Tutorial for Customized Power Simulations and Data Analyses for Human-AI Interaction Experiments using Generalized Linear Mixed Models

Timo K. Koch[1, 2], Florian Pargent[2], & Susanne Gaube[1]

[1] LMU Munich, Center for Leadership and People Management

[2] LMU Munich, Department of Psychology

[3] University of St. Gallen, Institute of Behavioral Science & Technology

[4] University College London

Author Note

Add complete departmental affiliations for each author here. Each new line herein must be indented, like this line.

Enter author note here.

The authors made the following contributions. Timo K. Koch: Conceptualization, Writing - Original Draft Preparation, Writing - Review & Editing; Florian Pargent: Writing - Review & Editing, Supervision; Susanne Gaube: Writing - Review & Editing, Supervision.

Correspondence concerning this article should be addressed to Timo K. Koch, Postal address. E-mail: timo.koch@psy.lmu.de

Abstract

Understanding how humans interact with articifical intelligence trough experiments becomes increasingly relevant. However, human-AI interaction researchers lack the appropriate tools to conduct power and data analyses for the required complex study designs. In this work, we provide a tutorial on how to run customized power analyses using data simulation based on Generalized Linear Mixed Models (GLMMs). By providing code in a case study, we equip human-AI interaction researchers to simulate their own data and run analyses based on GLMMs. We discuss the outlook.

Keywords: generalized linear mixed model, data simulation, sample size, power analysis, human-AI interaction

Word count: 5339

Tutorial for Customized Power Simulations and Data Analyses for Human-AI Interaction Experiments using Generalized Linear Mixed Models

Introduction

As the integration of artificial intelligence (AI) into our daily lives continues to increase, it has become increasingly important to study how humans interact with these intelligent systems. This is particularly important as AI becomes more sophisticated and is used in decision-making processes, for instance in medicine, that have significant impacts on individuals and society as a whole. Therefore, thorough experiments are needed to study human-AI interaction and understand how people perceive and respond to these technologies. When conducting research in this field, it is essential to determine an appropriate sample size to ensure that the results obtained are both statistically significant and reliable. Therefore, running sample size estimations is crucial for designing experiments that can provide meaningful insights into human-AI interaction.

Power simulations play a crucial role in addressing the replication crisis and meeting the requirements set forth by journals and funding agencies in Human-AI interaction research. The replication crisis has underscored the problem of high false discovery rates (FDR) coupled with low-powered studies. Power simulations offer a valuable solution by allowing researchers to estimate the appropriate sample sizes needed to achieve sufficient statistical power for reliable results. Moreover, many journals and funding agencies now mandate the inclusion of power simulations as part of study protocols and grant proposals, recognizing their significance in ensuring robust and meaningful findings. By incorporating power simulations into research planning, researchers can enhance the replicability and credibility of their work, ultimately contributing to the advancement of psychological science.

Customized power simulations are hard

Power simulations can be challenging, particularly when analyzing complex study designs using (generalized) linear mixed models (GLMMs). While user-friendly software is available for simple statistical models like t-tests, ANOVA, and linear regression, the increasing use of GLMMs in psychological research introduces greater complexity to power calculations. For GLMMs, simulations are often required, necessitating assumptions for both fixed and random effects. However, most tutorials on GLMMs primarily focus on the most common designs, limiting the guidance available for researchers facing more intricate scenarios. Moreover, existing tutorials often rely on heuristics for estimating random effect standard deviations and may incorporate meta-analytic results for both random and fixed effects, further adding to the challenge of conducting accurate power calculations for GLMMs.

When planning a Human-AI interaction experiment with complexities such as the use of Generalized Linear Mixed Models (GLMM), non-standard designs, and missing by design conditions, the process of power simulations becomes more challenging. Software packages designed for power simulations may not adequately address the complexities of the experiment, making it necessary to build data simulations tailored specifically to the study design. Unfortunately, there are no general recommendations that can be applied universally in such cases. Instead, researchers must acquire the necessary skills based on the specific application at hand. In this paper, we present a case study that serves as a practical demonstration of how to perform a bespoke data simulation for a concrete study. Through this case study, we aim to provide guidance and insights into the complexities involved in addressing complex design considerations and obtaining accurate power estimates.

LMMs, GLMMs and the lme4 package in R

Linear Mixed Models (LMM) and Generalized Linear Mixed Models (GLMM) are powerful statistical frameworks that handle complex data structures by incorporating both

fixed and random effects.

Linear Mixed Models (LMM) are extensions of linear regression models that account for correlated data and hierarchical structures. They are used when the outcome variable is continuous and follow a normal distribution. LMMs allow for the modeling of fixed effects, which capture the relationships between predictors and the outcome, as well as random effects, which account for the correlation and variability within groups or subjects. Random effects are typically assumed to follow a normal distribution with a mean of zero and a variance that quantifies the heterogeneity across the groups or subjects.

Generalized Linear Mixed Models (GLMM) extend the LMM framework to accommodate non-normal and categorical outcome variables. They are used when the outcome variable does not follow a normal distribution, but instead belongs to a different distribution family, such as binomial, Poisson, or gamma. GLMMs incorporate both fixed and random effects, similar to LMMs, but also involve a link function that connects the linear predictor to the expected value of the outcome variable. The link function allows for modeling the relationship between predictors and the outcome in a way that is appropriate for the specific distribution family of the response variable.

Generalized Linear Mixed Models (GLMMs) are gaining increasing popularity in the field of Human-AI interaction research. As the complexity of studying human interactions with artificial intelligence systems grows, researchers require more sophisticated statistical models to capture the nuanced relationships and hierarchical structures within the data. GLMMs offer a flexible framework for analyzing data with non-normal and categorical outcomes, accounting for both fixed and random effects. This versatility makes GLMMs particularly suitable for investigating various aspects of Human-AI interaction, such as user preferences, trust, engagement, and performance. By incorporating GLMMs into their analyses, researchers in the field of Human-AI interaction can obtain more robust and comprehensive insights into the intricate dynamics between humans and AI systems, leading

to a deeper understanding of the psychological and behavioral aspects involved.

The lme4 R package is a state-of-the-art tool for fitting frequentist Generalized Linear Mixed Models (GLMMs). It provides extensive capabilities for modeling complex data structures, including models with normally distributed random intercepts and random slopes. The package includes a useful function called "simulate" that allows researchers to simulate the dependent variable based on the same model formula used for model fitting, enabling power simulations and other related analyses.

In lme4, the parameterization used involves the concept of both the beta vector and the theta vector. When considering only random intercepts, the theta parameter represents the standard deviation of the random intercepts, providing information about the variability across the different levels of the random effect. This allows for a deeper understanding of the random intercepts' influence on the outcome variable.

Regarding random slopes, the interpretation of theta becomes more complex. It encompasses the covariance structure between the random slopes and the random intercepts. The theta vector accounts for the variability and correlation between these different random effects, offering insights into the multilevel nature of the data. However, it is important to note that the tutorial discussed in this paper focuses solely on the simpler case of random intercepts, while acknowledging the more intricate implications of theta in models involving random slopes.

Methods

In this section, we describe the steps to conduct a data simulation and a power analysis. In doing so, we provide a concrete example in the form of a case study.

We used R (Version 4.3.1; R Core Team, 2023) and the R-packages *papaja* (Version 0.1.1; Aust & Barth, 2022), and *tinylabels* (Version 0.2.3; Barth, 2022) for all our analyses.

The present case study

In the case study, we simulate the data for a Human-AI Interaction experiment where a health care AI is to be evaluated. Participants view head CT cases and evaluate if a bleeding is present. Also, there can be AI advice that is either correct or incorrect. This medical diagnosis can be correct or incorrect.

In this case, it would be more challenging to recruit task expert as there is only a limited amount of such people. Non-experts would be easier to recruit. The open question is how many task experts and non-experts to recruit to achieve sufficient power.

describe study design and sample size restrictions - describe research question (i.e. the coefficients targeted by power simulation)

Power simulations in generalized linear mixed models (GLMMs) are essential for estimating the statistical power of complex psychological study designs. The model equation for a GLMM combines fixed effects, random effects, and an appropriate link function to model the relationship between predictors and the outcome variable. The necessary assumptions for power simulations in GLMMs include assumptions about the distributional form of the outcome variable, the random effects, and the error structure. The distributional assumption specifies the family of distributions for the outcome variable, such as Gaussian, Poisson, or binomial. Assumptions about the random effects include the assumption of normality and the covariance structure among the random effects. Additionally, assumptions about the error structure, such as independence or correlation, must be specified. Interpreting these assumptions entails understanding the underlying assumptions of the model and ensuring they align with the characteristics of the data being analyzed.

TODO

points not discussed yet:

- discuss the specific missing by design complexity
- discuss different possible parametrizations of the fixed effects
- find the appropriate lme4 formula (simple models first, more complex models later)
- determine the correct labels for model coefficients in the lme4 output
- excursus: estimate expected with of confidence intervals for different sample sizes and number of stimuli

Generalized linear mixed models

In a Generalized linear mixed model (GLMM), the expected value of the dependent variable Y conditioned on the vector of predictor variables $\mathbf{X}$ and random effects $\mathbf{U}$, transformed by a link function $g()$ is modeled as a linear combination η of the predictor variables $\mathbf{X}$, the random effects $\mathbf{U}$ and the model parameters β.

$$g(E(Y|\mathbf{X} = \mathbf{x}, \mathbf{U} = \mathbf{u})) = \eta$$

Equivalently, the conditional expected value is modeled as the linear combination η, transformed by the inverse link function $g^{-1}()$.

$$E(Y|\mathbf{X} = \mathbf{x}, \mathbf{U} = \mathbf{u})) = g^{-1}(\eta)$$

If the dependent variable Y is a binary variable with values 0 or 1, the conditional expected value is equivalent to the probability:

$$P_{si} = P(Y = 1|\mathbf{X} = \mathbf{x}, \mathbf{U} = \mathbf{u})$$

In our case study, P_{si} is the conditional probability that subject s gives the correct response to item i.

In such a setting, we model the probability as

$$P_{si} = inverse\_logit(\eta_{si})$$

with the inverse-logit link $g^{-1}(\eta_{si}) = inverse\_logit(\eta_{si}) = \frac{exp(\eta_{si})}{1+exp(\eta_{si})}$ or equivalently

$$logit(P_{si}) = \eta_{si}$$

with the logit link $g(P_{si}) = logit(P_{si}) = ln(\frac{P_{si}}{1-P_{si}})$.

In our case study, the probability to give a correct response is assumed to depend on the predictors:

- $advice\_present_{si}$: whether subject s was presented with algorithmic advice (1) or not (0) when asked to asses item i
- $advice\_correct_{si}$: whether this advice was correct (1) or not (0)
- $expert_s$: whether subject s was a professional neurologist (1) or not (0)

and the random effects:

- u_{0s}: the deviation of suject s from the average ability to solve an item with average difficulty; assumed to be distributed as $u_{0s} \sim N(0, \sigma_S^2)$
- u_{0i}: the deviation of item i from the average difficulty to be solved by a person with average ability; assumed to be distributed as $u_{0i} \sim N(0, \sigma_I^2)$

In total, we assume the model

$$\begin{aligned} logit[P_{si}] = \ & (\beta_0 + u_{0s} + u_{0i}) + \\ & \beta_a \cdot advice\_present_{si} + \beta_c \cdot advice\_correct_{si} + \beta_e \cdot expert_s + \\ & \beta_{ea} \cdot expert_s \cdot advice\_present_{si} + \beta_{ec} \cdot expert_s \cdot advice\_correct_{si} \end{aligned}$$

or equivalently

$$\begin{aligned} P_{si} = inverse\_logit[& (\beta_0 + u_{0s} + u_{0i}) + \\ & \beta_a \cdot advice\_present_{si} + \beta_c \cdot advice\_correct_{si} + \beta_e \cdot expert_s + \\ & \beta_{ea} \cdot expert_s \cdot advice\_present_{si} + \beta_{ec} \cdot expert_s \cdot advice\_correct_{si}] \end{aligned}$$

with model parameters β_0, β_e, β_a, β_c, β_{ea}, β_{ec}, σ_S, and σ_I.

In the GLMM literature, this would be called a binomial GLMM with two random intercepts (for subjects and items), two level-1 predictors (*advice_present*, *advice_correct*), one level-2 predictor (*expert*) and two cross-level interactions ($expert \cdot advice\_present$, $expert \cdot advice\_correct$).

To limit complexity, we do not consider random slopes, additional predictors or higher-level interactions here.

Data Simulation

The following R function simulates a full dataset structured according to our case study. The faux package contains useful functions when simulating factorial designs including random effects.

```
simulate <- function(n_subjects = 100, n_items = 50,
  b_0 = 0.847, b_e = 1.350, b_a = -1.253, b_c = 2.603,
  b_ea = 0.790, b_ec = -1.393,
  sd_u0s = 0.5, sd_u0i = 0.5, ...){
  require(dplyr)
  require(faux)
  # simulate design
  dat <- add_random(subject = n_subjects, item = n_items) %>%
    add_between("subject", expert = c(1, 0), .prob = c(0.25, 0.75)) %>%
    mutate(advice_present = rbinom(n(), 1, prob = 2/3)) %>%
    mutate(advice_correct = if_else(advice_present == 1,
                                    rbinom(n(), 1, prob = 0.8), 0)) %>%
    # add random effects
    add_ranef("subject", u0s = sd_u0s) %>%
    add_ranef("item", u0i = sd_u0i) %>%
```

```
    # compute dependent variable
    mutate(linpred = b_0 + u0i + u0s +
        b_e * expert + b_a * advice_present + b_c * advice_correct +
        b_ea * expert * advice_present + b_ec * expert * advice_correct) %>%
    mutate(y_prob = plogis(linpred)) %>%
    mutate(y_bin = rbinom(n = n(), size = 1, prob = y_prob))
  dat
}
```

In our case study, each subject (`n_subjects` in total) is assumed to respond to each item (`n_items` in total). Thus the `add_random` command creates a fully-crossed `data.frame` with `n_subjects` $\times$ `n_items` rows. We add a between subject effect with the `add_between` command, simulating that about 25 of subjects are experts. The next two lines simulate that in about $\frac{2}{3}$ of trials, subjects will be presented with AI advice and if advice is presented, the advice will be correct in about 80 of cases (the variable `advice_correct` is always 0 when no advice is presented). Next we simulate one random effect each subject (`u0s`) and for each item (`u0i`). As assumed by standard GLMMs, the `add_ranef` function draws the random effects from a normal distribution with mean 0 and a standard deviation specified by the user. With all design variables done, we are ready to simulate our model equation as outlined in equation X. The linear predictor variable `linpred` (η in the GLMM model equations), combines the predictor variables, random effects and model parameters as assumed by our model. We then transform the linear predictor with the inverse-link function to compute `y_prob`, the probability that the subject correctly solved the item (in R the inverse-logit link is computed with `plogis` and the logit link with `qlogis`). In the final step, we simulate the binary dependent variable `y_bin` by – for each trial – drawing from a bernoulli distribution with success probability `y_prob`.

Model fitting

In this section, we show how to fit a GLMM with lme4, interpret the model and test hypotheses derived from a research question.

We simulate data according to our model, in which 100 subjects respond to 50 items (we use `set.seed` to make the simulation reproducible). However, for the sake of the exercise, we can imagine that this would be real data resulting from our future experiment and think about how we would analyse this data.

```
library(tidyverse)
set.seed(1)
dat <- simulate(n_subjects = 100, n_items = 50)
```

The lme4 package uses a special syntax for model specification. Our assumed GLMM is represented by the formula:

```
library(lme4)
f <- y_bin ~ 1 + expert + advice_present + advice_correct +
  expert:advice_present + expert:advice_correct +
  (1|subject) + (1|item)
```

The first two lines looks similar to any linear model in R (general intercept indicated by `1`; main effects indicated by variable names in the dataset; interactions indicated by `variable1:variable2`). The third line specifies a random intercept for each subject `(1|subject)` and for each item `(1|item)`. The complete set of rules for the syntax are outlined in (CITE LME4 PAPER) and in the documentation of the lme4 package.

In lme4, a GLMM is fitted with the `glmer` function. By setting `family = "binomial"` we request a binomial GLMM, appropriate for our binary dependent variable `y_bin` (the binomial GLMM used the canonical logit link by default).

```
fit <- glmer(f, data = dat, family = "binomial")
```

Model interpretation

We can inspect the estimated model parameters with the `summary` command:

```
summary(fit)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula:
## y_bin ~ 1 + expert + advice_present + advice_correct + expert:advice_present +
##     expert:advice_correct + (1 | subject) + (1 | item)
##    Data: dat
##
##      AIC      BIC   logLik deviance df.resid
##   4149.4   4201.6  -2066.7   4133.4     4992
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.7669  0.2125  0.3046  0.4317  2.1056
##
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.3148   0.5611
##  item    (Intercept) 0.1624   0.4029
## Number of obs: 5000, groups:  subject, 100; item, 50
##
```

```
## Fixed effects:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)             1.0339     0.1103   9.374  < 2e-16 ***
## expert                  1.1849     0.2096   5.654 1.57e-08 ***
## advice_present         -1.3436     0.1206 -11.143  < 2e-16 ***
## advice_correct          2.6154     0.1273  20.540  < 2e-16 ***
## expert:advice_present   1.0589     0.2940   3.601 0.000317 ***
## expert:advice_correct  -1.8104     0.2915  -6.211 5.27e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) expert advc_p advc_c exprt:dvc_p
## expert      -0.377
## advic_prsnt -0.349  0.176
## advic_crrct  0.023  0.001 -0.668
## exprt:dvc_p  0.143 -0.448 -0.412  0.276
## exprt:dvc_c -0.008  0.004  0.292 -0.435 -0.686
```

The output shows the estimates for all model parameters: the `Estimate` column in the `Fixed effects` table contains the estimates for the β parameters, while the `Std.Dev.` column in the `Random effects` table contains the estimates for σ_S and σ_I.

Unfortunately, the model parameters in a binomial GLMM are hard to interpret, because 1) the β parameters are connected to the modeled probability via the non-linear inverse-logit link, and 2) we also have to consider the random effects. The most simple interpretation works by imagining a subject with average ability ($u0s = 0$) responding to an item with average difficulty ($u0i = 0$). Then the model implied probability that such a

person solves such an item is given by:

$$P(Y = 1|\mathbf{X} = \mathbf{x}, \mathbf{U} = \mathbf{0}) =$$
$$= inverse\_logit[\beta_0 + \beta_a \cdot advice\_present_{si} + \beta_c \cdot advice\_correct_{si} + \beta_e \cdot expert_s +$$
$$\beta_{ea} \cdot expert_s \cdot advice\_present_{si} + \beta_{ec} \cdot expert_s \cdot advice\_correct_{si}]$$

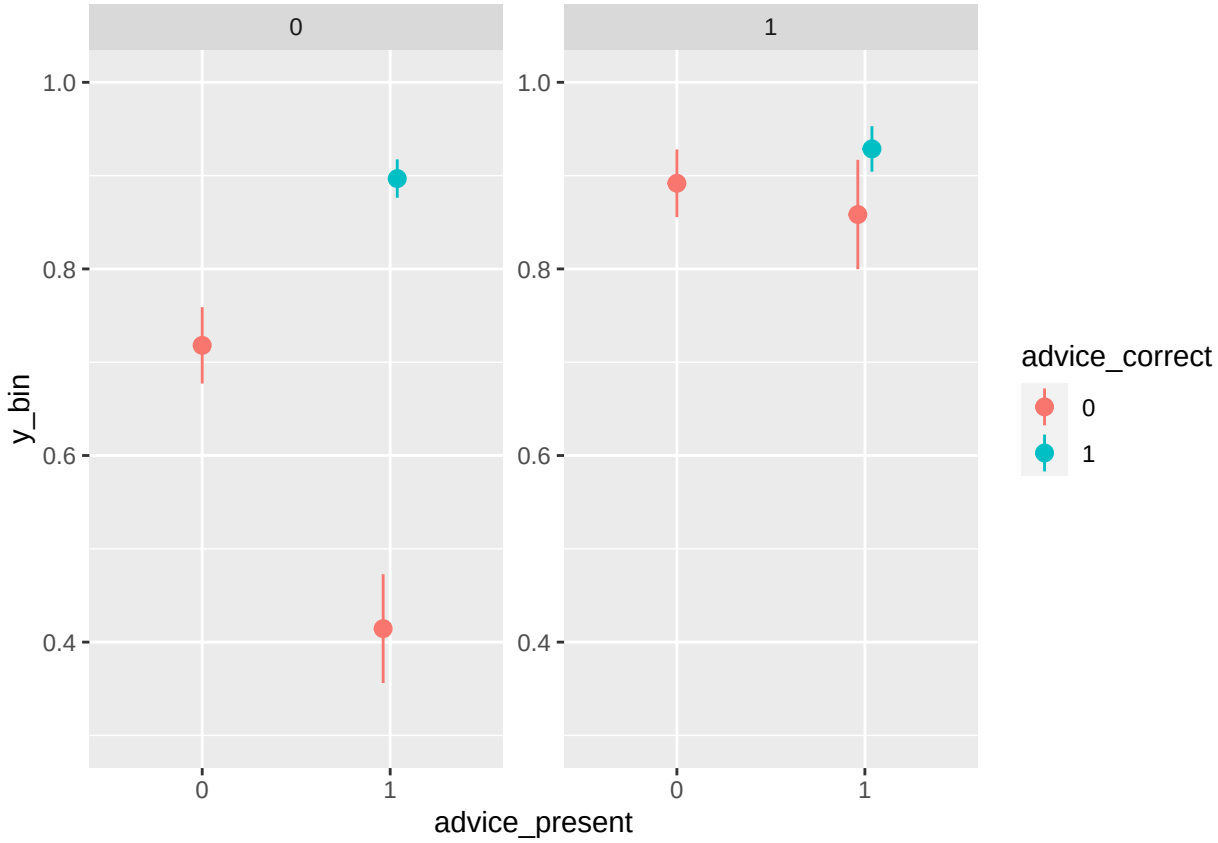
In fact, we would only need the full equation if the subject is an expert and correct advice is presented. In all other experimental conditions, some terms drop from the equation because they are multiplied by 0. The other extreme case would be the probability that a non-expert with average ability solves and item with average difficulty when presented without any advice:

$$P(Y = 1|expert = 0, advice\_present = 0, advice\_correct = 0, u_{0s} = 0, u_{0i} = 0) =$$
$$= inverse\_logit[\beta_0]$$

Due to this complicated relationship, many experts argue not to focus too much on interpreting single model parameters when working with GLMMs. Instead, it can be more intuitive to consider the implied predicted distribution of the dependent variable for each experimental conditions across all subjects and items.

With the marginaleffects package, we can easily compute predictions for all observations in the dataset based on the fitted GLMM (including all fixed **and** random effects), and plot the average probability with confidence intervals for each experimental condition:

```
library(marginaleffects)
plot_predictions(fit, by = c("advice_present", "advice_correct", "expert"),
  type = "response") + ylim(c(0.3, 1))
```



Hypothesis testing

However, we need to think about the model parameters again when wanting to test hypotheses which we have theoretically derived from some research question. Because the inverse-logit link is still a continuously increasing function, positive parameter values always correspond to increases in probability and vice versa.

The `Fixed effects` table in Figure X also includes p-values for hypothesis tests with null hypotheses of the style $H_0 : \beta = 0$. However, for many research questions of interest, we are not interested in these two-sided tests referring to only one parameter.

For our case study, imagine the following combined hypothesis: *We expect that for both experts and non-experts, correct advice leads to a higher probability to solve an item compared to no advice presented, AND, we expect that for both experts and non-experts, incorrect advice leads to a lower probability to solve an item compared to no advice presented.*

This combined hypothesis leads to the following four separate null hypotheses to be tested:

$$H_{01} : \beta_a + \beta_c + \beta_{ea} + \beta_{ec} \leq 0$$

$$H_{02} : \beta_a + \beta_c \leq 0$$

$$H_{03} : \beta_a + \beta_{ea} \geq 0$$

$$H_{04} : \beta_a \geq 0$$

We arrive at these inequalities based on the following logic, exemplified here only for H_{01}: The first null hypothesis states that *an expert responding to an item while presented with correct advice has lower or equal probability to solve the item compared to the same expert facing the same item without any advice.* This implies the following inequality for each subject s and item i

$$inverse\_logit[(\beta_0 + u_{0s} + u_{0i}) + \beta_e + \beta_a + \beta_c + \beta_{ea} + \beta_{ec}] \leq inverse\_logit[(\beta_0 + u_{0s} + u_{0i}) + \beta_e]$$

which simplifies to $\beta_a + \beta_c + \beta_{ea} + \beta_{ec} \leq 0$.

We can specify and test hypotheses like these with the multcomp package as follows:

```
library(multcomp)
null_hypotheses <- c(
  "advice_present + advice_correct + expert:advice_present +
   expert:advice_correct <= 0",
  "advice_present + advice_correct <= 0",
  "-1 * (advice_present + expert:advice_present) <= 0",
  "-1 * (advice_present) <= 0")
glht <- glht(fit, linfct = null_hypotheses)
summary(glht, test = univariate())$test$pvalues
```

```
## advice_present + advice_correct + expert:advice_present + expert:advice_correct
##                                                                     0.006407391
```

```
##                                                 advice_present + advice_correct
##                                                                    0.000000000
##                                  -1 * (advice_present + expert:advice_present)
##                                                                    0.143963670
##                                                        -1 * (advice_present)
##                                                                    0.000000000
```

Because all hypotheses tested simultaneously with the `glht` function must have the same direction, we flip the sign of inequalities three and four by multiplying them with -1. The multcomp package automatically adjusts p-values when multiple hypotheses are tested simultaneously. However, the combined null hypothesis in our exemplary research question should only be rejected if **all** individual null hypotheses are rejected. In such cases, the error probabilities do not accumulate and we would waste power when correcting for multiple testing. Thus, we request unadjusted p-values by setting `test = univariate()` in the `summary` command. With a standard significance level of $\alpha = 0.05$, we would reject all four null hypotheses and therefore also reject the combined null hypothesis for this simulated dataset.

Specification of plausible parameter values

When introducing our simulation function and simulating data for the above example, we have used theoretically plausible values as defaults for all model parameters (β_0, β_e, β_a, β_c, β_{ea}, β_{ec}, σ_S, and σ_I), but have not talked about where the numbers came from. All parameter values have been determined in repeated exchanges with our affiliated domain experts and we here outline a few strategies on how to determine plausible parameter values.

We have already seen in our discussion of model interpretation, how we can derive the model implied probability for each experimental condition, that a subject with average ability solves an item with average difficulty. We can revert this perspective by choosing

plausible probability values and deriving the parameter values implied by these probabilities (for an average subject and an average item).

Table X shows our set of assumptions concerning the probability that an average subject solves an average item for each experimental condition, as well as the corresponding equations implied by the model:

Experimental condition	$P(Y = 1 \mid \mathbf{X} = \mathbf{x}, \mathbf{U} = \mathbf{0})$	*Implied equation*
no advice, no expert	0.70	$logit(0.70) = \beta_0$
no advice, expert	0.90	$logit(0.90) = \beta_0 + \beta_e$
false advice, no expert	0.40	$logit(0.40) = \beta_0 + \beta_a$
false advice, expert	0.85	$logit(0.85) = \beta_0 + \beta_e + \beta_a + \beta_{ea}$
correct advice, no expert	0.90	$logit(0.90) = \beta_0 + \beta_a + \beta_c$
correct advice, expert	0.95	$logit(0.95) = \beta_0 + \beta_e + \beta_a + \beta_c + \beta_{ea} + \beta_{ec}$

This table can be used to compute the implied values for the β parameters, starting with the first equation and reinserting the computed β values in all following equations:

```
b_0 <- qlogis(0.7)
b_e <- qlogis(0.9) - b_0
b_a <- qlogis(0.4) - b_0
b_ea <- qlogis(0.85) - b_0 - b_e - b_a
b_c <- qlogis(0.9) - b_0 - b_a
b_ec <- qlogis(0.95) - b_0 - b_e - b_a - b_c - b_ea
c(b_0 = b_0, b_e = b_e, b_a = b_a, b_c = b_c, b_ea = b_ea, b_ec = b_ec)
```

```
##        b_0        b_e        b_a        b_c       b_ea       b_ec
##  0.8472979  1.3499267 -1.2527630  2.6026897  0.7901394 -1.3928518
```

It is always possible to double-check these computations by transforming the parameter values back to probabilities, e.g.

$$P(Y = 1|expert = 1, advice\_present = 1, advice\_correct = 1, u_{0s} = 0, u_{0i} = 0) =$$
$$= inverse\_logit[\beta_0 + \beta_e + \beta_a + \beta_c + \beta_{ea} + \beta_{ec}]$$

```
plogis(b_0 + b_e + b_a + b_c + b_ea + b_ec)
```

```
## [1] 0.95
```

Although the derivations above are straightforward, it is important not to misinterpret their implications: In binomial GLMMs, the average probability to solve an item (averaged across persons of varying ability and items of varying difficulty) is **not** equal to the probability that a person with average ability solves an item with average difficulty. For example, we determined the β parameters in a way that correspond to a desired probability, that an expert with average ability solves an item with average difficulty when presented with a correct advice. However even if the model were true, we would not observe this probability if we estimated the probability in a group of expert responding to items presented with correct advice from a big sample of subjects drawn from their natural distribution of ability and items drawn from their natural distribution of difficulty. This implies that one must be careful when specifying parameter values based on previous studies or pilot data.

The well known inequality of conditional and marginal effects in GLMMs makes their interpretation more difficult, however, this does not mean that we cannot use the marginal interpretation (average probability across persons and items) to inform plausible parameter values: When parameter values have selected, we can compute the implied marginal distributions and compare this information to our domain knowledge. Then we can iteratively adjust the parameter values until we are satisfied with the implied distributions.

Earlier, we have already encountered one way to visualize the implied marginal distributions: We can fit our model to a simulated dataset and use the convenience functions from the marginaleffects package to compute averaged predictions that correspond to our quantities of interest. However, the model predictions will only be close to the true distribution if the simulated dataset is very large, but then the model fitting consumes a lot of time and memory. A more sophisticated strategy is to simulate a large dataset and directly compute the averages, contrasts and distributions we are interested in.

```
library(tidyverse)
library(ggdist)
dat <- simulate(n_subjects = 2000, n_items = 2000, sd_u0s = 0.5, sd_u0i = 0.5)
dat %>%
  mutate(condition = fct_cross(
    factor(expert), factor(advice_present), factor(advice_correct))) %>%
  mutate(condition = fct_recode(condition,
    "no expert, no advice" = "0:0:0", "expert, no advice" = "1:0:0",
    "no expert, wrong advice" = "0:1:0", "expert, wrong advice" = "1:1:0",
    "no expert, correct advice" = "0:1:1", "expert, correct advice" = "1:1:1")) %>%
  ggplot(aes(x = y_prob, y = condition)) +
  stat_histinterval(point_interval = "mean_qi", slab_color = "gray45") +
  scale_x_continuous(breaks = seq(0, 1, 0.1), limits = c(0, 1))
```

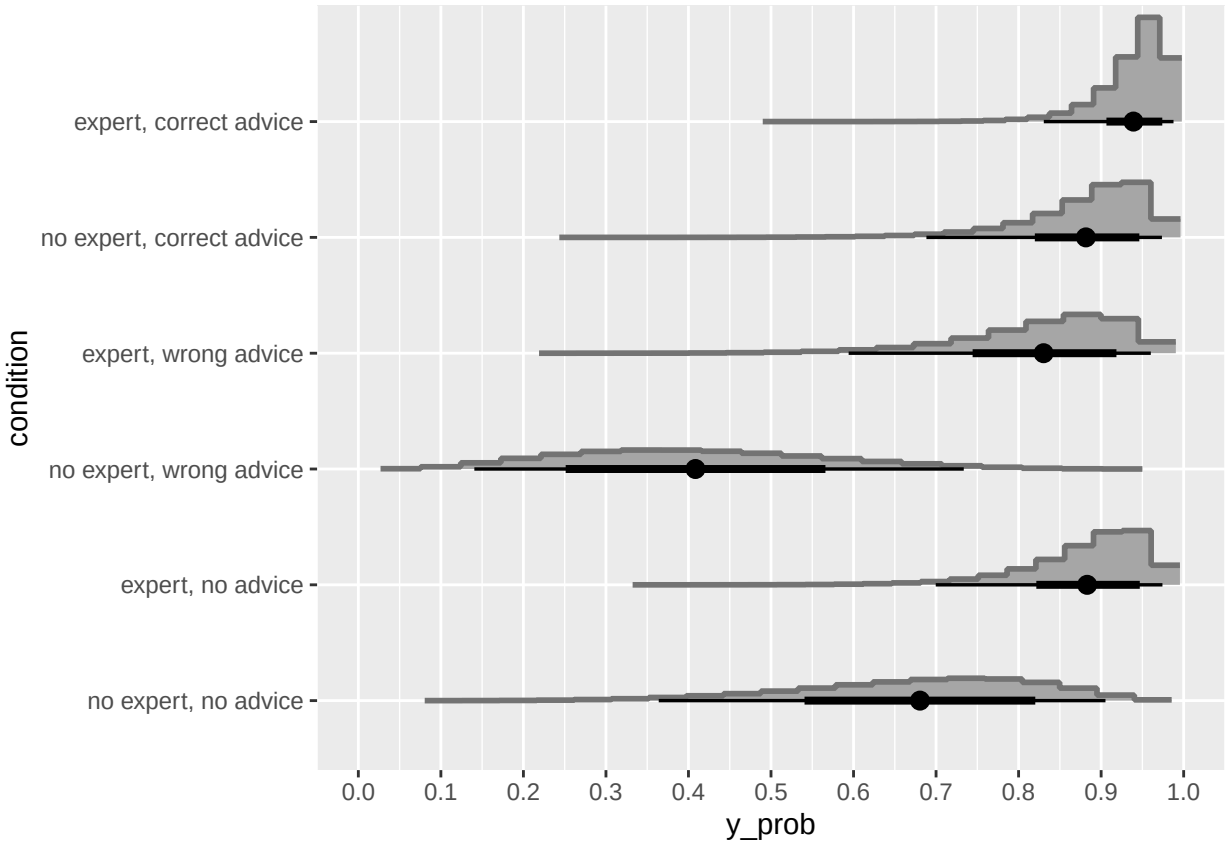
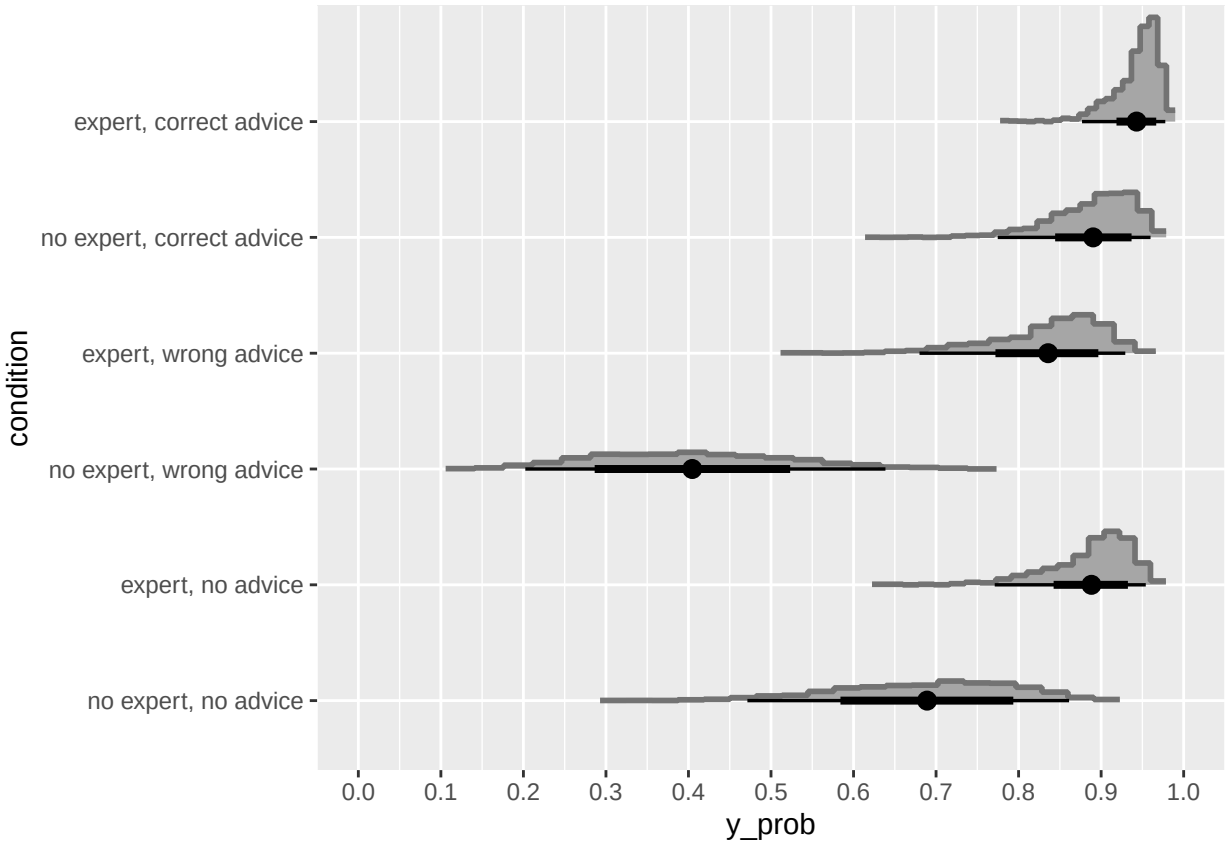


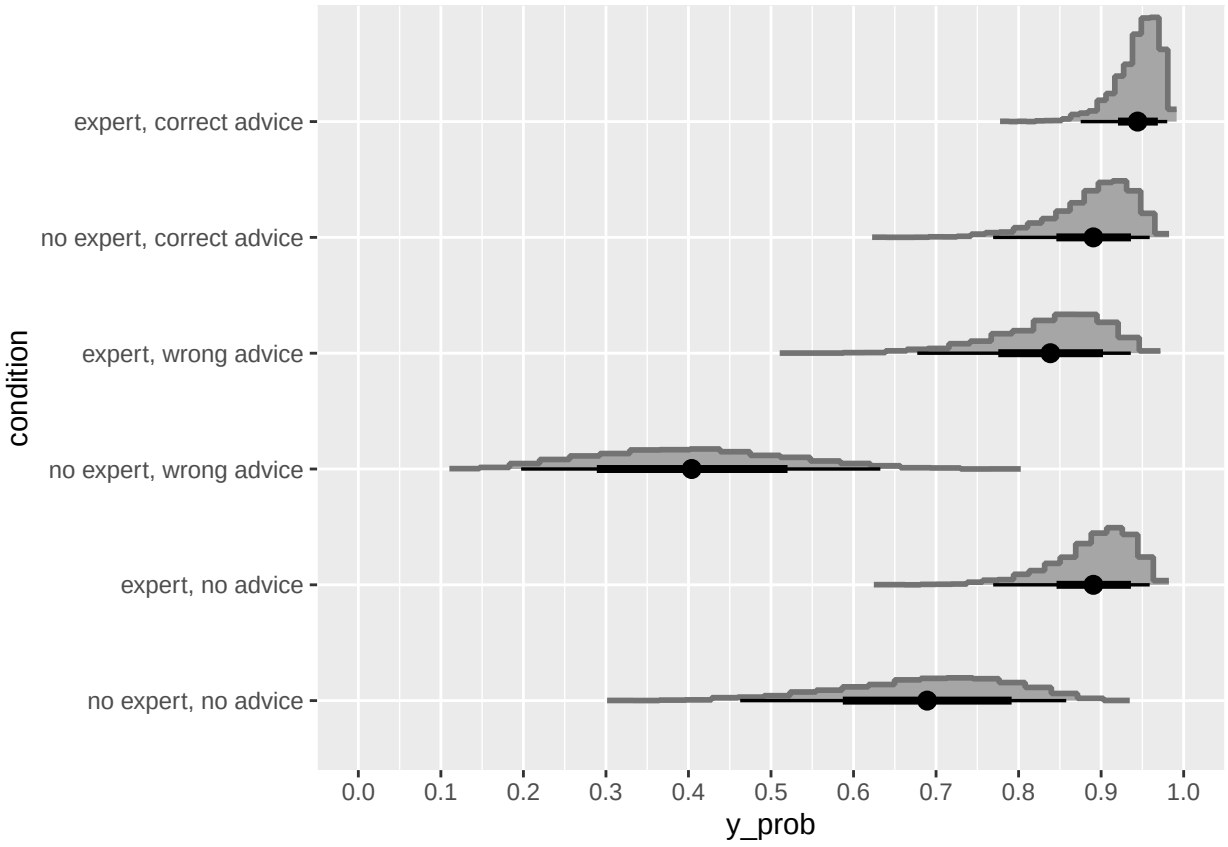
Figure X shows the model implied marginal distributions, including the mean, 66% and 95% intervals. We can see that indeed, the average probabilities (block dots) differ from the probabilities of average subjects and items considered in the previous section. This difference increases with the variability of the random effects.

In fact, up to this point we have not talked about plausible values for the standard deviations of the subject and item random intercepts (σ_S and σ_I). Plots like the one above are a useful tool to decide whether the specified standard deviations are reasonable, by comparing the ranges and overlap between conditions to domain knowledge.

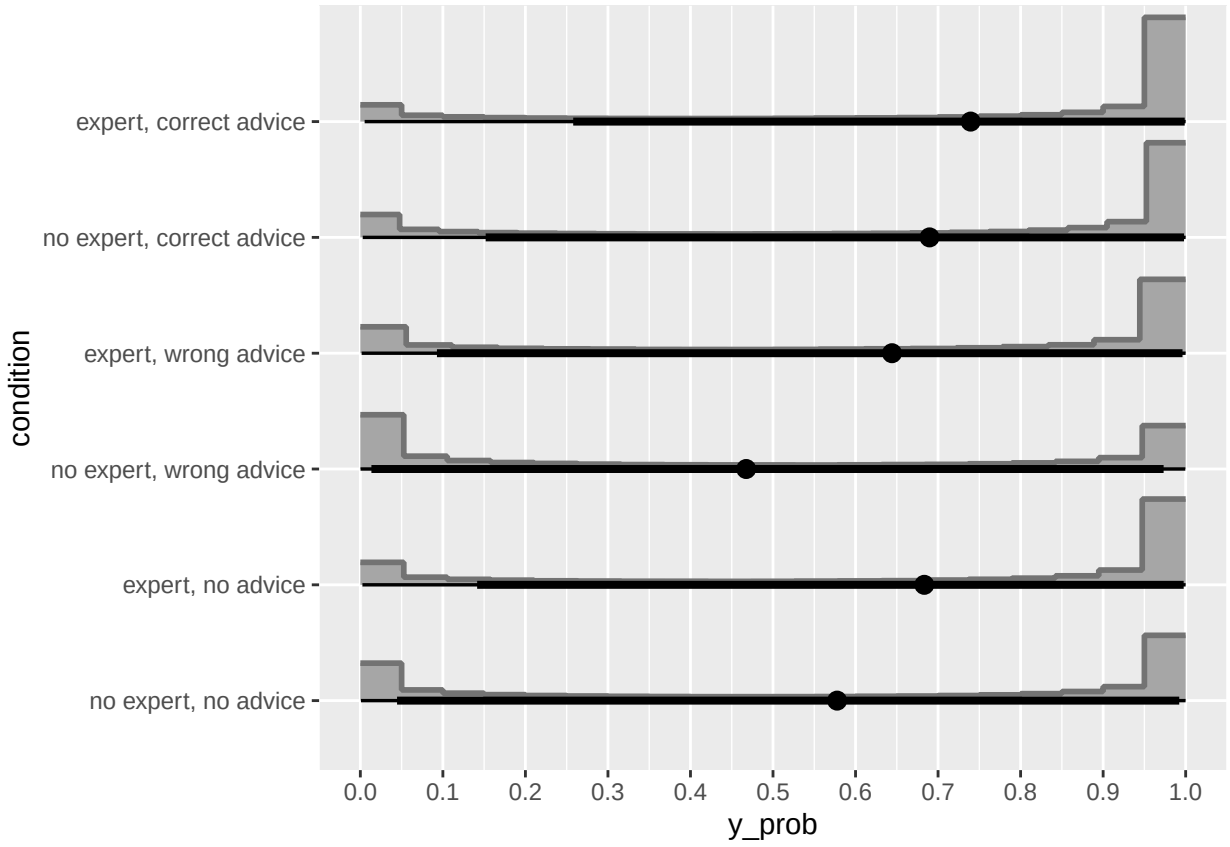
In the next plot, we have set the item standard deviation to almost zero ($\sigma_I = 0.01$). This gives us a better way to see the variability between persons.



In the next plot, we have set the subject standard deviation to almost zero ($\sigma_S = 0.01$. This gives us a better way to see the variability between items.



The final plot demonstrates, that these plots are also useful to spot standard deviations specified to high. For example, if we set $\sigma_S = 3$ and $\sigma_I = 3$ this implies that in each experimental condition, the probabilities that a subject solves an item are always close to either 0 or 1, which is not a plausible assumption.



Results

Power simulation

With all these considerations out of the way, we are finally ready to perform a power simulation. Wrapping the `simulate` function already introduced earlier, the helper function `sim_and_analyse` performs all previous steps (simulate a dataset, fit a GLMM, compute p-values) in a single command.

```
sim_and_analyse <- function(
  formula_chr = "y_bin ~ 1 + expert + advice_present + advice_correct +
    expert:advice_present + expert:advice_correct + (1|subject) + (1|item)",
  null_hypotheses = c("advice_present + advice_correct +
    expert:advice_present + expert:advice_correct <= 0",
    "advice_present + advice_correct <= 0",
```

```
    "-1 * (advice_present + expert:advice_present) <= 0",
    "-1 * (advice_present) <= 0"), ...){
  require(lme4)
  require(multcomp)
  # simulate data
  dat <- simulate(...)
  # fit model
  model <- glmer(as.formula(formula_chr), data = dat, family = "binomial")
  # compute p-values
  glht <- glht(model, linfct = null_hypotheses)
  pvalues <- summary(glht, test = univariate())$test$pvalues
  setNames(pvalues, paste0("p_H0", 1:length(null_hypotheses)))
}
```

Power analysis can quickly become computationally intensive, when we repeatedly simulate data and fit model for different parameter combinations or sample sizes. Here, we use the future and furr packages to perform computations in parallel. First, we enable parallelization and specify how many parallel cores of our computer to use (users can find out the maximum number of cores on their computer with the command `parallel::detectCores()`), and set a seed to make the simulation reproducible.

```
library(future)
plan("multisession", workers = 6)
set.seed(2)
```

The next code chunk, specifies a simulation design with different settings for both the number of subjects (`n_subjects`) and the number of items (`n_items`), each combination being repeated `rep` times.

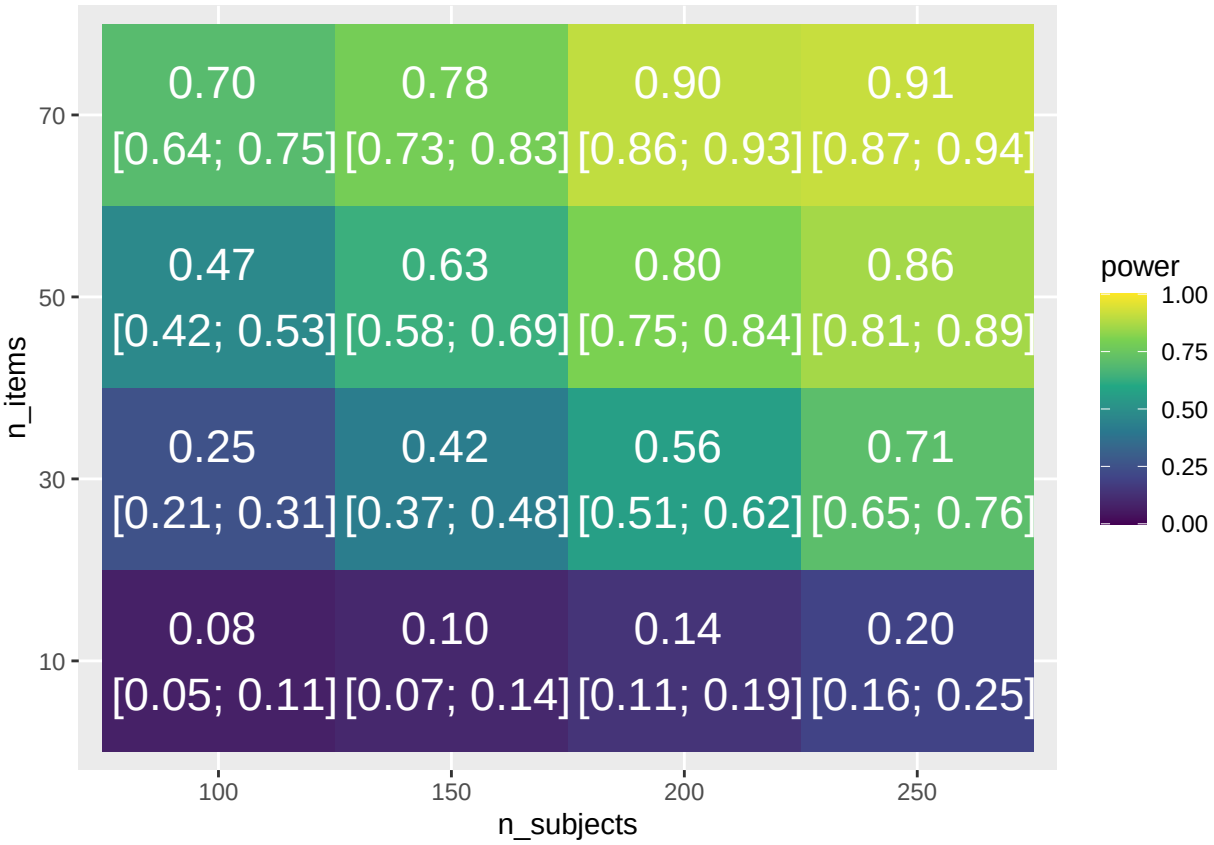
```r
library(furrr)
sim_design <- crossing(
  rep = 1:300,
  n_subjects = c(100, 150, 200, 250),
  n_items = c(10, 30, 50, 70)
) %>%
  mutate(pvalues = future_pmap(., sim_and_analyse,
    .options = furrr_options(seed = TRUE))) %>%
  unnest_wider(pvalues)
```

```
## Warning: There were 142 warnings in `mutate()`.
## The first warning was:
## i In argument: `pvalues = future_pmap(., sim_and_analyse, .options =
##   furrr_options(seed = TRUE))`.
## Caused by warning in `checkConv()`:
## ! Model failed to converge with max|grad| = 0.0158134 (tol = 0.002, component 1)
## i Run `dplyr::last_dplyr_warnings()` to see the 141 remaining warnings.
```

The result of the computation is a data.frame that contains the p-values of all tested hypotheses for each simulated dataset.

For our exemplary combined hypothesis, power is defined as the (long-run) percentage of simulations, in which all four p-values of our component hypotheses are significant at the $\alpha = 0.05$ level. Based on our simulation outcomes, we compute a power estimate for each combination of `n_subjects` $\times$ `n_items` (including 95% confidence intervals) and visualize the results with the following code (heavily inspired by the "Mixed Design Simulation" vignette of the faux package at https://debruine.github.io/faux/articles/sim_mixed.html).

```r
library(binom)
alpha <- 0.05
power <- sim_design %>%
  group_by(n_subjects, n_items) %>%
  summarise(power = mean(p_H01 < alpha & p_H02 < alpha &
                           p_H03 < alpha & p_H04 < alpha),
    n_sig = sum(p_H01 < alpha & p_H02 < alpha &
                  p_H03 < alpha & p_H04 < alpha),
    n = n(),
    ci.lwr = binom.confint(n_sig, n, method = "wilson")$lower,
    ci.upr = binom.confint(n_sig, n, method = "wilson")$upper,
    .groups = "drop")
power %>%
  mutate(across(c(n_subjects, n_items), factor)) %>%
  ggplot(aes(n_subjects, n_items, fill = power)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%.2f \n [%.2f; %.2f]",
                                power, ci.lwr, ci.upr)),
    color = "white", size = 6) +
  scale_fill_viridis_c(limits = c(0, 1))
```



Discussion

Human-AI interaction research requires meticulous planning and consideration of statistical power to ensure reliable and meaningful results. While heuristics and helper programs can be useful for simple designs and models, they often fall short when more complex and customized simulations are required. In this paper, we discuss the necessity of bespoke power simulations in human-AI interaction research, their implications for research design, and the importance of teaching these skills to Human-AI interaction researchers. Furthermore, we highlight the discrepancy between the perceived effort involved in performing bespoke power simulations and the actual effort required. Finally, we emphasize the value of power simulations as research contributions and their alignment with open science and preregistration practices.

1. The Need for Power Simulations in Human-AI Interaction Research: Human-AI

interaction research often involves intricate designs and complex models that cannot be adequately addressed by heuristics or simple helper programs. Bespoke power simulations offer a solution by providing Human-AI interaction researchers with a tailored approach to estimating statistical power. These simulations take into account the specific study design, account for the underlying assumptions, and offer more accurate power estimates.

2. Managing Simulations with Discrete Predictor Variables: Bespoke power simulations become more manageable when all predictor variables are discrete and fixed by the study design. This allows Human-AI interaction researchers to focus on simulating outcome variables while avoiding the need to simulate predictor values, which would introduce additional assumptions. By simplifying the simulation process, researchers can obtain reliable power estimates without compromising accuracy.

3. Teaching Bespoke Power Simulation Skills to Human-AI Interaction Researchers: The ability to conduct bespoke power simulations is a valuable skill that should be taught to Human-AI interaction researchers. By incorporating this training into research methods courses and workshops, researchers can gain a deeper understanding of statistical power and improve the quality of their experimental designs. Equipping Human-AI interaction researchers with the knowledge and tools to perform bespoke power simulations empowers them to make informed decisions and enhance the rigor of their studies.

4. Addressing the Mismatch in Effort Perception: There is often a significant disconnect between the perceived effort required to perform bespoke power simulations and the actual effort estimated by researchers and collaborators in Human-AI interaction research. Many content researchers and collaborators request power simulations from statisticians or methodological experts without fully comprehending the complexity and time-consuming nature of these simulations. It is crucial to raise awareness about

the effort involved in bespoke power simulations to ensure realistic expectations and effective collaboration between researchers and methodological experts.

5. Recognizing the Value of Bespoke Power Simulations: Bespoke power simulations are not mere technicalities; they are valuable research contributions that deserve recognition in Human-AI interaction research. They offer insights into the reliability and sensitivity of experimental designs, helping researchers make informed decisions about sample sizes, effect sizes, and statistical power. The importance of bespoke power simulations can be reflected by allocating them a separate publication or incorporating them as a significant component of stage 1 preregistered reports.

6. Integration with Open Science and Preregistration Practices: Bespoke power simulations align well with the principles of open science and preregistration in Human-AI interaction research. When researchers have access to simulated data based on their prespecified model, analyzing the collected dataset becomes straightforward and unambiguous. By preregistering their power simulations, researchers enhance transparency and accountability in their experimental procedures, contributing to the credibility and reproducibility of Human-AI interaction research.

Conclusion

Bespoke power simulations play a critical role in human-AI interaction research, allowing researchers to tailor power estimation to the unique aspects of their experiments. The skills required to perform these simulations should be taught to Human-AI interaction researchers, fostering a deeper understanding of statistical power and enhancing research design. It is essential to bridge the gap between perceived and actual effort associated with power simulations and recognize their value as research contributions. By integrating bespoke power simulations with open science and preregistration practices, researchers can improve the robustness and transparency of their findings, advancing the field.

References

Aust, F., & Barth, M. (2022). *papaja: Prepare reproducible APA journal articles with R Markdown*. Retrieved from https://github.com/crsh/papaja

Barth, M. (2022). *tinylabels: Lightweight variable labels*. Retrieved from https://cran.r-project.org/package=tinylabels

R Core Team. (2023). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/