E2 prediction

R. Cassano-Coleman

2025-09-06

This notebook analyzes prediction using Bayesian binomial generalized linear mixed effects models (GLMMs).

Set up

```r
set.seed(15000)
```

```r
data <- read_csv('../data/E1-E2-E4/prediction.csv')
```

```
## Rows: 3210 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): response, scramble, Musician
## dbl (3): exp_subject_id, Trial_Nr, yrs_mus_exp
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Convert variables to factors.

```r
data %<>%
  mutate(exp_subject_id = as.factor(exp_subject_id),
         response = ifelse(response == 'Correct', TRUE, FALSE),
         scramble = factor(scramble, levels = c('8B', '2B', '1B')),
         Musician = factor(Musician, levels = c('Yes', 'No'))) %>%
  filter(!is.na(response))
```

Set the contrast for condition.

```r
contrasts(data$scramble) <- contr.treatment(3)
print(contrasts(data$scramble))
```

```
##    2 3
## 8B 0 0
## 2B 1 0
## 1B 0 1
```

```r
contrasts(data$Musician)
```

```
##     No
## Yes  0
## No   1
```

Main analysis

Priors

Priors are expressed in log(odds) space.

Intercept: Given that chance is 50%, we assume that participants will perform somewhere between chance and ceiling. We expect the center of the distribution of accuracy to be somewhere around 75% or 80%. If we use a center of 80% and an SD of 1, 95% of the values fall between 35.1% and 96.7%.

```
prior_intercept <- set_prior('normal(log(0.8 / (1 - 0.8)), 1)', class = 'Intercept')
```

Group: We might expect musicians to do slightly better than non-musicians, on average.

In this range, a difference in 0.25 log odds gives us about a 5% decrease in accuracy.

```
prior_mus <- set_prior('normal(-0.25, 1)', coef = 'MusicianNo')
```

Scramble: We expect performance to improve as scramble level decreases. If we code 8B as reference level, then we expect 8B > 2B and 8B > 1B.

Since we're keeping the musician slope at SD = 1, we'll keep these (and the interactions) at SD = 1. This seems to be a pretty weak prior.

```
prior_scramble2B <- set_prior('normal(-0.1, 1)', coef = 'scramble2')
prior_scramble1B <- set_prior('normal(-0.2, 1)', coef = 'scramble3')
```

Interaction: We expect no interaction between group and scramble.

```
prior_int2B <- set_prior('normal(0, 1)', coef = 'MusicianNo:scramble2')
prior_int1B <- set_prior('normal(0, 1)', coef = 'MusicianNo:scramble3')
```

Random slope for subjects: *Leave this as default for now, may update.*

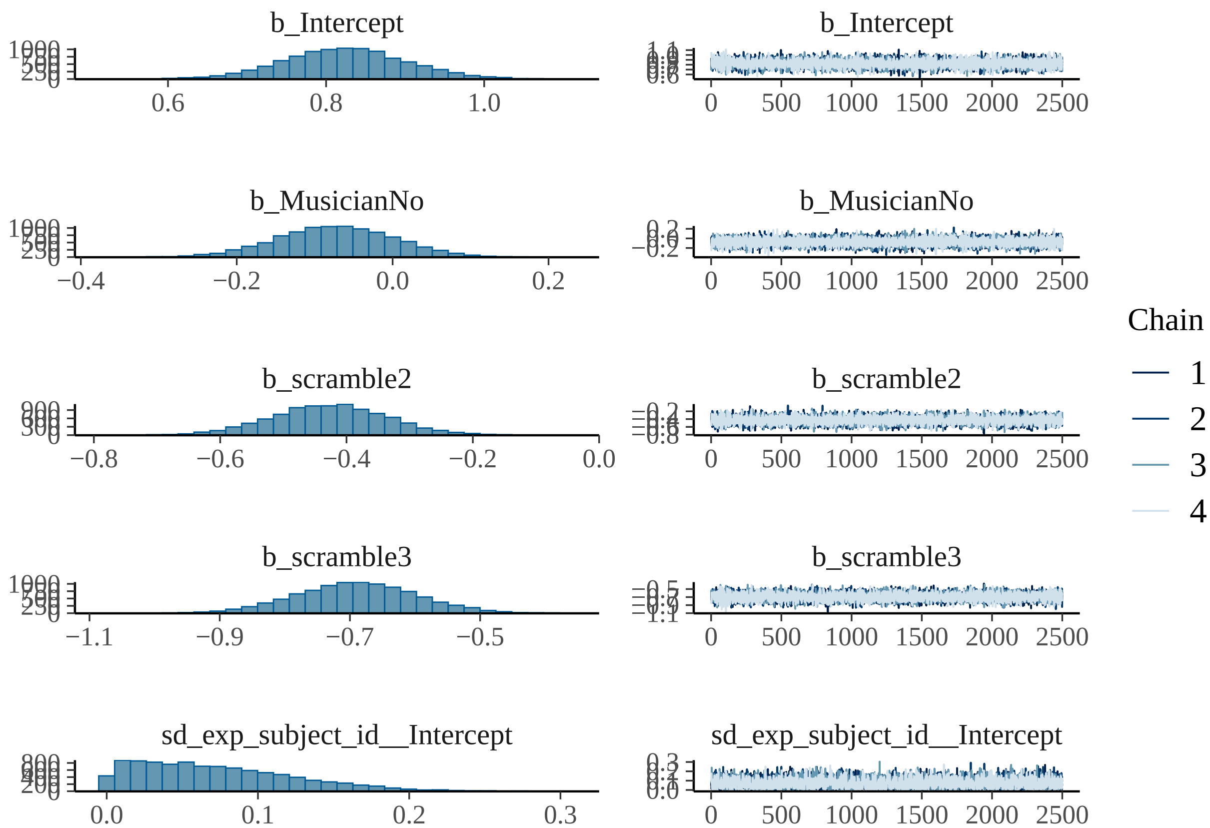
Main model with group and condition

```r
get_prior(response ~ Musician + scramble + (1 | exp_subject_id), data = data)
```

```
##                 prior     class       coef          group resp dpar nlpar lb ub
##                (flat)         b
##                (flat)         b MusicianNo
##                (flat)         b  scramble2
##                (flat)         b  scramble3
##  student_t(3, 0, 2.5) Intercept
##  student_t(3, 0, 2.5)        sd                                            0
##  student_t(3, 0, 2.5)        sd            exp_subject_id                  0
##  student_t(3, 0, 2.5)        sd  Intercept exp_subject_id                  0
##  student_t(3, 0, 2.5)     sigma                                            0
##        source
##       default
##  (vectorized)
##  (vectorized)
##  (vectorized)
##       default
##       default
##  (vectorized)
##  (vectorized)
##       default
```

```r
mus_scram <- brm(response ~ Musician + scramble + (1 | exp_subject_id), data = data,
                 family = bernoulli(),
                 prior = c(prior_intercept, prior_mus,
                           prior_scramble2B, prior_scramble1B),
                 save_pars = save_pars(all = TRUE), iter = 5000,
                 file = 'models/E2_mus_scram')
```

```r
plot(mus_scram)
```




```r
print(summary(mus_scram), digits = 4)
```

```
## Warning: There were 1 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

##  Family: bernoulli
##   Links: mu = logit
## Formula: response ~ Musician + scramble + (1 | exp_subject_id)
##    Data: data (Number of observations: 3158)
##   Draws: 4 chains, each with iter = 5000; warmup = 2500; thin = 1;
##          total post-warmup draws = 10000
##
## Multilevel Hyperparameters:
## ~exp_subject_id (Number of levels: 105)
##               Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## sd(Intercept)   0.0723    0.0510   0.0032   0.1866 1.0012     3170     4111
##
## Regression Coefficients:
##             Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## Intercept     0.8244    0.0776   0.6738   0.9791 1.0007     9598     6847
## MusicianNo   -0.0726    0.0749  -0.2196   0.0717 1.0000    12486     6999
## scramble2    -0.4270    0.0912  -0.6051  -0.2464 1.0004     9666     8092
## scramble3    -0.6900    0.0920  -0.8697  -0.5119 1.0005     9341     7771
##
## Draws were sampled 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
emm_mus_scram_s <- emmeans(mus_scram, specs = "scramble")
summary(emm_mus_scram_s)
```

```
##  scramble emmean lower.HPD upper.HPD
##  8B       0.7877    0.6573     0.922
##  2B       0.3615    0.2354     0.482
##  1B       0.0986   -0.0215     0.222
##
## Results are averaged over the levels of: Musician
## Point estimate displayed: median
## Results are given on the logit (not the response) scale.
## HPD interval probability: 0.95
```

```r
contrast(emm_mus_scram_s, method = "pairwise")
```

```
##  contrast estimate lower.HPD upper.HPD
##  8B - 2B     0.427    0.2486     0.607
##  8B - 1B     0.690    0.5118     0.869
##  2B - 1B     0.264    0.0868     0.434
##
## Results are averaged over the levels of: Musician
## Point estimate displayed: median
## Results are given on the log odds ratio (not the response) scale.
## HPD interval probability: 0.95
```

```r
emm_mus_scram_ms <- emmeans(mus_scram, specs = c("Musician", "scramble"))
summary(emm_mus_scram_ms)
```

```
##  Musician scramble emmean lower.HPD upper.HPD
##  Yes      8B        0.824   0.66884     0.973
##  No       8B        0.752   0.60539     0.905
##  Yes      2B        0.397   0.25214     0.535
##  No       2B        0.325   0.18342     0.470
##  Yes      1B        0.135  -0.00177     0.275
##  No       1B        0.062  -0.07678     0.210
##
## Point estimate displayed: median
## Results are given on the logit (not the response) scale.
## HPD interval probability: 0.95
```

```r
contrast(emm_mus_scram_ms, method = "pairwise")
```

```
##  contrast        estimate lower.HPD upper.HPD
##  Yes 8B - No 8B    0.0725   -0.0738     0.216
##  Yes 8B - Yes 2B   0.4271    0.2486     0.607
##  Yes 8B - No 2B    0.5002    0.2679     0.737
##  Yes 8B - Yes 1B   0.6899    0.5118     0.869
##  Yes 8B - No 1B    0.7634    0.5348     1.000
##  No 8B - Yes 2B    0.3548    0.1208     0.578
##  No 8B - No 2B     0.4271    0.2486     0.607
##  No 8B - Yes 1B    0.6177    0.3935     0.842
##  No 8B - No 1B     0.6899    0.5118     0.869
##  Yes 2B - No 2B    0.0725   -0.0738     0.216
##  Yes 2B - Yes 1B   0.2637    0.0868     0.434
##  Yes 2B - No 1B    0.3359    0.1035     0.553
##  No 2B - Yes 1B    0.1898   -0.0357     0.415
```

```
##  No 2B - No 1B     0.2637    0.0868     0.434
##  Yes 1B - No 1B    0.0725   -0.0738     0.216
##
## Point estimate displayed: median
## Results are given on the log odds ratio (not the response) scale.
## HPD interval probability: 0.95
```

Main effects

```r
main_BF <- describe_posterior(mus_scram,
                              estimate = "median", dispersion = TRUE,
                              ci = .95, ci_method = "HDI",
                              test = c("bayes_factor"))
```

```
## Warning: Bayes factors might not be precise.
##   For precise Bayes factors, sampling at least 40,000 posterior samples is
##   recommended.
```

```r
print(main_BF, digits = 5)
```

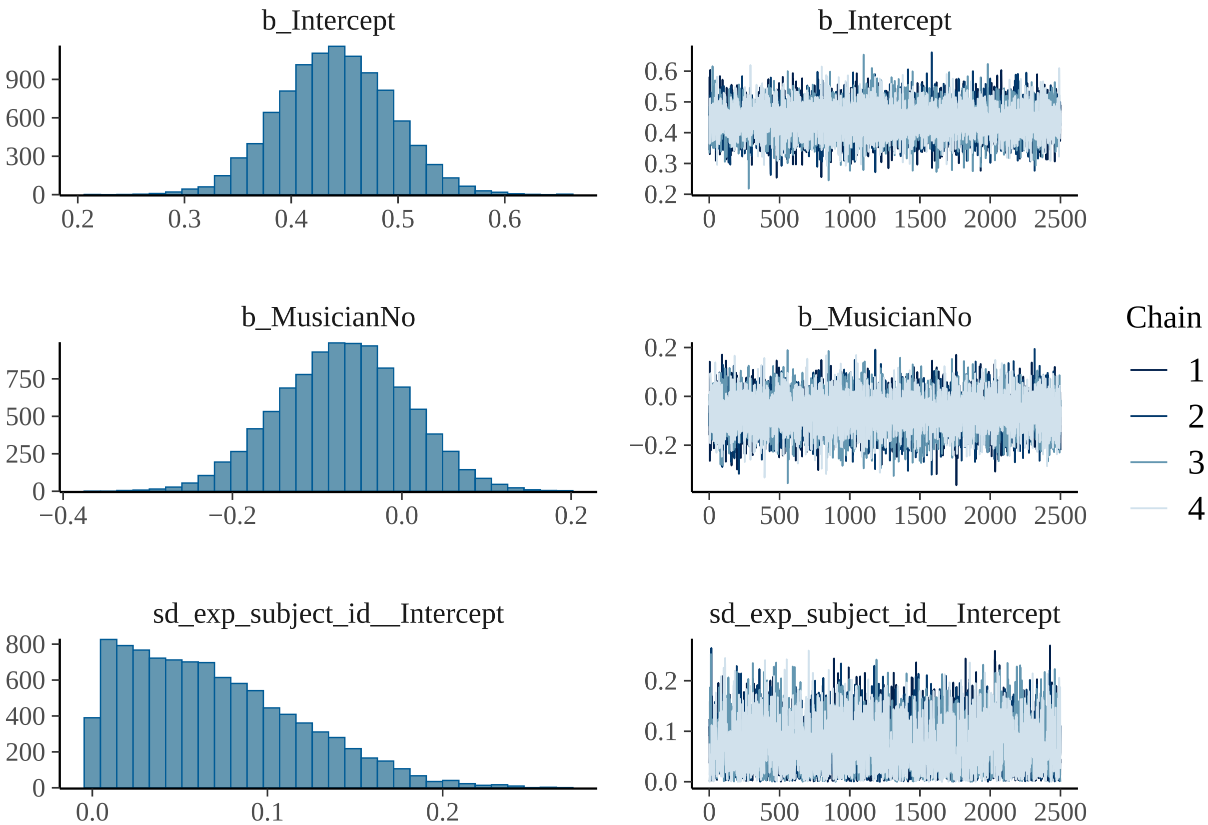
```
## Summary of Posterior Distribution
##
## Parameter   |   Median |     MAD |         95% CI |       BF |  Rhat |         ESS
## ---------------------------------------------------------------------------------
## (Intercept) |  0.82360 | 0.07590 | [ 0.67,  0.97] | 8.58e+09 | 1.000 |  9632.00000
## MusicianNo  | -0.07245 | 0.07581 | [-0.22,  0.07] |    0.114 | 1.000 | 12615.00000
## scramble2   | -0.42709 | 0.09068 | [-0.61, -0.25] |   648.50 | 1.000 |  9634.00000
## scramble3   | -0.68986 | 0.09222 | [-0.87, -0.51] | 1.38e+06 | 1.000 |  9300.00000
```

Strong evidence against a main effect of group.

To get the main effect of scramble level, fit the "null" model with group only to compare.

```
mus_only <- brm(response ~ Musician + (1 | exp_subject_id), data = data,
                 family = bernoulli(),
                 prior = c(prior_intercept, prior_mus),
                 save_pars = save_pars(all = TRUE), iter = 5000,
                 file = 'models/E2_mus_only')
```

```
plot(mus_only)
```




```
print(summary(mus_only), digits = 4)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: response ~ Musician + (1 | exp_subject_id)
##    Data: data (Number of observations: 3158)
##   Draws: 4 chains, each with iter = 5000; warmup = 2500; thin = 1;
##          total post-warmup draws = 10000
##
## Multilevel Hyperparameters:
## ~exp_subject_id (Number of levels: 105)
##               Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## sd(Intercept)   0.0693    0.0488   0.0029   0.1774 1.0006     3401     4362
##
## Regression Coefficients:
##            Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## Intercept    0.4410    0.0522   0.3403   0.5426 0.9999    13508     7350
## MusicianNo  -0.0688    0.0755  -0.2168   0.0746 1.0005    13455     7340
##
```

```
## Draws were sampled 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
BF_scramble <- bayes_factor(mus_scram, mus_only)
```

```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
```

```r
print(BF_scramble)
```

```
## Estimated Bayes factor in favor of mus_scram over mus_only: 72700233010.60995
```

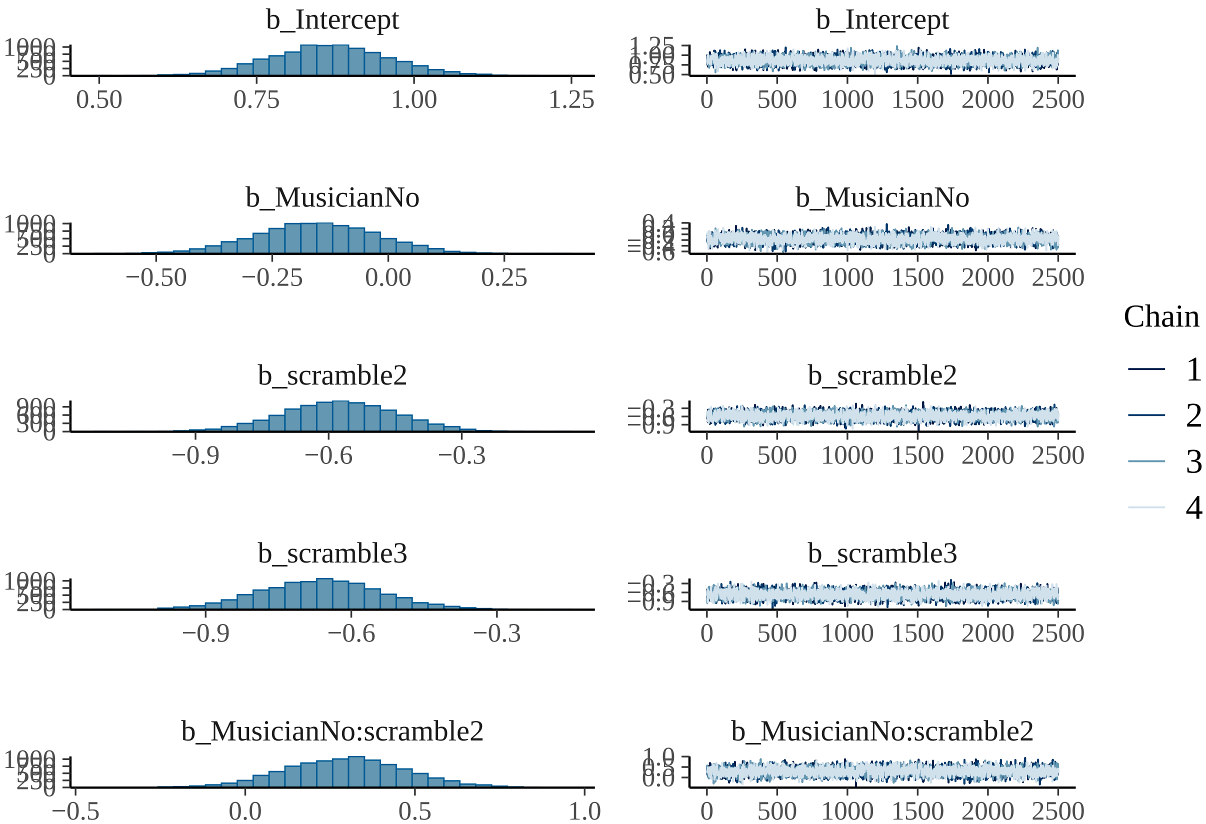
Very strong evidence for a main effect of scramble condition.

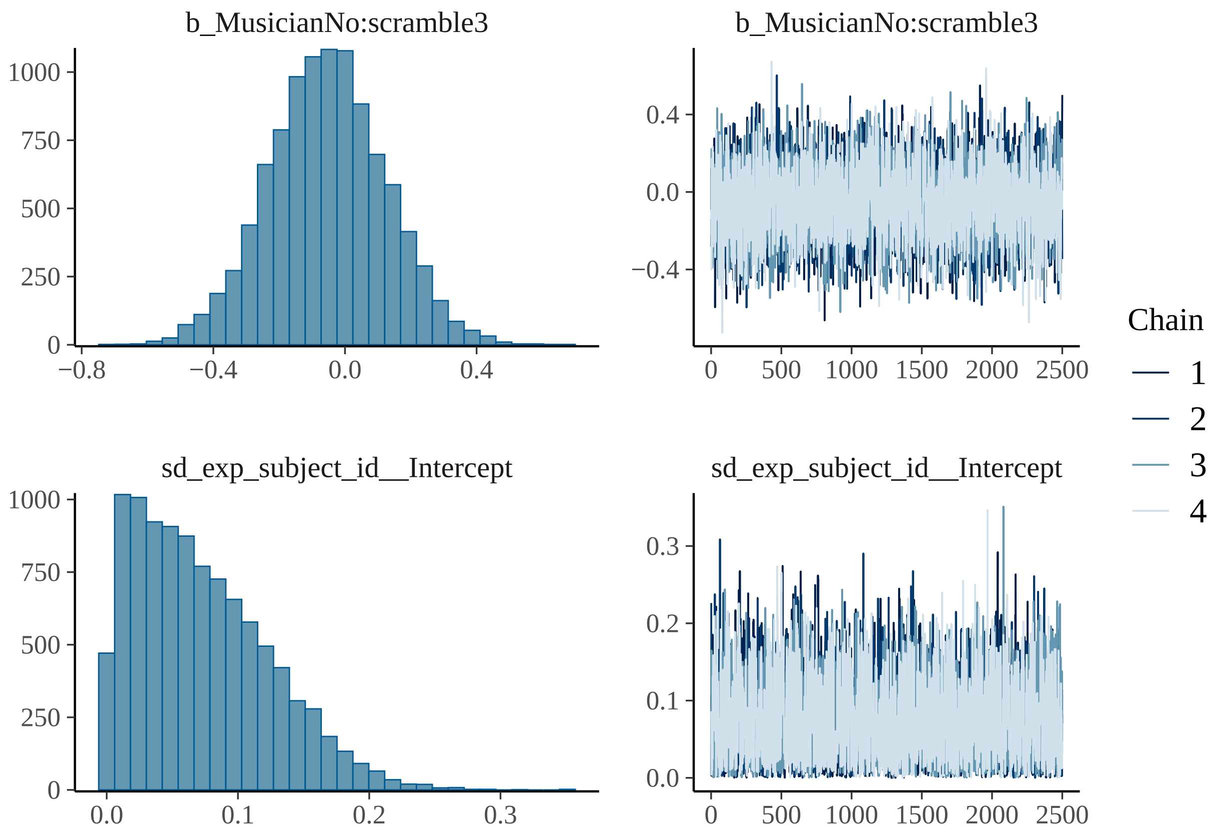
Interaction between group and condition?

Add an interaction between group and condition, and compare the model with the interaction to the one without.

```
mus_scram_int <- brm(response ~ Musician*scramble + (1 | exp_subject_id), data = data,
                  family = bernoulli(),
                  prior = c(prior_intercept, prior_mus,
                            prior_scramble2B, prior_scramble1B,
                            prior_int2B, prior_int1B),
                  save_pars = save_pars(all = TRUE), iter = 5000,
                  file = 'models/E2_mus_scram_int')
```

```
plot(mus_scram_int)
```






```r
print(summary(mus_scram_int), digits = 4)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: response ~ Musician * scramble + (1 | exp_subject_id)
##    Data: data (Number of observations: 3158)
##   Draws: 4 chains, each with iter = 5000; warmup = 2500; thin = 1;
##          total post-warmup draws = 10000
##
## Multilevel Hyperparameters:
## ~exp_subject_id (Number of levels: 105)
##               Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## sd(Intercept)   0.0725    0.0513   0.0030   0.1878 1.0005     3258     4589
##
## Regression Coefficients:
##                      Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS
## Intercept              0.8665    0.0941   0.6844   1.0559 1.0005     5201
## MusicianNo            -0.1543    0.1311  -0.4120   0.0998 1.0002     4588
## scramble2             -0.5752    0.1281  -0.8276  -0.3277 1.0001     5845
## scramble3             -0.6636    0.1252  -0.9095  -0.4141 1.0004     5707
## MusicianNo:scramble2   0.2957    0.1781  -0.0511   0.6444 1.0007     5023
## MusicianNo:scramble3  -0.0544    0.1770  -0.4008   0.2934 1.0000     4856
##                      Tail_ESS
## Intercept                7176
## MusicianNo               6067
## scramble2                7095
## scramble3                6800
## MusicianNo:scramble2     6670
```

```
## MusicianNo:scramble3     6214
##
## Draws were sampled 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).
```

```
BF_int <- bayes_factor(mus_scram_int, mus_scram)
```

```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
```

```
print(BF_int)
```

```
## Estimated Bayes factor in favor of mus_scram_int over mus_scram: 0.27155
```

Strong evidence against an interaction between group and condition.

Figure 2B

Create a helper function for the conversion from log odds to probability.

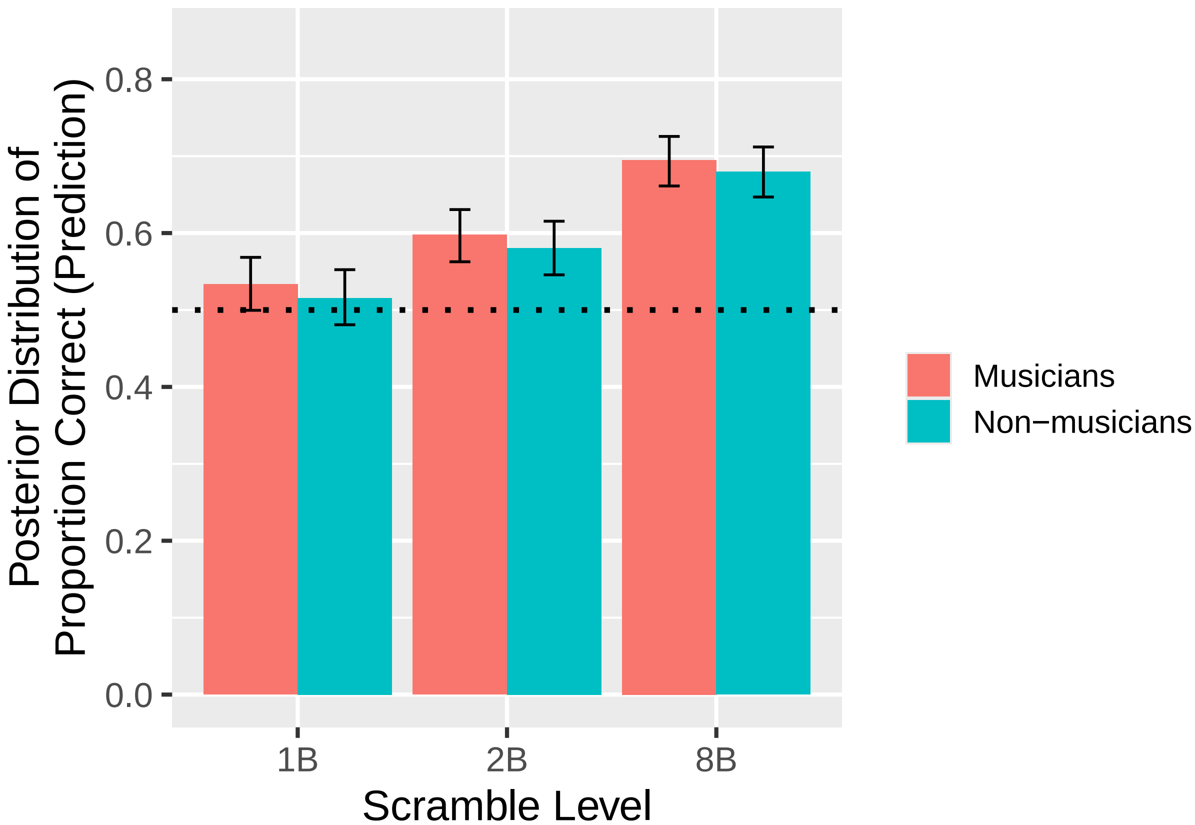
```
calculate_prob_from_logodds <- function(logodds) {
  return(exp(logodds) / (1 + exp(logodds)))
}
```

Visualize with posterior estimates and 95% CrI on the scale of accuracy.

```
posterior_est <- as.data.frame(emm_mus_scram_ms)
```

```
ggplot() +
  geom_col(aes(x = scramble, y = calculate_prob_from_logodds(emmean), fill = Musician),
           data = posterior_est,
           position = "dodge") +
  geom_errorbar(aes(x = scramble,
                    ymin = calculate_prob_from_logodds(lower.HPD),
                    ymax = calculate_prob_from_logodds(upper.HPD),
                    fill = Musician),
                data = posterior_est, position = position_dodge(width = 0.9), width = 0.2) +
  geom_hline(yintercept = 0.5, linetype = "dotted", color = "black", linewidth = 1) +
  theme_gray(base_size = 16) +
  scale_x_discrete(limits = rev) +
  ylim(0, 0.85) +
  xlab('Scramble Level') +
  ylab('Posterior Distribution of\nProportion Correct (Prediction)') +
  scale_fill_discrete(name="", labels=c('Musicians', 'Non-musicians')) +
  theme(legend.text = element_text(size = 12))
```

```
## Warning in geom_errorbar(aes(x = scramble, ymin =
## calculate_prob_from_logodds(lower.HPD), : Ignoring unknown aesthetics: fill
```




```
ggsave('../figures/Fig2B_prediction.png', width = 7, height = 5)
```

1B condition at chance?

There is technically no "right" answer, so performance in the 1B condition should be at chance.

```
data1B <- filter(data, scramble == '1B')
```

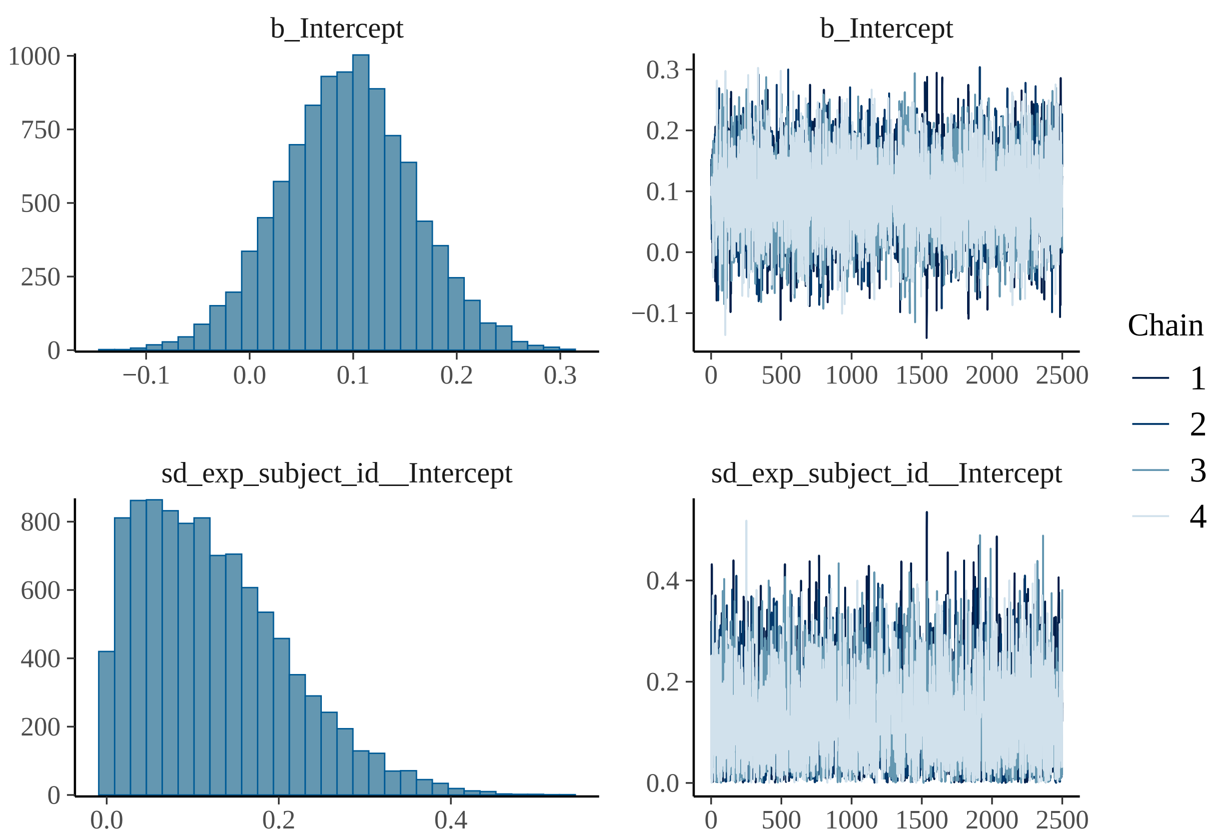
```
get_prior(response ~ 1 + (1 | exp_subject_id), data = data1B)
```

```
##                  prior     class      coef          group resp dpar nlpar lb ub
##  student_t(3, 0, 2.5) Intercept
##  student_t(3, 0, 2.5)        sd                                              0
##  student_t(3, 0, 2.5)        sd           exp_subject_id                     0
##  student_t(3, 0, 2.5)        sd Intercept exp_subject_id                     0
##  student_t(3, 0, 2.5)     sigma                                              0
##        source
##       default
##       default
##  (vectorized)
##  (vectorized)
##       default
```

(Leave the default prior for this intercept.)

```
only1B <- brm(response ~ 1 + (1 | exp_subject_id), data = data1B,
              family = bernoulli(),
              save_pars = save_pars(all = TRUE), iter = 5000,
              file = 'models/E2_only1B')
```

```
plot(only1B)
```



```r
print(summary(only1B), digits = 4)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: response ~ 1 + (1 | exp_subject_id)
##    Data: data1B (Number of observations: 1054)
##   Draws: 4 chains, each with iter = 5000; warmup = 2500; thin = 1;
##          total post-warmup draws = 10000
##
## Multilevel Hyperparameters:
## ~exp_subject_id (Number of levels: 105)
##               Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## sd(Intercept)   0.1247    0.0872   0.0055   0.3286 1.0004     3804     5028
##
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## Intercept   0.0944    0.0637  -0.0310   0.2206 1.0002    15621     6872
##
## Draws were sampled 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).
```

Is intercept different from 0?

```r
bf_pointnull(only1B, null = 0)
```

```
## Sampling priors, please wait...

## Warning: Bayes factors might not be precise.
##   For precise Bayes factors, sampling at least 40,000 posterior samples is
##   recommended.

## Bayes Factor (Savage-Dickey density ratio)
##
## Parameter   |    BF
## -------------------
## (Intercept) | 0.072
##
## * Evidence Against The Null: 0
```

There is strong evidence that performance in the 1B condition is at chance.

What if we just look at 8B and 2B?

The main thing here is to see if the interaction we see between group and condition (that we see visually) shows up when we take out 1B.

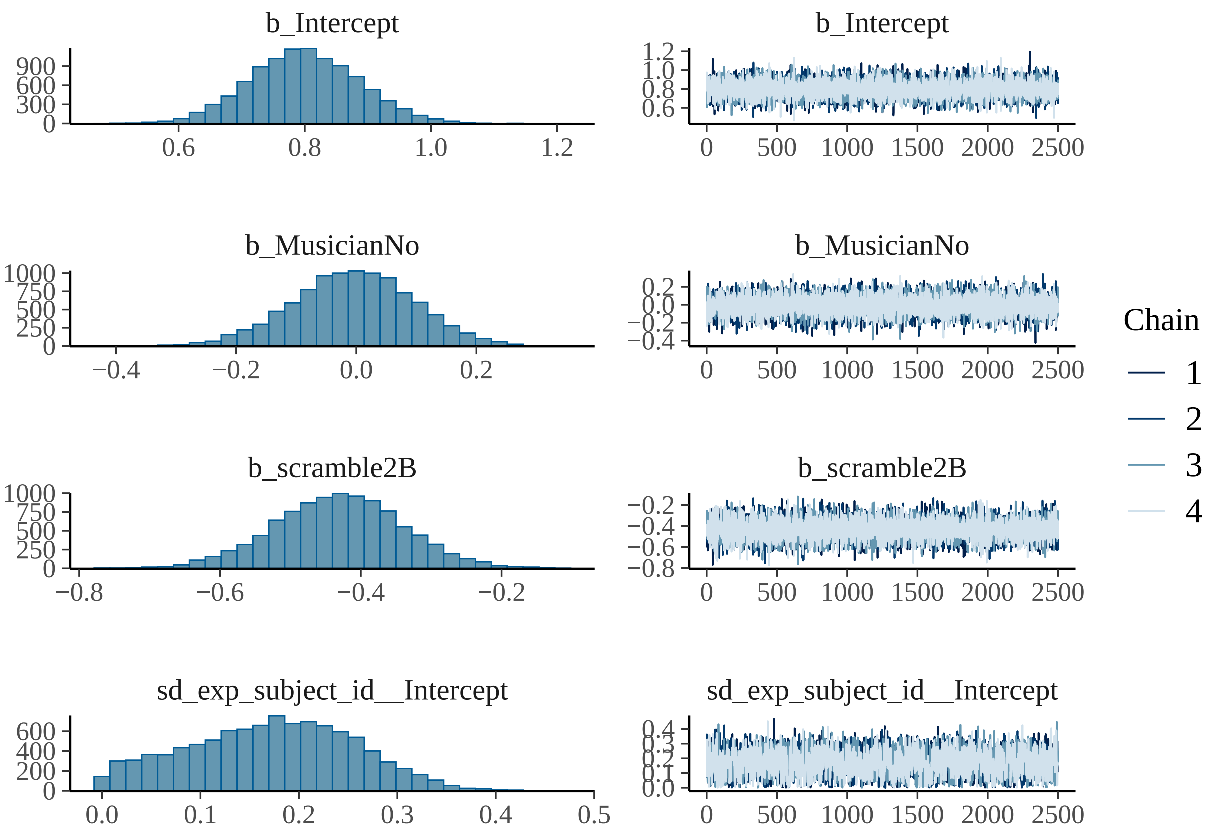
```
data_no1B <- filter(data, scramble != '1B')
```

```
get_prior(response ~ Musician + scramble + (1 | exp_subject_id), data = data_no1B)
```

```
## Warning: contrasts dropped from factor scramble due to missing levels

##                 prior     class       coef          group resp dpar nlpar lb ub
##                (flat)         b
##                (flat)         b MusicianNo
##                (flat)         b scramble2B
##  student_t(3, 0, 2.5) Intercept
##  student_t(3, 0, 2.5)        sd                                             0
##  student_t(3, 0, 2.5)        sd            exp_subject_id                   0
##  student_t(3, 0, 2.5)        sd  Intercept exp_subject_id                   0
##  student_t(3, 0, 2.5)     sigma                                             0
##        source
##       default
##  (vectorized)
##  (vectorized)
##       default
##       default
##  (vectorized)
##  (vectorized)
##       default
```

```
no1B <- brm(response ~ Musician + scramble + (1 | exp_subject_id), data = data_no1B,
            family = bernoulli(),
            prior = c(
              prior_intercept, prior_mus, set_prior('normal(-0.1, 1)', coef = 'scramble2B')
              ),
            save_pars = save_pars(all = TRUE), iter = 5000,
            file = 'models/E2_no1B')
```

```
plot(no1B)
```




```r
print(summary(no1B), digits = 4)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: response ~ Musician + scramble + (1 | exp_subject_id)
##    Data: data_no1B (Number of observations: 2104)
##   Draws: 4 chains, each with iter = 5000; warmup = 2500; thin = 1;
##          total post-warmup draws = 10000
##
## Multilevel Hyperparameters:
## ~exp_subject_id (Number of levels: 105)
##               Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## sd(Intercept)   0.1708    0.0843   0.0140   0.3284 1.0013     2069     2631
##
## Regression Coefficients:
##             Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## Intercept     0.7995    0.0862   0.6359   0.9701 1.0006     9772     6942
## MusicianNo   -0.0048    0.1004  -0.2056   0.1879 1.0012    10390     7624
## scramble2B   -0.4315    0.0913  -0.6143  -0.2526 1.0008    13723     7301
##
## Draws were sampled 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
get_prior(response ~ Musician*scramble + (1 | exp_subject_id), data = data_no1B)
```

```
## Warning: contrasts dropped from factor scramble due to missing levels
```

```
##                 prior     class                    coef          group resp dpar
```

```
##                (flat)         b
##                (flat)         b            MusicianNo
##                (flat)         b MusicianNo:scramble2B
##                (flat)         b            scramble2B
##  student_t(3, 0, 2.5) Intercept
##  student_t(3, 0, 2.5)        sd
##  student_t(3, 0, 2.5)        sd                         exp_subject_id
##  student_t(3, 0, 2.5)        sd             Intercept exp_subject_id
##  student_t(3, 0, 2.5)     sigma
##  nlpar lb ub       source
##                   default
##              (vectorized)
##              (vectorized)
##              (vectorized)
##                   default
##         0         default
##         0    (vectorized)
##         0    (vectorized)
##         0         default
```

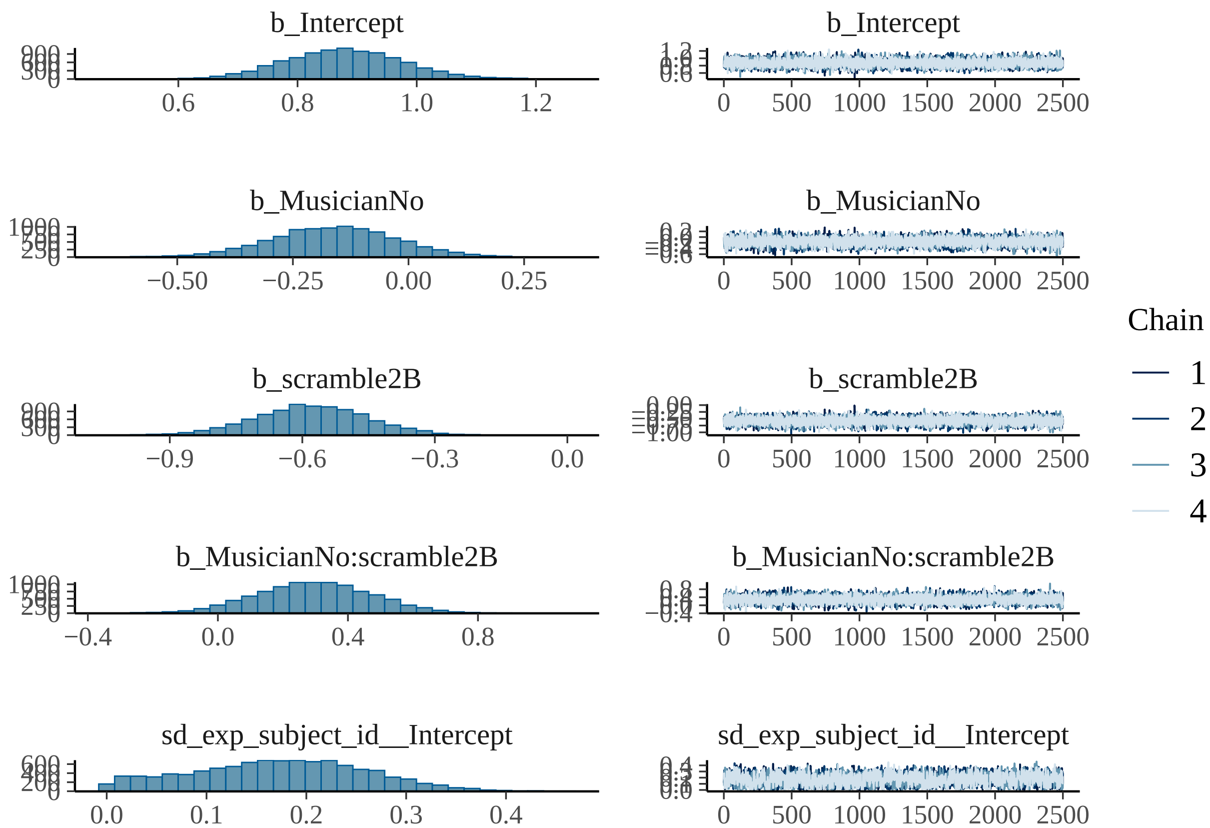
```
no1B_int <- brm(response ~ Musician*scramble + (1 | exp_subject_id), data = data_no1B,
                family = bernoulli(),
                prior = c(
                  prior_intercept, prior_mus,
                  set_prior('normal(-0.1, 1)', coef = 'scramble2B'),
                  set_prior('normal(0, 1)', coef = 'MusicianNo:scramble2B')
                  ),
                save_pars = save_pars(all = TRUE), iter = 5000,
                file = 'models/E2_no1B_int')
```

```
## Warning: contrasts dropped from factor scramble due to missing levels

## Compiling Stan program...

## Trying to compile a simple C file

## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## using C compiler: 'Apple clang version 16.0.0 (clang-1600.0.26.6)'
## using SDK: 'MacOSX15.2.sdk'
## clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Frame
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/src/Cor
##   679 | #include <cmath>
##       |          ^~~~~~~
## 1 error generated.
## make: *** [foo.o] Error 1

## Start sampling

##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000157 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.57 seconds.
```

```
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 3.598 seconds (Warm-up)
## Chain 1:                2.764 seconds (Sampling)
## Chain 1:                6.362 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 7.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.76 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 2: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 2: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 3.953 seconds (Warm-up)
## Chain 2:                2.761 seconds (Sampling)
## Chain 2:                6.714 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 7.5e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.75 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
```

```
## Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 3: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 3: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 3: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 3.664 seconds (Warm-up)
## Chain 3:                2.763 seconds (Sampling)
## Chain 3:                6.427 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 7.6e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.76 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 4: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 4: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 4: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 4.172 seconds (Warm-up)
## Chain 4:                2.796 seconds (Sampling)
## Chain 4:                6.968 seconds (Total)
## Chain 4:
```

```r
plot(no1B_int)
```




```r
print(summary(no1B_int), digits = 4)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: response ~ Musician * scramble + (1 | exp_subject_id)
##    Data: data_no1B (Number of observations: 2104)
##   Draws: 4 chains, each with iter = 5000; warmup = 2500; thin = 1;
##          total post-warmup draws = 10000
##
## Multilevel Hyperparameters:
## ~exp_subject_id (Number of levels: 105)
##               Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## sd(Intercept)   0.1714    0.0863   0.0125   0.3354 1.0029     1920     2082
##
## Regression Coefficients:
##                         Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS
## Intercept                 0.8786    0.0978   0.6925   1.0734 1.0003     6837
## MusicianNo               -0.1605    0.1351  -0.4267   0.1016 1.0009     6718
## scramble2B               -0.5794    0.1262  -0.8297  -0.3347 0.9999     7641
## MusicianNo:scramble2B     0.2971    0.1768  -0.0456   0.6387 1.0001     6225
##                         Tail_ESS
## Intercept                   5842
## MusicianNo                  6715
## scramble2B                  8120
## MusicianNo:scramble2B       7327
##
## Draws were sampled 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
BF_no1B_int <- bayes_factor(no1B_int, no1B)
```

```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
```

```r
print(BF_no1B_int)
```

```
## Estimated Bayes factor in favor of no1B_int over no1B: 0.80461
```

Still moderate evidence against an interaction between group and condition.

Years of experience

Keep only the subjects for which we have years of experience data and average accuracy per condition.

```
yrs_exp <- data %>%
  filter(!is.na(yrs_mus_exp)) %>%
  group_by(exp_subject_id, scramble, yrs_mus_exp) %>%
  summarize(count = n(),
            n_correct = sum(response),
            accuracy = n_correct / count)
```

```
## `summarise()` has grouped output by 'exp_subject_id', 'scramble'. You can
## override using the `.groups` argument.
```

Priors

For this analysis, we're operating on the scale of accuracy. Because we don't see ceiling effects (i.e. participants aren't getting too close to perfect accuracy), a linear model is appropriate enough.

```
these_priors <- c(
  set_prior('normal(0.75, 0.1)', class = 'Intercept'),
  set_prior('normal(-0.1, 0.1)', coef = 'scramble2'),
  set_prior('normal(-0.2, 0.1)', coef = 'scramble3'),
  set_prior('normal(0, 0.1)', coef = 'yrs_mus_exp')
)
```

Main model

```r
years_mus_scram <- brm(accuracy ~ scramble + yrs_mus_exp + (1|exp_subject_id), data = yrs_exp,
                       prior = these_priors,
                       save_pars = save_pars(all = TRUE), iter = 5000,
                       file = 'models/E2_years')
```

```
## Compiling Stan program...
```

```
## Trying to compile a simple C file
```

```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## using C compiler: 'Apple clang version 16.0.0 (clang-1600.0.26.6)'
## using SDK: 'MacOSX15.2.sdk'
## clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Frame
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/src/Cor
##   679 | #include <cmath>
##       |          ^~~~~~~
## 1 error generated.
## make: *** [foo.o] Error 1
```

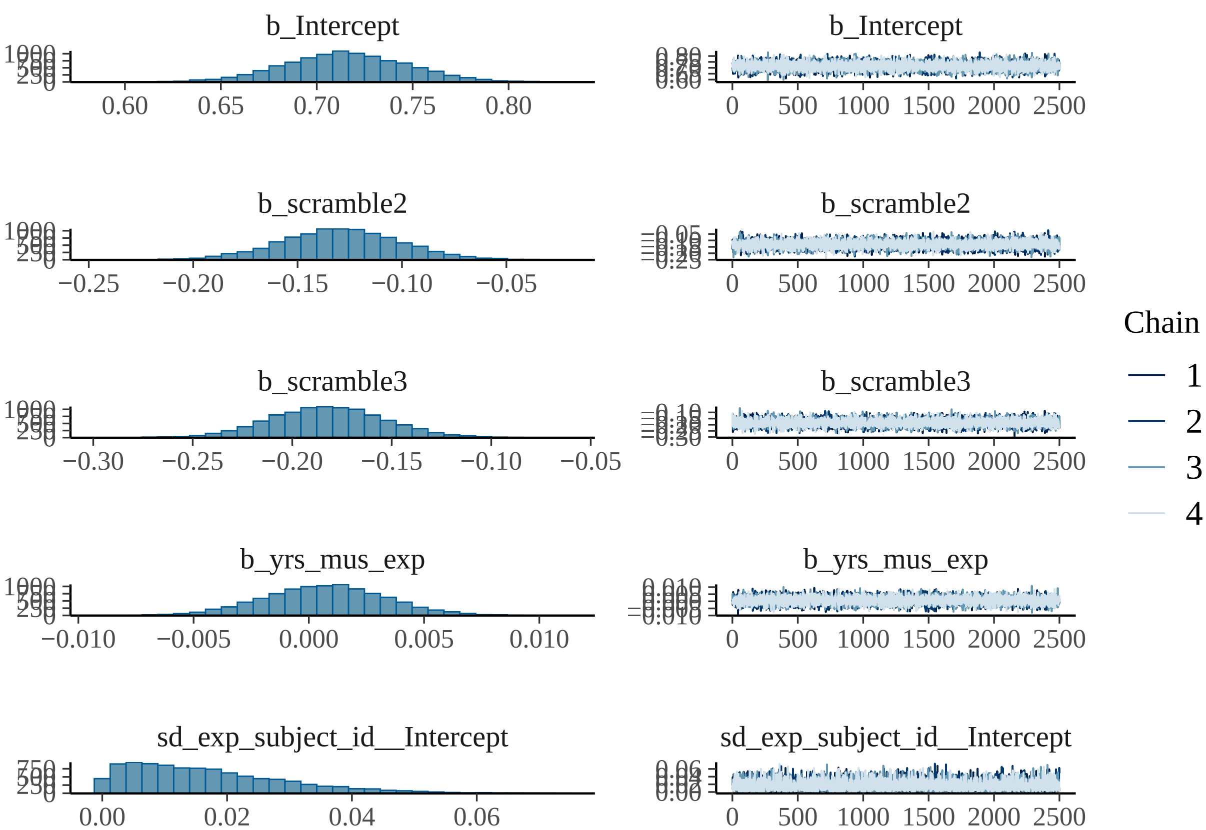
```
## Start sampling
```

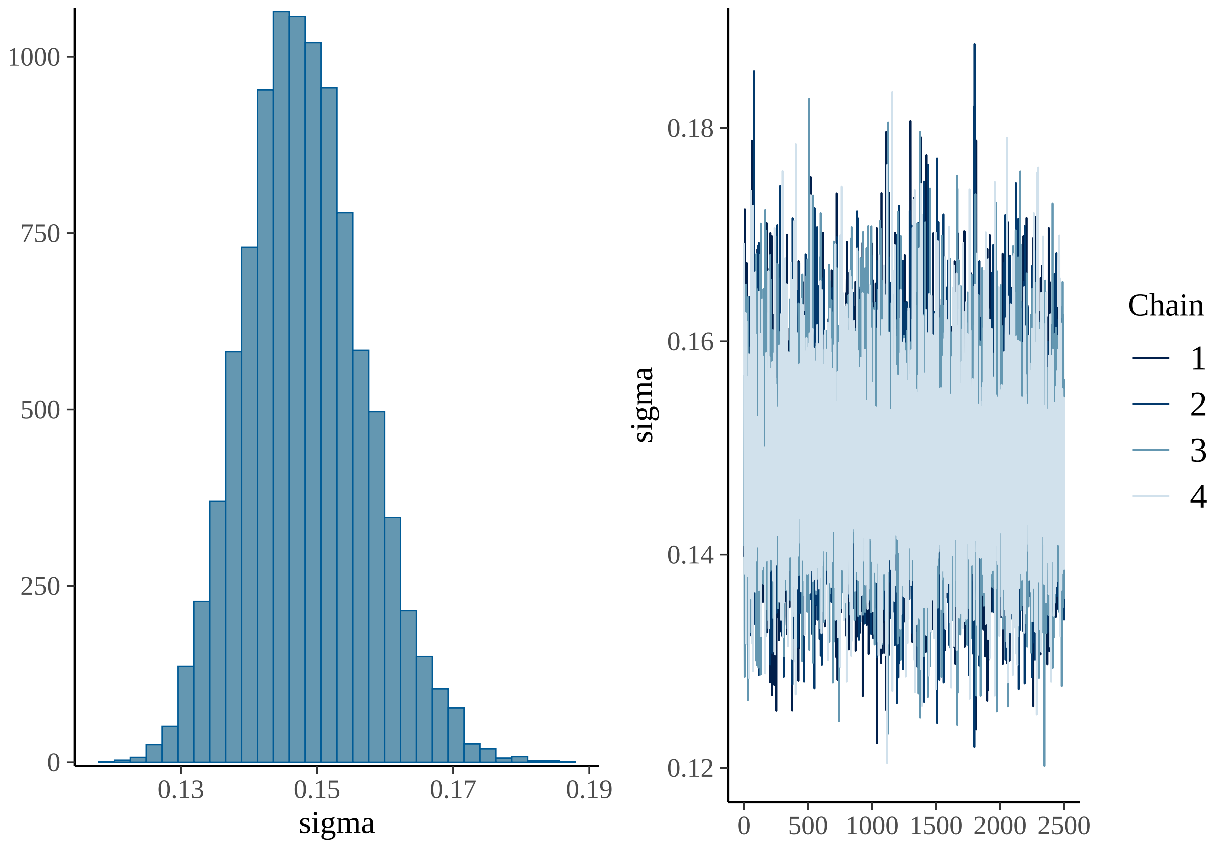
```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 5.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.54 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.671 seconds (Warm-up)
## Chain 1:                0.254 seconds (Sampling)
## Chain 1:                0.925 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 7e-06 seconds
```

```
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 2: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 2: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 1.09 seconds (Warm-up)
## Chain 2:                0.264 seconds (Sampling)
## Chain 2:                1.354 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.2e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.12 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 3: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 3: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 3: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.771 seconds (Warm-up)
## Chain 3:                0.25 seconds (Sampling)
## Chain 3:                1.021 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 7e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
```

```
## Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 4: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 4: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 4: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.705 seconds (Warm-up)
## Chain 4:                0.256 seconds (Sampling)
## Chain 4:                0.961 seconds (Total)
## Chain 4:
```

```
plot(years_mus_scram)
```






```
print(summary(years_mus_scram), digits = 5)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: accuracy ~ scramble + yrs_mus_exp + (1 | exp_subject_id)
##    Data: yrs_exp (Number of observations: 147)
##   Draws: 4 chains, each with iter = 5000; warmup = 2500; thin = 1;
##          total post-warmup draws = 10000
##
## Multilevel Hyperparameters:
## ~exp_subject_id (Number of levels: 49)
##               Estimate Est.Error l-95% CI u-95% CI    Rhat Bulk_ESS Tail_ESS
## sd(Intercept)  0.01696   0.01257  0.00071  0.04705 1.00025     6157     5414
##
## Regression Coefficients:
##             Estimate Est.Error l-95% CI u-95% CI    Rhat Bulk_ESS Tail_ESS
## Intercept    0.71535   0.03197  0.65312  0.77851 1.00018    14423     8004
## scramble2   -0.12931   0.02867 -0.18591 -0.07305 1.00061    15820     7708
## scramble3   -0.18209   0.02872 -0.23763 -0.12551 0.99990    14682     7968
## yrs_mus_exp  0.00070   0.00262 -0.00442  0.00586 1.00013    14731     7747
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI    Rhat Bulk_ESS Tail_ESS
## sigma  0.14825   0.00874  0.13233  0.16689 1.00030    17240     6916
##
## Draws were sampled 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).
```

Null model (for plotting purposes)

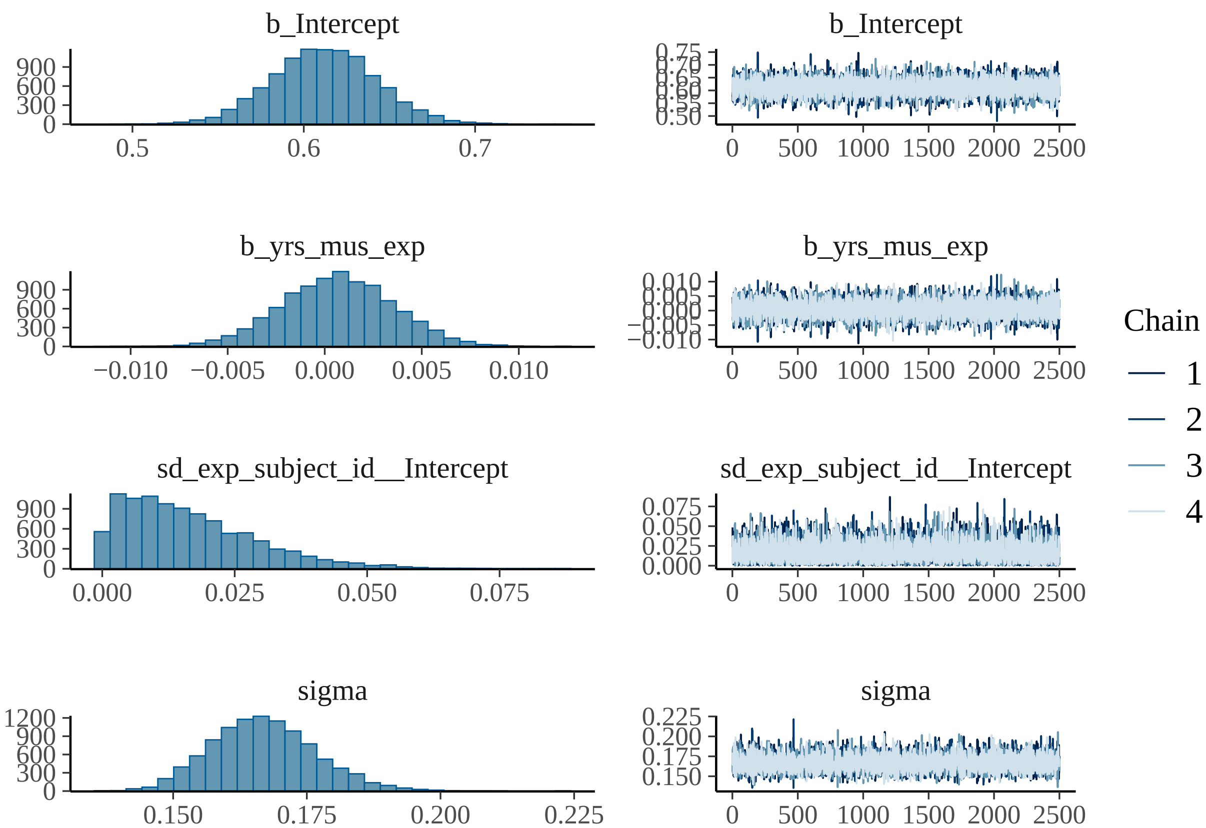
```r
years_mus <- brm(accuracy ~ yrs_mus_exp + (1|exp_subject_id), data = yrs_exp,
                 prior = c(
                   set_prior('normal(0.75, 0.1)', class = 'Intercept'),
                   set_prior('normal(0, 0.1)', coef = 'yrs_mus_exp')),
                 save_pars = save_pars(all = TRUE), iter = 5000,
                 file = 'models/E2_years_null')
```

```
## Compiling Stan program...

## Trying to compile a simple C file

## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## using C compiler: 'Apple clang version 16.0.0 (clang-1600.0.26.6)'
## using SDK: 'MacOSX15.2.sdk'
## clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Framew
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/src/Core
##   679 | #include <cmath>
##       |          ^~~~~~~
## 1 error generated.
## make: *** [foo.o] Error 1

## Start sampling

##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 5.5e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.55 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.67 seconds (Warm-up)
## Chain 1:                0.246 seconds (Sampling)
## Chain 1:                0.916 seconds (Total)
## Chain 1:
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 7e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 2: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 2: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.662 seconds (Warm-up)
## Chain 2:                0.251 seconds (Sampling)
## Chain 2:                0.913 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 9e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 3: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 3: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 3: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.646 seconds (Warm-up)
## Chain 3:                0.216 seconds (Sampling)
## Chain 3:                0.862 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 7e-06 seconds
```

```
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 4: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 4: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 4: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.647 seconds (Warm-up)
## Chain 4:                0.241 seconds (Sampling)
## Chain 4:                0.888 seconds (Total)
## Chain 4:
```

```
plot(years_mus)
```




```
print(summary(years_mus), digits = 4)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: accuracy ~ yrs_mus_exp + (1 | exp_subject_id)
```

```
##    Data: yrs_exp (Number of observations: 147)
##   Draws: 4 chains, each with iter = 5000; warmup = 2500; thin = 1;
##          total post-warmup draws = 10000
##
## Multilevel Hyperparameters:
## ~exp_subject_id (Number of levels: 49)
##               Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## sd(Intercept)   0.0166    0.0125   0.0006   0.0471 1.0003     6184     5124
##
## Regression Coefficients:
##             Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## Intercept     0.6120    0.0306   0.5531   0.6728 1.0000    15649     7815
## yrs_mus_exp   0.0007    0.0029  -0.0050   0.0063 0.9999    15083     6873
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## sigma   0.1670    0.0098   0.1494   0.1872 1.0005    17413     7186
##
## Draws were sampled 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
yrs_BF <- describe_posterior(years_mus_scram,
                             estimate = "median", dispersion = TRUE,
                             ci = .95, ci_method = "HDI",
                             test = c("bayes_factor"))
```

```
## Warning: Bayes factors might not be precise.
##   For precise Bayes factors, sampling at least 40,000 posterior samples is
##   recommended.
```

```r
print(yrs_BF, digits = 4)
```

```
## Summary of Posterior Distribution
##
## Parameter   |  Median |    MAD |          95% CI |       BF |  Rhat |        ESS
## -------------------------------------------------------------------------------
## (Intercept) |  0.7150 | 0.0319 | [ 0.65,  0.78] | 1.76e+23 | 1.000 | 14188.0000
## scramble2   | -0.1292 | 0.0284 | [-0.19, -0.07] |   790.53 | 1.000 | 15842.0000
## scramble3   | -0.1823 | 0.0286 | [-0.24, -0.13] | 2.30e+04 | 1.000 | 14605.0000
## yrs_mus_exp |  0.0007 | 0.0026 | [ 0.00,  0.01] |    0.028 | 1.000 | 14554.0000
```

```r
yrs_null_BF <- describe_posterior(years_mus,
                                  estimate = "median", dispersion = TRUE,
                                  ci = .95, ci_method = "HDI",
                                  test = c("bayes_factor"))
```

```
## Warning: Bayes factors might not be precise.
##   For precise Bayes factors, sampling at least 40,000 posterior samples is
##   recommended.
```

```r
print(yrs_null_BF, digits = 4)
```

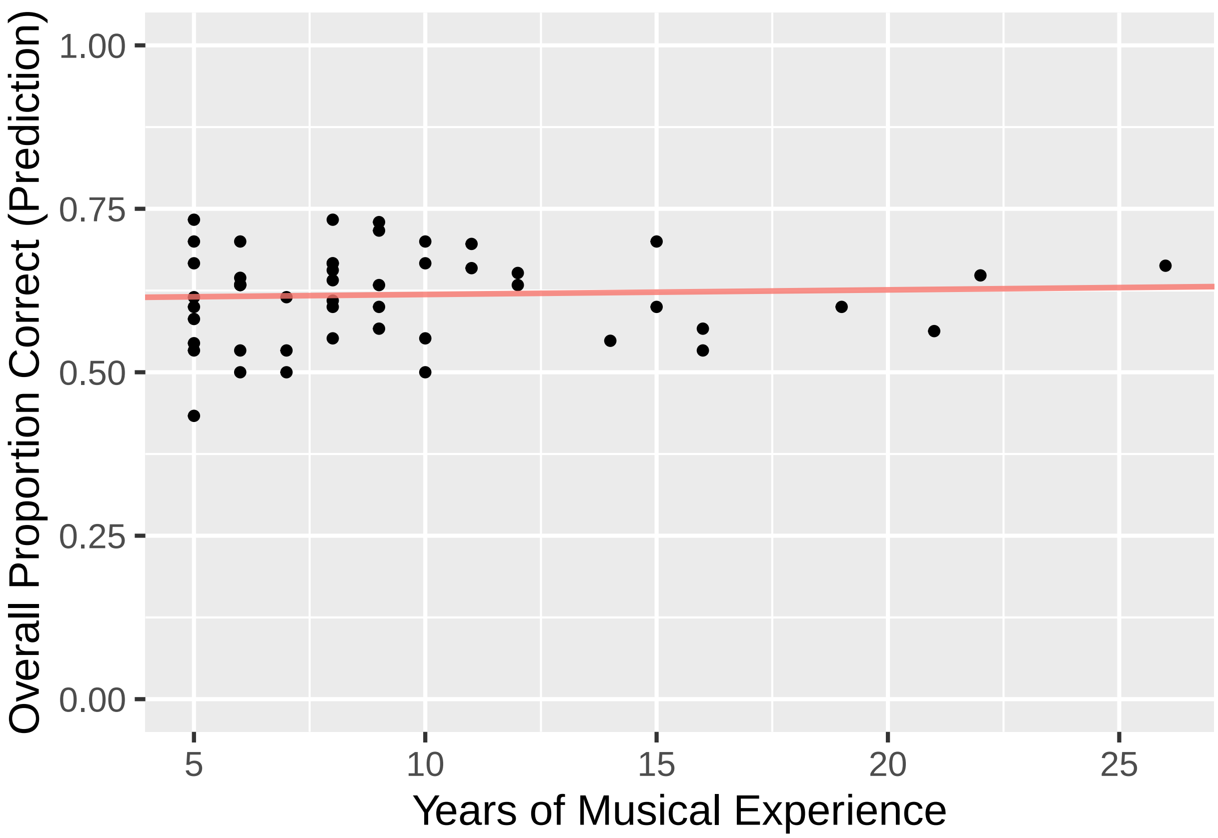
```
## Summary of Posterior Distribution
##
## Parameter   | Median |    MAD |        95% CI |       BF |  Rhat |        ESS
## -----------------------------------------------------------------------------
## (Intercept) | 0.6119 | 0.0303 | [ 0.55, 0.67] | 9.83e+20 | 1.000 | 15621.0000
## yrs_mus_exp | 0.0007 | 0.0029 | [ 0.00, 0.01] |    0.030 | 1.000 | 15081.0000
```

Strong evidence against an effect of years of musical experience.

Figure S1B

```
yrs_exp %>%
  group_by(exp_subject_id, yrs_mus_exp) %>%
  summarize(mean_acc = mean(accuracy)) %>%
  ggplot(aes(yrs_mus_exp, mean_acc)) +
  geom_point() +
  geom_abline(intercept = yrs_null_BF$Median[1], slope = yrs_null_BF$Median[2],
              color = '#F8766D', linewidth = 1, alpha = 0.8) +
  xlab('Years of Musical Experience') +
  ylab('Overall Proportion Correct (Prediction)') +
  scale_x_continuous(breaks = seq(5,30,5)) +
  scale_y_continuous(breaks = seq(0, 1, 0.1)) +
  ylim(0,1) +
  theme_gray(base_size = 16)
```

```
## `summarise()` has grouped output by 'exp_subject_id'. You can override using
## the `.groups` argument.
## Scale for y is already present. Adding another scale for y, which will replace
## the existing scale.
```




```
ggsave('../figures/FigS1B_prediction.png', width = 5, height = 5)
```