Data Analysis and Power Simulations with General Linear Mixed Modelling for Psychophysical Data – A Practical, R-Based Guide

Björn Jörges

Center for Vision Research, York University, 4700 Keele Street, Toronto, ON M3J 1P3, Canada

Abstract

Sample size planning is not straight-forward for the complex designs that are usually employed in psychophysical (two-alternative forced-choice) experiments: characteristics such as binary response variables and nested data structures where responses may be correlated differently within participants and experimental sessions than across participants and experimental sessions make it harder to estimate the necessary number of participants and trials with traditional means. In this practical R-based guide, we first show in detail how we can simulate verisimilar psychophysical data. We then use these simulations to compare two different methods by which two-alternative forced-choice data can be analyzed: (1) the "two-step" approach, where first psychometric functions are fitted and then statistical tests are performed over the parameters of these fitted psychometric functions; (2) an approach based on Generalized Linear Mixed Modeling (GLMM) that does not require the intermediary step of fitting psychometric functions. We argue that the GLMM approach enhances statistical validity and show that it can increase statistical power. Finally, we provide a sample implementation of a simulation-based power analysis that can be used as-is for many simple designs, but is also easily adaptable for more complex designs. Overall, we show that a GLMM-based approach can be beneficial for data analysis and sample size planning for typical (two-alternative forced-choice) psychophysical designs.

Introduction

Data from psychophysical experimental designs such as Two-Alternative Forced-Choice tasks are challenging to analyze. Dependent variables are often binary ("yes"/"no") and the data are often generated in a nested fashion, with multiple participants completing many trials that are often arranged in blocks, with manipulations occurring between and/or with participants and/or blocks. Simple statistical tools for Null Hypothesis Significance Testing like Student's *t* tests or Analyses of Variance (ANOVAs) are not equipped to deal with these specific properties. For these reasons, usually Psychometric Functions (Cumulative Gaussian or Weibull functions) are fitted for each condition, block and participant to obtain the Points of Subjective Equality (PSEs) and Just Noticeable Differences (JNDs); see Figure 1. This yields one data point per subject, condition, and block, over which a t test or an ANOVA is performed to test for statistical significance on the population level. While this can be a valid approach, it generally neglects that each PSE and JND is based on a large number of trials and thus fails to account for the added reliability this provides. Depending on the experimental design, this may lead to a loss of statistical power and smaller effect may go undetected. Furthermore, these approaches are prone to neglecting internal structures of the data, which can lead to biased population effect estimates: for example, responses from

one participant may correlate more strong within one block or condition than across blocks or conditions. As a solution to both problems, Moscatelli at al. (Moscatelli, Mezzetti, & Lacquaniti, 2012) have suggested the use of General Linear Mixed Modelling (GLMM). GLMM allows to fit population parameters across all data, while still taking into account that responses within each condition and participant are correlated more strongly than across conditions and participants. While Moscatelli et al. have provided a thorough introduction to this type of analysis, some specifics of this approach require further investigation. The first part of this manuscript thus uses simulations to answer some of the open questions and provides concrete recommendations for scientists aiming to use this type of analysis.

While Cognitive Psychology has remained largely unscathed, the replication crisis has shaken parts of Psychology to its core (Aarts et al., 2015; Hunter, 2001; Oberauer & Lewandowsky, 2019). In response, and among other suggestions for improvements (Chambers, Dienes, McIntosh, Rotshtein, & Willmes, 2015; Devezer, Nardin, Baumgaertner, & Buzbas, 2019; Guest & Martin, 2020; Nosek, Ebersole, DeHaven, & Mellor, 2018; Nosek & Lakens, 2014; Oberauer & Lewandowsky, 2019), the need for an open and thorough study planning has been acknowledged much more widely. Power analyses are a crucial part of study planning: they provide an idea of how many participants have to be tested in how many trials to obtain an adequate probability of detecting an effect, under the assumption that there is a true effect of a given strength. Psychophysical studies often rely on heuristics when it comes to sample size planning, such as “10 participants are sufficient to observe a within-participant effect”. While this practice may not have equally pernicious consequences in Cognitive Psychology as in other areas of psychology, replicability rates are still only about 50% in this area (Aarts et al., 2015). A more rigorous and principled approach to sample size planning, together with other efforts to increase theory-building, openness and replicability, could be useful to remedy this issue. Power analyses are slowly becoming more mainstream in other areas, but they still are the exception in the typical psychophysical study. Some general tutorials for power analyses through simulations have been brought forward that are quite easily adaptable to many different designs (Debruine & Barr, 2019; Kumle, Võ, & Draschkow, 2020). Based on Linear Mixed Modelling, these take into account complex data structures where often several participants complete a large number of trials in several conditions and blocks. However, common psychophysical designs such as two-alternative forced-choice tasks, which typically generate psychometric curves, require additional considerations. Among these additional considerations are the fact that responses are often binary, and that relationships between dependent and independent variables are usually not linear. Based on the considerations about the GLMM outlined in the first part of the paper, the second chapter thus provides guidelines for power analyses for common psychophysical designs that investigate the effect of a categorical experimental variable on precision (JNDs) and accuracy (PSEs) in two-alternative forced-choice paradigms.

In this manuscript, we thus first provide a step-by-step tutorial to simulating psychophysical datasets derived from Two-Alternative Forced-Choice (2AFC) tasks in R. We then use these data to compare different statistical approaches to Null Hypothesis Significance Testing for 2AFC data. And finally, we will provide an implementation for power analyses based on the analyses we identified as most robust in the previous chapter, both in R and in Julia. Finally, we compare different implementations and give concrete recommendations. This paper includes both theoretical considerations and practical implementations; we provide R and Julia code for the practical parts (extraction of simulation parameters from preexisting datasets, simulation of datasets, data analysis and power analysis) throughout the manuscript, and will provide GitHub links for the code we use to support theoretical points (e.g., to compare different analyses, model specifications and fitting methods).

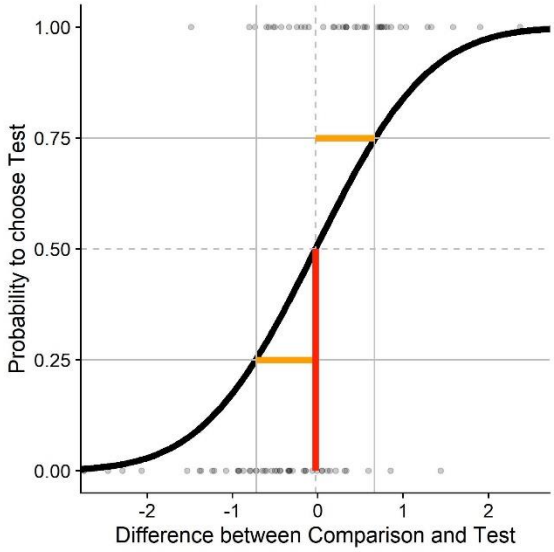


Figure 1: Sample psychometric function (a Cummulative Gaussian function in this case) for a two-alternative forced choice task. We plot the difference in stimulus intensity (x axis) against the probability to judge that the test stimulus had the higher intensity (black curve). The JND (Just Noticeable Difference), a measure of sensitivity/precision, is that difference in stimulus intensity that leads to a 25%/75% response probability (yellow); 0.7 in this example. The PSE (Point of Subjective Equality), a measure of biases/accuracy, is that stimulus intensity that leads to 50 % correct responses (red); 0 in this example.

Before diving into the main topics, we want to clarify a few basic psychophysical concepts that will be important throughout the paper.

Figure 1 displays a **Psychometric Function**. Psychometric functions are sigmoid functions (Cumulative Gaussians, Weibull or logistic functions) that are fitted to binomial response data ("yes-no", "bigger-smaller", …). These functions provide information about the **Point of Subjective Equality (PSE**; red in Figure 1) and the **Just Noticeable Difference (JND**; yellow in Figure 1). The **Point of Subjective Equality (PSE)** is that stimulus strength where the participant is equally likely to choose the **test stimulus** (i.e., where a manipulation is present) as a **comparison stimulus** (i.e., where no manipulation is present) as being more intense (bigger, brighter, faster, …). Graphically, this corresponds to the point on the x axis where the psychometric function reaches a y value of 0.5. It is used as a measure of biases, such as biases induced by the manipulation. Such biases indicate the **accuracy** of human performance. The **Just Noticeable Difference (JND)** corresponds to the distance between the PSE and those stimulus strengths where subjects have a probability of 25% or 75% to choose the test stimulus as more intense (bigger, brighter, faster, …). Graphically, this corresponds to the difference between the PSE (that is, the x value for which a y value of 0.5 is achieved) and the x value for which a y value of 0.25 or 0.75 is achieved. Since psychometric functions are symmetrical by definition, it doesn't matter whether 0.25 or 0.75 are chosen, but the JND is always positive. JNDs speak to the **precision** component of performance. In this paper, we work mostly with cumulative Gaussian functions as psychometric functions. Cumulative Gaussians are given by a **mean** and a **standard deviation**. The mean is equivalent to the PSE and the standard deviation is proportional to the JND. While the JND denotes the difference in stimulus intensity that leads to a probability of 25% or 75% to choose the test stimulus as more intense, one standard deviation denotes probabilities of 16.7% and 83.3%, respectively. To increase readability, we will therefore mostly use "JND", except when exact value of the JND or the standard deviation is relevant.

Simulating Psychophysical Data

We will first discuss the specifics of simulating psychophysical data for one typical case of psychophysical study: A Two-Alternative Forced-Choice task with a within-participant manipulation, where the presented stimuli are chosen according to a staircase procedure. Words in a different fond refer to variables in the script. For some of the variables, we demonstrate how to derive them from existing (pilot) datasets, using some pilot data from our lab. [The code and sample data are available on GitHub.]

Required values

This method requires estimates of all relevant parameters. Some pertain to the stimuli, some can be taken from the literature, and some must be guessed (educatedly).

`ID` is a vector containing one ID for each subject we want to simulate.

`ConditionOfInterest` is a vector containing IDs for a binary categorical variable related to the main hypothesis of the experiment. For example: Is there a pictorial background scene?

`StandardValues` is a vector containing values for a categorial variable that serves as comparison stimuli. It can contain one value if you want to determine PSEs/JNDs for only one stimulus intensity, but typically you will have several, e. g. when you want to diversify your stimuli to show that a certain effect is not tied to one specific stimulus strength.

`reps` is a vector containing an ID for each trial, the maximum number being the average number of trials we expect for any given staircase.

`PSE_Difference` is a value that indicates the percentage to which the PSEs differ between test and standard condition. It can be zero if the condition of interest is not expected to influence PSEs.

`JND_Difference` is a value that indicates the percentage to which the JNDs differ between test and standard condition. It can be zero if the condition of interest is not expected to influence JNDs.

`Mean_Standard` is the Mean of the psychometric function expected for the standard condition. In many cases, this is the stimulus strength of the comparison stimulus.

`Multiplicator_SD_Standard` is the Standard Deviation of the psychometric function expected for the standard condition, normalized to a mean of 1. We later multiply this normalized standard deviation by the Mean of the psychometric function we aim to simulate. That is, we assume that Weber fractions are constant across the tested stimulus range, which is generally assumed to hold for many cases. While this has been put into doubt (Krueger, 1989) and we recommend to verify to what extent Weber's law holds for the stimulus in question, we believe this to be a reasonable simplification.

The standard deviation is thus proportional to the relevant Weber fraction and JNDs, which are available in the literature for many different stimulation types. Weber fractions and JNDs can be converted into standard deviations of psychometric functions and vice-versa. The JND is that difference in stimulus intensity that leads the participant to choose the correct stimulus in 75 % of the cases. Weber fractions are normalized versions of this value. Normalization is achieved by dividing it by the intensity of the standard stimulus. To obtain the standard deviation, convert JNDs first into Weber fractions. The Weber fraction is that distance to the mean where the psychometric function yields 25% or 75% correct

responses. With the Weber Fraction given, we thus need to determine the appropriate standard deviation given these constraints.

`SD_Standard` is then the standard deviation of the psychometric function for each stimulus intensity (`Multiplicator_SD_Standard * Mean_Standard`).

`Type_ResponseFunction` describes the function the stimulus strengths are chosen from by the method. At this moment, we have implemented normal distributions and Cauchy distributions, which can accurately depict the distribution of stimulus intensities presented with a staircase procedure. For a comparison between both options, see further below. Figure 2 visualizes different response distributions. A Gaussian distribution with an adequate standard deviation should be accurate enough for most intents and purposes when staircase procedures are used. The Cauchy distribution has more heavy tails and could be used if the starting values are relatively far away from the expected PSEs, and the initial step sizes are small. Stimulus presentation according to the Method of Constant Stimuli could be simulated with a uniform distribution, but is not currently implemented in the code.

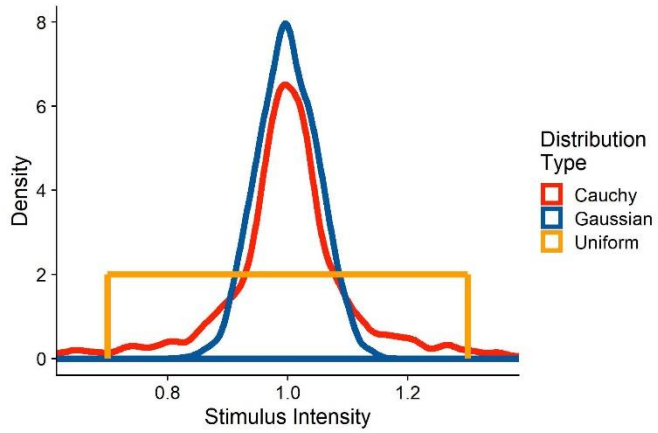


Figure 2: Two sample distributions of stimulus strengths, representative of stimulus intensities presented when using a staircase procedure. The red distribution corresponds to stimulus strengths drawn from a Cauchy function with a mode of 1 and a scale of 0.05. The blue distribution are responses drawn from a Gaussian distribution with a mean of 1 and a standard deviation of 0.1.

`SD_ResponseFunction` further describes the function the stimulus strengths are chosen from. For normal distributions, this value corresponds to its standard deviation; for Cauchy distributions, this corresponds to its scale; and for uniform distributions, this corresponds to a vector with the values tested.

We assume that there is between-participant variability in the means of the psychometric functions and that this variability is normally distributed. `Mean_Variability_Between` sets the standard deviation of the normal distribution these PSEs are drawn from. This normal distribution has a mean of 1 and works as a multiplier over `Mean_Standard`. that is, the standard deviation needs to be set accordingly.

We assume that there is also between-participant variability in the standard deviations of the psychometric functions. `SD_Variability_Between` sets the standard deviation of the normal distribution these standard deviations are drawn from. This normal distribution has a mean of 1, that is, the standard deviation has to be set accordingly.

Extracting the parameters from existing data

In the following, we will show with example pilot data from our lab how to extract the above values from an existing dataset. We collected these data in a velocity estimation task: Participants were shown two intervals of object motion in a 3D environment. One interval consisted in one big ball moving horizontally in front of the observer. The other consisted in a cloud of smaller balls moving in the same direction as the big target. The big ball moved at one of two speeds to the right (horizontal velocity signed positive) or to the left (horizontal velocity signed negative). The velocity of the ball cloud was controlled by a PEST staircase (Taylor & Creelman, 1967), with a slight adjustment: to get a more robust estimate of the JNDs, the stepsize did not change during the first 10 trials of each PEST. During the big target motion interval, the participant experienced visual self-motion in the same direction as the target ("congruent"), in the opposite direction of the target ("incongruent") or no self-motion at all ("no motion"). Participants then judged by button press which of the motions was faster.

The code and sample data are available on GitHub.

In the following, we provide the R code we use to compute the needed values to accurately simulate datasets. We first load the necessary packages: "dplyr" for data manipulation, "quickpsy" to fit psychometric functions and "MASS" for an way to determine response functions and the parameters of these response functions. When then set the working directory to the location of the script and read the Pilotdata.csv dataset, which should be located in the same directory as the script.

```
require(dplyr)
require(quickpsy)
require(MASS)

setwd(dirname(rstudioapi::getSourceEditorContext()$path))

Dataframe <- read.csv(header=TRUE,"PilotData.csv")
```

We then bring the data into the format needed for quickpsy: First, we indicate whether for each trial the participant judged the test stimulus to be faster or slower than the comparison stimulus (`Pest_Bigger`). We also compute the difference between test and comparison stimulus (`Difference`) and mark trials as `incongruent` (target and observer motion in opposite directions), `congruent` (target and observer motion in the same direction) and `no motion` (no observer motion). Then, we apply a very crude exclusion criterion by excluding all those trials where the test stimulus motion was more than two times faster than the comparison stimulus. We will furthermore only compare `incongruent` and `no motion` trials, as we limit these guidelines to comparing one baseline and one test condition.

```
Dataframe = Dataframe %>%
  mutate(
    Pest_Bigger = case_when(
      Response_Interval == Pest_Interval ~ 1,
      Response_Interval != Pest_Interval ~ 0),
    Difference = abs(velH_Pest)-abs(velH),
    Congruent = case_when(
      velH*velH_Subject < 0 ~ "incongruent",
      velH*velH_Subject > 0 ~ "congruent",
      velH*velH_Subject == 0 ~ "1no motion")) %>%
  filter(abs(velH_Pest) < abs(velH)*2 & Congruent != "1no motion")
```

While there are different methods to fit psychometric functions that each have their own benefits, we use a direction likelihood maximization method (Knoblauch & Maloney, 2012; Prins & Kingdom, 2016), implemented in the R package quicksy (Linares & López-Moliner, 2016). The bootstrap option is used to

compute confidence intervals, which allow for statistical comparisons. However, the quickpsy package currently does not include an option to estimate population-wide parameters. We thus deactivate the bootstrap option, which speeds up the fitting process significantly. We fit separate psychometric functions for each self-motion condition, participant and object velocity. Then, we can use the plot() function from base R to plot the psychometric functions.

```
PsychometricFunctions = quickpsy(Dataframe,Difference,Pest_Bigger,
               grouping = .(Congruent,participant,velH),
               bootstrap = "none")

plot(PsychometricFunctions)
```

From the `quickpsy` object, we can extract the estimates for means and standard deviations of the fitted cumulative Gaussians. We save means and standard deviations in separate tibbles.

```
PSEs = PsychometricFunctions$par %>%
  filter(parn == "p1" & Congruent != "congruent")

SDs = PsychometricFunctions$par %>%
  filter(parn == "p2" & Congruent != "congruent")
```

The PSE corresponds to the means of the fitted cumulative Gaussian functions. To get an estimate for `PSE_Difference`, we normalize the estimated mean for each condition by dividing it by the velocity of the comparison stimulus. We then take the mean of these values for `incongruent` and `no motion` conditions and subtract one from the other.

```
PSEs_Condition1_Absolute = (PSEs %>% filter(Congruent == "incongruent"))$par
velHs_Condition1 = abs((PSEs %>% filter(Congruent == "incongruent"))$velH)
PSEs_Condition1_Percentage = PSEs_Condition1_Absolute/velHs_Condition1
Mean_PSE_Condition1_Percentage = mean(PSEs_Condition1_Percentage)

PSEs_Condition2_Absolute = (PSEs %>% filter(Congruent == "1no motion"))$par
velHs_Condition2 = abs((PSEs %>% filter(Congruent == "1no motion"))$velH)
PSEs_Condition2_Percentage = PSEs_Condition2_Absolute/velHs_Condition2
Mean_PSE_Condition2_Percentage = mean(PSEs_Condition2_Percentage)

PSE_Difference = Mean_PSE_Condition1_Percentage-Mean_PSE_Condition2_Percentage
```

We follow the same procedure for `JND_Difference`. While the standard deviation of the fitted Cumulative Gaussian is not the same as the JND, they are proportional. Since `JND_Difference` is expressed as a percentage, the difference between standard deviation and JND in absolute values is not a problem.

```
SDs_Condition1_Absolute = (SDs %>% filter(Congruent == "incongruent"))$par
velHs_Condition1 = abs((SDs %>% filter(Congruent == "incongruent"))$velH)
SDs_Condition1_Percentage = SDs_Condition1_Absolute/velHs_Condition1
Mean_SD_Condition1_Percentage = mean(SDs_Condition1_Percentage)

SDs_Condition2_Absolute = (SDs %>% filter(Congruent == "1no motion"))$par
velHs_Condition2 = abs((SDs %>% filter(Congruent == "1no motion"))$velH)
SDs_Condition2_Percentage = SDs_Condition2_Absolute/velHs_Condition2
Mean_SD_Condition2_Percentage = mean(SDs_Condition2_Percentage)

JND_Difference = Mean_SD_Condition1_Percentage-Mean_SD_Condition2_Percentage
```

`Mean_Standard` is the mean PSE across participants for the `no motion` condition, after normalizing it by adding the mean target velocity and dividing this sum by the mean target velocity, and `Multiplicator_SD_Standard` is the mean standard deviation across participants for the `no motion` condition, again after normalizing. We already computed these values above:

```
Mean_Standard = mean(PSEs_Condition2_Percentage)
Multiplicator_SD_Standard = mean(SDs_Condition2_Percentage)
```

Similarly, we can use the values from above to get the between-participant variability for PSEs and standard deviations of the psychometric functions:

```
Mean_Variability_Between = sd(PSEs_Condition2_Percentage)
SD_Variability_Between = sd(SDs_Condition2_Percentage)
```

To choose whether a Gaussian or a Cauchy function is more appropriate for `ResponseFunction` and determine their standard deviation or scale, respectively, we can use the `fitdistr()` function from the `MASS` package to determine the best fit for each PEST. To get the normalized value, we use the function to fit Gaussian and Cauchy functions to the quotient `velH_Pest/velH`, separately for each congruency condition, participant, and target velocity. We furthermore extract the loglikelihood from the fit as a measure of model fit. We subtract the loglikelihood for the Normal distribution from the loglikelihood for the Cauchy distribution. Higher loglikelihoods signify a better model fit. When this difference is positive, the Cauchy distribution makes for the better fit, and if it is negative, the Normal distribution makes for the better fit. We then take the median for each of these parameters across all conditions as final values for `SD_ResponseFunction`.

```
Dataframe %>%
  group_by(participant) %>%
  mutate(Scale_Cauchy = fitdistr(velH_Pest/velH,"cauchy")$estimate[2],
         SD_Normal = fitdistr(velH_Pest/velH,"normal")$estimate[2],
         loglikelihood_Cauchy = fitdistr(velH_Pest/velH,"cauchy")$loglik,
         loglikelihood_Normal = fitdistr(velH_Pest/velH,"normal")$loglik,
         loglikelihood_Difference = loglikelihood_Cauchy-loglikelihood_Normal) %>%

dplyr::select(participant,Scale_Cauchy,loglikelihood_Cauchy,SD_Normal,loglikelihood_Norma
l, loglikelihood_Difference) %>%
  slice(1) %>%
  ungroup() %>%
  summarise(median_Scale_Cauchy = median(Scale_Cauchy),
            median_SD_Normal = median(SD_Normal),
            median_loglike_CauchyMinusNormal = median(loglikelihood_Difference))

if (ResponseDistribution[3] > 0){
  SD_ResponseFunction = ResponseDistribution[1]
} else {
  SD_ResponseFunction = ResponseDistribution[1]
}
```

Please note that this procedure yields the same variability parameter for both conditions. That is, we assume that the precision is not vastly different between the conditions. For JND differences bigger than 25 %, it might be advisable to use different variability parameters for the baseline and the test condition.

The whole package, including code and sample data, is available on GitHub.

Simulating the Data

We can then proceed to simulating the dataset. You can use either the above procedure to extract the values from a dataset or make estimated guesses about each value based on the literature. The code is available on GitHub.

```
require(dplyr)
require(ggplot2)
require(quickpsy)
set.seed(45)

nParticipants = 5
ID = paste0("S0",1:nParticipants)
```

```
ConditionOfInterest = c(0,1)
StandardValues = c(5,8)
reps = 100
PSE_Difference = 0.1
JND_Difference = 0.25
Multiplicator_PSE_Standard = 0
Multiplicator_SD_Standard = 0.15
Type_ResponseFunction = "Cauchy"
SD_ResponseFunction = 0.1
Mean_Variability_Between = 0.2
SD_Variability_Between = 0.2
```

Next, we simulate one whole data set based on the above values. We first create a data frame with one row for each trial and participant.

```
Psychometric = expand.grid(ID=ID,
       ConditionOfInterest=ConditionOfInterest,
       StandardValues=StandardValues,
       reps = 1:reps)
```

Then, we draw multiplicators for PSEs and JNDs per subject, accounting for between-subject differences in biases and precision.

```
Psychometric = Psychometric %>%
  group_by(ID) %>%#
  mutate(PSE_Factor_ID = rnorm(1,1,Mean_Variability_Between),
         SD_Factor_ID = rnorm(1,1,SD_Variability_Between))
```

Omitting this step amounts to the assumption that the effect of interest is equally strong in each participant. This can be a valid assumption, but it should not be the default. Rather, the value chosen here should be justified, independently of whether it is zero or above zero. Next, we simulate means and standard deviations of the psychometric functions for each condition. We also add in between-subject variability for PSEs and standard deviations.

```
Psychometric = Psychometric %>%
  mutate(
    Mean_Standard = StandardValues+StandardValues*Multiplicator_PSE_Standard,
    SD_Standard = StandardValues*Multiplicator_SD_Standard,
    Mean = (Mean_Standard + (ConditionOfInterest==1)*Mean_Standard*PSE_Difference)*
                                                        PSE_Factor_ID,
    SD = abs(SD_Standard + (ConditionOfInterest==1)*SD_Standard*JND_Difference)*
                                                        SD_Factor_ID)
```

Then, we draw the stimulus strengths likely to be presented in our experiment. As mentioned above, this varies depending on the way the experiment is controlled. For staircase procedures, the responses are more akin to normal distributions with relatively low standard deviations or Cauchy distributions with low scales. A good way to determine the most appropriate function would be to plot the distribution of presented stimulus strengths for pilot data and compare them to different distributions.

```
if (Type_ResponseFunction == "normal"){

  Psychometric = Psychometric %>%
    mutate(
      staircase_factor = rnorm(length(reps),
                1,
                SD_ResponseFunction*(1+ConditionOfInterest*JND_Difference)))

} else if (Type_ResponseFunction == "Cauchy"){

  Psychometric = Psychometric %>%
    mutate(
      staircase_factor = rcauchy(length(reps),
                1,
                SD_ResponseFunction*(1+ConditionOfInterest*JND_Difference)))
```

```
} else{

  print("distribution not valid")
}
```

We then use these multipliers (`staircase_factor`) to compute the test stimulus strengths presented in the experiment (`Presented_TestStimulusStrength`). Lastly, we compute the difference between test stimulus and standard stimulus for each trial (`Difference`). Then, we compute the probability on each trial to judge the test stimulus intensity as higher (bigger, brighter, faster, …) by feeding the simulated test stimulus strengths in a cummulative Gaussian with the mean and the standard deviations calculated above. We then use this value (`AnswerProbability`) to simulate binary answers (`Answer`) by drawing responses from a Bernoulli distribution. **Error! Reference source not found.** illustrates the stimulated data set for five subjects, where both PSE and JND differ between conditions.

```
Psychometric = Psychometric %>%
  mutate(Presented_TestStimulusStrength = Mean*staircase_factor,
         Difference = Presented_TestStimulusStrength - StandardValues,
         AnswerProbability = pnorm(Presented_TestStimulusStrength,Mean,SD),
         Answer = as.numeric(rbernoulli(length(AnswerProbability),AnswerProbability))
  )
```

As a next step, we bring the data into the format necessary for the glmer() function: We first remove extreme outliers (e.g., by a simple criterion such as excluding trials in which the difference between test and standard stimulus was higher than half the standard stimulus strength), which are not unlikely to occur when the Cauchy function is used. Then, we compute the number of "Test stimulus intensity was higher" responses for each Condition and difference between test and comparison stimulus strength and the number of total observerations for each condition and difference in intensities.

```
Psychometric = Psychometric %>%
  filter(abs(staircase_factor-1) < 0.75) %>%
  group_by(ID,ConditionOfInterest,StandardValues,Difference) %>%
  mutate(Yes = sum(Answer==1),
  Total = length(ConditionOfInterest))
```

We also pack the above code in a function that allows to comfortably simulate full data sets based on the above parameters. This function can be found here (on GitHub).

Now, we can inspect these psychometric functions visually to verify whether the values chosen above give rise to the expected psychometric functions in terms of PSE and slopes. We use the quickpsy package (Linares & López-Moliner, 2016) to fit the psychometric functions and plot them with the ggplot2 package.

```
PsychometricFunctions = quickpsy(Psychometric,
       Difference,
       Answer,
       grouping = .(ConditionOfInterest,ID,StandardValues),
       bootstrap = "none")

plot(PsychometricFunctions) +
  scale_color_manual(name = "",
                     values = c(Red,BlauUB),
                     labels = c("Control","Experimental")) +
  xlab("Difference between Comparison and Test") +
  ylab("Probability to choose Test") +
  geom_vline(linetype = 2, xintercept = 0, color = "grey") +
  geom_hline(linetype = 2, yintercept = 0.5, color = "grey")
```

Figure 3 illustrates the simulated psychometric functions for the above values. The vertical lines indicate the PSE for each participant and stimulus strength. We can see that the PSEs for Condition of Interest: 1

are shifted towards the right. Furthermore, the curves for Condition of Interest: 1 are more shallow, indicating higher JNDs. Both is in line with the values we chose for our simulations.

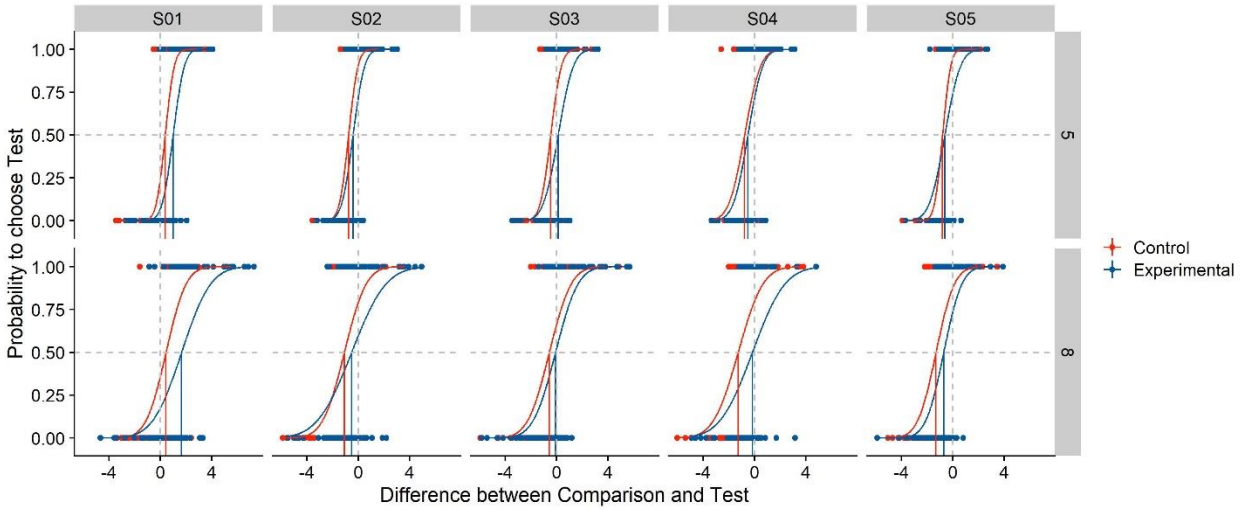


Figure 3: Simulated psychometric functions based on the example values chosen above. We plot the difference in stimulus intensity between test and standard stimulus (x axis) against the participants' probability to choose the test stimulus as more intense (y axis). Different panels are the psychometric functions per participant (columns) and per standard stimulus intensity (rows). The psychometric functions are color-coded blue for the experimental Condition of Interest, and red for the control condition without manipulation. The red and blue vertical lines indicate the Points of Subjective Equality, while the vertical and horizontal grey dashed lines denote a difference between test and comparison of 0, and a probability of 0.5 to choose either stimulus. Their intersection thus indicates perfect accuracy, with a PSE of 0. The curves are cumulative Gaussians fitted to the data, while the dots indicate the answer (0 or 1) for each trial.

Analyzing Psychophysical Data

In the introduction, we have briefly touched upon two different ways of applying Null Hypothesis Significance Testing to psychophysical data that are usually illustrated as psychometric functions:

- Fitting Psychometric functions, extracting PSEs and JNDs and conducting the appropriate statistical test over these values. This is usually a t test or an ANOVA, but for more accurate results, one can use Linear Mixed Models, which we will discuss below. We will call this the "Two-Step Approach".
- Following Moscatelli et al. (2012), one can also use Generalized Linear Mixed Models (GLMM) to extract population parameters directly without the intermediate step of fitting psychometric functions. This approach should, according to the authors, lead to a higher power for detecting these effects, at least for PSEs. We will call this the "GLMM Approach".

In the following, we will discuss R implementations, advantages and limitations of both methods and illustrate them with the data simulated above. The code snippets throughout this chapter [can be found here (GitHub).]

Two-Step Approach

How to – The first step for Two-Step Approaches is the estimation of the parameters of the psychometric functions with the best fits for the observed (or simulated) data. We first simulate a new dataset with the simulation function set up above and extract the means and standard deviation with the quickpsy package (Linares & López-Moliner, 2016). Quickpsy fits Cumulative Gaussian distributions by means of direct likelikihood maximization (Knoblauch & Maloney, 2012) and provides means and standard deviations of these Cumulative Gaussians as parameters, with means being interpreted as PSEs and standard deviations being interpreted as measure of sensitivity that is not equivalent, but proportional to JNDs.

```
set.seed(34)

Psychometric = SimulatePsychometricData(nParticipants = 10,
                                ConditionOfInterest = c(0,1),
                                StandardValues = c(5,6,7,8),
                                reps = 50,
                                PSE_Difference = 0.1,
                                JND_Difference = 0.25,
                                Multiplicator_PSE_Standard = 0,
                                Multiplicator_SD_Standard = 0.15,
                                Type_ResponseFunction = "normal",
                                SD_ResponseFunction = 0.1,
                                Mean_Variability_Between = 0.2,
                                SD_Variability_Between = 0.2)
PsychometricFunctions = quickpsy(Psychometric,
               Difference,
               Answer,
               grouping = .(ConditionOfInterest,ID,StandardValues),
               bootstrap = "none")

Parameters = PsychometricFunctions$par
Parameters2 = Parameters %>%
  filter(parn == "p1") %>%
  select(ID,ConditionOfInterest,Mean=par, StandardValues)
Parameters2$SD = Parameters$par[Parameters$parn == "p2"]
Parameters = Parameters2
```

Then, we compute a t-test over the computed PSEs and standard deviations to determine whether

```
Baseline_PSE = (Parameters %>% filter(ConditionOfInterest == 0))$Mean
CoI_PSE = (Parameters %>% filter(ConditionOfInterest == 1))$Mean
t.test(Baseline_PSE, CoI_PSE)
Baseline_SD = (Parameters %>% filter(ConditionOfInterest == 0))$SD
CoI_SD = (Parameters %>% filter(ConditionOfInterest == 1))$SD
t.test(Baseline_SD, CoI_SD)
```

Please not that these t tests are equivalent to Linear Regression Models with Mean or SD (respectively) as dependent variable and `ConditionOfInterest` as fixed effect.

```
LM_Mean = lm(Mean ~ ConditionOfInterest, Parameters)
LM_SD = lm(SD ~ ConditionOfInterest, Parameters)

summary(LM_Mean)
summary(LM_SD)
```

Discussion – This simple approach has three important pitfalls. First, statistical power is lost, especially when we have many trials for each condition, because this approach disregards that each PSE and standard deviation estimate is based on many trials and treats them as one measurement each. In our example script, each staircase is measured in 100 trials. With the Two-Step Approach, each of these

chunks of 100 trials are reduced to one mean and one standard deviation each, while there is no way to adjust for the added confidence in these values in a simple t test or ANOVA. Second, when participants and/or conditions contain different numbers of trials, this method still treats the means and standard deviations as equally reliable. In staircase procedures, participants or conditions tend to have unequal numbers of trials because often the staircases terminate once the step size has fallen under a certain threshold. Some participants may achieve such threshold substantially sooner or later than others. Therefore, each PSE and standard deviation is treated equally, despite one being based on more trials and therefore more reliable than the other. Third, one of the assumptions of an ANOVA is the independence of observations. This assumption is often violated in these analyses: mean and standard deviation estimates are more similar within each participant than across participants. This may not always be fatal, but a more careful modelling of the data is recommended. Mixed Effect Models allow to model these data structures more closely and provide a means to account for their partial dependence.

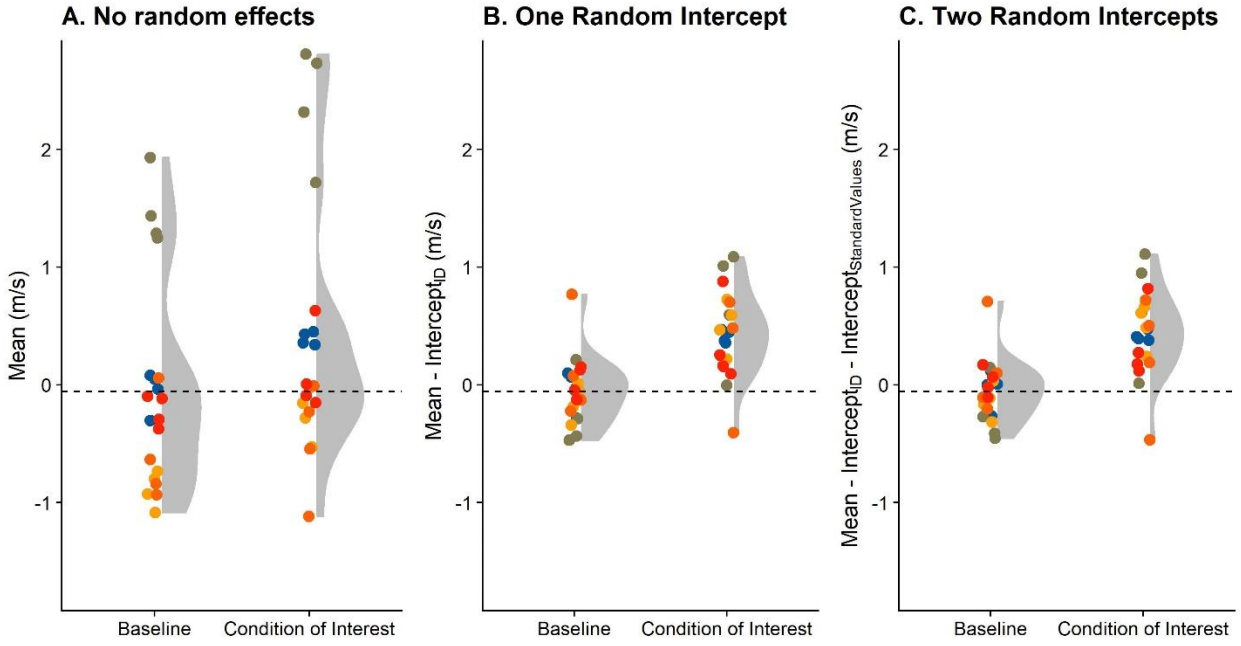


Figure 4: A. Simulated PSEs for five subjects (color-coded), for the baseline stimulus and the condition of interest, along with the distribution in grey. The dashed line corresponds to the mean PSE across participants in the baseline condition. B. As A., but after subtracting the random intercepts per ID computed with a Linear Mixed Model with these random effects. C. As A., but after subtracting the random intercepts per ID and the random intercepts per StandardValues computed with a Linear Mixed Model with these random effects.

Linear Mixed Models to the rescue – Mixed Effect Models are an extension of Linear Regression Models. Just like regular regression models, they fit straight lines through the data. Additionally, they can account for particular properties of sub-populations of the data. Effects that are examined across the whole population are called “fixed effects”, while these sub-group properties are called “random effects”. LMMs can accommodate a whole range of random effects. For example, in our case, LMMs can account for the fact that the psychometric functions may be shifted consistently to the left or to the right (equilvalent to lower or higher PSEs) for some participants, but not for others, or for that the fact that some participants are generally more sensitive than others, which in turn results in steeper psychometric functions and thus lower JNDs. In such cases of mean differences between groups, we speak of “random intercepts per

participant". With the inclusion of "random slopes" we can accommodate for between-participant differences in the effect of `ConditionOfInterest` or `StandardValues`, i.e., that participant s01 may be affected differently by `ConditionOfInterest` than participant s02. There is no generalizeable consensus on which random effects (random intercepts and/or random slopes) should be included and it generally regarded as situational decision to be taken based on domain knowledge about the data (Barr, Levy, Scheepers, & Tily, 2013; Bates, Mächler, Bolker, & Walker, 2015; Hodges, 2016; Matuschek, Kliegl, Vasishth, Baayen, & Bates, 2017). We recommend to include at least random intercepts per participant because there is usually a considerable variability between participants in psychophysical studies.

Figure 4 illustrates the difference between a regular t test and an LMM with random intercepts per participant graphically: a regular t test (equivalent to a Linear Model with PSEs as dependent variable and `ConditionOfInterest` as fixed effect) does not distinguish between participants (see Figure 4A), which leads to broader distributions overall. A Linear Mixed Model with `ConditionOfInterest` as fixed effect and random intercepts per participant allocates some of the variability in the data to idiosyncracies of each participant, which makes it easier to identify an overall consistent effect of `ConditionOfInterest`. Figure 4B illustrates this by plotting the same data shown in Figure 4A after subtracting the random intercepts per participant from the raw data. Speaking in a very simplified manner, rather than testing differences across all data, it would factor our this between-participant variable, and compute the difference between baseline and test condition on a much less variable residual dataset. The lme4 package (Bates et al., 2015) for R provides a user-friendly interface for the fitting of Linear Mixed Models. In lme4 syntax, the model specification looks like the following:

```
LMM_Mean_ID = lmer(Mean ~ ConditionOfInterest + (1 | ID),
         data = Parameters)
summary(LMM_Mean_ID)

LMM_SD_ID = lmer(SD ~ ConditionOfInterest + (1 | ID),
           data = Parameters)
summary(LMM_SD_ID)
```

This adjudicates some of the variability to differences between each participant and lowers the standard errors associated with the main effects, thus raising power. For example, for the data presented in Figure 4, a t test yields a p value of 0.13, while the equivalent Linear Regression Model provides a regression coefficient of 0.49, an associated standard errror of 0.31 and, as expected, the same p value of 0.13 for the effect of `ConditionOfInterest` on the PSE. A Linear Mixed Model that also adds random intercepts per participant (**LMM_Mean_ID**), in turn, yields the same regression coefficient 0.49) and a much lower associated standard error of 0.1. When the lmerTest package is loaded, calling the summary() function on the LMM object also provides a p value, based on degrees of freedom approximated with the Satterthwaite method (Kuznetsova et al., 2017). We will discuss another method for significance testing, Likelihood Ratio Tests, below. For our synthetic dataset, this estimated p value is0.00006. The Mixed Model analysis thus provides the same coefficient estimate, but a much lower standard error and a much lower p value. As for the random effects, it yields intercepts of -0.02, 1.72, -0.75, -0.71 and -0.25 for participants S01 through s05, which – as expected – corresponds roughly to these participants' average baseline performance (see Figure 4A). We find similar differences for the standard deviations: The Linear Regression Model yields a regression coefficient of 0.32, a standard error of 0.15 and a p value of 0.04 for the effect of `ConditionOfInterest` on the standard deviation. A Linear Mixed Model, in turn, returns a coeffcient of 0.32, a standard error of 0.13 and a p value of 0.19.

Please note that a pairwise t test with participants as grouping factor is roughly equivalent to the Linear Mixed Model (LMM) established above. However, Mixed Models have some advantages over such t tests: There are fewer issues with missing data and they are more flexible. For example, they allow the inclusion of more than one grouping factor, and more than one random effect per grouping factor, which can come in handy. We therefore recommend using Linear Mixed Models over pairwise t tests. For further information and an R-based introduction to Linear Mixed Models, please see Brown's guide (Brown, 2020).

In fact, the specification of the Mixed Model requires further discussion. For the standard deviations, Weber's Law dictates that higher `StandardValues` should elicit higher standard deviations in a lawful manner. We could thus include `StandardValues` as additional grouping factor. The lme4 syntax would then be:

```
LMM_Mean2 = lmer(Mean ~ ConditionOfInterest + (1 | ID) + (1 | StandardValues),
                 data = Parameters)
LMM_SD2 = lmer(SD ~ ConditionOfInterest + (1 | ID) + (1 | StandardValues),
               data = Parameters)
summary(LMM_Mean)
summary(LMM_SD)
```

We show the coefficients, their associated standard errors and the p values in Table 1. As you can see, including random intercepts for `StandardValues` leads to lower standard errors both for the PSE and the SD models, along with the resulting lower p values.

Two unaddressed issues – However, when using this more sophisticated version of the Two-Step approach, two issues remain unsolved:

(1) We might still lose power by disregarding that each PSE and JND estimate is based on a large number of trials (e.g., 100 in our case)
(2) PSEs and JNDs are still treated as equally reliable, even if one is based on a staircase with 25 trials and the other is based on a staircase with 60 trials.

Moscatelli et al. (Moscatelli et al., 2012) have suggested the use of Generalized Linear Mixed Models to avoid both issues, which we will discuss in the following section.

	Regression Coefficient	**Standard Error**	**P value**
PSE: LM *(equivalent to t test)*	0.49	0.31	0.129
PSE: LMM with one random effect *(random intercepts for ID)*	0.49	0.11	< 0.001
PSE: LMM with two random effects *(random intercepts for ID and StandardValues)*	0.49	0.1	< 0.001
SD: LM *(equivalent to t test)*	0.32	0.15	0.038
SD: LMM with one random effect *(random intercepts for ID)*	0.32	0.13	0.02
SD: LMM with two random effects *(random intercepts for ID and StandardValues)*	0.32	0.11	0.008

Table 1: Regression coefficients, associated standard errors and p values for different ways of analyzing the data.

Generalized Linear Mixed Modelling (GLMM)

How to – We recommend Moscatelli et al.'s paper (Moscatelli et al., 2012) for a more thorough discussion of the specifics of this method, both in terms of rationale and process. In this manuscript, we will only outline the basic idea. Our goal is to extract population coefficient estimates directly from the proportional or binomial data obtained in the experiment (i.e., the proportion of "Test Stimulus more intense" responses for each presented stimulus strength). Linear Mixed Models, however, (a) require continuous dependent variables and (b) can only fit straight lines. (a) is problematic because responses from psychophysical paradigms are generally either proportional or binomial. (b) is problematic because it is generally assumed that performance in such experiments is depicted accurately by psychometric functions, which are typically non-linear functions like Weibull, Cummulative Gaussian or Logistic functions. Generalized Linear Mixed Modelling circumvents these two problems by introducing a so called "link function" that transforms proportional/binomial data with a non-linear relation between independent and dependent variable(s) such that they can be captured by a linear fit. The relevant link functions that lme4 offers are "probit" and "logit" links. Both allow to transform the typical psychometric function types, Weibull, Cummulative Gaussian or logistic functions, into the linear space. We follow Moscatelli et al. in choosing a probit link, but a logit link yields practically the same results.

We reuse the same dataset we already asimulated above. To fit Generalized Linear Mixed Models, we have to make several decisions with regards to the specification of the model, particularly about the random effects. Moscatelli et al. (Moscatelli et al., 2012) use two different models in their examples: A model with `ConditionOfInterest`, `Difference` and their interaction as fixed effects, and random intercepts and random slopes for `Difference` per participant, and the same model, but with only random intercepts per participant. These are specified as follows in lme4 syntax:

```
GLMM_RandomIntercepts_JND = glmer(cbind(Yes, Total - Yes) ~
ConditionOfInterest*Difference + (1| ID),
             family = binomial(link = "probit"),
             data = Psychometric)

GLMM2_RandomInterceptsAndSlopes_JND = glmer(cbind(Yes, Total - Yes) ~
ConditionOfInterest*Difference + (1 + Difference| ID),
             family = binomial(link = "probit"),
             data = Psychometric)

summary(RandomIntercepts)
summary(RandomInterceptsAndSlopes)
```

`Difference` indicates to what extent the proportion of responses changes in response to the difference in stimulus strength between stimulus 1 and stimulus 2. The coefficient for this variable is thus proportional to the standard deviation of the psychometric function and the JND. `ConditionOfInterest` indicates to what extent the whole psychometric function is shifted to the left or to the right. The coefficient for `ConditionOfInterest` thus corresponds to the extent to which the presence of the manipulation shifts the mean of the psychometric function, that is, the PSE. The interaction between `ConditionofInterest` and `Difference` indicates to what extent the manipulation `ConditionOfInterest` changes the coefficient for `Difference`. This corresponds to the influence of the manipulation on the standard deviation of the psychometric function, and with that, its JND. The random effects (Intercepts per `ID` in the first model, and Intercepts and Slopes for `Difference` per `ID` in the second model) allow to account for individual differences in PSE and JND per participant.

When the lmerTest package (Kuznetsova et al., 2017) is loaded, calling the summary() function on the GLMER object automatically provides p values for each fixed effect coefficient; in this case, for the Intercept, `ConditionOfInterest`, `Difference` and their interaction. These p values are based on degrees of freedom approximated with the Sattherwaite method, which is generally considered a computationally cheap, reliable-enough, but imperfect procedure (Gaylor & Hopper, 1969). An alternative, which can be more accurate, are Likelihood Ratio Tests (Luke, 2017). A Likelihood Ratio Test compares a test model to the next simpler model that doesn't include the variable of interest in terms of their respective log-likelihoods, with a penalty for the additional parameter in the test model. The p value is then based on a model fit criterion. If we want to test for the impact of `ConditionOfInterest` on the JND, this null model would be a model that is equal to the test model, but lacks the interaction of `ConditionOfInterest` and `Difference`. The lme4 syntax would be:

```
GLMM_RandomIntercepts_Null_JND = glmer(cbind(Yes, Total - Yes) ~ ConditionOfInterest +
Difference + (1| ID),
                               family = binomial(link = "probit"),
                               data = Psychometric)
GLMM2_RandomInterceptsAndSlopes_Null_JND = glmer(cbind(Yes, Total - Yes) ~
ConditionOfInterest + Difference + (Difference| ID),
                                        family = binomial(link = "probit"),
                                        data = Psychometric)
```

The Likelihood Ratio Test is implemented in the `anova()` function from base R.

```
anova(GLMM_RandomIntercepts_JND,GLMM_RandomIntercepts_Null_JND)
anova(GLMM2_RandomInterceptsAndSlopes_JND,GLMM2_RandomInterceptsAndSlopes_Null_JND)
```

This Likelihood Ratio Test provides a p value which allows to judge whether the test model is significantly better than the null model. If it is, this is evidence that `ConditionOfInterest` has a significant influence on the slope of the psychometric function, that is, on the JND.

Testing for PSE differences requires a slightly different approach in model specification. Since Likelihood Ratio Testing requires isolating the effect to be tested in the test model, in order to drop it in the null model, the interaction between the fixed effects `ConditionOfInterest` and `Difference` can't be included:

```
GLMM_RandomIntercepts_PSE = glmer(cbind(Yes, Total - Yes) ~ ConditionOfInterest +
Difference + (1| ID),
                               family = binomial(link = "probit"),
                               data = Psychometric)
GLMM2_RandomInterceptsAndSlopes_PSE = glmer(cbind(Yes, Total - Yes) ~ ConditionOfInterest
+ Difference + (Difference| ID),
                                        family = binomial(link = "probit"),
                                        data = Psychometric)
```

The corresponding null models should be specified as follows:

```
GLMM_RandomIntercepts_Null_PSE = glmer(cbind(Yes, Total - Yes) ~ Difference + (1| ID),
                                    family = binomial(link = "probit"),
                                    data = Psychometric)
GLMM2_RandomInterceptsAndSlopes_Null_PSE = glmer(cbind(Yes, Total - Yes) ~ Difference +
(Difference| ID),
                                             family = binomial(link = "probit"),
                                             data = Psychometric)
```

And finally, a Likelihood Ratio Test is performed over both models, which yields a p value.

```
anova(GLMM_RandomIntercepts_PSE,GLMM_RandomIntercepts_Null_PSE)
anova(GLMM2_RandomInterceptsAndSlopes_PSE,GLMM2_RandomInterceptsAndSlopes_Null_PSE)
```

A challenge when using Generalized Linear Mixed Models is the model specification, in this case particularly the decision about the random effects to be included into the model. Moscatelli et al.'s (Moscatelli et al., 2012) analyses suggest that including (a) only random intercepts per participant or (b) both random intercepts and random slopes for `Difference` per participant doesn't lead to vastly different results. We can test this also for our simulated dataset:

```
anova(GLMM_RandomIntercepts_JND,GLMM2_RandomInterceptsAndSlopes_JND)
anova(GLMM_RandomIntercepts_PSE,GLMM2_RandomInterceptsAndSlopes_PSE)
```

The model fit was superior for (a), but the fixed effect coefficient estimates were roughly equal across both options. In our simulated dataset, a Likelihood Ratio Test yielded significant difference between (a) and (b), for neither of the models specified above, in which case the model with fewer parameters (`GLMM_RandomIntercepts_JND` and `GLMM_RandomIntercepts_PSE`, respectively), is preferrable.

Again, all code used in this section can be found on GitHub.

A deep dive into model specifications

A further method to determine the correct model specification is to study to what extent each meets the assumptions of Generalized Linear Mixed Models: (1) Homoskedasticity and (2) roughly uniformly distributed residuals. *Homoskedasticity* refers to the postulate that variance in responses should not depend in any lawful way on any experimental variables.

Homoskedascticity

For Linear Models, homoskedasticity is generally tested for by visual examination of plots where different conditions are plotted against the corresponding residuals. A statistical test for difference in variances, such as Levene's test (Levene, 1960), Bartlett's test (Bartlett, 1937) or the Brown-Forsythe test (Brown & Forsythe, 1974) can be applied, but visual inspection is usually considered sufficient. However, it is near impossible to visually ascertain heteroskedasticity from the raw residuals of Generalized Linear (Mixed) Models, as the expected pattern in the residuals depends on the fitted values. The R package DHARMa (Hartig, 2020) implements a method to standardize residuals for Generalized Linear Mixed Models (Dunn & Smyth, 1996; Gelman & Hill, 2008). These standardized residuals allow for visual analysis just like residuals of Linear Models; for a well specified model, one would expect these standardized residuals to be uniformly distributed. To judge whether heteroscedasticity might be a problem in the analysis of psychophysical data with GLMMs, we use the above procedure to produce extreme data, with `StandardValues` of 5, 10 and 50 m/s, and (high) inter-participant variability both for PSE and JND of 0.5, and plot the DHARMa-standardized residuals against each of the sub-groups in our data (`ConditionOfInterest`, `StandardValues`, `ID`; see Figure 5), for the simplest reasonable model GLMM_RandomIntercepts_JND and for the most complex reasonable model, including random intercepts and slopes for `Difference` and `ConditionOfInterest` both for `ID` and `StandardValues` (`GLMM2_ThreeRandomEffectsPerIDAndStandardValues`, see below).

```
require(DHARMa)
Sim_Simple = simulateResiduals(GLMM_RandomIntercepts_JND)

GLMM2_ThreeRandomEffectsPerIDAndStandardValues = glmer(cbind(Yes, Total - Yes) ~
ConditionOfInterest*Difference +
(Difference + ConditionOfInterest| ID) +
(Difference + ConditionOfInterest| StandardValues),
                                                family = binomial(link = "probit"),
```

```
                                                    data = Psychometric)
Sim_Complex = simulateResiduals(GLMM_RandomIntercepts_JND)
```

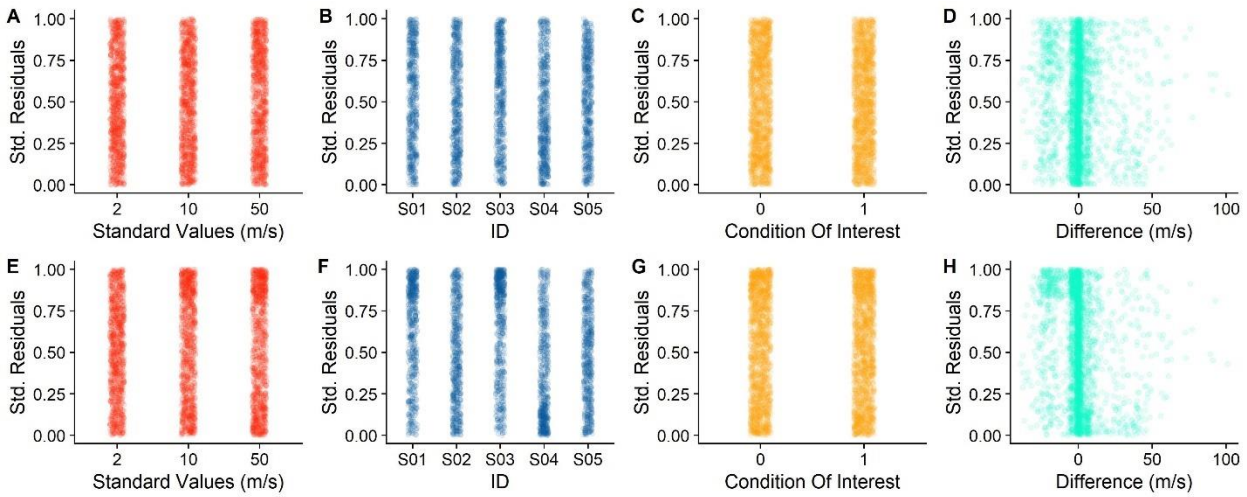


Figure 5. A, B, C, D. Standardized Residuals for the simple model, divided up by Standard Values (A), participant (B), Baseline versus Condition of Interest (C) and Differences (D). E, F, G, H. Same, but for the more complex model.

As evident from Figure 5, there are no notable patterns for any of factors. For the continuous variable `Difference`, there seems to be a slight pattern of negative values being connected to higher residuals. This pattern, however, is quite weak for the simplest model, and, contrary to the expectancy that a more complete model should reduce residual patterns, this pattern is stronger for the more complex model. However, the variability in residuals seems to be evenly distributed across `StandardValues`, `ID` and `ConditionOfInterest`.

This analysis is included in the same script as above (GitHub).

Patterns in residuals

Linear Models can furthermore be unreliable if the underlying data structure is not modelled correctly. This can be detected by plotting the values predicted by the models against the observed values, as a quantile-quantile (QQ) plot. Alternatively, the residuals should present no apreciable patterns. Statistical tests, like a Kolmogorov-Smirnoff test (Massey, 1951), can be performed to detect whether the predicted values match the observed values closely enough, but are not generally deemed necessary. To verify whether this criterion is satisfied by some model specifications, but not by others, we simulate five average datasets with 10 participants each (using the above procedure), analyze them with the 25 reasonable model specifications described in Table 2 and verify visually for each if the residuals are distributed uniformly. For computation and visualization of the standardized residuals, we again rely on the DHARMa package (Hartig, 2020). Figure 6 illustrates the residuals for one representative repetition. As you can see, some patterns are left in the residuals for nearly all models. Model M16 may be favoured slightly, but the difference with regards to the other models is very small.

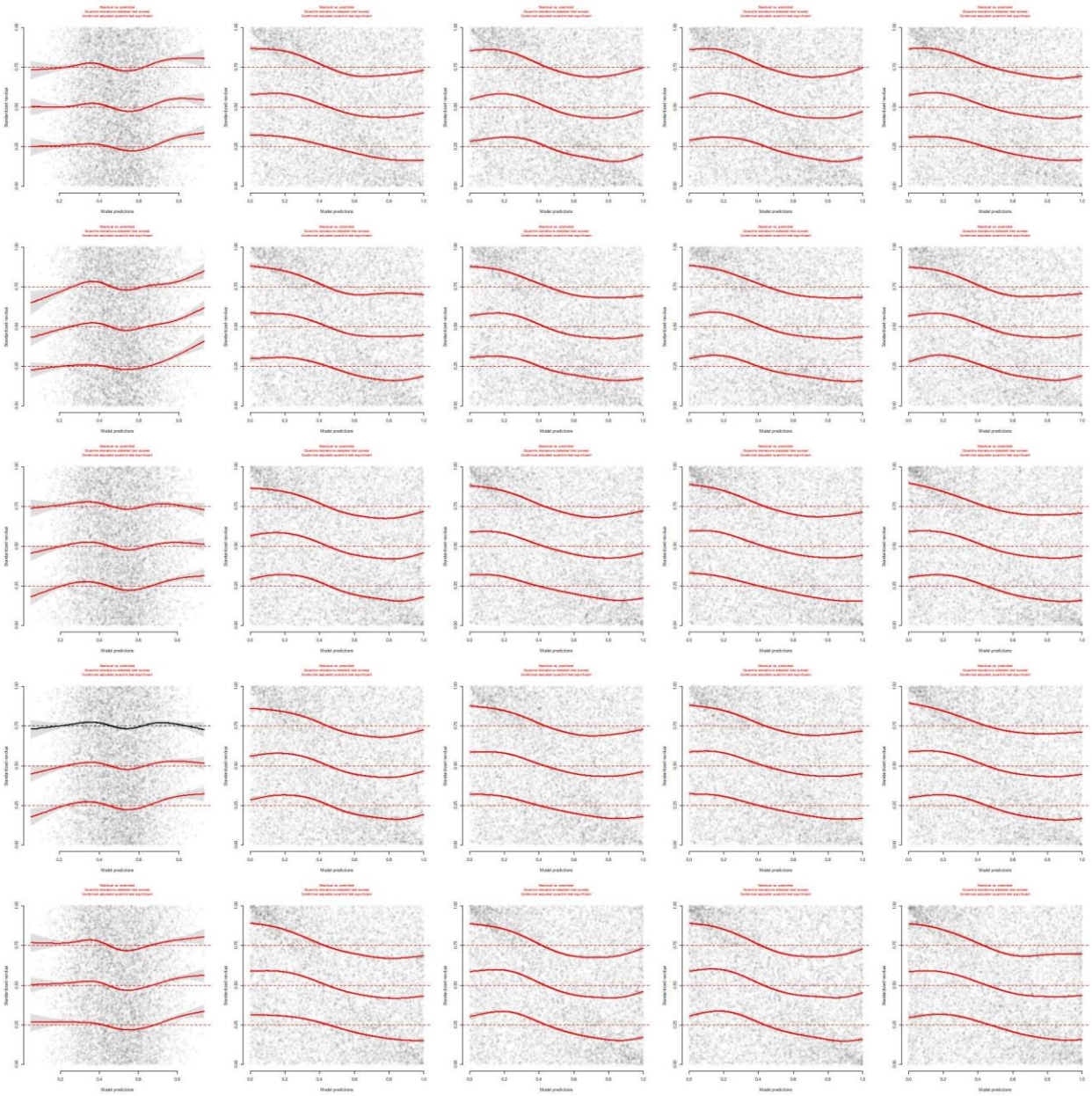


Figure 6: Residuals for Model M1-M25 (see Table 2), generated with the R package DHARMa for one simulated dataset.

Please find the code used for this analysis here (GitHub).

Recovery of coefficients, standard errors and model fits; power and false positives

Furthermore, and maybe most importantly, we can verify to what extent each model specification recovers the correct population parameters. Since we simulate the datasets, we know its underlying properties and can compare them to the output of each model specification. To this end, we simulate 200 datasets and analyze them with each of the 25 model specifications in Table 2. We choose "`nAGQ = 1`" and "`glmerControl(optimizer = "nloptwrap"))`" as options in the `glmer()` function, the rationale for which we

will discuss further below (see heading “A comparison of several model fitting parameters”). We repeat this for six different combinations of PSE and JND differences:

(1) a PSE difference of 1.25% & a JND difference of 8%
(2) a PSE difference of 1.25% & a JND difference of -8%
(3) a PSE difference of -1.25% & a JND difference of 8%
(4) a PSE difference of -1.25% & a JND difference of -8%
(5) a PSE difference of 1.25% & a JND difference of 0%
(6) a PSE difference of 0% & a JND difference of 8%

We then extract the most relevant features (PSEs & standard deviations of the psychometric functions, standard errors for the coefficient estimates, AICs and p values) and compare them to the values choosen for the simulations. The Supplementary Figures 2 – 25 show the most important results of this process. Supplementary Figures 2-7 show the difference between the estimated PSEs and standard deviations of the psychometric functions, Supplementary Figures 8-13 show the corresponding standard errors. Supplementary Figures 14-19 show the power (fraction of p values below 0.05) for each model and each combination of PSE and JND differences. And Supplementary Figures 20-25 show the AICs (a measure of model fit).

The Supplementary Figures 2-7 show that all models across all conditions recover the PSE difference between Baseline and Condition of Interest correctly on average. Models without random intercepts per ID (M01, M06, M11, M16 and M21) have, however, an extremely high variability in these recovered values and are thus very likely to recover inaccurate values. The same is true for models without random slopes for `Difference` per `StandardValues`, albeit to a much lower extent (M01-M10 and M16-M20).

For the JND difference between Baseline and Condition of Interest, fewer models recover the correct value correctly on average across all PSE and JND differences (M13, M14, M15, M23, M24 and M25). M15 and M25 have the lowest variability and thus the lowest risk of recovering an inaccurate value.

In terms of power (Supplementary Figures 14-19), both M15 and M25 perform very similarly across the board. For the PSE difference between Baseline and Condition of Interest, M15 provides a false positive rate (the rate of significant results when no effect is simulated) slightly above the alpha cutoff of 0.05, while M25 provides the expected false positive rate of 0.05. When an effect is simulated, both M15 and M25 provide high power. For the JND, the false positive rate is slightly above 0.05 for both M15 and M25, while M14 and M24 provide the expected false positive rate of 0.05. Both M15 and M25 have an equally high power when an effect is present.

In terms of model fits, M15 and M25 perform the best (Supplementary Figures 19-24), while M13 and M23 perform nearly as well, with M14 and M24 slightly behind. The AIC (Akaike Information Criterion) indicates the fit of a statistical model to the data. Lower values indicate a better model fit.

Finally, among these front runner models, M14 and M24 provide slightly lower standard errors than M15 and M25 for the PSE difference between Baseline and Condition of Interest, while M14 and M15 beat M24 and M25 in terms of providing the narrowest standard errors for the JND difference. A lower standard error should be associated with higher power.

Model name	**Random Effects**	
	Grouping Variable: StandardValues	**Grouping Variable: ID**

M01	-	-
M02	-	Intercept
M03	-	Intercept, Difference
M04	-	Intercept, ConditionOfInterest
M05	-	Intercept, Difference, ConditionOfInterest
M06	Intercept	-
M07	Intercept	Intercept
M08	Intercept	Intercept, Difference
M09	Intercept	Intercept, ConditionOfInterest
M10	Intercept	Intercept, Difference, ConditionOfInterest
M11	Intercept, Difference	-
M12	Intercept, Difference	Intercept
M13	Intercept, Difference	Intercept, Difference
M14	Intercept, Difference	Intercept, ConditionOfInterest
M15	Intercept, Difference	Intercept, Difference, ConditionOfInterest
M16	Intercept, ConditionOfInterest	-
M17	Intercept, ConditionOfInterest	Intercept
M18	Intercept, ConditionOfInterest	Intercept, Difference
M19	Intercept, ConditionOfInterest	Intercept, ConditionOfInterest
M20	Intercept, ConditionOfInterest	Intercept, Difference, ConditionOfInterest
M21	Intercept, Difference, ConditionOfInterest	-
M22	Intercept, Difference, ConditionOfInterest	Intercept
M23	Intercept, Difference, ConditionOfInterest	Intercept, Difference
M24	Intercept, Difference, ConditionOfInterest	Intercept, ConditionOfInterest
M25	Intercept, Difference, ConditionOfInterest	Intercept, Difference, ConditionOfInterest

Table 2: Specification of different models. The first column indicates the model names, the second column indicates the random effects per StandardValues and the third column shows the random effects per ID.

Conclusions – Overall, despite the slightly elevated false positive rate, M15 (with random intercepts and random slopes for `Difference` per `StandardValues` and random intercepts and random slopes for `Difference` and `ConditionOfInterest` per `ID`, along with `Difference`, `ConditionOfInterest` and their interaction as fixed effects) seems to be the best model specification for our purposes: it recovers accurate and precise estimates, associated with low standard errors, high power and the lowest AIC values.

Please find the code used for simulations here (GitHub) and for the plots here (GitHub). You can download the results of the (time consuming) simulations here (OSF).

Comparing the power for the GLMM and the Two-Step approach

How do the GLMM approach and the Two-Step approach compare, then? We repeat the same procedure as above to compare the best four GLMM models (M14, M15, M24, M25) to the Linear Mixed Model-based Two-Step solution we propose above. As described above, the Two-Step approach consists of obtaining PSEs and JNDs by fitting psychometric functions and then using Linear Mixed Modelling for significance testing over these PSEs and JNDs. The Linear Mixed Models used here have, respectively, the fitted PSEs and JNDs, as dependent variable, `ConditionOfInterest` as fixed effect and random intercepts per `ID` and `StandardValues`. As above, we compare the parameters used for simulations to the values

recovered by the models, and report on the power of each version. The plotted results of these simulations can be found in the supplementary materials (Supplementary Figure 11 and Supplementary Figure 12). Briefly described, we find that all tested models recover the PSEs and JNDs accurately, while the precision with which the PSEs are recoverd is higher for all GLMMs than for the LMMs, and the precision with which the JNDs are recovered is highest for the LMM and M15 and M25. The power to detect PSE differences is slightly higher for the GLMMs than for the LMM, and the power to detect JND differences is slightly higher for the LMM than for the GLMMs. The code for these simulations is available here (on GitHub), and the code for the supplementary figures can be found here (GitHub). The results of the simulations can be downloaded here (OSF).

To settle the exact circumstances under which each approach is favored, we furthermore perform one full power analysis with the best GLMM model (M15) and the LMM-based Two-Step approach. We use the same parameters as above, with 10, 12, 14, 16, 18, 20 participants, and 40, 70 and 100 trials per staircase. We furthermore compare different effect sizes, with JND differences between baseline and Condition of Interest of -10, -5, 0, 5 and 10% and PSE difference of -2.5, -1.25, 0, 1.25 and 2.5%. We furthermore include two different ways of analyzing the data: the Linear Mixed Modelling approach to Two-Step analyses, and the Generalized Linear Mixed Modelling approach with the specification we have identified above as optimal (M15 in Table 2).

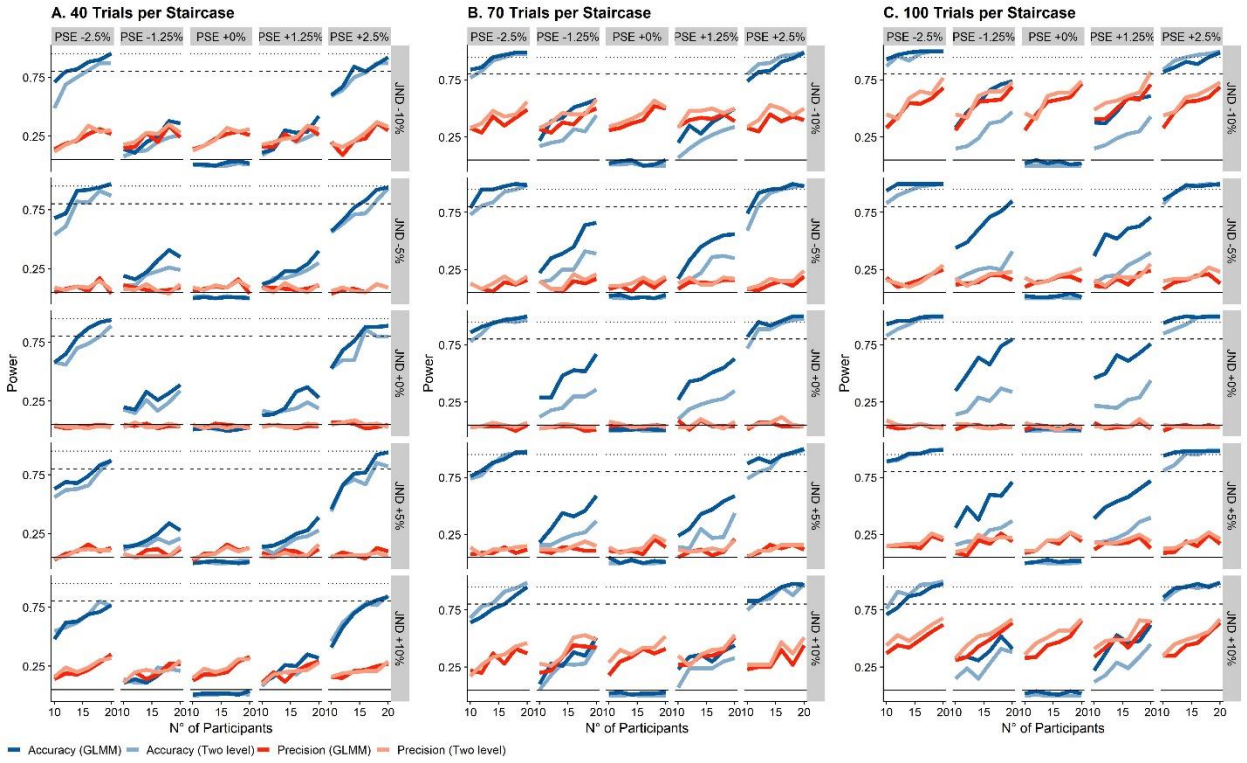


Figure 7: The solid lines indicate the power level across the range of participant numbers, red for accuracy and blue for precision. The darker shades denotes the GLMM approach, while the lighter shade denotes the Two-Step approach. The rows of panels in each subplot denote different differences in JND between test and comparison (between -10% and +10%), and the columns of panels show different differences in

PSE between test and comparison (between -2.5% and + 2.5%). The intermittent horizontal lines indicate power levels of 0.8 (bare minimum), 0.9 (acceptable) and 0.95 (quite good). A., B. and C. show all of the above for 40, 70 and 100 trials per staircase, respective.

As evident from Figure 7, the GLMM method (dark blue line) has a higher power for detection of PSE difference across all combinations of parameters than the Two-Step approach (light blue line). The difference is larger for larger trial numbers per staircase, but even for the smallest number we simulated (40 trials), the GLMM has a slightly higher power. For JND differences, the difference in power is miniscule, while favoring the Two-Step approach (light red line) ever so slightly over the GLMM approach (darker red line). Interestingly, this difference is stable across all trials per staircases we simulated.

The code used for simulation is available here (GitHub), the code used to generate Figure 7 is available here (GitHub) and the data obtained for the lengthy simulations is available here (GitHub).

Bayesian Linear Mixed Modelling

Finally, there are Bayesian methods of estimating JND and PSE that rely on Monte Carlo Markov Chain (MCMC) modelling. Particularly the brms package (Bürkner, 2018) for R is an interesting alternative that, due to its similar syntax, is very user-friendly for lme4 users. brms offers an interface for the package rstan (Stan Development Team, 2016), which is based on the probabilistic programming language Stan. The usage of brms is very similar to lme4, but users are advised to be familiar with Bayesian statistics before analyzing data with brms. The brms documentation provides an excellent starting point, especially if you already have some general knowledge about Bayesian analysis, but it focusses on applying the package correctly and is therefore by no means complete. For a more exhaustive reading, we recommend, for example, Spiegelhalter & Reis' book on Bayesian Statistics (Spiegelhalter & Rice, 2009).

To fit a Bayesian Generalized Linear Mixed Model that resembles the ones discussed above in structure, we first install and load the packages rstan (Stan Development Team, 2016) and brms (Bürkner, 2018), and optimize the system for the fitting of Bayesian GLMMs.

```
require(brms)
require(rstan)
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
Sys.setenv(LOCAL_CPPFLAGS = '-march=corei7')
```

brms allows Bayesian analysis based on the same principles as the GLMM approach described above. We can fit a model with `ConditionOfInterest`, `Difference` and their interaction as main effects, and random intercepts and slopes for `Difference` per participant `ID` and `StandardValues` as random effects.

```
BayesianGLMM = brm(bf(Yes ~ ConditionOfInterest*Difference + (ConditionOfInterest +
Difference | ID) + (ConditionOfInterest + Difference | StandardValues)),
                   data = Psychometric,
                   family = bernoulli(link = logit))
```

We can call the fitted object and examine the results.

```
summary(BayesianGLMM)
```

Finally, we can use the hypothesis() function to test which of two mutually exclusive hypotheses the data support and to what extent; for example whether the coefficient for `ConditionOfInterest` is larger than zero:

```
hypothesis(BayesianGLMM,c("ConditionOfInterest > 0"))
```

While Monte Carlo Markov-Chain modelling, the principle on which brms is based, is often considered the most accurate means to estimate parameters (Bolker et al., 2009), it also comes with a Bayesian flavour that requires a different lense for interpretation and may thus not be suitable for everyone. Nonetheless, we recommend that readers keep this option in mind.

Power Analyses for Psychophysical Data

To simulate the power with a given set of parameters, we need to execute the above procedure sufficient times (we recommend at least 1000 times, although this might be too time consuming in R for studies with a high count of subjects and/or trials and; we will discuss a potentially faster Julia implementation below), and calculate the ratio of simulations in which the p value associated with `ConditionOfInterest` (for the PSE) or the interaction between `ConditionOfInterest` and `Difference` (for the JND) is below a certain alpha (typically 0.05). This requires that you have the data file `SimulatePsychometricData()` (to be found here on GitHub) in the same folder as the script below. The following code is available here (on GitHub).

First, we set the relevant parameters. `RangeNs` is a vector with the number of participants for which we want to simulate the power (10, 12, 14, 16, 18 and 20 in our case), and `RangeRepetitions` is a vector with the number of trials per staircase for which we want to simulate the power (40, 70 and 100 in our case). Varying both allows us to find an optimal tradeoff between trials per participant and number of participants.

```
source(paste0(dirname(rstudioapi::getSourceEditorContext()$path),
              "/SimulateDataFunction.r"))
require(dplyr)
require(purrr)
require(lme4)
require(ggplot2)
set.seed(65)


RangeNs = c(10,12,14,16,18,20)
RangeRepetitions = c(40,70,100)

ConditionOfInterest = c(0,1)
StandardValues = c(5,6,7,8)
PSE_Difference = 0.1
JND_Difference = 0.25
Multiplicator_PSE_Standard = 0
Multiplicator_SD_Standard = 0.15
Type_ResponseFunction = "normal"
SD_ResponseFunction = 0.1
Mean_Variability_Between = 0.2
SD_Variability_Between = 0.2
```

We then choose a number of iterations per combination of number of repetitions and number of participants. We recommend a high number, but even with a relatively low number of 200 iterations this process can take several hours. To keep track of the overall duration of the simulation process, we also save the time before starting the simulation process.

```
nIterations = 200
TimeStartSimulations = Sys.time()
```

Then, we simulate the data once for each number of participants, number of trials and iterations, fit a GLMM for each and save the p values for the influence of `ConditionOfInterest` on PSEs and JNDs in a dataframe. We choose "`nAGQ = 1`" and "`glmerControl(optimizer = "nloptwrap"))`" in the `glmer()` function because these settings make for reliable p values, while maintining a reasonable fitting duration. We will discuss the simulations this decision is based on in more detail below when comparing different optimizers. We further use the model specification we had decided upon above (M15). We also add in a timer to keep you updated about how far through the power simulations you are.

```
PowerfulDataframe = data.frame()

for (nParticipants in RangeNs){
  for (reps in RangeRepetitions){

    TimeStartTrial = Sys.time() #get time at beginning of trial

    for(i in 1:nIterations){

    print(nParticipants)
    print(reps)
    print(nIterations)

    Psychometric = SimulatePsychometricData(nParticipants,
                                            ConditionOfInterest,
                                            StandardValues,
                                            reps,
                                            PSE_Difference,
                                            JND_Difference,
                                            Multiplicator_PSE_Standard,
                                            Multiplicator_SD_Standard,
                                            Type_ResponseFunction,
                                            SD_ResponseFunction,
                                            Mean_Variability_Between,
                                            SD_Variability_Between)


    GLMM = glmer(cbind(Yes, Total - Yes) ~ ConditionOfInterest*Difference +
                               (ConditionOfInterest+Difference| ID) +
                               (Difference| StandardValues),
                 family = binomial(link = "logit"),
                 data = Psychometric,
                 nAGQ = 1,
                 glmerControl(optimizer = "nloptwrap"))

    PowerfulDataframe = rbind(PowerfulDataframe,
                c(nParticipants=nParticipants,
                  reps=reps,
                  pvalue_PSE = summary(GLMM)$coefficients[14],
                  pvalue_JND = summary(GLMM)$coefficients[16],
                  iteration = i))
    }
    print(paste0("200 iterations took ", round(Sys.time() - TimeStartTrial), " seconds. The
power for the current run through (",nParticipants," Participants, ", reps, " Repetitions)
is ",mean(PowerfulDataframe$pvalue_PSE[PowerfulDataframe$nParticipants == nParticipants &
PowerfulDataframe$reps == reps] < 0.05)))
  }
}

colnames(PowerfulDataframe)                                                              =
c("nParticipants","reps","pvalue_PSE","pvalue_JND","iteration")
```

Finally, we can print the power for each combination of number of repetitions and number of participants by setting an alpha level (typically 0.05) and counting in how many iterations out of `nIterations` we found a p value below alpha.

```
alpha = 0.05

PowerfulDataframe = PowerfulDataframe %>% group_by(nParticipants,reps) %>%
  mutate(Power_PSE = mean(pvalue_PSE < alpha),
         Power_JND = mean(pvalue_JND < alpha))

PowerfulDataframe %>% group_by(nParticipants,reps) %>%
  slice(1)
```

We can then print the duration of the process.

```
Sys.time() - TimeStartSimulations
```

We can also plot the computed power with the following lines.

```
ggplot(PowerfulDataframe, aes(nParticipants,Power, color = as.factor(reps))) +
  geom_line(size = 1) +
  xlab("Number of Participants") +
  ylab("Power") +
  scale_x_continuous(breaks = c(10,12,14,16,18,20)) +
  scale_color_manual(name = "Repetitions\nper Staircase",
                     values = c(Red,BlauUB,Yellow)) +
  geom_hline(yintercept = 0.9, linetype=2) +
  geom_hline(yintercept = 0.95, linetype=3) +
  ylim(c(0,1)) +
  facet_wrap(WhichValue~.) +
  theme(legend.position = c(0.4,0.2))
```

Figure 8 shows this plot for the values chosen above. We can see, for example, that we can achieve a power of 0.8 for the influence of `ConditionOfInterest` on JNDs both by increasing the number of participants (see red line) or the number of trials per staircase, while keeping the overall number of participants low (see yellow and blue line).

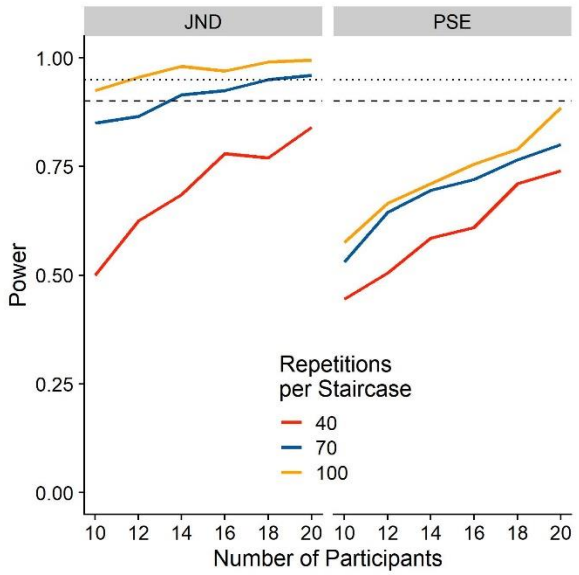


Figure 8: Power for the influence of ConditionOfInterest on JND (left panel) and PSE (right panel), for different participant numbers. The different colors indicate different numbers of repetitions for each staircase. The solid horizontal line indicates a power level of 0.8, the dashed horizontal line indicates a power level of 0.9 and the dotted horizontal line indicates a power level of 0.95.

Power Analyses in Julia

The relatively new programming language Julia markets itself as an up-and-coming, faster alternative to R. Developed with a focus on speed, it can achieve a performance similar to C, while R is routinely among the slowest languages in benchmarks. A package for the fitting of (Generalized) Mixed Models is already available for Julia, and it has been suggested that this package is up to ten times faster than our R package of choice, lme4. It may thus provide a means to speed up the simulation process, which, as mentioned above, can take one to several hours in R. We did some simulations in R and Julia to test out this speed advantage, but found that Julia does not actually provide a speed (or any other) advantage over an alternative implementation of one of the optimizers used by the R package lme, at least for our purposes. Please find the Julia code and detailed plots about its performance in the supplementary materials. The code is also available as Jupyter notebook here (GitHub).

A comparison of several model fitting parameters

We used the above procedures to measure the speed for the same operations in Julia and R. We used a Julia script (available here on GitHub) and use the RCall implementation to obtain lme4 fits. We used the Julia script because we also included some Julia implementations in our comparisons. But since we found that none of the Julia implementations were better than the R implementations by any metric, we report these comparisons only in the supplementary materials. We also tried using lme4 natively in R and found by-and-large the same fitting durations as when calling lme4 through RCall from the Julia script. Using RCall allows us to compare fitting durations across the same datasets, which helps to eliminate variability due to differences in the simulated datasets. We perform the procedure 200 times for each combination of Optimizer Configuration (see below), number of participants, number of trials and Effect condition (No Effect, `PSE_Difference = JND_Difference = 0`; and "`Small Effect`", `PSE_Difference = -0.025` and `JND_Difference = 0.05`). There are different implementations and configurations available, which we are evaluating based on fitting duration and model fit. Since p values for (Generalized) Linear Mixed Models are generally approximations, we are also going to evaluate the false positive rates to see how adequate these approximations are.

Optimizers

lme4 for R supports implementations of the Nelder-Mead method (Nelder & Mead, 1965) and the BOBYQA method (Powell, 2009). It furthermore supports the (generally faster) implementations of optimizing algorithms from the package nloptwrap; we thus also add the BOBYQA instantiation from this package for comparison. Julia offers the Nelder-Mead and the BOBYQA algorithms.

P value approximation

There are different approaches to significance testing in Mixed Modelling. Common approaches are Wald Z Tests, Likelihood Ratio Testing and bootstrapped confidence intervals. Bootstrapped confidence intervals are too computationally costly for the purpose of power simulations. We will therefore test the Wald Z Test (implemented with the Satterthwaite degrees of freedom Method in the R package lmerTest) and Likelihood Ratio Tests (implemented in the anova() function in R).

nAGQ = 0/1

Furthermore, the lme4 implementation in R offers the possibility to trade-off accuracy for higher speed (nAGQ = 0 argument in R, with nAGQ = 1 being the slower, more accurate default). We test whether the gains in model fit for the slower, more accurate version are worth the increased fitting duration.

Which configurations perform the best?

In the following, we compare which combination of the above parameters (optimizers, p value approximation, accuracy-speed trade-off) before the best along several axes (fitting speed, model fits, false positive rates). Please note that we chose model specification M15, as per the simulations above, to further analyze the impact of these parameters.

Speed

Overall, the fastest implementations are by far the BOBYQA implementation in R from the "nloptwrap" package (see Figure 9) in its nAGQ=0 configuration, followed by the nAGQ = 0 BOBYQA optimizer and the likelihood ratio test over two models fitted with the nAGQ = 0 version of the nloptwrap optimizer. All other implementations (the nAGQ=1 version of the BOBYQA and nloptwrap optimizers and both versions of the Nelder-Mead implementation) are much slower, taking between three and ten times longer than the fastest implementation.

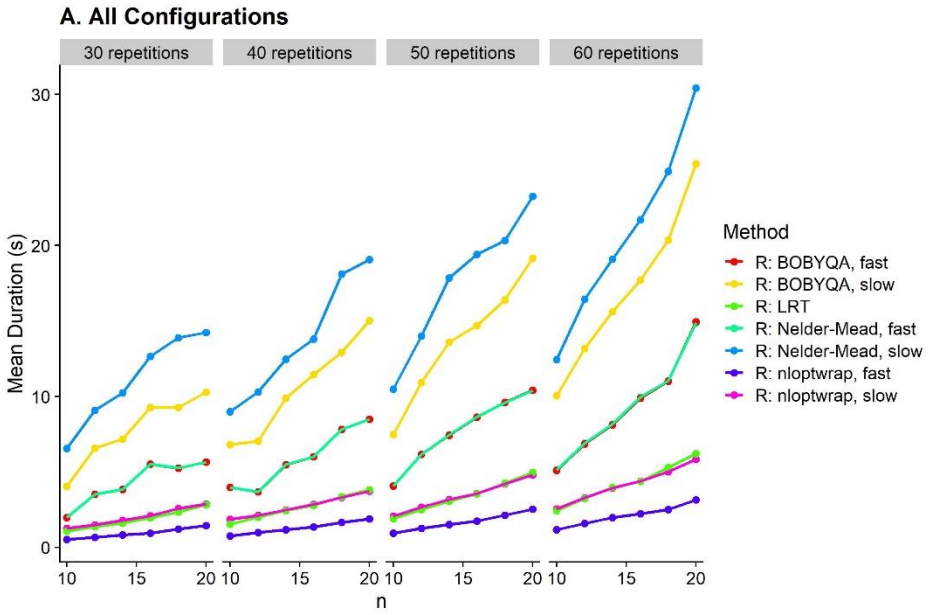


***Figure 9**: Mean durations to fit GLMMs in different languages and with different configurations (color-coded), for datasets of different sizes. We illustrate the durations for 10, 12, 14, 16, 18 and 20 participants (x axis), 30, 40, 50 and 60 trials per staircase (columns of panels) and no effect (PSE_Difference = JND_Difference = 0) and a small effect (PSE_Difference = 0.025 and JND_Difference = 0.05) in rows of panels. We plot the median values across 20 repetitions per combination of repetition number, participant number, effect (none, small) and GLMM fitting configuration. **A**. All optimizer configurations. **B**. Only the fastest four configurations, from fastest to slowest: Julia: BOYQA, fast; R: nloptwrap, fast; R: nloptwrap, slow; Julia: BOBYQA, slow.*

Model fits

To assess whether the slower algorithms might yield a better model fit, we subtract the AICs for the other optimizer configurations by the AIC for the slowest optimizer configuration ("R: Nelder-Mead, nAGQ=1"). This method yields a ratio where a value of below 1 indicates that the optimizer configuration in question makes for better fits than "R: Nelder-Mead, slow", and values above 1 indicate that the optimizer makes for worse fits. As evident from Figure 10, the AIC is indeed lowest for this optimizer. However, the differences with regards to the other optimizer configurations, including the fastest ones, are miniscule, with ratio differences below 0.0004.

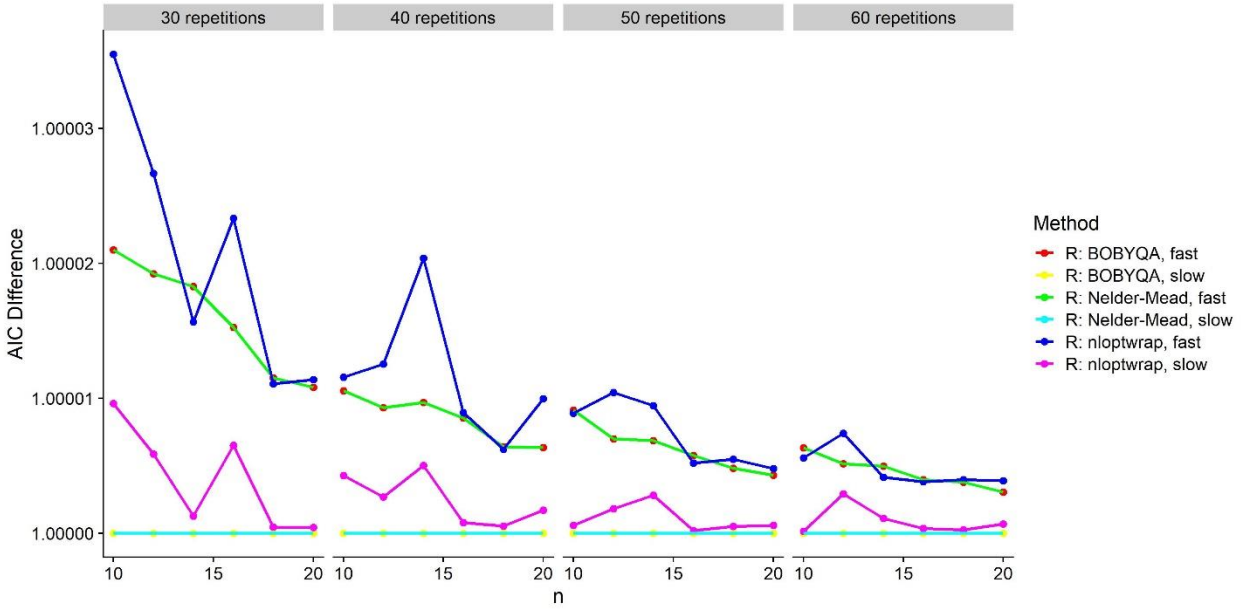


Figure 10: Median difference between each combination of configurations (color-coded) and the "R: Nelder-Mead, slow" combination for 10-20 participants (x axis) and 30-60 repetitions per staircase (panels).

False positive rates

We illustrate the distribution of p values in Figure 11A for accuracy and in Figure 11B for precision. For accuracy, that is, PSE differences, there are very small differences between the different fits and ways of obtaining p values. For precision, that is, for JND differences, however, several optimizer configurations and p value approximations yield inflated false positive rates. All `nAQP = 0` implementations give false positive rates of 10 to 20 %, in comparison to the expected 5 %. Only the three R implementations with `nAQP = 1` yield acceptable false positive rates. Notably, these inflated false positive rates seem to translate also to a higher rate of "true positives" in the presence of a very small effect. However, a comparison with those optimizer configurations that yield an acceptable false positive rate reveals a higher "true positive rate", indicating that using these implementations might overestimate power. With the exception of "R: nloptwrap, slow", the less false-positive-prone optimizer configurations are all extremely slow (see Figure 9) – so slow, in fact, that they are hardly suitable for power simulations. When using Likelihood Ratio Tests instead of the default Z Wald tests, the elevated false positive rates disappear. However, Likelihood Ratio Tests require fitting two models: a test model that contains the variable of interest (in our case the interaction between "`ConditionOfInterest`" and "`Difference`"), and the next simplest model that doesn't contain it (`ConditionOfInterest` and `Difference` as main effects, but not their interaction) as null model. That is, two models need to be fitted to obtain one p value instead of just one, which makes for nearly twice the fitting duration. We used the faster method of fitting GLMMs ("R: BOBYQA (nloptwrap), fast") and compared them with the Likelihood Ratio Test implemented in the `stats::anova()` function in R. This yields the expected false positive rates.

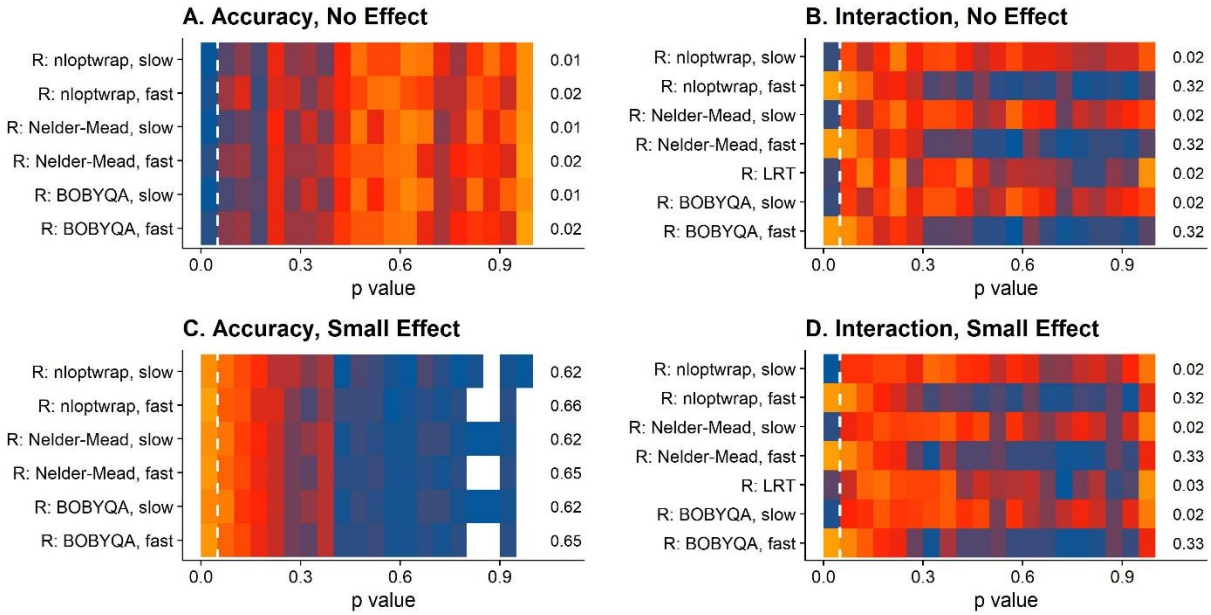


Figure 11: Frequency of p values per bin of 0.05 for different fitting methods. The gradient from blue over red to orange indicates the density. A. P values for PSE differences when there is no effect (PSE_Difference = 0). B. P values for JND differences when there is no effect (JND_Difference = 0). C. P values for PSE differences when there is a small effect (PSE_difference = 0.025). C. P values for JND differences when there is a small effect (JND_Difference = 0.05).

Concluding recommendations for power analyses

Overall, we find that fitting GLMMs in R with the nloptwrap implementation of the BOBYQA algorithm with nAQP = 1 is the way to go: it provides acceptable speed and good model fits, while not exceeding the expected false positive rate of 5%.

While we give very specific recommendations here, it is important to note that some of these recommendations may be specific to the parameters we chose to simulate our data (e.g., between- and within-participant variability in different dimensions). While we cover some of the potentially most impactful variations, such as presence or absence of JND and PSE differences or the direction of such differences, it is impossible to address all potentialities. To circumvent this issue, we recommend to conduct power analyses for two or three relevant model specifications (e.g., M15, M24 and M25) and aim for a sample size that allows to achieve the power level for the least powerful of these variations. When analyzing the data, we then recommend to compare model fits among these plausible options (with a Likelihood Ratio Test implemented in the anova() function in R) and ascertain which one is the most appropriate for your specific data set.

Conclusions and Recommendations

In this manuscript, we treat several related issues regarding simulation and analysis of psychophysical data, as well as sample size planning, with the R package lme4.

Simulating psychophysical data

- We provided practical recommendations for data simulation in R (implementation available here, on GitHub)

Analyzing psychophysical data

- When using a Two-Step approach in which first PSEs and JNDs are obtained, and then these PSEs and JNDs are used as dependent variables, we show that using Linear Mixed Models over regular ANOVAs or even repeated measures ANOVAs can raise statistical validity and power significantly.
- We analyzed a Generalized Linear Mixed Modelling-based approach in-depth and provided concrete recommendations on model specifications:
    - Overall, models that are more complete in terms of random effects are generally preferrable.
    - Smaller models (with fewer random effects) bear the risk of inflating the false positive rate, "detecting" effects when there are none.
    - Fitting the models with the faster option nAGQ = 1 can also inflate the false positive rate and should be avoided, at least when p values are approximated with the lmerTest package. Bootstrapping confidence intervals instead may remedy this issue.
- In terms of power, GLMM-based approaches have an advantage over Two-Step approaches to detect differences in PSE, while both methods are roughly tied when detecting JND difference, with a potential slight edge for LMM-based Two-Step approaches.
- Since GLMMs can be vulnerable to the specifics of a dataset (e.g., different types of variability), we recommend to assess model fits for several plausible GLMMs (e.g., via Likelihood Ratio Tests) once data collection is finalized.

Power analyses

- We provide a GLMM-based sample implementation here (on GitHub).
- When comparing different technical parameters to be considered when running these power analyses, we find that an R implementation (using the glmer() function from the package lme4) using the BOBYQA implementation from the nloptwrap package with the nAGQ = 1 option is ideally suited.
- Using a Julia implementation currently provides no advantages over the above R implementation.

Acknowledgements

BJ was supported by the Canadian Space Agency (CSA). No conflicts of interest are declared.

Bibliography

A.Brown, V. (2020). An approachable introduction to linear mixed effects modeling with implementation in R. *PsyArXiv*, *53*(9), 1689–1699. https://doi.org/10.1017/CBO9781107415324.004

Aarts, B., Anderson, C. J., Anderson, J. E., Kappes, H. B., Joanna, E., & Barry, H. (2015). Estimating the reproducibility of psychological science. *Science*, *349*(6251), aac4716. https://doi.org/10.1126/science.aac4716

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. https://doi.org/10.1016/j.jml.2012.11.001

Bartlett, M. S. (1937). Properties of sufficiency and statistical tests. *Proceedings of the Royal Society of London. Series A - Mathematical and Physical Sciences*, *160*(901), 268–282. https://doi.org/10.1098/rspa.1937.0109

Bates, D., Mächler, M., Bolker, B. M., & Walker, S. C. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, *67*(1). https://doi.org/10.18637/jss.v067.i01

Bolker, B. M., Brooks, M. E., Clark, C. J., Geange, S. W., Poulsen, J. R., Stevens, M. H. H., & White, J. S. S. (2009). Generalized linear mixed models: a practical guide for ecology and evolution. *Trends in Ecology and Evolution*, *24*(3), 127–135. https://doi.org/10.1016/j.tree.2008.10.008

Brown, M. B., & Forsythe, A. B. (1974). Robust tests for the equality of variances. *Journal of the American Statistical Association*, *69*(346), 364–367. https://doi.org/10.1080/01621459.1974.10482955

Bürkner, P. C. (2018). Advanced Bayesian multilevel modeling with the R package brms. *R Journal*, *10*(1), 395–411. https://doi.org/10.32614/rj-2018-017

Chambers, C. D., Dienes, Z., McIntosh, R. D., Rotshtein, P., & Willmes, K. (2015). Registered Reports: Realigning incentives in scientific publishing. *Cortex*, *66*, 1–2. https://doi.org/10.1016/j.cortex.2015.03.022

Debruine, L. M., & Barr, D. J. (2019). Understanding mixed effects models through data simulation. *PsyArXiv*.

Devezer, B., Nardin, L. G., Baumgaertner, B., & Buzbas, E. O. (2019). Scientific discovery in a model-centric framework: Reproducibility, innovation, and epistemic diversity. *PLoS ONE*, *14*(5), 1–23. https://doi.org/10.1371/journal.pone.0216125

Dunn, P. K., & Smyth, G. K. (1996). Randomized Quantile Residuals. *Journal of Computational and Graphical Statistics*, *5*(3), 236–244. https://doi.org/10.1080/10618600.1996.10474708

Gaylor, D. W., & Hopper, F. N. (1969). Estimating the Degrees of Freedom for Linear Combinations of Mean Squares by Satterthwaite's Formula. *Technometrics*, *11*(4), 691–706. https://doi.org/10.1080/00401706.1969.10490732

Gelman, A., & Hill, J. (2008). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Retrieved from https://books.google.com/books?hl=es&lr=&id=c9xLKzZWoZ4C&oi=fnd&pg=PR17&dq=Gelman,+A.+%26+Hill,+J.+Data+analysis+using+regression+and+multilevel/hierarchical+models+Cambridge+University+Press,+2006&ots=bbV8PWRtqd&sig=HL626z0jgrfA8fum8xIPBYlcVWA

Guest, O., & Martin, A. E. (2020). *How computational modeling can force theory building in psychological science*. 1–13.

Hartig, F. (2020). DHARMa: Residual Diagnostics for Hierarchical Regression Models. Retrieved July 23, 2020, from The Comprehensive R Archive Network website: https://cran.r-project.org/web/packages/DHARMa/vignettes/DHARMa.html#calculating-scaled-residuals

Hodges, J. S. (2016). Richly parameterized linear models: Additive, time series, and spatial models using random effects. In *Richly Parameterized Linear Models: Additive, Time Series, and Spatial Models Using Random Effects*.

Hunter, J. E. (2001). The Desperate Need for Replications. *Journal of Consumer Research*, *28*(1), 149–158. https://doi.org/10.1086/321953

Knoblauch, K., & Maloney, L. T. (2012). *Modelling psychophysical data in R*. New York: Springer.

Kumle, L., Võ, M. L., & Draschkow, D. (2020). Estimating power in (generalized) linear mixed models: an open introduction and tutorial in R . *PsyArXiv*, 1–29. Retrieved from https://psyarxiv.com/vxfbh/

Kuznetsova, A., Brockhoff, P. B., & Christensen, R. H. B. (2017). lmerTest Package: Tests in Linear Mixed Effects Models . *Journal of Statistical Software*, *82*(13). https://doi.org/10.18637/jss.v082.i13

Levene, H. (1960). Robust tests for equality of variances. *Contributions to Probability and Statistics: Essays in …*, *69*(346), 278–292. Retrieved from https://ci.nii.ac.jp/naid/10007628681

Linares, D., & López-Moliner, J. (2016). quickpsy: An R Package to Fit Psychometric Functions for Multiple Groups. *The R Journal*, *8*(1), 122–131. Retrieved from https://journal.r-project.org/archive/2016-1/linares-na.pdf

Luke, S. G. (2017). Evaluating significance in linear mixed-effects models in R. *Behavior Research Methods*, *49*(4), 1494–1502. https://doi.org/10.3758/s13428-016-0809-y

Massey, F. J. (1951). The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*, *46*(253), 68–78.

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. https://doi.org/10.1016/j.jml.2017.01.001

Moscatelli, A., Mezzetti, M., & Lacquaniti, F. (2012). Modeling psychophysical data at the population-level: The generalized linear mixed model. *Journal of Vision*, *12*(11), 1–17. https://doi.org/10.1167/12.11.26

Nelder, J. A., & Mead, R. (1965). A Simplex Method for Function Minimization. *The Computer Journal*, *7*(4), 308–313. https://doi.org/10.1093/comjnl/7.4.308

Nosek, B. A., Ebersole, C. R., DeHaven, A. C., & Mellor, D. T. (2018). The preregistration revolution. *Proceedings of the National Academy of Sciences of the United States of America*, *115*(11), 2600–2606. https://doi.org/10.1073/pnas.1708274114

Nosek, B. A., & Lakens, D. (2014). Registered reports: A method to increase the credibility of published results. *Social Psychology*, *45*(3), 137–141. https://doi.org/10.1027/1864-9335/a000192

Oberauer, K., & Lewandowsky, S. (2019). Addressing the theory crisis in psychology. *Psychonomic Bulletin and Review*, *26*(5), 1596–1618. https://doi.org/10.3758/s13423-019-01645-2

Powell, M. (2009). The BOBYQA algorithm for bound constrained optimization without derivatives. *NA Report NA2009/06*, 39. https://doi.org/10.1.1.443.7693

Prins, N., & Kingdom, F. (2016). *Psychophysics: A Practical Introduction*. https://doi.org/10.1017/CBO9781107415324.004

Spiegelhalter, D., & Rice, K. (2009). Bayesian statistics. *Scholarpedia*, *4*(8), 5230. https://doi.org/10.4249/scholarpedia.5230

Stan Development Team. (2016). *Stan: the R interface to Stan. R package version 2.14.1*. 1–23. Retrieved

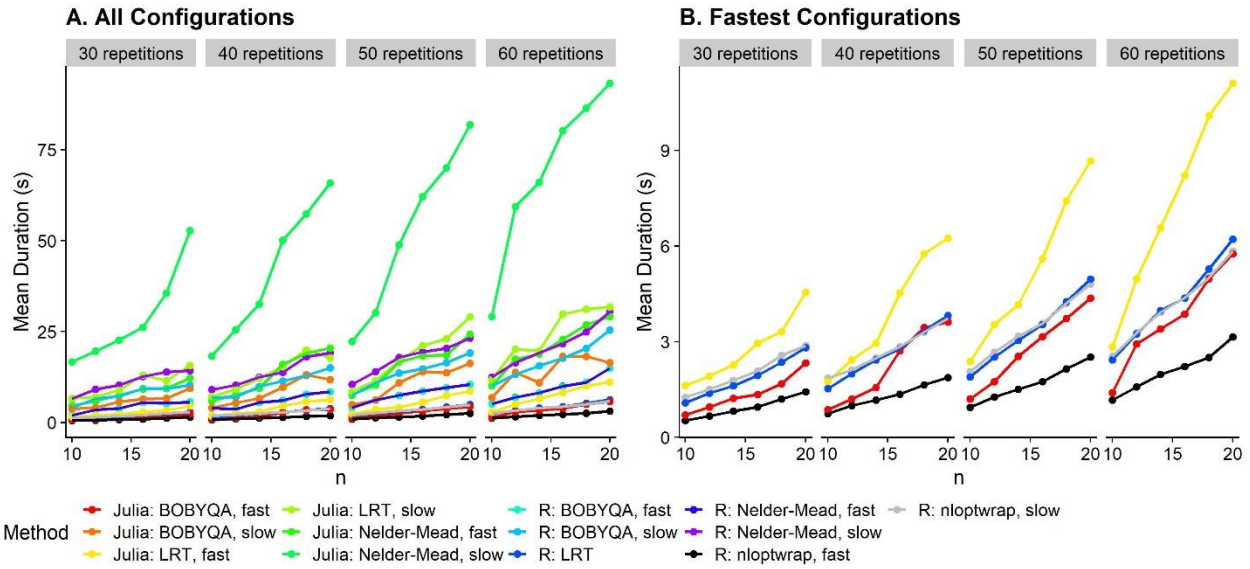
from http://mc-stan.org

Taylor, M. M., & Creelman, C. D. (1967). PEST: Efficient Estimates on Probability Functions. *The Journal of the Acoustical Society of America*, *41*(4A), 782–787. https://doi.org/10.1121/1.1910407

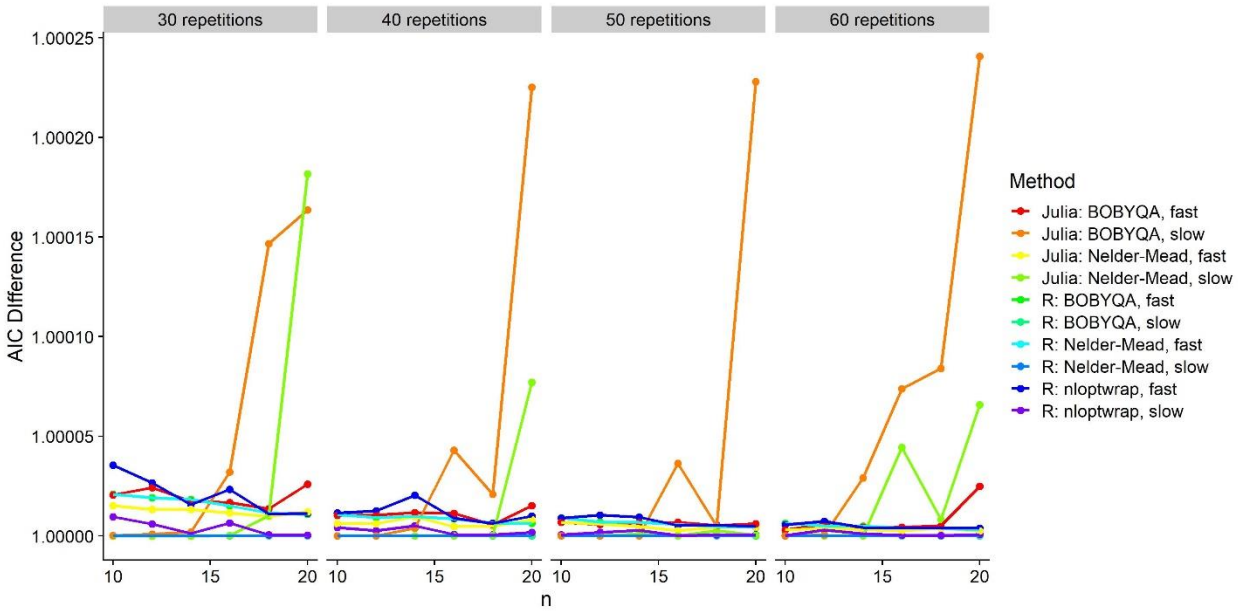
Supplementary Materials

Power analyses in Julia – not a faster alternative

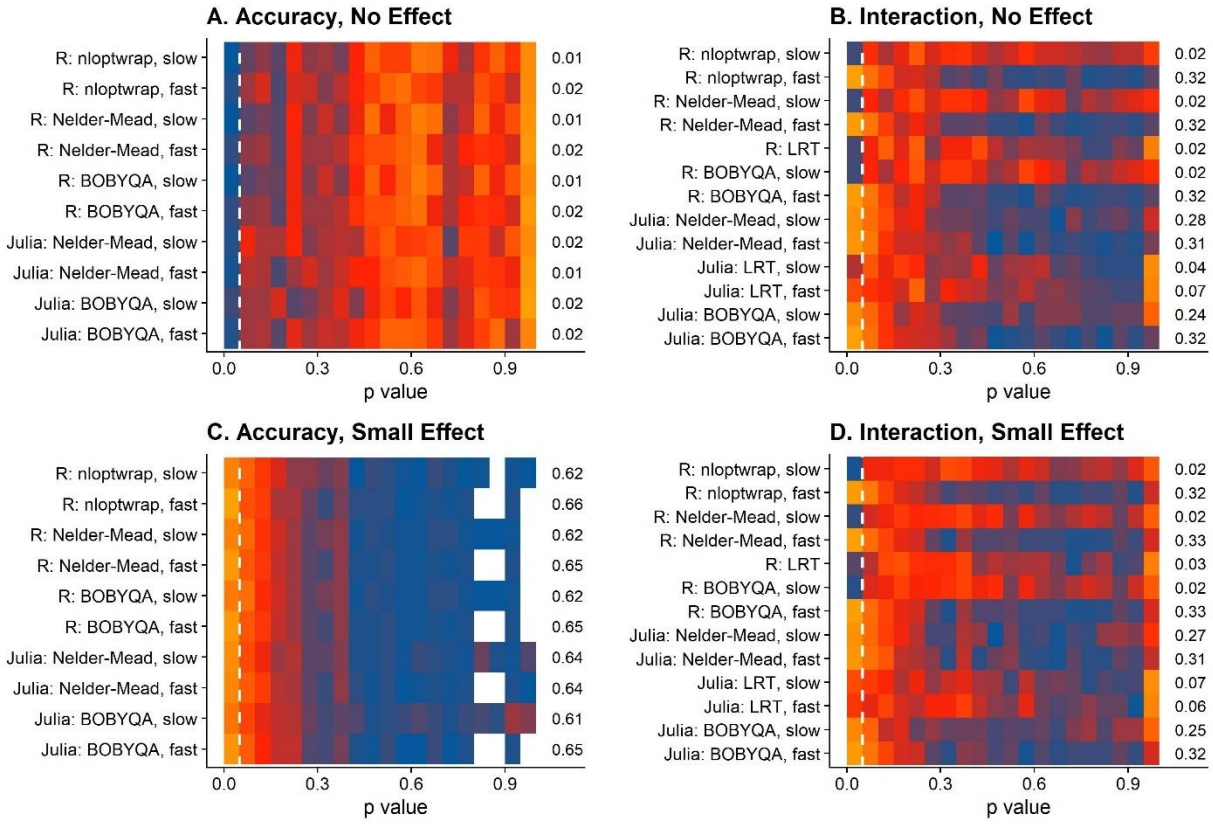
Julia is a programming language that is advertised as faster alternative to R for statistical programming. It is relatively intuitive and user-friendly for users with R experience and offers the ability to call R code from within Julia. This enables us to generate the datasets in R, export them and conduct the time-consuming fitting of the GLMMs in Julia. For the present paper, we expect readers to have already installed Julia. The script we used to compare different R and Julia implementations can be found here (GitHub). The script that generates Supplementary Plots 1, 2, and 3 can be found here (GitHub).



*Supplementary Figure 1: A. As **Figure 9**, but includes all Julia implementations. B. Durations of the fastest five implementations (R: nloptwrap, fast, R: nloptwrap, slow, R: LRT, Julia: LRT, and Julia: BOBYQA, fast).*



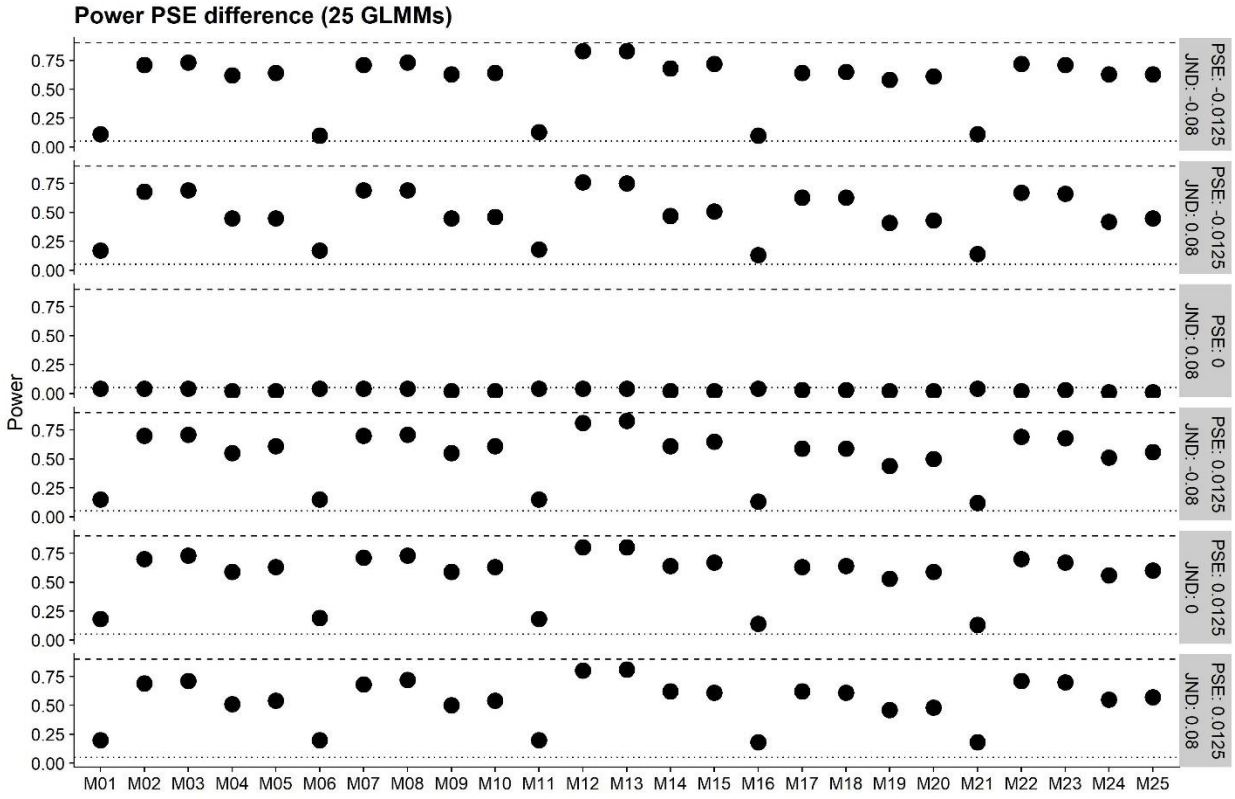
Supplementary Figure 2. As Figure 10, but including Julia implementations.



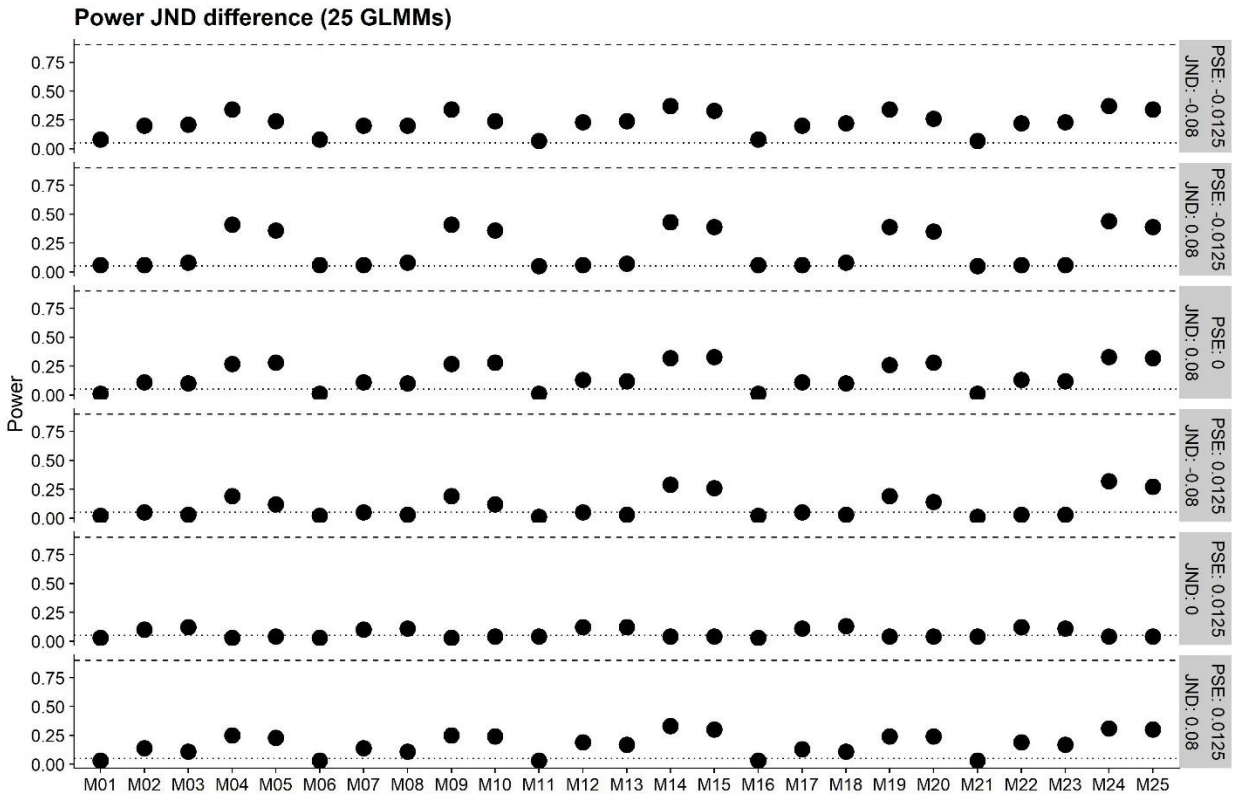
Supplementary Figure 3: As Figure 11, but including Julia implementations.

When comparing performance of the Julia optimizers and optimizer configurations (see Supplementary Figure 1 for speed, Supplementary Figure 2 for model fits and Supplementary Figure 3 for false positive rates) to the R implementations, we can see that the fast Julia option with an acceptable false positive

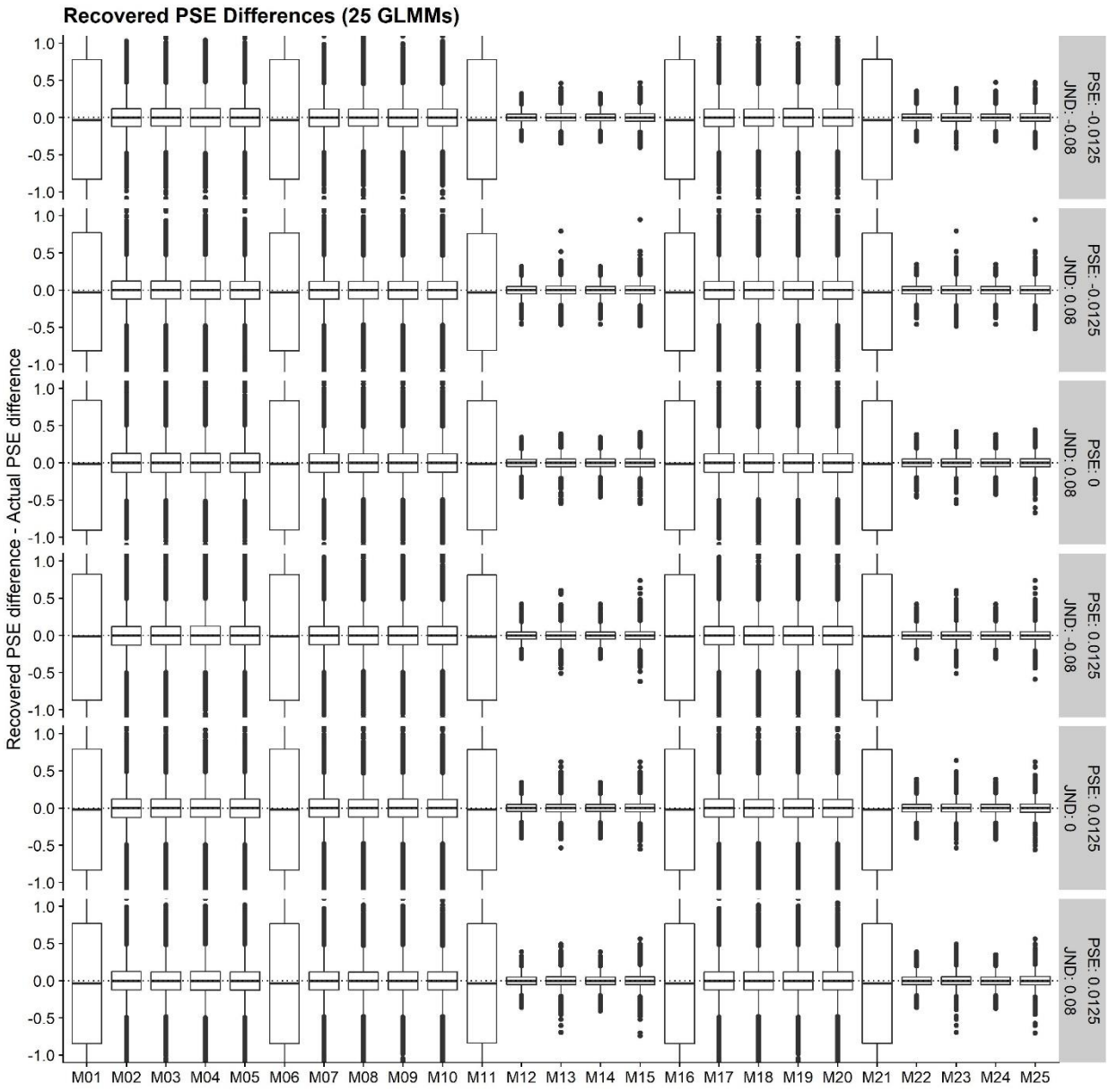
rate for the detection of JND differences (Likelilhood Ratio Tests performed over models fitted with the BOBYQA optimizer with the “fast” option) takes at least twice as long as the best R option (p value approximation with Z Wald tests performed in models fitted with the nloptwrap implementation of the BOBYQA optimizer and the nAGQ=1 option). Thus, we cannot recommend picking up Julia just for the sake of these power analyses.



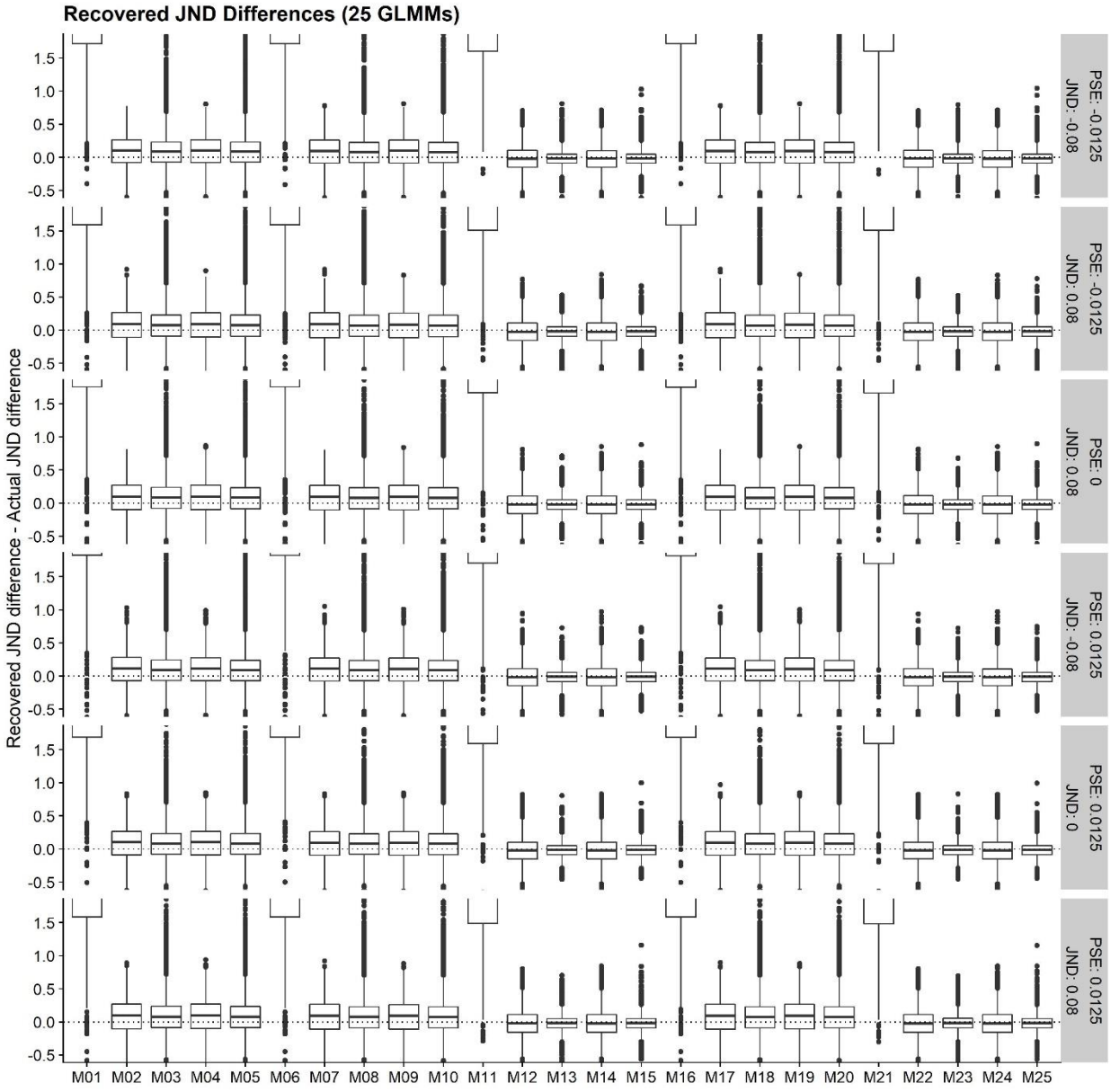
Supplementary Figure 4: Power for detecting PSE differences between Condition of Interest and Baseline for each Generalized Linear Mixed Model (see Table 2; x axis) and different combinations of PSE and JND differences (rows of panels), for 10 participants and staircases with 70 repetitions.



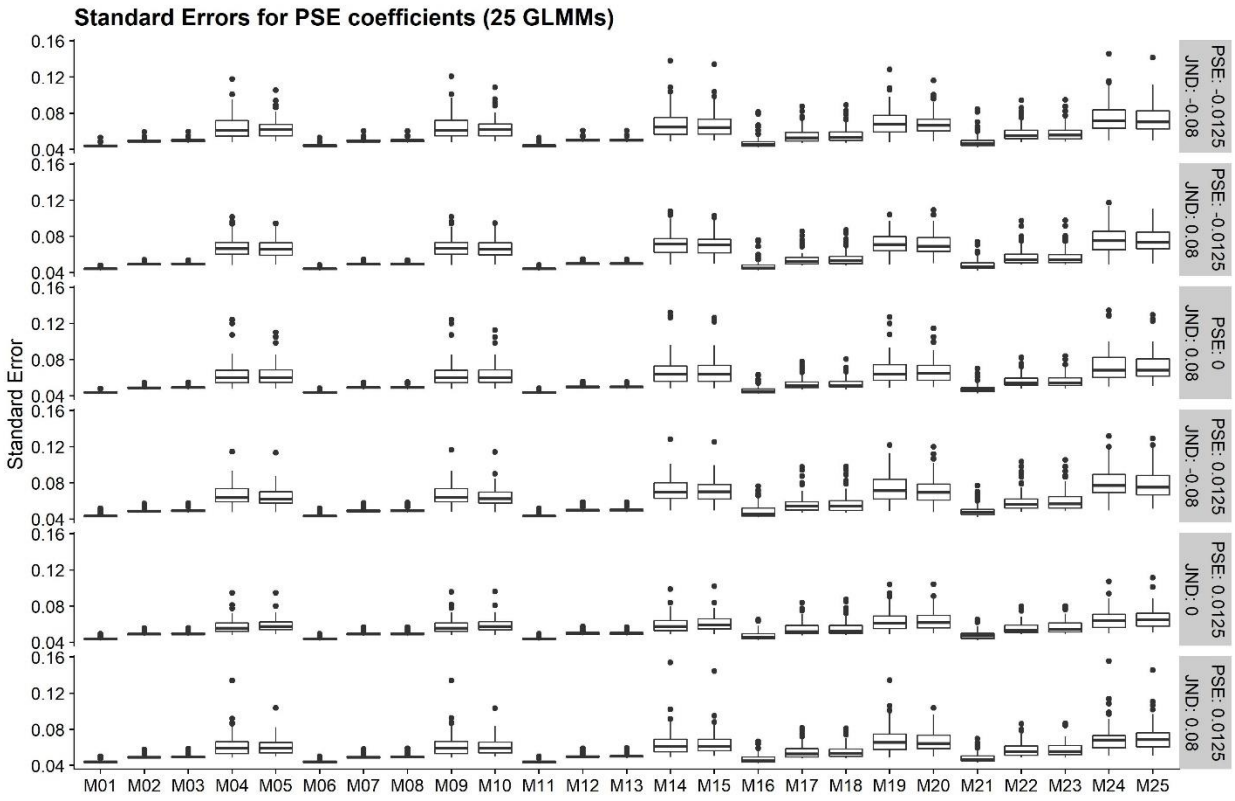
Supplementary Figure 5: As Supplementary Figure 4, but for JND differences between Condition of Interest and Baseline.



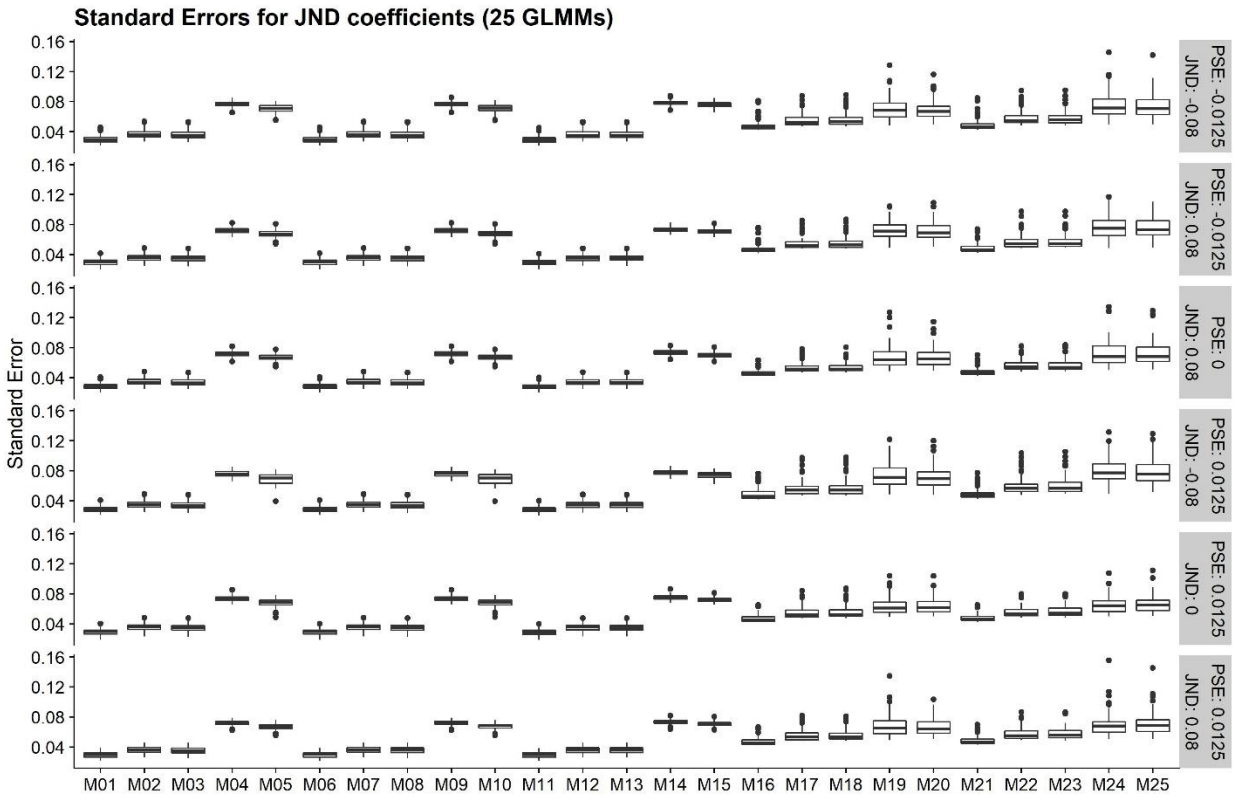
Supplementary Figure 6: PSE difference between Condition of Interest and Baseline recovered by each Generalized Linear Mixed Model (see Table 2; x axis) for each combination of PSE difference and JND difference (rows of panels).



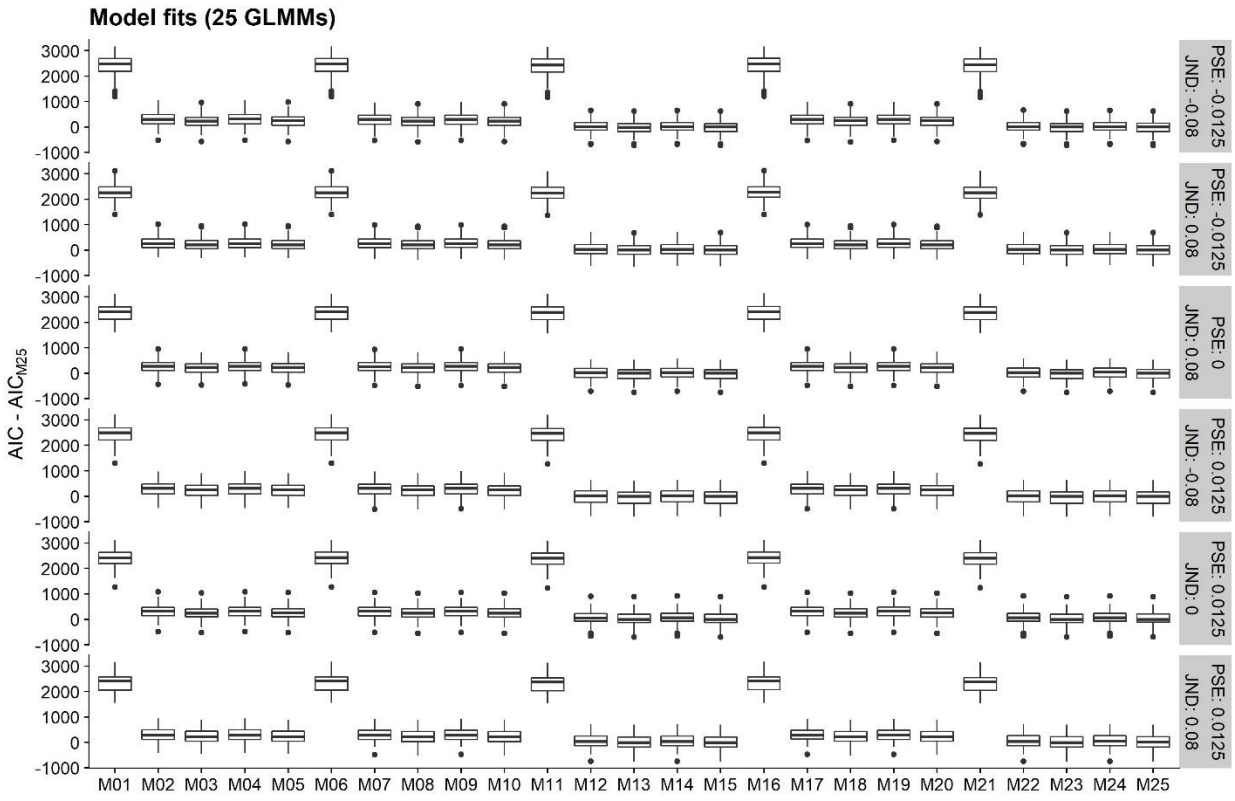
Supplementary Figure 7: As Supplementary Figure 6, but for the JND difference between Condition of Interest and Baseline.



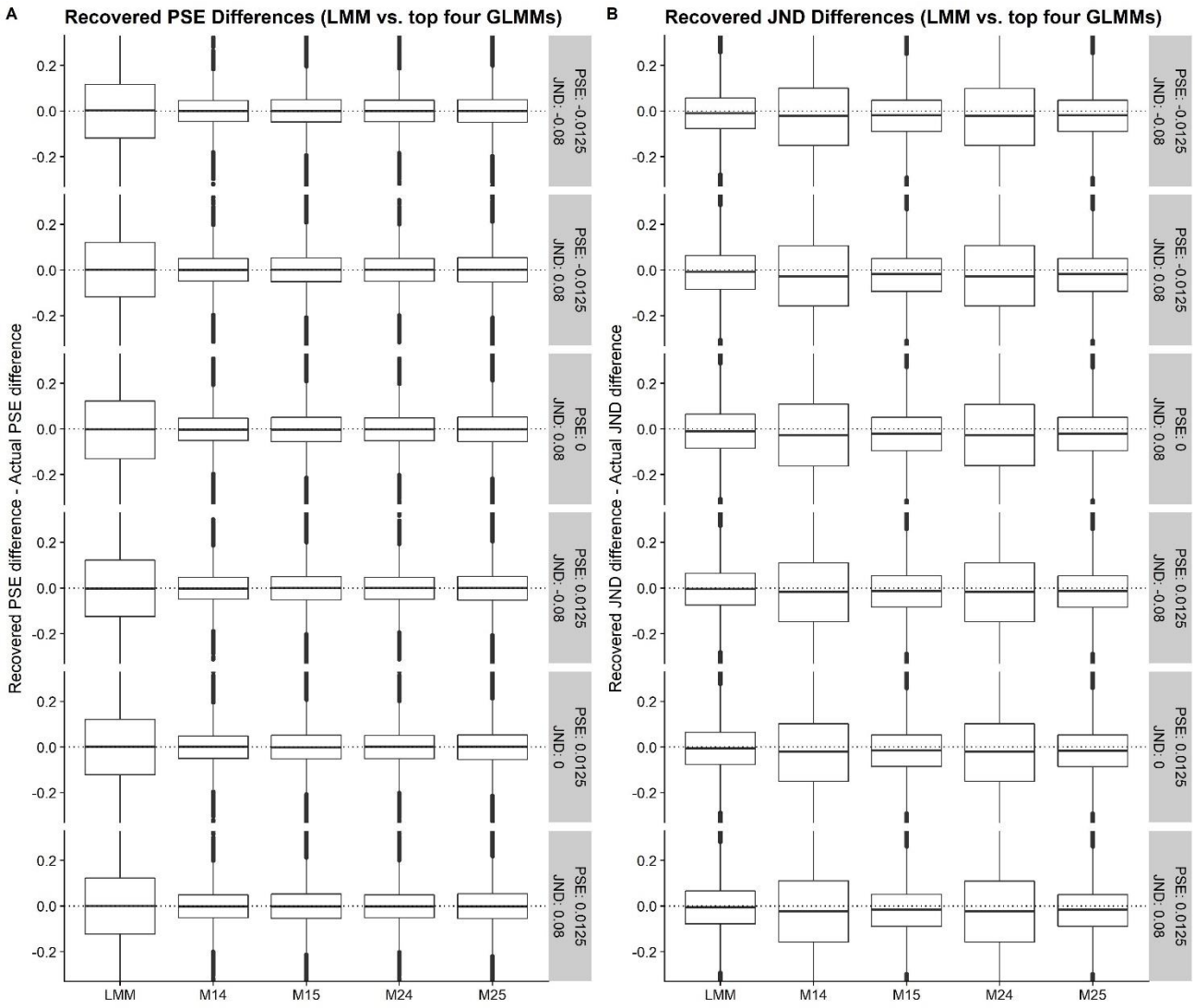
Supplementary Figure 8: Standard Errors for the regression coefficient related to PSE differences between Condition of Interest and Baseline for each Model (see Table 2; x axis) and different combinations of PSE and JND differences (rows of panels), for 10 participants and staircases with 70 repetitions.



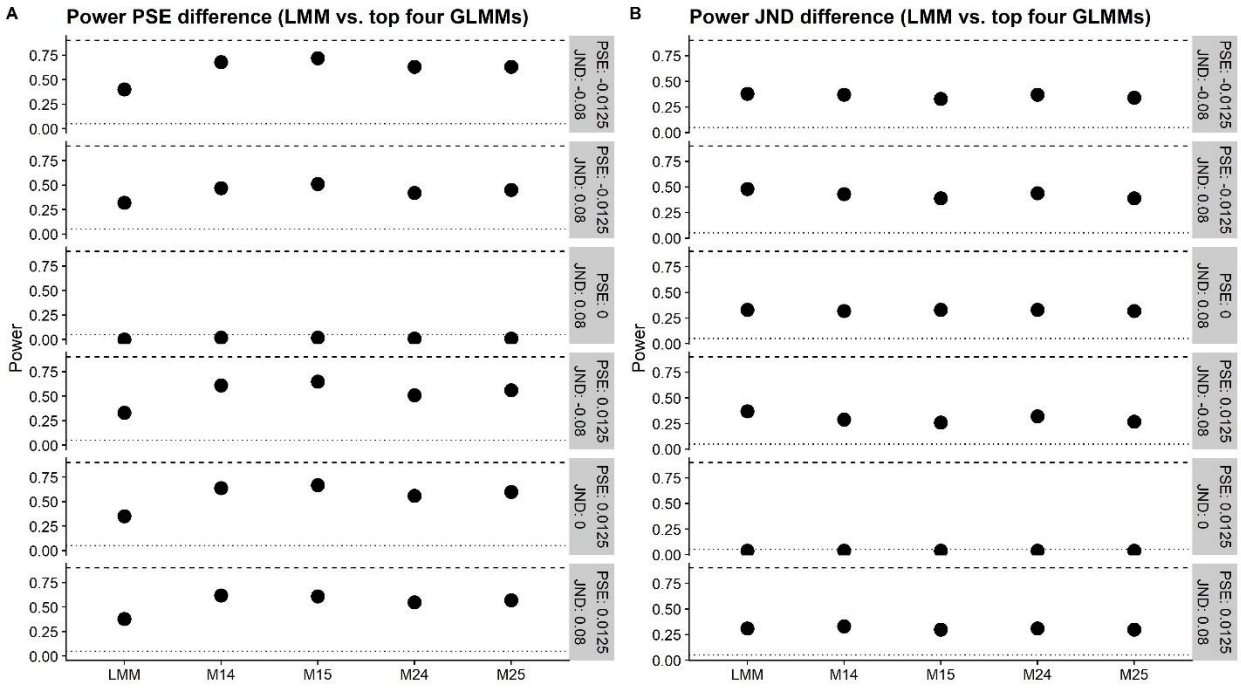
Supplementary Figure 9: As Supplementary Figure 8, but for JND differences between Condition of Interest and Baseline.



Supplementary Figure 10: AICs for each of the Generalized Linear Mixed Models (see Table 2, on the x axis), normalized by subtracting from them the AIC of the most comprehensive model (M25), separately for each combination of PSE and JND difference (rows of panels).



Supplementary Figure 11: A. As Supplementary Figure 6, but for the four best Generalized Linear Mixed Models (M14, M15, M24, M25) and a Linear Mixed Model solution (LMM). B. As Supplementary Figure 7, but for the four best Generalized Linear Mixed Models (M14, M15, M24, M25) and a Linear Mixed Model solution (LMM).



Supplementary Figure 12: A. As Supplementary Figure 4, but for the four best Generalized Linear Mixed Models (M14, M15, M24, M25) and a Linear Mixed Model solution (LMM). B. As Supplementary Figure 5, but for the four best Generalized Linear Mixed Models (M14, M15, M24, M25) and a Linear Mixed Model solution (LMM).