

# 1 Understanding mixed effects models through data simulation

2 Lisa M. DeBruine<sup>1</sup> & Dale J. Barr<sup>1</sup>

3 <sup>1</sup> Institute of Neuroscience and Psychology, University of Glasgow

4 Preprint submitted for publication

## 5 Abstract

Experimental designs that sample both subjects and stimuli from a larger population need to account for random effects of both subjects and stimuli using mixed effects models. However, much of this research is analyzed using ANOVA on aggregated responses because researchers are not confident specifying and interpreting mixed effects models. The tutorial will explain how to simulate data with random effects structure and analyse the data using linear mixed effects regression (with the lme4 R package), with a focus on interpreting the output in light of the simulated parameters. Data simulation can not only enhance understanding of how these models work, but also enables researchers to perform power calculations for complex designs. All materials associated with this article can be accessed at <https://osf.io/3cz2e/>.



*Keywords:* simulation, mixed effects models, power, lme4, R

Word count: 5572

## 7 Background

8 In this article, we walk through the simulation and analysis of multilevel data with  
9 crossed random effects of subjects and stimuli. The article's target audience is researchers  
10 who work with experimental designs that sample subjects and stimuli, such as is the case  
11 for a large amount of experimental research in face perception, psycholinguistics, or social  
12 cognition. The tutorial assumes basic familiarity with R programming.

---

Lisa DeBruine  <https://orcid.org/0000-0002-7523-5539>  
Dale J. Barr  <https://orcid.org/0000-0002-1121-4608>

Correspondence concerning this article should be addressed to Lisa M. DeBruine, 62 Hillhead Street, Glasgow, G12 8QB. E-mail: [lisa.debruine@glasgow.ac.uk](mailto:lisa.debruine@glasgow.ac.uk)

## Generalizing to a population of encounters

Many research questions in psychology and neuroscience are questions about certain types of *events*: What happens when people encounter particular types of stimuli? For example: Do people recognize abstract words faster than concrete words? What impressions do people form about a target person’s personality based on their vocal qualities? Can people categorize emotional expressions more quickly on the faces of social ingroup members than on the faces of outgroup members? How do brains respond to threatening versus non-threatening stimuli? In all of these situations, researchers would like to be able to make general statements about phenomena that go beyond the particular participants and particular stimuli that they happen to have chosen for the specific study. Traditionally, people speak of such designs as having *crossed random factors* of participants and stimuli, and have discussed the problem as one of simultaneous generalization to both populations. However, it may be more intuitive to think of the problem as wanting to generalize to a single population of events: in particular, to a population of *encounters* between the units from the sampled populations (Barr, 2018).

Most analyses using conventional statistical techniques, such as analysis of variance and t-test, commit the fallacy of treating stimuli as fixed rather than random. The problem, and the solutions to the problem, have been known in psycholinguistics for over 50 years (Clark, 1973; Coleman, 1964), and most psycholinguistic journals require authors to demonstrate generality of findings over stimuli as well as over subjects. Even so, the quasi- $F$  statistics for ANOVA ( $F'$  and min- $F'$ ) that Clark proposed as a solution were widely recognized as unreasonably conservative (Forster & Dickinson, 1976), and until fairly recently, most psycholinguists performed separate by-subjects ( $F_1$ ) and by-items analyses ( $F_2$ ), declaring an effect “significant” only if it was significant for both analyses. The  $F_1 \times F_2$  approach was widely used, despite the fact that Clark had already shown it to be invalid, since both  $F$  statistics have higher than nominal false positives in the presence of a null effect,  $F_1$  due to unmodeled stimulus variance, and  $F_2$  due to unmodeled subject variance.

Recently, psycholinguists have adopted linear mixed-effects modeling as the standard for the statistical analysis, given numerous advantages over ANOVA, including the ability to simultaneously model subject and stimulus variation, to gracefully deal with missing data or unbalanced designs, and to accommodate arbitrary types of continuous and categorical predictors or response variables (Baayen, Davidson, & Bates, 2008; Locker, Hoffman, & Bovaird, 2007). This development has been facilitated by the `lme4` package for R (Bates, Mächler, Bolker, & Walker, 2015), which provides powerful functionality for model specification and estimation. With an appropriately specified model, mixed-effects models yield major improvements in power over quasi- $F$  approaches and avoid the increased false positive rate associated with separate  $F_1$  and  $F_2$  (Barr et al., 2013).

Despite mixed-effects modeling becoming the *de facto* standard for analysis in psycholinguistics, the approach has yet to take hold in other areas where stimuli are routinely sampled, even in spite of repeated calls for improved analyses in social psychology (Judd, Westfall, & Kenny, 2012) and neuroimaging (Bedny, Aguirre, & Thompson-Schill, 2007; Westfall, Nichols, & Yarkoni, 2016). One of the likely reasons for the limited uptake outside

of psycholinguistics is because mixed-effects models expose the analyst to a level of statistical and technical complexity far beyond most researchers' training. While some of this complexity is specific to mixed-effects modeling, some of it is simply hidden away from users of traditional techniques by GUIs and function defaults. The novice mixed modeler is suddenly confronted with the need to make decisions about how to specify categorical predictors, which random effects to include or exclude, which of the statistics in the voluminous output to attend to, and whether and how to re-configure the optimizer function when a convergence error or singularity warning appears.

We are optimistic that the increasing adoption of the mixed-effects approach will improve the generality and thus reproducibility of studies in psychology and related fields, but empathize with the frustration — and sometimes, exasperation — expressed by many novices when they attempt to grapple with these models in their research. Much of the uncertainty and unease around mixed-effects models comes from using them in situations where the ground truth is unknown. A profitable way to improve understanding and user confidence is through data simulation. Knowing the ground truth allows the user to experiment with various modeling choices and observe their impact on a model's performance.

#### Box 1. Glossary of terms

crossed random factors	Refers to a design with multiple random factors, such as subjects and items, the levels of which are crossed (e.g., each subject encounters each stimulus)
data-generating process (DGP)	The mathematical model capturing assumptions about the processes giving rise to the data
fixed effect	An effect whose value is constant across realizations of the experiment
random effect	An effect whose value varies across realizations of the experiment (e.g., due to sampling)
random intercept	A random effect capturing the deviation of a sampling unit (subject or item) from the model intercept
random slope	A random effect capturing the deviation of a sampling unit (subject or item) from the model slope
variance components	Parameters describing the distribution of random effects in the population

#### Data, materials, and online resources

The code to reproduce the analyses reported in this article has been made publicly available via the Open Science Framework and can be accessed at <https://osf.io/3cz2e/>. This code is supplemented by two web apps at [http://shiny.psy.gla.ac.uk/lmem\\_sim/](http://shiny.psy.gla.ac.uk/lmem_sim/) and <http://shiny.psy.gla.ac.uk/crossed> that perform data simulation without requiring knowledge of R code. These apps allow you to change parameters and inspect the results of LMEM and ANOVA analyses, as well as calculate power and false positive rates for these analyses.

## Simulating data with crossed random factors

To give an overview of the simulation task, we will simulate data from a design with crossed random factors of subjects and stimuli, fit a model to the simulated data, and then try to recover the parameter values we put in from the output. In this hypothetical study, subjects classify the emotional expressions of faces as quickly as possible, and we use their response time as the primary dependent variable. Let's imagine that the faces are of two intrinsic types: either from the subject's ingroup or from an outgroup. For simplicity, we further assume that each face appears only once in the stimulus set. The key question is whether there is any difference in classification speed across the type of face.

### Required software

The simulation will be presented in the R programming language (R Core Team, 2018). To run the code, you will need to have some add-on packages available. Any packages you are missing can be installed using R's `install.packages()` function, except for the development package `faux` (DeBruine, 2019) which, at the time of writing, must be installed from the development repository on github.

```
# load required packages
library("lme4")          # model specification / estimation
library("afex")          # anova and deriving p-values from lmer
library("broom.mixed")  # extracting data from model fits
library("faux")          # data simulation
# NOTE: to install the 'faux' package, use:
# devtools::install_github("debruine/faux")
library("tidyverse")     # data wrangling and visualisation
```

Because the code uses random number generation, if you want to reproduce the exact results below you will need to set the random number seed at the top of your script and ensure you are using R version 3.6.0 or higher. If you change the seed or are using a lower version of R, your exact numbers will differ, but the procedure will still produce a valid simulation.

```
# ensure this script returns the same results on each run
set.seed(8675309)
```

### Establishing the data-generating parameters

The first thing to do is to set up the parameters that govern the process we assume to give rise to the data, the *data-generating process* or DGP. In this hypothetical study, each of 100 subjects will respond to all 50 stimulus items (25 ingroup and 25 outgroup), for a total of 5000 observations.

**Specify the data structure.** We want the resulting data to be in long format, with the structure shown below, where each row is a single observation for each trial. The variables `subj_id` run from S001 to S100 and index the subject number; `item_id` runs from I01 to I50 and indexes the item number; `category` says whether the face is ingroup or outgroup, with items 1-25 always ingroup and items 26-50 always outgroup; and RT is the participant's response time for that trial. Note that a trial is uniquely identified by the combination of the `subj_id` and `item_id` labels.

Table 2  
*The target data structure.*

row	subj_id	item_id	category	RT
1	S001	I01	ingroup	750.2
2	S001	I02	ingroup	836.1
...	...	...	...	...
49	S001	I49	outgroup	811.9
50	S001	I50	outgroup	801.8
51	S002	I01	ingroup	806.7
52	S002	I02	ingroup	805.9
...	...	...	...	...
5000	S100	I50	outgroup	859.9

Note that for independent variables in designs where subjects and stimuli are crossed, you can't think of factors as being solely "within" or "between" because we have two sampling units; you must ask not only whether independent variables are within- or between- subjects, but also whether they are within- or between- stimulus items. Recall that a within-subjects factor is one where each and every subject receives all of the levels, and a between-subjects factors is one where each subject receives only one of the levels. Likewise, a within-items factor is one for which each stimulus receives all of the levels. For our current example, the ingroup/outgroup factor is within subjects but between items, given that each stimulus item is either ingroup or outgroup.

**Specify the fixed effects.** Getting an appropriately structured dataset is the easy part. The difficult part is randomly generating the RT values. For this, we need to establish an underlying statistical model. Let us start with a basic model and build up from there. We want a model of RT for subject  $s$  and item  $i$  that looks something like:

$$RT_{si} = \beta_0 + \beta_1 X_i + e_{si}. \quad (1)$$

According to the formula, response  $RT_{si}$  for subject  $s$  and item  $i$  is defined as sum of an intercept term  $\beta_0$ , which in this example is the grand mean reaction time for the population of stimuli, plus  $\beta_1$ , the mean RT difference between ingroup and outgroup stimuli, plus random noise  $e_{si}$ . To make  $\beta_0$  equal the grand mean and  $\beta_1$  equal the mean outgroup minus the mean ingroup RT, we will code the `category` variable as -.5 for the ingroup category and +.5 for the outgroup category.

Although this model is incomplete, we can go ahead and choose parameters for  $\beta_0$  and  $\beta_1$ . For this example, we set a grand mean of 800 ms and a mean difference of 50 ms. You will need to use disciplinary expertise and/or pilot data to choose these parameters; by the end of this tutorial you will understand how to extract those parameters from an analysis.

```
# set fixed effect parameters
b0 <- 800 # intercept; i.e., the grand mean
b1 <- 50 # slope; i.e, effect of category
```

The parameters  $\beta_0$  and  $\beta_1$  are *fixed effects*: they characterize properties of the population of encounters between subjects and stimuli. Thus, we set the mean RT for a “typical” subject encountering a “typical” stimulus to 800 ms, and assume that responses are typically 50 ms slower for outgroup than ingroup faces.

**Specify the random effects.** This model is completely unrealistic, however, because it doesn’t allow for any individual differences among subjects or stimuli. Subjects are not identical in their response characteristics: some will be faster than average, and some slower. We can characterize the difference from the grand mean for each subject  $s$  in terms of a *random effect*  $S_{0s}$ , where the first subscript, 0, indicates that the deflection goes with the intercept term,  $\beta_0$ . In other words, we assign each subject a unique *random intercept*. Likewise, it is unrealistic to assume that it is equally easy to categorize emotional expressions across all faces in the dataset; some will be easier than others. We incorporate this assumption by including by-item random intercepts  $I_{0i}$ , with the subscript 0 reminding us that it is a deflection from the  $\beta_0$  term, and the  $i$  indicating a unique deflection for each of the 50 faces. Adding these terms to our model yields:

$$RT_{si} = \beta_0 + S_{0s} + I_{0i} + \beta_1 X_i + e_{si} \quad (2)$$

Now, the actual values for  $S_{0s}$  and  $I_{0i}$  in our sampled dataset will depend on the luck of the draw, i.e., on which subject and which stimuli we happened to have sampled from their respective populations. Unlike fixed effects, we assume these values will differ across different realizations of the experiment. Although the individual values will differ, we assume a fixed standard deviation for the random intercepts in the subject population. Below we assign the by-subject offsets a standard deviation of 100 ms (`S0s_sd`), and the by-item offsets a standard deviation of 80 ms (`I0i_sd`). We will discuss below how you can estimate these parameters for your own designs.

```
# set random effect parameters
S0s_sd <- 100 # by-subject random intercept sd
I0i_sd <- 80 # by-item random intercept sd
```

There is still a deficiency in our data-generating model related to  $\beta_1$ , the fixed effect of category. Currently our model assumes that each and every subject is exactly 80 ms faster to categorize emotions on ingroup faces than on outgroup faces. Clearly, this assumption is totally unrealistic; some participants will be more sensitive to ingroup/outgroup differences than others. We can capture this in an analogous way to which we captured variation in the intercept, namely by including by-subject *random slopes*  $S_{1s}$ .

$$RT_{si} = \beta_0 + S_{0s} + I_{0i} + (\beta_1 + S_{1s}) X_i + e_{si} \quad (3)$$

163 The random slope  $S_{1s}$  is an estimate of how much faster or slower subject  $s$  is in  
 164 categorizing ingroup/outgroup faces than the population mean effect  $\beta_1$ , which we already  
 165 set to 50 ms. Given how we coded the  $X_i$  variable, the mean effect for subject  $s$  is given by  
 166 the  $\beta_1 + S_{1s}$  term. So, a participant who is 90 ms faster on average to categorize ingroup  
 167 faces would have a random slope  $S_{1s}$  of 40 ( $\beta_1 + S_{1s} = 50 + 40 = 90$ ). Likewise, a participant  
 168 who goes against the grain and shows an effect 15 ms *slower* than average (-15 ms) would  
 169 have a random slope  $S_{1s}$  of -65 ( $\beta_1 + S_{1s} = 50 + -65 = -15$ ). As we did for the random  
 170 intercepts, we assume that the  $S_{1s}$  effects are the result of sampling, and so will differ across  
 171 different realizations of the experiment. What remains fixed is their standard deviation  
 172 **S1s\_sd**, which we set to 40 ms.

173 But note that we are sampling *two* random effects for each subject  $s$ , a random  
 174 intercept  $S_{0s}$  and a random slope  $S_{1s}$ . It is possible for these values to be correlated, in  
 175 which case we should not sample them independently. For instance, perhaps people who are  
 176 faster than average overall (negative random intercept) also show a smaller than average of  
 177 the ingroup/outgroup manipulation (negative random slope) due to allocating less attention  
 178 to the task. We can capture this by allowing for a small correlation between the two factors,  
 179 **scor**, which we assign to be 0.2.

180 Finally, we need to characterize the trial-level noise in the study ( $e_{si}$ ) in terms of its  
 181 standard deviation. Here we simply assign this parameter value **err\_sd** to be twice the size  
 182 of the by-subject random intercept SD.

```
# set more random effect and error parameters
S1s_sd <- 40 # by-subject random slope sd
scor <- .2 # correlation between intercept and slope
err_sd <- 200 # residual (error) sd
```

183 To summarize, we established a reasonable statistical model underlying the data  
 184 having the form:

$$RT_{si} = \beta_0 + S_{0s} + I_{0i} + (\beta_1 + S_{1s}) X_i + e_{si} \quad (4)$$

185 The response time for subject  $s$  on item  $i$ ,  $RT_{si}$ , is decomposed into a population grand  
 186 mean  $\beta_0$ , a by-subject random intercept  $S_{0s}$ , a by-item random intercept  $I_{0i}$ , a fixed slope  
 187  $\beta_1$ , a by-subject random slope  $S_{1s}$ , and a trial-level residual  $e_{si}$ . Our data-generating process  
 188 is fully determined by seven parameters: two fixed effects (intercept **b0** and slope **b1**), four  
 189 variance parameters governing the random effects (**S0s\_sd**, **S1s\_sd**, **scor**, and **I0i\_sd**), and  
 190 one parameter governing the trial level variance (**err\_sd**).

```
# set all data-generating parameters
b0 <- 800 # intercept; i.e., the grand mean
b1 <- 50 # slope; i.e., effect of category
S0s_sd <- 100 # by-subject random intercept sd
```

```
I0i_sd <- 80 # by-item random intercept sd
S1s_sd <- 40 # by-subject random slope sd
scor <- .2 # correlation between intercept and slope
err_sd <- 200 # residual (error) sd
```

191 In the next section we will apply this data-generating process to simulate the sampling of  
 192 subjects, items, and trials (encounters).

### 193 Simulating the sampling process

194 Let's first define parameters related to the number of observations. In this example,  
 195 we will simulate data from 100 subjects responding to 25 ingroup faces and 25 outgroup  
 196 faces. There are no between-subject factors, so we can set `nsubj` to 100. We set `nitem` to a  
 197 named vector specifying the number of items in each between-item group.

```
# set number of subjects and items
nsubj <- 100 # number of subjects
nitem <- c(ingroup = 25, outgroup = 25) # number of items
```

198 **Simulate the sampling of stimulus items.** We need to create a table listing  
 199 each item, which category it is in, and simulated values for its random effects. We can do  
 200 this with the code below, setting item ID to the numbers 1 through the total number of  
 201 items, category for the first 25 items to “ingroup” and the next 25 faces to “outgroup”, and  
 202 sampling 50 numbers from a normal distribution with a mean of 0 and a standard deviation  
 203 of `I0i_sd`.

```
# simulate a sample of items
items <- data.frame(
  item_id = 1:sum(nitem),
  category = rep(c("ingroup", "outgroup"), nitem),
  I0i = rnorm(sum(nitem), 0, I0i_sd)
)
```

204 The function `faux::sim_design()` is a more flexible way to generate data with  
 205 specified parameters. This function will create a dataset with any number of between and/or  
 206 within factors, `n` items per between-cell, and the specified means (`mu`), standard deviations  
 207 (`sd`) and correlations (`r`). By default, it plots a schematic of the design you specified. See  
 208 the package vignette (DeBruine, 2019) for more details.

209 Category is a between-items factor, so we need to include it in the `between` argument.  
 210 Set `n = nitem` to specify the number of items per category. Set `sd = I0i_sd` to set the  
 211 standard deviation for the by-item random effects. Set `dv = "I0i"` to give the random  
 212 effect column that name. Set `id = "item_id"`; we'll use this later to join this information  
 213 to the table of trials.

```
# simulate a sample of items using sim_design()
items <- faux::sim_design(
```



```

between = list(category = c("ingroup", "outgroup")),
n = nitem,
sd = IOi_sd,
dv = "IOi",
id = "item_id"
)

```

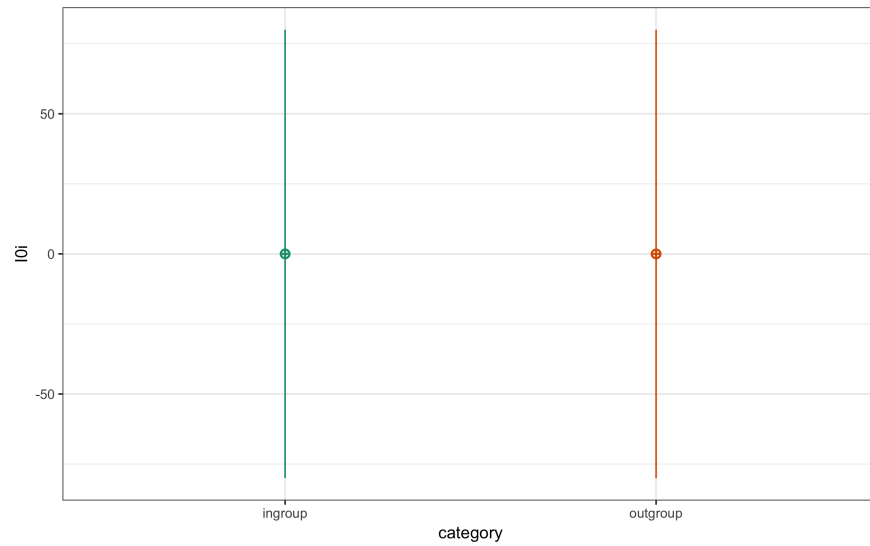


Figure 1. The specified distribution of random effects for ingroup and outgroup faces.

214 We will also introduce a numerical predictor to represent what category each stimulus  
 215 item  $i$  appears in (i.e., for the  $X_i$  in our model). Since we predict that responses to ingroup  
 216 faces will be faster than outgroup faces, we set ingroup to -0.5 and outgroup to +0.5. We will  
 217 later multiply this *effect coded* factor by the fixed effect of category ( $b_1 = 50$ ) to simulate  
 218 data where the ingroup faces are on average -25 ms different from the grand mean, while  
 219 the outgroup faces are on average 25 ms different from the grand mean.

```

# effect-code category
items$cat <- recode(items$category, "ingroup" = -0.5, "outgroup" = +0.5)

```

Table 3  
 The resulting table of item parameters.

item_id	category	IOi	cat
S01	ingroup	59.6	-0.5
S02	ingroup	-107.7	-0.5
S03	ingroup	26.4	-0.5
S04	ingroup	-1.0	-0.5
S05	ingroup	-37.1	-0.5
S06	ingroup	16.4	-0.5

220 **Simulate the sampling of subjects.** Now we will simulate the sampling of indi-  
 221 vidual subjects, resulting in a table listing each subject and their two correlated random  
 222 effects. We will again use `faux::sim_design()` for this task.

223 Set the `within` argument in `sim_design()` to a list with one factor (`effect`) that has  
 224 two levels: `S0s` and `S1s`. If you set a factor's levels as a named vector, the names (`S0s` and  
 225 `S1s`) become the column names in the data table and the values are used in plots created by  
 226 `faux`.

227 Set `n = nsubj` to specify the number of subjects. There are two random effects to  
 228 specify standard deviation for, so set `sd` using a named vector and set their correlation with  
 229 `r = scor`. Set `id = "subj_id"`; we'll use this later to join this information to the table of  
 230 trials.

```
# simulate a sample of subjects
subjects <- faux::sim_design(
  within = list(effect = c(S0s = "By-subject random intercepts",
                           S1s = "By-subject random slopes")),
  n = nsubj,
  sd = c(S0s_sd, S1s_sd),
  r = scor,
  id = "subj_id"
)
```

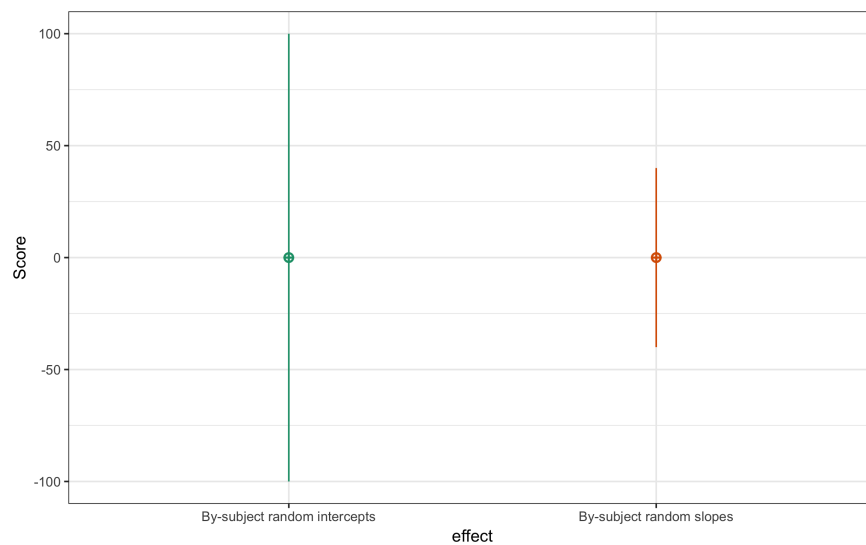


Figure 2. The specified distribution of random effects for subjects

Table 4  
The resulting table of subject parameters.

subj_id	S0s	S1s
S001	-98.77	-49.49
S002	37.33	23.77
S003	-127.42	39.56
S004	-56.37	10.13
S005	-39.73	-8.05
S006	43.70	37.66

231 **Simulate trials (encounters).** Since all subjects respond to all items, we can set  
 232 up a table of trials by making every possible combination of the subject and item IDs using  
 233 the function `crossing()` from `tidyr` (Wickham & Henry, 2019). Once we have a table of  
 234 all trials, we can join the information in this table to the information in our `subjects`  
 235 and `items` tables using `dplyr::inner_join()`. Each trial has random error associated; we  
 236 simulate this from a normal distribution with a mean of 0 and SD of `err_sd`.

```
# cross subject and item IDs; add an error term
trials <- crossing(subj_id = subjects$subj_id,
                  item_id = items$item_id) %>%
  mutate(err = rnorm(nrow(.), mean = 0, sd = err_sd))

# join subject and item tables
joined <- trials %>%
  inner_join(subjects, "subj_id") %>%
  inner_join(items, "item_id")
```

Table 5  
The resulting table of trials joined to the subject and item tables.

subj_id	item_id	err	S0s	S1s	category	I0i	cat
S001	S01	308.0	-98.8	-49.5	ingroup	59.6	-0.5
S001	S02	85.3	-98.8	-49.5	ingroup	-107.7	-0.5
S001	S03	-205.3	-98.8	-49.5	ingroup	26.4	-0.5
S001	S04	-138.1	-98.8	-49.5	ingroup	-1.0	-0.5
S001	S05	-190.8	-98.8	-49.5	ingroup	-37.1	-0.5
S001	S06	-351.1	-98.8	-49.5	ingroup	16.4	-0.5

237 **Calculate the response values.** Note how this resulting table contains the full  
 238 decomposition of effects that we need to compute the response according to the linear model  
 239 we defined above:

$$RT_{si} = \beta_0 + S_{0s} + I_{0i} + (\beta_1 + S_{1s}) X_i + e_{si} \quad (5)$$

240 Thus, we will calculate the response variable RT by adding together:

- 241 • the grand intercept (**b0**),
- 242 • each subject-specific random intercept (**S0s**),
- 243 • each item-specific random intercept (**I0i**),
- 244 • each sum of the category effect (**b1**) and the random slope (**S1s**), multiplied by the
- 245 numerical predictor (**cat**), and
- 246 • each residual error (**err**).

247 After this we will use `dplyr::select()` to keep the columns we need. Note that the  
 248 resulting table has the structure that we set as our goal at the start of this exercise, with  
 249 the additional column **cat**, which we will keep to use in the estimation process, described in  
 250 the next section.

```
# calculate the response variable
dat_sim <- joined %>%
  mutate(RT = b0 + I0i + S0s + (b1 + S1s) * cat + err) %>%
  select(subj_id, item_id, category, cat, RT)
```

Table 6  
*The final simulated dataset.*

subj_id	item_id	category	cat	RT
S001	S01	ingroup	-0.5	1,068.5
S001	S02	ingroup	-0.5	678.6
S001	S03	ingroup	-0.5	522.1
S001	S04	ingroup	-0.5	561.9
S001	S05	ingroup	-0.5	473.1
S001	S06	ingroup	-0.5	366.2

251 **Data simulation function.** To make it easier to try out different parameters or  
 252 to generate many datasets for the purpose of power analysis, you can put all of the code  
 253 above into a custom function. Set up the function to takes all of the parameters we set  
 254 above as arguments. We'll set the defaults to the values we used, but you can choose your  
 255 own defaults. The code below is just all of the code above, condensed a bit. It returns one  
 256 dataset with the parameters you specified.

```
# set up the custom data simulation function
my_sim_data <- function(
  nsubj = 100, # number of subjects
  nitem = c(ingroup = 25, outgroup = 25), # number of items
  b0 = 800, # grand mean
  b1 = 50, # effect of category
  I0i_sd = 80, # by-item random intercept sd
  S0s_sd = 100, # by-subject random intercept sd
  S1s_sd = 40, # by-subject random slope sd
  scor = 0.2, # correlation between intercept and slope
```

```

err_sd = 200){ # residual (standard deviation)

# simulate items
items <- faux::sim_design(
  between = list(category = c("ingroup", "outgroup")),
  n = nitem,
  sd = IOi_sd,
  dv = "IOi",
  id = "item_id",
  plot = FALSE
)

# effect code category
items$cat <- recode(items$category, "ingroup" = -0.5, "outgroup" = 0.5)

# simulate subjects
subjects <- faux::sim_design(
  within = list(effect = c(S0s = "By-subject random intercepts",
                           S1s = "By-subject random slopes")),
  n = nsubj,
  sd = c(S0s_sd, S1s_sd),
  r = scor,
  id = "subj_id",
  plot = FALSE
)

# simulate trials
dat_sim <- crossing(subj_id = subjects$subj_id,
                    item_id = items$item_id) %>%
  inner_join(subjects, "subj_id") %>%
  inner_join(items, "item_id") %>%
  mutate(err = rnorm(nrow(.), mean = 0, sd = err_sd)) %>%
  mutate(RT = b0 + IOi + S0s + (b1 + S1s) * cat + err) %>%
  select(subj_id, item_id, category, cat, RT)

dat_sim
}

```

257        Now you can generate a dataset with the default parameters using `my_sim_data()` or,  
 258        for example, a dataset with 500 subjects and no effect of category using `my_sim_data(nsubj`  
 259        `= 50, b1 = 0)`.

## Analyzing the simulated data

### Setting up the formula

Now we're ready to analyse our simulated data. The first argument to `lmer()` is a model formula that defines the structure of the linear model. The formula for our design maps onto how we calculated the response above.

```
RT ~ 1 + cat + (1 | item_id) + (1 + cat | subj_id)
```

- RT is the response
- 1 corresponds to the grand intercept ( $b_0$ ),
- cat corresponds to the effect of category ( $b_1 * cat$ ),
- (1 | item\_id) corresponds to the item-specific random intercept ( $I0i$ ),
- (1 + cat | subj\_id) corresponds to the subject-specific random intercept ( $S0s$ ) plus the subject-specific random slope of category ( $S1s$ ),
- the error term is automatically included in all models, so is left implicit

The “fixed” part of the formula,  $RT \sim 1 + cat$ , establishes the  $RT_{si} + \beta_0 + \beta_1 X_i + e_{si}$  part of our linear model, with the role of  $X_i$  being played by `cat`. Every model has an intercept ( $\beta_0$ ) term and residual term ( $e_{si}$ ) by default, so you could alternatively leave the 1 out and just write  $RT \sim cat$ .

The terms in parentheses with the `|` separator define the random effects structure. For each of these bracketed terms, the left-hand side of the `|` names the effects you wish to allow to vary and the right hand side names the variable identifying the levels of the random factor over which the terms vary (e.g., subjects or items). The first term, (1 | item\_id) allows the intercept (1) to vary over the random factor of items (item\_id). This is an instruction to estimate the parameter underlying the  $I0i$  values, namely  $I0i\_sd$ . The second term, (1 + cat | subj\_id), allows both the intercept and the effect of category (cat) to vary over the random factor of subjects (subj\_id). It is an instruction to estimate the three parameters that underlie the  $S0s$  and  $S1s$  values, namely  $S0s\_sd$ ,  $S1s\_sd$ , and `scor`.

### Interpreting the lmer summary

The other arguments to the `lme4` function are the name of the data frame where the values are found (`dat_sim`), and `REML = FALSE` which selects maximum-likelihood estimation, which is preferable to the default estimation technique when testing fixed effects. Use the `summary()` function to view the results.

```
# fit a linear mixed-effects model to data
mod_sim <- lmer(RT ~ 1 + cat + (1 | item_id) + (1 + cat | subj_id),
               data = dat_sim, REML = FALSE)

summary(mod_sim, corr = FALSE)
```

```

292 ## Linear mixed model fit by maximum likelihood . t-tests use
293 ##   Satterthwaite's method [lmerModLmerTest]
294 ## Formula: RT ~ 1 + cat + (1 | item_id) + (1 + cat | subj_id)
295 ##   Data: dat_sim
296 ##
297 ##           AIC          BIC    logLik deviance df.resid
298 ##   67720.6   67766.2 -33853.3   67706.6      4993
299 ##
300 ## Scaled residuals:
301 ##      Min       1Q   Median       3Q      Max
302 ## -3.7631 -0.6740 -0.0046  0.6778  3.6425
303 ##
304 ## Random effects:
305 ##   Groups      Name      Variance Std.Dev.  Corr
306 ##   subj_id (Intercept) 10334     101.66
307 ##           cat          2413      49.12   0.16
308 ##   item_id (Intercept)  4574      67.63
309 ##   Residual             40770     201.92
310 ## Number of obs: 5000, groups:  subj_id, 100; item_id, 50
311 ##
312 ## Fixed effects:
313 ##              Estimate Std. Error    df t value Pr(>|t|)
314 ## (Intercept)   816.05      14.25 126.62   57.28  <2e-16 ***
315 ## cat           52.42      20.56  55.35    2.55   0.0136 *
316 ## ---
317 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Let's break down the output step-by-step and try to find estimates of the seven parameters we used to generate the data: `b0`, `b1`, `S0s_sd`, `S1s_sd`, `scor`, `I0i_sd` and `err`. If you analyze existing data with a mixed effects model, you can use these estimates to help you set reasonable values for random effects in your own simulations.

After providing general information about the model fit, the output is divided into a **Random effects** and a **Fixed effects** section. The fixed effects section should be familiar from other types of linear models.

```

325 ## Fixed effects:
326 ##              Estimate Std. Error    df t value Pr(>|t|)
327 ## (Intercept)   816.05      14.25 126.62   57.28  <2e-16 ***
328 ## cat           52.42      20.56  55.35    2.55   0.0136 *

```

The **Estimate** column gives us parameter estimates for the fixed effects in the model, i.e.,  $\hat{\beta}_0$  and  $\hat{\beta}_1$ , which are estimated at about 816.05 and 52.42. The next columns give us the standard errors, estimated degrees of freedom (using the Satterthwaite approach),  $t$  value, and finally,  $p$  value.

The **Random effects** section is specific to mixed-effects models, and will be less

334 familiar to the reader.

335 **## Random effects:**

336 <b>##</b>	<b>Groups</b>	<b>Name</b>	<b>Variance</b>	<b>Std.Dev.</b>	<b>Corr</b>
337 <b>##</b>	<b>subj_id</b>	<b>(Intercept)</b>	10334	101.66	
338 <b>##</b>		<b>cat</b>	2413	49.12	0.16
339 <b>##</b>	<b>item_id</b>	<b>(Intercept)</b>	4574	67.63	
340 <b>##</b>	<b>Residual</b>		40770	201.92	

341 These are the estimates for the *variance components* in the model. To avoid confusion, it is  
 342 best to think of the information in this table as coming from three separate tables divided  
 343 up by the values in the **Groups** column.

344 The first subtable, where the value of **Groups** is **subj\_id**, gives the estimates for the  
 345 variance parameters defining the by-subject random effects.

346 <b>##</b>	<b>Groups</b>	<b>Name</b>	<b>Variance</b>	<b>Std.Dev.</b>	<b>Corr</b>
347 <b>##</b>	<b>subj_id</b>	<b>(Intercept)</b>	10334	101.66	
348 <b>##</b>		<b>cat</b>	2413	49.12	0.16

349 We have estimates for the variance of the intercept and slope (**cat**) in the **Variance** column,  
 350 which is just the square of the standard deviation in the **Std.Dev.** column. We obtain  
 351 estimates for **S0s\_sd** and **S1s\_sd** of 101.66 and 49.12 respectively. The **Corr.** column gives  
 352 us the estimated correlation between the by-subject random intercepts and slopes, estimated  
 353 here as 0.16.

354 The second subtable gives us the by-item random effect parameter estimates of which  
 355 there is only one, 67.63, corresponding to **I0i\_sd**. Again, the **Variance** column is just this  
 356 value squared.

357 <b>##</b>	<b>Groups</b>	<b>Name</b>	<b>Variance</b>	<b>Std.Dev.</b>	<b>Corr</b>
358 <b>##</b>	<b>item_id</b>	<b>(Intercept)</b>	4574	67.63	

359 The last subtable gives us the estimate of the residual term, 201.92.

360 <b>##</b>	<b>Groups</b>	<b>Name</b>	<b>Variance</b>	<b>Std.Dev.</b>	<b>Corr</b>
361 <b>##</b>	<b>Residual</b>		40770	201.92	

362 We have found all seven parameters in the output. Let's compare them to the values  
 363 that we put in.



Table 7

*The simulation parameters compared to the model estimations.*

variable	explanation	simulated value	estimated by model
b0	intercept (grand mean)	800.0	816.05
b1	fixed effect of category	50.0	52.42
S0s_sd	by-subject random intercept SD	100.0	101.66
S1s_sd	by-subject random slope SD	40.0	49.12
scor	cor between intercept and slope	0.2	0.16
I0i_sd	by-item random intercept SD	80.0	67.63
err_sd	residual (error) SD	200.0	201.92

You can also use `broom.mixed::tidy()` to output fixed and/or random effects in a tidy table. This is especially useful when you need to combine the output from hundreds of simulations to calculate power. The code below adds a column with the simulated parameters we set above so you can compare them to the estimated parameters from this simulated dataset.

```
# get a tidy table of results
broom.mixed::tidy(mod_sim) %>%
  mutate(sim = c(b0, b1, S0s_sd, S1s_sd, scor, I0i_sd, err_sd)) %>%
  select(1:3, 9, 4:8)
```

Table 8

*The output of the tidy function from broom.mixed.*

effect	group	term	sim	estimate	std.error	statistic	df	p.value
fixed	NA	(Intercept)	800.0	816.05	14.2	57.3	126.6	0.000
fixed	NA	cat	50.0	52.42	20.6	2.6	55.4	0.014
ran_pars	subj_id	sd__(Intercept)	100.0	101.66	NA	NA	NA	NA
ran_pars	subj_id	sd__cat	40.0	49.12	NA	NA	NA	NA
ran_pars	subj_id	cor__(Intercept).cat	0.2	0.16	NA	NA	NA	NA
ran_pars	item_id	sd__(Intercept)	80.0	67.63	NA	NA	NA	NA
ran_pars	Residual	sd__Observation	200.0	201.92	NA	NA	NA	NA

## Calculate Power

Data simulation is useful not only for illuminating modeling approaches, but also for calculating power when planning a study. The basic idea of a power simulation is to generate a large number of datasets encoding your assumptions about likely parameter values, fit models to each dataset, and then calculate the proportion of models that reject the null hypothesis as a measure of power.

First we create a function that analyses the simulated data and test it by running it once with default parameters.

```

# set up the power function
my_lmer_power <- function(...) {
  # ... is a shortcut that forwards any arguments to my_sim_data()
  dat_sim <- my_sim_data(...)
  mod_sim <- lmer(RT ~ cat + (1 | item_id) + (1 + cat | subj_id),
                 dat_sim, REML = FALSE)

  broom.mixed::tidy(mod_sim)
}

# run one model with default parameters
my_lmer_power()

```

Table 9

The output of `lmer_power()`.

effect	group	term	estimate	std.error	statistic	df	p.value
fixed	NA	(Intercept)	791.96	12.8	62.0	128.4	0.000
fixed	NA	cat	66.97	17.8	3.8	53.9	0.000
ran_pars	subj_id	sd__(Intercept)	93.52	NA	NA	NA	NA
ran_pars	subj_id	sd__cat	37.95	NA	NA	NA	NA
ran_pars	subj_id	cor__(Intercept).cat	0.10	NA	NA	NA	NA
ran_pars	item_id	sd__(Intercept)	58.13	NA	NA	NA	NA
ran_pars	Residual	sd__Observation	200.18	NA	NA	NA	NA

377 You can also change parameters. For example, what would happen if you increase the  
 378 number of items to 50 in each group and decrease the effect of category to 20 ms?

```

# run one model with new parameters
my_lmer_power(nitem = c(ingroup = 50, outgroup = 50), b1 = 20)

```

Table 10

The output of `lmer_power()` with 50 items per group and a category effect of 20 ms.

effect	group	term	estimate	std.error	statistic	df	p.value
fixed	NA	(Intercept)	830.71	11.7	70.9	183.9	0.000
fixed	NA	cat	19.58	16.6	1.2	125.8	0.241
ran_pars	item_id	sd__(Intercept)	74.90	NA	NA	NA	NA
ran_pars	subj_id	sd__(Intercept)	87.92	NA	NA	NA	NA
ran_pars	subj_id	sd__cat	59.95	NA	NA	NA	NA
ran_pars	subj_id	cor__(Intercept).cat	0.48	NA	NA	NA	NA
ran_pars	Residual	sd__Observation	198.86	NA	NA	NA	NA

379 You can use the `purrr::map_df` function to run the simulation repeatedly and save  
 380 the results to a data table. This will take a while, so test using just a few repetitions (**reps**)  
 381 first, then make sure you save the full results to a CSV file so you can set this code chunk to  
 382 not run (`eval = FALSE` in the chunk header) and load from the saved data for the rest of

383 your script in the future. You can use these data to calculate power for each fixed effect.

```
# run simulations and save to a file
reps <- 100
sims <- purrr::map_df(1:reps, ~my_lmer_power())
write_csv(sims, "sims/sims.csv")

# read saved simulation data
sims <- read_csv("sims/sims.csv", col_types = cols(
  # makes sure plots display in this order
  group = col_factor(ordered = TRUE),
  term = col_factor(ordered = TRUE)
))

# calculate mean estimates and power for specified alpha
alpha <- 0.05

sims %>%
  filter(effect == "fixed") %>%
  group_by(term) %>%
  summarise(
    mean_estimate = mean(estimate),
    mean_se = mean(std.error),
    power = mean(p.value < alpha)
  )
```

Table 11  
*Power calculation for fixed effects.*

term	Mean Estimate	Mean Std. Error	Power
(Intercept)	803.6	15.4	1.00
cat	47.6	23.6	0.53

384

## Conclusion

385 Mixed-effects modeling is a powerful technique for analyzing data from complex designs.  
 386 The technique is close to ideal for analyzing data with crossed random factors of subjects  
 387 and stimuli: it gracefully and simultaneously accounts for subject and item variance within  
 388 a single analysis, and outperforms traditional techniques in terms of type I error and power.  
 389 However, this additional power comes at the price of technical complexity. Through this  
 390 article, we have attempted to make mixed-effects models more approachable using data  
 391 simulation.

392 We considered only a simple, one-factor design. However, the general principles are  
 393 the same for higher-order designs. For instance, consider a 2x2 design, with factors *A* and *B*  
 394 both within subjects, but *A* within items and *B* between items. For such a design, you would  
 395 have four instead of two by-subject random effects: the intercept, main effect of *A*, main

effect of  $B$ , and the  $AB$  interaction. You would also need to specify correlations between each of these effects. You would also have two by-item random effects: one for the intercept and one for  $A$ . For further guidance and discussion on how to specify the random effects structure in complex designs, see (Barr, 2013).

We also have not said much in this tutorial about estimation issues, such as what to do when the fitting procedure fails to converge. Further guidance on this point can be found in (Barr et al., 2013), and by consulting the help materials in the `lme4` package (`?lme4::convergence`). We have also assumed that the random effects specification for the `lmer()` function should be based on the study design. However, we note that others have argued in favor of data-driven approaches for random effects specification (Matuschek, Kliegl, Vasishth, Baayen, & Bates, 2017).

In this tutorial, we have introduced the main concepts needed to get started with mixed effects models. Through data simulation of your own study designs, you can develop your understanding and perform power calculations to guide your sample size plans.

## Acknowledgements

**Author Contributions.** DJB drafted the substantive explanation and LDB drafted the tutorial. Both authors revised the draft and approved the final submitted version of the manuscript. LDB created the app [http://shiny.psy.gla.ac.uk/lmem\\_sim/](http://shiny.psy.gla.ac.uk/lmem_sim/) and DJB created the app <http://shiny.psy.gla.ac.uk/crossed/>.

**Declaration of Conflicting Interests.** The author(s) declared that there were no conflicts of interest with respect to the authorship or the publication of this article.

**Funding.** LMD is supported by European Research Council grant #647910.

## References

- Baayen, R. H., Davidson, D. J., & Bates, D. M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59, 390–412.
- Barr, D. J. (2013). Random effects structure for testing interactions in linear mixed-effects models. *Frontiers in Psychology*, 4, 328.
- Barr, D. J. (2018). Generalizing over encounters: Statistical and theoretical considerations. In S.-A. Rueschemeyer & M. G. Gaskell (Eds.), *Oxford handbook of psycholinguistics*. Oxford University Press.
- Barr, D. J., Levy, R., Scheepers, C., & Tily, H. J. (2013). Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, 68(3), 255–278.
- Bates, D., Mächler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1), 1–48. doi:10.18637/jss.v067.i01
- Bedny, M., Aguirre, G. K., & Thompson-Schill, S. L. (2007). Item analysis in functional magnetic resonance imaging. *Neuroimage*, 35(3), 1093–1102.
- Clark, H. H. (1973). The language-as-fixed-effect fallacy: A critique of language statistics in psychological research. *Journal of Verbal Learning and Verbal Behavior*, 12, 335–359.
- Coleman, E. B. (1964). Generalizing to a language population. *Psychological Reports*, 14, 219–226.
- DeBruine, L. (2019). *Faux (beta) (version v0.0.0.9011-beta)*. Zenodo. doi:10.5281/zenodo.2669587
- Forster, K., & Dickinson, R. (1976). More on the language-as-fixed-effect fallacy: Monte carlo estimates of error rates for  $F_1, F_2, F'$ , and  $\min F'$ . *Journal of Verbal Learning and Verbal Behavior*, 15, 135–142.
- Judd, C. M., Westfall, J., & Kenny, D. A. (2012). Treating stimuli as a random factor in social psychology: A new and comprehensive solution to a pervasive but largely ignored problem. *Journal of Personality and Social Psychology*, 103, 54.
- Locker, L., Hoffman, L., & Bovaird, J. (2007). On the use of multilevel modeling as an alternative to items analysis in psycholinguistic research. *Behavior Research Methods*, 39, 723–730.
- Matuschek, H., Kliegl, R., Vasishth, S., Baayen, H., & Bates, D. (2017). Balancing type I error and power in linear mixed models. *Journal of Memory and Language*, 94, 305–315.
- R Core Team. (2018). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from <https://www.R-project.org/>

- 455 Westfall, J., Nichols, T. E., & Yarkoni, T. (2016). Fixing the stimulus-as-fixed-effect fallacy  
456 in task fMRI. *Wellcome Open Research*, 1.
- 457 Wickham, H., & Henry, L. (2019). *Tidyr: Easily tidy data with 'spread()' and 'gather()'*  
458 *functions*. Retrieved from <https://CRAN.R-project.org/package=tidyr>