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

# 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 participants and/or blocks or within 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. While this can be a valid approach, this generally neglects that each PSE and JND is based on a large number of trials and thus fails to account for the added reliability of the measures provided. Depending on the experimental design, this can lead to a loss of statistical power. As a solution, 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. In the first part of this manuscript, we will compare different statistical approaches to Null Hypothesis Significance Testing for typical psychophysical data. We furthermore provide a step-by-step tutorial to simulating psychophysical data that R users can easily adapt to match their own experiments. We will use these simulated data to illustrate the different approaches in significance testing for psychophysical data and give concrete recommendations for a variety of use cases.



Figure 1: Sample psychometric function 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.

In the second part of this paper, we will show how we can leverage simulations to plan sample sizes more efficiently. While power analyses are becoming more mainstream in other areas, they still are the exception in the typical psychophysical study. Some tutorials 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. However, very common psychophysical designs such as two-alternative forced-choice tasks, which rely heavily on fitting 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. The present paper thus aims to give advice on how to conduct power analyses for common psychophysical designs that investigate the effect of a categorical experimental variable on precision and accuracy in two-alternative forced-choice paradigms. In the following, we thus first discuss the two main methods for analyzing data from Two-Alternative Forced-Choice Tasks. We then elaborate on how to simulate datasets that resemble the data expected for a task, and provide a sample analysis for this fictional dataset. Then, we give recommendations on how to obtain the power of an experimental setup given certain assumptions, and compare the two approaches of analysis in terms of their power. We accompany this with example implementations in R and Julia and show ways to determine the most efficient trials-per-participant ratio.

# Simulating Psychophysical Data

We will first discuss the specifics of simulating psychophysical data for one typical case of psychometric 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 bold and between quotation marks refer to variables in the script. For some of the variables, we demonstrate how to derive them from existing datasets. To this end, we will use pilot data from our lab.

## 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. It can take the values "normal", "Cauchy" and "uniform". "Normal" and "Cauchy" are recommended when you are using a staircase procedure, while "uniform" corresponds to methods of constant stimuli. For a comparison between the three options, see further below. Figure 2visualizes 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. For the Method of Constant Stimuli, no randomness is involved in how the presented stimulus strengths are chosen. In this case, we use the values chosen for your stimulus.



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. “**Mean\_Variability\_Between**” sets the standard deviation of the normal distribution these PSEs are drawn from. This normal distribution has a mean of 1, that is, the standard deviation needs to be set accordingly.

We assume that there is 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.

In the following, we provide the 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 define and use the “Where\_Am\_I” function to 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)

Where\_Am\_I <- function(path=T){

if (path == T){

dirname(rstudioapi::getSourceEditorContext()$path)

}

else {

rstudioapi::getSourceEditorContext()$path

}

}

setwd(Where\_Am\_I())

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")

We then fit the psychometric functions using the “quickpsy” package. 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 “congruent” 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 Cummulative 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 modelfit. 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\_Normal, 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](https://github.com/b-jorges/Power-Analyses-Psychophysics/tree/master/Get%20parameters%20from%20existing%20data).

## 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.

ID = paste0("s",1:15)

ConditionOfInterest = c(0,1)

StandardValues = c(5,8)

reps = 1:100

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 = 0.1

SD\_Variability\_Between = 0.1

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

Psychometric = expand.grid(ID=ID, ConditionOfInterest=ConditionOfInterest, StandardValues=StandardValues, reps = 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 and we add between-subject variability, and factor between-subject variability in.

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),

SD = abs(SD\_Standard + (ConditionOfInterest==1)\*SD\_Standard\*JND\_Difference))

Psychometric = Psychometric %>%

mutate(

Mean = Mean\*PSE\_Factor\_ID,

SD = SD\*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. For the method of constant stimuli, the responses are typically uniformly distributed across 5 to 9 values around the standard stimulus strength.

if (Type\_ResponseFunction == "normal"){

Psychometric = Psychometric %>%

mutate(

staircase\_factor = pnorm(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)))}

We then use these multipliers ("**staircase\_facto**r") 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**").

Psychometric = Psychometric %>%

mutate(

staircase\_factor = rcauchy(length(reps),1,SD\_ResponseFunction),

Presented\_TestStimulusStrength = Mean\*staircase\_factor,

Difference = Presented\_TestStimulusStrength - StandardValues)

Then, we compute the probability on each trial to judge the test stimulus intensity as higher (e. g. the test stimulus was faster, brighter, longer, ...) 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. Figure 3 illustrates the stimulated data set for five subjects, where both PSE and JND differ between conditions.

Psychometric = Psychometric %>%

mutate(

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 likely occur to some extent 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))

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.

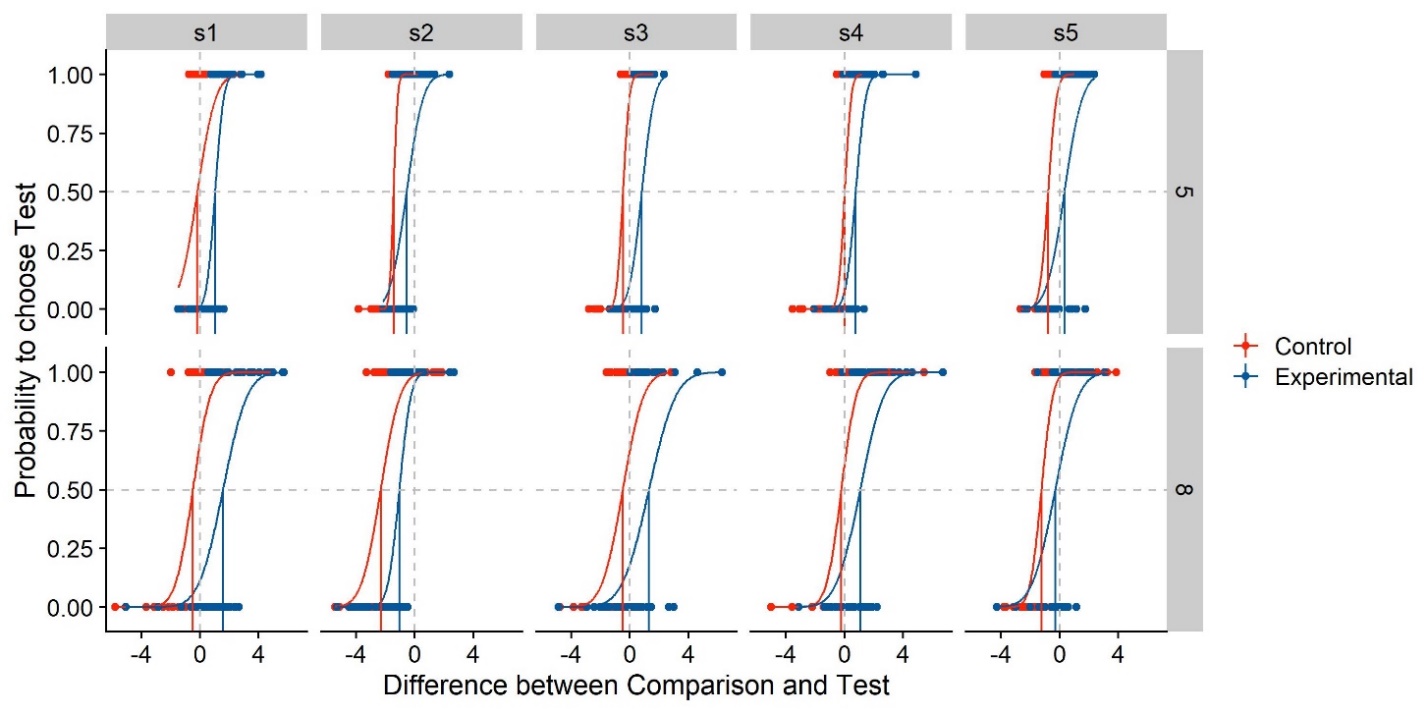


Figure 3: Simulated psychometric functions based on the example values chosen above. We plot the difference in stimulus intensity between test and standard stimulus (x axis) against the participants’ probability to choose the test stimulus as more intense (y axis). Different panels are the psychometric 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 their means and standard deviations as proxies for PSEs and JNDs (see ) and conducting the appropriate statistical test over these values (usually a *t* test or an ANOVA). We will call this two-level approach, following Moscatelli et al. (2012), “Parameter-As-Outcome Model” or short “PAOM”.
* Following Moscatelli et al. (2012), one can use Generalized Linear Mixed Models 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.

In the following, we will discuss the advantages and limitations of both methods and illustrate them with data simulated above.

## Parameter-As-Outcome Model (PAOM)

**How to** – The first step for PAOM methods is the estimation of means and standard deviations of those psychometric functions with the best fits for the observed (or simulated) data. We can extract means and standard deviations from the fit we used above to display the simulated psychometric functions:

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 load the “lmerTest” package which computes p values for ANOVAs and includes these in the regular summary() out put and conduct 2x2 ANOVAs (StandardValues x ConditionOfInterest) for both means and standard deviations.

require(lmerTest)

ANOVA\_Mean = aov(Mean ~ as.factor(ConditionOfInterest)\*StandardValues,Parameters)

ANOVA\_SD = aov(SD ~ as.factor(ConditionOfInterest)\*StandardValues,Parameters)

Then, we can call display the output of the ANOVAs by calling the summary() function on the fitted ANOVA objects:

summary(ANOVA\_Mean)

summary(ANOVA\_SD)

**Discussion** – Most researchers in this area will be familiar with this technique. However, it has three important pitfalls. First, statistical power is lost, especially when we have many trials for each condition, because this approach disregards the reliability of each mean and standard deviation estimate and treats them as one measurement each. In our example script, each staircase is measured in 100 trials. With the PAOM, 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 t test of an 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 this threshold substantially sooner or later than others. Third, one of the assumptions of an ANOVA is the independence of the dependant variables. This assumption is often violated in these analyses: mean and standard deviation estimates from one participant are likely more similar to other mean and standard deviation estimates from the same participant than to estimates from other 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.

**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. For example, in our case, they can account for the fact that the psychometric functions may be shifted consistently to the left or to the right for some participants, but not for others, or for that the fact that the psychometric functions are shallower for some, and steeper for others. Figure 4 illustrates this difference graphically. A regular 2x4 ANOVA (2 Conditions x 4 Standard Values) does not distinguish between participants (see Figure 4A), which leads to a higher variability overall. A Linear Mixed Model with Condition and Standard Values and their interaction as main effects and random intercepts per participant (See Figure 4B), allocates some of the variability in the data to idiosyncrasies of each participant. The lme4 package (Bates, Mächler, Bolker, & Walker, 2015) for R provides a user-friendly interface for the fitting of Linear Mixed Models. For the above example, the model specification could look like the following:

LMM = lmer(Mean/StandardValues ~ ConditionOfInterest\*StandardValues + (1 | ID),

data = Parameters)

summary(LMM)

This filters out some of the random variability and lowers the standard errors associated with the effects, thus raising power. For example, for the data presented in Figure 4, an ANOVA (**ConditionOfInterest** x **StandardValues**) yields a regression coefficient of -0.1, an associated standard error of 0.17 and a p value of 0.58 for **ConditionOfInterest**. A Linear Mixed Model that also adds random intercepts per participant, in turn, yields a regression coefficient of -0.1, an associated standard error of 0.04 and a p value of 0.032. The Mixed Model analysis thus provides the same coefficient estimate, but a much lower standard error and even a significant difference between **Baseline** and **ConditionOfInterest** that wasn’t captured by the ANOVA. As for the random effects, it yields intercepts of -0.06 for S01, 0 for S02, -0.1 for S03, 0.15 for S04 and 0.025 for S05, which – as expected – corresponds roughly to these participants average baseline performance (see Figure 4B).

It has to be noted that a repeated measures ANOVA with participants as grouping factor is roughly equivalent to the Linear Mixed Model established above. However, Mixed Models have some advantages with regards to repeated measures ANOVAs: They deal better with missing data and are more flexible. For example, they allow the inclusion of more than one grouping factor (or “random effects” in Mixed Model linguo), which can come in handy. We therefore recommend using Mixed Models over repeated measures ANOVAs. For further information and an R-based introduction to Mixed Models, please see Brown’s guide (Brown, 2020).

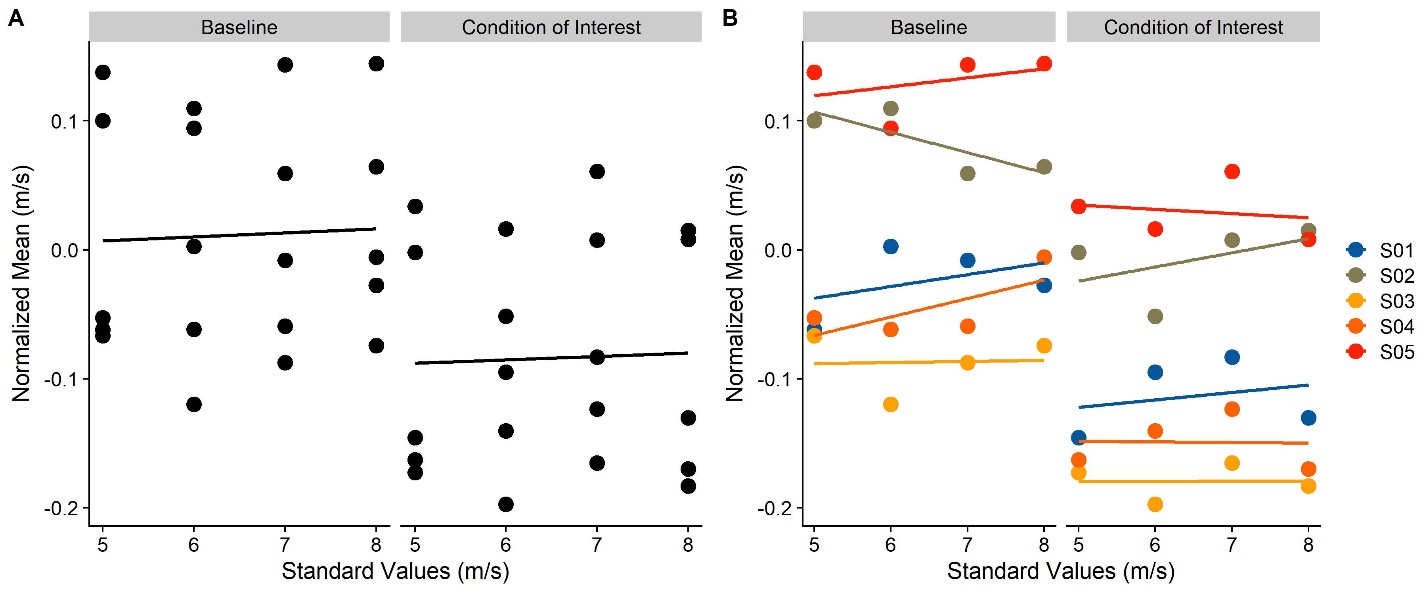


Figure 4: Means of simulated psychometric functions for five subjects (S01, S02, S03, S04, S05) and four different standard values (5, 6, 7, 8 m/s), for the baseline stimulus (left panel) and the condition of interest (right panel). In A, different participants are not color coded, mimicking the statistical behavior of a regular ANOVA. In B, the participants are color-coded, mimicking the statistical behavior of a Linear Mixed Model with random intercepts per participants or a Repeated Measures ANOVA.

We recommend (Brown, 2020) as an approachable, yet thorough introduction to Mixed Effect Models in R.

## Generalized Linear Mixed Modelling (GLMM)

**How to** – We recommend Moscatelli et al.’s paper (Moscatelli et al., 2012) as a more thorough discussion of the specifics of this method, both in terms of rationale and process. The general idea is to fit regression lines to the data that are transformed by a so called “link function”, such that they can be captured by a linear fit. The relevant link functions that R offers are “probit” and “logit” links. Both allow to transform the typical psychometric function types, Weibull or Cummulative Gaussian functions, into linear. We follow Moscatelli et al. in choosing a probit link. We first filter for potential presented outlier values (e.g., two mistakes early in a staircase can lead to the presentation of extreme values that might influence the analysis unduely). Then, we bring the data into a suitable format for the glmer() function from the lme4 package (Bates et al., 2015): it requires the responses for each stimulus value to be represented as a fraction of “Yes” responses and the total number of trials where this stimulus strength was presented.

Psychometric = Psychometric %>%

filter(abs(staircase\_factor-1) < 0.75) %>% group\_by(ID,ConditionOfInterest,StandardValues,Difference) %>%

mutate(Yes = sum(Answer==1),

Total = length(ConditionOfInterest))

We can then proceed to fitting the Generalized Mixed Model. Here, 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 is loaded (Kuznetsova, Brockhoff, & Christensen, 2017), 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 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 determine 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 p values.

anova(GLMM\_RandomIntercepts\_PSE,GLMM\_RandomIntercepts\_Null\_PSE)

anova(GLMM2\_RandomInterceptsAndSlopes\_PSE,GLMM2\_RandomInterceptsAndSlopes\_Null\_PSE)

**Discussion** – 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)

While the model fit was superior for (b), the fixed effect coefficient estimates were roughly equal across both options. In our simulated dataset, a Likelihood Ratio Test yielded no 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.

The main assumption for Generalized Linear Mixed Models is *homoskedasticity*: variance in responses should not depend in any lawful way on any experimental variables. For Linear Models, heteroskedasticity 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)

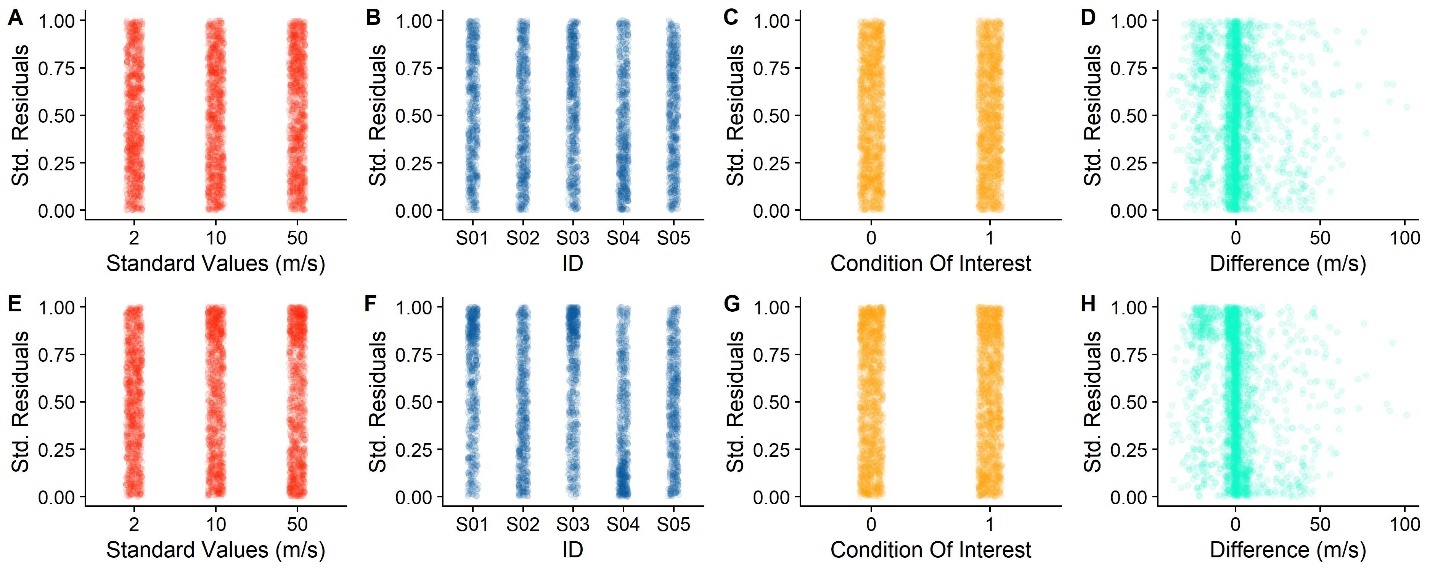


Figure 5. A, B, C, D. Standardized Residuals for the simple model, divided up by Standard Values (A), participant (B), Baseline versus Condition of Interest (C) and Differences (D). E, F, G, H. Same, but for the more complex model.

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**.

Linear Models can furthermore be unreliable if the underlying formula is not specified correctly. This can be detected by plotting the values predicted by the models against the observed values, as a quantile-quantile (QQ) plot. 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. In particular, the observed residuals should depend linearly

## Bayesian Linear Mixed Modelling

Finally, there are Bayesian methods of estimating JND and PSE. Particularly the brms package (Bürkner, 2018) for R, is an interesting alternative that is, due its similar syntax, very user-friendly for lme4 users.

The following is the most complete version, with **ConditionOfInterest** and **Difference** and their interaction as fixed effects, as well as **ConditionofInterest**, **Difference** and the intercept per **ID** and **StandardValues** as random effects.

GLMM = glmer(cbind(Yes, Total - Yes) ~ ConditionOfInterest\*Difference + (ConditionOfInterest+Difference| ID) + (ConditionOfInterest+Difference| StandardValues),

family = binomial(link = "probit"),

data = Psychometric)

# Conducting Power Analyses for Psychophysical Data

A further assumption is the absence of overdispersion and underdispersion. Overdispersion refers to an observed variability that much higher than the variability predicted by the model, and underdispersion designates an observed variability that is much lower than expected. Again, the DHARMa package offers a simple dispersion test. We perform this test over the simple and the complex model above and also add a third model with a medium complexity.

Sim\_Medium = simulateResiduals(GLMM2\_RandomInterceptsAndSlopes\_JND)

testDispersion(Sim\_Simple)

testDispersion(Sim\_Medium)

testDispersion(Sim\_Complex)

The test is significant for the simplest model (Sim\_Simple; with a Dispersion Factor of 1.02; p = 0.016), and not significant for the medium model (Sim\_Medium; Dispersion Factor of 1.05; p = 0.34) and the complex model (Sim\_Complex; Dispersion Factor of 1.34; p = 0.36). The Dispersion Factor is the quotient of the sum of squared Pearson residuals and the residual degrees of freedom. A value of 1 constitutes a perfect dispersion. Values below 1 indicate underdispersion and values above 1 indicate overdispersion.