  0.2515       0.462    0.644
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
         (Intr) cndtnN
conditnNtrl -0.485
conditnPstv -0.488  0.507
```

There's not much going on here!

Taking a closer look at our data...

We don't actually have a huge amount of data, and about three times as many trials where we have zero regressions relative to there being a regression.

```
tidied_regressions_data %>%
  group_by(Condition, DV) %>%
  summarise(n())

# A tibble: 6 x 3
# Groups:   Condition [3]
  condition    DV       `n()`
  <fct>        <dbl>    <int>
1 Negative     0        140
2 Negative     1        46
3 Neutral      0        133
4 Neutral      1        48
5 Positive     0        136
6 Positive     1        50
```

Checking our model assumptions

Instead of checking for normality of residuals, with a generalised linear mixed model using binomial data we build a binned residual plot using the `binnedplot()` function from `{arm}`. We expect 95% of residuals to fall between the jagged lines (+- 2SEs).

```
binnedplot(fitted(binomial_model), resid(binomial_model, type = "response"))
```


Binned residual plot
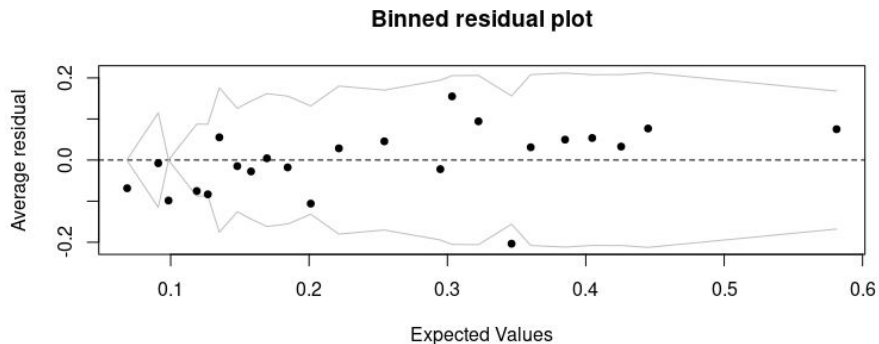
Modelling ordinal data

Often we might collect data using a Likert scale. These data are ordinal and so we should use the cumulative-link mixed model function `clmm()` from the package `{ordinal}`. Works similarly to LMMs in `{lme4}` but with one or two minor syntax changes…

An example: we had 42 participants rate images of sports on a scale of 0-10 corresponding to how much they liked each one. Before each rating measure, they saw a video of a sport that matched or mismatched the one they then had to rate (with a neutral video as baseline).

Modelling ordinal data

We want to know whether people's ratings were influenced by whether or not the sport they rated matched the one they had just seen. Our data file is called `ordinal_data_tidied`.

We have Subject, and SportType as our random effects.

VideoCondition corresponds to our condition - with the levels "Match", "Mismatch", and "Neutral".

Our DV is the column "Ratings". We need to ensure our DV is coded as an ordered factor using:
`ordinal_data_tidied$Ratings <- as.ordered(ordinal_data_tidied$Ratings)`

Plotting our data
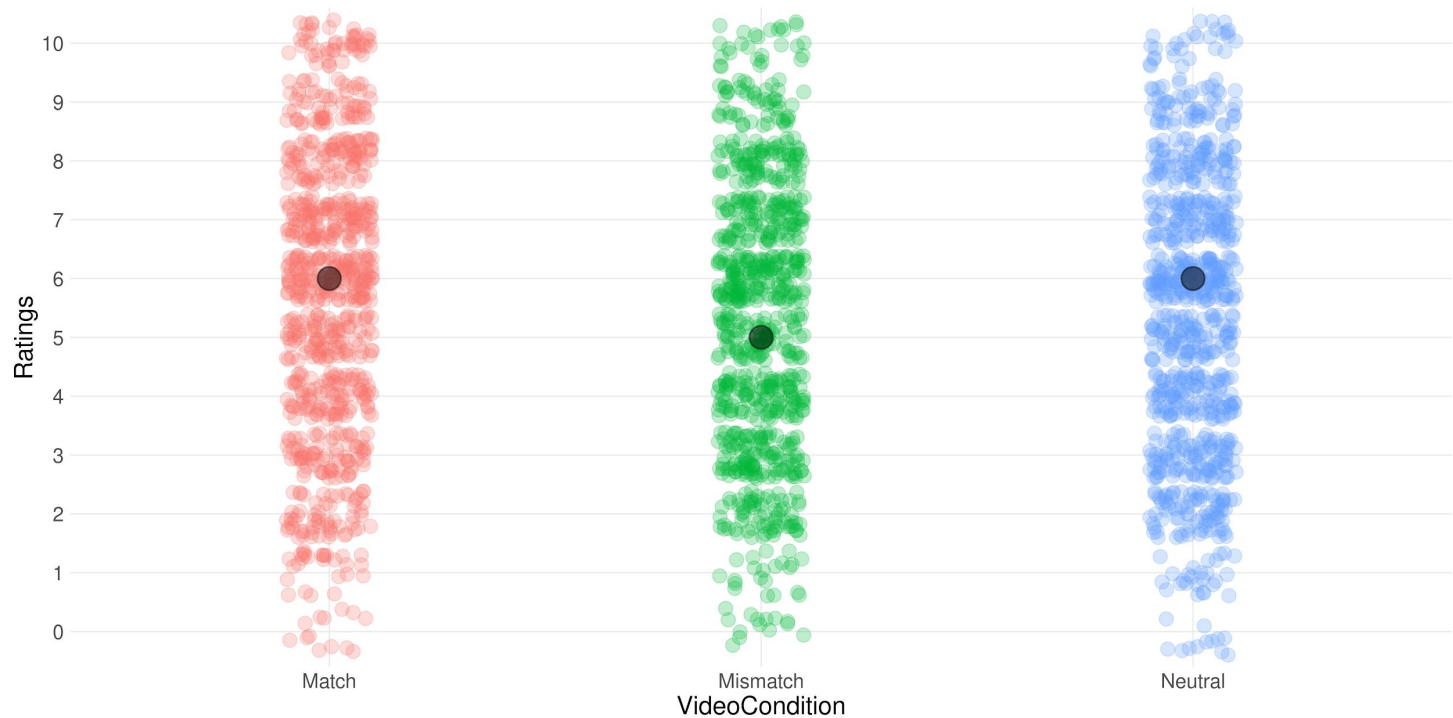
Building our model

We can build an ordinal mixed model using the `clmm()` function. Here we model a fixed effect of VideoCondition, and random effects of Subject and SportType.

```
ordinal_model <- clmm(Ratings ~ VideoCondition +
                    (1 + VideoCondition | Subject) +
                    (1 + VideoCondition | SportType),
                  data = ordinal_data_tidied)
```

Here we drop the fixed effect to build a null mode - note we make the intercept explicit:

```
ordinal_model_null <- clmm(Ratings ~ 1 +
                    (1 + VideoCondition | Subject) +
                    (1 + VideoCondition | SportType),
                  data = ordinal_data_tidied)
```

Comparing the two models

```
anova(ordinal_model, ordinal_model_null)

                   no.par     AIC        logLik     LR.stat    df    Pr(>Chisq)
ordinal_model_null 22         10829      -5392.6
ordinal_model      24         10826      -5389.0    7.3433     2     0.02543 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see the two models differ from each other - the model with the fixed effect has the lower AIC value.

Interpreting our model

We can use the `emmeans()` function to run our pairwise comparisons in order to tell which level of our VideoCondition factor differs from which other level(s).

```
emmeans(ordinal_model, pairwise ~ VideoCondition)
$emmeans
 VideoCondition emmean   SE  df asymp.LCL asymp.UCL
 Match           0.609 0.262 Inf    0.0948     1.123
 Mismatch        0.294 0.243 Inf   -0.1832     0.771
 Neutral         0.319 0.245 Inf   -0.1624     0.800

Confidence level used: 0.95

$contrasts
 contrast              estimate   SE        df  z.ratio  p.value
 Match - Mismatch       0.3148    0.1019    Inf  3.088    0.0057
 Match - Neutral        0.2902    0.1025    Inf  2.832    0.0129
 Mismatch - Neutral    -0.0245    0.0855    Inf -0.287    0.9556

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

We can see that the Match vs Mismatch conditions differ from each other, as do the Match vs. Neutral conditions. We can conclude that people's ratings for how much they liked a particular sport was influenced by whether they had just seen a video depicting the sport. When the video and sport matched, they give the sport a higher rating when when the video and sport mismatched.

Summary

- GLMMs are super flexible and much better than building models using data that are aggregated.
- But, you do need to be careful that you're building the right kind of model given your design and your outcome variable.
- Make sure you use appropriate coding when you're looking at main effects and interactions - many published papers don't.
- Always visualise your data before building any model.
- When several possible models can be built - consider which model(s) makes most sense given the theoretical framework you're testing in your experiment.
- When writing up the results of mixed models, ensure to specify all the detail necessary for someone else to re-create *exactly* the same analysis as you have run.