Addendum: Modeling nonlinear effects of predictors

(Bayesian GLMMs and GAMMs)

T. Florian Jaeger

November 15, 2020

Contents

1 Overview

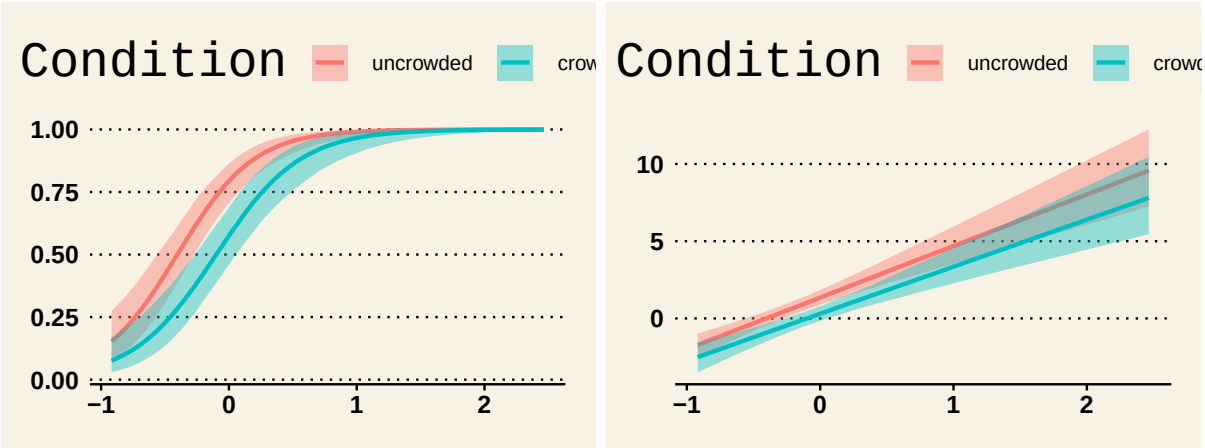
This document builds on the GLMM homework and in-class material to use the same data set to model nonlinear effects of predictors. For convenience, I use the Bayesian GLMM developed at the end of the in-class GLMM materials as a starting point. As a reminder, this GLMM modeled the trial-level effects of the crowdedness condition, stimulus size and their interaction on the accuracy participants' responses. Here I am visualizing the models predictions both in proportion space (left) and log-odds (right):

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

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))

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

```
summary(bm)
```

```
 Family: bernoulli
  Links: mu = logit
Formula: ResponseCorrect ~ 1 + Size * Condition + (1 + Size * Condition | 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 Tail_ESS
sd(Intercept)                                                       0.67      0.19     0.40     1.15 1.00     1609     2342
sd(Size)                                                            1.20      0.37     0.67     2.12 1.00     2237     2560
sd(ConditionCrowded.vs.Uncrowded)                                   0.41      0.22     0.04     0.89 1.00      864      755
sd(Size:ConditionCrowded.vs.Uncrowded)                              0.42      0.35     0.02     1.29 1.00     2030     2511
cor(Intercept,Size)                                                 0.36      0.29    -0.27     0.82 1.00     2358     2213
cor(Intercept,ConditionCrowded.vs.Uncrowded)                        0.07      0.35    -0.62     0.69 1.00     4282     3089
cor(Size,ConditionCrowded.vs.Uncrowded)                            -0.15      0.36    -0.77     0.58 1.00     3113     3045
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                  -0.15      0.43    -0.85     0.72 1.00     5217     2741
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                        0.12      0.41    -0.71     0.83 1.00     4139     3018
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)  0.04      0.44    -0.79     0.82 1.00     3992     3537

Population-Level Effects:
                                Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept                           0.83      0.22     0.40     1.29 1.00     1063     1522
Size                                3.21      0.44     2.37     4.10 1.00     1722     2055
ConditionCrowded.vs.Uncrowded      -1.05      0.18    -1.40    -0.70 1.00     2816     2638
Size:ConditionCrowded.vs.Uncrowded -0.29      0.35    -0.96     0.44 1.00     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, effects = "Size:Condition", method = "posterior_epred"), ask = F)
plot(conditional_effects(bm, effects = "Size:Condition", method = "posterior_linpred"), ask = F)
```



2 Is there a *linear* effect of eye-movements on the probability of a correct answer?

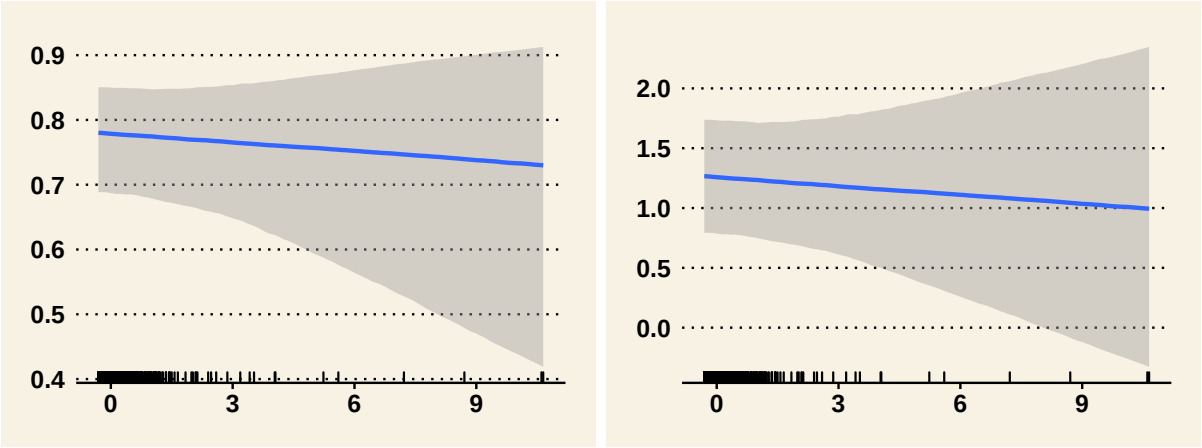
We test whether the amount of eye-movements during a trial—operationalized as the diffusion constant—has an effect on the probability of a correct answer. For the purpose of this example, we intentionally do not assess whether there are outliers, despite the fact that the diffusion constant is known to be a very noisy measure at the trial level (previous analyses only assessed its effect at the subject-level, i.e., effects of the average diffusion constant across all trials of a subject in a given condition). We begin with a model that assumes a linear effect of diffusion constant (on the log-odds of a correct answer) in addition to the effects of crowdedness condition, stimulus size, and their interaction. To put the diffusion constant on the same scale as the other predictors—so that the priors on all

coefficients have the same weakly regularizing effect—we center and standarize the diffusion constant following the same procedure as applied to stimulus size in the in-class part of the GLMM tutorial. We visualize the effect of the diffusion constant on the proportion (left) and log-odds (right) of a correct answer:

```
DiffusionConstant.mu = mean(d$DiffusionConstant)
DiffusionConstant.sd = sd(d$DiffusionConstant)

d %<>%
    mutate(
      DiffusionConstant = (DiffusionConstant - mean(DiffusionConstant)) / (2 * sd(DiffusionConstant)))

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

summary(bm.wDiffusionConstant)
```

```
 Family: bernoulli
  Links: mu = logit
Formula: ResponseCorrect ~ 1 + Size * Condition + DiffusionConstant + (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 Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)                                                         0.64      0.20     0.37     1.12 1.00     1282     2171
sd(Size)                                                              1.05      0.33     0.60     1.88 1.00     1811     2301
sd(ConditionCrowded.vs.Uncrowded)                                     0.38      0.18     0.08     0.82 1.00      845      640
sd(Size:ConditionCrowded.vs.Uncrowded)                                0.36      0.30     0.02     1.06 1.00     1635     1749
cor(Intercept,Size)                                                   0.44      0.26    -0.16     0.85 1.00     2498     2544
cor(Intercept,ConditionCrowded.vs.Uncrowded)                          0.07      0.33    -0.58     0.68 1.00     3260     2995
cor(Size,ConditionCrowded.vs.Uncrowded)                              -0.06      0.34    -0.66     0.63 1.00     2361     2262
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                    -0.11      0.42    -0.81     0.73 1.00     4815     3039
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                          0.10      0.40    -0.71     0.80 1.00     4465     2893
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)     0.21      0.43    -0.68     0.89 1.00     3932     3288

Population-Level Effects:
                                Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept                           0.81      0.21     0.36     1.23 1.00      951     1470
Size                                2.89      0.38     2.12     3.60 1.00     1508     1786
ConditionCrowded.vs.Uncrowded      -0.91      0.16    -1.22    -0.59 1.00     2150     2362
DiffusionConstant                  -0.02      0.06    -0.14     0.09 1.00     6629     2817
Size:ConditionCrowded.vs.Uncrowded -0.25      0.27    -0.73     0.32 1.00     3487     2346

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.wDiffusionConstant, effects = "DiffusionConstant", method = "posterior_epred"),
  ask = F, rug = T)
plot(
  conditional_effects(bm.wDiffusionConstant, effects = "DiffusionConstant", method = "posterior_linpred"),
  ask = F, rug = T)
```



One issue to expect from the plots shown above is the sparsity of data for high values of the diffusion constant, as evidenced in the data rug along the x-axis. Extreme values like this can be overly influential on the model fit, and this risk increases as we increase the functional flexibility of the effect of the diffusion constant in the next section (in order to entertain nonlinear effects).

3 Is there a *nonlinear* effect of eye-movements on the probability of a correct answer?

Next we entertain three ways of detecting non-linear effects of the diffusion constant. It should be noted though that a purely exploratory approach like this inflates the number of tests we conduct and thus the family-wise Type I error rate. In particular, since we are entertaining nonlinear effects a blind exploration of possible fits risks overfitting the model to the data. It is thus highly advisable to take a theory-driven approach, where the theory constraints the functional shape of the nonlinear effect. Whenever an exploratory approach is taken, it should be clearly indicated along with the consequences (risk of overfitting and inflated Type I error rate).

3.1 Polynomials

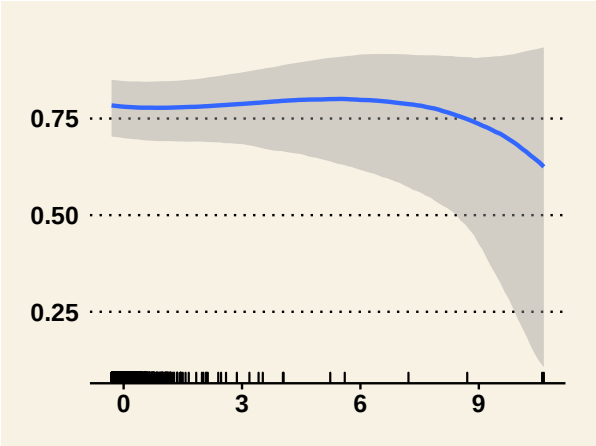
One approach to modeling nonlinear effect are polynomials. Here we use R's `poly` function to obtain *orthogonal* polynomials of the third order. We again visualize the effect of the diffusion constant on the proportion (left) and log-odds (right) of a correct answer. Based on both the model output and the plots, there is little evidence of nonlinearities in the effect of the diffusion constant, and no evidence of any non-zero effect of diffusion constant. The model output, for example, shows that the 95% credible intervals of all three components of the polynomial—including the linear component—include zero:

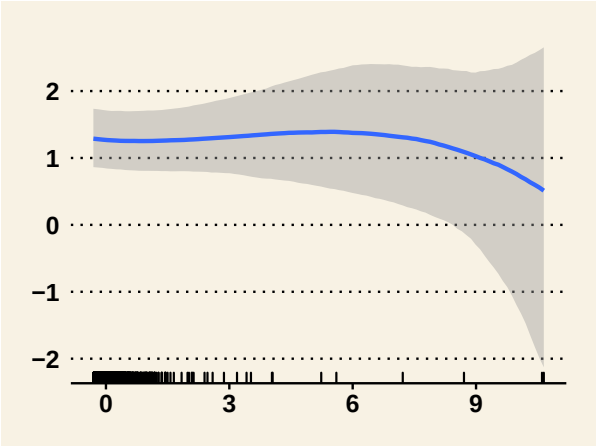
```
bm.wDiffusionConstant.poly <- brm(
  formula = ResponseCorrect ~ 1 + Size * Condition + poly(DiffusionConstant,3) +
    (1 + Size * Condition | Subject),
  data = d,
  family = bernoulli("logit"),
  iter = 2000,
  prior = my.priors,
  file = "../models/GLMM-with-DiffusionConstant-poly",
  control = list(adapt_delta = .95)
)

summary(bm.wDiffusionConstant.poly)
```

```
 Family: bernoulli
  Links: mu = logit
Formula: ResponseCorrect ~ 1 + Size * Condition + poly(DiffusionConstant, 3) + (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 Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)                                                        0.63      0.19     0.38     1.10 1.00     1289     1988
sd(Size)                                                             1.06      0.32     0.60     1.84 1.00     1718     2496
sd(ConditionCrowded.vs.Uncrowded)                                    0.39      0.18     0.09     0.81 1.00     1219     1158
sd(Size:ConditionCrowded.vs.Uncrowded)                               0.36      0.29     0.01     1.04 1.00     1552     1347
cor(Intercept,Size)                                                  0.44      0.26    -0.15     0.84 1.00     1851     2288
cor(Intercept,ConditionCrowded.vs.Uncrowded)                         0.07      0.33    -0.58     0.68 1.00     2869     2596
cor(Size,ConditionCrowded.vs.Uncrowded)                             -0.06      0.34    -0.65     0.60 1.00     2339     2832
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                   -0.11      0.41    -0.81     0.71 1.00     4290     2945
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                         0.10      0.41    -0.72     0.82 1.00     3977     3113
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)  0.19      0.43    -0.70     0.88 1.00     3237     2933

Population-Level Effects:
                               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept                          0.82      0.21     0.40     1.24 1.01      742     1377
Size                               2.90      0.37     2.15     3.62 1.00     1208     1525
ConditionCrowded.vs.Uncrowded     -0.91      0.15    -1.23    -0.60 1.00     2091     2560
polyDiffusionConstant31           -0.56      1.82    -4.27     2.99 1.00     5001     2260
polyDiffusionConstant32           -0.31      1.78    -3.91     3.20 1.00     5568     2628
polyDiffusionConstant33           -1.25      1.83    -5.18     2.17 1.00     5457     2541
Size:ConditionCrowded.vs.Uncrowded -0.24     0.26    -0.73     0.30 1.00     2723     2331

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).
```

```r
plot(
  conditional_effects(bm.wDiffusionConstant.poly, effects = "DiffusionConstant", method = "posterior_epred"),
  ask = F, rug = T)
plot(
  conditional_effects(bm.wDiffusionConstant.poly, effects = "DiffusionConstant", method = "posterior_linpred"),
  ask = F, rug = T)
```





3.2 Non-parametric smooths in a GAMM

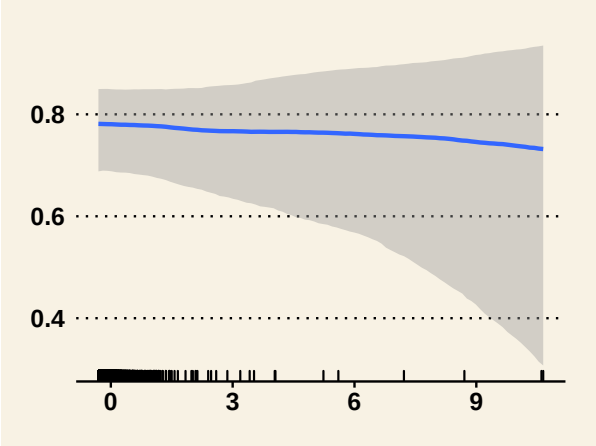
Alternatively, we can use `brms`'s ability to fit generalized additive mixed models (GAMMs) and model the effect of the diffusion constant as a non-parametric smooth. If this method is applied in an actual research project, it is important to read up on the many different options you can select from when fitting such smooths, their consequences, and potential risks. We again visualize the effect of the diffusion constant on the proportion (left) and log-odds (right) of a correct answer. Based on this output, too, we see no evidence of linear or non-linear effects of the diffusion constant.

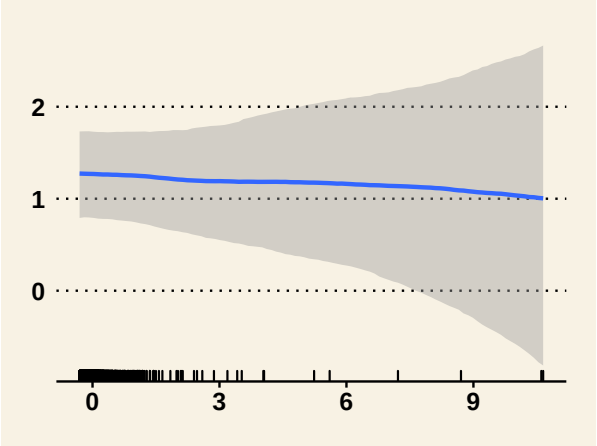
```r
bm.wDiffusionConstant.smooth <- brm(
  formula = ResponseCorrect ~ 1 + Size * Condition + s(DiffusionConstant) +
    (1 + Size * Condition | Subject),
  data = d,
  family = bernoulli("logit"),
  iter = 2000,
```

```
  prior = my.priors,
  file = "../models/GLMM-with-DiffusionConstant-smooth",
  control = list(adapt_delta = .95)
)

summary(bm.wDiffusionConstant.smooth)
```

```
 Family: bernoulli
  Links: mu = logit
Formula: ResponseCorrect ~ 1 + Size * Condition + s(DiffusionConstant) + (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

Smooth Terms:
                         Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sds(sDiffusionConstant_1)     0.73      0.76     0.02     2.73 1.00     1864     2047

Group-Level Effects:
~Subject (Number of levels: 10)
                                                                                Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)                                                                       0.64      0.19     0.38     1.11 1.00     1080     1597
sd(Size)                                                                            1.06      0.33     0.59     1.88 1.00     1457     1812
sd(ConditionCrowded.vs.Uncrowded)                                                   0.38      0.18     0.06     0.79 1.00     1013      959
sd(Size:ConditionCrowded.vs.Uncrowded)                                              0.36      0.29     0.01     1.05 1.00     1757     1903
cor(Intercept,Size)                                                                 0.44      0.27    -0.20     0.84 1.01     1529     2216
cor(Intercept,ConditionCrowded.vs.Uncrowded)                                        0.06      0.34    -0.59     0.67 1.00     3037     2852
cor(Size,ConditionCrowded.vs.Uncrowded)                                            -0.07      0.35    -0.70     0.62 1.00     1809     2328
cor(Intercept,Size:ConditionCrowded.vs.Uncrowded)                                  -0.12      0.41    -0.80     0.72 1.00     3921     2815
cor(Size,Size:ConditionCrowded.vs.Uncrowded)                                        0.09      0.41    -0.70     0.80 1.00     4372     3142
cor(ConditionCrowded.vs.Uncrowded,Size:ConditionCrowded.vs.Uncrowded)               0.19      0.44    -0.72     0.89 1.00     3290     2955

Population-Level Effects:
                               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept                          0.81      0.22     0.34     1.24 1.01      801     1160
Size                               2.90      0.38     2.13     3.65 1.00     1247     1453
ConditionCrowded.vs.Uncrowded     -0.91      0.16    -1.23    -0.60 1.00     1939     2114
Size:ConditionCrowded.vs.Uncrowded -0.25      0.26    -0.73     0.30 1.00     3083     1856
sDiffusionConstant_1              -0.41      1.35    -3.29     2.22 1.00     3334     2535

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.wDiffusionConstant.smooth, effects = "DiffusionConstant", method = "posterior_epred"),
  ask = F, rug = T)
plot(
  conditional_effects(bm.wDiffusionConstant.smooth, effects = "DiffusionConstant", method = "posterior_linpred"),
  ask = F, rug = T)
```





Notice further how this non-parametric smooth avoids the perhaps misleading impression that one might get from

the polynomial in the previous section: the indication of a non-linear effect for large values of diffusion constant. Polynomial fits are well-known to be overly sensitive to data points with extreme values for the predictor, where the data is often sparse. This is one way in which polynomials are likely to overfit the data, yielding bad predictions in particular for novel data with predictor values that fall outside of the range observed in the sample the model was fit to.

4 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     2020-11-15

- Packages -------------------------------------------------------------------------------------------------------------------------
 package        * version    date       lib source
 abind            1.4-5      2016-07-21 [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)
 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)
 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)
 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)
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)
mgcv             1.8-33     2020-08-27 [1] CRAN (R 4.0.2)
mime             0.9        2020-02-04 [1] CRAN (R 4.0.2)
miniUI           0.1.1.1    2018-05-18 [1] CRAN (R 4.0.2)
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)
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)
SparseM          1.78       2019-12-13 [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)
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)
 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)
 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
```