E3 rate

R. Cassano-Coleman

2025-09-06

This notebook takes response rates by subject, log-transforms them, and runs a Bayesian version of a mixed effects model.

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

Load the data.

```
data <- read_csv("../data/E3/response_rate_by_sub.csv")
```

```
## Rows: 380 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Musician, scramble
## dbl (3): exp_subject_id, stimulus_set, mean_response_rate
##
## 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.
```

Change musician and scramble into a factor.

```
data %<>% mutate(Musician = factor(Musician, levels = c('Yes', 'No')),
                 scramble = factor(scramble, levels = c('Intact', '8B', '2B', '1B')))
```

Set Intact as reference level.

```
contrasts(data$scramble) <- contr.treatment(4)
```

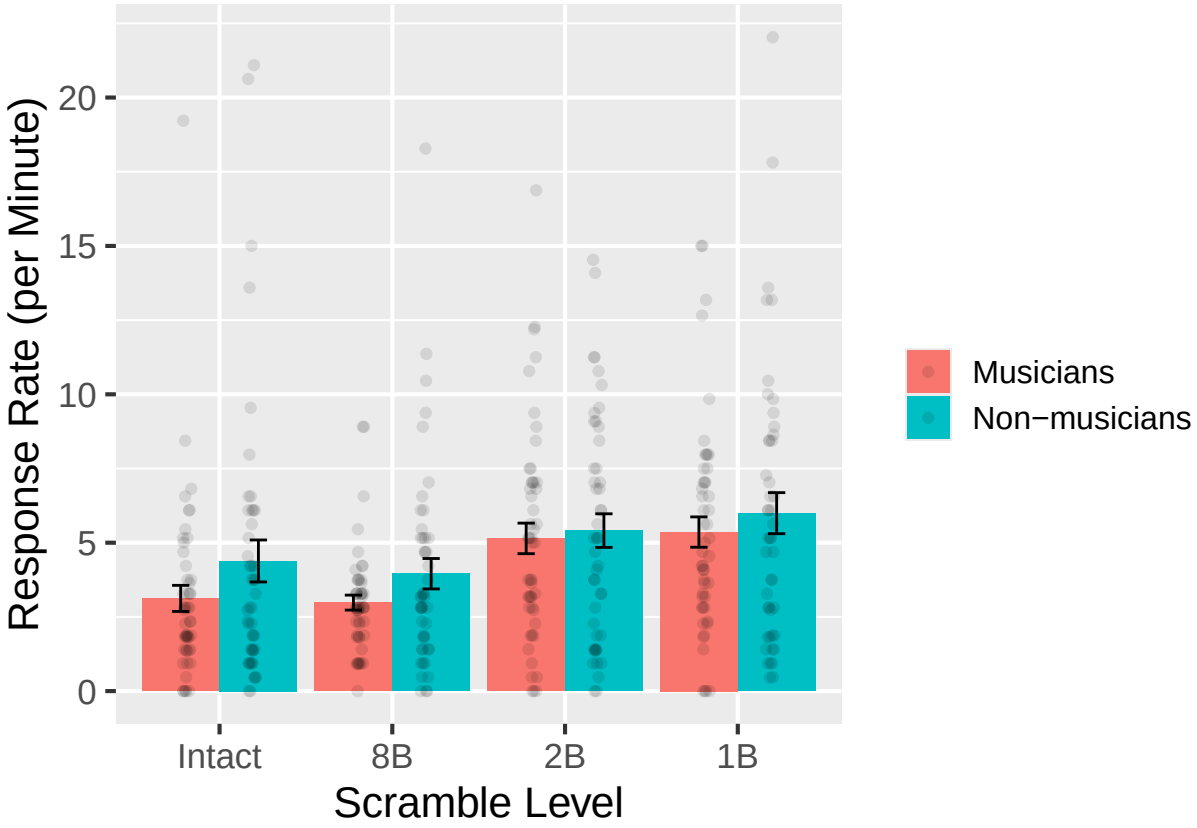
Check normality of the true response rate data. (Ignoring stimulus set.)

```
data %>%
  group_by(Musician, scramble) %>%
  shapiro_test(mean_response_rate)
```

```
## # A tibble: 8 x 5
##   Musician scramble variable           statistic            p
##   <fct>    <fct>    <chr>                  <dbl>        <dbl>
## 1 Yes      Intact   mean_response_rate     0.724 0.0000000283
## 2 Yes      8B       mean_response_rate     0.863 0.0000429
## 3 Yes      2B       mean_response_rate     0.941 0.0162
## 4 Yes      1B       mean_response_rate     0.929 0.00575
## 5 No       Intact   mean_response_rate     0.744 0.000000135
## 6 No       8B       mean_response_rate     0.839 0.0000157
## 7 No       2B       mean_response_rate     0.948 0.0401
## 8 No       1B       mean_response_rate     0.894 0.000530
```

```
data %>%
  ggplot(aes(x = scramble, y = mean_response_rate, fill = Musician)) +
  geom_bar(position = "dodge", stat = "summary", fun = mean) +
  geom_errorbar(position = position_dodge(width = 0.9), width = 0.2, stat = "summary") +
  geom_point(position = position_jitterdodge(jitter.width = 0.1), alpha = 0.1) +
  theme_gray(base_size = 16) +
  xlab('Scramble Level') +
  ylab('Response Rate (per Minute)') +
  scale_fill_discrete(name="", labels=c('Musicians', 'Non-musicians')) +
  theme(legend.text = element_text(size = 12))
```

```
## No summary function supplied, defaulting to `mean_se()`
```


Response Rate (per Minute)
20
15
10
5
0
Intact
8B
2B
1B
Scramble Level
Musicians
Non−musicians

Log-transform the rates and check for normality.

```
data %<>% mutate(log_rate = log(1 + mean_response_rate))
# add 1 so rates that are zero transform to 0 (rather than negative infinity)
```

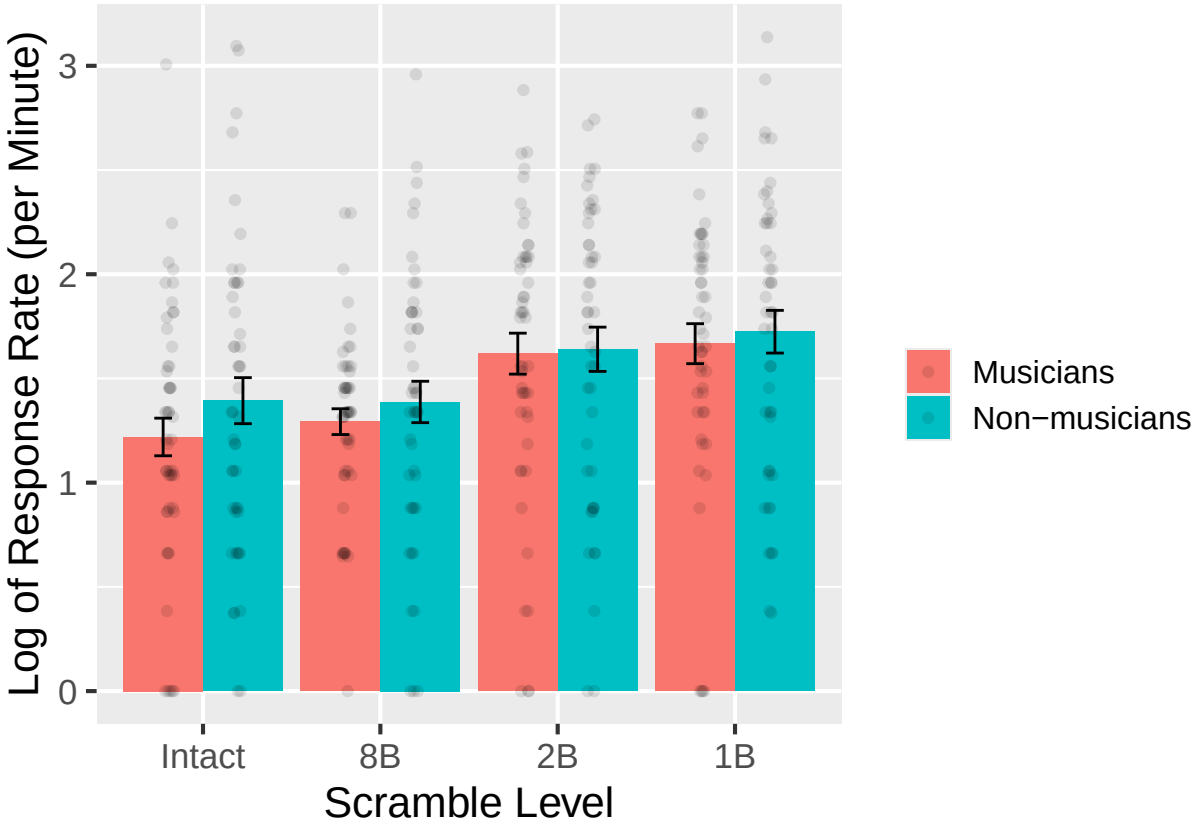
```
data %>%
  group_by(Musician, scramble) %>%
  shapiro_test(log_rate)
```

```
## # A tibble: 8 x 5
##   Musician scramble variable statistic       p
##   <fct>    <fct>    <chr>        <dbl>   <dbl>
## 1 Yes      Intact   log_rate     0.963 0.130
## 2 Yes      8B       log_rate     0.947 0.0284
## 3 Yes      2B       log_rate     0.943 0.0193
## 4 Yes      1B       log_rate     0.913 0.00153
## 5 No       Intact   log_rate     0.976 0.457
## 6 No       8B       log_rate     0.982 0.674
## 7 No       2B       log_rate     0.946 0.0319
## 8 No       1B       log_rate     0.973 0.358
```

Some of these are worse than others. Visualize:

```
data %>%
  ggplot(aes(x = scramble, y = log_rate, fill = Musician)) +
  geom_bar(position = "dodge", stat = "summary", fun = mean) +
  geom_errorbar(position = position_dodge(width = 0.9), width = 0.2, stat = "summary") +
  geom_point(position = position_jitterdodge(jitter.width = 0.1), alpha = 0.1) +
  theme_gray(base_size = 16) +
  xlab('Scramble Level') +
  ylab('Log of Response Rate (per Minute)') +
  scale_fill_discrete(name="", labels=c('Musicians', 'Non-musicians')) +
  theme(legend.text = element_text(size = 12))
```

```
## No summary function supplied, defaulting to `mean_se()`
```




```
#ggsave('log_rate.png', width = 7, height = 5)
```

It seems like this lack of normality is driven by the zero rates.

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

```
##                   prior     class       coef          group resp dpar nlpar lb
##                  (flat)         b                                              
##                  (flat)         b MusicianNo                                   
##                  (flat)         b  scramble2                                   
##                  (flat)         b  scramble3                                   
##                  (flat)         b  scramble4                                   
##  student_t(3, 1.5, 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
##  ub       source
##          default
##     (vectorized)
##     (vectorized)
##     (vectorized)
##     (vectorized)
##          default
##          default
##     (vectorized)
##     (vectorized)
##          default
```

```
these_priors <- c(
  set_prior('normal(0, 0.5)', coef = "MusicianNo"), # don't necessarily expect a difference between gro
  set_prior('normal(0, 0.5)', coef = "scramble2"), # intact vs 8B
  set_prior('normal(0, 0.5)', coef = "scramble3"), # intact vs 2B
  set_prior('normal(0, 0.5)', coef = "scramble4") # intact vs 1B
)
```

```
brm_log_rate <- brm(log_rate ~ Musician + scramble + (1|exp_subject_id), data = data,
                    prior = these_priors,
                    save_pars = save_pars(all = TRUE), iter = 5000,
                    file = 'models/E3_log_rate')
```

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

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

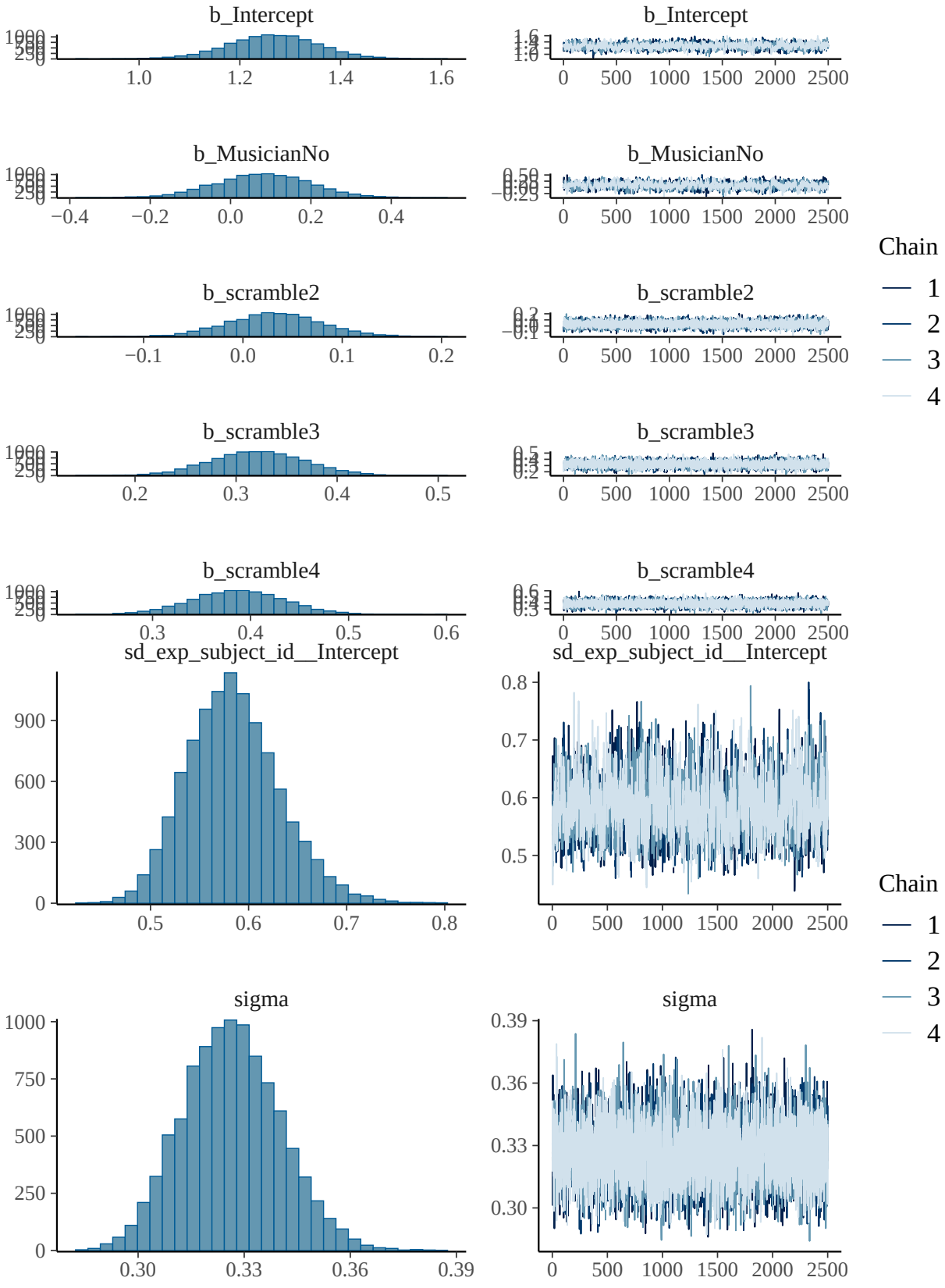
```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## using C compiler: 'Apple clang version 16.0.0 (clang-1600.0.26.6)'
## using SDK: 'MacOSX15.2.sdk'
## clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Framew
## In file included 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 7.2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.72 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: 0.644 seconds (Warm-up)
## Chain 1:                0.518 seconds (Sampling)
## Chain 1:                1.162 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.16 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: 0.643 seconds (Warm-up)
## Chain 2:                0.524 seconds (Sampling)
## Chain 2:                1.167 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.8e-05 seconds
```

```
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.18 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: 0.648 seconds (Warm-up)
## Chain 3:                0.526 seconds (Sampling)
## Chain 3:                1.174 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.4e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.14 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: 0.643 seconds (Warm-up)
## Chain 4:                0.518 seconds (Sampling)
## Chain 4:                1.161 seconds (Total)
## Chain 4:
```

```
plot(brm_log_rate)
```



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

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: log_rate ~ Musician + scramble + (1 | exp_subject_id)
##    Data: data (Number of observations: 380)
##   Draws: 4 chains, each with iter = 5000; warmup = 2500; thin = 1;
##          total post-warmup draws = 10000
##
## Multilevel Hyperparameters:
## ~exp_subject_id (Number of levels: 95)
##               Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## sd(Intercept)   0.5846    0.0472   0.5001   0.6858 1.0015     1633     3194
##
## Regression Coefficients:
##            Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## Intercept    1.2668    0.0894   1.0882   1.4409 1.0025     1204     2473
## MusicianNo   0.0803    0.1189  -0.1542   0.3128 1.0034     1202     2111
## scramble2    0.0325    0.0476  -0.0596   0.1270 1.0001     8674     7794
## scramble3    0.3220    0.0474   0.2295   0.4154 1.0000     8598     7619
## scramble4    0.3871    0.0475   0.2931   0.4805 1.0003     9029     8085
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## sigma   0.3265    0.0138   0.3007   0.3544 1.0004     8017     7831
##
## 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_log_rate <- emmeans(brm_log_rate, specs = c("scramble", "Musician"))
summary(emm_log_rate)
```

```
##  scramble Musician emmean lower.HPD upper.HPD
##  Intact   Yes        1.27      1.09      1.44
##  8B       Yes        1.30      1.13      1.47
##  2B       Yes        1.59      1.42      1.77
##  1B       Yes        1.66      1.48      1.83
##  Intact   No         1.35      1.18      1.53
##  8B       No         1.38      1.20      1.56
##  2B       No         1.67      1.48      1.84
##  1B       No         1.73      1.56      1.91
##
## Point estimate displayed: median
## HPD interval probability: 0.95
```

```r
emm_log_rate_s <- emmeans(brm_log_rate, specs = "scramble")
summary(emm_log_rate_s)
```

```
##  scramble emmean lower.HPD upper.HPD
##  Intact     1.31      1.17      1.44
##  8B         1.34      1.20      1.47
##  2B         1.63      1.50      1.76
##  1B         1.70      1.56      1.83
##
## Results are averaged over the levels of: Musician
## Point estimate displayed: median
## HPD interval probability: 0.95
```

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

```
##  contrast    estimate lower.HPD upper.HPD
##  Intact - 8B  -0.0323    -0.125    0.0610
##  Intact - 2B  -0.3216    -0.419   -0.2330
##  Intact - 1B  -0.3873    -0.478   -0.2915
##  8B - 2B      -0.2892    -0.381   -0.1980
##  8B - 1B      -0.3545    -0.446   -0.2634
##  2B - 1B      -0.0649    -0.154    0.0316
##
## Results are averaged over the levels of: Musician
## Point estimate displayed: median
## HPD interval probability: 0.95
```

```
log_rate_BF <- describe_posterior(brm_log_rate,
                                  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.
```

```
print(log_rate_BF, digits = 4)
```

```
## Summary of Posterior Distribution
##
## Parameter   | Median |    MAD |        95% CI |       BF |  Rhat |       ESS
## ------------------------------------------------------------------------------
## (Intercept) | 1.2671 | 0.0895 | [ 1.09, 1.44] | 6.79e+14 | 1.002 | 1228.0000
## MusicianNo  | 0.0817 | 0.1184 | [-0.15, 0.32] |    0.302 | 1.003 | 1201.0000
## scramble2   | 0.0323 | 0.0468 | [-0.06, 0.13] |    0.120 | 1.000 | 8636.0000
## scramble3   | 0.3216 | 0.0471 | [ 0.23, 0.42] | 1.29e+05 | 1.000 | 8568.0000
## scramble4   | 0.3873 | 0.0474 | [ 0.29, 0.48] | 2.15e+08 | 1.000 | 9090.0000
```

Compare the full model to a model without scramble condition.

```
brm_log_rate_null <- brm(log_rate ~ Musician + (1|exp_subject_id), data = data,
                         prior = set_prior('normal(0, 0.5)', class = 'b'),
                         save_pars = save_pars(all = TRUE), iter = 5000,
                         file = 'models/E3_log_rate_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/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
```

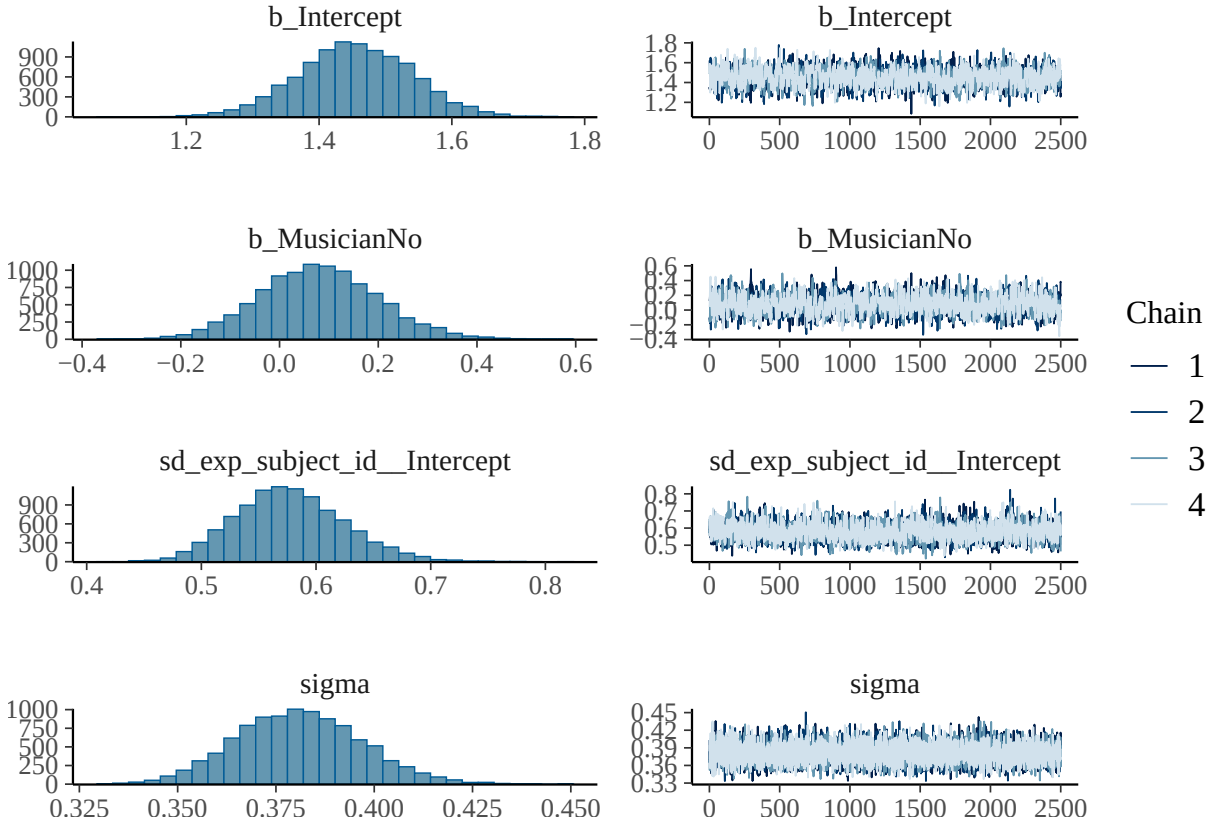
```
## Start sampling
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 6.2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.62 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: 0.597 seconds (Warm-up)
## Chain 1:                0.508 seconds (Sampling)
## Chain 1:                1.105 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.3e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.13 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: 0.597 seconds (Warm-up)
## Chain 2:                0.507 seconds (Sampling)
## Chain 2:                1.104 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.3e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.13 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: 0.592 seconds (Warm-up)
## Chain 3:                0.513 seconds (Sampling)
## Chain 3:                1.105 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.3e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.13 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: 0.583 seconds (Warm-up)
## Chain 4:                0.511 seconds (Sampling)
## Chain 4:                1.094 seconds (Total)
## Chain 4:
```

```
plot(brm_log_rate_null)
```




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

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: log_rate ~ Musician + (1 | exp_subject_id)
##    Data: data (Number of observations: 380)
##   Draws: 4 chains, each with iter = 5000; warmup = 2500; thin = 1;
##          total post-warmup draws = 10000
##
## Multilevel Hyperparameters:
```

```
## ~exp_subject_id (Number of levels: 95)
##               Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## sd(Intercept)   0.5775    0.0476   0.4915   0.6767 1.0011     2152     3874
##
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## Intercept    1.4546    0.0856   1.2863   1.6241 1.0014     1642     2751
## MusicianNo   0.0802    0.1204  -0.1490   0.3253 1.0020     1594     2875
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI   Rhat Bulk_ESS Tail_ESS
## sigma   0.3809    0.0160   0.3509   0.4132 1.0000     9504     7487
##
## 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_log_rate <- bayes_factor(brm_log_rate, brm_log_rate_null)
```

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

```
print(BF_log_rate)
```

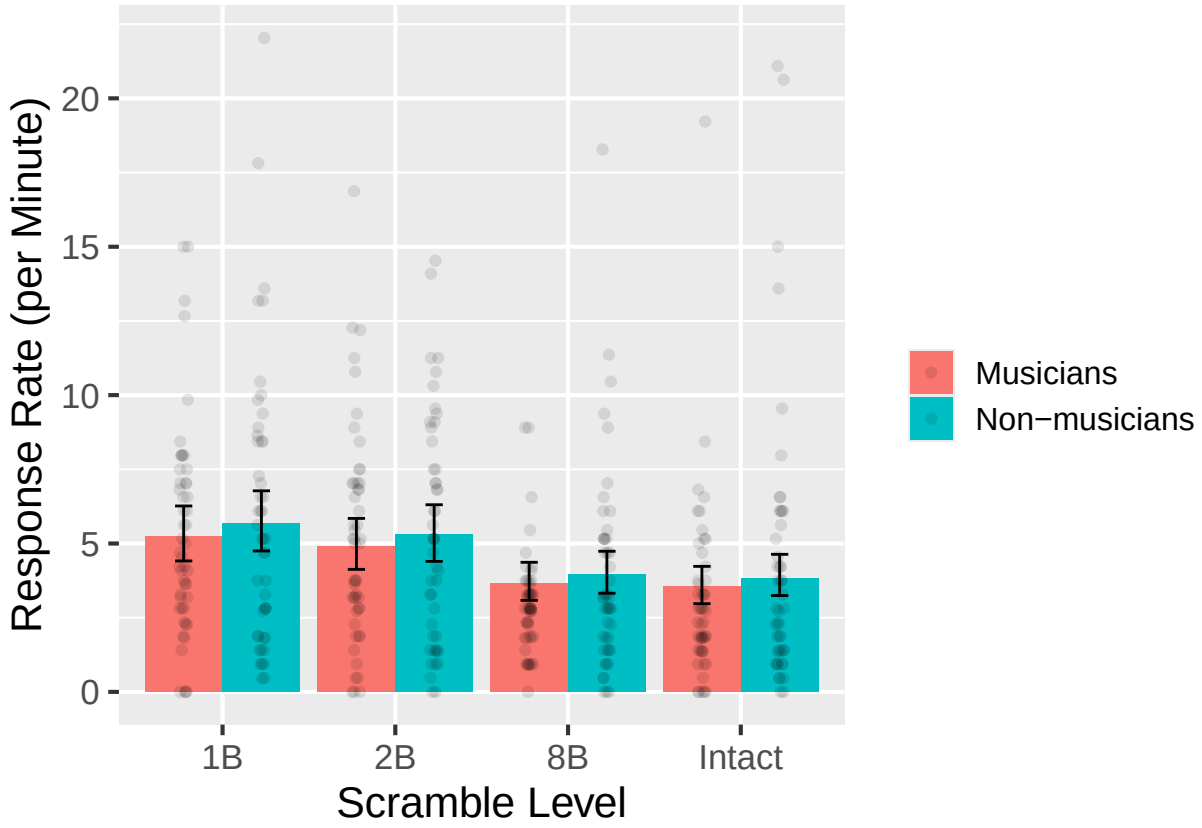
```
## Estimated Bayes factor in favor of brm_log_rate over brm_log_rate_null: 27179310654613700.00000
```

Visualize with posterior estimates and 95% CrI

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

```
ggplot() +
  geom_col(aes(x = scramble, y = exp(emmean), fill = Musician), data = posterior_est,
           position = "dodge") +
  geom_errorbar(aes(x = scramble, ymin = exp(lower.HPD), ymax = exp(upper.HPD), fill = Musician),
                data = posterior_est, position = position_dodge(width = 0.9), width = 0.2) +
  geom_point(aes(x = scramble, y = mean_response_rate, fill = Musician), data = data,
             position = position_jitterdodge(dodge.width = 0.9, jitter.width = 0.1), alpha = 0.1) +
  theme_gray(base_size = 16) +
  scale_x_discrete(limits = rev) +
  xlab('Scramble Level') +
  ylab('Response Rate (per Minute)') +
  scale_fill_discrete(name="", labels=c('Musicians', 'Non-musicians')) +
  theme(legend.text = element_text(size = 12))
```

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




```
ggsave('../figures/Fig3_rate.png', width = 7, height = 5)
```