Psychometric models

A brief applied overview

T. Florian Jaeger

December 15, 2020

Contents

1 Goals of this tutorial **1**

2 Readings and other materials **2**

3 Psychometric functions **3**

4 Lapsing model **3**

5 A simple example **4**
5.1 Setting up the model . . . 4
5.2 Preparing the data . . . 5
5.3 Fitting the model . . . 5
5.4 Determining the subject-specific threshold . . . 6

6 Illustrating the consequences of assumptions about lapse rates (λ) **7**
6.1 Simulation study . . . 7
6.1.1 Visualizing the correlation among the intercept and slope . . . 10
6.1.2 Visualizing bias in the estimates . . . 10
6.2 Working with real data (unknown ground truth) . . . 11
6.2.1 Assuming a fixed lapse rate . . . 11
6.2.2 Inferring the lapse rate from the data . . . 14
6.3 Combining the data from all subjects . . . 15

7 Weibull regression **15**

8 Appendix **15**

9 Inferring γ **15**
9.1 Mixture formulation of the lapsing mixed-effects logistic model . . . 16
9.2 NLM formulation of the lasping logistic model . . . 20
9.3 NLMM formulation of the lasping mixed-effect logistic model . . . 24

10 Session info **26**

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.

```
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 ...
 $ 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" ...
```

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.

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, who pointed me to them):

- 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 model. It's not yet clear to me whether `brms` would also be able to fit the cumulative Gaussian model. In any case though, at least the mixture specification is only available if the response part of the 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. 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 functions

In psychometric modeling, we predict subjects' responses from a stimulus value. A number of different predictive models are frequently used, with choices varying between labs, based on goodness of fit, or field-specific conventions. Of relevance to this class, all of these models fall into the class of non-linear models (NLM) and some are also generalized linear models (GLMs). That is, psychometric functions/modeling is partly a subset of the type of models we cover in this class, and partly closely related. 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 psychometric functions 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 also the readings listed at the top of this tutorial.

Any of these functions for the stimulus-driven model 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. We'll start with some more background on the lapsing model.

4 Lapsing model

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

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

where $p(correct\ response|stimulus, \alpha, \beta)$ is the **perceptual model**—i.e., the function that maps input stimuli onto responses (e.g,. the logistic model discussed above). This perceptual model depends on two parameters α and β. 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 Wiechman 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**.

An alternative formulation of the same processes, describes the lapse model as a mixture model of two components. Like in the Wiechman and Hill formulation, one component 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 Wiechman and Hill formulation (neither is *bias* the same as γ/*guess*).

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

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

$$lapse = \gamma + \lambda \tag{3}$$

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

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

$$\gamma = lapse * bias \tag{6}$$

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 changes 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 (but it can still be fit with maximum likelihood fitting).

5 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.

5.1 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 way to formulate this model directly as a mixture model (see replies to https://discourse.mc-stan.org/t/fitting-lapsing-psychometric-functions-with-brms/5762/2) but the following non-linear model syntax is perhaps 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)
)
```

5.2 Preparing the data

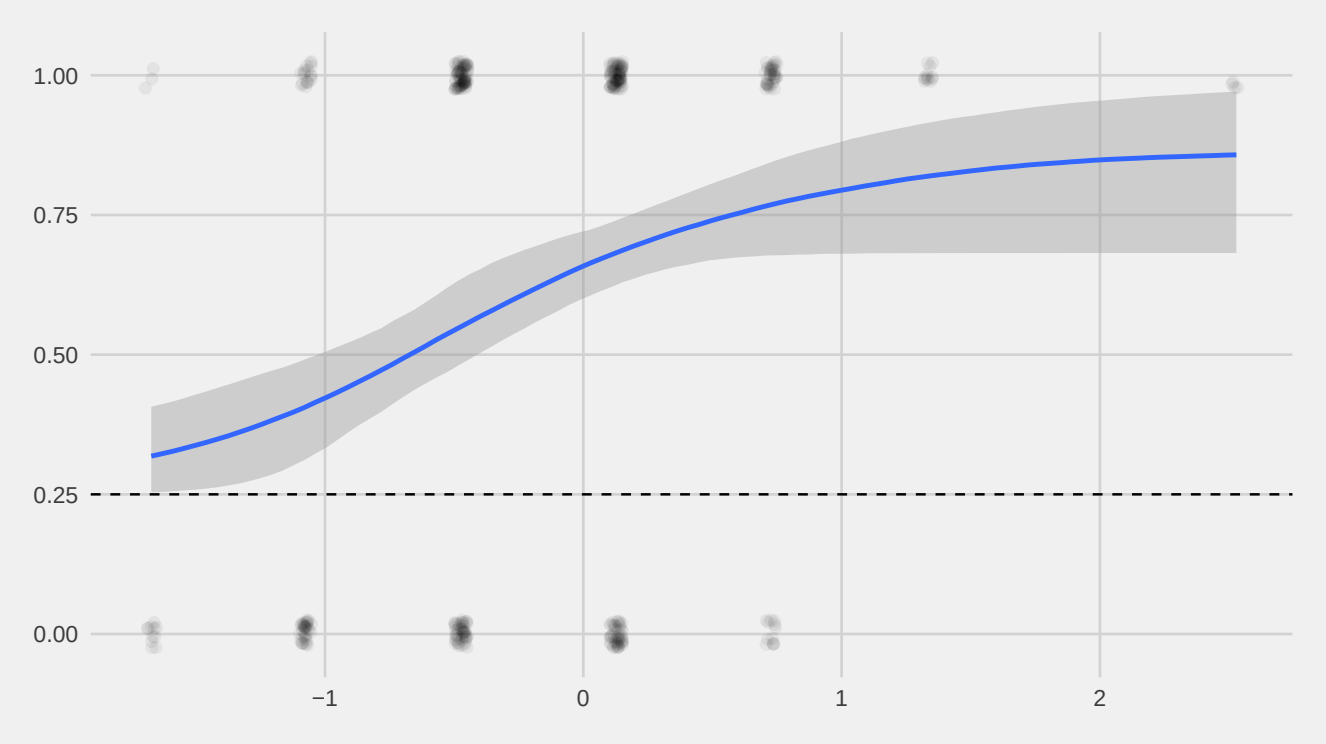
So that the priors are on the right 'scale', we follow (Gelman et al., 2008), and standardize the continuous predictors in the model. As is typical for analyses of experimental data with two balanced conditions, we deviation/sum/abova/effect-code the condition variable, with the 'treatment' receiving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

5.3 Fitting the model

```
 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 l-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).
```



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.

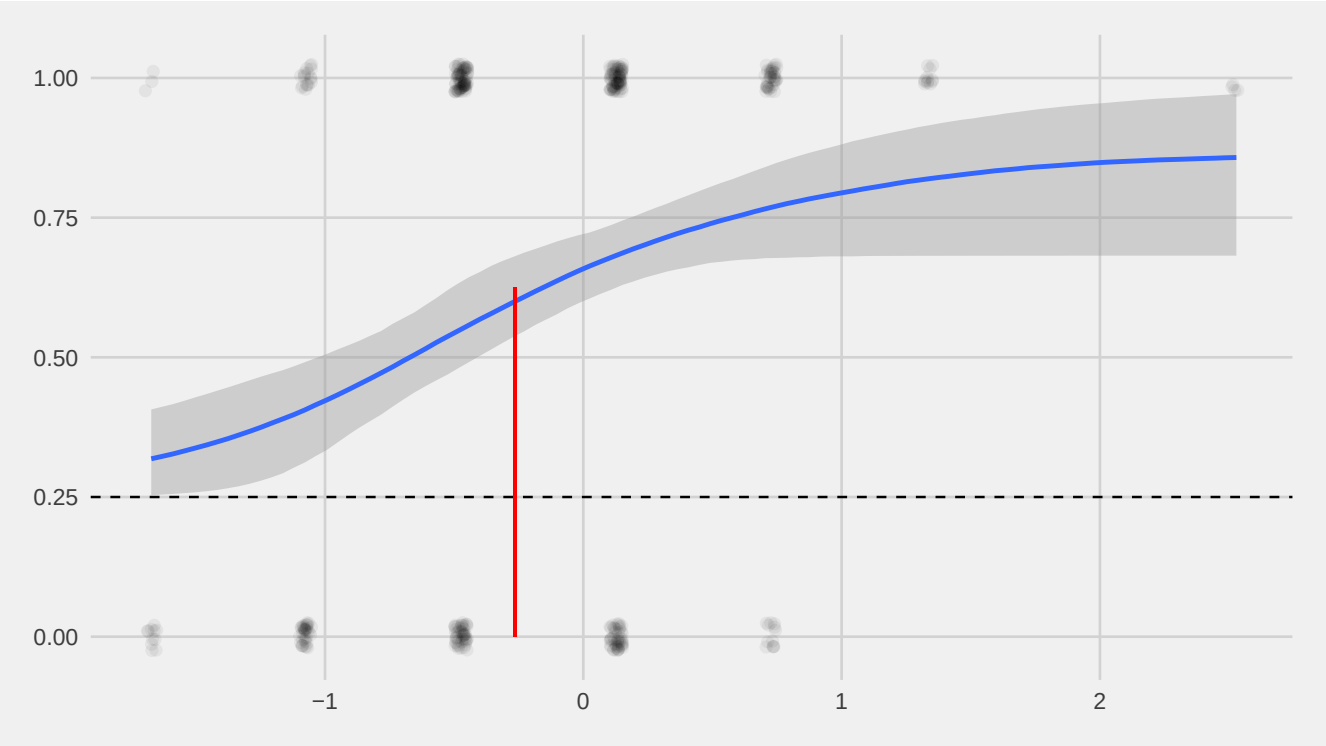
5.4 Determining the subject-specific threshold

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 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.

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

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

$$logit(.625) = \alpha + \beta * Size \Rightarrow \quad (8)$$
$$0.5108256 = 1.04 + 2.00 * Size \Leftrightarrow \quad (9)$$
$$-0.2645872 = Size \quad (10)$$



6 Illustrating the consequences of assumptions about lapse rates (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.

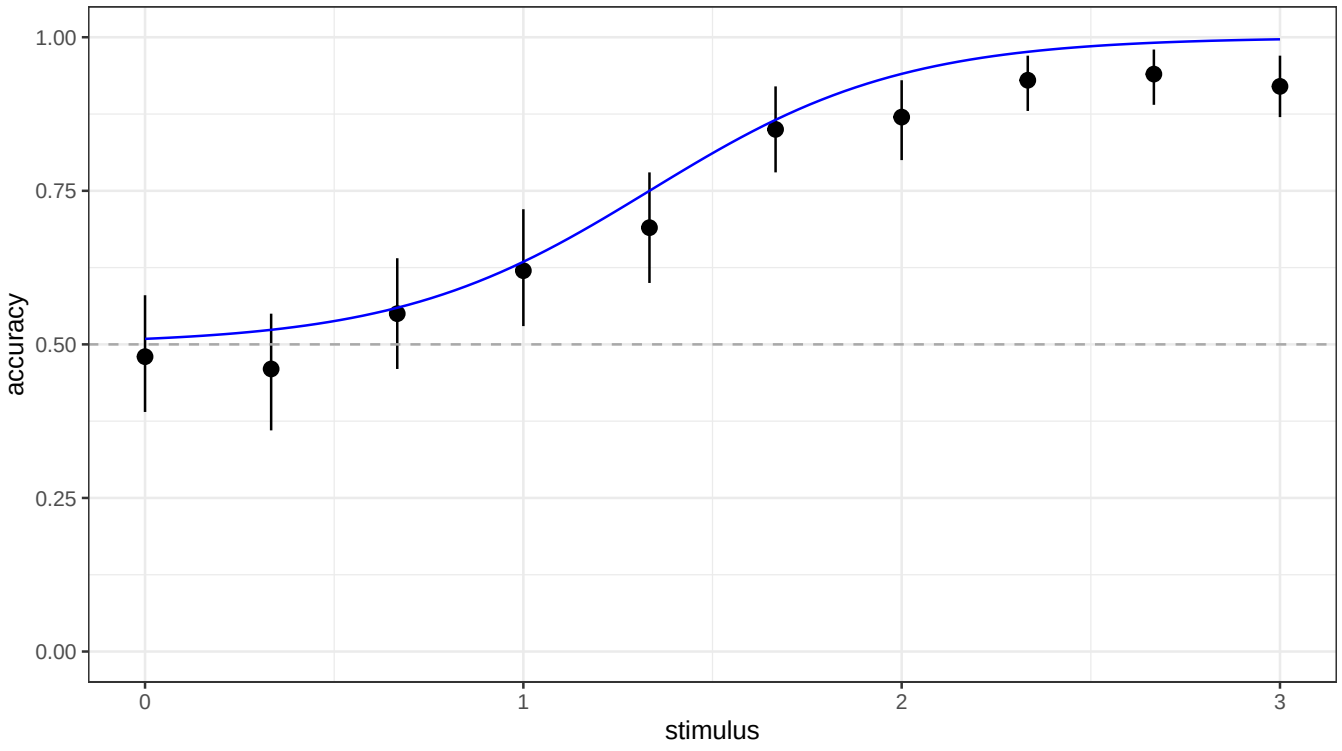
6.1 Simulation study

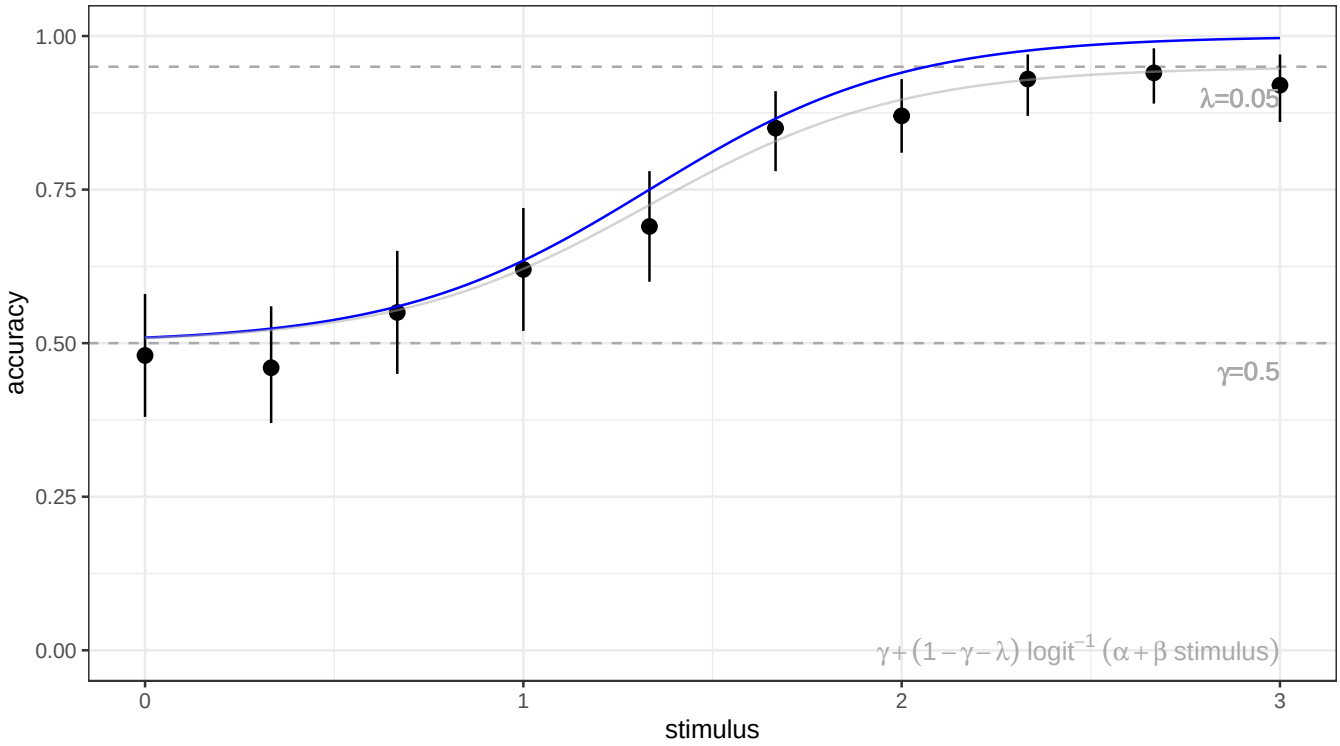
We first make a data generator for a 2AFC task that creates data following the Wiechman and Hill (2001) model with a logistic response function:

```
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,
```

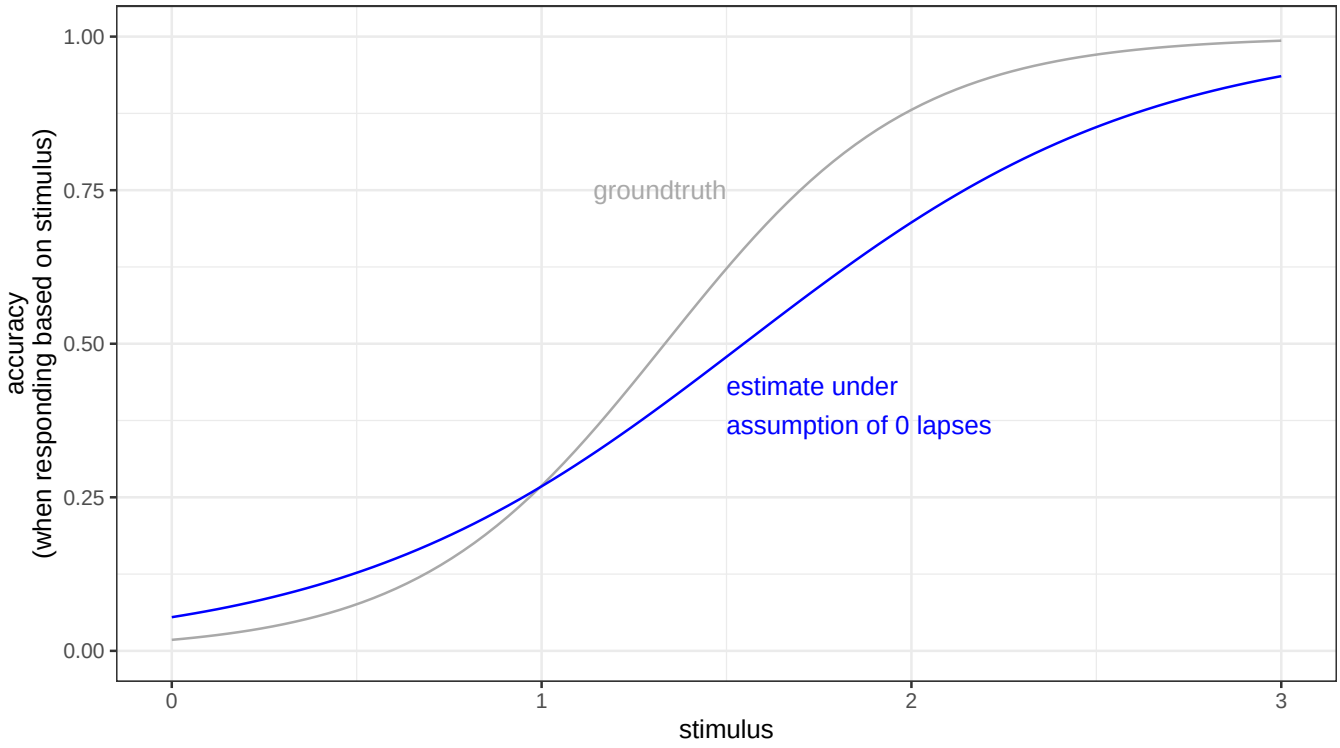
```
    trial = 1:n_per_stimulus) %>%
    mutate(
      s_stimulus = gelman_scale(stimulus),
      p = inv_logit_scaled(
        intercept + slope * stimulus,
        lb = gamma,
        ub = 1 - lambda),
      accuracy = rbinom(nrow(.), 1, p))
}

# 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 first plot shows a fit if the lapse rate is assumed to be zero. The next plot adds the ground truth functional relation, including the lapse rate. The third plot shows the resulting bias in the intercept and slope of the perceptual model when the λ is assumed to be 0 (blue) compared to the groundtruth of $\lambda = .05$ (gray).





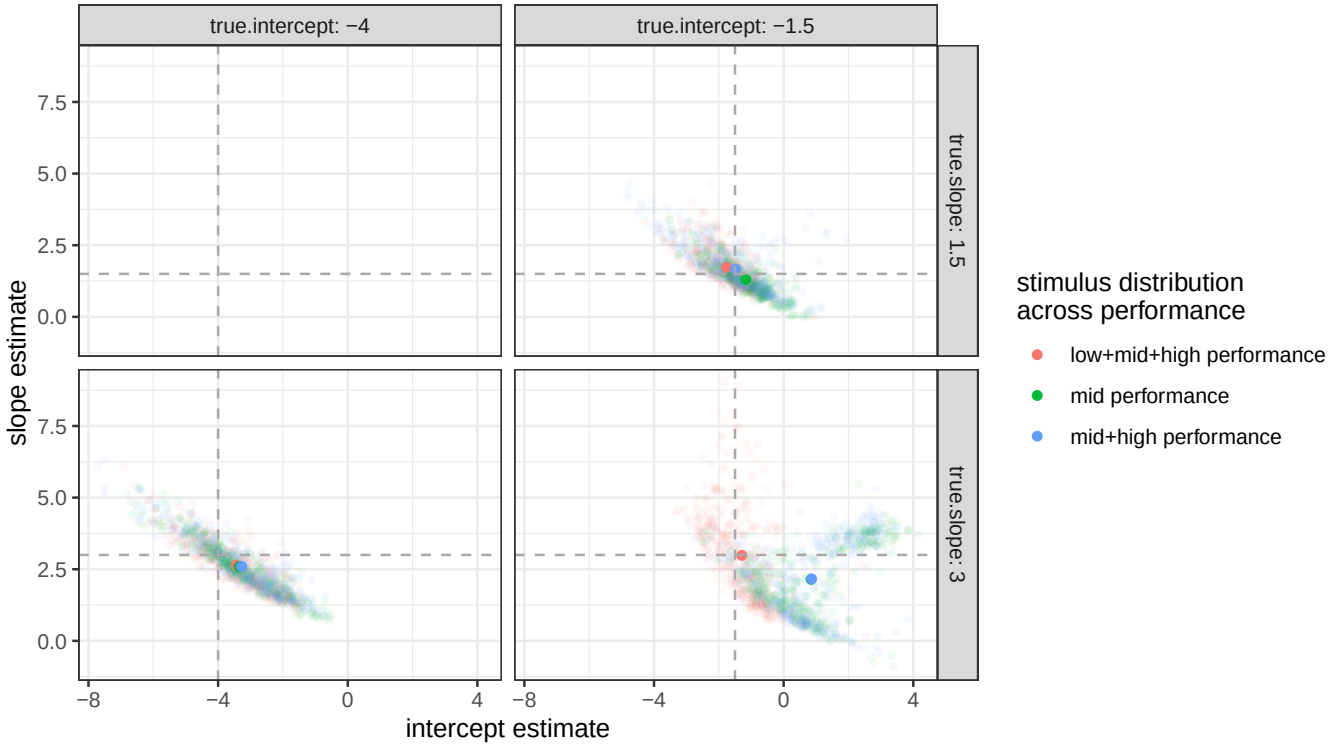

```
Start sampling
```



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). e will consider four different analyses, each assuming a different fixed lapse rate (0, .025, .05, .1).

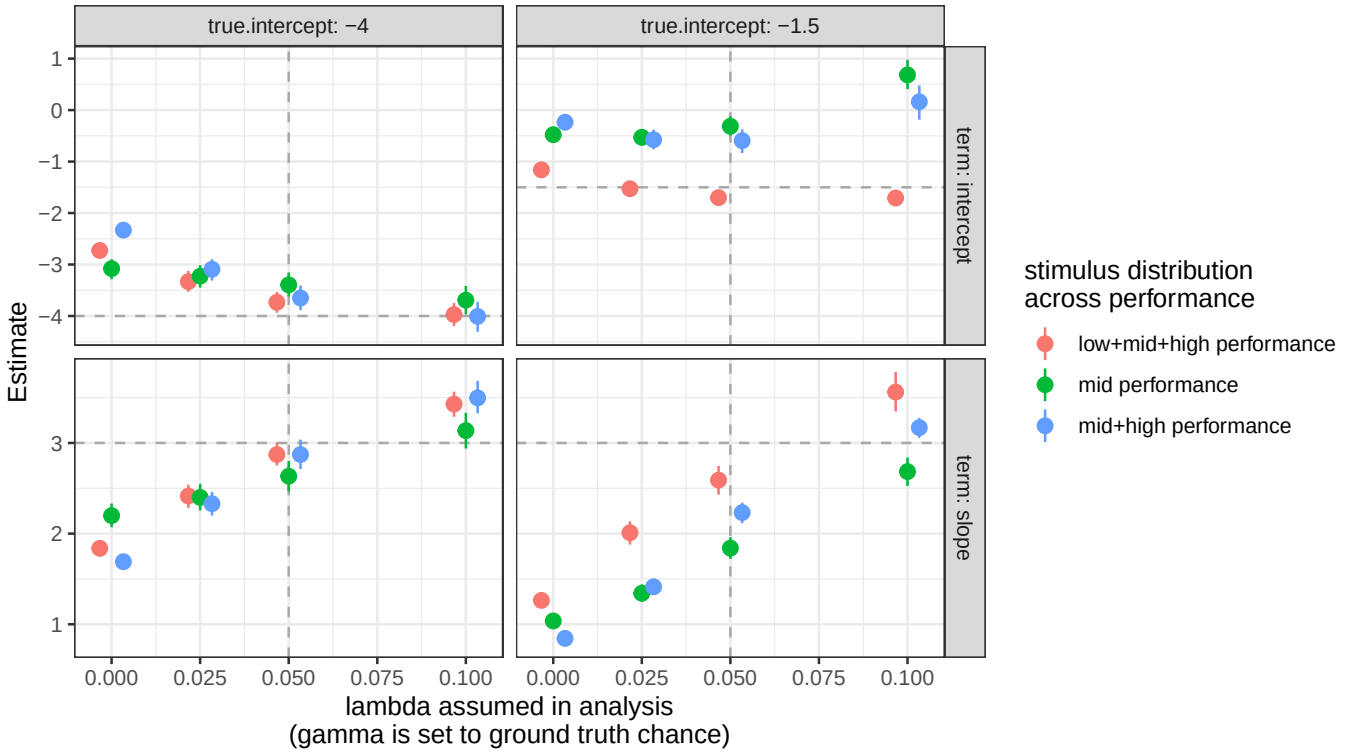
6.1.1 Visualizing the correlation among the intercept and slope

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



6.1.2 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).



6.2 Working with real data (unknown ground truth)

We fit a series of model to the data from Subject 3 from Ashley data. 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.

6.2.1 Assuming a fixed lapse rate

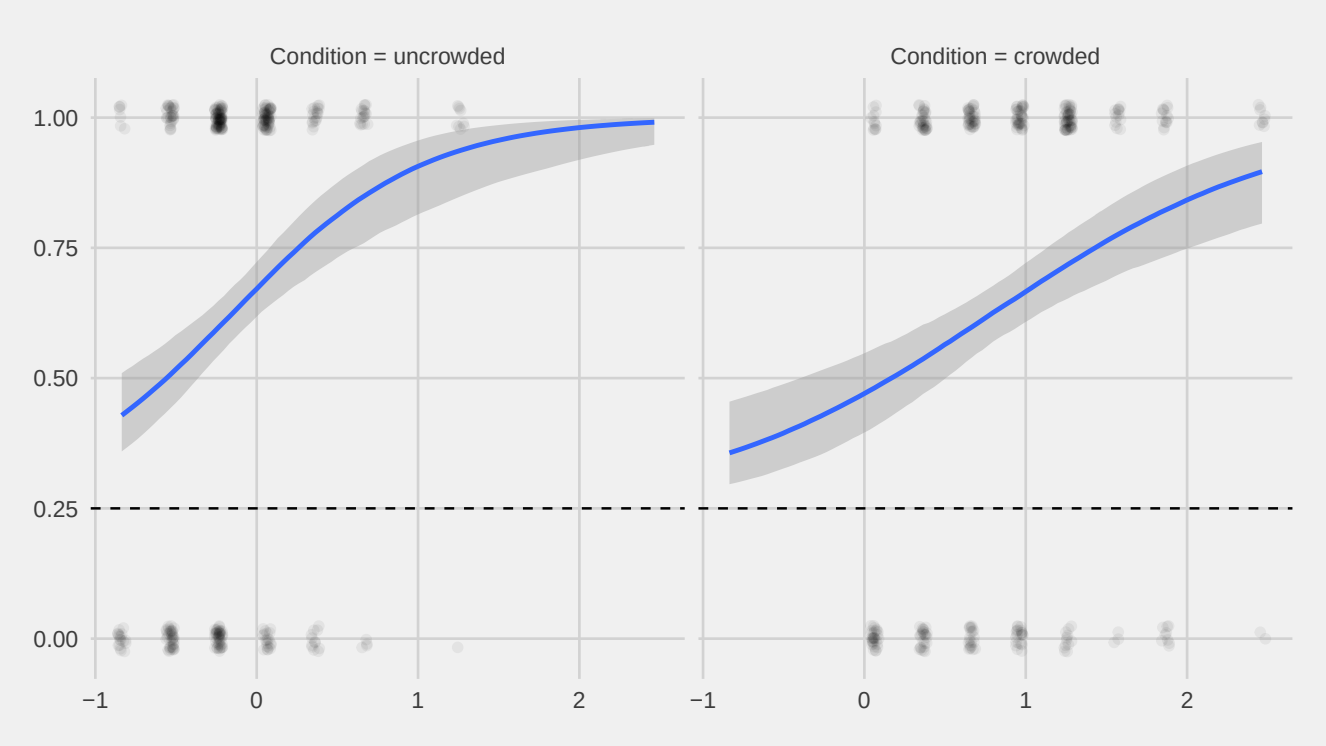
The first model assumes $\lambda = 0$ (no lapses):

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

my.priors <- c(
  prior(student_t(3, 0, 2.5), class = "b", nlpar = "eta")
)

fit.nl.s3.noLapse <- brm(
  BF,
  data = d %>%
    filter(Subject == "3"),
  control = list(adapt_delta = 0.999),
  iter = 4000,
  thin = 5,
  prior = my.priors,
  file = "../models/subject3-bias-noLapse"
)

my_hypotheses(fit.nl.s3.noLapse)
```

Hypothesis	Estimate	Est.Error	CI.Lower	CI.Upper	Evid.Ratio	Post.Prob	Star
(eta_Size) > 0	1.3972760	0.2184872	1.047113	1.7768233	Inf	1.0000	*

Hypothesis	Estimate	Est.Error	CI.Lower	CI.Upper	Evid.Ratio	Post.Prob	Star
(eta__ConditionCrowded.vs.Uncrowded) > 0	-1.1295060	0.2959469	-1.622108	-0.6517860	0.0000000	0.0000	
(eta__Size:ConditionCrowded.vs.Uncrowded) > 0	-0.5917648	0.4399998	-1.332263	0.1483214	0.0899183	0.0825	

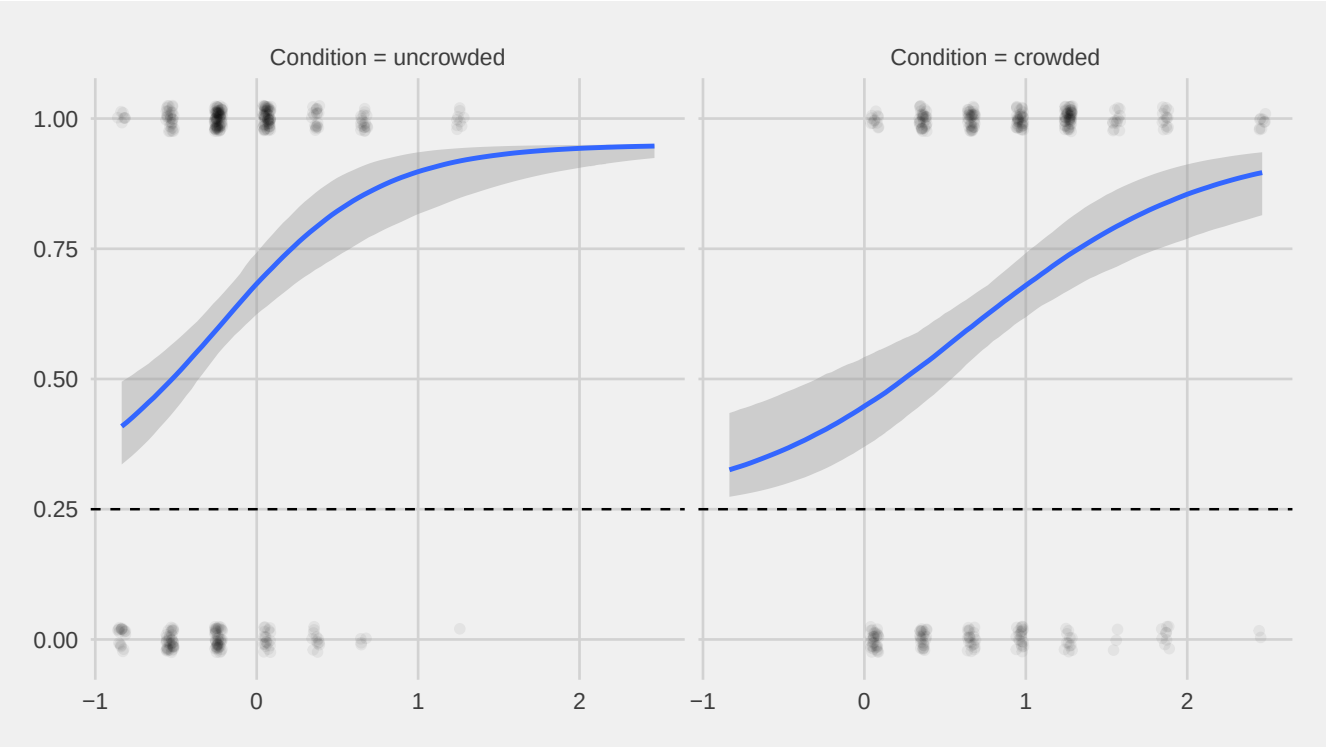
```
p = plot(
  conditional_effects(
    fit.nl.s3.noLapse,
    conditions = make_conditions(d, "Condition")),
  plot = F,
  points = T,
  point_args = list(alpha = .05, height = .025, width = .025),
  theme = theme_fivethirtyeight())[[1]]
p + p.common
```



The next two models assume non-zero lapse rates of $\lambda = .05$ and $\lambda = .2$, respectively:

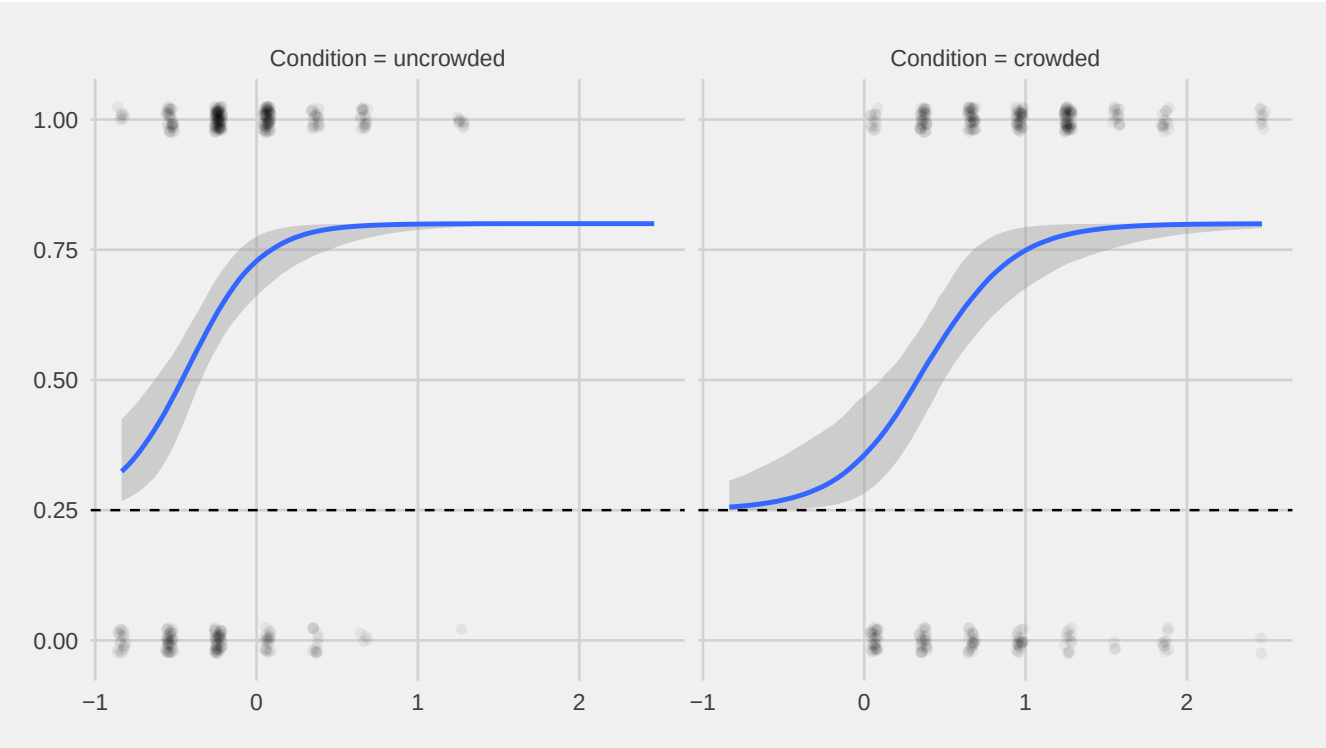
```
Warning in kable_pipe(x = structure(c("(Size) > 0", "(ConditionCrowded.vs.Uncrowded) > 0", : The table should
have a header (column names)
```

(Size) > 0	1.73089401699649	0.305125571995131	1.12517244118482	2.23427134567425	Inf	1	*
(ConditionCrowded.vs.Uncrowded) > 0	-1.42042742170828	0.372485369915146	-2.03915125941120	-0.818622086295474	-	0	
(Size:ConditionCrowded.vs.Uncrowded) > 0	0.577476367179234	0.308418608214790	0.149425287356302	1.63321277687844	213		



Warning in kable_pipe(x = structure(c("(Size) > 0", "(ConditionCrowded.vs.Uncrowded) > 0", : The table should have a header (column names)

(Size) > 0	4.26738491709795	0.978367280581025	4.36912750357719	6.0571353909586	Inf	1	*
(ConditionCrowded.vs.Uncrowded) > 0	0.830407299325215			-	0	0	
(Size:ConditionCrowded.vs.Uncrowded) > 0	3.43080655143919	1.52094732037026	4.942220220051	2.19805914188805	1.72985205830960	0.422222222222220	0.296875
	0.7302546002008		3.16587563055301				



6.2.2 Inferring the lapse rate from the data

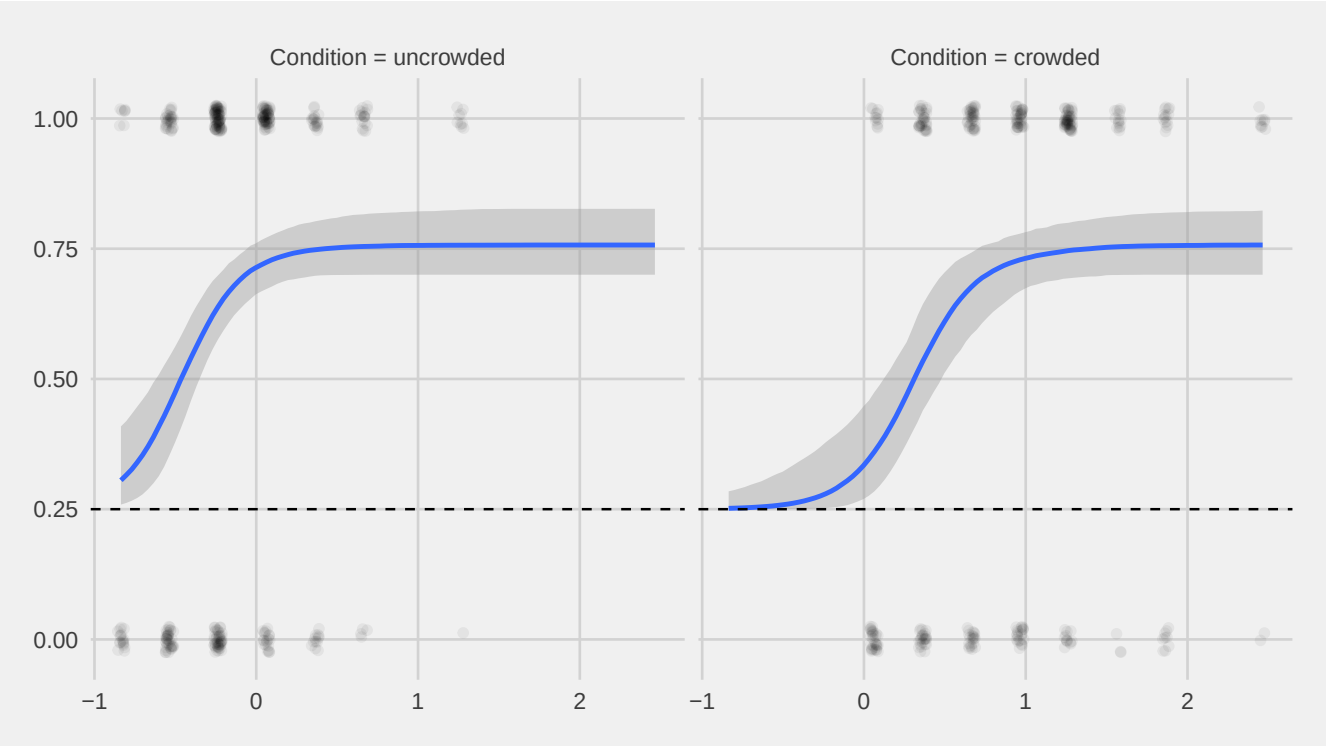
We infer the lapse rate from the data while assuming it's constant across conditions:

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

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

fit.nl.s3.Lapse <- brm(
  BF,
  data = d %>%
    filter(Subject == "3"),
  control = list(adapt_delta = 0.999),
  iter = 4000,
  thin = 5,
  prior = my.priors,
  file = "../models/subject3-lapse"
)

my_hypotheses(fit.nl.s3.Lapse)
```

```
Warning in kable_pipe(x = structure(c("(Size) > 0", "(ConditionCrowded.vs.Uncrowded) > 0", : The table should
have a header (column names)
```

(Size) > 0	5.539609908643551.71233108665925	3.26157760244318	8.63285533679322	Inf	1	*
(ConditionCrowded.vs.Uncrowded) > 0	1.259388011529 - 4.28941800198742	- 6.53213054822638	2.51779904607693	0	0	
(Size:ConditionCrowded.vs.Uncrowded) > 0	1.868613031298714 0.350891918218304	2.83136753550680 3.16102767048338	0.69491525423728841			

```
p = plot(
  conditional_effects(
    fit.nl.s3.Lapse,
    conditions = make_conditions(d, "Condition")),
  plot = F,
  points = T,
  point_args = list(alpha = .05, height = .025, width = .025),
  theme = theme_fivethirtyeight())[[1]]
p + p.common
```



6.3 Combining the data from all subjects

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). At this point, I am switching to the mixture formulation of the model as the NLMM formulation didn't converge (2 divergent transitions even with typical priors, thinning, etc.), and the mixture formulation is computationally more effective.

Now that we're using a bit more data (all 6500 or so trials), the model will take a moment to fit. On my machine it took about 30 minutes.

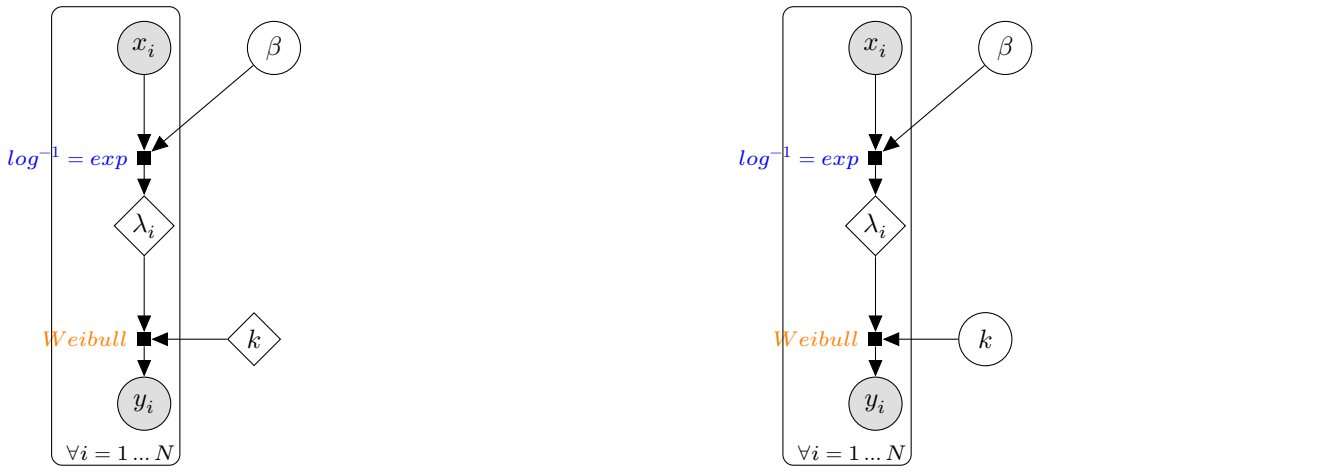
7 Weibull regression

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

9 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.

9.1 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 mode, as it allows us to orthogonalize γ and λ:

```
Warning in kable_pipe(x = structure(c("(Size) > 0", "(ConditionCrowded.vs.Uncrowded) > 0", : The table should
have a header (column names)
```

(Size) > 0	4.85176405734352	2.72005733144163	1.82677804324097	6.16855515524788	Inf	1 *
(ConditionCrowded.vs.Uncrowded) > 0	0.416946642655282	0.160568050871549	-2.28416968752824	0.957512916236385	0.0015022533800701	0.0015
(Size:ConditionCrowded.vs.Uncrowded) > 0	0.0201854509497087	0.00968361854332	-4.55868709999124	1.14534521732114	1.02942668696093	0.50725

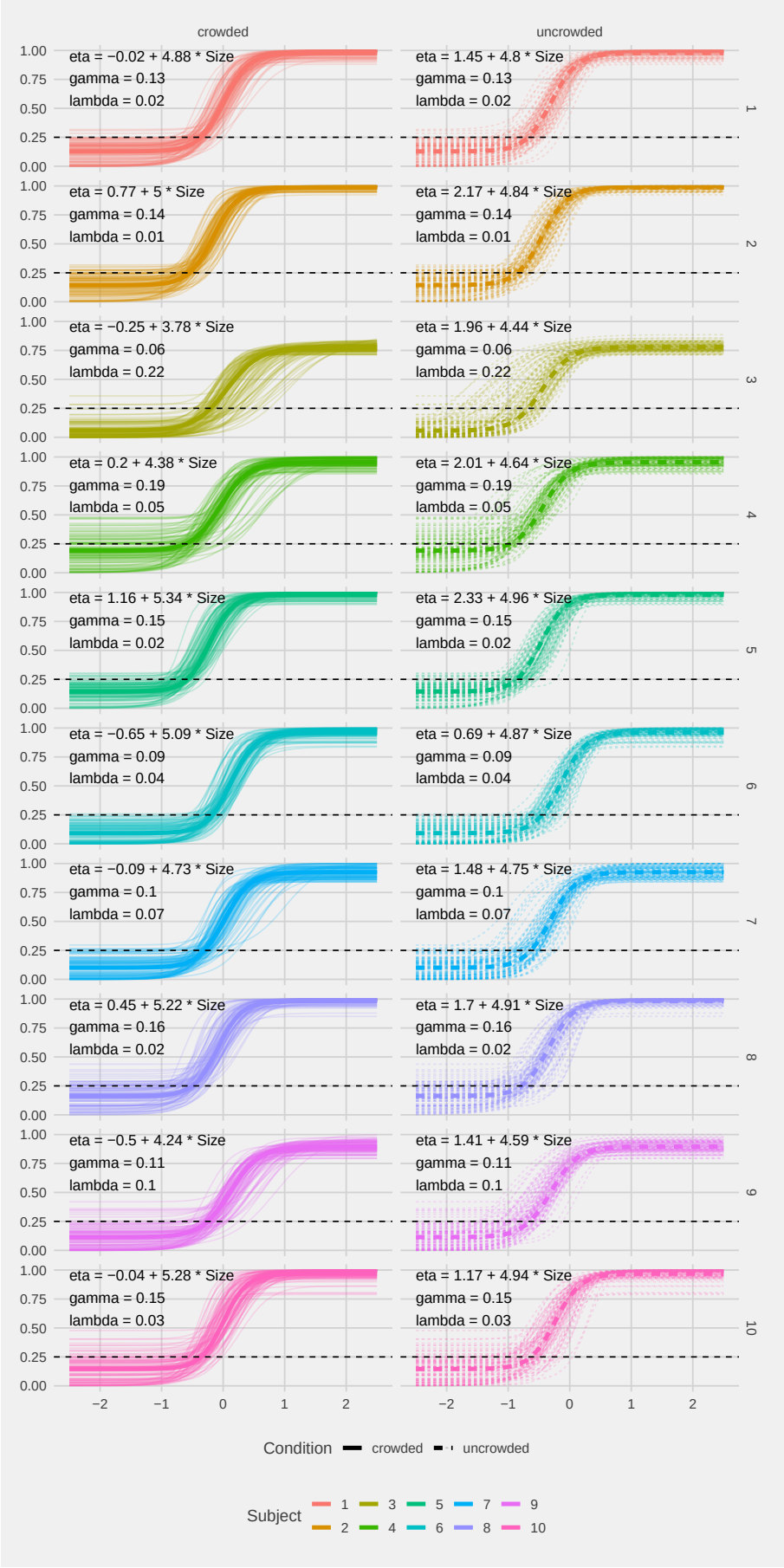
The following plots shows the predictions for each of 100 random posterior samples from each subjects (each curve corresponds to one posterior sample). The solid lines shows the predictions based on the *means* of the posterior samples. This line does not have to fall in the center of the other lines 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.

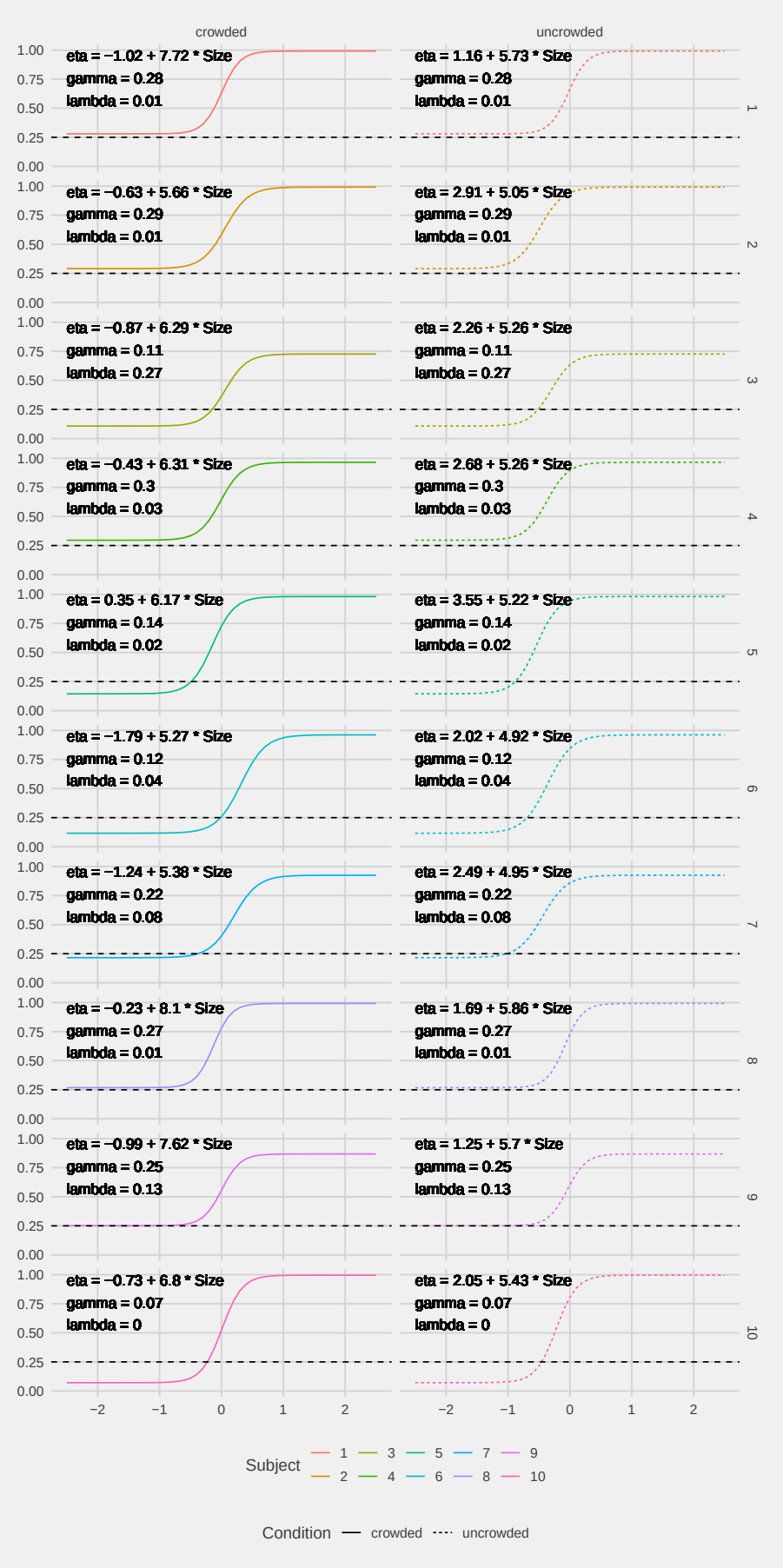
Distribution of lapse rate (mixture 1 weight), bias (mixture 2 mean), and mean of psychometric model (mixture 2):

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

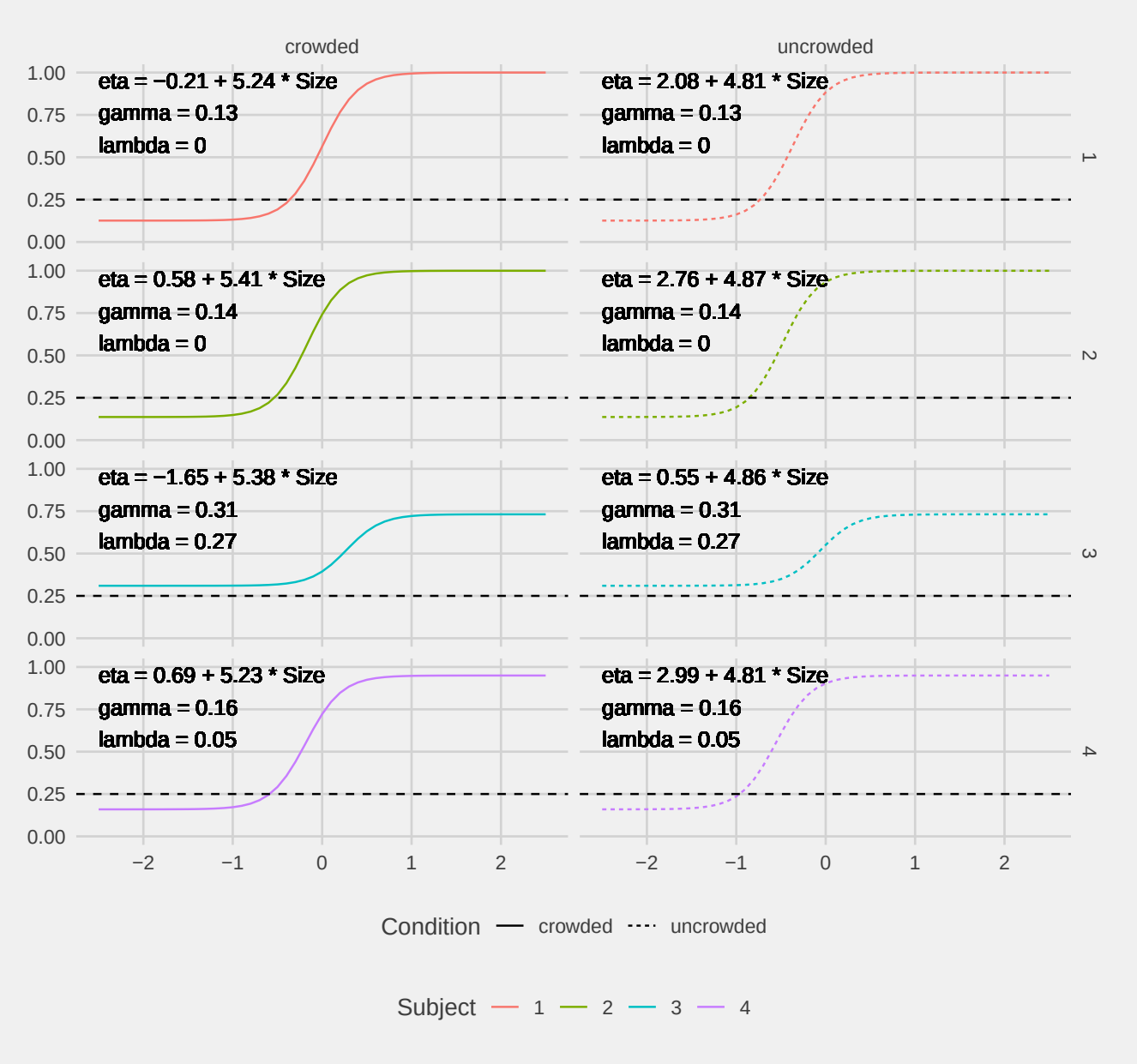
Lapse rate vs. mean and slope of psychometric model:

Visualizing predictions of the model:


crowded
uncrowded
eta = −0.02 + 4.88 * Size
gamma = 0.13
lambda = 0.02
eta = 1.45 + 4.8 * Size
gamma = 0.13
lambda = 0.02
eta = 0.77 + 5 * Size
gamma = 0.14
lambda = 0.01
eta = 2.17 + 4.84 * Size
gamma = 0.14
lambda = 0.01
eta = −0.25 + 3.78 * Size
gamma = 0.06
lambda = 0.22
eta = 1.96 + 4.44 * Size
gamma = 0.06
lambda = 0.22
eta = 0.2 + 4.38 * Size
gamma = 0.19
lambda = 0.05
eta = 2.01 + 4.64 * Size
gamma = 0.19
lambda = 0.05
eta = 1.16 + 5.34 * Size
gamma = 0.15
lambda = 0.02
eta = 2.33 + 4.96 * Size
gamma = 0.15
lambda = 0.02
eta = −0.65 + 5.09 * Size
gamma = 0.09
lambda = 0.04
eta = 0.69 + 4.87 * Size
gamma = 0.09
lambda = 0.04
eta = −0.09 + 4.73 * Size
gamma = 0.1
lambda = 0.07
eta = 1.48 + 4.75 * Size
gamma = 0.1
lambda = 0.07
eta = 0.45 + 5.22 * Size
gamma = 0.16
lambda = 0.02
eta = 1.7 + 4.91 * Size
gamma = 0.16
lambda = 0.02
eta = −0.5 + 4.24 * Size
gamma = 0.11
lambda = 0.1
eta = 1.41 + 4.59 * Size
gamma = 0.11
lambda = 0.1
eta = −0.04 + 5.28 * Size
gamma = 0.15
lambda = 0.03
eta = 1.17 + 4.94 * Size
gamma = 0.15
lambda = 0.03
1.00
0.75
0.50
0.25
0.00
−2
−1
0
1
2
1
2
3
4
5
6
7
8
9
10
Condition
crowded
uncrowded
Subject
1
2
3
4
5
6
7
8
9
10


crowded
uncrowded
eta = −1.02 + 7.72 * Size
gamma = 0.28
lambda = 0.01
eta = 1.16 + 5.73 * Size
gamma = 0.28
lambda = 0.01
eta = −0.63 + 5.66 * Size
gamma = 0.29
lambda = 0.01
eta = 2.91 + 5.05 * Size
gamma = 0.29
lambda = 0.01
eta = −0.87 + 6.29 * Size
gamma = 0.11
lambda = 0.27
eta = 2.26 + 5.26 * Size
gamma = 0.11
lambda = 0.27
eta = −0.43 + 6.31 * Size
gamma = 0.3
lambda = 0.03
eta = 2.68 + 5.26 * Size
gamma = 0.3
lambda = 0.03
eta = 0.35 + 6.17 * Size
gamma = 0.14
lambda = 0.02
eta = 3.55 + 5.22 * Size
gamma = 0.14
lambda = 0.02
eta = −1.79 + 5.27 * Size
gamma = 0.12
lambda = 0.04
eta = 2.02 + 4.92 * Size
gamma = 0.12
lambda = 0.04
eta = −1.24 + 5.38 * Size
gamma = 0.22
lambda = 0.08
eta = 2.49 + 4.95 * Size
gamma = 0.22
lambda = 0.08
eta = −0.23 + 8.1 * Size
gamma = 0.27
lambda = 0.01
eta = 1.69 + 5.86 * Size
gamma = 0.27
lambda = 0.01
eta = −0.99 + 7.62 * Size
gamma = 0.25
lambda = 0.13
eta = 1.25 + 5.7 * Size
gamma = 0.25
lambda = 0.13
eta = −0.73 + 6.8 * Size
gamma = 0.07
lambda = 0
eta = 2.05 + 5.43 * Size
gamma = 0.07
lambda = 0
1.00
0.75
0.50
0.25
0.00
−2
−1
0
1
2
1
2
3
4
5
6
7
8
9
10
Subject
1 2 3 4 5 6 7 8 9 10
Condition
crowded
uncrowded

Zooming in on Subject 1-4:



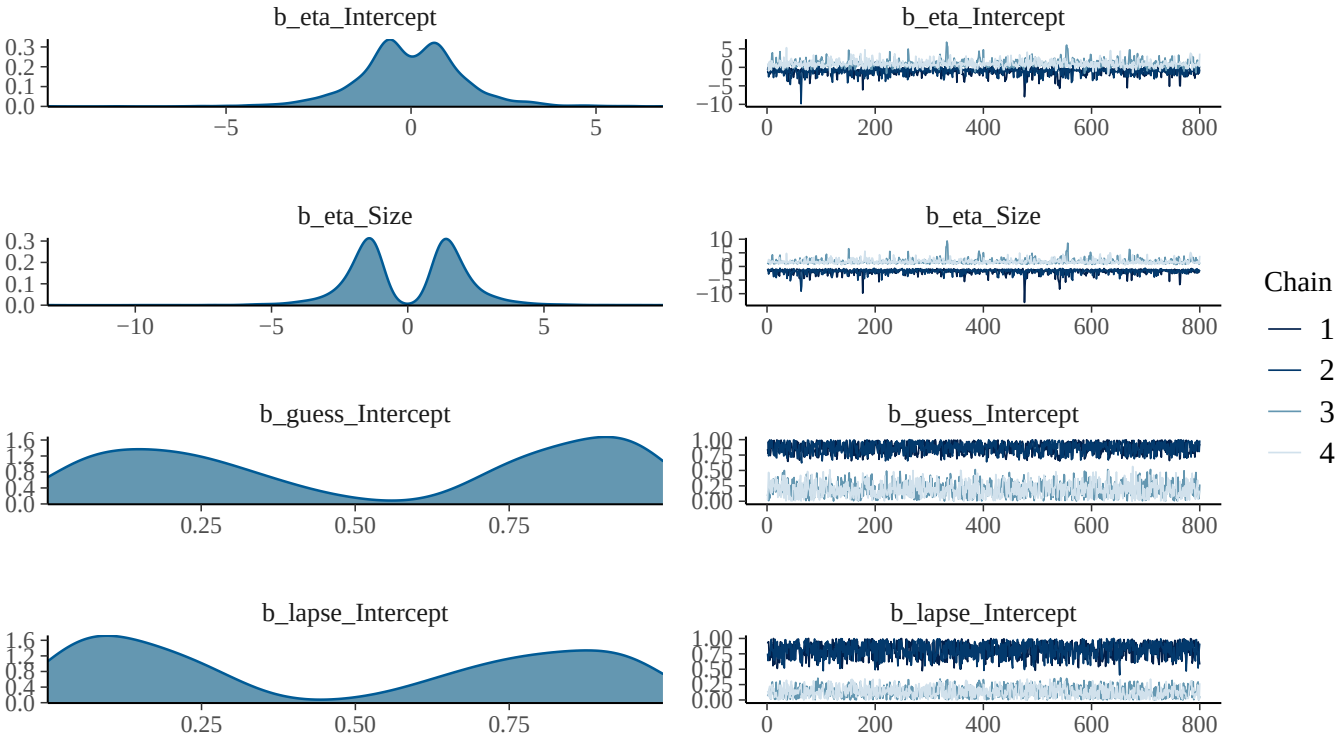
9.2 NLM formulation of the lasping logistic model

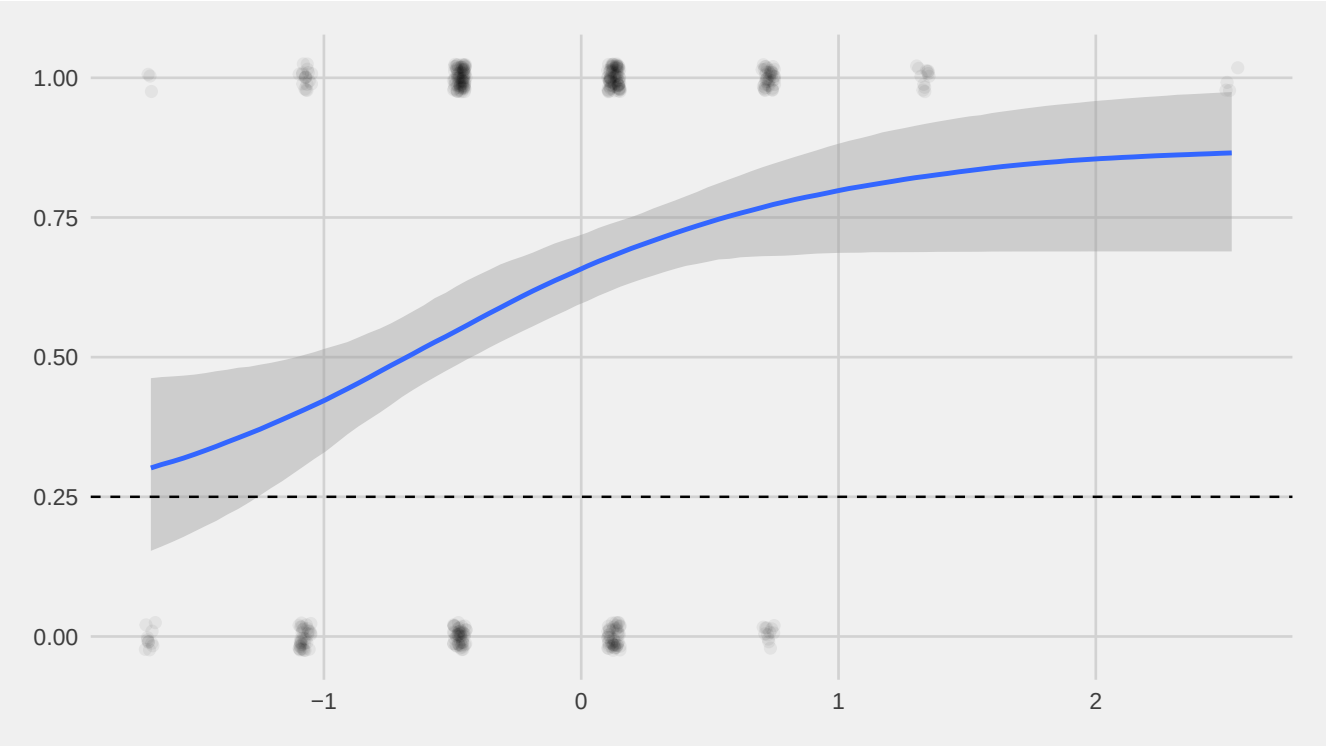
A less computationally efficient (or practical) way of inferring both γ and λ uses the NLM formulation used above when inferring only λ. This is illustrated here for a single subject (Subject 1):

```
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 l-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; symmetrica around .5 for the guess and lapse rate, which are expressed on the scale of proportions).

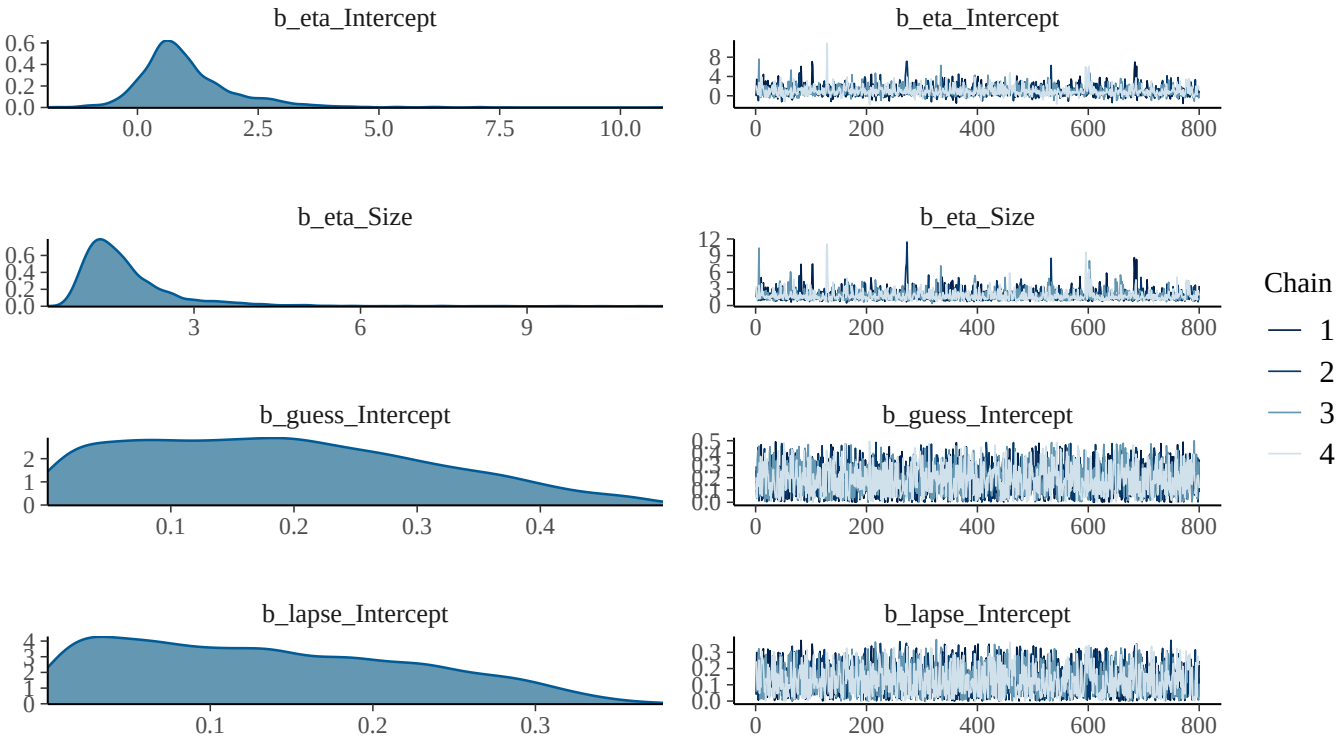
We refit the model while constraining the guess and lapse rate to be less than .5, i.e., less than half of the trials are attentional lapses. This is a very weak assumption.

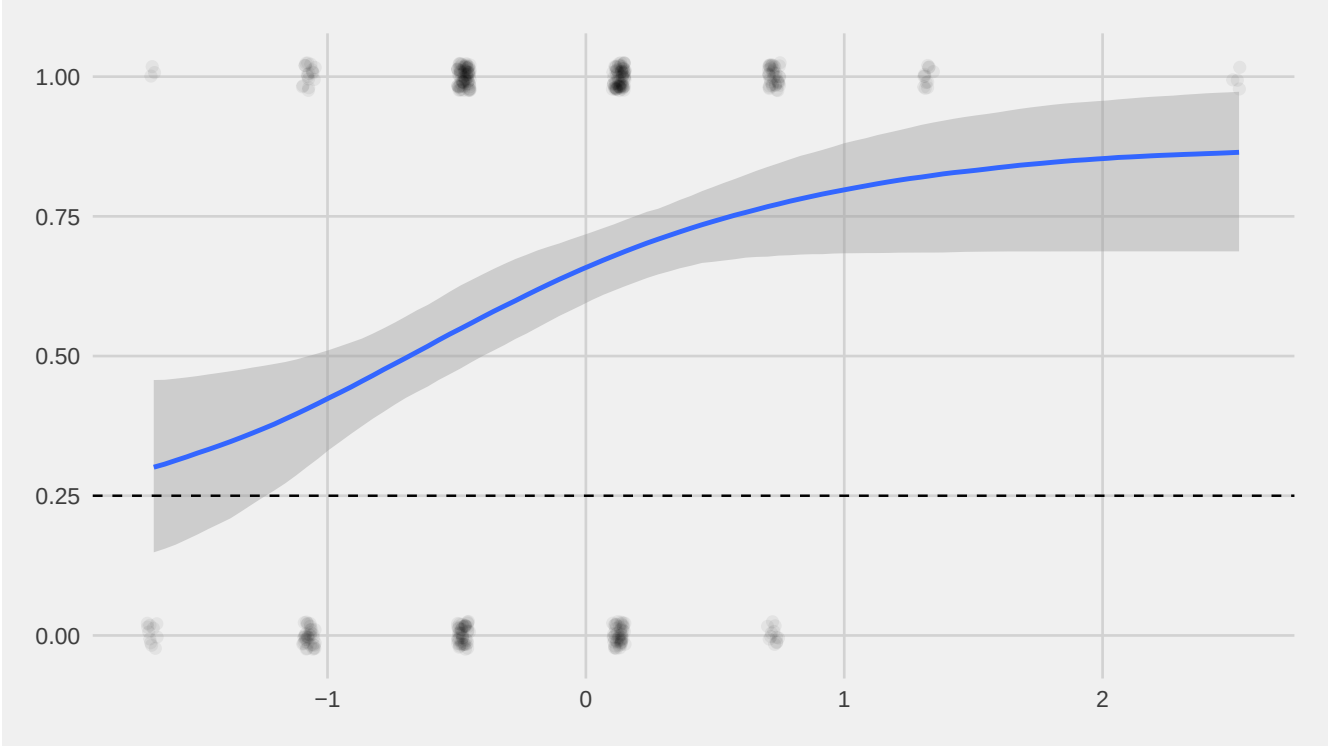
```
my.priors <- c(
  prior(student_t(3, 0, 2.5), class = "b", nlpar = "eta"),
  prior(beta(1, 1), nlpar = "lambda", lb = 0, ub = .5),
  prior(beta(1, 1), nlpar = "gamma", lb = 0, ub = .5)
)
```

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).
```





9.3 NLMM formulation of the lasping mixed-effect logistic model

An example of the NLMM specification of the mixed-effects logistic model, but without crowdedness condition or its interaction with letter size since even this simplified model did not converge. (I first took this approach and then switched to the mixture formulation presented above.)

```
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 + (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
```

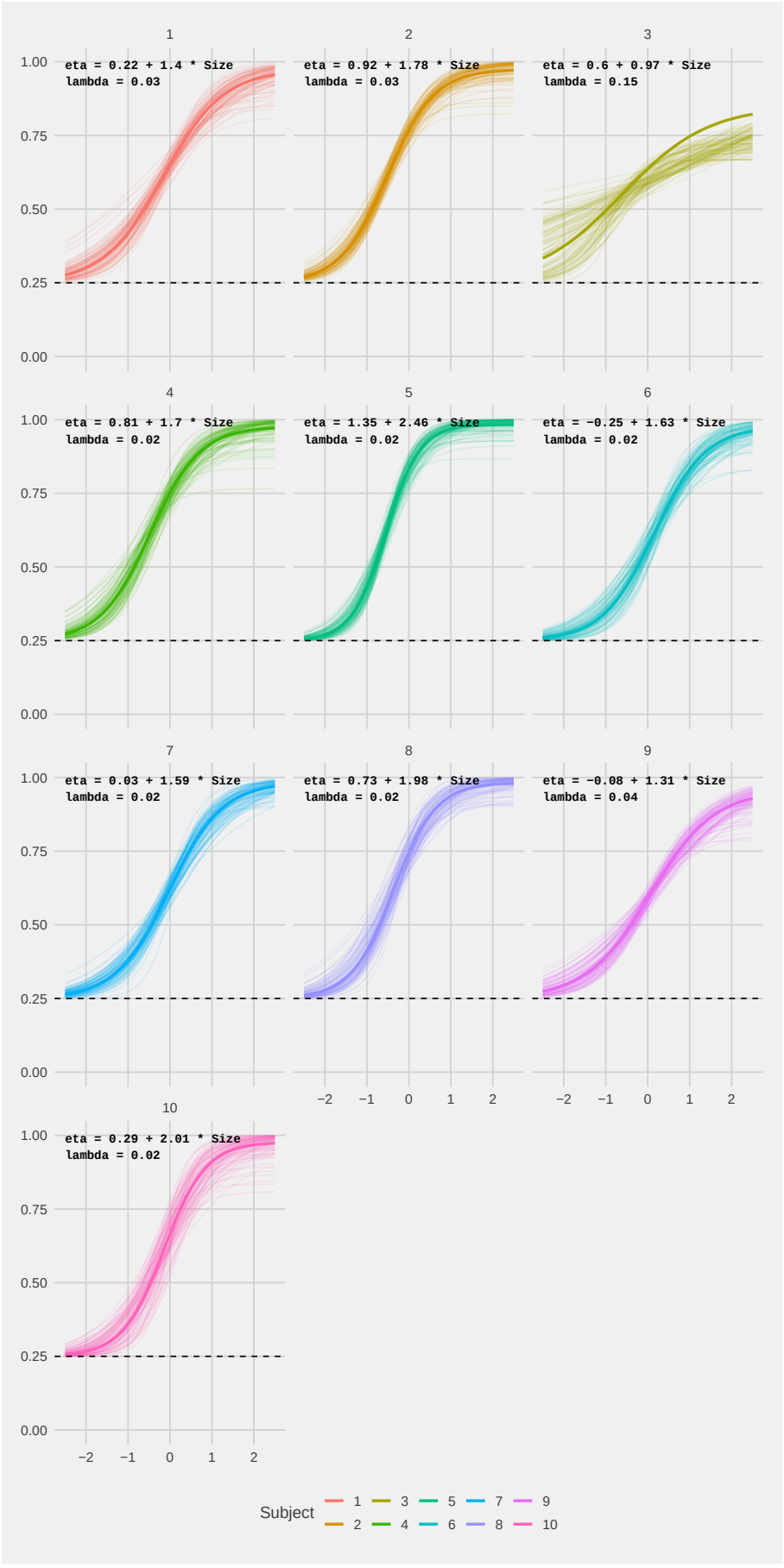
```
 Family: bernoulli
  Links: mu = identity
Formula: ResponseCorrect ~ 0.25 + (0.75 - inv_logit(lapse)) * inv_logit(eta)
         eta ~ 1 + Size + (1 + Size | Subject)
         lapse ~ 1 + (1 | Subject)
   Data: d (Number of observations: 3949)
Samples: 4 chains, each with iter = 8000; warmup = 4000;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

Group-Level Effects:
~Subject (Number of levels: 10)
                           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(eta_Intercept)              0.71      0.24     0.38     1.33 1.00     2791     2908
sd(eta_Size)                   0.64      0.30     0.11     1.32 1.00     1547     1956
sd(lapse_Intercept)            1.98      1.79     0.09     6.88 1.00     2008     2735
cor(eta_Intercept,eta_Size)    0.49      0.35    -0.37     0.94 1.00     2609     2630

Population-Level Effects:
               Estimate Est.Error l-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).
```

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)
```


1
eta = 0.22 + 1.4 * Size
lambda = 0.03
2
eta = 0.92 + 1.78 * Size
lambda = 0.03
3
eta = 0.6 + 0.97 * Size
lambda = 0.15
4
eta = 0.81 + 1.7 * Size
lambda = 0.02
5
eta = 1.35 + 2.46 * Size
lambda = 0.02
6
eta = −0.25 + 1.63 * Size
lambda = 0.02
7
eta = 0.03 + 1.59 * Size
lambda = 0.02
8
eta = 0.73 + 1.98 * Size
lambda = 0.02
9
eta = −0.08 + 1.31 * Size
lambda = 0.04
10
eta = 0.29 + 2.01 * Size
lambda = 0.02
1.00
0.75
0.50
0.25
0.00
−2
−1
0
1
2
Subject
1
2
3
4
5
6
7
8
9
10

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:

```
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 specifed:
  Setting the mode to markers
  Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```

10 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     2020-12-15

- 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)
 Brobdingnag      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)
e1071            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)
fansi            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)
ggthemes       * 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
```