Mixed Models 1

Random intercepts

Daniela Palleschi

2024-01-12

Table of contents

Learning Objectives

Today we will learn…

- what linear mixed models are
- how to fit a random-intercepts model
- how to inspect and interpret a mixed effects model

Resources

- this lecture covers
  - Chapter 14 'Mixed Models 1: Conceptual Introduction' (until Section 14.8; Winter, 2019)
  - Winter (2014) (until page 16)
  - Sections 8.1-8.3 in Sonderegger (2023)
- we will be using the data from Biondo et al. (2022)

Set-up

```
# suppress scientific notation
options(scipen=999)
```

```
library(broman)
# function to format p-values
format_pval <- function(pval){
    dplyr::case_when(
        pval < .001 ~ "< .001",
        pval < .01 ~ "< .01",
        pval < .05 ~ "< .05",
        TRUE ~ broman::myround(pval, 3)
    )
}
```

Load packages

```
# load libraries
pacman::p_load(
               tidyverse,
               here,
               broom,
               janitor,
               ggeffects,
               sjPlot,
               # new packages for mixed models:
               lme4,
               lmerTest,
               broom.mixed,
               lattice)
```

Resolve conflicts

- both `lme4` and `lmerTest` have a function `lmer()`
    - for now we want to use the `lme4` version

```
lmer <- lme4::lmer
```

Load data

- data from Biondo et al. (2022)

```
df_biondo <-
  read_csv(here("data", "Biondo.Soilemezidi.Mancini_dataset_ET.csv"),
           locale = locale(encoding = "Latin1") ## for special characters in Spanish
           ) |>
  clean_names() |>
  mutate(gramm = ifelse(gramm == "0", "ungramm", "gramm")) |>
  mutate_if(is.character,as_factor) |> # all character variables as factors
  filter(adv_type == "Deic") |>
  droplevels()
```

And take a look at the data:

```
head(df_biondo)
```

```
# A tibble: 6 x 13
  sj     item adv_type adv_t verb_t gramm   roi label       fp    gp    tt    ri
  <fct> <dbl> <fct>    <fct> <fct>  <fct> <dbl> <fct>    <dbl> <dbl> <dbl> <dbl>
1 1        54 Deic     Past  Past   gramm     1 En la c~  1173  1173  1173     0
2 1        54 Deic     Past  Past   gramm     2 ayer te~   474   474   474     0
3 1        54 Deic     Past  Past   gramm     3 los car~   910   910   910     0
4 1        54 Deic     Past  Past   gramm     4 encarga~  1027  1027  1027     0
5 1        54 Deic     Past  Past   gramm     5 muchas ~   521   521   521     0
6 1        54 Deic     Past  Past   gramm     6 al prov~  1029  1029  1029     0
# i 1 more variable: ro <dbl>
```

Set contrasts

```
contrasts(df_biondo$verb_t) <- c(-0.5,+0.5)
contrasts(df_biondo$gramm) <- c(-0.5,+0.5)
```

```
contrasts(df_biondo$verb_t)
```

```
       [,1]
Past   -0.5
Future  0.5
```

```r
contrasts(df_biondo$gramm)
```

```
        [,1]
gramm   -0.5
ungramm  0.5
```

Linear mixed (effects) models

- mixed models allow for varying intercepts and slopes per level of some grouping factor
- recall that intercepts (can) represent the grand mean of the data
- slopes represent a change in y for a 1-unit change in x ($\frac{\Delta y}{\Delta x}$, "rise over run")
  - i.e., the difference between two categories, or for a 1-unit change of a continuous predictor
- random intercepts take into account that each level of a grouping factor can vary in their mean
- random slopes take into account that each level of a grouping factor can vary in the effect of a predictor

Random intercepts vs. random slopes

- Biondo et al. (2022) used a within-participant, a.k.a. repeated measures design
  - 60 participants saw 96 items, rotated throughout the conditions in a Latin square design
- we would expect some participants to be faster readers than others
  - this would be reflected in a shorter mean reading time
- some participants will tend to have a stronger effect of e.g., grammaticality than others
  - this would be reflected in a steeper slope for `gramm`aticality
- the same could be said for certain experimental items
  - reading times will vary by item e.g., for word length or familiarity
  - some items will also tend to have a stronger effect than others

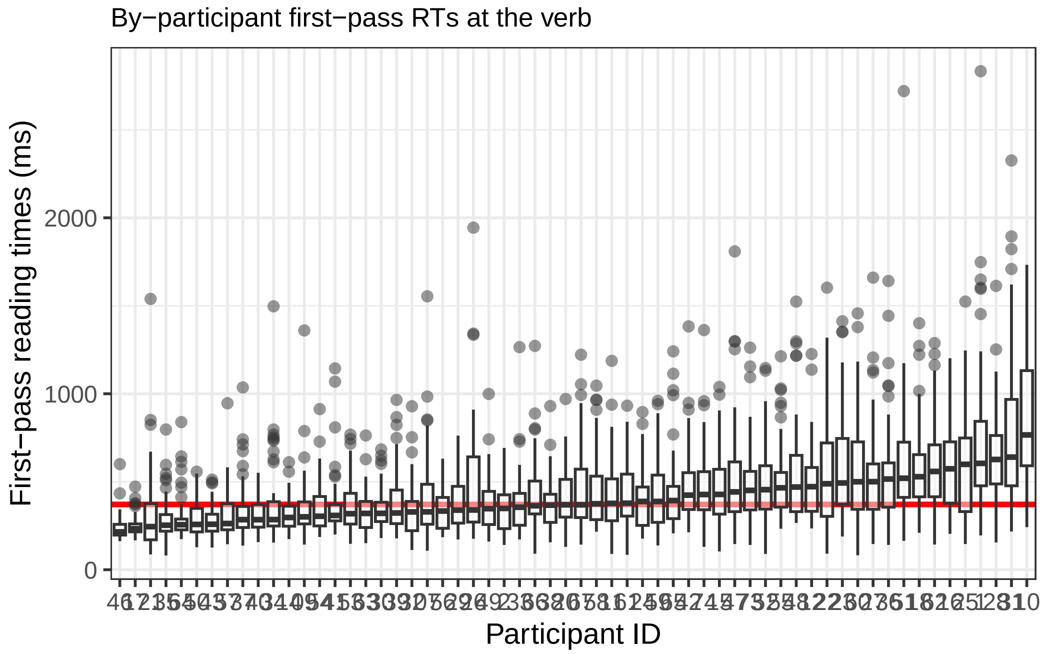


Figure 1: By-participant boxplot of first-pass RTs at the verb region with overall median FP value in red

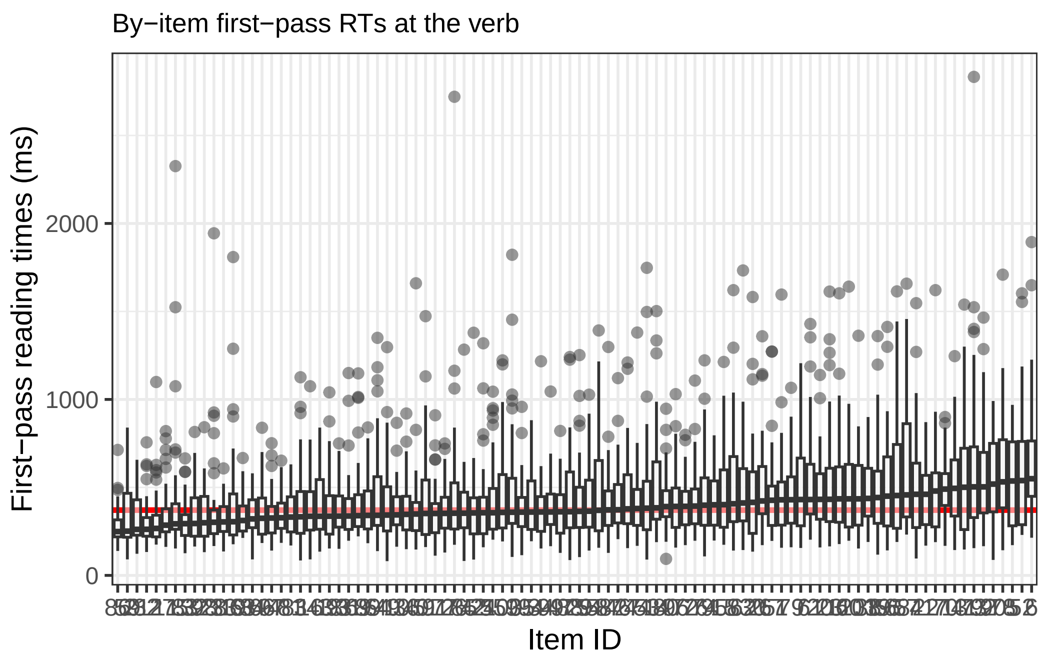


Figure 2: By-item boxplot of first-pass RTs at the verb region with overall median FP value in red

By-participant variance

By-item variance

By-participant varying intercepts and slopes

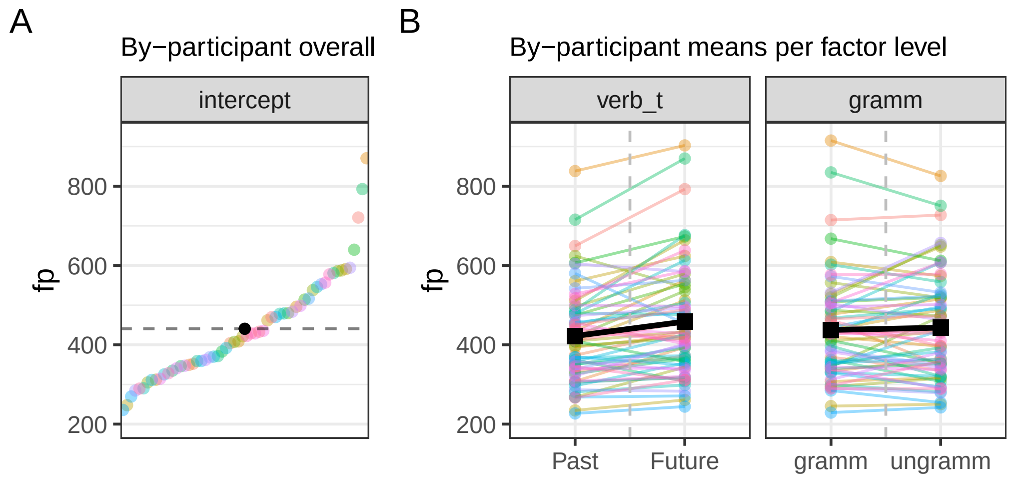


Figure 3: Predicted by-participant varying intercepts (A) and slopes (B) with overall effects in black

By-item varying intercepts and slopes

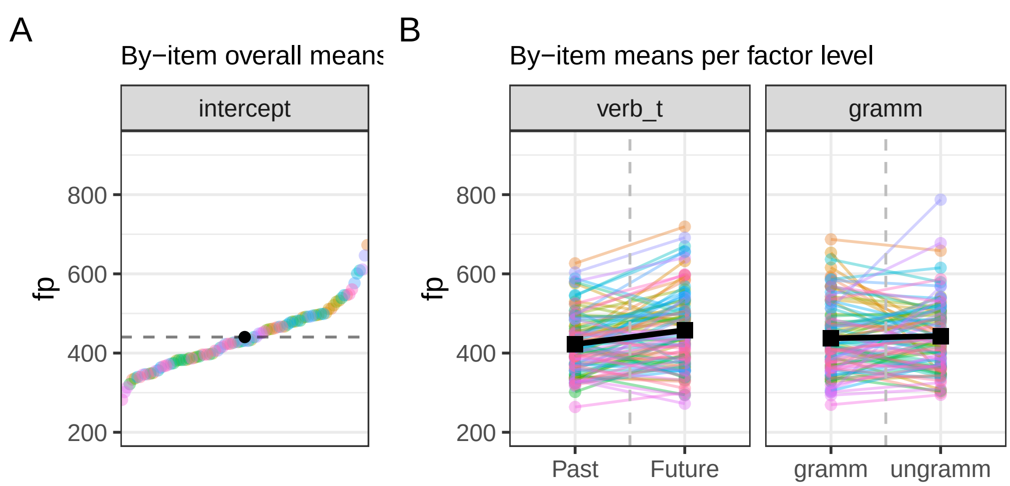


Figure 4: By-item varying intercepts (A) and slopes (B) with overall effects in black

Comparing participant and item

- just by eye-balling our data we see there was more variability in intercepts and effects by-participant than by-item
  - this is typical: People tend to vary more than our highly controlled experimental items
- how can we take this variability of both item and participant into account?
  - mixed models!
  - today we'll focus on varying intercepts, first for by-participants and then by-item

Random intercepts

- random intercepts = taking group-level variance in overall tendencies into account

Random intercepts: one grouping factor

- below is our first *mixed effects model*

```
fit_lmm_fp_sj <-
  lmer(log(fp) ~ verb_t*gramm +
         (1|sj),
       data = df_biondo,
       subset = roi == 4)
```

1. create an object `fit_lmm_fp_sj`, which contains...
2. a mixed model (`lmer()`): log first-pass RTs as a function of our fixed effects, plus...
3. varying intercepts (`1`) by-participant (`|sj`)
4. from our dataset
5. subsetted to only include the verb region
   - can also be done above the model using `filter(roi == 4)`

Summary

- we can use the `summary()` function, just as we did with (g)`lm()`

```
summary(fit_lmm_fp_sj)
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: log(fp) ~ verb_t * gramm + (1 | sj)
   Data: df_biondo
 Subset: roi == 4

REML criterion at convergence: 4479.1

Scaled residuals:
    Min      1Q  Median      3Q     Max
-4.0560 -0.6427 -0.0419  0.6168  4.0901

Random effects:
 Groups   Name        Variance Std.Dev.
 sj       (Intercept) 0.06573  0.2564
 Residual             0.18030  0.4246
Number of obs: 3795, groups:  sj, 60

Fixed effects:
                Estimate Std. Error t value
(Intercept)     5.957102   0.033809 176.199
verb_t1         0.062209   0.013787   4.512
gramm1          0.003466   0.013787   0.251
verb_t1:gramm1 -0.015741   0.027573  -0.571

Correlation of Fixed Effects:
            (Intr) vrb_t1 gramm1
verb_t1      0.000
gramm1       0.000 -0.002
vrb_t1:grm1  0.000  0.002  0.000
```

Model info

```
Linear mixed model fit by REML ['lmerMod']. #<1>
Formula: log(fp) ~ verb_t * gramm + (1 | sj) #<2>
   Data: df_biondo #<3>
 Subset: roi == 4 #<4>

REML criterion at convergence: 4479.1 #<5>

Scaled residuals: #<6>
    Min      1Q  Median      3Q     Max
```

```
-4.0560 -0.6427 -0.0419  0.6168  4.0901
```

① Model description (object class = `lmerMod`)
② Model formula
③ Data
④ Any subsetting
⑤ REML: Restricted Maximum Likelihood; important for model comparison, but not for us today
⑥ Residuals: do these look normally distributed?

Fixed effects

```
Fixed effects: #<1>
                Estimate Std. Error t value #<2>
(Intercept)     5.957102   0.033809 176.199 #<3>
verb_t1         0.062209   0.013787   4.512 #<4>
gramm1          0.003466   0.013787   0.251 #<5>
verb_t1:gramm1 -0.015741   0.027573  -0.571 #<6>

Correlation of Fixed Effects:  #<7>
            (Intr) vrb_t1 gramm1
verb_t1      0.000
gramm1       0.000 -0.002
vrb_t1:grm1  0.000  0.002  0.000
```

① Our fixed effects:
② Estimate (coefficient), standard error, t-value (no p-value…)
③ Intercept (grand mean)
④ Effect of tense
⑤ Effect of grammaticality
⑥ Interaction Effect
⑦ Correlation matrix of fixed effects

Random effects

```
Random effects: #<1>
 Groups   Name        Variance Std.Dev. #<2>
 sj       (Intercept) 0.06573  0.2564   #<3>
 Residual             0.18030  0.4246 #<4>
Number of obs: 3795, groups:  sj, 60 #<5>
```

1. Random effects
2. Grouping factor, effect name, overall variance and standard deviation
3. By-participant random intercepts
4. Residual error not accounted for by our random effects
5. Number of observations and grouping factor levels

Interpreting random effects

- we can also selectively print our random effects (variance components) using the `VarCorr()` function from `lme4`
  - only gives us the standard deviation, which is the square root of the variance

```
VarCorr(fit_lmm_fp_sj)
```

```
Groups   Name        Std.Dev.
sj       (Intercept) 0.25638
Residual             0.42462
```

- to also get the variance

```
print(VarCorr(fit_lmm_fp_sj),
      comp = c("Variance", "Std.Dev"))
```

```
Groups   Name        Variance Std.Dev.
sj       (Intercept) 0.065731 0.25638
Residual             0.180305 0.42462
```

- if we compute the mean, variance, and SD of the by-participant intercepts we get

```
coef(fit_lmm_fp_sj) |>
  pluck("sj") |>
as_tibble() |> rownames_to_column(var = "sj") |>
  rename(
```

```
      intercept = 2
    ) |>
    summarise(
      mean = mean(intercept),
      var = var(intercept),
      sd = sd(intercept)
    )
```

```
# A tibble: 1 x 3
   mean    var    sd
  <dbl>  <dbl> <dbl>
1  5.96 0.0630 0.251
```

68-95% rule

- in a normal distribution, 68% of the data will lie within +/-1 SD of the mean, and 95% will lie within +/-2 SDs of the mean
- our model intercept is 5.957102, so 68% of our participant intercepts will lie roughly between 5.7007211 and 6.2134829

```
 Groups   Name        Variance Std.Dev.
 sj       (Intercept) 0.065731 0.25638
 Residual             0.180305 0.42462
```

- so we can interpret the standard deviation as quantifying the variability of the by-participant intercepts around the mean (i.e., intercept)

`lmerTest::lmer()`

- recall we had a conflict for the function `lmer()` between `lme4` and `lmerTest`, and we set `lme4` to be our default for this function
- let's refit our model using `lmerTest::lmer()`

```
fit_lmm_fp_sj <-
  lmerTest::lmer(log(fp) ~ verb_t*gramm +
          (1|sj),
        data = df_biondo,
        subset = roi == 4)
```

- the only change is the addition of `lmerTest::`, which specifies which package to retreive the `lmer()` function from

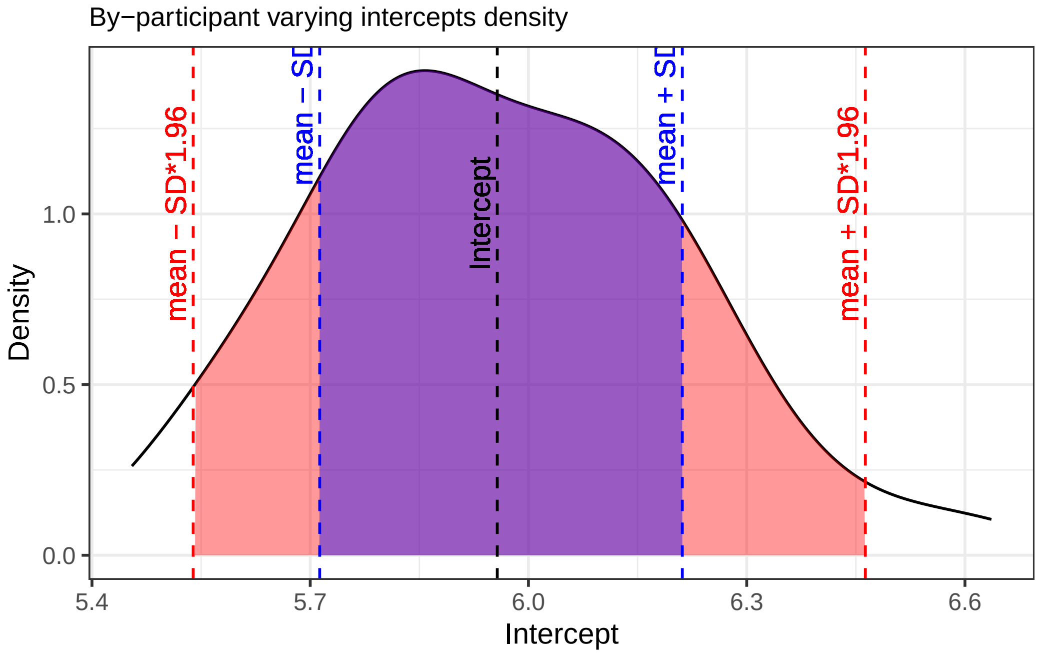


Figure 5: Density plot of by-participant intercepts with 95% (+/-SD*1.96) and 68% (+/-SD) ranges

- what's different in the summary?

```
summary(fit_lmm_fp_sj)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: log(fp) ~ verb_t * gramm + (1 | sj)
   Data: df_biondo
 Subset: roi == 4

REML criterion at convergence: 4479.1

Scaled residuals:
    Min      1Q  Median      3Q     Max
-4.0560 -0.6427 -0.0419  0.6168  4.0901

Random effects:
 Groups   Name        Variance Std.Dev.
```

```
 sj       (Intercept) 0.06573  0.2564
 Residual             0.18030  0.4246
Number of obs: 3795, groups:  sj, 60

Fixed effects:
                 Estimate  Std. Error          df t value             Pr(>|t|)
(Intercept)      5.957102    0.033809   58.991899 176.199 < 0.0000000000000002
verb_t1          0.062209    0.013787 3732.065822   4.512           0.00000662
gramm1           0.003466    0.013787 3732.032139   0.251                0.802
verb_t1:gramm1  -0.015741    0.027573 3732.037124  -0.571                0.568

(Intercept)    ***
verb_t1        ***
gramm1
verb_t1:gramm1
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) vrb_t1 gramm1
verb_t1      0.000
gramm1       0.000 -0.002
vrb_t1:grm1  0.000  0.002  0.000
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method ['lmerModLmerTest'] #<1
...

Fixed effects:
                 Estimate  Std. Error          df t value             Pr(>|t|)     #<2>
(Intercept)      5.957102    0.033809   58.991899 176.199 < 0.0000000000000002 ***
verb_t1          0.062209    0.013787 3732.065822   4.512           0.00000662 ***
gramm1           0.003466    0.013787 3732.032139   0.251                0.802
verb_t1:gramm1  -0.015741    0.027573 3732.037124  -0.571                0.568

...
```

① New: `Satterthwaite's method` and object class (`lmerModLmerTest`)
② Fixed effects include `df` (degrees of freedom) and *p*-values (`Pr(>|t|)`)

- `lme4::lmer()` doesn't provide degrees of freedom or *p*-values

Table 1: Fixed effects for `lm_fp_sj`

term	estimate	std.error	statistic	p.valu
(Intercept)	5.957	0.008	741.568	0.00000
verb_t1	0.061	0.016	3.809	0.00014
gramm1	0.003	0.016	0.193	0.84695
verb_t1:gramm1	-0.015	0.032	-0.474	0.63521

  - defining degrees of freedom (and therefore calculating p-values) is more complex and not trivial in mixed models
  - `lmerTest` uses the Satterthwaite method, which is fine for our purposes

- importantly, everything else is exactly the same as when we use `lme4::lmer()`

Fixed effects for `lm()` and `lmer()`

- let's compare our fixed effects to those from a model without random effects

```
fit_lm_fp <-
  lm(log(fp) ~ verb_t*gramm,
       data = df_biondo,
       subset = roi == 4)
```

Comparing fixed effects

- so far we see that our model estimates are still descriptively similar, there are only some slight quantitative differences
- your fixed effects will typically be unchanged with the addition of random effects
  - what changes will be usually be the standard error, t-value (or z-value for generalised linear (mixed) models), confidence intervals, and p-values
  - the magnitude of this change will depend on whether the inclusion of the random effects better accounts for variability in your data than your fixed effects alone

Table 2: Fixed effects for `lmm_fp_sj`

term	estimate	std.error	statistic	df	
(Intercept)	5.957	0.034	176.199	59	0.[illegible]
verb_t1	0.062	0.014	4.512	3732	0.[illegible]
gramm1	0.003	0.014	0.251	3732	0.[illegible]
verb_t1:gramm1	-0.016	0.028	-0.571	3732	0.[illegible]

Comparing residual error

- the residual error for our fixed-effects-only model was is 0.49

```
glance(fit_lm_fp)$sigma
```

```
[1] 0.494879
```

- for our by-participant varying intercepts model it goes down to 0.42

```
glance(fit_lmm_fp_sj)$sigma
```

```
[1] 0.424623
```

- this tells us that our inclusion of by-participant varying intercepts accounts for some of the variance in the model that was not accounted for in fixed-effects-only model
- are there any other possible sources of variance that we haven't taken into account?

Crossed random effects: two grouping factors

- we still haven't taken by-item variance into account, let's now include *crossed* random effects in our model

```
fit_lmm_fp_sj_item <-
  lmerTest::lmer(log(fp) ~ verb_t*gramm +
                  (1|sj) +
```

```r
                    (1|item),
                  data = df_biondo,
                  subset = roi == 4)
```

- crossed random effects refer to a property of your data (/experimental design), i.e., repeated measures for items *and* participants
  - one level of a grouping factor contains observations from all levels from another grouping factor (e.g., each item has an observations from each participant and vice versa)

```r
summary(fit_lmm_fp_sj_item)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: log(fp) ~ verb_t * gramm + (1 | sj) + (1 | item)
   Data: df_biondo
 Subset: roi == 4

REML criterion at convergence: 4220.3

Scaled residuals:
    Min      1Q  Median      3Q     Max 
-4.1568 -0.6169 -0.0257  0.6006  4.0422 

Random effects:
 Groups   Name        Variance Std.Dev.
 item     (Intercept) 0.01940  0.1393  
 sj       (Intercept) 0.06654  0.2580  
 Residual             0.16089  0.4011  
Number of obs: 3795, groups:  item, 96; sj, 60

Fixed effects:
                 Estimate  Std. Error          df t value              Pr(>|t|)
(Intercept)      5.956404    0.036790   79.200811 161.903 < 0.0000000000000002
verb_t1          0.061892    0.013025 3637.133151   4.752            0.00000209
gramm1           0.003212    0.013025 3637.183379   0.247                 0.805
verb_t1:gramm1  -0.014316    0.026049 3637.102346  -0.550                 0.583
```

Table 3: By-participant varying variance component

group	term	estimate
sj	sd___(Intercept)	0.2563809
Residual	sd___Observation	0.4246230

Table 4: By-participant and -item variance components

group	term	estimate
item	sd___(Intercept)	0.1392928
sj	sd___(Intercept)	0.2579514
Residual	sd___Observation	0.4011073

```
(Intercept)    ***
verb_t1        ***
gramm1
verb_t1:gramm1
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) vrb_t1 gramm1
verb_t1      0.000
gramm1       0.000 -0.002
vrb_t1:grm1  0.000  0.002  0.000
```

Comparing random effects

- we now have the variance of by-item random intercepts (Table 4)
- the *variance component* for the by-subjects intercepts is not much changed
  - the value is slightly different because it now also takes by-item variability into account
- the *residual error* also goes down in `fit_lmm_fp_sj_item` (0.4), because by-item intercepts account for some of the variance that was unaccounted for in `fit_lmm_fp_sj` (0.42)

Table 5: Fixed effects for `lmm_fp_sj`

term	estimate	std.error	statistic	df	
(Intercept)	5.957	0.034	176.199	59	0.
verb_t1	0.062	0.014	4.512	3732	0.
gramm1	0.003	0.014	0.251	3732	0.
verb_t1:gramm1	-0.016	0.028	-0.571	3732	0.

Table 6: Fixed effects for `lmm_fp_sj_item`

term	estimate	std.error	statistic	df	
(Intercept)	5.956	0.037	161.903	79	0.
verb_t1	0.062	0.013	4.752	3637	0.
gramm1	0.003	0.013	0.247	3637	0.
verb_t1:gramm1	-0.014	0.026	-0.550	3637	0.

- note that there is most by-participant than by-item variance, this is typical and reflects what we saw in our boxplots

Comparing fixed effects

- again we see there isn't much change to our coefficient estimates

Comparing predictions

```
sjPlot::plot_model(fit_lm_fp, type = "int") +
  geom_line(position = position_dodge(.1)) +
  ylim(340,440) +
  labs(title = "fit_lm_fp") +

sjPlot::plot_model(fit_lmm_fp_sj, type = "int") +
```

```
  geom_line(position = position_dodge(.1)) +
  ylim(340,440) +
  labs(title = "fit_lmm_fp_sj") +

sjPlot::plot_model(fit_lmm_fp_sj_item, type = "int") +
  geom_line(position = position_dodge(.1)) +
  ylim(340,440) +
  labs(title = "fit_lmm_fp_sj_item") +

  plot_layout(guides = "collect")
```

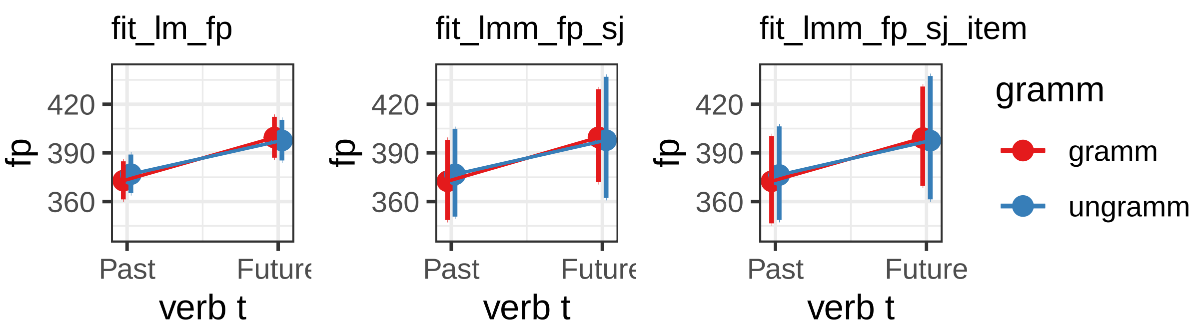


Figure 6: Comparison of predicted estimates and 95% confidence intervals for the three models

- if we plot the results from all three models we've fit so far we see the estimates are similar but the confidence intervals are wider for the mixed models
  - this is despite the fact that the p-values are significant for all three

Model comparison

- we can use the `anova()` function to compare model fit

```
anova(fit_lmm_fp_sj, fit_lmm_fp_sj_item)
```

```
Data: df_biondo
Subset: roi == 4
Models:
fit_lmm_fp_sj: log(fp) ~ verb_t * gramm + (1 | sj)
fit_lmm_fp_sj_item: log(fp) ~ verb_t * gramm + (1 | sj) + (1 | item)
                   npar    AIC    BIC  logLik deviance  Chisq Df
fit_lmm_fp_sj         6 4467.3 4504.8 -2227.7   4455.3
```

```
fit_lmm_fp_sj_item    7 4210.3 4254.0 -2098.2   4196.3 258.98  1
                               Pr(>Chisq)
fit_lmm_fp_sj
fit_lmm_fp_sj_item < 0.00000000000000022 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- here we see that the AIC, BIC, and logLik are all lower for our model with by-participant and -item varying intercepts
  - *lower* AIC and BIC indicate better model fit
  - *higher* logLik indicates better fit
- the inclusion of by-item random intercepts significantly improves the fit of our model

Exploring our random effects estimates

- what we saw in our model summary were the variance components
  - a description of the variance of our by-item and by-participant random intercepts
- our model also contains intercept estimates for each level of item and participant
  - we can extract the intercept estimates
  - or we extract their deviance from the model intercept

Extracting fixed effects

- we've already used `coef()` to extract fixed effect estimates from `lm` objects

```
coef(fit_lm_fp)
```

```
 (Intercept)       verb_t1        gramm1 verb_t1:gramm1
 5.957251870    0.061204153    0.003101061   -0.015245374
```

- to extract our fixed effect estimates from `lmer` objects we need `fixef()`

```
fixef(fit_lmm_fp_sj_item)
```

```
 (Intercept)       verb_t1        gramm1 verb_t1:gramm1
  5.95640363     0.06189237     0.00321152    -0.01431578
```

- or we can append `$coefficients` to the model summary

```r
summary(fit_lmm_fp_sj_item)$coefficients |>
  as_tibble()
```

```
# A tibble: 4 x 5
  Estimate `Std. Error`     df `t value` `Pr(>|t|)`
     <dbl>        <dbl>  <dbl>     <dbl>      <dbl>
1  5.96          0.0368   79.2   162.      1.31e-101
2  0.0619        0.0130 3637.      4.75    2.09e-  6
3  0.00321       0.0130 3637.      0.247   8.05e-  1
4 -0.0143        0.0260 3637.     -0.550   5.83e-  1
```

Extract random intercept estimates

- `coef()` behaves very differently with `lmer` objects, extracting the random effects estimates per level

```r
coef(fit_lmm_fp_sj_item) |> pluck("item") |>
  rownames_to_column(var = "item") |> head()
```

```
  item (Intercept)    verb_t1      gramm1 verb_t1:gramm1
1    1    6.022184 0.06189237 0.00321152    -0.01431578
2    2    5.761268 0.06189237 0.00321152    -0.01431578
3    3    5.854873 0.06189237 0.00321152    -0.01431578
4    4    6.056862 0.06189237 0.00321152    -0.01431578
5    5    6.138213 0.06189237 0.00321152    -0.01431578
6    6    6.331058 0.06189237 0.00321152    -0.01431578
```

- which outputs a `list` object, with one data frame for `item` and one for `sj`
  - in the code above I've 'plucked' just the by-item coefficients

- we can extract just one or the other (`head()` is for presentation purposes):

```r
coef(fit_lmm_fp_sj_item) |> pluck("item") |>
  rownames_to_column(var = "item") |> head()
```

```
  item (Intercept)    verb_t1     gramm1 verb_t1:gramm1
1    1    6.022184 0.06189237 0.00321152    -0.01431578
2    2    5.761268 0.06189237 0.00321152    -0.01431578
3    3    5.854873 0.06189237 0.00321152    -0.01431578
4    4    6.056862 0.06189237 0.00321152    -0.01431578
5    5    6.138213 0.06189237 0.00321152    -0.01431578
6    6    6.331058 0.06189237 0.00321152    -0.01431578
```

```r
coef(fit_lmm_fp_sj_item) |> pluck("sj") |>
  rownames_to_column(var = "sj") |> head()
```

```
  sj (Intercept)    verb_t1     gramm1 verb_t1:gramm1
1  1    6.401777 0.06189237 0.00321152    -0.01431578
2  2    5.794179 0.06189237 0.00321152    -0.01431578
3 07    5.869627 0.06189237 0.00321152    -0.01431578
4 09    5.782527 0.06189237 0.00321152    -0.01431578
5 10    6.621081 0.06189237 0.00321152    -0.01431578
6 11    5.913712 0.06189237 0.00321152    -0.01431578
```

- why do our intercepts vary, but not `verb_t1`, `gramm1`, or `verb_t1:gramm1`?

Extract deviations from the intercept

- the `ranef()` function provides the deviance from the model intercept and each random intercept estimate
  - the output is a `list` with a one element per grouping factor

```r
ranef(fit_lmm_fp_sj_item)
```

```
$item
     (Intercept)
1    0.065780608
2   -0.195135717
3   -0.101530802
4    0.100458122
5    0.181809783
6    0.374654251
7    0.092819196
8    0.136954752
```

9	0.058102873
10	-0.054265683
11	-0.149873360
12	0.110751479
13	0.147096084
14	0.127958914
15	0.057606192
16	-0.081076541
17	0.125828603
18	-0.073509315
19	-0.012746330
20	0.110139903
21	-0.155506252
22	0.126398878
23	0.166876070
24	-0.034901551
25	0.146486591
26	0.074730265
27	0.088381259
28	-0.092678849
29	0.014411435
30	0.066763872
31	-0.038994027
32	-0.120446386
33	-0.194224589
34	-0.072322285
35	-0.084322521
36	-0.103494886
38	0.118984111
39	0.091933443
40	-0.086216865
41	0.134573606
42	-0.117043412
43	-0.062584500
44	-0.001550553
46	0.008950192
47	-0.099435593
48	-0.107745509
49	-0.062632428
51	-0.023401507
52	0.206710720
53	-0.016527981
54	-0.032560697

55	-0.023730903
56	0.046093704
57	0.217142741
58	0.023060958
59	-0.217157054
60	0.059161953
61	0.148130135
62	0.076164596
63	-0.162879330
64	0.003286442
66	-0.145269266
67	0.038320168
68	0.144997093
69	-0.011825617
70	0.141234735
72	-0.019280976
73	0.243151378
74	0.005062608
75	-0.078003084
76	0.030885201
77	-0.105347877
78	0.267962295
79	0.049474689
80	0.012452861
81	-0.126197903
82	-0.270275547
83	-0.231166335
84	-0.148988678
85	-0.074619155
86	-0.316661058
87	-0.021441852
88	0.004467788
89	-0.068689368
90	0.168752654
91	-0.130311515
92	0.181189404
93	-0.113007279
94	-0.038028968
95	0.018891686
96	-0.010844951
97	-0.160693789
98	-0.189160313
99	-0.046282799

```
100  0.038520282
101  0.031027185

$sj
    (Intercept)
1    0.4453736686
2   -0.1622245920
07  -0.0867769204
09  -0.1738770087
10   0.6646773896
11  -0.0426912397
12   0.1966276739
14  -0.2534130428
15   0.0744365289
16   0.3102703155
17  -0.4416672585
18   0.3483163760
20  -0.0004747722
21  -0.3905578006
22   0.1588478248
23   0.2750736081
24  -0.0164601317
26   0.0545049227
27   0.1681915401
28   0.3829174064
29  -0.0724729416
30  -0.1307018626
31   0.5591486944
32  -0.2297591961
33  -0.1058829882
34  -0.1929520704
25   0.2651328954
73   0.1010490787
74   0.1222262133
35  -0.3466865197
36   0.2458666126
37  -0.2586775333
38  -0.1170898527
39  -0.0707477418
40  -0.2595291388
41  -0.1337344231
42   0.1285752340
43  -0.3904903423
```

```
46 -0.5196576089
47  0.1586324543
48  0.2627372846
49 -0.1194323811
50 -0.3470873902
51  0.2998361957
52  0.1196091237
53 -0.1241739663
54 -0.1457289205
55  0.1724245610
56 -0.1790151326
57 -0.2663456249
58  0.0161788674
59  0.0070594181
60  0.2180027256
61 -0.0234205903
62  0.2711544584
63 -0.2167262452
64 -0.3457197069
65  0.0165792343
66  0.0426515896
67  0.0780730481

with conditional variances for "item" "sj"
```

- `ranef()$grouping_factor` or `pluck("grouping_factor")` selects the relevant grouping factor

```r
ranef(fit_lmm_fp_sj_item)$sj |>
  head()
```

```
   (Intercept)
1   0.44537367
2  -0.16222459
07 -0.08677692
09 -0.17387701
10  0.66467739
11 -0.04269124
```

Table 7: Random intercept estimates versus deviance

sj	sj_est	sj_dev	est_minus_dev	model_intercept	e
1	6.402	0.445	5.956	5.956	
2	5.794	-0.162	5.956	5.956	
07	5.870	-0.087	5.956	5.956	
09	5.783	-0.174	5.956	5.956	
10	6.621	0.665	5.956	5.956	
11	5.914	-0.043	5.956	5.956	

```r
ranef(fit_lmm_fp_sj_item) |>
  pluck("sj") |> head()
```

```
   (Intercept)
1   0.44537367
2  -0.16222459
07 -0.08677692
09 -0.17387701
10  0.66467739
11 -0.04269124
```

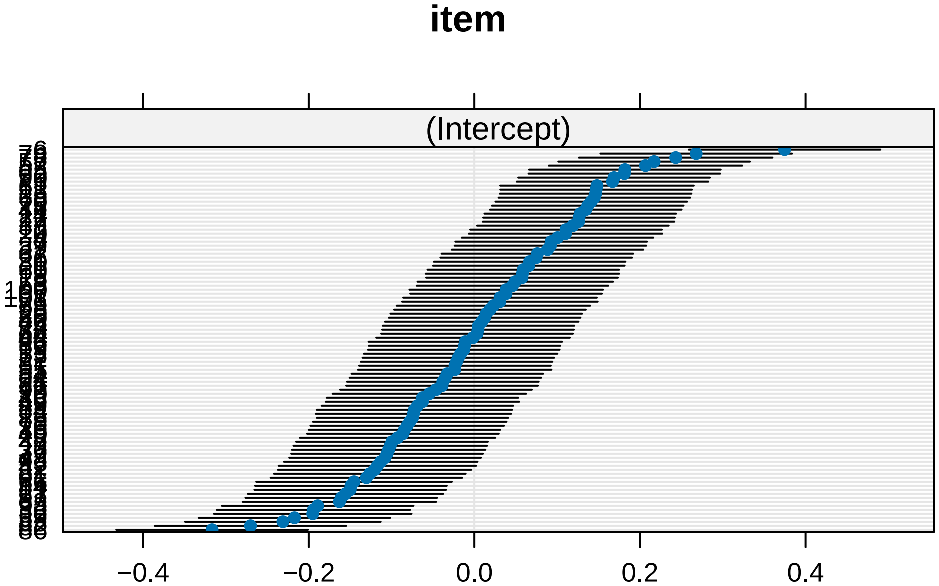
Compare estimates and deviances

- the values extracted by `ranef()` (`sj_dev` in Table 7) equal the difference (`difference`) between the model intercept (`model_intercept`) and the by-participant random intercept estimates (`sj_est`)
- so we can either look at each participant's (or item's) estimate, or look at how much it deviates from the model intercept

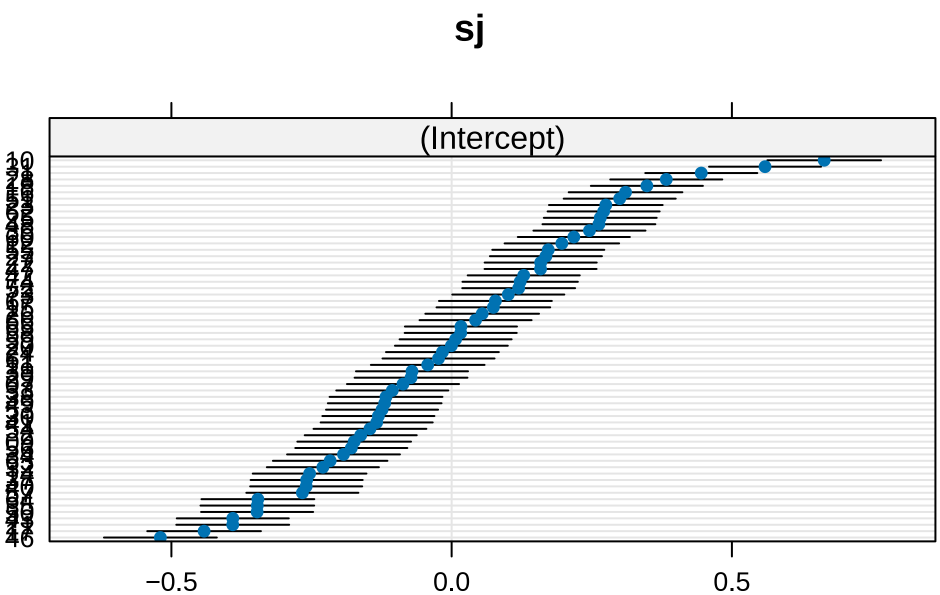
Visualise random effects

- the `lattice` package automatically produces plots of random effects estimates

```r
lattice::dotplot(ranef(fit_lmm_fp_sj_item))$item
```




```r
lattice::dotplot(ranef(fit_lmm_fp_sj_item))$sj
```



Reporting your model

- according to Sonderegger (2023) (p. 297), we should report:

  1. model definition (sometimes in 'Data Analysis' section)
  2. Fixed effects
  3. Random effects
  4. Sample size (number of observations, number of levels for each grouping factor)
  5. one or more quantitative summaries of the model, e.g., AIC, BIC, or logLik (although they're only informative in comparison to another model fit to the same data)

Model definition

> We conducted the analysis by fitting `linear mixed-effect models` to our data, using the R package lme4 (Bates et al., 2014). We included Time Reference (past, future), and Verb Match (match, mismatch) as `fixed-effect factors` [...] by adopting `sum contrast coding` (Schad et al., 2020): past and match conditions were `coded as -.5`. while future and mismatch conditions were `coded as .5`. [...] Moreover, we included `crossed random intercepts` and random slopes for all fixed-effect parameters `for subject and item` grouping factors (Barr et al., 2013) in all models. [...] Logit mixed-effect models were employed (Jaeger, 2008) for the analysis of the probability of regression measure. [...] P-values were derived by using the lmerTest package (Kuznetsova et al., 2017).

— Biondo et al. (2022), p. 9

- could also explicitly mention method used for *p*-values, an example:

> P-values for individual predictors were computed using `lmerTest`, with the `Satterthwaite` option for denominator degrees of freedom for F statistics.

— Troyer & Kutas (2020), p. 9

- but here they don't cite the package
  - so you see, there's alway something you miss…
- FYI, to get a package's citation, run `citation("lmerTest")` in the Console

Results

- a combination of tables, figures, and in-text coefficient estimates is always key
- in-text, the t- and p-values should be included at minimum, Estimate and standard error (Est = …, SE = …,) could also be included if you aren't reporting many effects but must at least be included in a table
- figures will typically only show the distribution of raw observations and model predictions for fixed effects

In-text

A main effect of tense was found in first-pass reading times at the verb region (Est = 0.062, t = 4.8, $p < .001$), with the future tense (M = 458ms, SD = 274ms) eliciting longer first-pass reading times than the past tense.

Tables

Fixed effects

```
tidy(fit_lmm_fp_sj_item,
     effects = "fixed") |>
  as_tibble() |>
  select(-effect) |>
  mutate(p.value = format_pval(p.value),
         across(c(estimate,std.error, statistic), round, 3),
         df = round(df,1)) |>
  mutate(term = fct_recode(term,
    "Intercept" = "(Intercept)",
    "Tense" = "verb_t1",
    "Grammticality" = "gramm1",
    "Tense x Gramm" = "verb_t1:gramm1"
  )) |>
  kable(
        col.names = c("Coefficient", "$\\hat{\\beta}$", "SE", "t", "df", "p")) |>
  kable_styling()
```

Random effects

Table 8: Table of fixed effects from fit_lmm_fp_sj_item

Coefficient	$\hat{\beta}$	SE	t	df	p
Intercept	5.956	0.037	161.903	79.2	$<$ .001
Tense	0.062	0.013	4.752	3637.1	$<$ .001
Grammticality	0.003	0.013	0.247	3637.2	0.805
Tense x Gramm	-0.014	0.026	-0.550	3637.1	0.583

Table 9: Table of random effects from fit_lmm_fp_sj_item

Group	Term	Variance	SD
item	(Intercept)	0.019	0.139
sj	(Intercept)	0.067	0.258
Residual	NA	0.161	0.401

```
as.data.frame(VarCorr(fit_lmm_fp_sj_item),comp=c("Variance","Std.Dev.")) |>
  as_tibble() |>
  select(-var2) |>
  # mutate(var1 = ifelse(var1 == "NA", " ", var1)) |>
  kable(digits = 3,
        col.names = c("Group", "Term", "Variance", "SD")) |>
  kable_styling()
```

Figures

- we don't usually include plots of our random effects in publications, but these can be useful for model exploration and can be included in supplementary materials

`lattice`

- as already mentioned, we can simply use the the `lattice` package

```
library(lattice)

dotplot(ranef(fit_lmm_fp_sj_item))["sj"]
```

```
$sj
```

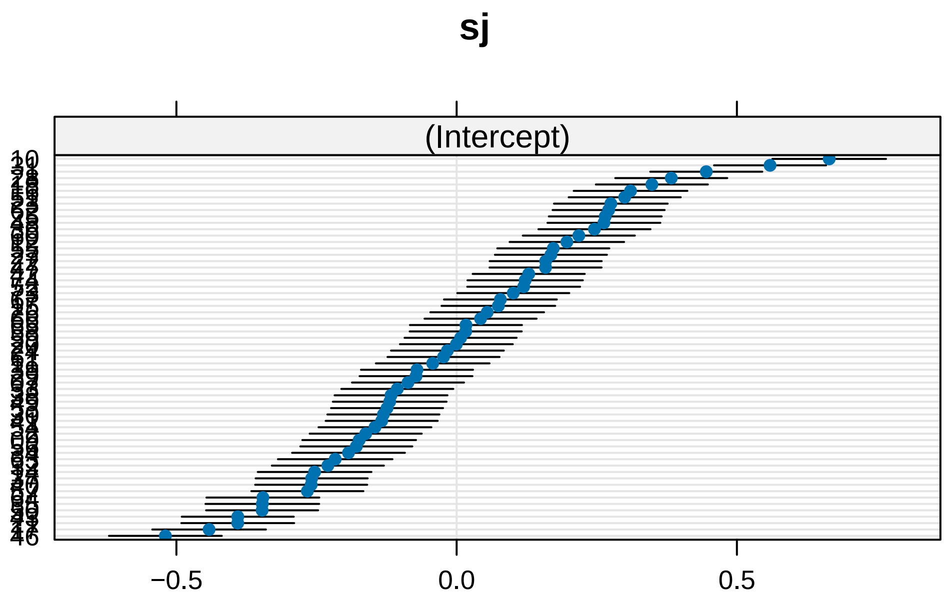


Figure 7: By-participant varying slopes (`lattice::dotplot(res(model))`)

`broom.mixed`

- or we can also generate the same plots using `tidy()` from the `broom.mixed` package + `ggplot()` (Figure 8 A)
- and we can add the model intercept to get each by-participant estimate, i.e., the values we get with `coef()` (Figure 8 B)

```
fig_res_dev <-
  broom.mixed::tidy(fit_lmm_fp_sj_item, effects = "ran_vals", conf.int = TRUE) |>
  filter(group == "sj") |>
  ggplot() +
  aes(x = estimate, y = reorder(level, estimate))  +
  labs(title = "By-participant varying intercepts with 95% CIs",
       y = "Participant ID",
       x = "Deviation from the intercept") +
  geom_vline(xintercept = 0, colour = "red", linetype = "dashed") +
  geom_point(colour = "blue") +
  geom_errorbar(
    aes(xmin = conf.low,
        xmax = conf.high)
  ) +
  scale_x_continuous(breaks = c(-0.5,0,0.5)) +
```

```
  facet_grid(~term)

fig_res_est <- broom.mixed::tidy(fit_lmm_fp_sj_item, effects = "ran_vals", conf.int = TRUE
  filter(group == "sj") |>
  # back-transform to ms
  mutate(across(c(estimate,conf.low,conf.high),~.+fixef(fit_lmm_fp_sj_item)[1])) |>
  # mutate(across(c(estimate,conf.low,conf.high),exp)) |>
  # plot
  ggplot() +
  aes(x = estimate, y = reorder(level, estimate))  +
  labs(title = "By-participant varying intercepts with 95% CIs",
       y = "Participant ID",
       x = "Estimate (log)") +
  geom_vline(xintercept = fixef(fit_lmm_fp_sj)[1], colour = "red", linetype = "dashed") +
  geom_point(colour = "blue") +
  geom_errorbar(
    aes(xmin = conf.low,
        xmax = conf.high)
  ) +
  scale_x_continuous(breaks = c(5.457102,5.957102 ,6.457102)) +
  facet_grid(~term)

fig_res_dev + fig_res_est +
  plot_annotation(tag_levels = "A")
```

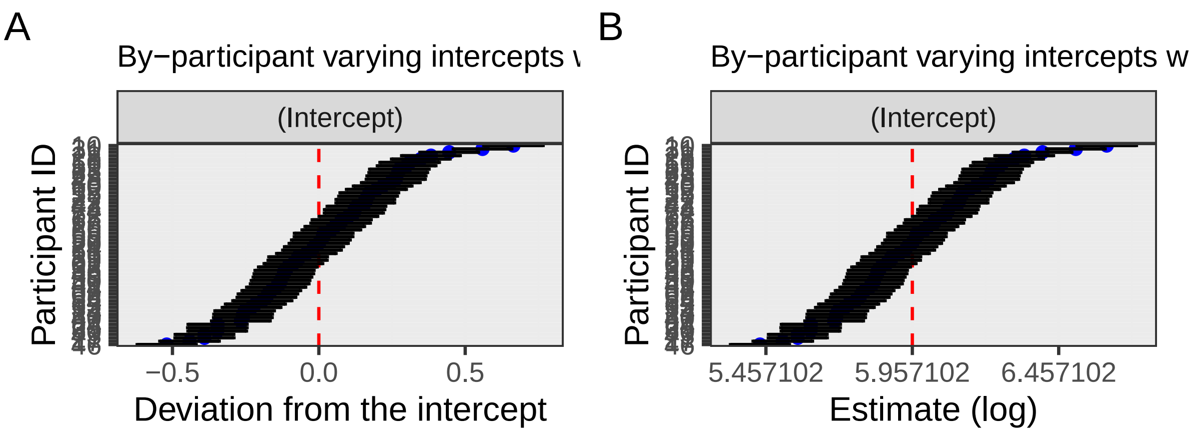


Figure 8: Back-transformed first-pass reading times (ms) at the verb region with 95% CIs

- and we can back-transform these values to milliseconds by exponentiating the estimates in the log scale (Figure 9 A)
- and we can back-transform deviances by subtracting the exponentiating model estimate from the back-transformed estimates (Figure 9 B)

```
fig_res_est_ms <-
  broom.mixed::tidy(fit_lmm_fp_sj_item, effects = "ran_vals", conf.int = TRUE) |>
  filter(group == "sj") |>
  # back-transform to ms
  mutate(across(c(estimate,conf.low,conf.high),~.+fixef(fit_lmm_fp_sj_item)[1])) |>
  mutate(across(c(estimate,conf.low,conf.high),exp)) |>
  # plot
  ggplot() +
  aes(x = estimate, y = reorder(level, estimate))  +
  labs(title = "By-participant varying intercepts with 95% CIs",
       y = "Participant ID",
       x = "Estimate (ms)") +
  geom_vline(xintercept = exp(fixef(fit_lmm_fp_sj)[1]), colour = "red", linetype = "dashed
  geom_point(colour = "blue") +
  geom_errorbar(
    aes(xmin = conf.low,
        xmax = conf.high)
  ) +
  scale_x_continuous(breaks = c(186.4884,386.4884, 586.4884, 786.4884)) +
  facet_grid(~term)

fig_res_dev_ms <-
broom.mixed::tidy(fit_lmm_fp_sj_item, effects = "ran_vals", conf.int = TRUE) |>
  filter(group == "sj") |>
  # back-transform to ms
  mutate(across(c(estimate,conf.low,conf.high),~.+fixef(fit_lmm_fp_sj_item)[1])) |>
  mutate(across(c(estimate,conf.low,conf.high),exp)) |>
  mutate(across(c(estimate,conf.low,conf.high),~.-exp(fixef(fit_lmm_fp_sj_item)[1]))) |>
  # plot
  ggplot() +
  aes(x = estimate, y = reorder(level, estimate))  +
  labs(title = "Deviances in by-participant varying intercepts with 95% CIs",
       y = "Participant ID",
       x = "Deviance (ms)") +
  geom_vline(xintercept = 0, colour = "red", linetype = "dashed") +
```

```
  geom_point(colour = "blue") +
  geom_errorbar(
    aes(xmin = conf.low,
        xmax = conf.high)
  ) +
  # scale_x_continuous(breaks = c(-0.5,0,0.5)) +
  facet_grid(~term)

fig_res_est_ms + fig_res_dev_ms + plot_annotation(tag_levels = "A")
```

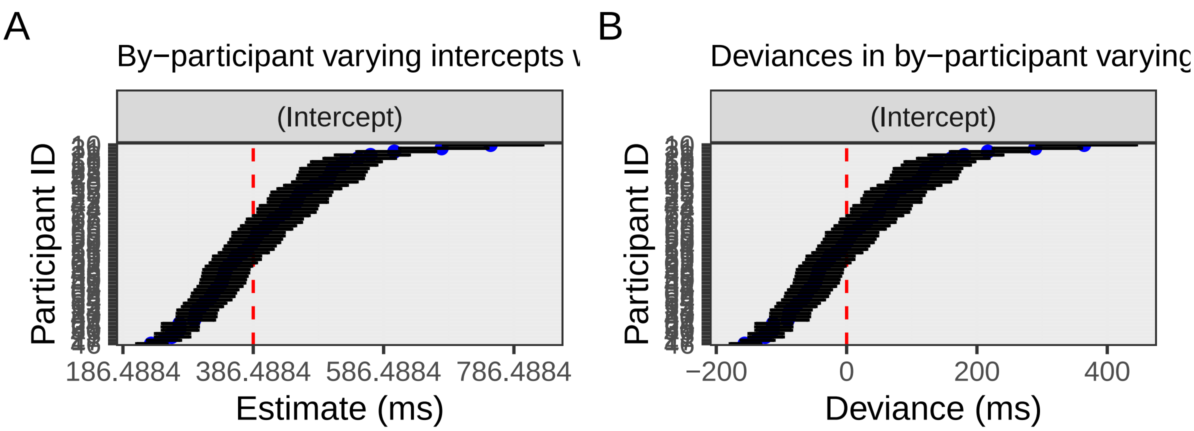


Figure 9: By-participant estimates back-transformed to milliseconds

Learning objectives

Today we learned…

- what linear mixed models are
- how to fit a random-intercepts model
- how to inspect and interpret a mixed effects model

Important terms

Term	Definition	Equation/Code
linear mixed (effects) model	NA	NA

Task

1. Fit a linear mixed model (`lm()` function) to log-transformed total reading times (`tt`) at the adverb region (`roi == 2`), with adverb time reference (`adv_t`) and gramm (`gramm`) and their interaction as fixed effects and by-participant and by-item varying intercepts. Use sum contrast coding (`Past` and `gramm` = `-0.5`, `Future` and `ungramm` = `+0.5`). Save this model as `fit_lmm_adv_tt`.

3. Inspect the fixed effect of your model.

5. Plot the fixed effects for `fit_lm_adv_tt` and `fit_lmm_adv_tt`.

```
coef_fixed <-
  broom.mixed::tidy(
  fit_lmm_adv_tt,
  effects="fixed",
  conf.int = T
)

pred_back <-
  tibble(
  tense = c(rep("Past",2),rep("Future",2)),
  gramm = rep(c("gramm","ungramm"),2)
  )
```

4. Inspect the random effects for `fit_lmm_adv_tt`. Describe what you see.
5. Plot the random effects per participant and item.
6. Write up a description of your model as if for a publication (model formula, contrasts, random effects structure, packages/methods used).
7. Write up the results (coefficient estimates, etc.).

References

Biondo, N., Soilemezidi, M., & Mancini, S. (2022). Yesterday is history, tomorrow is a mystery: An eye-tracking investigation of the processing of past and future time reference during sentence reading. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *48*(7), 1001–1018. https://doi.org/10.1037/xlm0001053

Kuznetsova, A., Brockhoff, P. B., & Christensen, R. H. B. (2017). lmerTest package: Tests in linear mixed effects models. *Journal of Statistical Software*, *82*(13), 1–26. https://doi.org/10.18637/jss.v082.i13

Sonderegger, M. (2023). *Regression Modeling for Linguistic Data*.

Troyer, M., & Kutas, M. (2020). To catch a Snitch: Brain potentials reveal variability in the functional organization of (fictional) world knowledge during reading. *Journal of Memory and Language*, *113*(August 2019), 104111. https://doi.org/10.1016/j.jml.2020.104111

Winter, B. (2014). *A very basic tutorial for performing linear mixed effects analyses (Tutorial 2)*.

Winter, B. (2019). Statistics for Linguists: An Introduction Using R. In *Statistics for Linguists: An Introduction Using R*. Routledge. https://doi.org/10.4324/9781315165547