Supplemental Materials for “Approaches for Quantifying the ICC in Multilevel Logistic Models–A Didactic Demonstration”

Sean Devine, A. Ross Otto, James O. Uanhoro, Jessica Kay Flake

2023-03-13

What is this?

This is a supplemental companion piece for “Approaches for Quantifying the ICC in Multilevel Logistic Models: A Didactic Demonstration” (**PUBLICATION LINK**). This document is meant to contain all the code (code boxes and other miscellaneous analyses) used in the paper in one place for easy to follow use. Moreover, it also presents some review material referenced in the main text about multilevel modeling more broadly.

The goal of this document is to showcase all of the techniques presented in the paper in a reproducible and easy to follow manner. Throughout this document, we will refer to R functions. We assume readers are familiar with the basics of R syntax and some basic functionality, but will do our best to remain as clear as possible if they are not. This document is meant to be read *alongside* the original paper, as a means of reproducing the code in the main text, in a less didactic fashion. We will not go into as much detail about the concepts being presented and, as such, urge readers to use this as a supplement to the original paper, rather than a wholly complete tutorial in its own right.

This document will proceed in the same order as the original paper and will include all of the same code and data. Note: in the paper, we included paranthetical referencing the associated supplemental section to follow along with. These sections are listed here, with Code Box number from the main paper in parentheses:

Contents

1. Loading and cleaning the data

First, the data is loaded from a .csv file provided on the OSF page for Bogdanov et al. (2022)[https://osf.io/26w4u/]. This is done with the `read.csv()` function that is built-in to R.

```r
data <- read.csv('Bogdanovetal2021/DST_data_osf2.csv')
```

We then subset the data such that `data.Ctl` only contains data from participants in the control condition.

```r
data.Ctl <- data[data$condition=='control', ]
```

Finally, we reverse-code the `effort_choice` variable to make it easier to interpret, such that `0` refers to low effort choices and `1` refers to high effort choices.

```r
data <- data.Ctl$effort_choice <- abs(data.Ctl$effort_choice-1)
```

2. Fitting and summarizing `logistic_MLM0`.

Next, we fit `logisticMLM0` from the paper. This is a random intercept model predicting effort choices in the DST.

To begin, we load the `lme4` package. If you do not have it installed already, run `install.packages('lme4')` to do so.

```r
# if the lme4 is not installed, run this command first:
# install.packages('lme4')
library(lme4)
```

Now we can fit the model described in the text. To do so, we use the `glmer()` function from `lme4`. The first variable before the `~` is the binary outcome variable, `effort_choice` in our case. After the `~`, we input the predictor variables, which in this case is just the intercept, designated by a `1`. Finally, we specify the random effects and grouping variable. Here, we only estimate random intercepts, hence we input a `1` before the `|` followed by the grouping factor, which here is participant number (`PID`). We then specify the data to be used, using the `data=data.Ctl` argument, and finally specify the distribution to be used, which in this case is the `binomial` distribution. For more information on `lme4` syntax in R, see https://www.learn-mlms.com/.

```r
logistic_MLM0 <- glmer(effort_choice ~ 1 + (1|PID), data=data.Ctl, family='binomial')
```

We then output the summary of the model to the console.

```r
summary(logistic_MLM0)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: effort_choice ~ 1 + (1 | PID)
##    Data: data.Ctl
##
##      AIC      BIC   logLik deviance df.resid
##  14234.5  14249.2  -7115.3  14230.5    11202
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.6890 -0.9050 -0.3631  1.0096  4.7586
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  PID    (Intercept) 0.7554   0.8691
## Number of obs: 11204, groups:  PID, 38
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.3338     0.1426  -2.341   0.0192 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As described in the text, the grand average intercept (γ_{00}) is -0.34, which represents the average log-odds of choosing the high-demand option. This can be converted into a probability as follows:

$P(\text{Choose High Demand}) = \frac{\exp(-0.34)}{1+\exp(-0.34)} = 0.42.$

In R, we can do this computation this as follows:

Extract the fixed effects from `logistic_MLM0` using the `fixef` function from `lme4`. Since this is a null model, the only fixed effect is the grand intercept (γ_{00}).

```
gamma_00 = fixef(logistic_MLM0)
```

Convert this to a probability using Eq. 2-3 in the main paper.

```
pChooseHighEffort <- exp(gamma_00)/(1+exp(gamma_00))
cat('estimated group average probability of choosing the high-effort cue = ', pChooseHighEffort, '\n')
```

```
## estimated group average probability of choosing the high-effort cue =  0.4173215
```

Print this estimate to the console.

```
cat('estimated group average probability of choosing the high-effort cue = ', pChooseHighEffort, '\n')
```

```
## estimated group average probability of choosing the high-effort cue =  0.4173215
```

3. Reproducing Figure 2 in the paper

To reproduce Figure 2 in the paper, we will extract the estimated demand preferences (in log odds) per participant in the DST. These estimates are also known as "empirical Bayes' estimates", hence we will store these in a variable called `eb`.

```
eb  <- coef(logistic_MLM0)$PID
head(eb)
```

```
##   (Intercept)
## 1 -1.35734356
## 2 -0.28074761
## 3 -0.03893175
## 4 -0.03199605
## 5  0.77900510
## 6 -0.42353287
```

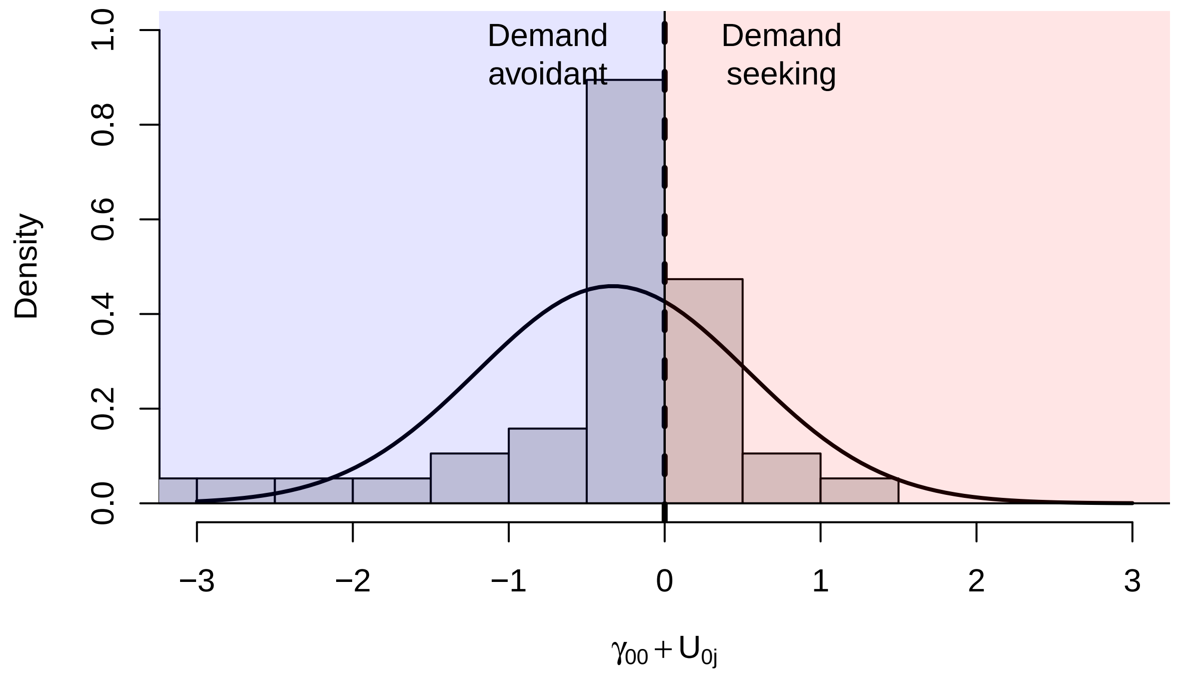
Next, we extract the variance around the intercept (τ_0^2) using the `VarCorr` function from `lme4$`. The syntax here is a little convoluted. The `VarCorr` command extracts the random variance components from a fitted model (here `logistic_MLM0`). Alone, this yields a variance component per grouping variable and per number of random variance parameters specified in the model (τ_0^2, τ_1^2, etc.). In our case, we have one grouping variable, `PID`, and so we specify that we are interested in this the random intercept variance for this group using `$PID[1]`.

```
tau <- VarCorr(logistic_MLM0)$PID[1]
```

Finally, we visualize this distribution in a histogram and overall an estimated normal density curve with $\mu = \gamma_{00}$ and $\sigma = \sqrt{\tau_0^2}$. The specific steps taken here are not of critical importance to the topic of this tutorial, as they mainly involve plotting commands in base R. If this section is confusing for readers and they wish to be able to follow along better, we suggest this introductory website that explains basic visualization techniques inherent to R: https://bookdown.org/rdpeng/exdata/the-base-plotting-system-1.html.

```r
# plot the histogram of the random intercepts and overlay an estimated normal density curve with
# mu = gamma00 and sd = sqrt(tau^2_0)
hist(eb[[1]], main='', xlab=expression(gamma[`00`] + U[`0j`]), freq = F, xlim=c(-3,3), ylim=c(0,1))
curve(dnorm(x, fixef(logistic_MLM0), sqrt(tau)), add=T, lwd=2)

# finally, shade in areas of demand-avoidance (b0 < 0) and demand-seeking respectively (b0 > 0)
abline(v=0, lty='dashed', lwd=3)
rect(0, 0, 10, 100, col = scales::alpha('red', .1))
rect(0, 0, -10, 100, col = scales::alpha('blue', .1))
text(-.75, .95, labels = 'Demand\navoidant')
text(.75, .95, labels = 'Demand\nseeking')
```



4. Computing the ICC

4a. Latent Threshold Method

First, We extract the random intercept variance (τ_0^2) using the `VarCorr` command (see explanation above).

```r
tau20 <- VarCorr(logistic_MLM0)$PID[1]
```

Then, we specify the within-subjects (residual) variance to be $\sigma^2 = \frac{\pi^2}{3}$.

```r
sigma2    <- pi^2/3
```

Finally, we compute the ICC using the following equation $\frac{\tau_0^2}{\tau_0^2+\sigma^2}$ and print this output to the console.

```r
threshICC <- tau20/(tau20+sigma2)
cat('ICC using the Latent Threshold Approach = ',threshICC,'\n')
```

```
## ICC using the Latent Threshold Approach =  0.1867374
```

4b. Simulation Method

The simulation method requires multiple steps.

First, we set a seed for reproducibility. This ensures that the subsequent results will reproduce on any device that runs this code.

```r
set.seed(2022)
```

Next, we extract the the random intercept variance using `VarCorr`.

```r
tau20 <- VarCorr(logistic_MLM0)$PID[1]
```

We extract the grand intercept (γ_{00}) using `fixef` (see explanation above).

```r
gamma00 <- fixef(logistic_MLM0)[[1]]
```

We set a large number of simulations to execute (here `1e5`or 100 000). We call this variable `M`.

```r
M <- 1e5
```

Next, we simulate the random effects from `M` draws and store these values in a variable called `U0j`. Note. we take the square root of τ_0^2 because `rnorm()` requires the standard deviation, not the variance.

```r
U0j <- rnorm(M, 0, sqrt(tau20))
```

We then compute the expected probability (in log odds) of choosing the high demand option on each draw. Because `logisticMLM_0` has no predictors, this is computed as the grand intercept (γ_{00}) plus the random deviations from this intercept we simulated in the previous step.

```r
logit_p_hat <- gamma00+U0j
```

This is then converted into a probability (using Eq. 2-3 from the paper).

```r
p_hat <- exp(logit_p_hat)/(1+exp(logit_p_hat))
```

We can then compute the level 1 variance using the equation for the Bernoulli variance: $\text{var} = \hat{p} * (1 - \hat{p})$.

```r
var_L1 <- p_hat*(1-p_hat)
```

Finally, we compute the ICC in the traditional way, taking the average level 1 variance as σ^2 and the variance of the predicted probabilities from each simulation $\hat{p}$ as τ_0^2

```r
sigma2 <- mean(var_L1)
simICC <- var(p_hat)/(var(p_hat) + sigma2)
```

We then print the result to the console.

```r
cat('ICC using the Simulation Model = ',simICC,'\n')
```

```
## ICC using the Simulation Model =  0.1390484
```

4c. Linearization Method

The linearization method also proceeds in a series of steps.

First, τ_0^2 and γ_{00} are extracted from `logistic_MLM0` using `VarCorr` and `fixef` as we have done above.

```r
tau20   <- VarCorr(logistic_MLM0)$PID[1]
gamma00 <- fixef(logistic_MLM0)[[1]]
```

Next, we evaluate the probability of success at the mean of random effects (i.e., at the fixed effect, γ_{00}).

```r
p <- exp(gamma00)/(1+exp(gamma00))
```

We then compute the Bernouilli variance of this fixed estimate and store this value in a variable called `var1`.

```r
var1 <- p*(1-p)
```

Next, we compute the variance in the level-1 outcome using the linearized equation provided in the main text. We store this in a variable called `var2`.

```r
var2  <- tau20*p^2*(1 + exp(gamma00))^(-2)
```

Finally, with these values in hand, we compute the ICC, taking `var2` as our measure of between-person variance, τ_0^2, and `var1` as our measure of within-person variance, σ^2, and print the result of this computation to the console.

```r
linICC  <- var2/(var1+var2)

# print result
cat('ICC using the Linearization method = ',linICC,'\n')
```

```
## ICC using the Linearization method =  0.1551821
```

4d. Median Odds Ratio

The MOR is straightforward to compute.

First, we extract the random intercept variance from the model using `VarCorr`.

```r
tau20 <- VarCorr(logistic_MLM0)$PID[1]
```

Next, compute the 75th percentile of the cumulative distribution function of a standard normal distribution. In R, this can be done using the `qnorm` function and specifying the percentile (here 0.75).

```r
phi75 <- qnorm(.75) # 75th percentile of normal CDF
```

Now we can compute the MOR as: $MOR = \exp\left(\sqrt{2\tau_0^2}\phi(0.75)\right)$ and print this value to the console.

```r
MOR <- exp(sqrt(2*tau20)*phi75)
cat('Median Odds Ratio = ',MOR,'\n')
```

```
## Median Odds Ratio =  2.291137
```

5. Bootstrapping

Below we provide two bootstrapping techniques. First, we describe how to implement the bootstrapping procedure using the functions we provide alongside the paper. These functions are meant to be simple to implement, without the need for complex code that involves loops and storage. That being said, some readers may be interested in understanding this process. Accordingly, we provide a fully-worked example that goes through such a process.

5a. Bootstrapping using provided functions

To utilize the function calls stored in `fx.R`, we first have to source them using the `source()` function from R.

```
source('fx.R')
```

Once this is done, we should have access to all of the custom functions provided in `fx.R`. Of interest to us here are those used for bootstrapping. Specifically, `bootstrap_icc`. This function takes the following arguments:

- `fit`: The fitted model
- `gr` : he grouping variable (in our case, `PID`)
- `method`: The method to use: `icc_thr` (Threshold method), `icc_sim` (Simulation method), `icc_lin` (Linearization method), `MOR` (MOR).
- `B`: The number of samples to produce.
- `seed`: The random seed to use.

As an example, we can compute 100 bootstraps of the ICC using the latent threshold method as follows. We can the print out these bootstrapped values.

Note: This can take a few moments.

```
bootstrap_thr <- bootstrap_icc(logistic_MLM0, gr='PID',
                               method='icc_thr', B = 100, seed = 2022)
```

```
## ==========================================================================
```

```
bootstrap_thr
```

```
##   [1] 0.16830697 0.15787487 0.23262770 0.19854612 0.17972198 0.16440196
##   [7] 0.17151428 0.17047279 0.15203957 0.14435074 0.20494568 0.22442203
##  [13] 0.27947478 0.15180074 0.16084417 0.11891926 0.12382954 0.15670157
##  [19] 0.21180850 0.26129530 0.09869003 0.12523149 0.18097583 0.18429135
##  [25] 0.24291502 0.18805121 0.27131427 0.18858676 0.19876350 0.19104779
##  [31] 0.17898273 0.19081998 0.21721753 0.19838683 0.13443018 0.16345277
##  [37] 0.17571961 0.21163977 0.25371038 0.18071584 0.19019566 0.22854664
##  [43] 0.22408069 0.16930695 0.20692029 0.13554932 0.19115447 0.15771066
##  [49] 0.17324681 0.15087385 0.21250849 0.22208254 0.21865074 0.21256008
##  [55] 0.18931581 0.21051402 0.15417479 0.19446001 0.10452172 0.22578986
##  [61] 0.21902593 0.17891327 0.16474834 0.16896050 0.23221552 0.18091489
##  [67] 0.22636177 0.17349255 0.14743257 0.09563111 0.23128899 0.15047848
##  [73] 0.14955724 0.15063867 0.15099533 0.19727423 0.16227090 0.19690115
##  [79] 0.17141104 0.23512419 0.16254044 0.11387893 0.22550295 0.17085960
##  [85] 0.14860604 0.22323194 0.16042631 0.21371768 0.17335656 0.19818017
##  [91] 0.23903077 0.21792764 0.16886654 0.22587631 0.20353066 0.14575272
##  [97] 0.14468275 0.14297011 0.21971101 0.18426185
```

Using the built-in `quantile` function in R, we can then compute 95% confidence intervals based on these bootstrapped estimates.

```
quantile(bootstrap_thr, c(.025, .975))
```

```
##      2.5%     97.5%
## 0.1089664 0.2576925
```

For completeness, we illustrate the same syntax for bootstrapping estimates using the other methods described above, but do not run this code for time purposes.

```r
bootstrap_sim <- bootstrap_icc(logistic_MLM0, gr='PID', method='icc_sim', B = 100, seed = 2022)
quantile(bootstrap_sim, c(.025, .975))

bootstrap_lin <- bootstrap_icc(logistic_MLM0, gr='PID', method='icc_lin', B = 100, seed = 2022)
quantile(bootstrap_lin, c(.025, .975))

bootstrap_MOR <- bootstrap_icc(logistic_MLM0, gr='PID', method='MOR', B = 100, seed = 2022)
quantile(bootstrap_MOR, c(.025, .975))
```

5b. Bootstrapping “by-hand”

Below we provide commented code to estimate bootstrapped samples “by-hand” for readers who are interested. We will not comment on this in depth, as it is outside the scope of the tutorial and we provide functions to accomplish the same effect, but interested readers are welcome to explore the code below.

```r
# 0. Set constants for bootstrapping procedure
B       <- 1000                    # number of bootstraps
ids     <- logistic_MLM0@frame$PID # extract id vector from model data
K       <- length(unique(ids))     # number of clusters (subjects)
nTrials <- table(ids)              # number of trials per subject
tau20   <- VarCorr(logistic_MLM0)$PID[1]
g00     <- fixef(logistic_MLM0)[[1]]
output  <- matrix(NA, B, 7, dimnames = list(NULL, c('iteration', 'threshICC', 'simICC', 'linICC', 'MOR'

# 1. Cycle through iterations (i) of bootstrapped samples
for(i in 1:B) {
  if(i%%100==0) cat('bootstrapping is', round(i/B,2)*100, '% complete.')
  # 1.1 Simulate data
  U0j           <- rnorm(K, 0, tau20)        # random deviations per subject
  LOR_j         <- g00 + U0j                 # log odds of response==1 per subject
  p_j           <- exp(LOR_j)/(1+exp(LOR_j)) # convert LOR to probability
  y_ij          <- sapply(1:K,
                          function(k)
                            rbinom(nTrials[k], 1, p_j[k]))
  y_ij          <- unlist(y_ij)              # break out of a list format

  # 1.2 Fit new model and compute values of interest
  thisMLM       <- glmer(y_ij ~ 1 + (1|ids), family='binomial')
  thisG00       <- fixef(thisMLM)[[1]]
  thistau20     <- VarCorr(thisMLM)$ids[1]

  # 1.2.1. Latent Threshold ICC
  thisthreshICC <- thistau20/(thistau20+pi^2/3)

  # 1.2.2. Simulation ICC
  thisU0j       <- rnorm(1e5, 0, sqrt(thistau20))
  logit_p_hat   <- thisG00+U0j
  p_hat         <- exp(logit_p_hat)/(1+exp(logit_p_hat))
  var_L1        <- p_hat*(1-p_hat)
  sigma2        <- mean(var_L1)
  thissimICC    <- var(p_hat)/(var(p_hat) + sigma2)

  # 1.2.3. Linearlization
  p             <- exp(thisG00)/(1+exp(thisG00))
  var1          <- p*(1-p)
  var2          <- thistau20*p^2*(1 + exp(gamma00))^(-2)
  thislinICC    <- var2/(var1+var2)

  # 1.2.4. MOR
  thisMOR       <- exp(sqrt(2*thistau20)*qnorm(.75))

  # 1.3. Save output
  output[i,] = c(i, thisthreshICC, thissimICC,
                 thislinICC, thisMOR, thisG00, thistau20)

}
```

6. Background Review and Additional Information

a. Linear regression of response times

After selecting an effort cue in the DST (Bogdanov et al., 2021), participants had to judge a digit between 1 and 9, excluding 5, according to its font colour (TS_correct represents whether participant were correct (1) in their judgements or not (0) and TS_RT is the response time for digit judgement in seconds): yellow = parity (whether the number was even or odd), blue = magnitude (whether the digit was greater or smaller than 5). If participants selected the high-demand option, the task rules (font colours) would switch with a 0.9 probability. If they chose the low-demand option, they would switch with a 0.1 probability (see Figure 1). It is well-documented that trials where the task rule switches from one trial to the next requires increased cognitive effort to resolve than those where rules repeat (Monsell, 2003; Switch, 1 = task switched from previous trial, 0 = task repeated).

The goal of traditional linear regression is to predict a continuous outcome variable as a linear combination of predictor variables. Formally,

$Y_i = b_0 + b_1 X_i + e_i$

Using the example of task-switching reaction time in the DST described above, Y_i is the predicted reaction time on trial i and X_i is the effects-coded switch condition (-1 = repeat, -1 = switch). b_1 represents the relationship between Y_i and X_i or more specifically the increase in Y_i for a one-unit increase in X. Accordingly, b_0 represents the value of Y_i when the result of $b_1 X = 0$ (the intercept), which in this case refers to the grand mean reaction time across switch conditions because the outcome has been centered. Finally, e_i is the residual: the distance between the actually observed value of Y on trial i and its predicted value, Y_i. In traditional linear regression, these residuals are assumed to be normally distributed, with mean equal to zero and variance set to a constant value, $\sigma^2 : N(0, \sigma^2)$.

Using data from Bogdanov et al. (2021), we can estimate this model in R as follows and use the `summary()` command to print the output.

```r
linear_regression <- lm(TS_RT ~ Switch, data=data.Ctl)
print(summary(linear_regression))
```

```
##
## Call:
## lm(formula = TS_RT ~ Switch, data = data.Ctl)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2348 -0.4562 -0.2658  0.0910 27.1693
##
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.00780    0.01307   77.12   <2e-16 ***
## SwitchTRUE   0.27067    0.01940   13.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.022 on 11202 degrees of freedom
## Multiple R-squared:  0.01708,    Adjusted R-squared:  0.01699
## F-statistic: 194.6 on 1 and 11202 DF,  p-value: < 2.2e-16
```

Doing so, we find a grand mean reaction time (b_0) of 1.01 s. and a significant effect of switch condition on reaction time (b_1), such that judgements on switch trials were 0.27 s. slower than those on repeat trials.

Furthermore, we can extract an estimate of σ^2 from the model, using sigma(linear_regression)ˆ2‘, which yields a value of 1.05. Conceptually, this value corresponds to the "noisiness" of the distribution relative to our model's predictions. In the context of a psychological experiment, this value is important. Like any model however, linear regression relies on certain assumptions to be met for its estimates to be valid and for proper inferences to be made. Critically for our purposes, standard linear regression relies on the assumption that residuals are unrelated—in other words, that each observation is independent from the last. Problematically, this assumption is regularly violated in the context of within-participants laboratory experiments, where responses are known to come from the same participants—as is the case in the current data. As discussed in the introduction, in these cases, data naturally cluster together—e.g., some participants are slower on average than others—which violates the assumption of independence.

b. Multilevel linear regression of response times

Data from Bogdanov et al. (2021) are nested: a single participant provides 300 responses during the DST. As a result, observations are not independent and simple linear regression is inappropriate to model these data. To account for dependencies in the data, we can extend simple linear regression techniques to account for both "levels" of the data: the observation (the responses a participant made during the DST) and the participant (participant 1, participants 2, and so on)—hence a multilevel model:

$$Level1 : Y_{ij} = b_{0j} + b_{1j}X_{ij} + R_{ij}$$
$$Level2 : b_{0j} = \gamma_{00} + U_{0j}$$
$$Level2 : b_{1j} = \gamma_{10} + U_{1j}$$

Here, Y_{ij} represents the predicted value of the outcome variable (for this example, reaction time during task-switching, `TS_RT`) on trial i for participant j. Note that unlike simple linear regression, we have explicitly recognized that trial i is nested within participant j. Accordingly, b_{0j} and b_{1j} represent the intercept and slope for participant j specifically: that is, the value of Y_{ij} when X_{ij} is equal to zero and the estimated effect of X_{ij} on Y. Similarly, X_{ij} corresponds to the value of X on trial i for participant j and R_{ij} corresponds to the residual between the actual value of Y on trial i for participant j and the predicted value, Y_{ij}, which are normally distributed, $N(0, \sigma^2)$.

Conceptually–though not mathematically (see Snijders & Bosker, 2011, p. 88)–Level 1 of equation 1 can be thought of as applying the standard linear regression to the data from each participant. If one were to do so, they would obtain a vector of coefficients for each participant, the "average" of which would correspond to the fixed effects at level 2 (γ_{00}, γ_{11}). More precisely, γ_{00} represents the estimated grand mean of Y–the predicted value of Y when X is equal to zero across participants in the data. Similarly, γ_{10} represents the overall–i.e., across participant–influence that X exerts on Y. These fixed effects can be thought of as 'group-level' predictors: they represent the effect of a predictor on an outcome across individual participants, at the group-level. In this regard, the regression coefficients used in Level-1 of equation S2 (the bs) are the sum of these group-level estimates, γ, and each participant's deviation from this group-level effect, U_{0j} and

U_{1j}–i.e., how much more or less sensitive a given participant was on the outcome variable to changes in a predictor than γ_{00} and γ_{10} respectively.

To continue with the analogy of standard linear regression, these deviations from the fixed effects, can be thought of as level-2 residuals: each participant's distance from the group-level "average" effect. These values play a central role in a multilevel framework, as they explicitly capture variation between participants. Just like level-1 residuals, level-2 residuals are assumed to be normally distributed with mean zero and variance τ_0^2: $U \sim N(0, \tau_0^2))$.

In R, we can fit a linear multilevel model using the `lme4` package, where `PID` is the grouping variable preceded by `1 + Switch`, which represents random intercepts for each level of `PID`, and random slopes for `Switch`. We can print the output from this model using the `summary(linear_MLM)` command.

```
linear_MLM <- lmer(TS_RT ~ Switch + (1+Switch|PID), data=data.Ctl)
summary(linear_MLM)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: TS_RT ~ Switch + (1 + Switch | PID)
##    Data: data.Ctl
##
## REML criterion at convergence: 30829.1
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.3938 -0.3956 -0.1769  0.1238 26.0282
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  PID      (Intercept) 0.09685  0.3112
##           SwitchTRUE  0.18078  0.4252   0.16
##  Residual             0.89786  0.9476
## Number of obs: 11204, groups:  PID, 38
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  1.00471    0.05198  19.330
## SwitchTRUE   0.33839    0.07151   4.732
##
## Correlation of Fixed Effects:
##            (Intr)
## SwitchTRUE 0.113
```

Across participants, we find a mean reaction time (γ_{00}) of 1.00 s (`(Intercept)`) and a switch effect on reaction time (γ_{10}) of 0.34 s (`SwitchTRUE`).

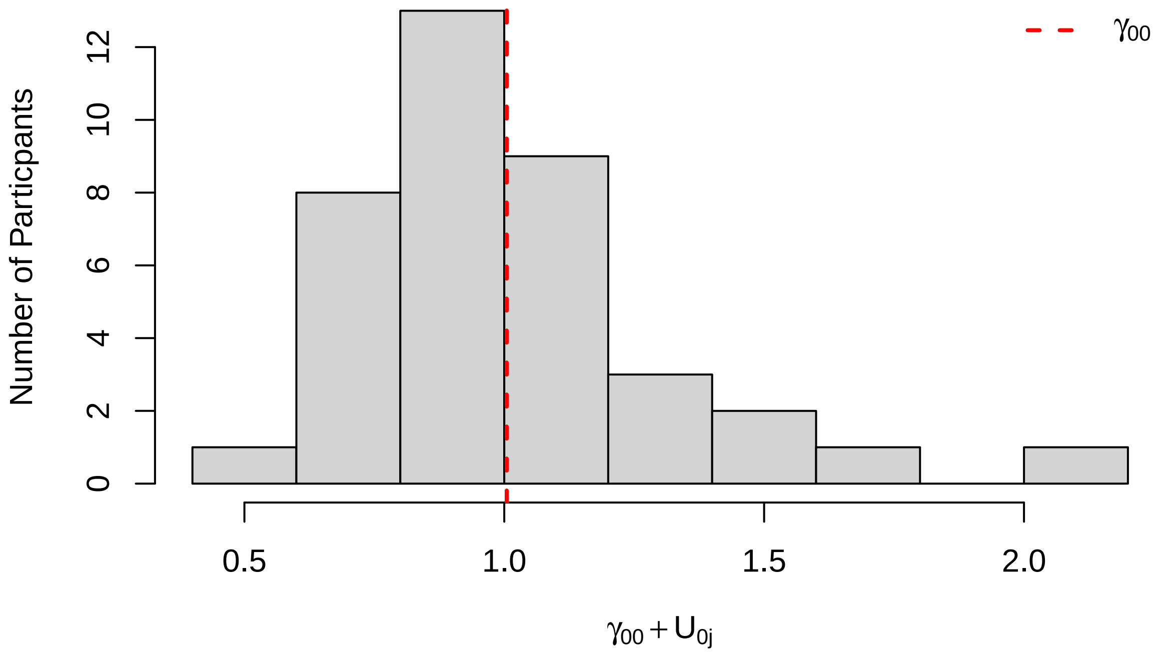
Measures of variation in linear MLM

In the `Random effects` section of the output, we also see that the variance about the intercept (τ_0^2) is 0.12 and the residual variance (σ^2) is 0.93. This variation can be extracted using the `ranef(linear_MLM)` command, which produces a data frame of the estimated deviations from γ_{00} per level of the cluster variable, i.e., U_{0j}

```
head(ranef(linear_MLM)$PID)
```

```
##   (Intercept)  SwitchTRUE
## 1 -0.01022218  0.33011574
## 2 -0.22426419 -0.09013081
## 3  0.01205297 -0.04077640
## 4  0.07883989  0.47799209
## 5 -0.12060272 -0.12462159
## 6  0.02519668 -0.14967480
```

This variation is depicted in the figure below, where, it is clear that, while the fixed intercept capture group-level average reaction time well, there is sizable variation between individuals with regards to their overall reaction time.



In a multilevel framework, the total variance in the outcome is the sum of the individual variance components; in this case: $var = \tau_0^2 + \sigma^2$. As such, the proportion of total variance accounted for by variations in sensitivities to effects between individuals,τ_0^2– a value called the intraclass correlation coefficient (ICC, a.k.a. variance partition coefficient), or simply ρ–can be computed as $\rho = \frac{\tau_0^2}{var}$. Conceptually, the ICC corresponds to the degree of nesting in a dataset, or the “importance” of the cluster variable (i.e., the variable level-1 data are nested within, here: participants) in explaining variance in the outcome (Goldstein et al., 2002). In the case above, the ICC is approximately 0.12 ($\frac{0.12}{0.12+0.93}$). Because the data are nested within different participants, the ICC is a very useful summary of the degree to which the outcome variable is sensitive to unknown individual differences. Here, 12% of variation in participant’s mean reaction time is attributable to unmeasured differences between participants. Additionally, the ICC can be used to quantify the proportion of unexplained variance within participants—accounted for by σ^2. In this case, 88% ($1 - \rho$) of variance within participants’ reaction times is unexplained by the current model.

c. Numerical Integration Approach

The simulation approach assumes that the model’s predicted probabilities are logit-normal distributed and estimates the mean and variance of this logit-normal distribution via simulation. Alternatively, the mean and variance can be estimated via numerical integration using the model’s predicted responses (or average predictions in the case of a model with covariates), denoted below as γ, and the intercept variance, τ_0^2:

$$\mu \approx \frac{1}{K-1}\sum_{i=1}^{K-1} logit(i)\phi(\gamma, \tau_0^2)$$

$$\sigma^2 \approx \frac{1}{K-1}\sum_{i=1}^{K-1} logit(i) - \mu^2\phi(\gamma, \tau_0^2)$$

where ϕ is a normal density function. With these values in hand, the ICC can be computed using the equation presented in step 3 of the simulation approach in the main text, accounting for the logit scale:

$$\rho = \frac{\sigma^2}{\mu - \mu^2}$$

This approach yields very similar numeric values to the simulation approach but does not require as many computational resources to compute—especially in combination with the bootstrapping methods discussed in the main text. In the function calls described in the next section, this approach is called using `icc_num()`.

d. Function Call Examples

Code boxes in the main text were designed to be as explicit as possible so readers could follow along with each step. In practice, much of the computation can simplified and abstracted so that end-users can simply apply a pre-programmed function to their already-fit model and obtain the desired statistic.

To this end, we have made a series of functions available to readers that computes the statistics described in this tutorial: `icc_thr`, `icc_sim`, `icc_num`, `icc_lin`, `MOR`, `bootstrap_icc`. These functions are available on GitHub.

Below, we demonstrate how to use call each function to an already fit model, `logistic_MLM0`.

```r
source('fx.R') # load all functions

# For all functions:
# first argument specifies the model (glmer object)
# second argument specifies the group identifier (string)

# Threshold Approach

icc_thr(logistic_MLM0, "PID")
```

```
## [1] 0.1867374
```

```r
# Simulation Approach
# third argument specifies the random seed for the simulations
# fourth argument specifies the numbers of simulations

icc_sim(logistic_MLM0, 'PID', seed=2022, nsims=1e4)
```

```
## [1] 0.1383959
```

```r
# Numerical Integration Approach
icc_num(logistic_MLM0, "PID")
```

```
## [1] 0.139142
```

```r
# Linearization

icc_lin(logistic_MLM0, "PID")
```

```
## [1] 0.1551821
```

```r
# MOR

MOR(logistic_MLM0, "PID")
```

```
## [1] 2.291137
```

```r
# Bootstrapping
# third argument specifies the method for which bootstrapped samples are to be computed (string)
# fourth argument specifies the number of bootstraps (integer)
# fifth argument specifies the random seed

bootstrap_icc(logistic_MLM0,"PID","icc_lin",B=10,seed=2022)
```

```
## ================================================================================
##  [1] 0.1339525 0.1265975 0.1928965 0.1747576 0.1361845 0.1340821 0.1366112
##  [8] 0.1314135 0.1304067 0.1105326
```

References

Bogdanov, M., Nitschke, J. P., LoParco, S., Bartz, J. A., & Otto, A. R. (2021). Acute psychosocial stress increases cognitive-effort avoidance. *Psychological Science, 32*(9), 1463-1475.

Goldstein, H., Browne, W., & Rasbash, J. (2002). Partitioning variation in multilevel models. *Understanding statistics: statistical issues in psychology, education, and the social sciences, 1*(4), 223-231.

Monsell, S. (2003). Task switching. *Trends in cognitive sciences, 7*(3), 134-140.