

Psychometric models

A brief applied overview

T. Florian Jaeger

December 18, 2020

Contents

1	Goals of this tutorial	1
2	Readings and other materials	2
3	Psychometric models	3
3.1	Perceptual model	3
3.2	Lapsing model (Wiechmann and Hill formulation)	3
3.3	Lapsing model (mixture formulation)	3
4	A simple example	4
4.1	Data	4
4.2	Setting up the model	5
4.3	Preparing the data	6
4.4	Fitting the model	6
4.5	Determining the subject-specific threshold	7
5	Illustrating the consequences of assumptions about the lapse rate (λ)	8
5.1	Simulation study	8
5.1.1	Visualizing the correlation among the intercept and slope	11
5.1.2	Visualizing bias in the estimates	11
5.2	Working with real data (unknown ground truth)	12
5.2.1	Assuming a λ	12
5.2.2	Inferring λ from the data	15
6	Combining the data from all subjects: mixed-effects psychometric models	16
7	Alternative perceptual models	19
7.1	Weibull regression	19
8	Appendix	19
8.1	Inferring γ	19
8.1.1	Mixture formulation of the lapsing mixed-effects logistic model	20
8.1.2	NLM formulation of the lapsing logistic model	25
9	Session info	28

1 Goals of this tutorial

This tutorial aims to illustrate the relation between psychometric models (as used in psychophysics) and generalized linear models (GLMs) / generalized linear models (GLMMs). It's best to read it *after* reading the accompanying GL(M)M tutorial within the same git repository.

Psychometric models aim infer the ‘threshold’ and ‘slope’ of a categorization/recognition function along one or more stimulus dimensions (e.g., visual contrast, intensity, size, etc.). One challenge in doing so is that the exact function is unknown, though plausible candidates—such as the Weibull, Gumbel, logistic, cumulative Gaussian, etc.—exist ([quick side-by-side of these functions](#)). Another challenge is that the estimates of the threshold and slope parameters can be affected (and biased) if attentional lapses are not considered or fixed to specific values. These biases persist even if the true lapse rate is constant across conditions of interest.

This tutorial is very much a work in progress and will benefit from your questions and comments. I am aware that many sections are lacking context and some sections are incomplete. It is also possible that parts of the tutorial still contain errors. So please ask / comment away.

2 Readings and other materials

To learn more about psychometric models, you might also find Gilchrist et al. (2005) and Wichmann and Hill (2001a, b) helpful (both in *Perception & Psychophysics*). Prins (2013, *Journal of Vision*) provides a counterpoint to the solution proposed in Wichmann and Hill (2001). All of these papers are relatively heavy on math but the Prins reading is written in a comparatively accessible style.

Luckily, there is a wealth of information out there, including introductory walk-throughs and powerful animated demonstrations. Here are some walk-through demonstrations (thanks to Martina Poletti for pointing me to these resources):

- <http://www.dlinares.org/lapsesquickpsy.html> (partially replicating Wichmann’s point about biases, but also replicating Prins’s failure to replicate Wichmann’s proposed solution)
- <http://www.palamedestoolbox.org/understandingfitting.html> (beautiful animations and visualizations explaining the fitting process, and the relation between lapse, threshold, and slope; really drives home the point that these three parameters form a joint distribution for which we’re trying to find the set of values that maximize the likelihood of the observed outcomes). Also goes through the Prins and Palamedes papers, and ends with a list of tips.
- <http://www.palamedestoolbox.org/weibullandfriends.html> (Nice explanation of the psychometric-specific jargon and naming of models; e.g., Gumbel vs. Weibull, a matter of log-transforming the predictor)

Some toolboxes/libraries for psychometric models:

- Matlab:
- [psignifit/](#)
- R:
 - [quickpsy](#)
 - [psyphy](#) include GLM-based psychometric model fitting, lapse models, and a variety of psychometric models (incl. cumulative Gaussian)
 - [modelfree](#): non-parametric estimation of psychometric functions.
 - [brms](#) can in principle fit outcomes that follow a Bernoulli, Weibull, or Generalized Extreme Value Distribution. This should cover the logistic, Weibull, Gumbel, and Generalized Extreme Value models for psychometric fitting. It’s not yet clear to me whether [brms](#) would also be able to fit the cumulative Gaussian model. The main appeal of [brms](#) in my view is that it’s part of a more general framework of Bayesian model estimation and thus access to the full posterior distribution of all parameters (e.g., via [tidybayes](#)). [Summary of distributional families that can be specified in brms.](#)

Further readings:

- Abrahamyan et al. (2016) <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4922170/> (on adaptive procedures)
- Jigo & Carrasco (2020) <https://jov.arvojournals.org/article.aspx?articleid=2770148> (on exogenous and endogenous attention and their effects on the psychometric function)

3 Psychometric models

Psychometric models consist of two parts: a perceptual model that describes behavior (or neural responses) as a function of the stimulus and a component that describes what happens on the trials when a subject does not respond based on the stimulus.

3.1 Perceptual model

In psychometric modeling, we predict subjects' responses from a stimulus value. Let's refer to this as the *perceptual model*. A number of different types of perceptual models are common in the field, with choices varying between labs, based on goodness of fit, or field-specific conventions. All of these models fall into the class of non-linear models (NLM) and some are also generalized linear models (GLMs). The goal of this section is to illustrate the relation between GL(M)Ms and psychometric functions a bit further, so that your understanding of GL(M)Ms can also inform your analysis choices for psychometric data.

Four commonly used perceptual models include the cumulative Gaussian, logistic, Weibull, and Gumble (Generalized Extreme Value distribution a.k.a. log Weibull). For formal introductions, the relation between the different functions, and their relative trade-offs, see the readings listed at the top of this tutorial. Of relevance here is that any of these perceptual models can be expanded to account for the proportion of trials on which the subject responds independent of the stimulus (e.g., because the subject experienced an attentional lapse) and optionally the bias (or guessing rate) on those lapsing trials.

3.2 Lapsing model (Wiechmann and Hill formulation)

The lapse model can be specified in two different ways. First, in the Wiechmann and Hill (2001) formulation:

$$p(\text{correct response}|\text{stimulus}, \alpha, \beta, \lambda, \gamma) = \gamma + (1 - \lambda - \gamma) * p(\text{correct response}|\text{stimulus}, \alpha, \beta) \quad (1)$$

where $p(\text{correct response}|\text{stimulus}, \alpha, \beta)$ is the **perceptual model**—i.e., the function that maps input stimuli onto responses (e.g., the logistic model). This perceptual model depends on two parameters α and β (under the general modeling framework used in this tutorial, we can include further effects, including non-linear effects of the stimulus). The specific interpretation of these two parameters depends on the psychometric function, but each of the functions can be parameterized such that α is some targeted threshold performance, and β is the 'slope' at that threshold. In addition to the two parameters of the perceptual model, the formula contains two more parameters: γ , which is sometimes called the *{guess} rate*, and λ , which is sometimes called the *{lapse} rate*. Guess and lapse are seemingly intuitive but ambiguous terms, which can cause confusion. In the Wiechmann and Hill formulation, γ describes the performance that would be achieved if the participants guessed on all trials (i.e., chance performance), rather than the proportion of trials on which the participant guesses or the bias in the participant's responses *when* the participant guesses. Under this formulation, γ **thus describes floor performance: the minimum proportion of correct answers**. The second parameter in the Wiechman and Hill formulation, λ , describes the proportion of trials on which participants provide the wrong answer, rather than the proportion of trials on which the participant does not respond based on the stimulus (e.g., because of attentional lapses). Under the Wiechman and Hill formulation, $1 - \lambda$ **thus describes the ceiling performance: the maximum proportion of correct answer**. This is the formulation that I use for most of the analyses below.

3.3 Lapsing model (mixture formulation)

An alternative formulation of the same process, describes the lapse model as a mixture model of two components. This formulation is computationally more efficient and easier to fit, as it orthogonalizes the two parameters that jointly determine the lapsing model (γ and λ in the Wiechmann and Hill formulation). But as far as I can see, the mixture formulation does not allow holding γ or λ constant. That is, the mixture formulation is efficient for scenarios in which we want to estimate both γ or λ (see example in appendix).

Like in the Wiechmann and Hill formulation, one component of the mixture formulation is the perceptual model describing how responses depend on the stimulus for those trials on which participant does respond based on the stimulus. The second component of the mixture model describes responses for trials on which the participant does not respond based on the stimulus. On those trials, the participant is assumed to respond with some *bias* that is independent of (uninformed by) the stimulus. Finally, the respective weights of the two components is determined by the proportion of trials on which the participant does not respond based on the stimulus (*lapse*). Note that *lapse* here is not the same as λ under the Wiechmann and Hill formulation (neither is *bias* the same as γ/guess).

$$p(\text{response}|\text{stimulus}, \alpha, \beta, \text{lapse}, \text{bias}) = (1 - \text{lapse}) * p(\text{response}|\text{stimulus}, \alpha, \beta) + \text{lapse} * \text{bias} \quad (2)$$

Under this second formulation, the floor performance is $\text{lapse} * \text{bias}$ and the ceiling performance is $1 - (\text{lapse} - \text{lapse} * \text{bias}) = 1 - (1 - \text{bias}) * \text{lapse}$. While the *lapse* and *bias* parameters of this formulation are not the same as the λ and γ parameters of the Wiechmann and Hill formulation, the parameters can be translated into each other:

$$\text{lapse} = \gamma + \lambda \quad (3)$$

$$\text{bias} = \frac{\gamma}{(\gamma + \lambda)} \quad (4)$$

$$\lambda = (1 - \text{bias}) * \text{lapse} \quad (5)$$

$$\gamma = \text{lapse} * \text{bias} \quad (6)$$

For a demonstration of the second formulation, you can visit the interactive website at <https://hlplab.shinyapps.io/BayesianIdentificationAndDiscrimination/>. The site's first tab describes category identification (categorization) and perceptual discrimination for two Gaussian categories by an ideal observer. We'll focus on the "Ideal identification" plot (right side, in the middle of the lower half of the screen) and how it changes if you change as a function of the rate of attentional lapsing (top left) and bias (unclick "Prior as bias" and then try out different biases). The ideal categorization function between two Gaussian categories turns out to be a logistic function (with linear and quadratic stimulus effects)—i.e., one of the psychometric functions mentioned above (and a GLM). So the effects of lapse rate and bias that you see on this site give you an initial idea of how these parameters can affect the fit of psychometric functions.

The two formulations describe the same outcomes but split up the relevant quantities in different ways. Regardless of the formulation, the lapse model is *not* a GLM anymore.

4 A simple example

This tutorial uses the R library **brms** for Bayesian regression modeling, which—among many other things—fits non-linear models. **brms** converts GL(M)Ms, GA(M)Ms, NL(M)Ms, and a number of other model types into **stan** code. **stan** is a model specification language with interfaces to Matlab, R, Python and other languages. It comes with an efficient sampler. **stan** programs are compiled into C++ code, sampled from, and the resulting posterior samples of the fitted model are returned ... in this case to R's **brms** functions. Using a Bayesian approach has a number of upsides, one of which is that the straightforwardness of obtaining (Bayesian) confidence intervals for any of the parameters in the model.

4.1 Data

We use the same data as in the GLMM tutorial from Ashley's experiment. Please consult the GLMM tutorial for a detailed description of the experiment. As a reminder, here is an overview of the data:

```
tibble [3,949 x 18] (S3: tbl_df/tbl/data.frame)
 $ Subject      : Factor w/ 10 levels "1","2","3","4",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ Condition    : Factor w/ 2 levels "uncrowded","crowded": 1 1 1 1 1 1 1 1 1 1 ...
 $ Threshold.Subj : num [1:3949] 1.39 1.39 1.39 1.39 1.39 1.39 ...
```

```

$ DiffusionConstant.Subj      : num [1:3949] 11.1 11.1 11.1 11.1 11.1 ...
$ Area.Subj                  : num [1:3949] 137 137 137 137 137 ...
$ Span.Subj                  : num [1:3949] 3.13 3.13 3.13 3.13 3.13 ...
$ Speed.Subj                  : num [1:3949] 40.6 40.6 40.6 40.6 40.6 ...
$ Curvature.Subj              : num [1:3949] 12 12 12 12 12 ...
$ Size                        : num [1:3949] 0.505 0.505 0.505 0.505 0.505 ...
$ Size.AvgPerformance         : num [1:3949] 0.135 0.135 0.135 0.135 0.135 ...
$ Response                    : num [1:3949] 4 1 4 3 2 3 4 3 3 1 ...
$ ResponseExpected            : num [1:3949] 4 4 2 2 4 1 1 2 3 2 ...
$ ResponseCorrect             : num [1:3949] 1 0 0 0 0 0 0 0 1 0 ...
$ ResponseTime                : num [1:3949] 1537 2203 2266 2385 2029 ...
$ Curvature                   : num [1:3949] 16.1 14.6 14.1 14.1 11.3 ...
$ Speed                       : num [1:3949] 33.6 38.5 43.5 39.8 42.7 ...
$ Span                        : num [1:3949] 3.03 2.25 2.9 2.13 3.65 ...
$ individualDiffusionConstant: num [1:3949] 7.75 7.63 7.44 3.88 12.94 ...
- attr(*, "na.action")= 'omit' Named int [1:2590] 1 6 9 12 15 20 24 26 27 33 ...
..- attr(*, "names")= chr [1:2590] "1" "6" "9" "12" ...

```

4.2 Setting up the model

As a first illustration, here is the formula describing a psychometric model with a logistic link function for the case of non-hierarchical data (data from a single subject). `brms` also allows us a more elegant mixture formulation (see appendix and replies to <https://discourse.mc-stan.org/t/fitting-lapsing-psychometric-functions-with-brms/5762/2>) but the non-linear model implementation following Wiechmann and Hill (2001) is more transparent for the present purpose:

```

# Defining a brms formula for the non-linear model. It has three components:
# the lapse rate, the guess (bias), and the linear predictor eta. Each of these
# components can be described by a linear predictor. For this psychometric
# model we describing eta as the linear combination of the intercept and the
# effect of the stimulus. The other two parameters are just described by only
# their respective 'intercepts'.
BF <- brmsformula(
  ResponseCorrect ~ gamma + (1-gamma-lambda) * inv_logit(eta),
  eta ~ 1 + Size,
  gamma ~ 1,
  lambda ~ 1,
  family = bernoulli(link="identity"),
  nl = TRUE
)

```

Typically, the design of psychometric experiments is set up such that chance level accuracy (γ) is known. In that case, the model simplifies somewhat. For example, for the 4AFC task employed in Ashley’s data, we would get the following model formula:

```

BF.lapse <- brmsformula(
  ResponseCorrect ~ .25 + (1-.25-lambda) * inv_logit(eta),
  eta ~ 1 + Size,
  lambda ~ 1,
  family = bernoulli(link="identity"),
  nl = TRUE
)

```

For non-linear models, it can be important to incorporate constraints on the parameters, or any knowledge we have about plausible values for the parameters. Since we are taking a Bayesian approach here, this is achieved through the definition of priors. Throughout this tutorial, we only use very weakly regularizing priors—i.e., priors that do not bias the coefficient values in either direction but that ‘pull’ them closer to zero (following recommendations from <https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations>). In effect, these priors implement Occam’s razor: unless there is sufficient evidence in the data itself, we assume that the null hypothesis is true for all of our parameters. We can also define lower and upper bounds (lb and ub in the code) for, e.g., the λ parameter. For example, for a model without any constraints on λ :

```

# Define weakly regularizing prior, including lower (lb) and upper bounds (ub)
# for lambda
my.priors.lapse <- c(
  prior(student_t(3, 0, 2.5), class = "b", nlpar = "eta"),

```

```
prior(beta(1, 1), nlpar = "lapse", lb = 0, ub = .5)
)
```

4.3 Preparing the data

So that the priors are on the right ‘scale’, we follow (Gelman et al., 2008), and standardize the continuous predictors in the model. As is typical for analyses of experimental data with two balanced conditions, we deviation/sum/abova/effect-code the condition variable, with the ‘treatment’ receiving the positive predictor value:

```
# We store the mean and sd of Size so that we can later transform the model's predictions back
# onto the scale of original Size predictor
Size.mu = mean(d$Size)
Size.sd = sd(d$Size)

# Standardize Size and make sure order of levels for Condition is as intended
d %<>%
  mutate(
    Condition = factor(Condition, levels = c("uncrowded", "crowded")),
    Size = (Size - mean(Size)) / (2 * sd(Size))

# Sum-code condition
contrasts(d$Condition) = cbind("Crowded.vs.Uncrowded" = c(-.5,.5))
```

4.4 Fitting the model

We fit the model with the function `brm()` from the library `brms`, and summarize the output:

```
Family: bernoulli
Links: mu = identity
Formula: ResponseCorrect ~ 0.25 + (0.75 - lapse) * inv_logit(eta)
        eta ~ 1 + Size
        lapse ~ 1
Data: d %>% filter(Subject == "1") (Number of observations: 396)
Samples: 4 chains, each with iter = 8000; warmup = 4000; thin = 5;
         total post-warmup samples = 3200

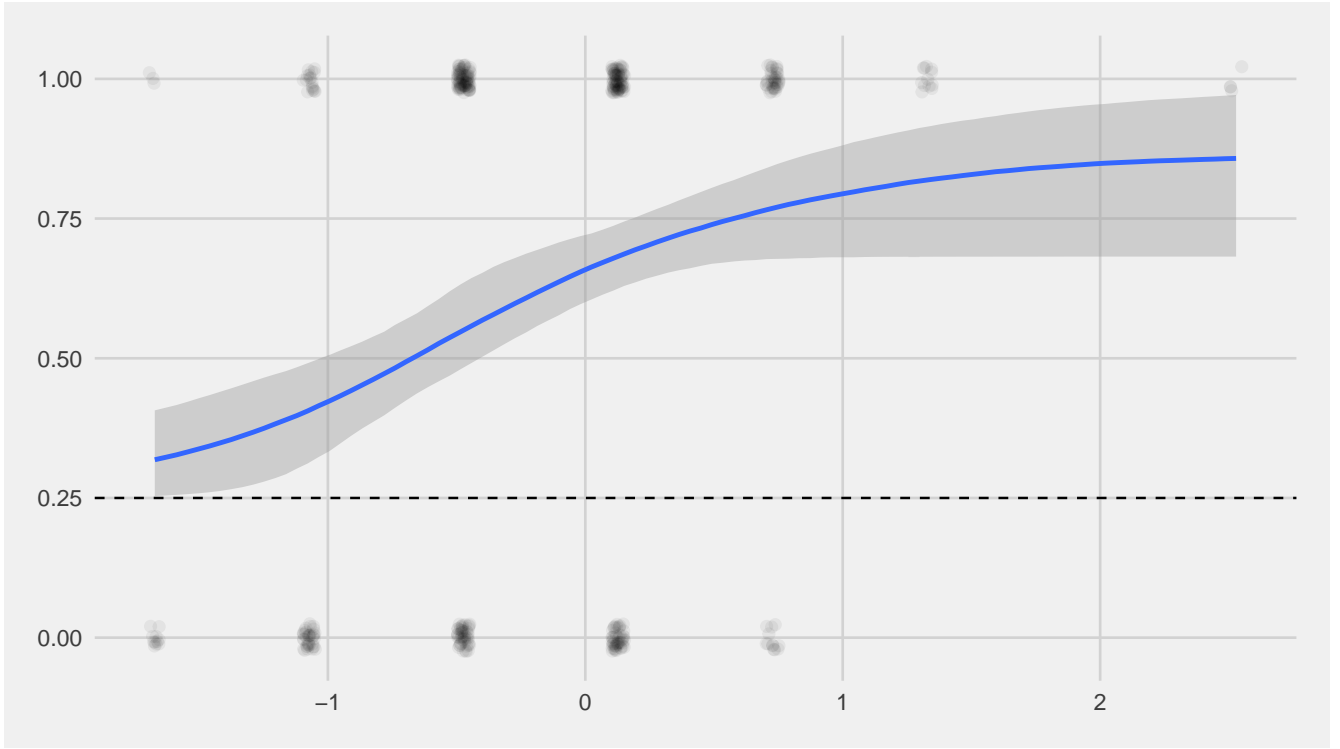
Population-Level Effects:
              Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
eta_Intercept      1.04      1.14    0.00    4.02 1.00    1883    1579
eta_Size            2.00      1.18    0.93    5.31 1.00    2026    1549
lapse_Intercept     0.14      0.09    0.01    0.32 1.00    2029    2006
```

Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS and Tail_ESS are effective sample size measures, and Rhat is the potential scale reduction factor on split chains (at convergence, Rhat = 1).

We can test the hypotheses we hold about the various parameters in the model using Bayesian hypothesis testing, implemented via the function `hypothesis` from the same package:

Hypothesis	Estimate	Est.Error	CI.Lower	CI.Upper	Evid.Ratio	Post.Prob	Star
$\beta_{Size} > 0$	1.996	1.179	1.020	4.246	Inf	1	*
$\lambda > 0$	0.143	0.093	0.012	0.301	Inf	1	*

And, finally, we visualize the fit against the data, the predictions of the model along with 95% uncertainty intervals.



As this illustrates, fitting a lapsing model (likely adequately) leads to substantial uncertainty about λ , α , and β . This is not a rare scenario: for psychometric data with only a few measurement points along the mid-performance range of the stimulus, we can often not distinguish between effects of attentional lapses and the effect of the stimulus. This is worth noting since higher lapse rates in a model like this one will result in steeper (larger) slope estimates, and will also affect our threshold estimate. We illustrate this in the next section, right after we have shown how to obtain the threshold estimate from a model like the one we have just fit.

4.5 Determining the subject-specific threshold

Psychometric models are often used to obtain thresholds and slopes that are then submitted to secondary analysis. For example, we might obtain a stimulus threshold—i.e., a stimulus value at which a certain threshold performance (accuracy) is achieved—and then analyze whether that stimulus threshold differs across conditions. From a model like that fit in the previous section, we can obtain both naive and lapse-corrected performance thresholds. For lapse-corrected threshold, we need to solve the logistic part of the model for the desired threshold. Note that this is an approximate estimate. To get a better estimate, we should marginalize over all posterior samples. This will not necessarily be the same as using the (mean) estimates for both the intercept and slope since the posterior distributions of the intercept and slope might be correlated.

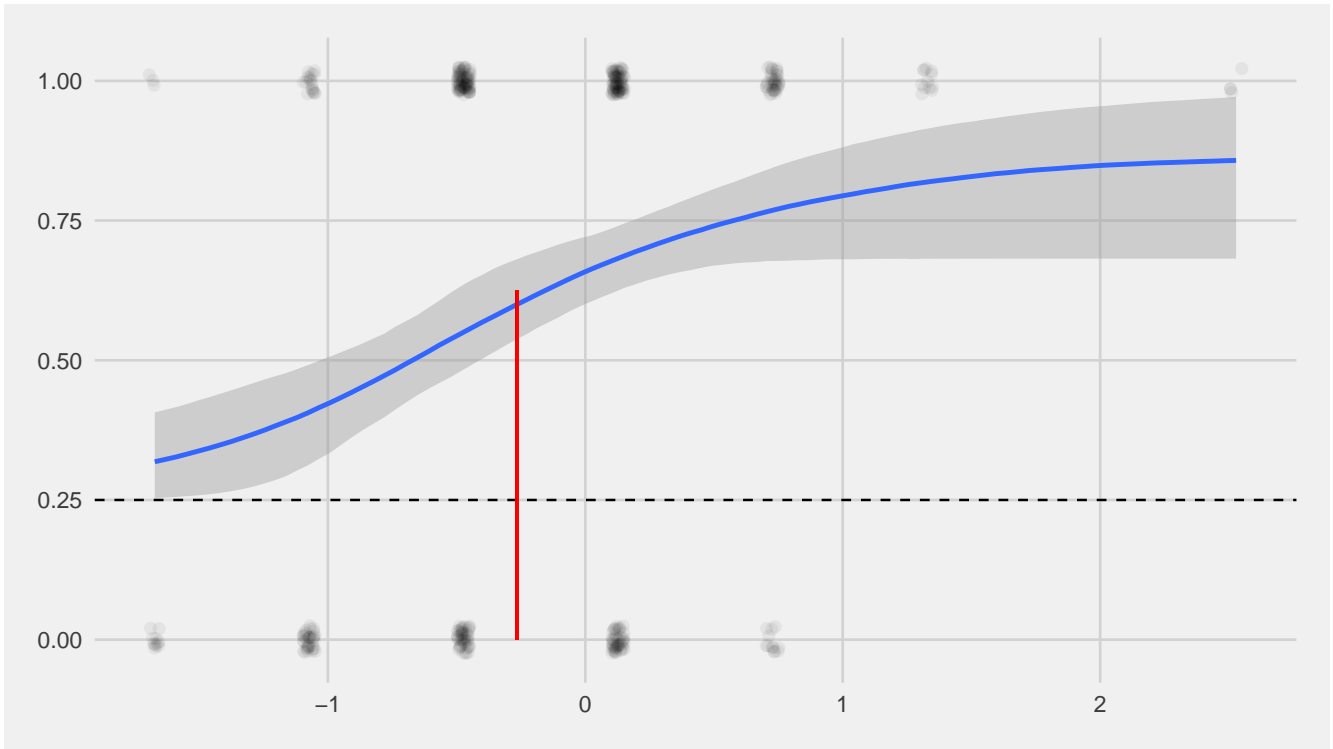
$$\text{logit}(\text{threshold}) = \alpha + \beta * \text{Size} \quad (7)$$

For example, for a threshold of .625, we set:

$$\text{logit}(.625) = \alpha + \beta * \text{Size} \Rightarrow \quad (8)$$

$$0.5108256 = 1.04 + 2.00 * \text{Size} \Leftrightarrow \quad (9)$$

$$-0.2645872 = \text{Size} \quad (10)$$



Similarly, we can obtain the lapse-corrected slope from the model. It is simply β . Being able to obtain thresholds or slopes is useful given the current standards of the field. Note though that off-the-shelf analyses techniques like GLMMs and extensions thereof make allow for a more direct approach of testing the effect of other factors (such as conditions) by simply including those predictors in the psychometric model. This more direct approach has the advantage that it does not throw away trial-level data (as the standard secondary analyses do not have access to the uncertainty of the threshold and slope estimates obtained from the psychometric model), allowing the combined model to more appropriately account for sources of uncertainty. The analyses presented in the remaining sections contain examples of this approach.

The advantage of this more direct approach becomes even further pronounced when we combined the data from all subjects and submit it to trial-level mixed-effects (hierarchical/multilevel) analyses. These analyses allow us to test whether the effects we observe in the trial-level data (e.g., about the effect of condition on aspects of the perceptual model) generalize across the population of interest.

5 Illustrating the consequences of assumptions about the lapse rate (λ)

This section first present a simulation study (for which the ground truth is known) and then analyses based on Ashley Clark's data (for which the ground truth is unknown). Both case studies are intended to illustrate the consequences of assumptions about lapse rates—i.e., what happens when lapse rates are assumed to have a certain value, rather than being estimated.

5.1 Simulation study

For our simulation study, we continue to consider a simple model as in the previous section, with only the intercept and an effect of the stimulus (rather than considering additional conditions). We first make a data generator for a 2AFC task that creates data following the Wichman and Hill (2001) model with a logistic response function:

```
gelman_scale = function(x) {
  x = (x - mean(x)) / (2 * sd(x))
}
```



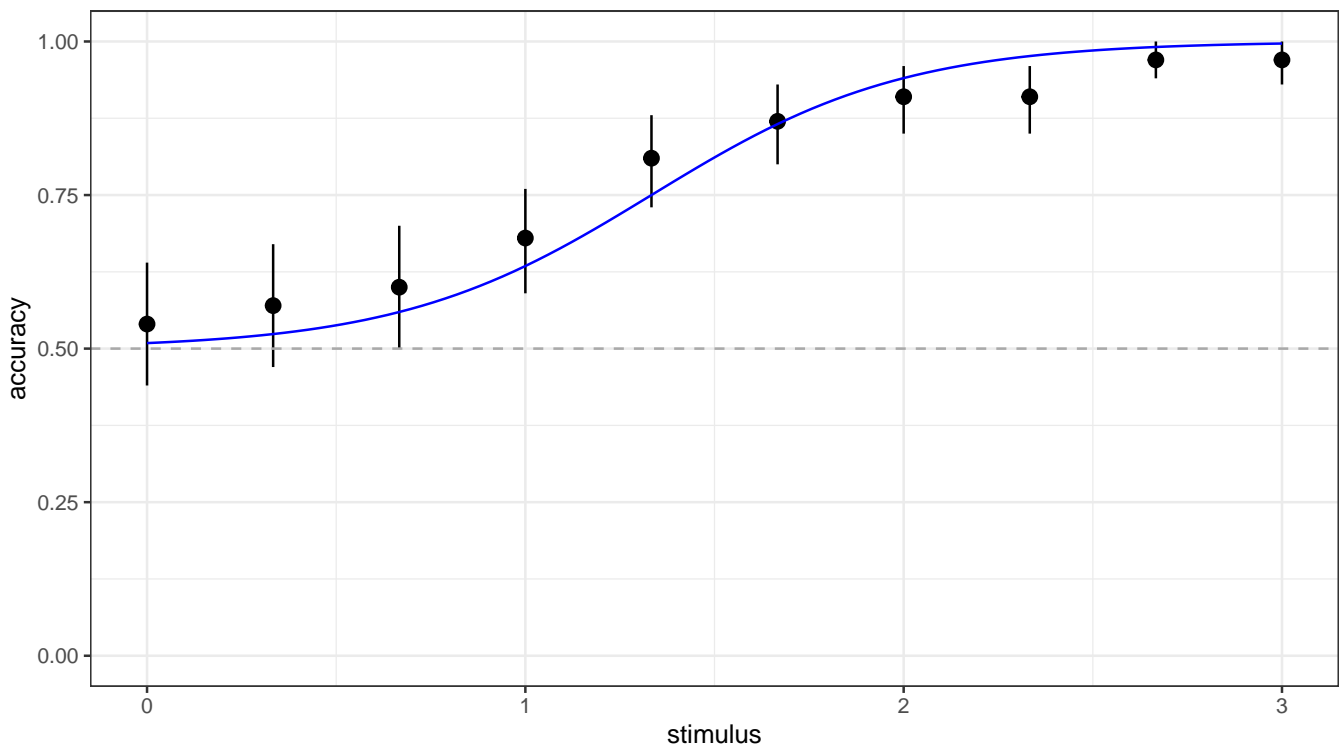
```

generate_data = function(
  intercept = 1,
  slope = .5,
  lambda = 0,
  gamma = .5,
  stimuli = seq(-3, 3, length.out = 20),
  n_per_stimulus = 20
) {
  crossing(
    stimulus = stimuli,
    trial = 1:n_per_stimulus) %>%
    mutate(
      s_stimulus = gelman_scale(stimulus),
      p = inv_logit_scaled(
        intercept + slope * stimulus,
        lb = gamma,
        ub = 1 - lambda),
      accuracy = rbinom(nrow(.), 1, p))
}

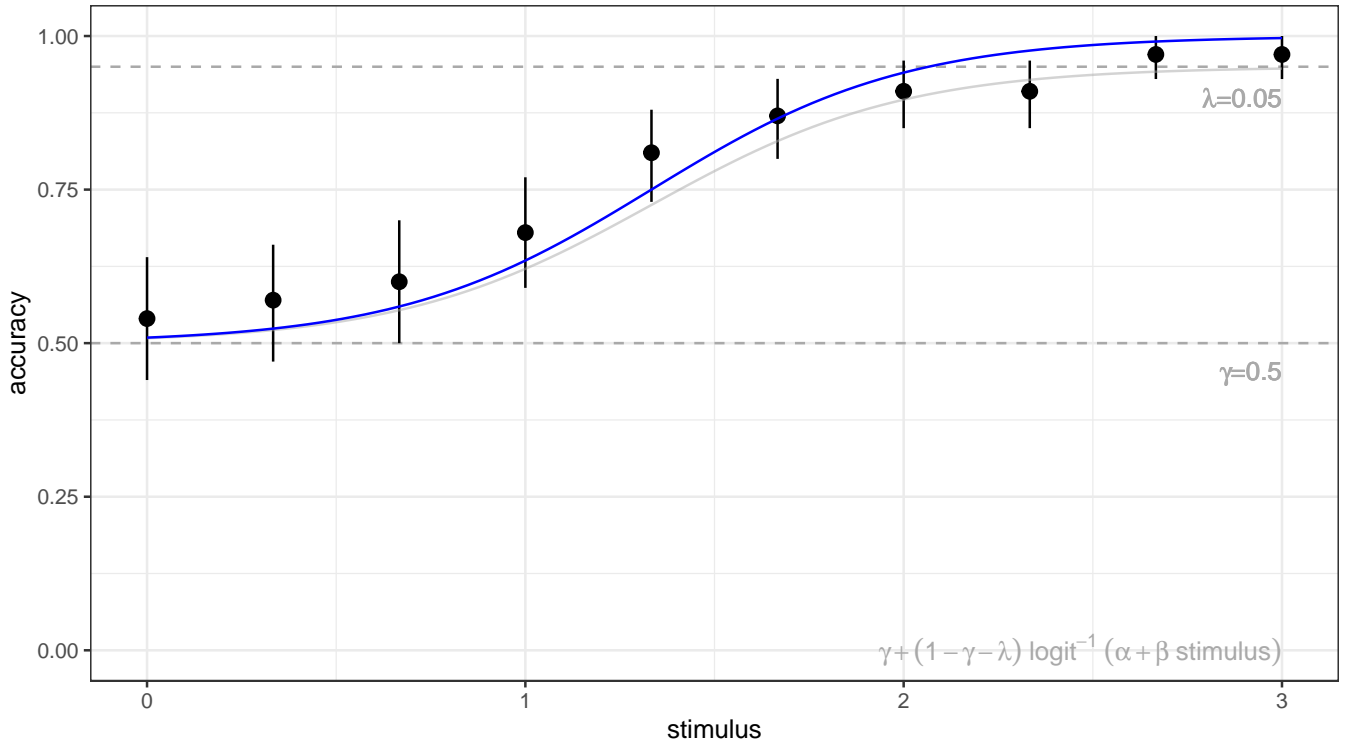
# generate data
true.intercept = -4
true.slope = 3
true.gamma = .5
true.lambda = .05
d.temp = generate_data(
  intercept = true.intercept,
  slope = true.slope,
  gamma = true.gamma,
  lambda = true.lambda,
  stimuli = seq(0, 3, length.out = 10),
  n_per_stimulus = 100)

```

Here is an example draw from the data generator with 100 data points per stimulus at five different stimulus locations between 0 and 3 for $\lambda = .05$. The plot shows a fit if the lapse rate is assumed to be zero:

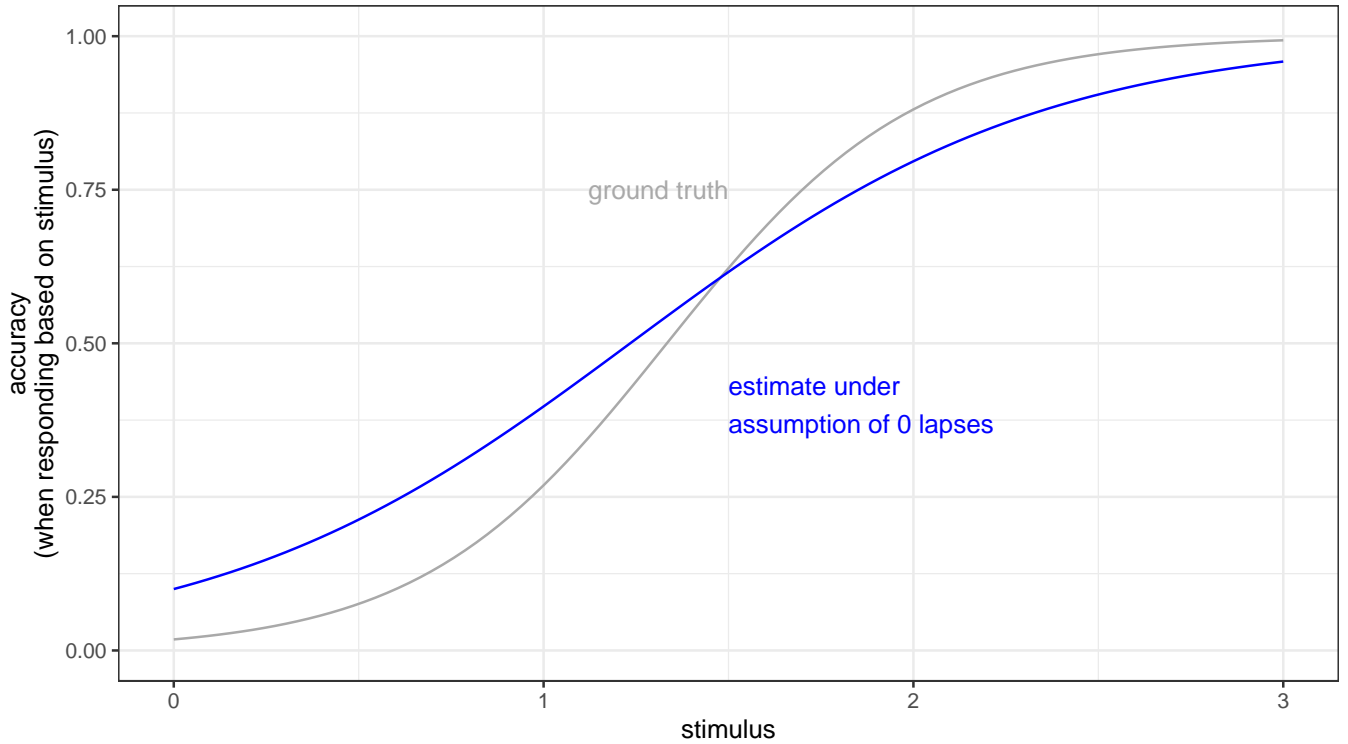


Next, we add the ground truth functional relation, including the lapse rate:



And, finally, we can compare the *perceptual model*—i.e., the functional relation (intercept and slope) between the stimulus and the responses when the participant responds based on the stimulus—when the λ is assumed to be 0 (blue) compared to the ground truth of $\lambda = .05$ (gray).

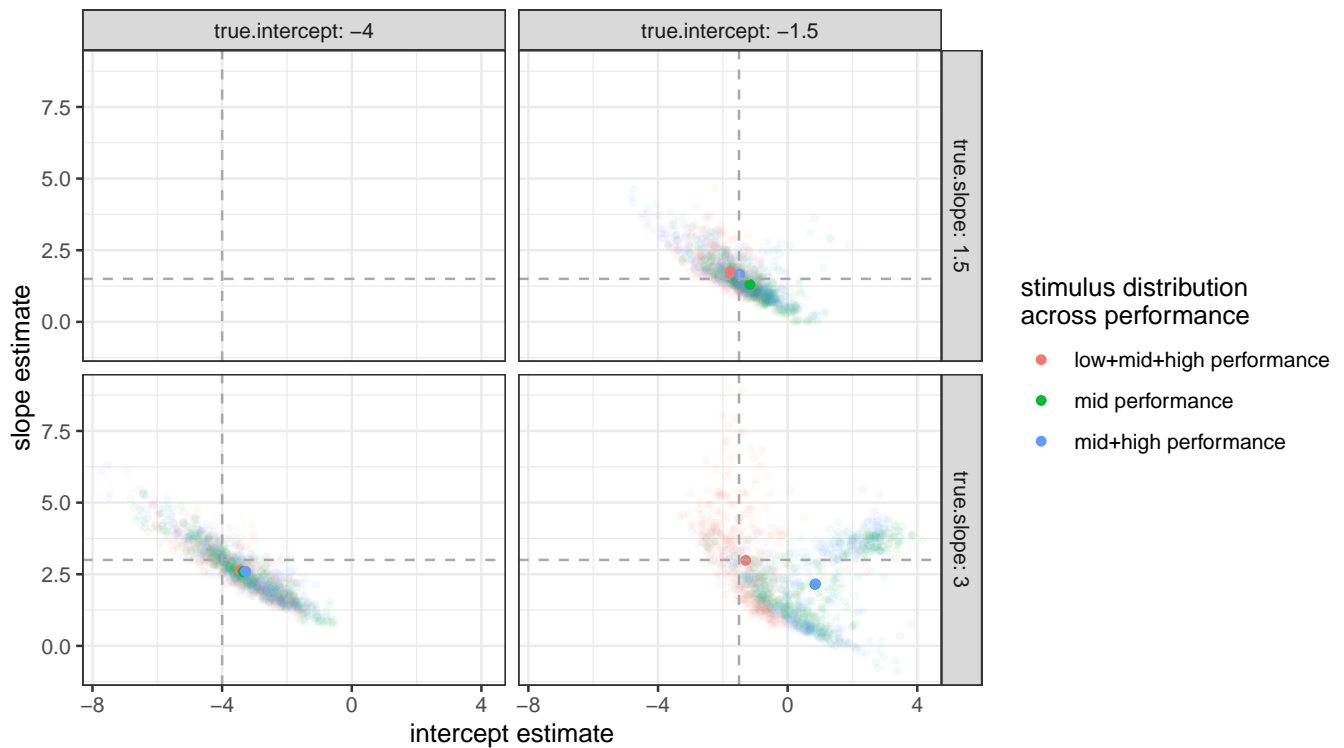
Start sampling



For the simulation, we obtain 100 simulated data sets for each of a variety of combinations of values for the true intercept, slope, and lambda (gamma is always .5 since we're simulated a 2AFC task). We will consider four different analyses, each assuming a different fixed lapse rate (0, .025, .05, .1).

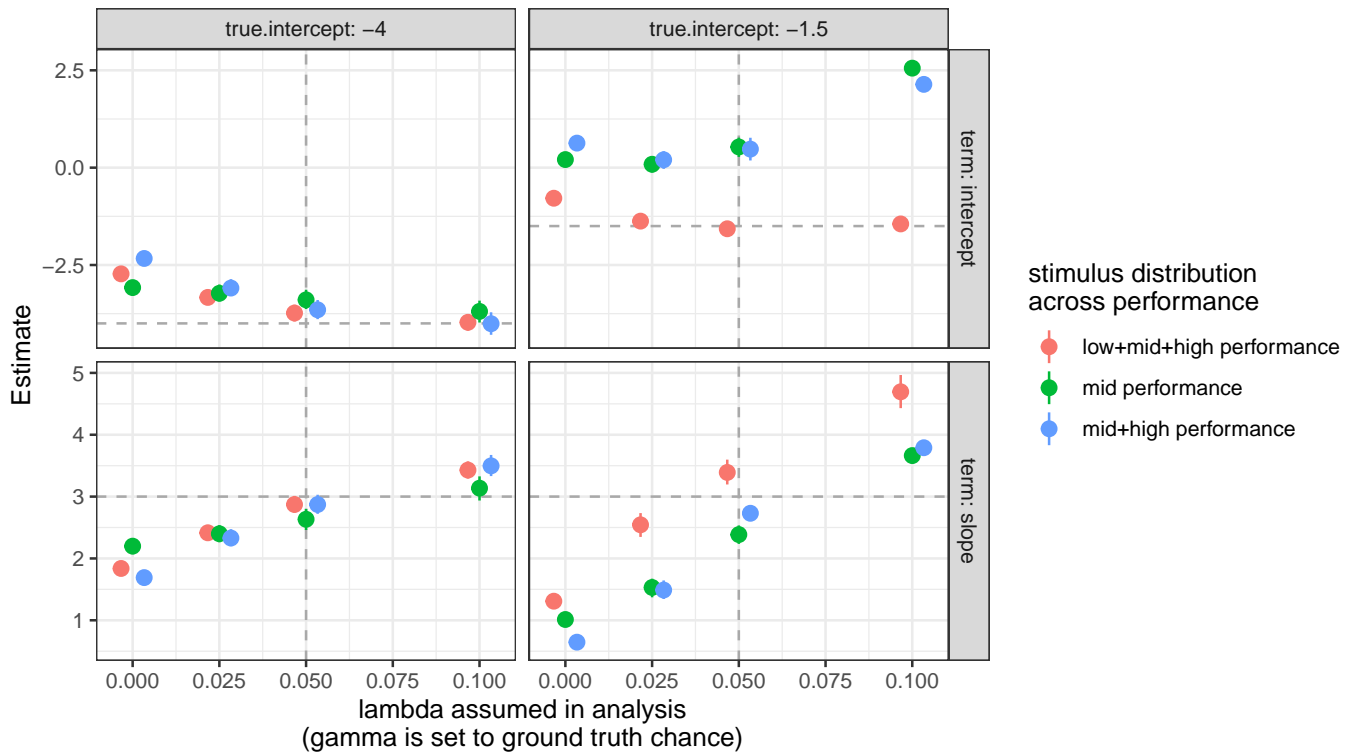
5.1.1 Visualizing the correlation among the intercept and slope

The following plots illustrate the correlation among the intercept and slope in the different simulation conditions.



5.1.2 Visualizing bias in the estimates

In the following plots, the gray dashed lines indicate the ground truth. We are summarizing the distribution of estimates for the intercept and slope obtained for the different types of simulation runs (each with 100 simulation samples).



5.2 Working with real data (unknown ground truth)

Next, we illustrate the consequences of assumption about λ for data from an actual experiment. We again use Ashley's data, this time focusing on Subject 3 from Ashley. Unlike in the simple example shown above, we now consider the effect of an experimental condition (crowdedness). For a description of this condition, see the GLMM tutorial. Here, we test whether the crowdedness of the display affected performance and whether this effect interacted with the effect of stimulus size. We do so while comparing psychometric models that differ in their assumptions about λ .

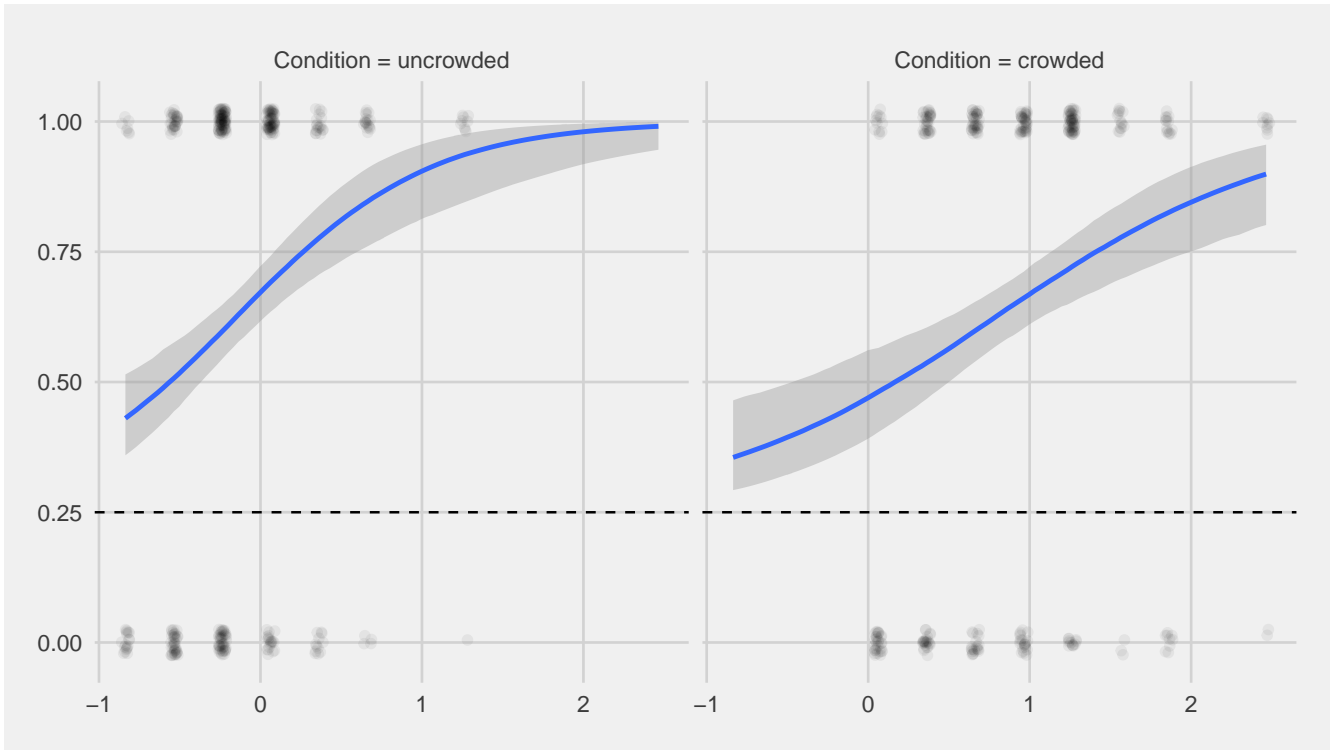
For all models, we assume that $\gamma = .25$ (chance level accuracy in the 4AFC task). Like in the simulation study, all models considered in this section assume a logistic perceptual model. The models do, however, differ in their assumptions about λ .

5.2.1 Assuming a λ

The first model assumes $\lambda = 0$ (no lapses). This model seems to present some evidence for an interaction between the crowdedness condition and the effect of stimulus size:

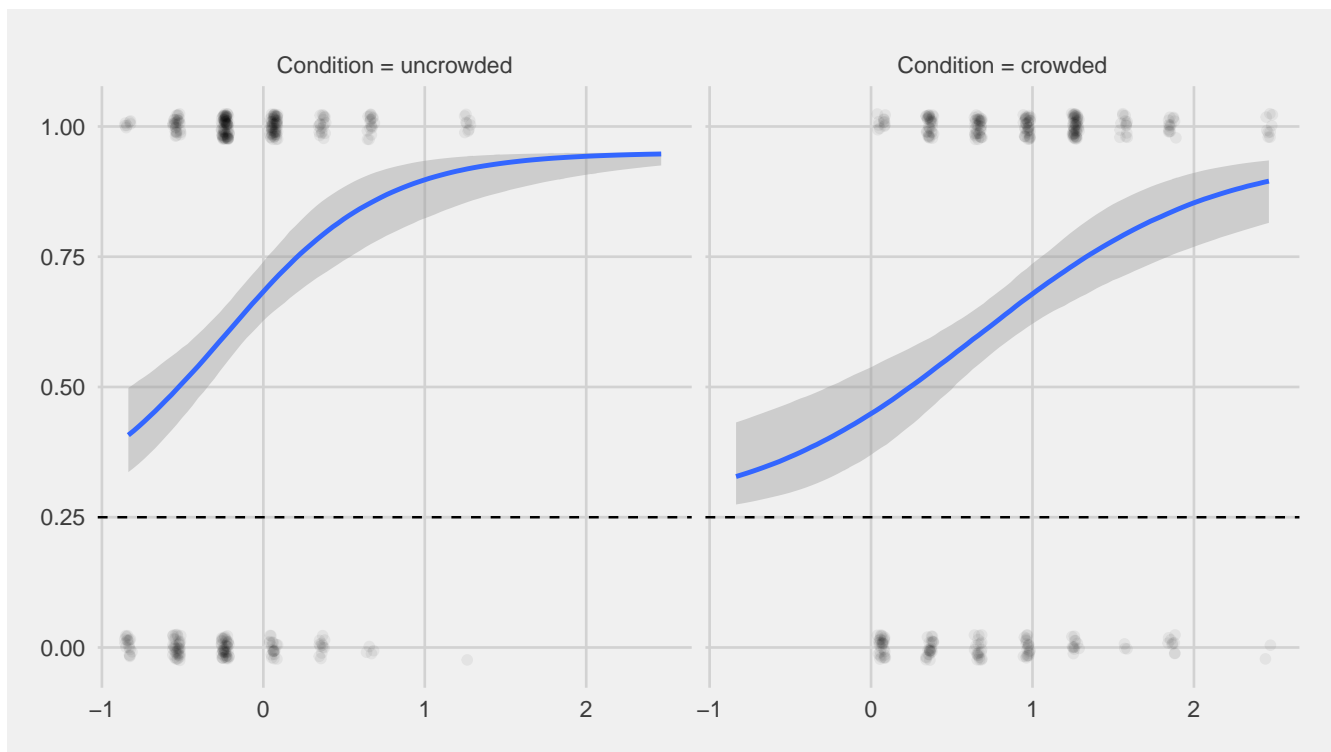
```
BF <- brmsformula(
  ResponseCorrect ~ .25 + (1-.25-0) * inv_logit(eta),
  eta ~ 1 + Size * Condition,
  family = bernoulli(link="identity"),
  nl = TRUE
)
```

Hypothesis	Estimate	Est.Error	CI.Lower	CI.Upper	Evid.Ratio	Post.Prob	Star
$\beta_{Size} > 0$	1.401	0.229	1.014	1.780	Inf	1.000	*
$\beta_{Crowded} < 0$	-1.137	0.305	-1.645	-0.638	Inf	1.000	*
$\beta_{Size:Crowded} < 0$	-0.565	0.446	-1.284	0.144	8.357	0.893	

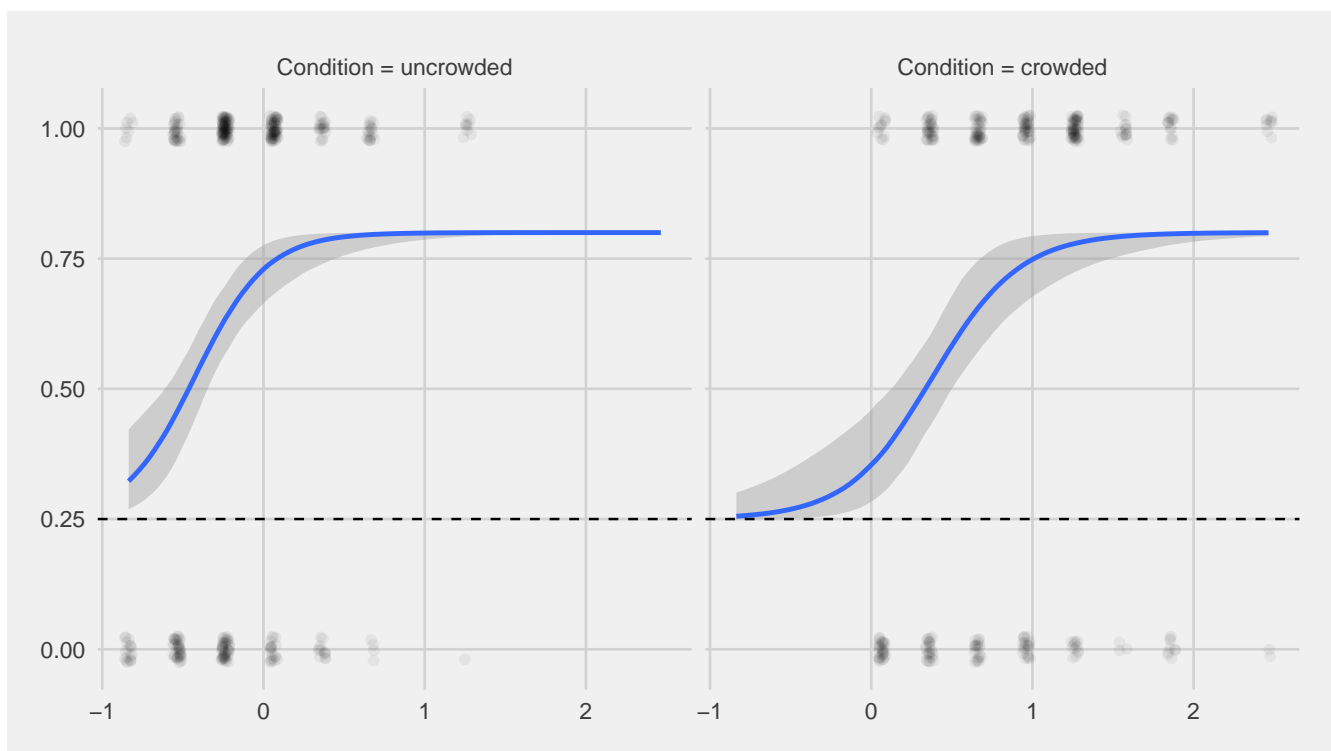


The next two models assume non-zero lapse rates of $\lambda = .05$ and $\lambda = .2$, respectively. Note how the strength of the evidence for an interaction systematically becomes weaker as we increase the λ . Of course, we do not know the ground truth, but it is worth noting that Subject 3 does not seem to achieve performance close to 100% accuracy even for the largest stimuli tested (there are both accurate and inaccurate answers at the largest size values in the crowded condition). Indeed, we will see in the next section that the *estimated* λ is close to 25%.

Hypothesis	Estimate	Est.Error	CI.Lower	CI.Upper	Evid.Ratio	Post.Prob	Star
$\beta_{Size} > 0$	1.735	0.291	1.278	2.232	Inf	1.000	*
$\beta_{Crowded} < 0$	-1.422	0.363	-2.073	-0.842	Inf	1.000	*
$\beta_{Size:Crowded} < 0$	-0.673	0.573	-1.580	0.266	7.556	0.883	



Hypothesis	Estimate	Est.Error	CI.Lower	CI.Upper	Evid.Ratio	Post.Prob	Star
$\beta_{Size} > 0$	4.280	0.939	2.917	5.942	Inf	1.000	*
$\beta_{Crowded} < 0$	-3.462	0.810	-4.905	-2.251	Inf	1.000	*
$\beta_{Size:Crowded} < 0$	-0.727	1.494	-3.206	1.689	2.265	0.694	

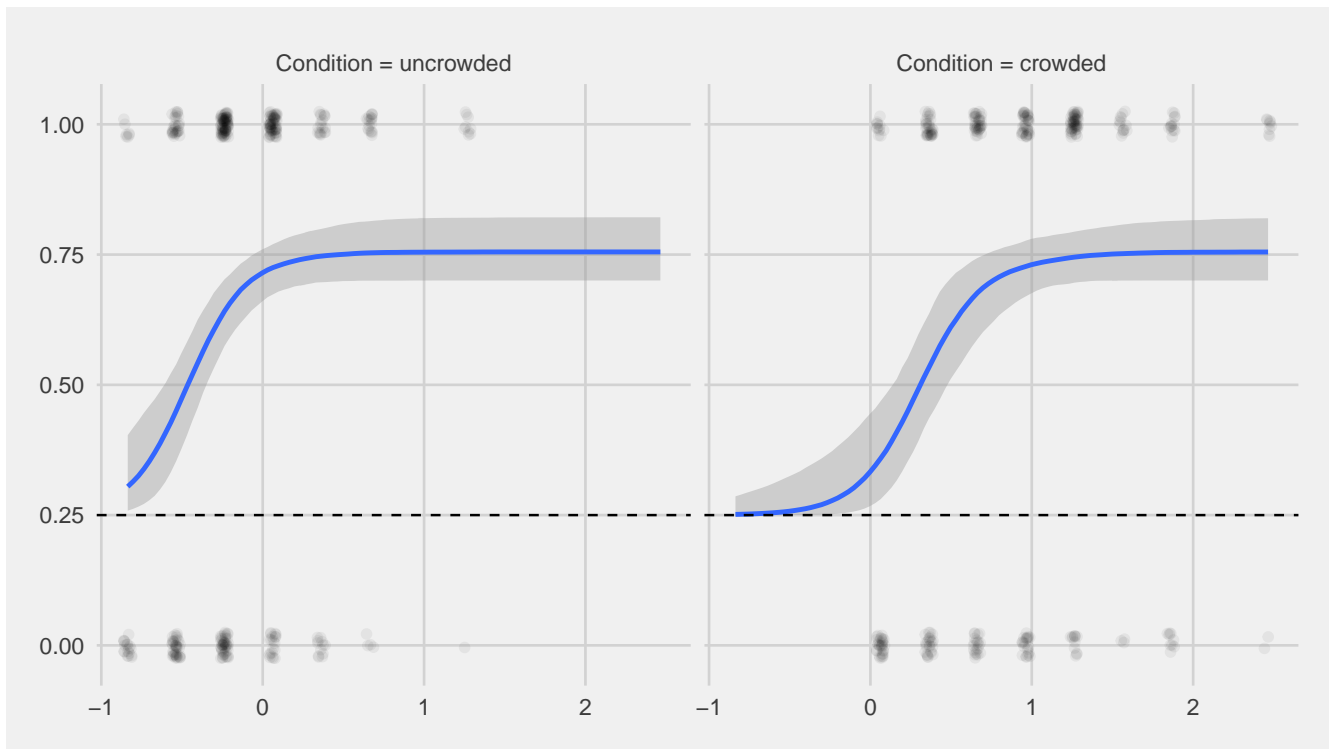


5.2.2 Inferring λ from the data

We infer the lapse rate from the data while assuming that it is constant across conditions. This is the assumption made in most psychometric analyses. It is, however, worth noting that this assumption can introduce bias. We have seen this bias in the simulation study presented above (see also the readings listed at the beginning of this document).

```
BF <- brmsformula(
  ResponseCorrect ~ .25 + (1-.25-lambda) * inv_logit(eta),
  eta ~ 1 + Size * Condition,
  lambda ~ 1,
  family = bernoulli(link="identity"),
  nl = TRUE
)
```

Hypothesis	Estimate	Est.Error	CI.Lower	CI.Upper	Evid.Ratio	Post.Prob	Star
$\beta_{Size} > 0$	5.669	1.766	3.295	8.774	Inf	1.000	*
$\beta_{Crowded} < 0$	-4.401	1.332	-6.729	-2.587	Inf	1.000	*
$\beta_{Size:Crowded} < 0$	-0.361	1.927	-3.216	2.866	1.516	0.602	
$\lambda > 0$	0.244	0.031	0.189	0.291	Inf	1.000	*



The estimated λ is close to 25%. Of course, we do not know the ground truth for this data. It is possible, for example, that Subject 3's performance would continue to increase had the subject been tested at even larger stimulus sizes. And this means that estimating the joint distribution of λ , α , and β (as we have done here) can introduce bias into the estimated of α and β , despite the fact that it is the model that arguably more closely reflects that generative process underlying the data. Further adding to researchers' conundrum is the fact that the adaptive procedure used in this, and many other, experiments makes it particularly hard to reliably estimate λ (see e.g., Prins, 2013). And this in turn can affect how biased and reliable the estimates of the *perceptual model* are (α and β , also Prins, 2013).

6 Combining the data from all subjects: mixed-effects psychometric models

One appeal of approaching psychometric data from the perspective of GLMMs is that we can apply the same random effect approach as in GLMMs for the lapse rate model (which then falls into the class of NLMMs). The following is an example of the NLMM specification of the mixed-effects logistic model fit to Ashley's data.

```
BF.lapse.mixed <- brmsformula(
  # reparameterizing the model by fitting lapses in log-odds space (in order to aid convergence)
  # this means we need to convert the lapses back into proportion space below.
  ResponseCorrect ~ .25 + (.75 - inv_logit(lapse)) * inv_logit(eta),
  eta ~ 1 + Size * Condition (1 + Size | Subject),
  lapse ~ 1 + (1 | Subject),
  family = bernoulli(link="identity"),
  nl = TRUE
)
```

Warning: There were 2 divergent transitions after warmup. Increasing adapt_delta above 0.999 may help. See <http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup>

```
Family: bernoulli
Links: mu = identity
Formula: ResponseCorrect ~ 0.25 + (0.75 - inv_logit(lapse)) * inv_logit(eta)
        eta ~ 1 + Size + (1 + Size | Subject)
        lapse ~ 1 + (1 | Subject)
Data: d (Number of observations: 3949)
Samples: 4 chains, each with iter = 8000; warmup = 4000; thin = 5;
         total post-warmup samples = 3200
```

```
Group-Level Effects:
~Subject (Number of levels: 10)

              Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS
sd(eta_Intercept)    0.71     0.24    0.38    1.33 1.00    2791
sd(eta_Size)         0.64     0.30    0.11    1.32 1.00    1547
sd(lapse_Intercept)  1.98     1.79    0.09    6.88 1.00    2008
cor(eta_Intercept,eta_Size) 0.49     0.35   -0.37    0.94 1.00    2609

              Tail_ESS
sd(eta_Intercept)    2908
sd(eta_Size)         1956
sd(lapse_Intercept)  2735
cor(eta_Intercept,eta_Size) 2630
```

```
Population-Level Effects:
              Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
eta_Intercept    0.45     0.28   -0.07    1.06 1.00    2052    2722
eta_Size         1.68     0.29    1.13    2.28 1.00    2264    2498
lapse_Intercept -5.41     3.99  -14.73   -2.24 1.00    2035    1870
```

Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS and Tail_ESS are effective sample size measures, and Rhat is the potential scale reduction factor on split chains (at convergence, Rhat = 1).

Hypothesis	Estimate	Est.Error	CI.Lower	CI.Upper	Evid.Ratio	Post.Prob	Star
$\beta_{Size} > 0$	1.678	0.290	1.228	2.155	3199.000	1.000	*
$\lambda > 0$	-5.413	3.988	-11.081	-2.514	0.001	0.001	

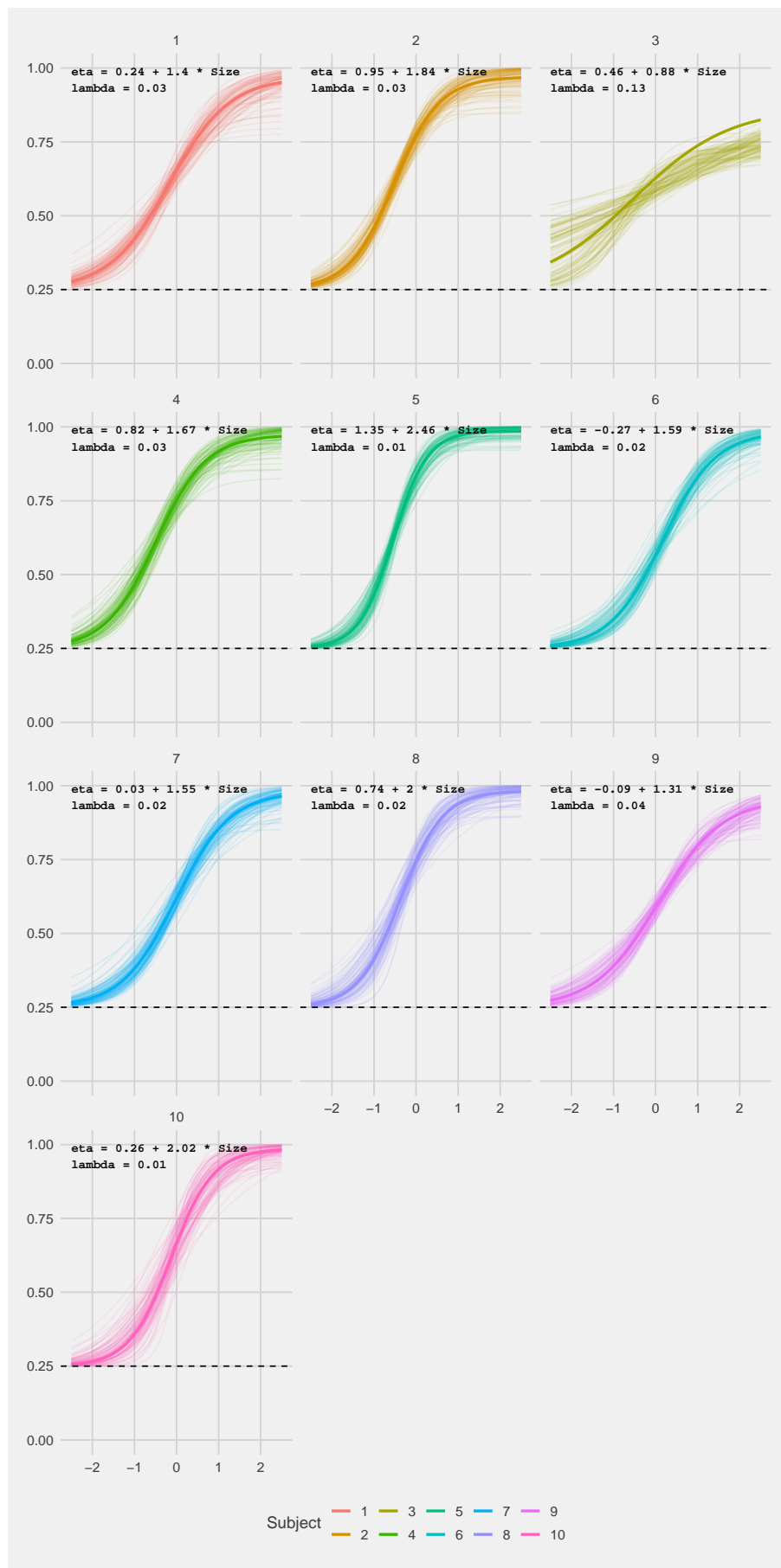
The following plots shows the predictions for each of 100 random posterior samples from each subjects (each curve corresponds to one posterior sample). The solid lines shows the predictions based on the *means* of the posterior samples. The reason this line does not always fall in the center of the other lines is that the three parameters of the model can be correlated (within and across participants). Taking the mean, rather than marginalizing over the entire posterior distribution, ignores those correlations (but is a *lot* faster to compute and thus used for the present purpose). The parameter summaries at the top left of each graph show those mean parameter values—i.e., those are the parameters of the solid line.

Scale for 'y' is already present. Adding another scale for 'y', which will

replace the existing scale.

```
`summarise()` regrouping output by 'Subject' (override with `.groups` argument)
```

```
`summarise()` ungrouping output (override with `.groups` argument)
```



And the joint posterior distribution of the model's three fixed effect parameters (and could do similarly for each subject). This reveals a correlation between the intercept and slope of the model. To view this interactive plot you might have to compile (knit) this document to HTML (rather than PDF) or go through the code incrementally and then execute just this code chunk.

```
No trace type specified:  
  Based on info supplied, a 'scatter3d' trace seems appropriate.  
  Read more about this trace type -> https://plot.ly/r/reference/#scatter3d  
  
No scatter3d mode specified:  
  Setting the mode to markers  
  Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode  
  
Warning: `arrange_()` is deprecated as of dplyr 0.7.0.  
Please use `arrange()` instead.  
See vignette('programming') for more help  
This warning is displayed once every 8 hours.  
Call `lifecycle::last_warnings()` to see where this warning was generated.
```

WebGL is not
supported by your
browser - visit
<https://get.webgl.org>
for more info

7 Alternative perceptual models

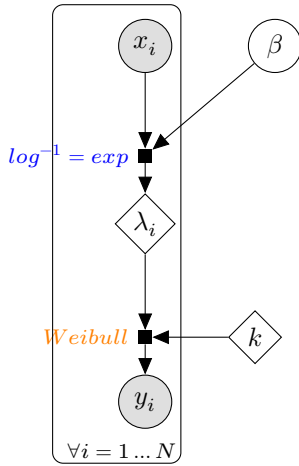
7.1 Weibull regression

The Weibull model is not a GLM, even if not combined with the lapse rate model. The Weibull has two parameters, the shape parameter k and the scale parameter λ . Only if the shape parameter is known, is the remaining model a GLM (see <https://stats.stackexchange.com/questions/277466/is-weibull-distribution-a-exponential-family>).

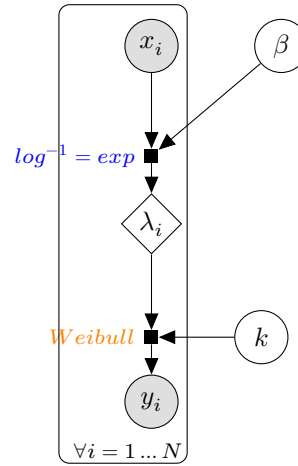
8 Appendix

8.1 Inferring γ

The examples shown above assume that γ is known. That is commonly the case. However, there are cases in which we want to fit both γ and λ .



(a) only scale parameter (λ) is inferred



(b) both shape (k and scale parameters (λ) are inferred

Figure 1: The Weibull regression is *not* a GLM, unless the shape parameter k is fixed (a). The Weibull model with both its scale and shape parameter can thus be seen as an infinite set of GLMs.

8.1.1 Mixture formulation of the lapsing mixed-effects logistic model

When inferring both γ and λ is substantially more efficient (and often necessary) to switch to the mixture formulation of the model, as it allows us to orthogonalize γ and λ . This makes this formulation of the model for computationally efficient, and more likely to converge. The mixture specification is, however, only available if the distributional family of the perceptual model is defined over real numbers (like the lapse part of the model). This is not, for example, the case for the Weibull (and hence Gumbel) model, which is defined over counts (integers).

Hypothesis	Estimate	Est.Error	CI.Lower	CI.Upper	Evid.Ratio	Post.Prob	Star
$\beta_{Size} > 0$	4.852	0.720	3.827	6.169	Inf	1.000	*
$\beta_{Crowded} < 0$	-1.606	0.417	-2.284	-0.958	665.667	0.999	*
$\beta_{Size:Crowded} < 0$	0.030	0.656	-1.010	1.145	0.971	0.493	

The following plots shows the predictions for each of 100 random posterior samples from each subjects (each curve corresponds to one posterior sample). The solid lines shows the predictions based on the *means* of the posterior samples. This line does not have to fall in the center of the other lines is that the three parameters of the model can be correlated (within and across participants). Taking the mean, rather than marginalizing over the entire posterior distribution, ignores those correlations (but is a *lot* faster to compute and thus used for the present purpose). The parameter summaries at the top left of each graph show those mean parameter values—i.e., those are the parameters of the solid line.

Distribution of lapse rate (mixture 1 weight), bias (mixture 2 mean), and mean of psychometric model (mixture 2):

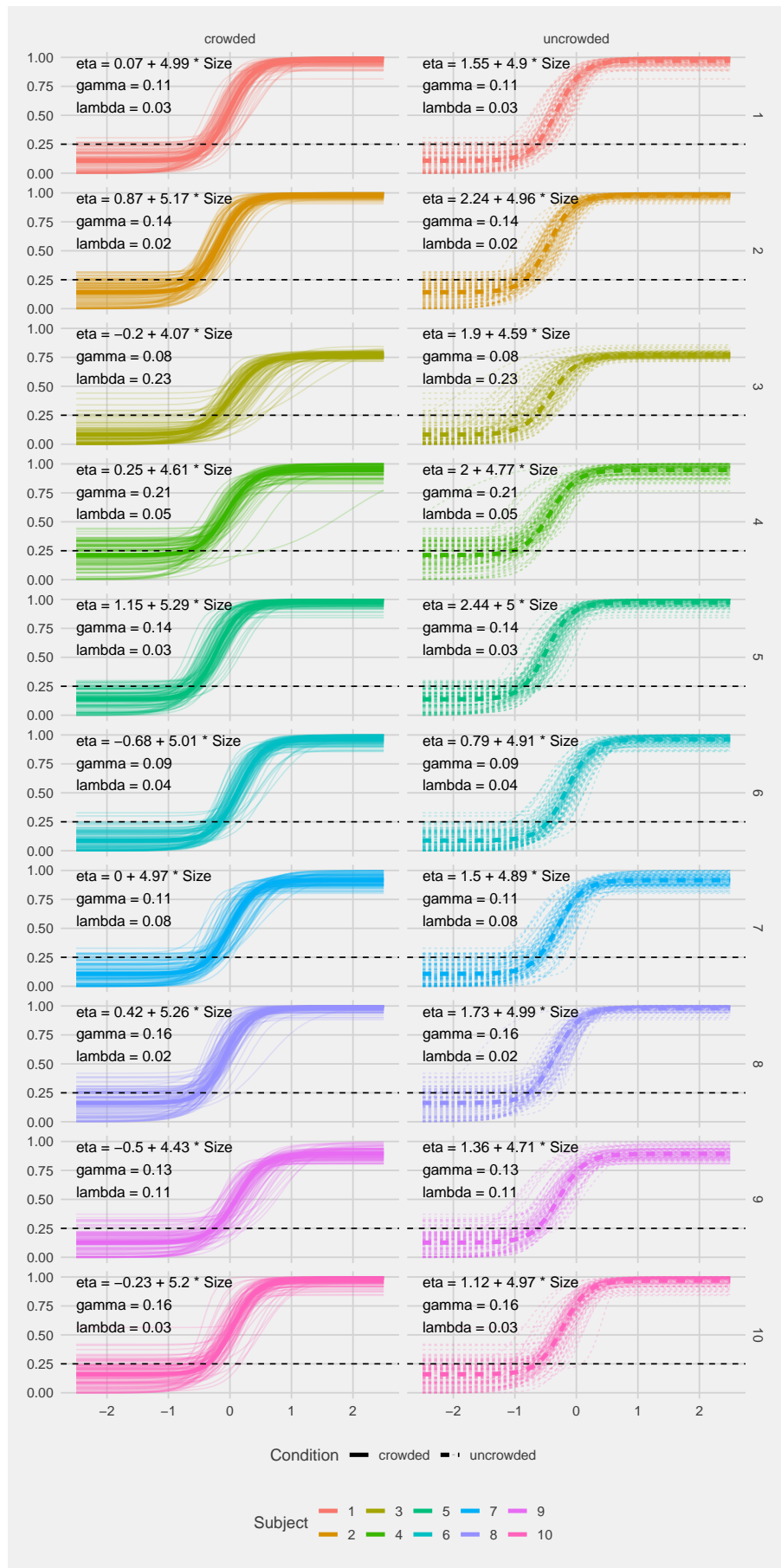
WebGL is not
supported by your
browser - visit
<https://get.webgl.org>
for more info

WebGL is not
supported by your
browser - visit
<https://get.webgl.org>
for more info

Lapse rate vs. mean and slope of psychometric model:

WebGL is not
supported by your
browser - visit
<https://get.webgl.org>
for more info

Visualizing predictions of the model:



We can use animation to further illustrate the hierarchical nature of the model. Note how the different posterior samples (each animation step) ‘move in sync’. That is because the subject-specific predictions include the population-level predictions that are common to all subjects:

8.1.2 NLM formulation of the lapsing logistic model

A less computationally efficient (or practical) way of inferring both γ and λ uses the NLM formulation used above when inferring only λ . This is illustrated here for a single subject (Subject 1). Note that this model did not converge and should not be trusted. More constraining priors and/or longer sampling might achieve convergence. This illustrates the challenges of inferring both λ and γ , and highlights the advantage of the mixture formulation.

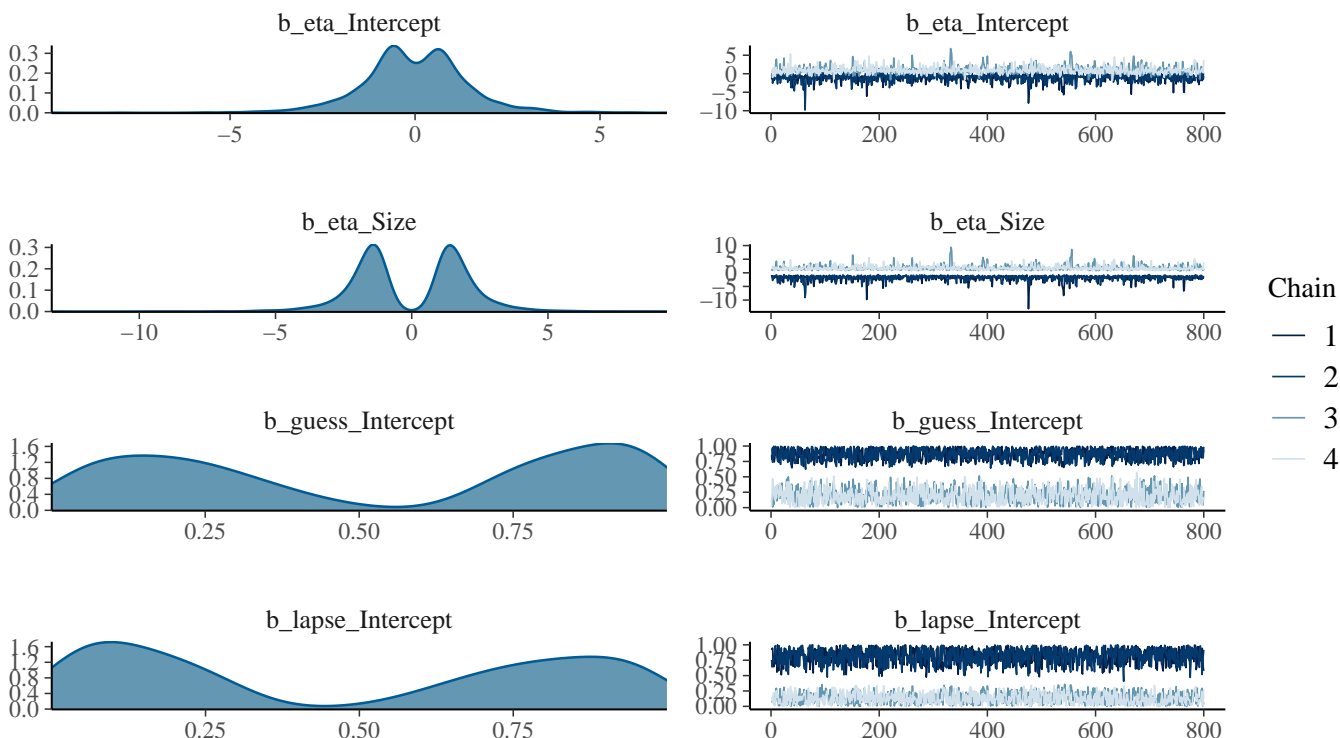
Warning: Parts of the model have not converged (some Rhats are > 1.05). Be careful when analysing the results! We recommend running more iterations and/or setting stronger priors.

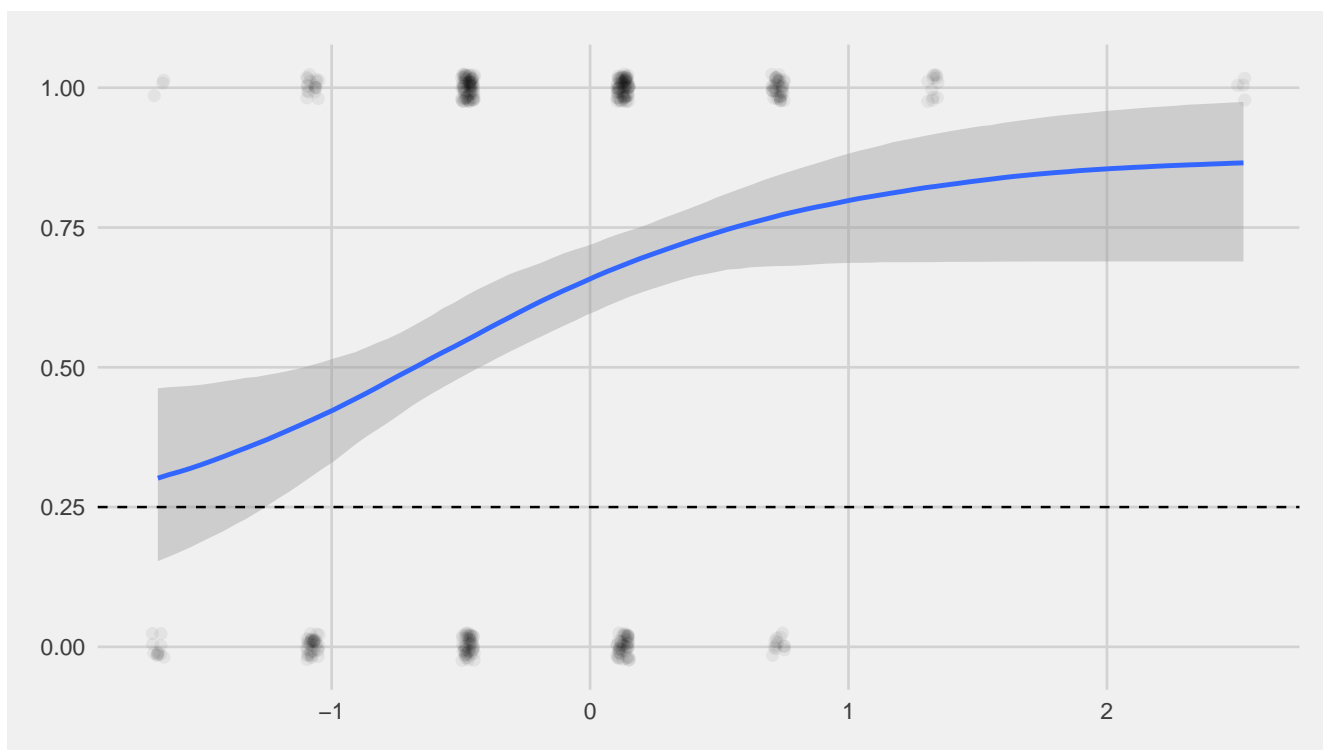
```
Family: bernoulli
Links: mu = identity
Formula: ResponseCorrect ~ guess + (1 - guess - lapse) * inv_logit(eta)
        eta ~ 1 + Size
        guess ~ 1
        lapse ~ 1
Data: d %>% filter(Subject == "1") (Number of observations: 396)
Samples: 4 chains, each with iter = 8000; warmup = 4000; thin = 5;
        total post-warmup samples = 3200
```

Population-Level Effects:

	Estimate	Est.Error	1-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
eta_Intercept	-0.02	1.40	-2.83	2.84	1.54	7	137
eta_Size	-0.01	2.01	-3.58	3.40	1.73	6	153
guess_Intercept	0.53	0.35	0.02	0.99	1.73	6	116
lapse_Intercept	0.47	0.36	0.01	0.98	1.73	6	127

Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS and Tail_ESS are effective sample size measures, and Rhat is the potential scale reduction factor on split chains (at convergence, Rhat = 1).





This model fit does not converge, and a look at the posterior samples suggests why. This model has the typical hallmarks of unidentifiability with two solutions emerging (symmetrical around 0 log-odds for the logistic part of the model; symmetrical around .5 for the guess and lapse rate, which are expressed on the scale of proportions).

We refit the model while constraining the guess and lapse rate to be less than .5, i.e., less than half of the trials are attentional lapses. This is a very weak assumption.

```
my.priors <- c(
  prior(student_t(3, 0, 2.5), class = "b", nlpar = "eta"),
  prior(beta(1, 1), nlpar = "lambda", lb = 0, ub = .5),
  prior(beta(1, 1), nlpar = "gamma", lb = 0, ub = .5)
)
```

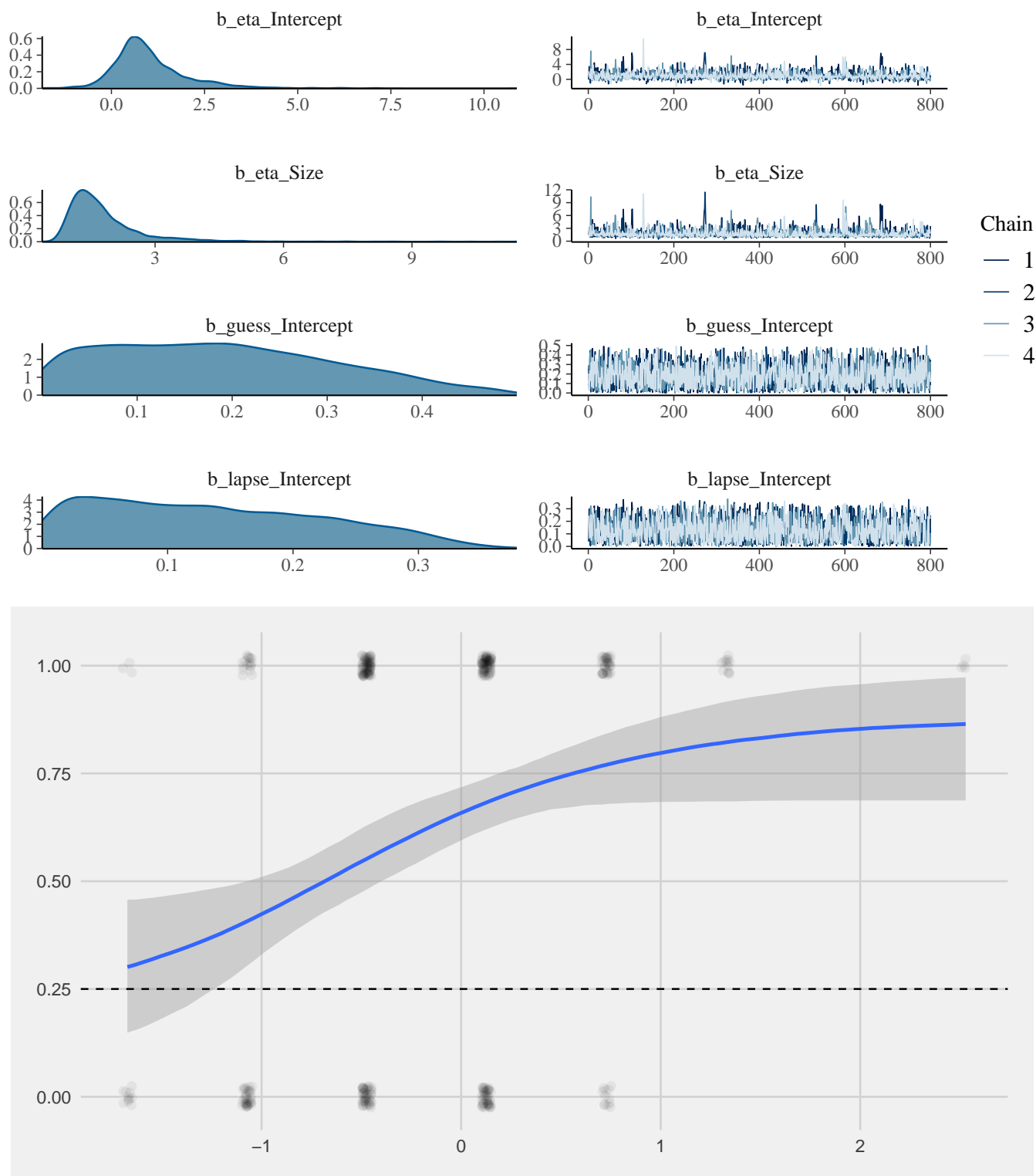
Let's refit the model with these revised priors to the data from Subject 1. This model converges, shows no signs of multimodality in the posterior, and reveals an effect of letter size (the 95% credible interval does not include 0). The predictions of the model also make sense (e.g., that it converges against chance at 25%). It is, however, worth noting that there is *substantial* uncertainty about the lapse rate and bias terms. This makes sense: although we have a lot of trials for each subject, we have very little information about the effect of size on the subject's responses since we're only testing at a few points along the size continuum. Additionally, psychometric designs tend to test mostly in the mid-performane part of the continuum. This is the case here, too: note that most of the data is in the middle of this subject's data. These common properties of psychometric designs makes it difficult to reliably distinguish between the effect of stimulus and effects of lapsing.

```
Family: bernoulli
Links: mu = identity
Formula: ResponseCorrect ~ guess + (1 - guess - lapse) * inv_logit(eta)
      eta ~ 1 + Size
      guess ~ 1
      lapse ~ 1
Data: d %>% filter(Subject == "1") (Number of observations: 396)
Samples: 4 chains, each with iter = 8000; warmup = 4000; thin = 5;
         total post-warmup samples = 3200
```

Population-Level Effects:

	Estimate	Est.Error	l-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
eta_Intercept	1.01	1.00	-0.42	3.43	1.00	2526	2189
eta_Size	1.79	0.93	0.84	4.15	1.00	2406	1986
guess_Intercept	0.19	0.12	0.01	0.44	1.00	3053	2903
lapse_Intercept	0.13	0.09	0.01	0.31	1.00	2452	2508

Samples were drawn using `sampling(NUTS)`. For each parameter, Bulk_ESS and Tail_ESS are effective sample size measures, and Rhat is the potential scale reduction factor on split chains (at convergence, Rhat = 1).



9 Session info

```
devtools::session_info()
```

```
- Session info -----
setting value
version      R version 4.0.2 (2020-06-22)
os           macOS High Sierra 10.13.6
system       x86_64, darwin17.0
ui           X11
language     (EN)
collate      en_US.UTF-8
ctype        en_US.UTF-8
tz           America/New_York
date         2020-12-16

- Packages -----
package      * version      date      lib source
abind         1.4-5        2016-07-21 [1] CRAN (R 4.0.2)
arrayhelpers  1.1-0        2020-02-04 [1] CRAN (R 4.0.2)
assertthat   0.2.1        2019-03-21 [1] CRAN (R 4.0.2)
backports     1.2.0        2020-11-02 [1] CRAN (R 4.0.2)
base64enc     0.1-3        2015-07-28 [1] CRAN (R 4.0.2)
bayesplot     1.7.2        2020-05-28 [1] CRAN (R 4.0.2)
boot          1.3-25       2020-04-26 [1] CRAN (R 4.0.2)
bridgesampling 1.0-0        2020-02-26 [1] CRAN (R 4.0.2)
brms          * 2.14.4       2020-11-03 [1] CRAN (R 4.0.2)
Broddingnag   1.2-6        2018-08-13 [1] CRAN (R 4.0.2)
broom         * 0.7.2        2020-10-20 [1] CRAN (R 4.0.2)
broom.mixed   * 0.2.6        2020-05-17 [1] CRAN (R 4.0.2)
callr         3.5.1        2020-10-13 [1] CRAN (R 4.0.2)
cellranger    1.1.0        2016-07-27 [1] CRAN (R 4.0.2)
checkmate     2.0.0        2020-02-06 [1] CRAN (R 4.0.2)
class         7.3-17       2020-04-26 [1] CRAN (R 4.0.2)
classInt      0.4-3        2020-04-07 [1] CRAN (R 4.0.2)
cli           2.2.0        2020-11-20 [1] CRAN (R 4.0.2)
cluster       2.1.0        2019-06-19 [1] CRAN (R 4.0.2)
coda          0.19-4       2020-09-30 [1] CRAN (R 4.0.2)
codetools     0.2-18       2020-11-04 [1] CRAN (R 4.0.2)
colorspace    2.0-0        2020-11-11 [1] CRAN (R 4.0.2)
colourpicker  1.1.0        2020-09-14 [1] CRAN (R 4.0.2)
crayon        1.3.4        2017-09-16 [1] CRAN (R 4.0.2)
crosstalk     1.1.0.1      2020-03-13 [1] CRAN (R 4.0.2)
curl          4.3          2019-12-02 [1] CRAN (R 4.0.1)
data.table    1.13.4       2020-12-08 [1] CRAN (R 4.0.2)
DBI           1.1.0        2019-12-15 [1] CRAN (R 4.0.2)
dbplyr        2.0.0        2020-11-03 [1] CRAN (R 4.0.2)
desc          1.2.0        2018-05-01 [1] CRAN (R 4.0.2)
devtools      2.3.2        2020-09-18 [1] CRAN (R 4.0.2)
digest        0.6.27       2020-10-24 [1] CRAN (R 4.0.2)
distributional 0.2.1        2020-10-06 [1] CRAN (R 4.0.2)
dplyr         * 1.0.2        2020-08-18 [1] CRAN (R 4.0.2)
DT            0.16         2020-10-13 [1] CRAN (R 4.0.2)
dygraphs      1.1.1.6      2018-07-11 [1] CRAN (R 4.0.2)
ei071         1.7-4        2020-10-14 [1] CRAN (R 4.0.2)
ellipsis      0.3.1        2020-05-15 [1] CRAN (R 4.0.2)
emmeans       1.5.2-1      2020-10-25 [1] CRAN (R 4.0.2)
estimability  1.3          2018-02-11 [1] CRAN (R 4.0.2)
evaluate      0.14         2019-05-28 [1] CRAN (R 4.0.1)
fansl         0.4.1        2020-01-08 [1] CRAN (R 4.0.2)
farver        2.0.3        2020-01-16 [1] CRAN (R 4.0.2)
fastmap       1.0.1        2019-10-08 [1] CRAN (R 4.0.2)
forcats       * 0.5.0        2020-03-01 [1] CRAN (R 4.0.2)
foreign       0.8-80       2020-05-24 [1] CRAN (R 4.0.2)
Formula       1.2-4        2020-10-16 [1] CRAN (R 4.0.2)
fs            1.5.0        2020-07-31 [1] CRAN (R 4.0.2)
gamm4         0.2-6        2020-04-03 [1] CRAN (R 4.0.2)
generics      0.1.0        2020-10-31 [1] CRAN (R 4.0.2)
gganimate     * 1.0.7        2020-10-15 [1] CRAN (R 4.0.2)
ggdist        2.3.0        2020-10-31 [1] CRAN (R 4.0.2)
```

ggplot2	* 3.3.2	2020-06-19	[1]	CRAN	(R 4.0.2)
ggridges	0.5.2	2020-01-12	[1]	CRAN	(R 4.0.2)
gthemes	* 4.2.0	2019-05-13	[1]	CRAN	(R 4.0.2)
gifski	* 0.8.6	2018-09-28	[1]	CRAN	(R 4.0.2)
glue	1.4.2	2020-08-27	[1]	CRAN	(R 4.0.2)
gridExtra	2.3	2017-09-09	[1]	CRAN	(R 4.0.2)
gtable	0.3.0	2019-03-25	[1]	CRAN	(R 4.0.2)
gtools	3.8.2	2020-03-31	[1]	CRAN	(R 4.0.2)
haven	2.3.1	2020-06-01	[1]	CRAN	(R 4.0.2)
highr	0.8	2019-03-20	[1]	CRAN	(R 4.0.2)
Hmisc	4.4-2	2020-11-29	[1]	CRAN	(R 4.0.2)
hms	0.5.3	2020-01-08	[1]	CRAN	(R 4.0.2)
htmlTable	2.1.0	2020-09-16	[1]	CRAN	(R 4.0.2)
htmltools	0.5.0	2020-06-16	[1]	CRAN	(R 4.0.2)
htmlwidgets	1.5.2	2020-10-03	[1]	CRAN	(R 4.0.2)
httpuv	1.5.4	2020-06-06	[1]	CRAN	(R 4.0.2)
httr	1.4.2	2020-07-20	[1]	CRAN	(R 4.0.2)
igraph	1.2.6	2020-10-06	[1]	CRAN	(R 4.0.2)
inline	0.3.17	2020-12-01	[1]	CRAN	(R 4.0.2)
jpeg	0.1-8.1	2019-10-24	[1]	CRAN	(R 4.0.2)
jsonlite	1.7.1	2020-09-07	[1]	CRAN	(R 4.0.2)
KernSmooth	2.23-18	2020-10-29	[1]	CRAN	(R 4.0.2)
knitr	* 1.30.4	2020-12-15	[1]	Github (yihui/knitr@f8f90ba)	
labeling	0.4.2	2020-10-20	[1]	CRAN	(R 4.0.2)
later	1.1.0.1	2020-06-05	[1]	CRAN	(R 4.0.2)
lattice	0.20-41	2020-04-02	[1]	CRAN	(R 4.0.2)
latticeExtra	0.6-29	2019-12-19	[1]	CRAN	(R 4.0.2)
lazyeval	0.2.2	2019-03-15	[1]	CRAN	(R 4.0.2)
lifecycle	0.2.0	2020-03-06	[1]	CRAN	(R 4.0.2)
lme4	1.1-26	2020-12-01	[1]	CRAN	(R 4.0.2)
loo	2.4.0	2020-12-05	[1]	CRAN	(R 4.0.2)
lpSolve	5.6.15	2020-01-24	[1]	CRAN	(R 4.0.2)
lubridate	1.7.9.2	2020-11-13	[1]	CRAN	(R 4.0.2)
magrittr	* 2.0.1	2020-11-17	[1]	CRAN	(R 4.0.2)
markdown	1.1	2019-08-07	[1]	CRAN	(R 4.0.2)
MASS	7.3-53	2020-09-09	[1]	CRAN	(R 4.0.2)
Matrix	1.2-18	2019-11-27	[1]	CRAN	(R 4.0.2)
matrixStats	0.57.0	2020-09-25	[1]	CRAN	(R 4.0.2)
memoise	1.1.0	2017-04-21	[1]	CRAN	(R 4.0.2)
mgcv	1.8-33	2020-08-27	[1]	CRAN	(R 4.0.2)
mime	0.9	2020-02-04	[1]	CRAN	(R 4.0.2)
miniUI	0.1.1.1	2018-05-18	[1]	CRAN	(R 4.0.2)
minqa	1.2.4	2014-10-09	[1]	CRAN	(R 4.0.2)
modelr	0.1.8	2020-05-19	[1]	CRAN	(R 4.0.2)
multcomp	1.4-15	2020-11-14	[1]	CRAN	(R 4.0.2)
munSELL	0.5.0	2018-06-12	[1]	CRAN	(R 4.0.2)
mvtnorm	1.1-1	2020-06-09	[1]	CRAN	(R 4.0.2)
nlme	3.1-150	2020-10-24	[1]	CRAN	(R 4.0.2)
nloptr	1.2.2.2	2020-07-02	[1]	CRAN	(R 4.0.2)
nnet	7.3-14	2020-04-26	[1]	CRAN	(R 4.0.2)
pillar	1.4.7	2020-11-20	[1]	CRAN	(R 4.0.2)
pkgbuild	1.1.0.9000	2020-08-06	[1]	Github (r-lib/pkgbuild@3a87bd9)	
pkgconfig	2.0.3	2019-09-22	[1]	CRAN	(R 4.0.2)
pkgload	1.1.0	2020-05-29	[1]	CRAN	(R 4.0.2)
plotly	* 4.9.2.1	2020-04-04	[1]	CRAN	(R 4.0.2)
plyr	1.8.6	2020-03-03	[1]	CRAN	(R 4.0.2)
png	0.1-7	2013-12-03	[1]	CRAN	(R 4.0.2)
prettyunits	1.1.1	2020-01-24	[1]	CRAN	(R 4.0.2)
processx	3.4.5	2020-11-30	[1]	CRAN	(R 4.0.2)
progress	1.2.2	2019-05-16	[1]	CRAN	(R 4.0.2)
projpred	2.0.2	2020-10-28	[1]	CRAN	(R 4.0.2)
promises	1.1.1	2020-06-09	[1]	CRAN	(R 4.0.2)
ps	1.5.0	2020-12-05	[1]	CRAN	(R 4.0.2)
purrr	* 0.3.4	2020-04-17	[1]	CRAN	(R 4.0.2)
R6	2.5.0	2020-10-28	[1]	CRAN	(R 4.0.2)
RColorBrewer	1.1-2	2014-12-07	[1]	CRAN	(R 4.0.2)
Rcpp	* 1.0.5	2020-07-06	[1]	CRAN	(R 4.0.2)
RcppParallel	5.0.2	2020-06-24	[1]	CRAN	(R 4.0.2)
readr	* 1.4.0	2020-10-05	[1]	CRAN	(R 4.0.2)
readxl	1.3.1	2019-03-13	[1]	CRAN	(R 4.0.2)
remotes	2.2.0	2020-07-21	[1]	CRAN	(R 4.0.2)

reprex	0.3.0	2019-05-16	[1]	CRAN (R 4.0.2)
reshape2	1.4.4	2020-04-09	[1]	CRAN (R 4.0.2)
rlang	0.4.9	2020-11-26	[1]	CRAN (R 4.0.2)
rmarkdown	2.5	2020-10-21	[1]	CRAN (R 4.0.2)
rpart	4.1-15	2019-04-12	[1]	CRAN (R 4.0.2)
rprojroot	2.0.2	2020-11-15	[1]	CRAN (R 4.0.2)
rsconnect	0.8.16	2019-12-13	[1]	CRAN (R 4.0.2)
rstan	2.21.2	2020-08-04	[1]	Github (stan-dev/rstan@a9a44bb)
rstantools	2.1.1	2020-07-06	[1]	CRAN (R 4.0.2)
rstudioapi	0.13	2020-11-12	[1]	CRAN (R 4.0.2)
rvest	0.3.6	2020-07-25	[1]	CRAN (R 4.0.2)
sandwich	3.0-0	2020-10-02	[1]	CRAN (R 4.0.2)
scales	* 1.1.1	2020-05-11	[1]	CRAN (R 4.0.2)
sessioninfo	1.1.1	2018-11-05	[1]	CRAN (R 4.0.2)
sf	0.9-6	2020-09-13	[1]	CRAN (R 4.0.2)
shiny	1.5.0	2020-06-23	[1]	CRAN (R 4.0.2)
shinyjs	2.0.0	2020-09-09	[1]	CRAN (R 4.0.2)
shinystan	2.5.0	2018-05-01	[1]	CRAN (R 4.0.2)
shinythemes	1.1.2	2018-11-06	[1]	CRAN (R 4.0.2)
StanHeaders	2.21.0-6	2020-08-16	[1]	CRAN (R 4.0.2)
statmod	1.4.35	2020-10-19	[1]	CRAN (R 4.0.2)
stringi	1.5.3	2020-09-09	[1]	CRAN (R 4.0.2)
stringr	* 1.4.0	2019-02-10	[1]	CRAN (R 4.0.2)
survival	3.2-7	2020-09-28	[1]	CRAN (R 4.0.2)
svUnit	1.0.3	2020-04-20	[1]	CRAN (R 4.0.2)
testthat	3.0.0	2020-10-31	[1]	CRAN (R 4.0.2)
TH.data	1.0-10	2019-01-21	[1]	CRAN (R 4.0.2)
threejs	0.3.3	2020-01-21	[1]	CRAN (R 4.0.2)
tibble	* 3.0.4	2020-10-12	[1]	CRAN (R 4.0.2)
tidybayes	* 2.3.1	2020-11-02	[1]	CRAN (R 4.0.2)
tidyr	* 1.1.2	2020-08-27	[1]	CRAN (R 4.0.2)
tidyselect	1.1.0	2020-05-11	[1]	CRAN (R 4.0.2)
tidyverse	* 1.3.0	2019-11-21	[1]	CRAN (R 4.0.2)
TMB	1.7.18	2020-07-27	[1]	CRAN (R 4.0.2)
transformr	0.1.3	2020-07-05	[1]	CRAN (R 4.0.2)
tweenr	1.0.1	2018-12-14	[1]	CRAN (R 4.0.2)
units	0.6-7	2020-06-13	[1]	CRAN (R 4.0.2)
usethis	1.6.3	2020-09-17	[1]	CRAN (R 4.0.2)
V8	3.4.0	2020-11-04	[1]	CRAN (R 4.0.2)
vctrs	0.3.5	2020-11-17	[1]	CRAN (R 4.0.2)
viridisLite	0.3.0	2018-02-01	[1]	CRAN (R 4.0.1)
webshot	0.5.2	2019-11-22	[1]	CRAN (R 4.0.2)
withr	2.3.0	2020-09-22	[1]	CRAN (R 4.0.2)
xfun	0.19	2020-10-30	[1]	CRAN (R 4.0.2)
xml2	1.3.2	2020-04-23	[1]	CRAN (R 4.0.2)
xtable	1.8-4	2019-04-21	[1]	CRAN (R 4.0.2)
xts	0.12.1	2020-09-09	[1]	CRAN (R 4.0.2)
yaml	2.2.1	2020-02-01	[1]	CRAN (R 4.0.2)
zoo	1.8-8	2020-05-02	[1]	CRAN (R 4.0.2)

[1] /Library/Frameworks/R.framework/Versions/4.0/Resources/library