Psychometric models

A brief applied overview

T. Florian Jaeger

November 2, 2020

Contents

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 17] (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 ...
 - 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 is essentially a mixture model with two components. One component describes the responses that depend on the stimulus (e.g,. the logistic model discussed above), as a function of 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. The second component if the mixture model describes responses that are *in*dependent of the stimulus (i.e., the bias term γ). The weight of the second component is the lapse rate (λ):

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

which is also sometimes written as

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

where $p(correctresponse|stimulus, \alpha, \beta)$ could be, for example, a logistic GLM, or any of the other common psychometric functions. The resulting model is *not* a GLM anymore (but it can still be fit with maximum likelihood fitting). **To develop an intuition** about what the lapse rate (λ) and bias parameters (γ) do, check out 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.

5 Applying a lapsing logistic GLMM to Ashley's data

Were is an example in R using the library `brms` for Bayesian regression modeling, which—among many other things—provides efficient fitting of 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 Case study: understanding the consequences of assumptions about attentional lapses

As a first illustration, here is the formula describing this model 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 ~ guess + (1-guess-lapse) * inv_logit(eta),
  eta ~ 1 + Size,
  guess ~ 1,
  lapse ~ 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 lapse rate and bias parameters, but we first fit this model without any constraint on the lapse rate and bias:

```
# Define weakly regularizing prior, including lower (lb) and upper bounds (ub)
# for the lapse and guess parameters
my.priors <- c(
  prior(student_t(3, 0, 2.5), class = "b", nlpar = "eta"),
  prior(beta(1, 1), nlpar = "lapse", lb = 0, ub = 1),
  prior(beta(1, 1), nlpar = "guess", lb = 0, ub = 1)
)
```

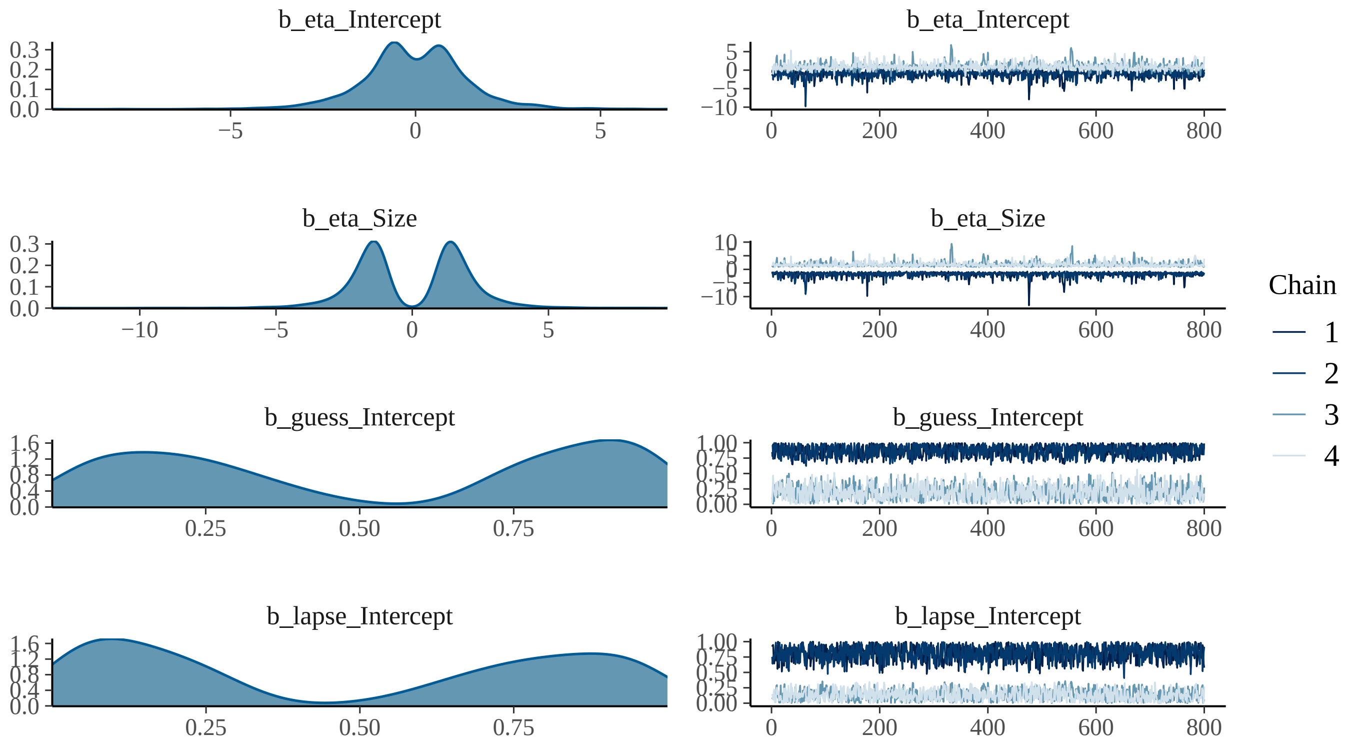
So that the priors are on the right 'scale', we follow (Gelman et al., 2008), and standardize the continuous predictors in the model. As is typical for analyses of experimental data with two balanced conditions, we deviation/sum/abova/effect-code the condition variable, with the 'treatment' receiving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

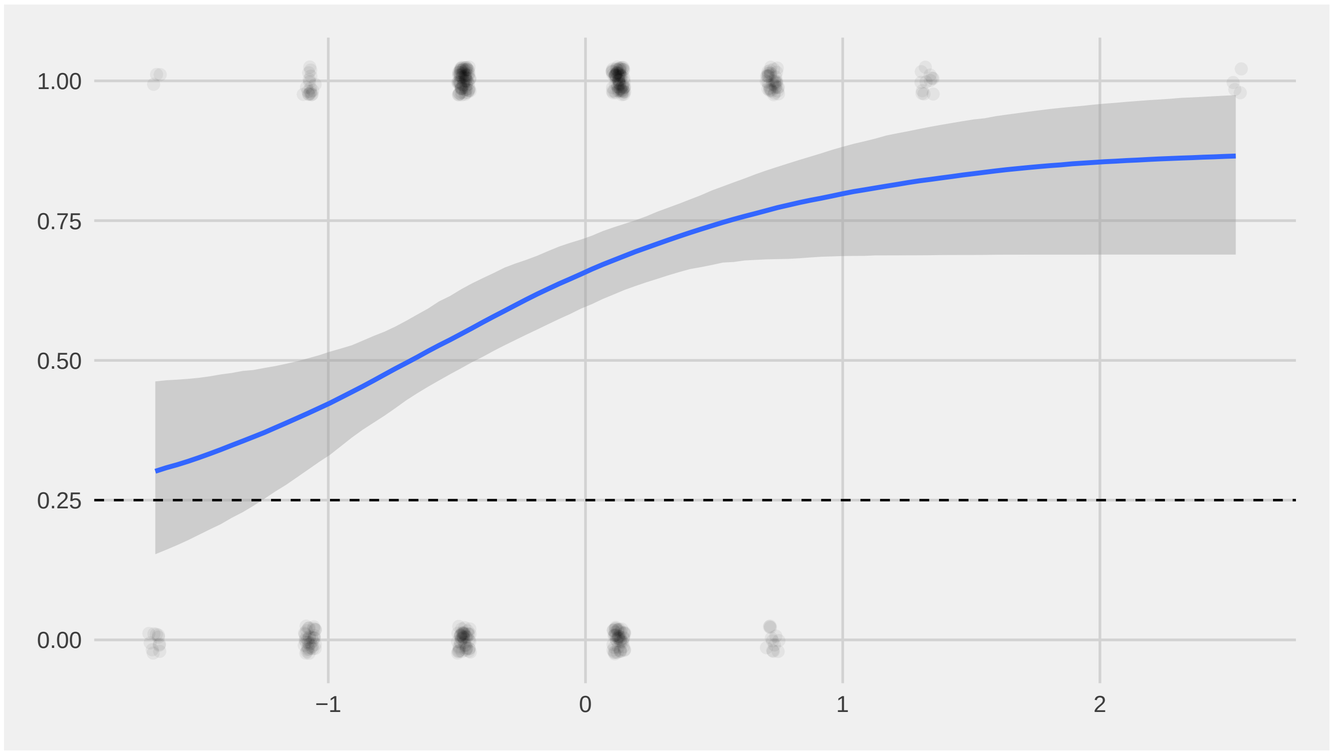
```
# We store the mean and sd of Size so that we can later transform the model's predictions back
# onto the scale of original Size predictor
Size.mu = mean(d$Size)
Size.sd = sd(d$Size)

# 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)) / sd(Size))

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

Let's fit this model to the data from Subject 1.

```
Warning: Parts of the model have not converged (some Rhats are > 1.05). Be careful when analysing the results! We recommend running

 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).

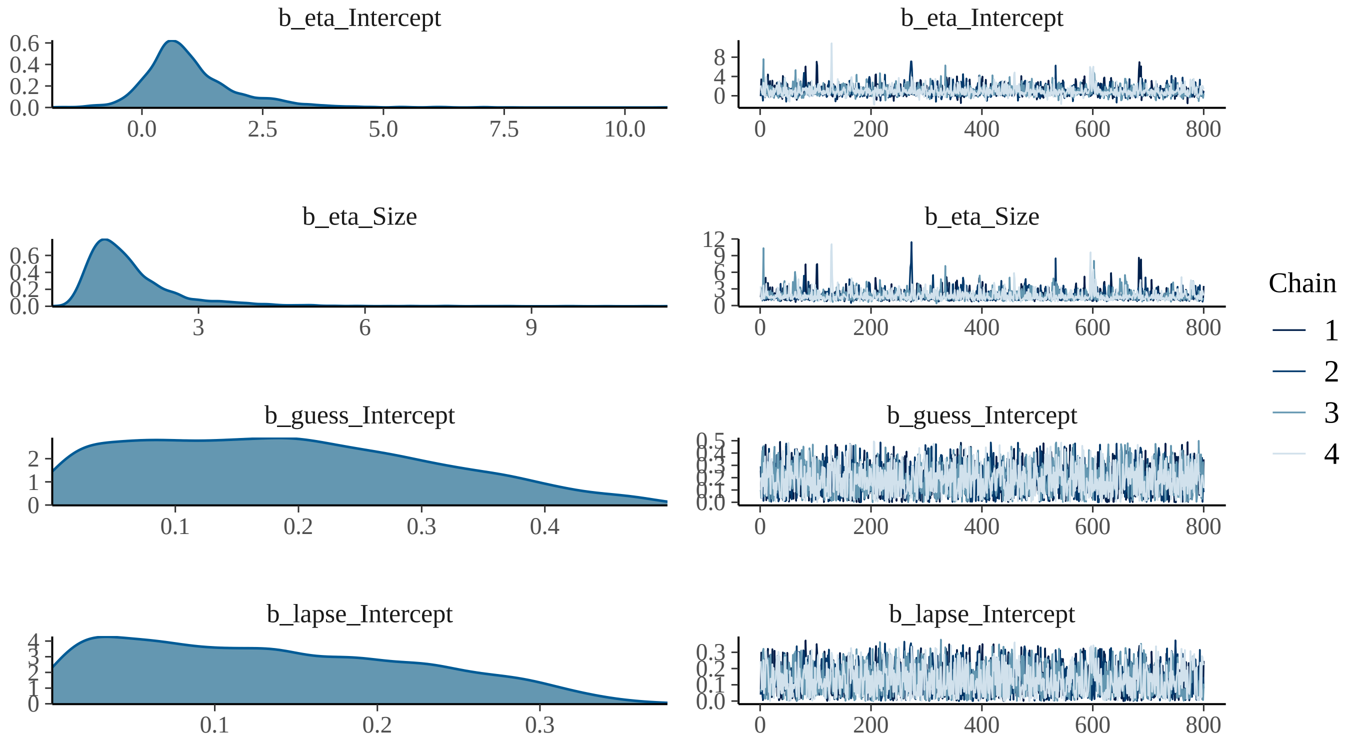
We refit the model while constraining the guess and lapse rate to be less than .5, i.e., less than half of the trials are attentional lapses. This is a very weak assumption.

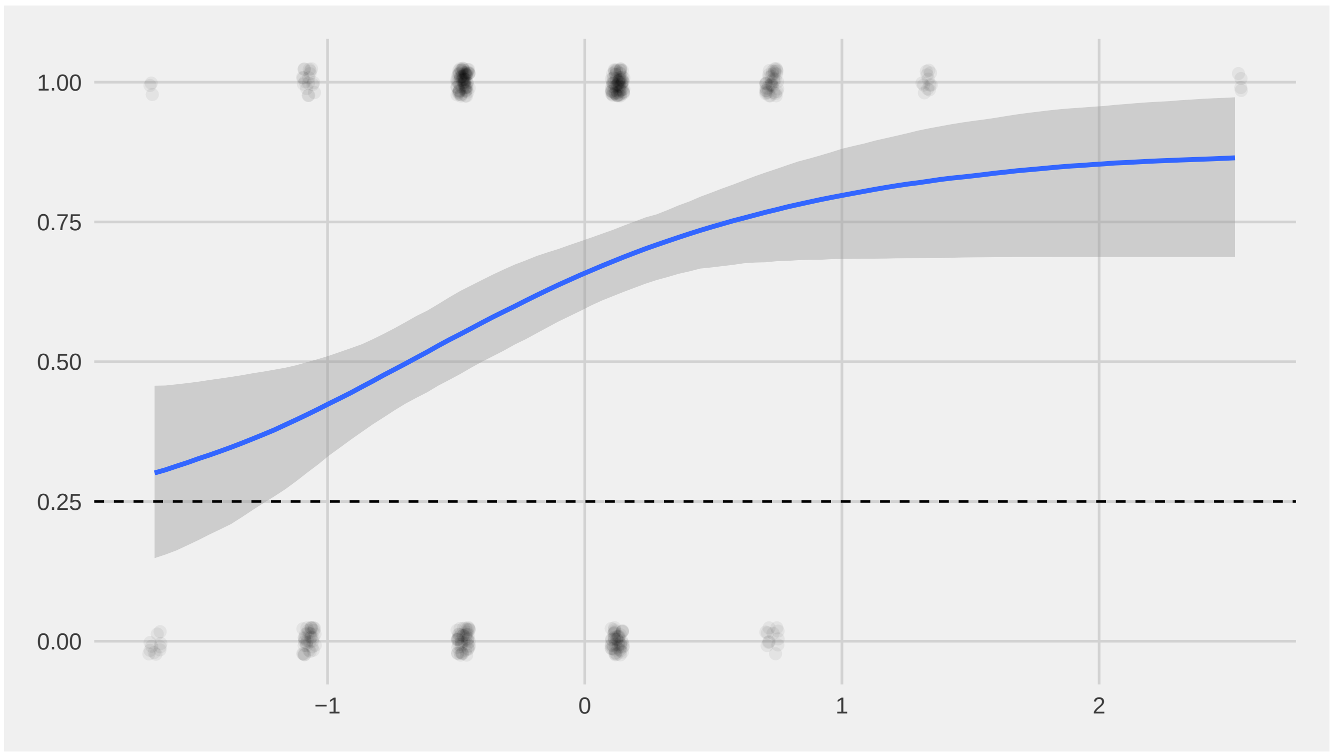
```
my.priors <- c(
  prior(student_t(3, 0, 2.5), class = "b", nlpar = "eta"),
  prior(beta(1, 1), nlpar = "lapse", lb = 0, ub = .5),
  prior(beta(1, 1), nlpar = "guess", lb = 0, ub = .5)
)
```

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).
```

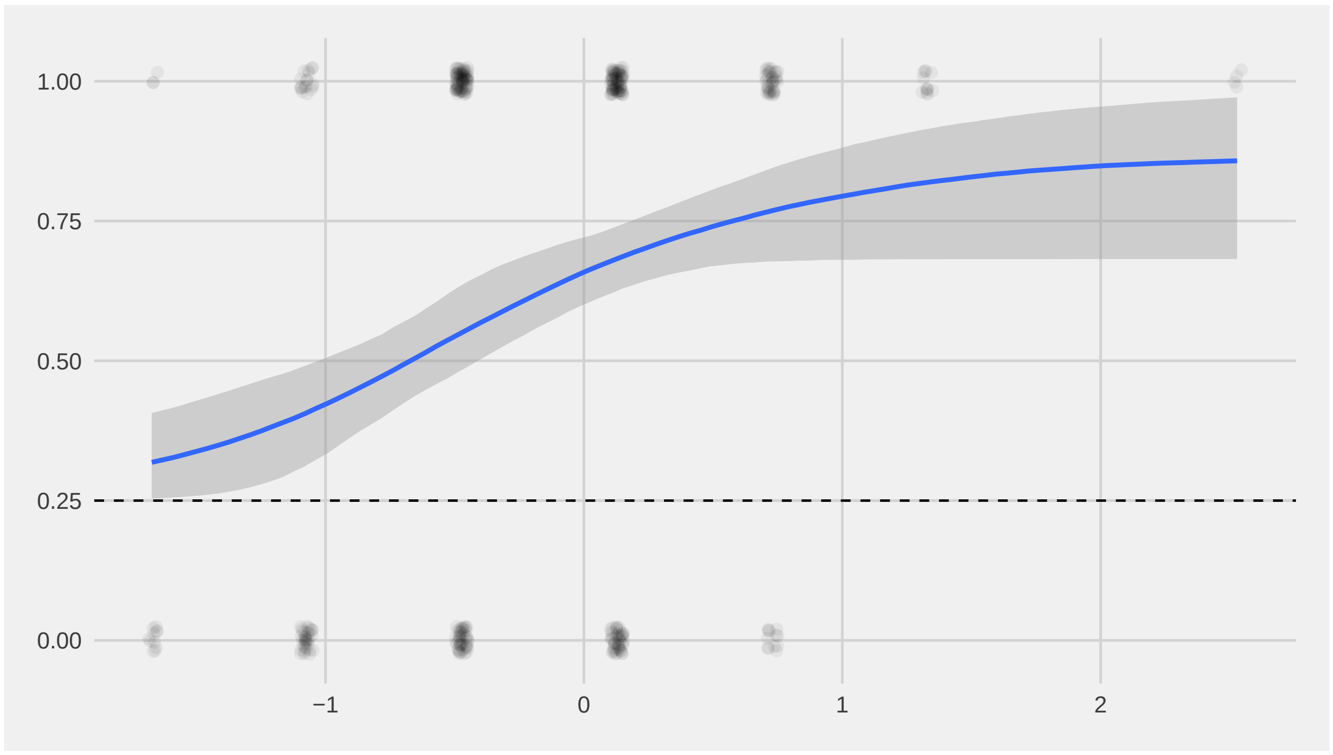




What is often done in this type of research is to fix the bias term to the chance level, which is known (here it is .25). Under this simplifying assumption, we can thus rewrite and refit the model. This increases the certainty about the lapse rate parameter, but still leaves substantial uncertainty. 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.

```
BF.lapse <- brmsformula(
  ResponseCorrect ~ .25 + (.75-lapse) * inv_logit(eta),
  eta ~ 1 + Size,
  lapse ~ 1,
  family = bernoulli(link="identity"),
  nl = TRUE
)

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)
)
```

```
 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).
```



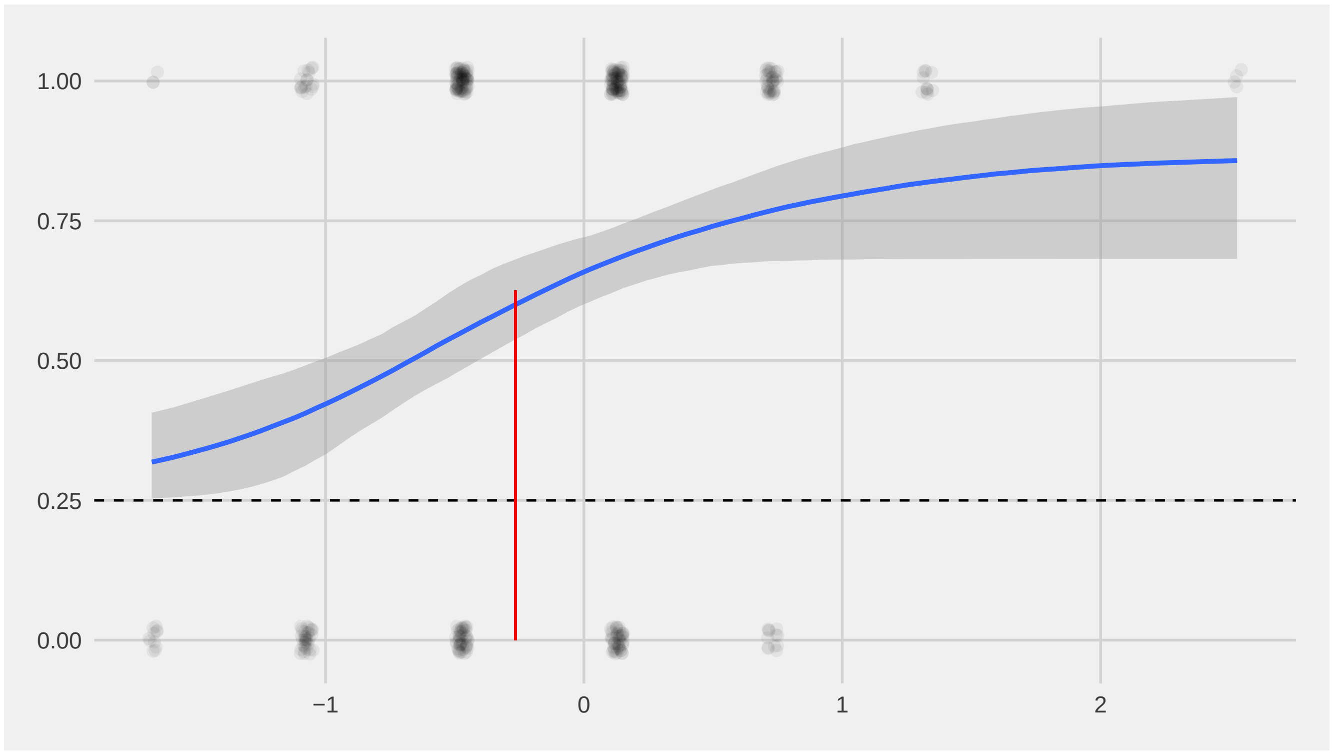
5.2 Determining the subject-specific threshold

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.

$$logit(threshold) = \alpha + \beta * Size \tag{3}$$

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

$$\begin{aligned} logit(.625) &= \alpha + \beta * Size \Rightarrow & (4)\\ 0.5108256 &= 1.04 + 2.00 * Size \Leftrightarrow & (5)\\ -0.2645872 &= Size & (6) \end{aligned}$$



5.3 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.

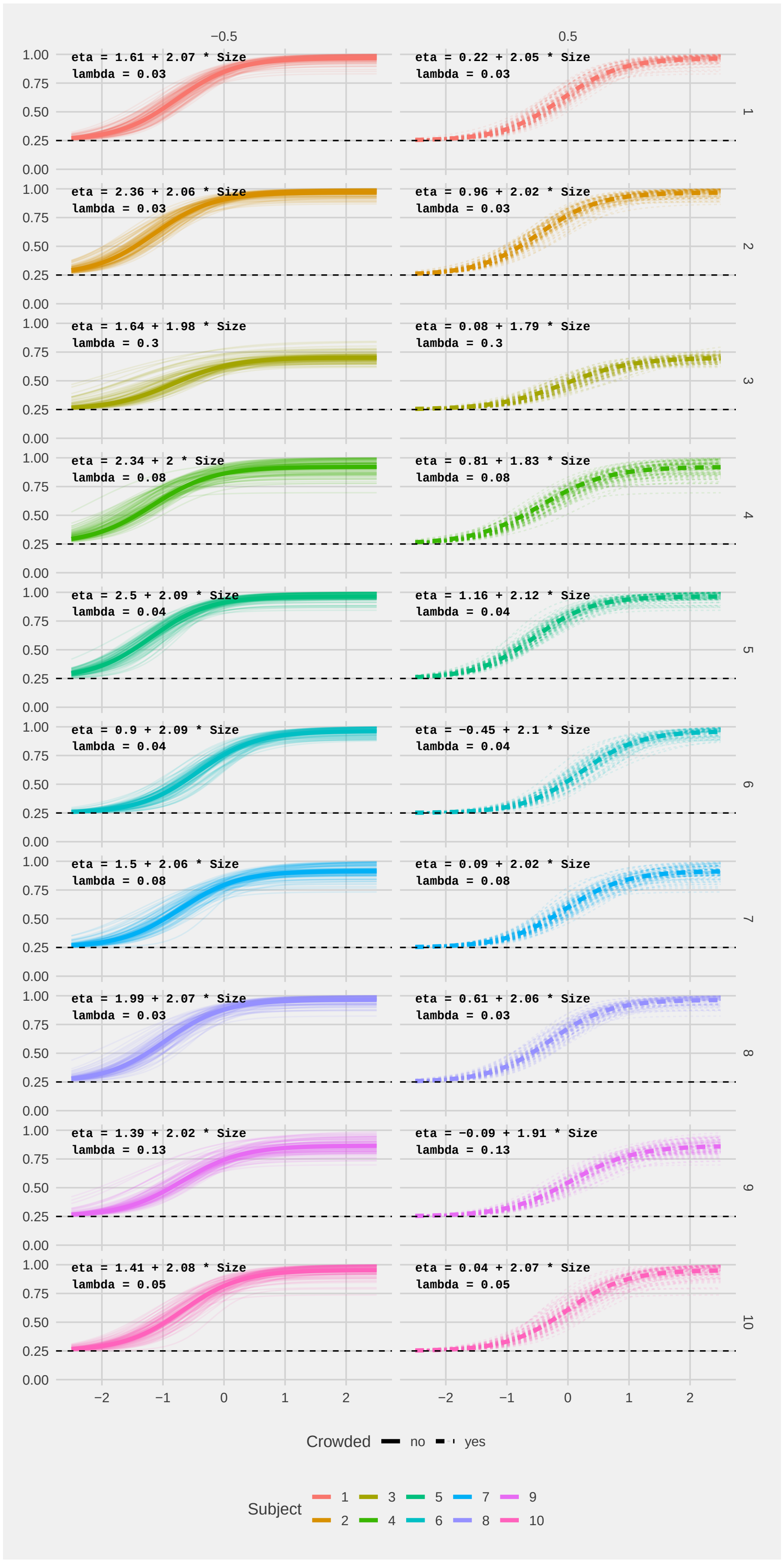
```
 Family: mixture(bernoulli, bernoulli)
  Links: mu1 = logit; mu2 = logit; theta1 = identity; theta2 = identity
Formula: ResponseCorrect ~ 1
         mu1 ~ 0 + offset(qlogis(0.25))
         mu2 ~ 1 + Size * Condition + (1 + Size * Condition | Subject)
         theta1 ~ 1 + (1 | Subject)
   Data: d (Number of observations: 3949)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Group-Level Effects:
~Subject (Number of levels: 10)
                                                                           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Ta
sd(mu2_Intercept)                                                              0.65      0.20     0.37     1.15 1.00     1849
sd(mu2_Size)                                                                   0.22      0.17     0.01     0.65 1.00     1347
sd(mu2_ConditionCrowded.vs.Uncrowded)                                          0.82      0.31     0.35     1.54 1.00     1552
sd(mu2_Size:ConditionCrowded.vs.Uncrowded)                                     0.34      0.26     0.01     0.96 1.00     2190
sd(theta1_Intercept)                                                           1.85      1.01     0.68     4.43 1.00     2520
cor(mu2_Intercept,mu2_Size)                                                    0.02      0.42    -0.77     0.78 1.00     5185
cor(mu2_Intercept,mu2_ConditionCrowded.vs.Uncrowded)                           0.16      0.32    -0.47     0.74 1.00     3166
cor(mu2_Size,mu2_ConditionCrowded.vs.Uncrowded)                                0.02      0.43    -0.80     0.81 1.00      949
cor(mu2_Intercept,mu2_Size:ConditionCrowded.vs.Uncrowded)                     -0.28      0.42    -0.90     0.65 1.00     4966
cor(mu2_Size,mu2_Size:ConditionCrowded.vs.Uncrowded)                           0.01      0.44    -0.79     0.80 1.00     3534
cor(mu2_ConditionCrowded.vs.Uncrowded,mu2_Size:ConditionCrowded.vs.Uncrowded)     0.12      0.43    -0.74     0.83 1.00     3524

Population-Level Effects:
                                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
mu2_Intercept                            1.04      0.24     0.57     1.51 1.00     1040     1889
theta1_Intercept                        -3.01      0.86    -4.97    -1.56 1.00     2497     2546
mu2_Size                                 2.02      0.17     1.70     2.37 1.00     3212     2744
mu2_ConditionCrowded.vs.Uncrowded       -1.39      0.31    -2.00    -0.75 1.00     2447     2443
mu2_Size:ConditionCrowded.vs.Uncrowded    -0.07      0.26    -0.57     0.47 1.00     3386     3052

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 Tests for class b:
      Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star
1 (mu2_Size) > 0     2.02      0.17     1.76      2.3        Inf         1    *
---
'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
'*': For one-sided hypotheses, the posterior probability exceeds 95%;
for two-sided hypotheses, the value tested against lies outside the 95%-CI.
Posterior probabilities of point hypotheses assume equal prior probabilities.

Hypothesis Tests for class b:
                Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star
1 (mu2_ConditionCro... < 0    -1.39      0.31    -1.89     -0.9       3999         1    *
---
'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
'*': For one-sided hypotheses, the posterior probability exceeds 95%;
for two-sided hypotheses, the value tested against lies outside the 95%-CI.
Posterior probabilities of point hypotheses assume equal prior probabilities.

Hypothesis Tests for class b:
                Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star
1 (mu2_Size:Conditi... < 0    -0.07      0.26    -0.47     0.36       1.59      0.61
---
'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
'*': For one-sided hypotheses, the posterior probability exceeds 95%;
for two-sided hypotheses, the value tested against lies outside the 95%-CI.
Posterior probabilities of point hypotheses assume equal prior probabilities.
```

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.

```
Scale for 'y' is already present. Adding another scale for 'y', which will replace the existing scale.
```

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

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


−0.5
0.5
eta = 1.61 + 2.07 * Size
lambda = 0.03
eta = 0.22 + 2.05 * Size
lambda = 0.03
eta = 2.36 + 2.06 * Size
lambda = 0.03
eta = 0.96 + 2.02 * Size
lambda = 0.03
eta = 1.64 + 1.98 * Size
lambda = 0.3
eta = 0.08 + 1.79 * Size
lambda = 0.3
eta = 2.34 + 2 * Size
lambda = 0.08
eta = 0.81 + 1.83 * Size
lambda = 0.08
eta = 2.5 + 2.09 * Size
lambda = 0.04
eta = 1.16 + 2.12 * Size
lambda = 0.04
eta = 0.9 + 2.09 * Size
lambda = 0.04
eta = −0.45 + 2.1 * Size
lambda = 0.04
eta = 1.5 + 2.06 * Size
lambda = 0.08
eta = 0.09 + 2.02 * Size
lambda = 0.08
eta = 1.99 + 2.07 * Size
lambda = 0.03
eta = 0.61 + 2.06 * Size
lambda = 0.03
eta = 1.39 + 2.02 * Size
lambda = 0.13
eta = −0.09 + 1.91 * Size
lambda = 0.13
eta = 1.41 + 2.08 * Size
lambda = 0.05
eta = 0.04 + 2.07 * Size
lambda = 0.05
1
2
3
4
5
6
7
8
9
10
0.00
0.25
0.50
0.75
1.00
−2
−1
0
1
2
Crowded
no
yes
Subject
1
2
3
4
5
6
7
8
9
10

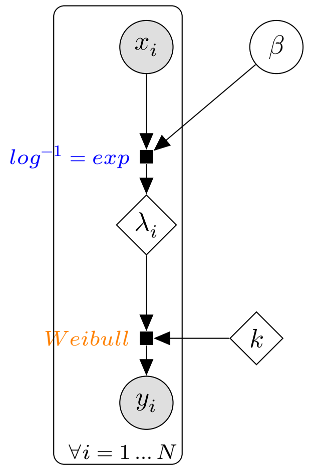
```
`summarise()` regrouping output by 'Subject', 'Size' (override with `.groups` argument)
`summarise()` regrouping output by 'Subject' (override with `.groups` argument)
```

We can also visualize 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:

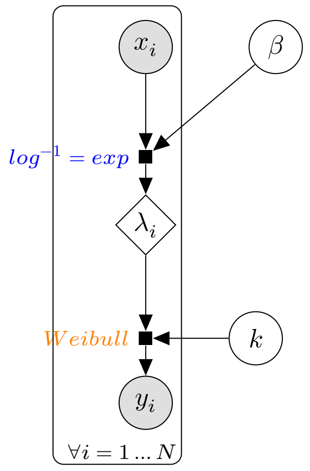
```
PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, please make sure the phantomjs execu

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

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.
```

6 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).



(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.

7 Appendix

7.1 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.)

```r
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/
```

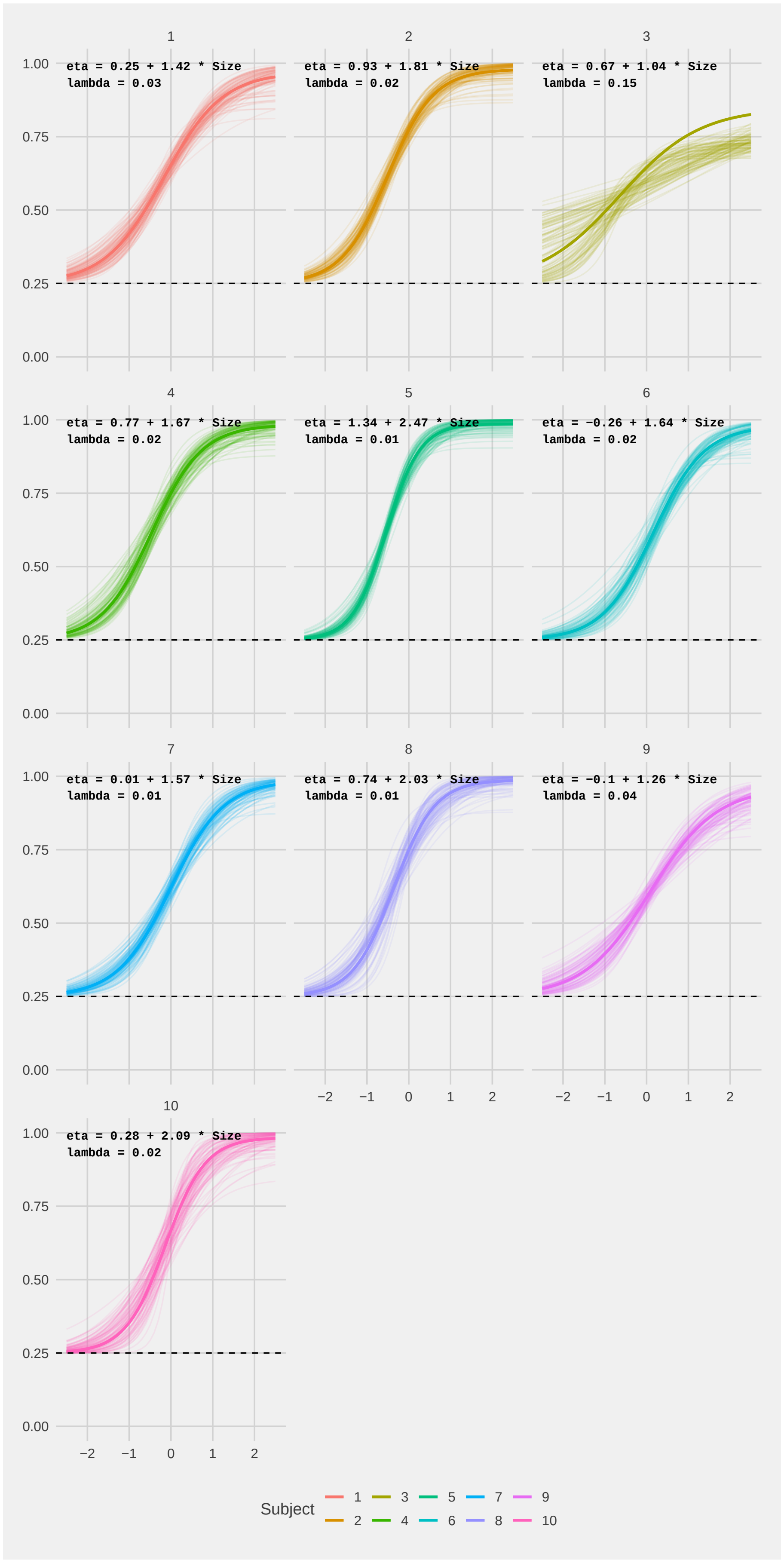
```
 Family: bernoulli
  Links: mu = identity
Formula: ResponseCorrect ~ 0.25 + (0.75 - inv_logit(lapse)) * inv_logit(eta)
         eta ~ 1 + Size + (1 + Size | Subject)
         lapse ~ 1 + (1 | Subject)
   Data: d (Number of observations: 3949)
Samples: 4 chains, each with iter = 8000; warmup = 4000; 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
2
3
eta = 0.25 + 1.42 * Size
lambda = 0.03
eta = 0.93 + 1.81 * Size
lambda = 0.02
eta = 0.67 + 1.04 * Size
lambda = 0.15
4
5
6
eta = 0.77 + 1.67 * Size
lambda = 0.02
eta = 1.34 + 2.47 * Size
lambda = 0.01
eta = −0.26 + 1.64 * Size
lambda = 0.02
7
8
9
eta = 0.01 + 1.57 * Size
lambda = 0.01
eta = 0.74 + 2.03 * Size
lambda = 0.01
eta = −0.1 + 1.26 * Size
lambda = 0.04
10
eta = 0.28 + 2.09 * Size
lambda = 0.02
1.00
0.75
0.50
0.25
0.00
−2
−1
0
1
2
Subject

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
```

8 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     2020-11-02

- 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        1.1.10     2020-09-15 [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)
 blob             1.2.1      2020-01-20 [1] CRAN (R 4.0.2)
 bridgesampling   1.0-0      2020-02-26 [1] CRAN (R 4.0.2)
 brms           * 2.14.0     2020-10-08 [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)
 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)
 cli              2.1.0      2020-10-12 [1] CRAN (R 4.0.2)
 coda             0.19-4     2020-09-30 [1] CRAN (R 4.0.2)
 codetools        0.2-16     2018-12-24 [1] CRAN (R 4.0.2)
 colorspace       1.4-1      2019-03-18 [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       1.13.2     2020-10-19 [1] CRAN (R 4.0.2)
 DBI              1.1.0      2019-12-15 [1] CRAN (R 4.0.2)
 dbplyr           1.4.4      2020-05-27 [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)
 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)
 fs               1.5.0      2020-07-31 [1] CRAN (R 4.0.2)
 generics         0.1.0      2020-10-31 [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)
 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)
 hms              0.5.3      2020-01-08 [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           0.3.16     2020-09-06 [1] CRAN (R 4.0.2)
 jsonlite         1.7.1      2020-09-07 [1] CRAN (R 4.0.2)
 knitr          * 1.30       2020-09-22 [1] CRAN (R 4.0.2)
 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)
 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)
 loo              2.3.1      2020-07-14 [1] CRAN (R 4.0.2)
 lubridate        1.7.9      2020-06-08 [1] CRAN (R 4.0.2)
 magrittr       * 1.5        2014-11-22 [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)
 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)
 modelr           0.1.8      2020-05-19 [1] CRAN (R 4.0.2)
 multcomp         1.4-14     2020-09-23 [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)
 pillar           1.4.6      2020-07-10 [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)
 prettyunits      1.1.1      2020-01-24 [1] CRAN (R 4.0.2)
 processx         3.4.4      2020-09-03 [1] CRAN (R 4.0.2)
 promises         1.1.1      2020-06-09 [1] CRAN (R 4.0.2)
 ps               1.4.0      2020-10-07 [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)
 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            0.4.8      2020-10-08 [1] CRAN (R 4.0.2)
 rmarkdown        2.5        2020-10-21 [1] CRAN (R 4.0.2)
 rprojroot        1.3-2      2018-01-03 [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       0.11       2020-02-07 [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)
 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)
 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)
 usethis          1.6.3      2020-09-17 [1] CRAN (R 4.0.2)
 V8               3.3.1      2020-10-26 [1] CRAN (R 4.0.2)
 vctrs            0.3.4      2020-08-29 [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
```