# Introduction

The so-called reproducibility crisis has shaken Psychology to the core. Many effect that the scientific community had deemed established, could not be reproduced in highly powered replication studies. Among these are, prominently, Ego Depletion (@Hagger2016, but see also @Dang) and Terror Management Theory (@Klein2019). @Klein2016 tested a whole series of effects, some of which were not reproduced at all, and some of which were reproduced in terms of statistical significance, but with smaller effect sizes. Different reasons for the lack of reproducibility have been suggested: wide-spread p-hacking, Hypothesizing After Results are Known ("HARKing"), publication bias, lacking power and lacking theory.

Interestingly, Cognitive Psychology, and more specifically Cognitive Psychology that relies on psychophysical methods, has generated less attention in terms of the reproducibility of its results. @Hesse1986 compared different psychophysical methods along criteria such as threshold estimates, efficiency (that is, how many trials it takes to achieve reliable results) and their intra-subject reproducibility in the auditory domain. The reproducibility in the psychophysical measurement of pain sensations has received some attention (@Rosier, @Nilsson).

Prima facie, the study of perception is a methodological outlier in several aspects: the number of subjects tested in a typical experiment is quite low, starting from two or three in older papers. And even the typical sample size in modern studies is rarely much higher than 10. On the other hand, each subject typically performs large numbers of trials, starting from around 50 per condition up to several hundred. Furthermore, psychophysical studies are much more likely to be between-subject designs, which lowers the random variability in responses, thus raising power.

# Different Approaches to Null Hypothesis Testing in Psychophysics

Interestingly, psychophysics is one of the few areas of psychology where Furthermore, there are different established ways of hypo results. Classically, pychometric functions (Cummulative Gaussian or Weibull functions) are fitted for each condition and participant to obtain the Points of Subjective Equality (PSEs) and Just Noticeable Differences (JNDs). This yields one data point per subject and condition, over which a t.test or an ANOVA is performed to test for statistical significance between conditions. This approach 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 approach sacrifices vast amounts of statistical power. As a solution, @Moscatelli2012 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 following, we will estimate power for both types of analyses, to quantify how much power is lost with the two-level approach.

Interestingly, Cognitive Psychology, and more specifically Cognitive Psychology that relies on psychophysical methods, has generated less attention in terms of the reproducibility of its results. @Hesse1986 compared different psychophysical methods along criteria such as threshold estimates, efficiency (that is, how many trials it takes to achieve reliable results) and their reproducibility in the auditory domain. Furthermore, the reproducibility in the psychophysical measurement of pain sensations has received some attention (@Rosier, @Nilsson).

Prima facie, the study of perception is a methodological outlier in several aspects: the number of subjects tested in a typical experiment is quite low, starting from two or three in older papers (CITATION NEEDED). And even the typical sample size in modern studies is rarely much higher than 10. On the other hand, each subject typically performs large numbers of trials, starting from around 50 per condition up to several hundred. Furthermore, there are different established ways of analyzing results. Classically, psychometric functions (Cumulative Gaussian or Weibull functions) are fitted for each condition and participant to obtain the Points of Subjective Equality (PSEs) and Just Noticeable Differences (JNDs). The mean of the Cumulative Gaussian corresponds to the PSE and its standard deviation corresponds to the JND. This yields one data point per subject and condition, over which a t-test or an ANOVA are performed to test for statistically significant differences between conditions.

[FIGURE WITH PSYCHOMETRIC FUNCTION]

This approach neglects that each PSE and JND is based on a large number of trials, which leads to a loss in statistical power. @Moscatelli2012 have suggested the use of General Linear Mixed Modelling (GLMM). GLMM allows to obtain population-wide parameters for PSEs and JNDs, while still accounting for inter-subject variability in responses. Mixed Modelling is a more flexible form of linear regression. It allows to fit regression coefficients across the whole population for some parameters, while allowing the coefficients for other parameters to vary within subgroups of the dataset. A classic example is the modelling of the efficacy of a learning intervention on population of students from different classes in one school. Mixed models can account for inherent performance differences between classes, but fit a population-wide coefficient for the efficacy of the intervention. General Linear Mixed Modelling extends this principle by allowing to fit not only linear regression lines, but also other functions – such as cumulative Gaussians, which are commonly used as approximations for psychometric functions. You can find a more thorough explanation and examples in @Moscatelli2012.

One of improvements that have been demanded in the wake of the reproducibility crisis is a more thorough and meticulous study planning. Researchers need to be more aware of underlying theoretical considerations, specify hypotheses before analyzing the data, make precise predictions of how their hypotheses should manifest in their data and formulate statistical models to test these hypotheses. One important step in this process is to make sure that the experiment has sufficient statistical power to detect the postulated effects. We will first show that many experiments using the two-level approach of hypothesis testing described above often lacks power and indicate how much power could be gained for the same experimental designs by using @Moscatelli2002's GLMM approach. In the second part of this paper, we will demonstrate how to plan the sample size in a psychophysical experiment, levering the advantages of the GLMM approach.

# Power analyses for the GLMM approach using simulations in R

In the following we will provide an example of how to compute the power for common psychophysical designs, using the GLMM approach for analysis. Further below, we will also compare the power we obtain for the same designs when using the Two-Level approach.

## Information Needed

This method requires you to provide estimates of all relevant parameters. Some pertain to the stimuli, some can be taken from the literature, and some have to 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.

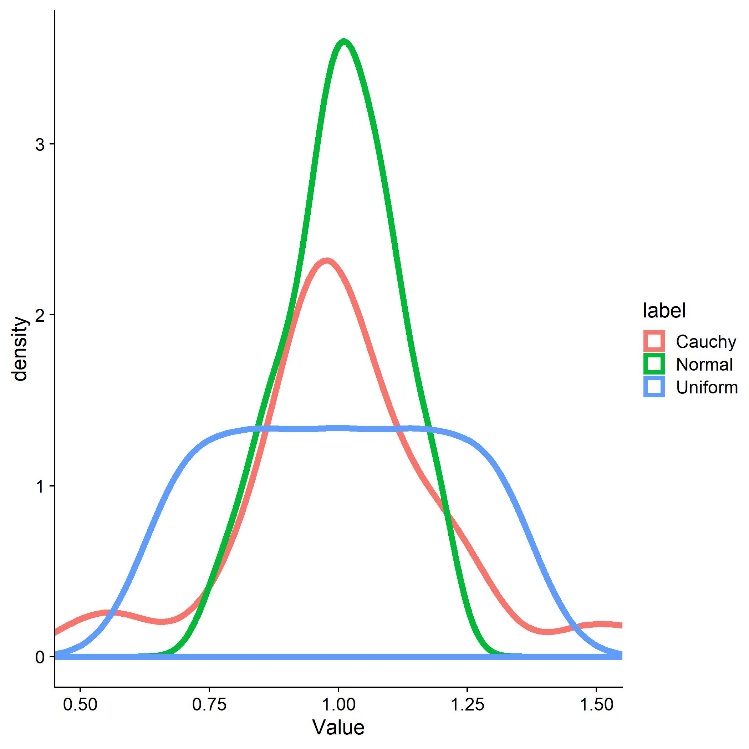
**JND\_Difference** is a value that indicates the percentage to which the JNDs differ between test and standard condition. It can be zero if the condition of interest is not expected to influence JNDs.

**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 a multiplicator that transforms Weber Fractions or JNDs from the literature into standard deviations for each comparison stimulus.

**SD\_Standard** is the Standard Deviation of the psychometric function expected for the standard condition. This standard deviation is proportional to the relevant Weber fraction and JNDs, which are available in the literature. Further below, we explain the simple link between standard deviations of psychometric functions, Weber fractions and JNDs.

**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 XX 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. For the Method of Constant Stimuli, no randomness is involved in how the presented stimulus strengths are chosen. In this case, you can therefore simply set the values you have chosen for your stimulus.



**SD\_ResponseFunction** further describes the 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.

ID = paste0("s",1:5)

ConditionOfInterest = c(0,1)

StandardValues = c(6.6, 8, 10)

reps = seq(1,55,1)

PSE\_Difference = 1/8

JND\_Difference = 1/3

PSE\_Standard = StandardValues

Multiplicator\_SD\_Standard = 0.108

SD\_Standard = StandardValues\*Multiplicator\_SD\_Standard

Type\_ResponseFunction = "Cauchy"

SD\_ResponseFunction = 0.06

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, that is a total of length(ID) x length(ConditionOfInterst) x length(StandardValues) x length(reps) rows.

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. Omitting this step amounts to the assumption that thisthe effect of interest is equally strong in each participant.

Psychometric = Psychometric %>%

group\_by(ID) %>%#

mutate(PSE\_Factor\_ID = rnorm(1,1,Mean\_Variability\_Between), #how much variability is in the means of the psychometric functions between subjects?

SD\_Factor\_ID = rnorm(1,1,SD\_Variability\_Between)) #how much variability is in the standard deviations of the psychometric functions between subjects?

Next, we simulate means and standard deviations of the psychometric functions for each condition and we add between-subject variability. The following lines take the standard PSEs (PSE\_Standard) and standard deviations (SD\_Standard) as baseline and add the PSE and JND difference in trials assigned to the test condition.

Psychometric = Psychometric %>%

mutate(

Mean = (StandardValues + (ConditionOfInterest==1)\*StandardValues\*PSE\_Difference),

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

Now, we factor the between-subject variability in.

Psychometric = Psychometric %>%

mutate(

Mean = Mean\*PSE\_Factor\_ID,

SD = SD\*SD\_Factor\_ID)

Next, 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 a normal distribution with relatively low standard deviations or to a Cauchy distribution with a low scale. A good way to determine the most appropriate function would be to plot the 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. 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").

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

Psychometric = Psychometric %>%

mutate(

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

} else if (Type\_ResponseFunction == "Cauchy"){

Psychometric = Psychometric %>%

mutate(

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

#} else if (Type\_ResponseFunction == "uniform"){

#

# Psychometric = Psychometric %>%

# mutate(

# staircase\_factor = seq(SD\_ResponseFunction[1],SD\_ResponseFunction[2],(SD\_ResponseFunction[2]-SD\_ResponseFunc#tion[1]/6)))

} else{

print("distribution not valid")

}

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 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.

Psychometric = Psychometric %>%

mutate(

AnswerProbability = pnorm(Presented\_TestStimulusStrength,Mean,SD),

##get binary answers ("Test was stronger" yes/no) from probabilities for each trial

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). 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.

###prepare for glmer() - needs sum of YES/Total per stimulus strength and condition

Psychometric = Psychometric %>%

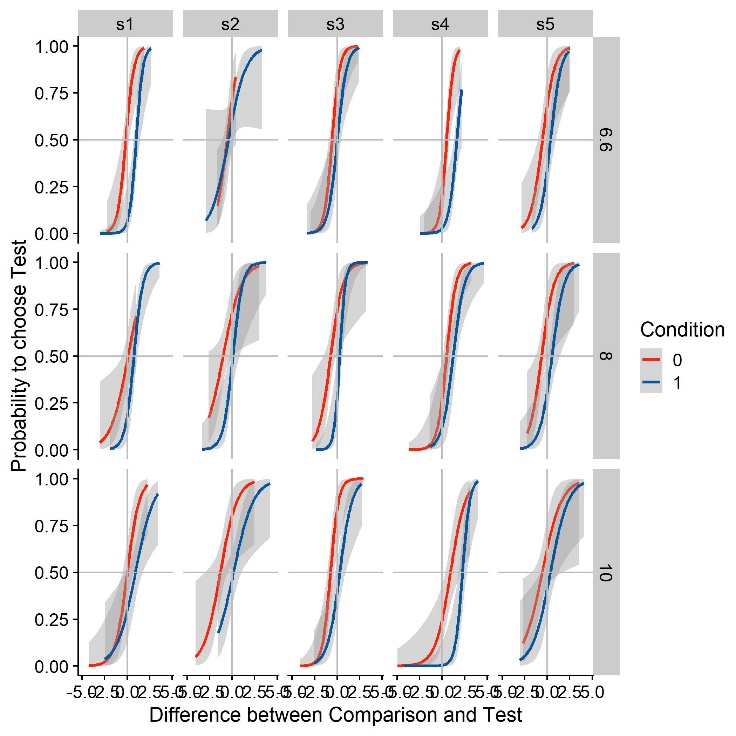
filter(abs(Difference) < 0.5\*abs(Mean)) %>%

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. Figure XX illustrates the simulated psychometric function for the above values. As is usual for psychometric functions, we have the difference in stimulus strength between test and comparison on the x axis, the probability to choose the test stimulus as stronger on the y axis. The Condition of Interest is color-coded in blue, while the baseline condition is color-coded in red. The facets correspond to IDs and different standard stimulus strengths, respectively.



## Estimating population parameters of the psychometric functions with the GLMM approach

### Accuracy

Next, we establish the statistical models we use to test our hypotheses. Following (Moscatelli 2012), we use Generalized Linear Mixed Models for this purpose. For differences in PSEs in our simulated data set, this model would look like the following:

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

family = binomial(link = "probit"),

data = Psychometric,

nAGQ = 0,

control = glmerControl(optimizer = "nloptwrap"))

Note that there are different ways of specifying the model depending on assumptions about the data and the hypotheses. This sample specification:

* Assumes
  + That we are interested in a population-wide estimate of the the impact of the condition of interest on PSEs (fixed effect of Condition of Interest)
  + That we are not interested in its population-wide impact on JNDs (no interaction between "Condition of Interest" and "Difference")
* Allows intercepts and slopes to vary per participant. Intercepts correspond to PSEs, while slopes correspond to JNDs. That is, we acommodate individual differences in sensitivity and accuracy.
* Allows intercepts and slopes to vary per standard value. It thus acommodates that different standard values might lead to differences in PSEs and JNDs. For example, higher standard values should lead to more shallow slopes if Weber Fractions hold for the stimulus type under investigation.

Applying the summary() function to the statistical model yields estimates for the coefficients, along with standard errors. Furthermore, the lmerTest package provides the possibility to compute p values using the Satherthwaite degrees of freedom method. The authors of the lme4 package recommend to rely on coefficients and their standard errors alone to estimate the impact of the Condition of Interest. p values from the lmerTest package should thus be regarded as complimentary rather than essential tool. Nonetheless, we believe that p values for the variables of interest are and appropriate, simple proxy for detection of differences between conditions, that allows a quick-and-dirty judgement of whether the Condition of Interest has a significant impact on a simulated dataset. After loading lmerTest, we can inspect coefficient, standard errors and p values as follows:

require(lmerTest)

summary(GLMM\_Accuracy)$coefficients

### Precision

Keep in mind that, in the above model, the coefficient of "Difference" corresponds to the slope of the psychometric function. To capture to what extent a manipulation impacts the slope (as a proxy for precision), one needs to determine how much the slope differs between two conditions. That is, we need to estimate the interaction between "Condition of Interest" and "Difference" in our model. A sample specification is the following:

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

family = binomial(link = "probit"),

data = Psychometric,

nAGQ = 0,

control = glmerControl(optimizer = "nloptwrap"))

We can again use the summary() function in combination with the lmerTest package to extract coefficients, standard errors and p values:

summary(GLMM\_Precision)$coefficients

Computing the Power from Simulations

To simulate the power with a given set of parameters, we need to execute the above procedure sufficient times (we recommend 1000 times, although this might be too time consuming for studies with a high count of subjects and or trials; further below we give some recommendations on how to speed up the relatively slow glmer() model fitting implementation in the R package lme4), and calculate the ratio of simulations in which the test model is significantly better than the test model, given a certain false positive rate (typically 0.05). To this end, we establish functions containing the above procedure:

SimulatePsychometricFunction\_Staircase = function(ID, ConditionOfInterest, StandardValues, reps, PSE\_Difference, JND\_Difference, Mean\_Standard, SD\_Standard, SD\_ResponseFunction,Mean\_Variability\_Between = 0.1, SD\_Variability\_Between = 0.1){

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

Psychometric = Psychometric %>%

group\_by(ID) %>%#

mutate(PSE\_Factor\_ID = rnorm(1,1,Mean\_Variability\_Between),

SD\_Factor\_ID = rnorm(1,1,SD\_Variability\_Between))

Psychometric = Psychometric %>%

mutate(

Mean = (StandardValues + (ConditionOfInterest==1)\*StandardValues\*PSE\_Difference)\*PSE\_Factor\_ID,

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

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

Presented\_TestStimulusStrength = Mean\*staircase\_factor,

Difference = Presented\_TestStimulusStrength - StandardValues,

AnswerProbability = pnorm(Presented\_TestStimulusStrength,Mean,SD),

Answer = as.numeric(rbernoulli(length(AnswerProbability),AnswerProbability))

)

Psychometric = Psychometric %>%

filter(abs(Difference) < 0.5\*abs(Mean)) %>%

group\_by(ID,ConditionOfInterest,StandardValues,Difference) %>%

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

Total = length(ConditionOfInterest))

Psychometric

}

Analyze\_Pychometric\_Accuracy\_GLMM = function(Psychometric){

TimeBeginning = Sys.time()

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

family = binomial(link = "probit"),

data = Psychometric,

nAGQ = 0,

control = glmerControl(optimizer = "nloptwrap"))

p = summary(GLMM\_Accuracy)$coefficients[8]

#print(TimeBeginning - Sys.time()) ###This is two show how long each iteration takes

#print(p)

p

}

Analyze\_Pychometric\_Precision\_GLMM = function(Psychometric){

TimeBeginning = Sys.time()

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

family = binomial(link = "probit"),

data = Psychometric,

nAGQ = 0,

control = glmerControl(optimizer = "nloptwrap"))

p = summary(GLMM\_Precision)$coefficients[16]

#print(p)

p

}

Now, we determine for how many subjects we want to simulate the power. Fewer than 10 subjects are usually not recommended.

NumbersOfSubjects = c(10,12,14,16,18,20) #for how many subjects are we simulating the power?

Then, we execute the above method "nIterations" times for each number of subjects specified above, both for precision and accuracy.

nIterations = 1000

pvalue = 0.05

Power = data.frame()

for (i in NumbersOfSubjects){

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

TimeBeginning = Sys.time()

Dataframe\_Temp = c()

for (j in 1:nIterations){

Dataframe = SimulatePsychometricFunction\_Staircase(ID, ConditionOfInterest, StandardValues, reps, PSE\_Difference, JND\_Difference, Mean\_Standard, SD\_Standard, SD\_ResponseFunction, Mean\_Variability\_Between, SD\_Variability\_Between)

Parameters = GetParametersOfPsychometricFunction(Dataframe)

p = c(Analyze\_Pychometric\_Accuracy\_GLMM(Dataframe),

Analyze\_Pychometric\_Precision\_GLMM(Dataframe))

Dataframe\_Temp = rbind(Dataframe\_Temp,p)

if ((j/50) %in% 1:(nIterations/50)){ #print the number of the current iteration every 50 trials

(print(j))

}

}

Power = rbind(Power,

data.frame(value = c(mean(Dataframe\_Temp[,1] < pvalue),

mean(Dataframe\_Temp[,2] < pvalue)),

label = c("Accuracy GLMM",

"Precision GLMM"),

nSubjects = i))

print(paste0("This iteration has taken ", Sys.time() - TimeBeginning)) ###This is two show how long each iteration takes

print(paste0("Accuracy GLMM for ", i, " subjects: ", mean(Dataframe\_Temp[,1] < pvalue))) #outputs an estimate of the power for each n

print(paste0("Precision GLMM for ", i, " subjects: ", mean(Dataframe\_Temp[,2] < pvalue))) #outputs an estimate of the power for each n

}

Once the simulation process is finished, we plot the data for each number of subjects:

ggplot(Power,aes(nSubjects,value, color = label)) +

geom\_line(size = 2) +

xlab("N° of Subjects") +

ylab("Power") +

scale\_color\_manual(values = c(BlauUB,Red)) +

geom\_hline(linetype = 2, yintercept = 0.8) +

geom\_hline(linetype = 3, yintercept = 0.95) +

scale\_x\_continuous(breaks=c(10,12,14,16,18,20))

# Comparing the power for the GLMM and the Two-Level approach

Moscatelli & Lacquaniti (2012) argue that power is lost when using the Two-Level approach. While this is an intuitive notion, it has, to our knowledge, not been confirmed analytically nor synethetically. We thus use the above power simulations to quantify just how much power is lost when using the Two-Level approach in comparison to the GLMM approach. We will use the above procedure to simulate power for the GLMM approach.

## Estimating population parameters of the psychometric functions with the Two-Level approach

For the Two-Level approach, one would first fit psychometric functions for each condition and participant. Then, one would conduct a t.test or an anova to test whether they are different. While there are different methods to fit psychometric functions that each have their own benefits, we use a direction likelihood maximization method (Prins & Kingdom 2010; Knoblauch & Maloney 2012), implemented in the R package quicksy (Linares 2017). 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 deactive the bootstrap option, which speeds up the fitting process significantly.

require(quickpsy)

Parameters = quickpsy(Psychometric,

Difference,

Answer,

grouping = .(ID,ConditionOfInterest,StandardValues),

bootstrap = "none")$par

Then, we extract the parameters and bring the output of quickpsy into the adequate format for ANOVA analysis.

Parameters2 = Parameters %>%

filter(parn == "p1") %>%

select(ID,ConditionOfInterest,Mean=par, StandardValues)

Parameters2$SD = Parameters$par[Parameters$parn == "p2"]

Finally, we perform an ANOVA over means and standard deviations of the fitted psychometric functions. We are mainly interest in the main effect of Condition of Interest, so we extract the p value for this main effect for each means and standard deviations of the psychometric function.

ANOVA\_Mean = aov(Mean ~

as.factor(ConditionOfInterest)\*

StandardValues,

Data = Parameters2)

summary(ANOVA\_Mean)

ANOVA\_SD = aov(SD ~

as.factor(ConditionOfInterest)\*

StandardValues,

Data = Parameters2)

summary(ANOVA\_SD)

## Comparing the statistical power of each approach

We pack these computations into functions and follow the same approach as above to compute the power for PSE and JND differences with the GLMM approach and the Two-Level approach. We use the same values we chose above.

GetParametersOfPsychometricFunction = function(Psychometric){

Parameters = quickpsy(Psychometric,

Difference,

Answer,

grouping = .(ID,ConditionOfInterest,StandardValues),

bootstrap = "none")$par

Parameters2 = Parameters %>%

filter(parn == "p1") %>%

select(ID,ConditionOfInterest,Mean=par, StandardValues)

Parameters2$SD = Parameters$par[Parameters$parn == "p2"]

Parameters2

}

Analyze\_Pychometric\_Precision\_2Level = function(Parameters){

ANOVA\_SD = aov(SD ~

as.factor(ConditionOfInterest)\*

StandardValues,

Parameters)

Coefficients = summary(ANOVA\_SD)[[1]]

Coefficients$`Pr(>F)`[1]

}

Analyze\_Pychometric\_Accuracy\_2Level = function(Parameters){

ANOVA\_Mean = aov(Mean ~

as.factor(ConditionOfInterest)\*

StandardValues,

Parameters)

Coefficients = summary(ANOVA\_Mean)[[1]]

Coefficients$`Pr(>F)`[1]

}

Power = data.frame()

nIterations = 1000

pvalue = 0.05

for (i in NumbersOfSubjects){

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

TimeBeginning = Sys.time()

Dataframe\_Temp = c()

for (j in 1:nIterations){

Dataframe = SimulatePsychometricFunction\_Staircase(ID, ConditionOfInterest, StandardValues, reps, PSE\_Difference, JND\_Difference, Mean\_Standard, SD\_Standard, SD\_ResponseFunction)

Parameters = GetParametersOfPsychometricFunction(Dataframe)

p = c(Analyze\_Pychometric\_Accuracy\_GLMM(Dataframe),

Analyze\_Pychometric\_Precision\_GLMM(Dataframe),

Analyze\_Pychometric\_Accuracy\_2Level(Parameters),

Analyze\_Pychometric\_Precision\_2Level(Parameters))

Dataframe\_Temp = rbind(Dataframe\_Temp,p)

if ((j/25) %in% 1:40){

(print(j))

}

}

Power = rbind(Power,

data.frame(value = c(mean(Dataframe\_Temp[,1] < pvalue),

mean(Dataframe\_Temp[,2] < pvalue),

mean(Dataframe\_Temp[,3] < pvalue),

mean(Dataframe\_Temp[,4] < pvalue)),

label = c("Accuracy GLMM",

"Precision GLMM",

"Accuracy Two-Level",

"Precision Two-Level"),

nSubjects = i))

print(paste0("This iteration has taken ", Sys.time() - TimeBeginning)) ###This is two show how long each iteration takes

print(paste0("Accuracy GLMM for ", i, " subjects: ", mean(Dataframe\_Temp[,1] < pvalue))) #outputs an estimate of the power for each n

print(paste0("Precision GLMM for ", i, " subjects: ", mean(Dataframe\_Temp[,2] < pvalue))) #outputs an estimate of the power for each n

print(paste0("Accuracy 2Level for ", i, " subjects: ", mean(Dataframe\_Temp[,3] < pvalue))) #outputs an estimate of the power for each n

print(paste0("Precision 2Level for ", i, " subjects: ", mean(Dataframe\_Temp[,4] < pvalue))) #outputs an estimate of the power for each n

}

Here, we visualize the simulated power for each approach (GLMM/Two-Level) and measure (PSE/JND).

ggplot(Power,aes(nSubjects,value, color = label)) +

geom\_line(size = 2) +

xlab("N° of Subjects") +

ylab("Power") +

scale\_color\_manual(values = c(BlauUB,Red,LightBlauUB,LightRed)) +

geom\_hline(linetype = 2, yintercept = 0.8) +

geom\_hline(linetype = 3, yintercept = 0.95) +

scale\_x\_continuous(breaks=c(10,12,14,16,18,20))