Generalized Linear Mixed/Multilevel Models (GLMMs)

In-class materials

T. Florian Jaeger

November 6, 2020

Contents

1 Overview

This document is intended as in-class material. Make sure to first work through the homework in preparation for this class. We now continue to work through the same data set as in the homework.

2 Going over homework

Here is a possible write-up based on the many great examples you provided in your homework. I highlighted a few parts that not everyone had quite right (minor typos are not italicized).

> Figure XXX shows the proportion of correct responses by number size and crowdedness condition. We analyzed *trial-level* data with mixed-effects logistic regression using the function `glmer` from the lme4 package (Bates et al., 2013) in R (R Core Team 2020). Correct responses (correct response = 1 and an incorrect response = 0) were regressed against letter size (centered), condition (deviation coded, where .5 = crowded and -.5 = uncrowded) and their interaction. Following standards of the field, we fit the maximal random effect structure required by the design, random by subject intercepts and slopes for condition, size, and their interaction. *The model converged but indicated a possible singular fit. Since additional results with reduced random effect structures confirmed the qualitative results of the model with the full random effect structure, we report the latter.*
>
> We found a statistically significant main effect of letter size ($\hat{\beta}$ = 1.847, z = 10.2, $p < .01$), such that larger letter sizes were associated with better performance. We also found a significant effect of crowdedness ondition ($\hat{\beta}$ = -.929, $z = -6.9$, $p < .01$) such that performance was reduced in the crowded,

compared to the crowded, condition. The interaction between letter size and condition was not found to be significant ($\hat{\beta}$ = -.101, $z = -.5$, $p > .5$).

Note also a few other changes that reflect my personal preference:

- I try to keep the sentence structure parallel between the different effect (e.g., "We found …" *or* "There was …" but not both).
- I try to keep the perspective on the results constant across effect (e.g. "increased/decreased performance" *or* "higher/lower proportion of correct answers")
- I try to code condition in a way that the values for the condition predictor are positive for the 'treatment' and negative for the 'control'/'baseline'. Similarly, I report the results using language that compares the treatment against the control rather than the other way around.

3 Testing a causal (mediation) hypothesis

Sometimes we—or our reviewers—are interesting in ruling out confounding effects of other variables, or we have a specific hypothesis about how precisely our manipulation comes to affect the outcome. For example, what drives the effect of crowding that we observe in Ashley's data? Consider the following two competing hypotheses about the causal chain:

- H1, direct causation: visual crowding → reduced accuracy
- H2, indirect causation: visual crowding → different eye-movements (information seeking) → reduced accuracy

Note that H1 does not rule out that visual crowding affects eye-movements. It also does not rule out that changes in eye-movements affect the accuracy of our visual decision-making. But it *does* state that visual crowding has effects on accuracy that are not reducible to change in eye-movements that may or may not be caused by the crowding. **How can we test whether Ashley's data provide information that distinguishes between H1 and H2?** It turns out that one approach to this question involves hypothesis-driven model comparisons. So let's use this question to get an initial introduction to model comparison that avoid the utter open-endedness of more exploratory model comparison (which we'll learn about in another class).

3.1 Mediation analysis

To compare H1 and H2, we need to fit and compare a few models:

- **Model 1 (nested)** with only Size * Condition
- **Model 2 (nested)** with only Size * Eye-movement measure
- **Model 3 (nesting)** with Size * (Condition + Eye-movement measure), which is the model that *nests* the both of the first two models.

For GL*M*Ms, it is important to keep in mind that we're interested in assessing the role of the *fixed* effects. We therefore use the *same random effect structure across all of the three models.* If we didn't, whatever differences we find in the models' fit might be driven by differences in their random effects, and we want to avoid that. Specifically, this means that we're using the full random effect structure that includes random slopes for both condition and eye-movements in all of the three models.

Here, we use the *diffusion constant* as a measure of the relevant eye-movement. The fixed effects of the three resulting models are summarized below. Note that the combined model (model 3) indeed contains a superset of the predictors of the two nested models:

```
boundary (singular) fit: see ?isSingular
boundary (singular) fit: see ?isSingular
boundary (singular) fit: see ?isSingular
```

```
# A tibble: 4 x 5
  term                                    estimate std.error statistic  p.value
  <chr>                                      <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)                                0.921     0.175     5.26  1.41e- 7
2 center(Size)                               1.92      0.192    10.0   1.39e-23
3 ConditionCrowded.vs.Uncrowded             -0.978     0.153    -6.37  1.87e-10
4 center(Size):ConditionCrowded.vs.Uncrow~  -0.145     0.192    -0.755 4.50e- 1
```

```
# A tibble: 4 x 5
  term                                  estimate std.error statistic p.value
  <chr>                                    <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept)                            0.840     0.413       2.03  0.0419
2 center(Size)                           1.33      0.423       3.14  0.00171
3 center(DiffusionConstant)             -0.0103    0.00489    -2.10  0.0361
4 center(Size):center(DiffusionConstant) -0.00470   0.00745    -0.630 0.528
```

```
# A tibble: 6 x 5
  term                                   estimate std.error statistic  p.value
  <chr>                                     <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)                             0.826     0.168       4.91  9.17e- 7
2 center(Size)                            1.84      0.185       9.97  2.06e-23
3 ConditionCrowded.vs.Uncrowded          -0.932     0.141      -6.59  4.45e-11
4 center(DiffusionConstant)              -0.00341   0.00298    -1.14  2.53e- 1
5 center(Size):ConditionCrowded.vs.Uncrow~ -0.112     0.184      -0.610 5.42e- 1
6 center(Size):center(DiffusionConstant)   0.00127   0.00397     0.320 7.49e- 1
```

Specifically, we will compare the third against the first model, and against the second model. If the nesting model is better than both of the nested models, then both crowding and eye-movement make unique contributions to explaining variability in the outcome. If the nesting model is better than the first model, but cannot be distinguished from the second model then the effect of crowding—which we already know to exist—is entirely subsumed by the effect of eye-movements. That is, eye-movements would explain all variability in the outcome that crowdedness explains, and more. If the nesting model is better than the second model, but not better than the first model then crowding explains any potential effect of eye-movements on the outcome (in order to determine whether the eye-movements have any effect at all, we'd have to look at the second model). Finally, if the fit of the nesting model is indistinguishable from either of the two nested models then the effect of crowding and eye-movements cannot be distinguished between by this data.

So, how do we determine which models are 'better'?

3.2 Model comparison

Both of the model comparisons (model 3 against model 1, and model 3 against model 2) are *nested model comparisons* since model 3 contains all predictors contained in model 1 or model 2. As a measure of the goodness of fit, we can use the model's deviance (recall, this is simply the model's log likelihood * -2). As the name suggests, deviance is a measure of error, so smaller is better and larger is worse. When we add predictors to a model the maximum likelihood fit of the new model will always provide at least as good or better a fit against the data as the original model. That is, deviance will never go up when we add a predictor to a maximum likelihood-fitted GLMM. This makes sense: if we provide the model with more information that it can use, it will perform at least as well as before (since we're only ever considered the model parameterization that maximize the probability of the data under the model). Let's call the difference between the nested model's deviance minus the nesting model's deviance $ _{deviance}. This difference will always be 0 or larger.

But we don't want to just say that models with less error are automatically *significantly* better. To determine whether a change in the deviance is significant, we compare it against the change in the complexity of the model. We measure this additional complexity as the increase in the number of degrees of freedom from the nested to the nesting model. Let's call this difference in the number parameters in the model δ_k. In the limit, the difference in deviance between the two models approximates a χ^2-distribution with δ_k degrees of freedom. That is, if we have sufficiently much data we can assess whether a model fit has improved significantly by asking whether $ ^2(*k)* = {deviance} is significant.

In R and Matlab, the anova functions allow us to directly compare nested LMs, GLMs, or GLMMs. For example, for the comparison of the nesting model (model 3) against the model without DiffusionConstant (model 1):

```
anova(m.both, m.condition, test = "chisq")
```

```
Data: d
Models:
m.condition: ResponseCorrect ~ 1 + center(Size) * Condition + (1 + center(Size) *
m.condition:     (Condition + center(DiffusionConstant)) | Subject)
m.both: ResponseCorrect ~ 1 + center(Size) * (Condition + center(DiffusionConstant)) +
m.both:     (1 + center(Size) * (Condition + center(DiffusionConstant)) |
m.both:         Subject)
```

```
            Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
m.condition 25 7831.5 8001.1 -3890.8   7781.5
m.both      27 7833.3 8016.5 -3889.6   7779.3 2.2493      2     0.3248
```

And for the comparison of the nesting model (model 3) against the model without condition (model 2):

```
anova(m.both, m.DiffusionConstant, test = "chisq")
```

```
Data: d
Models:
m.DiffusionConstant: ResponseCorrect ~ 1 + center(Size) * center(DiffusionConstant) +
m.DiffusionConstant:     (1 + center(Size) * (Condition + center(DiffusionConstant)) |
m.DiffusionConstant:         Subject)
m.both: ResponseCorrect ~ 1 + center(Size) * (Condition + center(DiffusionConstant)) +
m.both:     (1 + center(Size) * (Condition + center(DiffusionConstant)) |
m.both:         Subject)
                    Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
m.DiffusionConstant 25 7848.4 8018.0 -3899.2   7798.4
m.both              27 7833.3 8016.5 -3889.6   7779.3 19.115      2  7.067e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.2.1 Discuss in class

1. In your own words, what would you conclude from this mediation analysis?
2. Looking back at the model without condition (model 2), notice how the effect of number size is reduced compared to the model without the diffusion constant (model 1). What do you make out of that?

3.2.2 Further readings on model comparison

Gelman and Hill (2007, Ch. 22) provides a quick introduction to the relation between ANOVA and model comparison. James et al. (2013) cover model selection in Ch. 6.1, under the heading "Subset selection". And, Harrell (2001) provides an excellent introduction to the general principles of model selection on the first 70ish pages of his book.

4 Bayesian GLMMs

Data analyses standards keep changing, and it seems fair to say that there is an increasing push towards Bayesian approaches. We cover the general pros and cons of Bayesian approaches in another class. Here I'll just recap a few main points of why someone might switch to Bayesian GLMMs for data analysis now:

- Unlike p-values, Bayes Factors provide a coherent measure of the amount of evidence for/against a hypothesis (e.g., Wagenmakers, 2007).
  - This includes coherent measures for replication success that avoid the dichotomy of significance-based reasoning (e.g., Verhagen & Wagenmakers, 2014; for a Bayesian replication test for GLMMs, see Xie et al., 2020).
- Priors can help convergence, especially for more complex models. There are pros and cons to the role of priors, but weakly regularizing priors can serve as an intuitive implementation of Occam's razor without biasing the direction of the inferred effects.
- Access to the full posterior distribution of *all* parameters in the model. For GLMMs, this includes the estimated variances and correlations of the grouping (random) effects, which are point estimates under the frequentist approach.

4.1 An example: The Bayesian equivalent of the GLMM in the homework document

We can refit the same model we fit for the homework within a Bayesian GLMM. We first define weakly regularizing priors for the fixed effects, the variances of the random effects, and the correlations of these random effects following recommendations for GLMM priors:

```
my.priors <- c(
  prior(student_t(3, 0, 2.5), class = "b"),
  prior(cauchy(0,2.5), class = "sd"),
  prior(lkj(1), class = "cor")
)
```

Here, class "b" is the prior for the fixed effect coefficients (i.e., the **b**etas); class "sd" and "cor" are the priors for the **s**tandard **d**eviations and **cor**relations of the random effects, respectively. The prior for the fixed effects parameters, for example, is centered around 0 and attributes increasingly less probability mass to increasingly more extreme coefficient values. For these priors to make sense, it is important that they are on the right scale. The priors shown above are the recommendations for categorical predictors that are coded with 1 unit distance between levels and continuous predictors that have been scaled by *twice* their standard deviation (Gelman et al., 2008). We thus transform our predictors following accordingly:[1]

```
# 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))
```

Now we are ready to fit the Bayesian GLMM. We will be using the `brms` library. In this library, the formula is the same as for the frequentist GLMM:

```
bm <- brm(
  formula = ResponseCorrect ~ 1 + Size * Condition +
    (1 + Size * Condition | Subject),
  data = d,
  family = bernoulli("logit"),
  iter = 2000,
  prior = my.priors,
  file = "../models/GLMM"
)

my.priors <- c(
  prior(student_t(3, 0, 2.5), class = "b"),
  prior(student_t(3, 0, 2.5), class = "b", coef = "Size:ConditionCrowded.vs.Uncrowded"),
  prior(cauchy(0,2.5), class = "sd"),
  prior(lkj(1), class = "cor")
)

brm(
  formula = ResponseCorrect ~ 1 + Size * Condition +
    (1 + Size * Condition | Subject),
  data = d,
  family = bernoulli("logit"),
  iter = 2000,
  prior = my.priors
)
```

```
Compiling the C++ model

Trying to compile a simple C file

Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Frameworks/R.framework/Versions/3.6/Resources/li
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/3.6/Resources/library/StanHeaders/include/stan/math/prim/mat/fun/Eig
In file included from /Library/Frameworks/R.framework/Versions/3.6/Resources/library/RcppEigen/include/Eigen/Dense:1:
In file included from /Library/Frameworks/R.framework/Versions/3.6/Resources/library/RcppEigen/include/Eigen/Core:88:
```

[1] The same recommendations are also often given for frequentist models because they can aid convergence (by avoiding very large or small coefficients, increasing numerical accuracy during the model fitting) and facilitate comparison of effect sizes across predictors and studies (see Gelman et al., 2008).

```
/Library/Frameworks/R.framework/Versions/3.6/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknown
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/3.6/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:16: error: expecte
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/3.6/Resources/library/StanHeaders/include/stan/math/prim/mat/fun/Eig
In file included from /Library/Frameworks/R.framework/Versions/3.6/Resources/library/RcppEigen/include/Eigen/Dense:1:
/Library/Frameworks/R.framework/Versions/3.6/Resources/library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file not f
#include <complex>
         ^~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1
```

```
Start sampling


SAMPLING FOR MODEL '1cebe7a2f68b850cbe93d34a7ae30f67' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.002829 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 28.29 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 75.775 seconds (Warm-up)
Chain 1:                68.141 seconds (Sampling)
Chain 1:                143.916 seconds (Total)
Chain 1:

SAMPLING FOR MODEL '1cebe7a2f68b850cbe93d34a7ae30f67' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.001176 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 11.76 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 75.3716 seconds (Warm-up)
Chain 2:                67.4635 seconds (Sampling)
Chain 2:                142.835 seconds (Total)
Chain 2:

SAMPLING FOR MODEL '1cebe7a2f68b850cbe93d34a7ae30f67' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.001061 seconds
```

```
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 10.61 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 74.9412 seconds (Warm-up)
Chain 3:                67.5003 seconds (Sampling)
Chain 3:                142.442 seconds (Total)
Chain 3:

SAMPLING FOR MODEL '1cebe7a2f68b850cbe93d34a7ae30f67' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.001383 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 13.83 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 77.3551 seconds (Warm-up)
Chain 4:                69.6787 seconds (Sampling)
Chain 4:                147.034 seconds (Total)
Chain 4:

 Family: bernoulli
  Links: mu = logit
Formula: ResponseCorrect ~ 1 + Size * Condition + (1 + Size * Condition | Subject)
   Data: d (Number of observations: 6539)
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
sd(Intercept)                                                           0.63
sd(Size)                                                                1.05
sd(ConditionCrowded.vs.Uncrowded)                                       0.37
sd(Size:ConditionCrowded.vs.Uncrowded)                                  0.36
cor(Intercept,Size)                                                     0.44
cor(Intercept,ConditionCrowded.vs.Uncrowded)                            0.06
cor(Size,ConditionCrowded.vs.Uncrowded)                                -0.05
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                      -0.12
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                            0.10
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)   0.20
                                                                    Est.Error
sd(Intercept)                                                            0.18
sd(Size)                                                                 0.31
sd(ConditionCrowded.vs.Uncrowded)                                        0.18
sd(Size:ConditionCrowded.vs.Uncrowded)                                   0.28
```

```
cor(Intercept,Size)                                                          0.26
cor(Intercept,ConditionCrowded.vs.Uncrowded)                                 0.33
cor(Size,ConditionCrowded.vs.Uncrowded)                                      0.35
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                            0.41
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                                 0.41
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)        0.44
                                                                         l-95% CI
sd(Intercept)                                                               0.38
sd(Size)                                                                    0.61
sd(ConditionCrowded.vs.Uncrowded)                                           0.05
sd(Size:ConditionCrowded.vs.Uncrowded)                                      0.01
cor(Intercept,Size)                                                        -0.14
cor(Intercept,ConditionCrowded.vs.Uncrowded)                               -0.58
cor(Size,ConditionCrowded.vs.Uncrowded)                                    -0.67
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                          -0.80
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                               -0.70
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)      -0.71
                                                                         u-95% CI
sd(Intercept)                                                               1.07
sd(Size)                                                                    1.84
sd(ConditionCrowded.vs.Uncrowded)                                           0.79
sd(Size:ConditionCrowded.vs.Uncrowded)                                      1.05
cor(Intercept,Size)                                                         0.84
cor(Intercept,ConditionCrowded.vs.Uncrowded)                                0.68
cor(Size,ConditionCrowded.vs.Uncrowded)                                     0.64
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                           0.72
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                                0.80
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)       0.88
                                                                         Rhat
sd(Intercept)                                                            1.00
sd(Size)                                                                 1.00
sd(ConditionCrowded.vs.Uncrowded)                                        1.00
sd(Size:ConditionCrowded.vs.Uncrowded)                                   1.00
cor(Intercept,Size)                                                      1.00
cor(Intercept,ConditionCrowded.vs.Uncrowded)                             1.00
cor(Size,ConditionCrowded.vs.Uncrowded)                                  1.00
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                        1.00
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                             1.00
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)    1.00
                                                                     Bulk_ESS
sd(Intercept)                                                            1437
sd(Size)                                                                 2026
sd(ConditionCrowded.vs.Uncrowded)                                        1125
sd(Size:ConditionCrowded.vs.Uncrowded)                                   2142
cor(Intercept,Size)                                                      2462
cor(Intercept,ConditionCrowded.vs.Uncrowded)                             3724
cor(Size,ConditionCrowded.vs.Uncrowded)                                  2903
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                        4662
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                             4533
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)     3571
                                                                     Tail_ESS
sd(Intercept)                                                            2458
sd(Size)                                                                 2198
sd(ConditionCrowded.vs.Uncrowded)                                         932
sd(Size:ConditionCrowded.vs.Uncrowded)                                   2473
cor(Intercept,Size)                                                      2792
cor(Intercept,ConditionCrowded.vs.Uncrowded)                             2634
cor(Size,ConditionCrowded.vs.Uncrowded)                                  2768
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                        2899
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                             2850
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)     2921

Population-Level Effects:
                                 Estimate Est.Error l-95% CI u-95% CI Rhat
Intercept                            0.83      0.21     0.38     1.26 1.00
Size                                 2.90      0.35     2.19     3.62 1.00
ConditionCrowded.vs.Uncrowded       -0.91      0.15    -1.22    -0.62 1.00
Size:ConditionCrowded.vs.Uncrowded  -0.25      0.26    -0.73     0.30 1.00
                                 Bulk_ESS Tail_ESS
Intercept                            1026     1391
Size                                 1477     2056
```

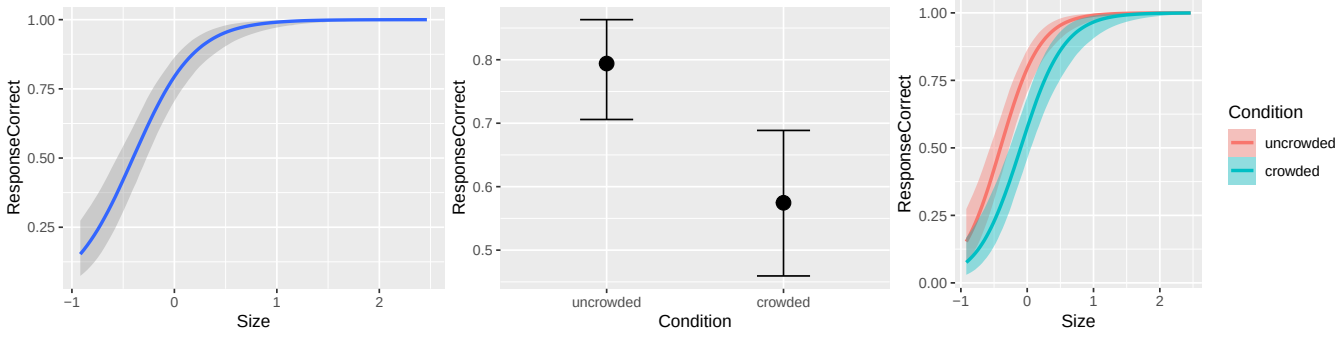
```
ConditionCrowded.vs.Uncrowded          2492     2500
Size:ConditionCrowded.vs.Uncrowded     4019     2847

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).
```

```
summary(bm)
```

```
 Family: bernoulli
  Links: mu = logit
Formula: ResponseCorrect ~ 1 + Size * Condition + (1 + Size * Condition | Subject)
   Data:  (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
sd(Intercept)                                                           0.67
sd(Size)                                                                1.20
sd(ConditionCrowded.vs.Uncrowded)                                       0.41
sd(Size:ConditionCrowded.vs.Uncrowded)                                  0.42
cor(Intercept,Size)                                                     0.36
cor(Intercept,ConditionCrowded.vs.Uncrowded)                            0.07
cor(Size,ConditionCrowded.vs.Uncrowded)                                -0.15
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                      -0.15
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                            0.12
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)   0.04
                                                                    Est.Error
sd(Intercept)                                                            0.19
sd(Size)                                                                 0.37
sd(ConditionCrowded.vs.Uncrowded)                                        0.22
sd(Size:ConditionCrowded.vs.Uncrowded)                                   0.35
cor(Intercept,Size)                                                      0.29
cor(Intercept,ConditionCrowded.vs.Uncrowded)                             0.35
cor(Size,ConditionCrowded.vs.Uncrowded)                                  0.36
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                        0.43
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                             0.41
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)    0.44
                                                                    l-95% CI
sd(Intercept)                                                           0.40
sd(Size)                                                                0.67
sd(ConditionCrowded.vs.Uncrowded)                                       0.04
sd(Size:ConditionCrowded.vs.Uncrowded)                                  0.02
cor(Intercept,Size)                                                    -0.27
cor(Intercept,ConditionCrowded.vs.Uncrowded)                           -0.62
cor(Size,ConditionCrowded.vs.Uncrowded)                                -0.77
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                      -0.85
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                           -0.71
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)  -0.79
                                                                    u-95% CI
sd(Intercept)                                                           1.15
sd(Size)                                                                2.12
sd(ConditionCrowded.vs.Uncrowded)                                       0.89
sd(Size:ConditionCrowded.vs.Uncrowded)                                  1.29
cor(Intercept,Size)                                                     0.82
cor(Intercept,ConditionCrowded.vs.Uncrowded)                            0.69
cor(Size,ConditionCrowded.vs.Uncrowded)                                 0.58
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                       0.72
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                            0.83
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)   0.82
                                                                    Rhat
sd(Intercept)                                                       1.00
sd(Size)                                                            1.00
sd(ConditionCrowded.vs.Uncrowded)                                   1.00
sd(Size:ConditionCrowded.vs.Uncrowded)                              1.00
cor(Intercept,Size)                                                 1.00
cor(Intercept,ConditionCrowded.vs.Uncrowded)                        1.00
cor(Size,ConditionCrowded.vs.Uncrowded)                             1.00
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                   1.00
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                        1.00
```

```
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded) 1.00
                                                                      Bulk_ESS
sd(Intercept)                                                             1609
sd(Size)                                                                  2237
sd(ConditionCrowded.vs.Uncrowded)                                          864
sd(Size:ConditionCrowded.vs.Uncrowded)                                    2030
cor(Intercept,Size)                                                       2358
cor(Intercept,ConditionCrowded.vs.Uncrowded)                              4282
cor(Size,ConditionCrowded.vs.Uncrowded)                                   3113
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                         5217
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                              4139
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)      3992
                                                                      Tail_ESS
sd(Intercept)                                                             2342
sd(Size)                                                                  2560
sd(ConditionCrowded.vs.Uncrowded)                                          755
sd(Size:ConditionCrowded.vs.Uncrowded)                                    2511
cor(Intercept,Size)                                                       2213
cor(Intercept,ConditionCrowded.vs.Uncrowded)                              3089
cor(Size,ConditionCrowded.vs.Uncrowded)                                   3045
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                         2741
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                              3018
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)      3537

Population-Level Effects:
                                  Estimate Est.Error l-95% CI u-95% CI Rhat
Intercept                             0.83      0.22     0.40     1.29 1.00
Size                                  3.21      0.44     2.37     4.10 1.00
ConditionCrowded.vs.Uncrowded        -1.05      0.18    -1.40    -0.70 1.00
Size:ConditionCrowded.vs.Uncrowded   -0.29      0.35    -0.96     0.44 1.00
                                  Bulk_ESS Tail_ESS
Intercept                             1063     1522
Size                                  1722     2055
ConditionCrowded.vs.Uncrowded         2816     2638
Size:ConditionCrowded.vs.Uncrowded    3746     2599

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).
```

```
plot(conditional_effects(bm), ask = F)
```



We can use Bayesian hypothesis tests to obtain a measure of the support for/against two competing hypotheses. For example, to assess the support for the hypothesis that increasing letter size results in better performance (against the hypothesis of the *opposite* effect, including a zero effect):

```
hypothesis(bm, "Size > 0")
```

```
Hypothesis Tests for class b:
  Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star
1 (Size) > 0     3.21      0.44     2.51     3.93        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.
```

Here, the evidence ratio is the Bayes factor in support of the hypothesis ($BF_{H_{\beta_{size}>0}, H_{\beta_{size}<0}}$). The posterior

probability is the posterior probability of the hypothesis being true given the data, the modeling assumptions we made, and a uniform prior over both hypotheses. The fact that the evidence ratio is reported as "infinite" just means that none of the 2000 posterior samples we obtained supported the competing hypothesis. So we might more cautiously write $BF_{H_{\beta_{size}>0}, H_{\beta_{size}<0}} > 1999, p_{posterior} > .9995$. Following Rafterty (1995), Bayes Factors are sometimes described as providing "non-decisive" or "weak" support (BF 1-3), "positive" support (BF > 3), "strong" support (BF > 20), or "very strong" support (BF > 150), though one might criticize this as running counter to the idea of avoiding dichotomizing language when summarizing results. In this case, we would conclude that we have "very strong" support for the hypothesis that performance increases with letter size, and—for the benefit of readers unfamiliar with this language—we might add that the support was "well above the limits conventionally required for significance" though such language would hopefully become unnecessary as more researchers become familiar with Bayesian data analysis.

And so forth for the other effects. Just like we might report the fixed effects of a frequentist analysis, we might report a table of hypotheses tests for the Bayesian analysis (there are many different ways of summarizing the posterior distribution of parameters, and so you will find many complementary recommendations in the literature; what I describe here is one acceptable approach).

```
hypothesis(bm, "ConditionCrowded.vs.Uncrowded < 0")
```

```
Hypothesis Tests for class b:
                Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star
1 (ConditionCrowded... < 0    -1.05      0.18    -1.33    -0.77        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(bm, "Size:ConditionCrowded.vs.Uncrowded < 0")
```

```
Hypothesis Tests for class b:
                Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star
1 (Size:ConditionCr... < 0    -0.29      0.35    -0.85      0.3       4.43      0.82
---
'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.
```

4.1.1 Discuss in class

1. How does the Bayesian model compare to the frequentist fit? What do they share and where do they differ? For comparison, here is the frequentist GLMM from the homework refit for the standardized size predictor:

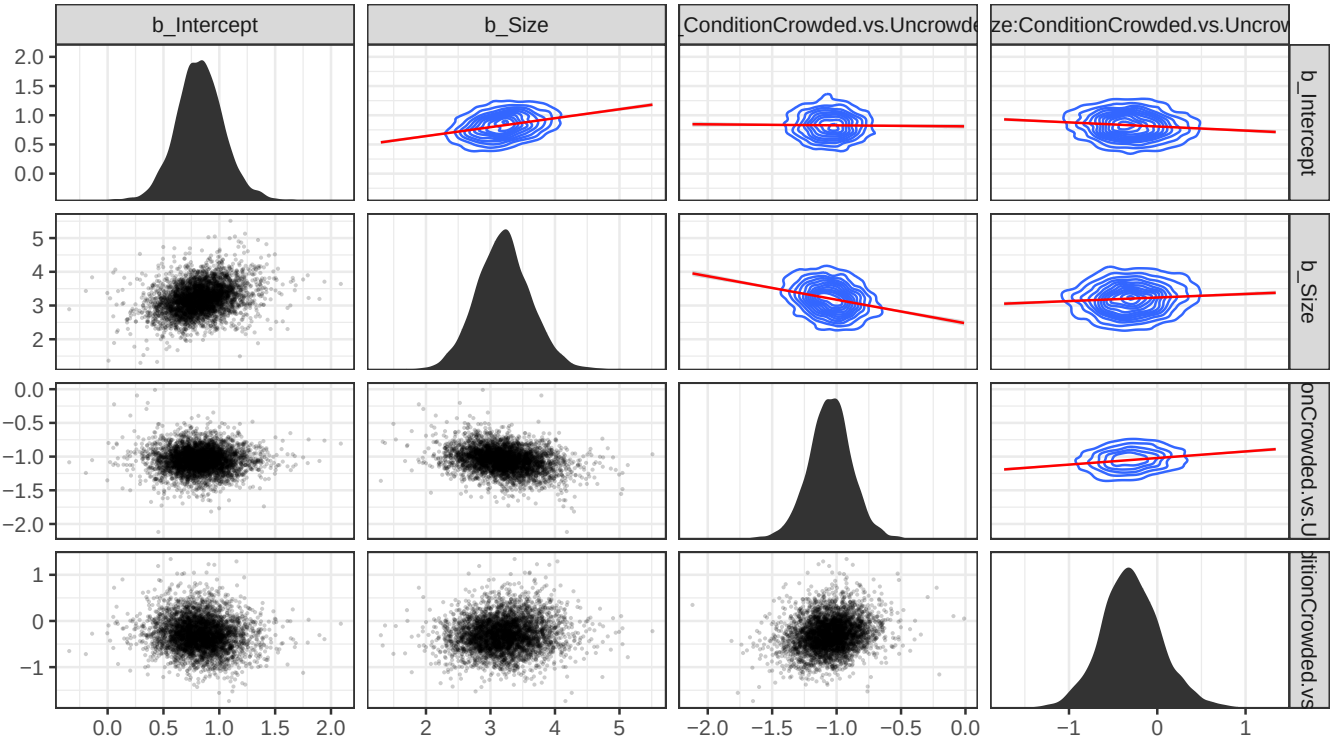
```
boundary (singular) fit: see ?isSingular
```

```
Generalized linear mixed model fit by maximum likelihood (Laplace Approximation) ['glmerMod']
 Family: binomial  ( logit )
Formula: ResponseCorrect ~ 1 + Size * Condition + (1 + Size * Condition |      Subject)
   Data: d
Control: glmerControl(optimizer = c("bobyqa"), optCtrl = list(npt = 10,      maxfun = 2e+06))

     AIC      BIC   logLik deviance df.resid
  7816.3   7911.3  -3894.2   7788.3     6525

Scaled residuals:
     Min       1Q   Median       3Q      Max
-14.2769  -0.9618   0.4980   0.7375   2.5184

Random effects:
 Groups  Name                              Variance Std.Dev. Corr
 Subject (Intercept)                       0.2709   0.5205
         Size                              0.6790   0.8240    0.66
         ConditionCrowded.vs.Uncrowded     0.1077   0.3282   -0.08 -0.14
         Size:ConditionCrowded.vs.Uncrowded 0.2024   0.4499   -0.48  0.11  0.63
Number of obs: 6539, groups:  Subject, 10
```

```
Fixed effects:
                                  Estimate Std. Error z value Pr(>|z|)
(Intercept)                         0.8387     0.1699   4.936 7.97e-07 ***
Size                                2.9451     0.2898  10.164  < 2e-16 ***
ConditionCrowded.vs.Uncrowded      -0.9290     0.1342  -6.924 4.39e-12 ***
Size:ConditionCrowded.vs.Uncrowded -0.1617     0.2841  -0.569    0.569
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) Size   CnC..U
Size         0.618
CndtnCrw..U -0.109 -0.227
Sz:CndtC..U -0.331  0.050  0.463
convergence code: 0
boundary (singular) fit: see ?isSingular
```

2. We can visualize the correlation between the posteriors samples of different fixed effect parameters. What do you make out of these correlations?

```
`geom_smooth()` using formula 'y ~ x'
```



5 Session info

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

```
- Session info -------------------------------------------------------------------------------------------------------------------------------------
 setting  value
 version  R version 3.6.0 (2019-04-26)
 os       macOS High Sierra 10.13.6
 system   x86_64, darwin15.6.0
 ui       X11
 language (EN)
 collate  en_US.UTF-8
 ctype    en_US.UTF-8
 tz       America/New_York
 date     2020-11-05
```

```
- Packages ---------------------------------------------------------------------------------------------------------------------------------------
 package        * version  date       lib source
 abind            1.4-5    2016-07-21 [1] CRAN (R 3.6.0)
 arrayhelpers     1.1-0    2020-02-04 [1] CRAN (R 3.6.0)
 assertthat       0.2.1    2019-03-21 [1] CRAN (R 3.6.0)
 backports        1.1.7    2020-05-13 [1] CRAN (R 3.6.2)
 base64enc        0.1-3    2015-07-28 [1] CRAN (R 3.6.0)
 bayesplot        1.7.1    2019-12-01 [1] CRAN (R 3.6.0)
 bayestestR       0.5.2    2020-02-12 [1] CRAN (R 3.6.0)
 boot             1.3-24   2019-12-20 [1] CRAN (R 3.6.0)
 bridgesampling   1.0-0    2020-02-26 [1] CRAN (R 3.6.0)
 brms           * 2.12.0   2020-02-23 [1] CRAN (R 3.6.0)
 Brobdingnag      1.2-6    2018-08-13 [1] CRAN (R 3.6.0)
 broom            0.5.5    2020-02-29 [1] CRAN (R 3.6.0)
 broom.mixed    * 0.2.4    2019-02-21 [1] CRAN (R 3.6.0)
 callr            3.4.3    2020-03-28 [1] CRAN (R 3.6.2)
 cellranger       1.1.0    2016-07-27 [1] CRAN (R 3.6.0)
 cli              2.0.2    2020-02-28 [1] CRAN (R 3.6.0)
 coda             0.19-3   2019-07-05 [1] CRAN (R 3.6.0)
 codetools        0.2-16   2018-12-24 [1] CRAN (R 3.6.0)
 colorspace       1.4-1    2019-03-18 [1] CRAN (R 3.6.0)
 colourpicker     1.0      2017-09-27 [1] CRAN (R 3.6.0)
 crayon           1.3.4    2017-09-16 [1] CRAN (R 3.6.0)
 crosstalk        1.0.0    2016-12-21 [1] CRAN (R 3.6.0)
 DBI              1.1.0    2019-12-15 [1] CRAN (R 3.6.0)
 dbplyr           1.4.2    2019-06-17 [1] CRAN (R 3.6.0)
 desc             1.2.0    2018-05-01 [1] CRAN (R 3.6.0)
 devtools         2.2.2    2020-02-17 [1] CRAN (R 3.6.0)
 digest           0.6.25   2020-02-23 [1] CRAN (R 3.6.0)
 dplyr          * 1.0.2    2020-08-18 [1] CRAN (R 3.6.2)
 DT               0.12     2020-02-05 [1] CRAN (R 3.6.0)
 dygraphs         1.1.1.6  2018-07-11 [1] CRAN (R 3.6.0)
 effectsize       0.2.0    2020-02-25 [1] CRAN (R 3.6.0)
 ellipsis         0.3.1    2020-05-15 [1] CRAN (R 3.6.2)
 emmeans          1.4.5    2020-03-04 [1] CRAN (R 3.6.0)
 estimability     1.3      2018-02-11 [1] CRAN (R 3.6.0)
 evaluate         0.14     2019-05-28 [1] CRAN (R 3.6.0)
 fansi            0.4.1    2020-01-08 [1] CRAN (R 3.6.0)
 farver           2.0.3    2020-01-16 [1] CRAN (R 3.6.0)
 fastmap          1.0.1    2019-10-08 [1] CRAN (R 3.6.0)
 forcats        * 0.5.0    2020-03-01 [1] CRAN (R 3.6.0)
 fs               1.3.2    2020-03-05 [1] CRAN (R 3.6.0)
 generics         0.0.2    2018-11-29 [1] CRAN (R 3.6.0)
 ggeffects        0.14.1   2020-01-28 [1] CRAN (R 3.6.0)
 ggforce        * 0.3.2    2020-06-23 [1] CRAN (R 3.6.2)
 ggplot2        * 3.3.0    2020-03-05 [1] CRAN (R 3.6.0)
 ggridges         0.5.2    2020-01-12 [1] CRAN (R 3.6.0)
 ggthemes       * 4.2.0    2019-05-13 [1] CRAN (R 3.6.0)
 glue             1.4.1    2020-05-13 [1] CRAN (R 3.6.0)
 gridExtra        2.3      2017-09-09 [1] CRAN (R 3.6.0)
 gtable           0.3.0    2019-03-25 [1] CRAN (R 3.6.0)
 gtools           3.8.1    2018-06-26 [1] CRAN (R 3.6.0)
 haven            2.2.0    2019-11-08 [1] CRAN (R 3.6.0)
 hms              0.5.3    2020-01-08 [1] CRAN (R 3.6.0)
 htmltools        0.4.0    2019-10-04 [1] CRAN (R 3.6.0)
 htmlwidgets      1.5.1    2019-10-08 [1] CRAN (R 3.6.0)
 httpuv           1.5.2    2019-09-11 [1] CRAN (R 3.6.0)
 httr             1.4.1    2019-08-05 [1] CRAN (R 3.6.0)
 igraph           1.2.4.2  2019-11-27 [1] CRAN (R 3.6.0)
 inline           0.3.15   2018-05-18 [1] CRAN (R 3.6.0)
 insight          0.8.2    2020-03-06 [1] CRAN (R 3.6.0)
 isoband          0.2.1    2020-04-12 [1] CRAN (R 3.6.2)
 jsonlite         1.6.1    2020-02-02 [1] CRAN (R 3.6.0)
 knitr          * 1.28     2020-02-06 [1] CRAN (R 3.6.0)
 labeling         0.3      2014-08-23 [1] CRAN (R 3.6.0)
 later            1.0.0    2019-10-04 [1] CRAN (R 3.6.0)
 lattice          0.20-40  2020-02-19 [1] CRAN (R 3.6.0)
 lifecycle        0.2.0    2020-03-06 [1] CRAN (R 3.6.0)
 lme4           * 1.1-21   2019-03-05 [1] CRAN (R 3.6.0)
 loo              2.2.0    2019-12-19 [1] CRAN (R 3.6.0)
```

```
lubridate       1.7.4    2018-04-11 [1] CRAN (R 3.6.0)
magrittr      * 1.5      2014-11-22 [1] CRAN (R 3.6.0)
markdown        1.1      2019-08-07 [1] CRAN (R 3.6.0)
MASS            7.3-51.5 2019-12-20 [1] CRAN (R 3.6.0)
Matrix        * 1.2-18   2019-11-27 [1] CRAN (R 3.6.0)
matrixStats     0.56.0   2020-03-13 [1] CRAN (R 3.6.0)
memoise         1.1.0    2017-04-21 [1] CRAN (R 3.6.0)
mgcv            1.8-31   2019-11-09 [1] CRAN (R 3.6.0)
mime            0.9      2020-02-04 [1] CRAN (R 3.6.0)
miniUI          0.1.1.1  2018-05-18 [1] CRAN (R 3.6.0)
minqa           1.2.4    2014-10-09 [1] CRAN (R 3.6.0)
modelr          0.1.6    2020-02-22 [1] CRAN (R 3.6.0)
multcomp        1.4-12   2020-01-10 [1] CRAN (R 3.6.0)
munsell         0.5.0    2018-06-12 [1] CRAN (R 3.6.0)
mvtnorm         1.1-0    2020-02-24 [1] CRAN (R 3.6.0)
nlme            3.1-145  2020-03-04 [1] CRAN (R 3.6.0)
nloptr          1.2.1    2018-10-03 [1] CRAN (R 3.6.0)
parameters      0.5.0    2020-02-09 [1] CRAN (R 3.6.0)
performance     0.4.4    2020-02-10 [1] CRAN (R 3.6.0)
pillar          1.4.4    2020-05-05 [1] CRAN (R 3.6.2)
pkgbuild        1.0.8    2020-05-07 [1] CRAN (R 3.6.2)
pkgconfig       2.0.3    2019-09-22 [1] CRAN (R 3.6.0)
pkgload         1.0.2    2018-10-29 [1] CRAN (R 3.6.0)
plyr            1.8.6    2020-03-03 [1] CRAN (R 3.6.0)
polyclip        1.10-0   2019-03-14 [1] CRAN (R 3.6.0)
prettyunits     1.1.1    2020-01-24 [1] CRAN (R 3.6.0)
processx        3.4.2    2020-02-09 [1] CRAN (R 3.6.0)
promises        1.1.0    2019-10-04 [1] CRAN (R 3.6.0)
ps              1.3.3    2020-05-08 [1] CRAN (R 3.6.2)
purrr         * 0.3.3    2019-10-18 [1] CRAN (R 3.6.0)
R6              2.4.1    2019-11-12 [1] CRAN (R 3.6.0)
Rcpp          * 1.0.4.6  2020-04-09 [1] CRAN (R 3.6.0)
readr         * 1.3.1    2018-12-21 [1] CRAN (R 3.6.0)
readxl          1.3.1    2019-03-13 [1] CRAN (R 3.6.0)
remotes         2.1.1    2020-02-15 [1] CRAN (R 3.6.0)
reprex          0.3.0    2019-05-16 [1] CRAN (R 3.6.0)
reshape2        1.4.3    2017-12-11 [1] CRAN (R 3.6.0)
rlang           0.4.7    2020-07-09 [1] CRAN (R 3.6.2)
rmarkdown       2.1      2020-01-20 [1] CRAN (R 3.6.0)
rprojroot       1.3-2    2018-01-03 [1] CRAN (R 3.6.0)
rsconnect       0.8.16   2019-12-13 [1] CRAN (R 3.6.0)
rstan           2.19.3   2020-02-11 [1] CRAN (R 3.6.0)
rstantools      2.0.0    2019-09-15 [1] CRAN (R 3.6.0)
rstudioapi      0.11     2020-02-07 [1] CRAN (R 3.6.0)
rvest           0.3.5    2019-11-08 [1] CRAN (R 3.6.0)
sandwich        2.5-1    2019-04-06 [1] CRAN (R 3.6.0)
scales          1.1.1    2020-05-11 [1] CRAN (R 3.6.2)
sessioninfo     1.1.1    2018-11-05 [1] CRAN (R 3.6.0)
shiny           1.4.0    2019-10-10 [1] CRAN (R 3.6.0)
shinyjs         1.1      2020-01-13 [1] CRAN (R 3.6.0)
shinystan       2.5.0    2018-05-01 [1] CRAN (R 3.6.0)
shinythemes     1.1.2    2018-11-06 [1] CRAN (R 3.6.0)
sjlabelled      1.1.3    2020-01-28 [1] CRAN (R 3.6.0)
sjmisc          2.8.3    2020-01-10 [1] CRAN (R 3.6.0)
sjPlot        * 2.8.3    2020-03-09 [1] CRAN (R 3.6.0)
sjstats         0.17.9   2020-02-06 [1] CRAN (R 3.6.0)
StanHeaders     2.21.0-1 2020-01-19 [1] CRAN (R 3.6.0)
stringi         1.4.6    2020-02-17 [1] CRAN (R 3.6.0)
stringr       * 1.4.0    2019-02-10 [1] CRAN (R 3.6.0)
survival        3.1-11   2020-03-07 [1] CRAN (R 3.6.0)
svUnit          0.7-12   2014-03-02 [1] CRAN (R 3.6.0)
testthat        2.3.2    2020-03-02 [1] CRAN (R 3.6.0)
TH.data         1.0-10   2019-01-21 [1] CRAN (R 3.6.0)
threejs         0.3.3    2020-01-21 [1] CRAN (R 3.6.0)
tibble        * 3.0.1    2020-04-20 [1] CRAN (R 3.6.2)
tidybayes     * 2.0.1    2020-01-28 [1] CRAN (R 3.6.0)
tidyr         * 1.0.2    2020-01-24 [1] CRAN (R 3.6.0)
tidyselect      1.1.0    2020-05-11 [1] CRAN (R 3.6.2)
tidyverse     * 1.3.0    2019-11-21 [1] CRAN (R 3.6.0)
TMB             1.7.16   2020-01-15 [1] CRAN (R 3.6.0)
tweenr          1.0.1    2018-12-14 [1] CRAN (R 3.6.0)
```

```
usethis          1.5.1    2019-07-04 [1] CRAN (R 3.6.0)
utf8             1.1.4    2018-05-24 [1] CRAN (R 3.6.0)
vctrs            0.3.4    2020-08-29 [1] CRAN (R 3.6.2)
withr            2.2.0    2020-04-20 [1] CRAN (R 3.6.2)
xfun             0.12     2020-01-13 [1] CRAN (R 3.6.0)
xml2             1.2.2    2019-08-09 [1] CRAN (R 3.6.0)
xtable           1.8-4    2019-04-21 [1] CRAN (R 3.6.0)
xts              0.12-0   2020-01-19 [1] CRAN (R 3.6.0)
yaml             2.2.1    2020-02-01 [1] CRAN (R 3.6.0)
zoo              1.8-7    2020-01-10 [1] CRAN (R 3.6.0)

[1] /Library/Frameworks/R.framework/Versions/3.6/Resources/library
```