A tutorial of Bayesian beta regressions with brms in R

Stefano Coretta[a,1,*]

[a] *University of Edinburgh, Linguistics and English Language, 3 George Sq, Edinburgh, EH8 9AD*

Abstract

This is the abstract. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum augue turpis, dictum non malesuada a, volutpat eget velit. Nam placerat turpis purus, eu tristique ex tincidunt et. Mauris sed augue eget turpis ultrices tincidunt. Sed et mi in leo porta egestas. Aliquam non laoreet velit. Nunc quis ex vitae eros aliquet auctor nec ac libero. Duis laoreet sapien eu mi luctus, in bibendum leo molestie. Sed hendrerit diam diam, ac dapibus nisl volutpat vitae. Aliquam bibendum varius libero, eu efficitur justo rutrum at. Sed at tempus elit.

Keywords: keyword1, keyword2

1. Introduction

Phonetic research often involves numeric continuous outcome variables, like durations, frequencies, loudness and ratios. Another commonly employed type of outcome variable are proportions: for example, proportion of voicing during closure, vocal folds contact quotient, gesture amplitude, nasalance. Moreover, virtually any measure can be MIN-MAX normalised, a procedure which transforms values so that they are in the range 0–1.

Regression models are very common, but there is a tendency of using Gaussian distribution families (i.e. probability distributions for the outcome variable) for anything that is numeric. A possible reason is that the base R function for fitting regression models, `lm()`, and the lme4 function used to fit regression models with varying terms, `lmer()`, both fit Gaussian regressions by default and the user does not have to specify the distribution family. This tacit defaulting to Gaussian models is also reflected in teaching practices, where test and models using the Gaussian distribution are the first to be taught, due to their relative simplicity and the fact that other models are generalisations of Gaussian models.

However, proportion are naturally not Gaussian, since they are limited between 0 and 1. The theoretical distribution that generates values with this characteristics is the beta distribution. Thus, regression models with proportions as the outcome variable should be fitted using a beta distribution as the distribution family. This tutorial introduces researchers to beta regression models in R using the package brms. Familiarity with regression modelling in R with a package like lme4 is assumed, but no prior knowledge of Bayesian statistics is necessary.

2. The beta distribution

...

*Corresponding author
Email address: `s.coretta@ed.ac.uk` (Stefano Coretta)
[1]This is the first author footnote.

3. Case study 1: voicing within consonant closure

For the first case study, we will model the proportion of voicing within consonant closure. The measurements come from a data set of audio and electroglottographic (EGG) recordings of 19 speakers of Northwestern Italian. The participants read frame sentences which included target words of the form /CVCo/, where /C/ was either /k, t, p/ in all permutations and /V/ was either /i, e, a, , u/ (two resulting words, /peto/ and /kako/ were excluded because they are profanities), for a total of 43 target words. There were 4 different frame sentence, with a total of 172 trials per participant (3,268 grand total). The actual observation count is 2,419, after removing speech errors, EGG measurement errors, and cases in which voicing ceased before the closure onset/after the closure offset of the second /C/.

The proportion of voicing during the closure of the second /C/ was calculated as the proportion of contiguous voicing duration after closure onset to total duration of closure. The following code chunk attaches the tidyverse packages (for reading and wrangling data) and loads the `ita_egg` tibble (data frame). The tibble is filtered so as to remove voicing proportions (`voi_clo_prop`) that are smaller than 0 and greater than 1. The variables `vowel` and `c2` are converted to factors to specify the order of the levels.

```r
# attach tidyverse and set light theme for plots
library(tidyverse)
theme_set(theme_light())

# load tibble
load("data/coretta2018/ita_egg.rda")

# filter and mutate data
ita_egg <- ita_egg |>
  filter(voi_clo_prop > 0, voi_clo_prop < 1) |>
  mutate(
    vowel = factor(vowel, levels = c("i", "e", "a", "o", "u")),
    c2 = factor(c2, levels = c("k", "t", "p"))
  )
```

Here is what the tibble looks like.

```r
ita_egg
```

```
# A tibble: 2,419 x 53
   speaker ipu    stimulus    sentence_ons sentence_off word_ons word_off v1_ons
   <chr>   <chr>  <chr>              <dbl>        <dbl>    <dbl>    <dbl>  <dbl>
 1 it01    ipu_1  Ripete 'po~         13.2         14.9     13.7     14.1   13.9
 2 it01    ipu_2  Ripete 'to~         16.9         18.6     17.4     17.9   17.5
 3 it01    ipu_3  Ripete 'pa~         20.2         21.9     20.7     21.1   20.8
 4 it01    ipu_4  Sentivo 't~         23.5         25.1     24.0     24.4   24.1
 5 it01    ipu_5  Sentivo 't~         26.3         27.8     26.8     27.2   26.9
 6 it01    ipu_6  Scrivete '~         29.2         30.9     29.7     30.1   29.8
 7 it01    ipu_7  Sentivo 'c~         32.1         33.6     32.6     33.1   32.8
 8 it01    ipu_8  Scrivete '~         35.0         36.6     35.5     35.9   35.6
 9 it01    ipu_9  Ha detto '~         41.9         43.5     42.3     42.7   42.4
10 it01    ipu_10 Sentivo 'p~         47.4         48.9     47.9     48.4   48.1
# i 2,409 more rows
# i 45 more variables: c2_ons <dbl>, v2_ons <dbl>, voice_ons <dbl>,
#   voice_off <dbl>, c1_rel <dbl>, c2_rel <dbl>, stimulus_id <dbl>,
#   sentence <chr>, word <chr>, c1 <chr>, vowel <fct>, c2 <fct>,
#   backness <chr>, height <fct>, c1_place <fct>, c2_place <fct>,
#   v1_duration <dbl>, c2_clos_duration <dbl>, rel_voff <dbl>,
#   sent_duration <dbl>, speech_rate <dbl>, speech_rate_c <dbl>, ...
```

Figure 1 shows the raw voicing duration proportion values split by vowel /i, e, a, , u/ and second consonant /k, t, p/ in the /CVCo/ target words. The plot suggests that, on average, the voicing proportion is slightly lower with /k/ than with /t, p/. Moreover, there is greater variability between vowels in /t, p/ than in /k/. We will use a beta regression model to assess these patterns. [“expectations” XXX]

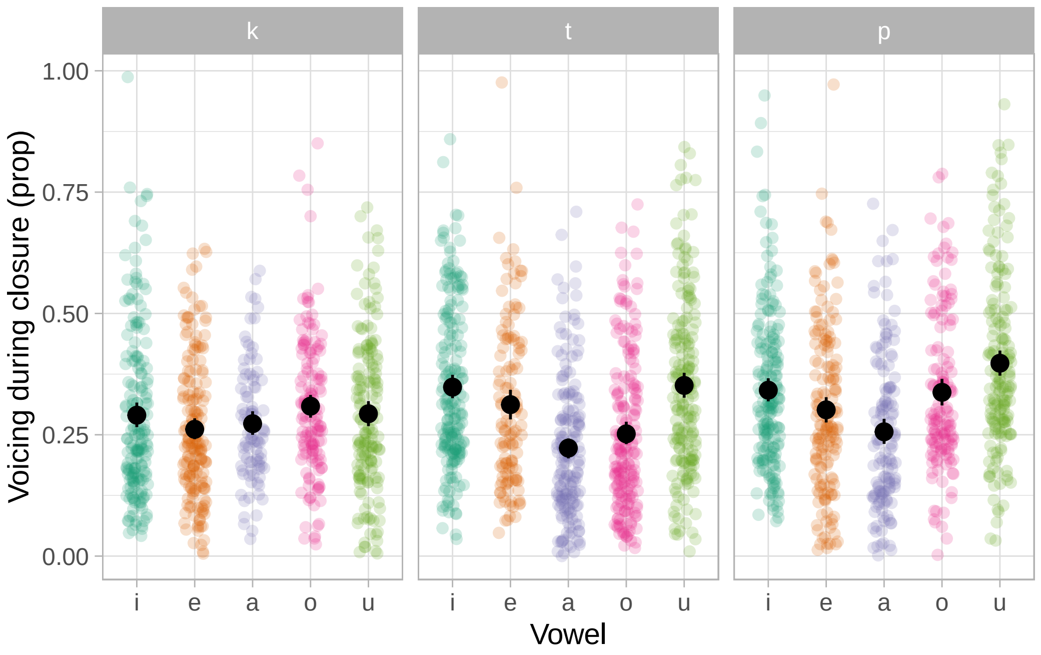


Figure 1: Proportion of voicing during the closure of the second consonant in /CVCo/ words by vowel and the second consonant.

We will use brms to fit Bayesian beta regressions. [XXX why Bayesian]. The model has voicing proportion as the outcome variable and the following terms: an interaction between vowel (/i, e, a, , u/) and second consonant C2 (/k, t, p/), centred speech rate (number of syllables per second); as varying (aka random) terms, by-speaker varying coefficients for the vowel/consonant interaction and for centred speech rate.[2] The categorical predictors vowel and C2 are coded using indexing rather than traditional R contrasts: in R, this corresponds to suppressing the model’s intercept with the `0 +` syntax; using indexing instead of contrasts makes it easier to specify priors. For pedagogical simplicity, the model will use the default priors, but note that in real data analyses contexts, priors should be specified by the user. I refer the readers to XXX.

```r
# attach brms
library(brms)

# fit the model
# Takes 3 minutes on MacBook Pro 2020, M1
```

[2]Footnote about Gelman’s terminology for random effects.

```
voi_prop_bm <- brm(
  # model formula
  voi_clo_prop ~
    # constant terms
    0 + vowel:c2 + speech_rate_c +
    # varying terms
    (0 + vowel:c2 + speech_rate_c | speaker),
  # uses the beta family for the outcome
  family = Beta,
  data = ita_egg,
  cores = 4,
  seed = 3749,
  file = "data/cache/voi_prop_bm"
)
```

The summary() function prints the full model summary. For conciseness, we will use the fixef() function which prints the regression coefficients. The full summary with an explanation of each part can be found in XXX. For each coefficient in the model, fixef() prints the name of the coefficient, the mean estimate, the estimate error and the lower and upper limits of a Bayesian Credible interval (CrI). Here, we print an 80% CrI. There is nothing special about 95% CrI within Bayesian inference and instead experts recommend to check and report a variety of CrIs. Bayesian CrIs indicate that at a certain probability levels the "true" estimate lies within that interval: so, for example, a 90% CrI [A, B] indicates that there is a 90% probability that the "true" estimate is between A and B. Different probability levels correspond to different levels of confidence: the higher the probability the higher the confidence (always conditional on data and model). Obtaining CrIs at different probability levels allows researchers to make more fine-grained inferential statements than the frequentist significance dichotomy affords. For simplicity of exposition, we will use 80% CrIs in this case study but we strongly recommend researchers to always obtain CrIs at different levels of probability and base their inferences on all and not one in particular. To reiterate, in Bayesian inference, an 80% CrI indicates the range of values within which the true estimate falls at 80% probability or confidence. We round all values to the nearest 2 digits for clarity.

```
round(
  fixef(voi_prop_bm, prob = c(0.1, 0.9)),
  digits = 2
)
```

```
              Estimate Est.Error   Q10   Q90
speech_rate_c     0.08      0.06  0.01  0.15
voweli:c2k       -0.91      0.14 -1.08 -0.74
vowele:c2k       -1.08      0.11 -1.22 -0.94
vowela:c2k       -0.99      0.12 -1.14 -0.84
vowelo:c2k       -0.79      0.14 -0.96 -0.62
vowelu:c2k       -1.00      0.16 -1.20 -0.80
voweli:c2t       -0.66      0.11 -0.79 -0.53
vowele:c2t       -0.84      0.14 -1.02 -0.66
vowela:c2t       -1.43      0.13 -1.60 -1.26
vowelo:c2t       -1.15      0.13 -1.31 -0.99
vowelu:c2t       -0.68      0.12 -0.83 -0.54
voweli:c2p       -0.68      0.11 -0.81 -0.54
vowele:c2p       -0.88      0.15 -1.07 -0.68
vowela:c2p       -1.14      0.13 -1.31 -0.98
```

```
vowelo:c2p       -0.66      0.11 -0.80 -0.53
vowelu:c2p       -0.44      0.12 -0.59 -0.28
```

The coefficients of a beta regression are estimated on the log-odds scale, as in Bernoulli/binomial (aka logistic) regressions. From the summary, we see that speech rate (number of syllables per second) has a positive effect on voicing proportion: the 80% CrI is between 0.01 and 0.15 log-odds [$\beta = 0.8$, SD = 0.06]. Log-odds can be converted to odd ratios by exponentiating the value: 0.01-0.15 log odds correspond to an odd ratio of 1.01 to 1.16, or as percentages, to an increase of voicing of 1 to 16% for every increase of one syllable per second. Since this is an 80% CrI, we can be 80% confident that the true effect of speech rate is between 1-16% increase of voicing proportion, conditional on the data and model. Note that transforming measures this way is appropriate *only* with quantile-based measures (like CrIs) but not with moments like the mean and standard deviation: to correctly get mean and SDs in the transformed scale, you must first extract the posterior draws (see below), convert them and then take moments such as mean and SD (for a more detailed explanation, see XXX).

Turning now to the coefficients for vowel and C2, given the indexing approach of coding these variables the model summary and the output of `fixef()` reports the *predictions* in log-odds for each combination of vowel and C2, rather than differences between levels. The CrIs of the vowel/C2 coefficients span all negative log-odds values: these correspond to proportions that are lower than 0.5 (which is 0 in log-odds). This matches the general trends in the raw data, which we plotted in Figure 1.

Next, we will plot the predicted proportions of each vowel/C2 combination at mean speech rate (i.e. centred speech rate = 0) and then calculate the average pair-wise difference in voicing proportion between /k, t, p/. Finally, we will assess whether there is greater between-vowel variability in /t, p/ relative to /k/.

Before being able to plot the predictions, it's important to get familiar with the so-called posterior draws. [Bayesian MCMC XXX]. Posterior draws can be conveniently obtained with the `as_draws_df()`. For the moment, we will extract only the draws of the constant regression coefficients (model variables starting with `b_`). To check which coefficients are available in a model, use `get_variables()` from the tidybayes package. `as_draws_df()` returns a tibble where each column contains the drawn values of a coefficient. The probability distribution of the drawn values of each coefficient is the posterior probability distribution of that coefficient. Note that, due to the indexing coding of vowel and C2, all coefficient except `b_speech_rate_c` are *predicted log-odds* for each vowel/C2 combination (the drawn values for `b_speech_rate_c` are drawn *differences* in log-odds for each unit increase of speech rate). The drawn values are in log-odds, but we can convert them to proportions with `plogis()` (we will do this when plotting below).

```r
# extract only coefficient variables starting with "b_"
voi_prop_bm_draws <- as_draws_df(voi_prop_bm, variable = "^b_", regex = TRUE)
voi_prop_bm_draws
```

```
# A draws_df: 1000 iterations, 4 chains, and 16 variables
   b_speech_rate_c b_voweli:c2k b_vowele:c2k b_vowela:c2k b_vowelo:c2k
1            0.159        -0.72        -1.05        -1.12        -1.02
2            0.035        -0.75        -1.39        -1.06        -0.89
3            0.126        -0.85        -0.98        -1.10        -0.64
4            0.064        -0.70        -1.07        -1.05        -0.83
5            0.032        -0.77        -1.18        -0.94        -0.70
6            0.073        -0.85        -1.23        -1.06        -0.77
7            0.101        -0.95        -1.07        -1.14        -0.86
8            0.074        -1.02        -1.07        -1.17        -0.83
9            0.087        -1.04        -1.07        -1.26        -0.79
10           0.092        -1.20        -1.04        -0.92        -0.64
```

```
   b_vowelu:c2k b_voweli:c2t b_vowele:c2t
1         -1.05        -0.62        -0.80
2         -1.04        -0.67        -0.99
3         -0.91        -0.66        -0.80
4         -1.07        -0.53        -0.87
5         -1.06        -0.63        -0.87
6         -1.00        -0.69        -0.73
7         -1.22        -0.74        -1.08
8         -1.22        -0.73        -1.04
9         -1.14        -0.66        -0.91
10        -0.84        -0.63        -0.61
# ... with 3990 more draws, and 8 more variables
# ... hidden reserved variables {'.chain', '.iteration', '.draw'}
```

We can now wrangle this tibble and plot the posterior distributions for each vowel/C2 combination.

```
voi_prop_bm_draws_long <- voi_prop_bm_draws |>
  # drop b_speech_rate_c before pivoting
  select(-b_speech_rate_c) |>
  # pivot vowel:c2 columns
  pivot_longer(`b_voweli:c2k`:`b_vowelu:c2p`, names_to = "coeff") |>
  # separate "coeff" labels into type ("b"), vowel and c2
  separate(coeff, into = c("type", "vowel", "c2"))
```

```
Warning: Dropping 'draws_df' class as required metadata was removed.
```

```
voi_prop_bm_draws_long
```

```
# A tibble: 60,000 x 7
   .chain .iteration .draw type  vowel  c2     value
    <int>      <int> <int> <chr> <chr>  <chr>  <dbl>
 1      1          1     1 b     voweli c2k   -0.721
 2      1          1     1 b     vowele c2k   -1.05
 3      1          1     1 b     vowela c2k   -1.12
 4      1          1     1 b     vowelo c2k   -1.02
 5      1          1     1 b     vowelu c2k   -1.05
 6      1          1     1 b     voweli c2t   -0.625
 7      1          1     1 b     vowele c2t   -0.795
 8      1          1     1 b     vowela c2t   -1.40
 9      1          1     1 b     vowelo c2t   -1.10
10      1          1     1 b     vowelu c2t   -0.668
# i 59,990 more rows
```

For plotting, we can use ggplot2 statistics layers from the ggdist package. `stat_halfeye()` plots the density of the posterior probability (in grey), its median (point) and CrIs (lines). Let's use 60 and 80% CrIs and transform the log-odds values to proportions with `plogis()`.

```
# attach ggdist package
library(ggdist)
```

```
Attaching package: 'ggdist'


The following objects are masked from 'package:brms':

    dstudent_t, pstudent_t, qstudent_t, rstudent_t
```

```
voi_prop_bm_draws_long |>
  ggplot(aes(plogis(value), vowel)) +
  stat_halfeye(.width = c(0.6, 0.8)) +
  facet_grid(rows = vars(c2))
```

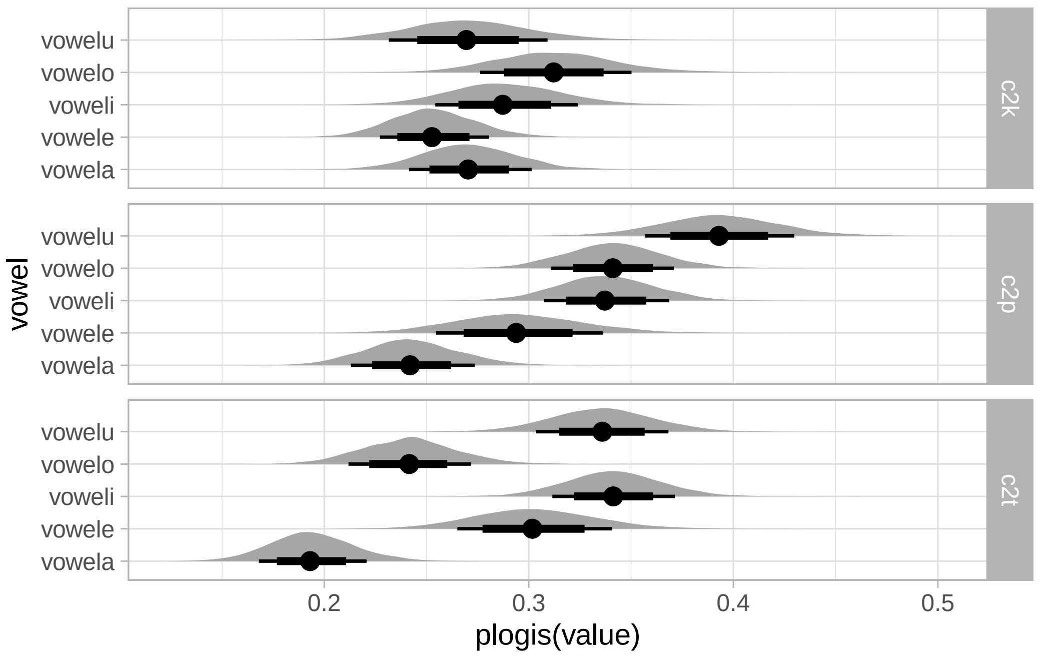


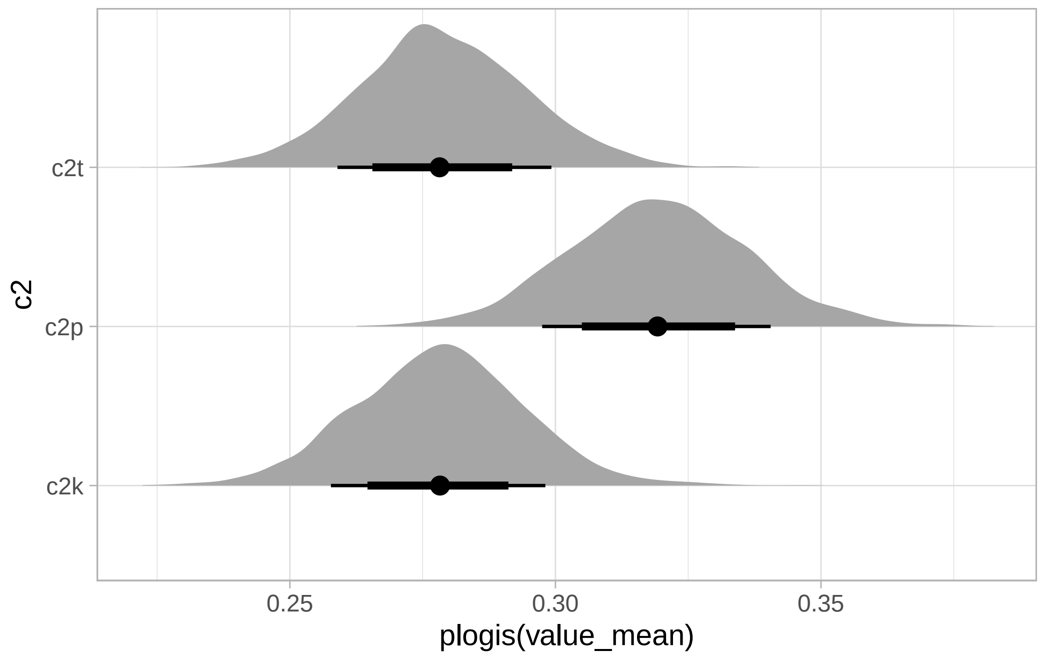
Figure 2

What if we want to plot the average predicted voicing proportion for the three consonants /k, t, p/? One approach is to take the mean across vowels within each consonant for each posterior draw, and the posterior distribution of the resulting list of values is the predicted posterior distribution of voicing proportion for each consonant, assuming an "average" vowel.

```
voi_prop_bm_draws_long_c2 <- voi_prop_bm_draws_long |>
  # grouping by .draw and c2 ensures that averaging applies only within draw and c2
  group_by(.draw, c2) |>
  # calculate the mean value within draw/c2
  summarise(
    value_mean = mean(value), .groups = "drop"
  )

voi_prop_bm_draws_long_c2 |>
```

```
  ggplot(aes(plogis(value_mean), c2)) +
  stat_halfeye(.width = c(0.6, 0.8))
```



Based on the posterior distributions of the mean voicing proportion by consonant, /p/ has a somewhat higher voicing proportion than /k/ and /t/. The real question is: how much higher? We can quantify this by taking the difference of the drawn values for /p/ and those for /t, k/. Since we want to compare /t, k/ to /p/, we should first average the draws of /t, k/ and then take the difference of the averaged draws and the draws of /p/.

```
voi_prop_bm_diff <- voi_prop_bm_draws_long_c2 |>
  # pivot data to create one column per consonant with the mean drawn values,
  # with one draw per raw
  pivot_wider(names_from = c2, values_from = value_mean) |>
  mutate(
    # calculate the mean of /k/ and /t/, for each draw
    c2tk = mean(c(c2k, c2t)),
    # calculate the difference of /p/ and /t, k/
    c2p_tk_diff = c2p - c2tk
  )

voi_prop_bm_diff
```

```
# A tibble: 4,000 x 6
   .draw    c2k    c2p    c2t   c2tk c2p_tk_diff
   <int>  <dbl>  <dbl>  <dbl>  <dbl>       <dbl>
 1     1 -0.992 -0.745 -0.917 -0.953       0.208
 2     2 -1.02  -0.790 -1.03  -0.953       0.163
 3     3 -0.895 -0.687 -0.890 -0.953       0.266
 4     4 -0.944 -0.814 -0.910 -0.953       0.140
 5     5 -0.928 -0.791 -0.920 -0.953      0.162 
 6     6 -0.983 -0.728 -0.926 -0.953      0.225 
 7     7 -1.05  -0.873 -1.08  -0.953      0.0802
 8     8 -1.06  -0.875 -1.07  -0.953      0.0782
 9     9 -1.06  -0.974 -1.05  -0.953     -0.0206
10    10 -0.929 -0.829 -0.953 -0.953      0.125 
# i 3,990 more rows
```

```
voi_prop_bm_diff |>
  ggplot(aes(c2p_tk_diff)) +
  stat_halfeye(.width = c(0.6, 0.8, 0.9)) +
  geom_vline(xintercept = 0)
```

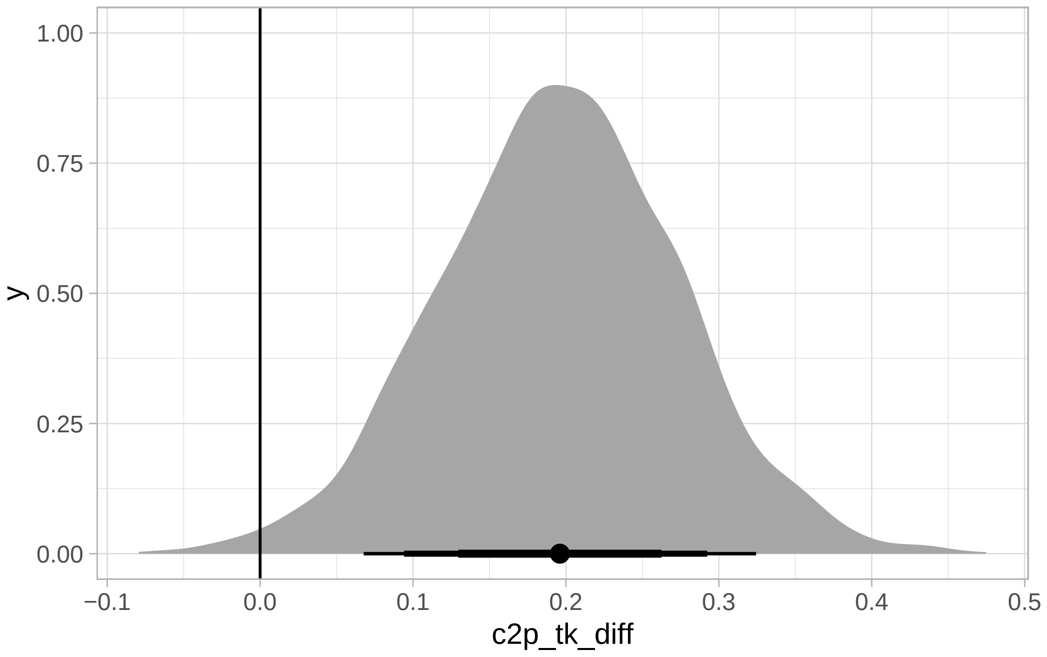


Figure 3

Once we have the posterior difference, we can obtain CrIs of the difference using `quantile2()` from the posterior package. Beware that the values of the difference are in log-odds! We can convert these into odd ratios with `exp()`.

```
library(posterior)
```

```
This is posterior version 1.6.0


Attaching package: 'posterior'

The following objects are masked from 'package:stats':

    mad, sd, var
```

```
The following objects are masked from 'package:base':

    %in%, match
```

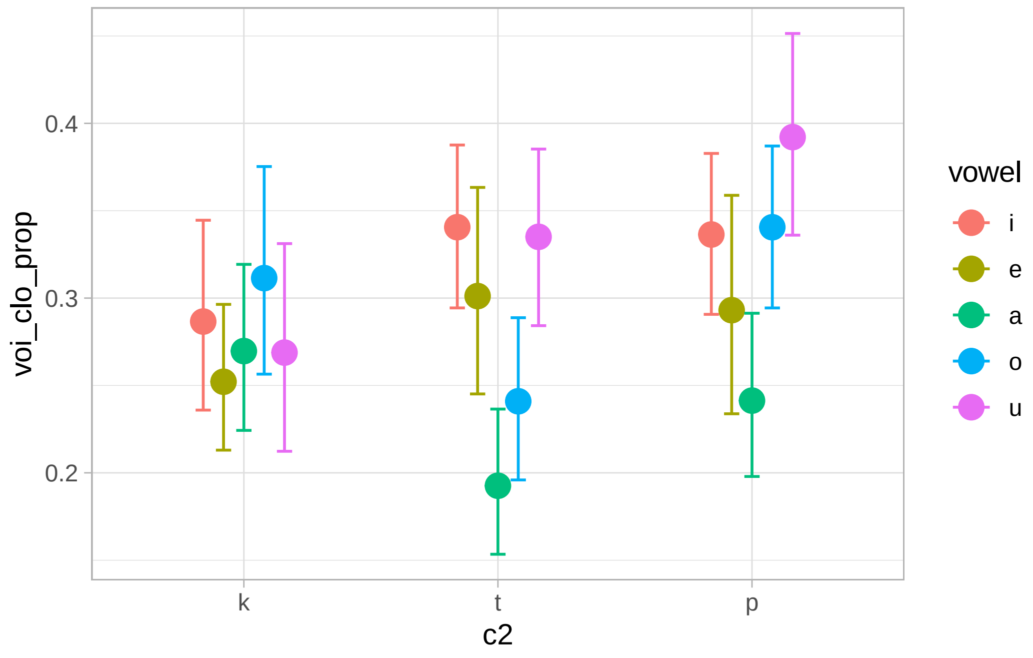
```r
voi_prop_bm_diff |>
  reframe(
    # 90% CrI
    q90 = quantile2(c2p_tk_diff, probs = c(0.05, 0.95)),
    # 80% CrI
    q80 = quantile2(c2p_tk_diff, probs = c(0.1, 0.9)),
    # 60% CrI
    q60 = quantile2(c2p_tk_diff, probs = c(0.2, 0.8)),
  ) |>
  # exponentiate to get odd ratios and round to 2 digits
  mutate(across(everything(), ~round(exp(.x), 2)))
```

```
# A tibble: 2 x 3
    q90   q80   q60
  <dbl> <dbl> <dbl>
1  1.07  1.1   1.14
2  1.38  1.34  1.3
```

So, based on the model and data, there is a 90% probability that the voicing proportion in /p/ is 1.07-1.38 times longer (or 7-38% increase) than in /t, k/. At 80% confidence, the change ratio is 1.10-1.34 (or 10-34% increase) while at 60% confidence is 1.14-1.30 (14-30% increase). In other words we can be quite confident that the voicing proportion in /p/ is longer than in /t, k/ and that the increase is less than 35%.

brms comes with a convenient function, `conditional_effects()`, to plot posterior means and CrIs based on predictors in the model. In the following example, we plot the predicted proportion of voicing by consonant and vowel.

```r
conditional_effects(voi_prop_bm, "c2:vowel")
```



Finally, the package marginaleffects has two other convenience functions that return CrIs of comparisons across predictor levels (`avg_comparisons()`) and CrIs of posterior predictions across predictor levels (`avg_predictions`). [XXX]

```r
library(marginaleffects)

avg_comparisons(voi_prop_bm, variables = list(c2 = "pairwise"), conf_level = 0.8, type = "link")
```

```

          Contrast Estimate    10.0 % 90.0 %
 mean(t) - mean(k)   0.0348 -0.00418 0.0738
 mean(p) - mean(k)   0.2176  0.17837 0.2558
 mean(p) - mean(t)   0.1829  0.14474 0.2192

Term: c2
Type:  link
Columns: term, contrast, estimate, conf.low, conf.high, predicted_lo, predicted_hi, predicted, tmp_idx
```

```r
avg_predictions(voi_prop_bm, variables = "vowel", conf_level = 0.8)
```

```

 vowel Estimate 10.0 % 90.0 %
     i    0.331  0.325  0.338
     e    0.300  0.293  0.307
     a    0.245  0.238  0.252
     o    0.305  0.298  0.312
     u    0.348  0.341  0.354

Type:  response
Columns: vowel, estimate, conf.low, conf.high
```

4. Case study 2: coarticulatory vowel nasalisation

For the second case study we will use data from Carignan et al. (2021). The study looked at properties of nasality in German VNC sequences. Here, we will focus on the effect of C voicing (voiceless /t/ vs voiced /d/) on the proportion of nasalisation within the vowel in the VNC sequence. Previous work on coarticulatory nasalisation in English has suggested that vowels followed by an NC sequence where C is voiceless should show earlier coarticulatory nasalisation than vowels followed by an NC sequence where C is voiced (see review in Carignan et al., 2021).

```
nasal <- read_csv("data/carignan2021/nasal.csv")
```

```
Rows: 643 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (5): speaker, label, vowel, post, voicing
dbl (1): nas_prop

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.
```

```
nas_prop_bm <- brm(
  nas_prop ~ 0 + voicing:vowel + (0 + voicing:vowel | speaker),
  data = nasal,
  family = Beta,
  cores = 4,
  seed = 3749,
  file = "data/cache/nas_prop_bm"
)
```

```
summary(nas_prop_bm, prob = 0.8)
```

```
 Family: beta
  Links: mu = logit; phi = identity
Formula: nas_prop ~ 0 + voicing:vowel + (0 + voicing:vowel | speaker)
   Data: nasal (Number of observations: 643)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Multilevel Hyperparameters:
~speaker (Number of levels: 35)
                                                       Estimate Est.Error
sd(voicingvoiced:vowela_)                                  0.45      0.13
sd(voicingvoiceless:vowela_)                               0.37      0.10
sd(voicingvoiced:vowelE_)                                  0.56      0.18
sd(voicingvoiceless:vowelE_)                               0.34      0.13
sd(voicingvoiced:vowelI_)                                  0.71      0.18
sd(voicingvoiceless:vowelI_)                               0.83      0.16
sd(voicingvoiced:vowelO_)                                  0.36      0.22
sd(voicingvoiceless:vowelO_)                               0.24      0.18
cor(voicingvoiced:vowela,_voicingvoiceless:vowela_)        0.49      0.22
cor(voicingvoiced:vowela,_voicingvoiced:vowelE_)           0.29      0.25
cor(voicingvoiceless:vowela,_voicingvoiced:vowelE_)        0.16      0.25
cor(voicingvoiced:vowela,_voicingvoiceless:vowelE_)        0.25      0.27
cor(voicingvoiceless:vowela,_voicingvoiceless:vowelE_)     0.42      0.25
cor(voicingvoiced:vowelE,_voicingvoiceless:vowelE_)        0.17      0.28
cor(voicingvoiced:vowela,_voicingvoiced:vowelI_)           0.15      0.26
cor(voicingvoiceless:vowela,_voicingvoiced:vowelI_)        0.17      0.26
cor(voicingvoiced:vowelE,_voicingvoiced:vowelI_)           0.23      0.26
cor(voicingvoiceless:vowelE,_voicingvoiced:vowelI_)        0.29      0.27
cor(voicingvoiced:vowela,_voicingvoiceless:vowelI_)       -0.11      0.24
cor(voicingvoiceless:vowela,_voicingvoiceless:vowelI_)     0.16      0.23
cor(voicingvoiced:vowelE,_voicingvoiceless:vowelI_)        0.14      0.25
cor(voicingvoiceless:vowelE,_voicingvoiceless:vowelI_)     0.33      0.25
cor(voicingvoiced:vowelI,_voicingvoiceless:vowelI_)        0.34      0.23
cor(voicingvoiced:vowela,_voicingvoiced:vowelO_)           0.20      0.31
cor(voicingvoiceless:vowela,_voicingvoiced:vowelO_)        0.13      0.31
cor(voicingvoiced:vowelE,_voicingvoiced:vowelO_)           0.10      0.31
cor(voicingvoiceless:vowelE,_voicingvoiced:vowelO_)        0.07      0.31
cor(voicingvoiced:vowelI,_voicingvoiced:vowelO_)           0.14      0.31
cor(voicingvoiceless:vowelI,_voicingvoiced:vowelO_)       -0.06      0.29
cor(voicingvoiced:vowela,_voicingvoiceless:vowelO_)        0.03      0.32
cor(voicingvoiceless:vowela,_voicingvoiceless:vowelO_)    -0.05      0.32
cor(voicingvoiced:vowelE,_voicingvoiceless:vowelO_)        0.09      0.32
cor(voicingvoiceless:vowelE,_voicingvoiceless:vowelO_)     0.03      0.33
cor(voicingvoiced:vowelI,_voicingvoiceless:vowelO_)        0.12      0.32
cor(voicingvoiceless:vowelI,_voicingvoiceless:vowelO_)     0.03      0.31
cor(voicingvoiced:vowelO,_voicingvoiceless:vowelO_)        0.08      0.33
                                                        l-80% CI u-80% CI Rhat
sd(voicingvoiced:vowela_)                                  0.29     0.62 1.00
sd(voicingvoiceless:vowela_)                               0.25     0.50 1.00
sd(voicingvoiced:vowelE_)                                  0.33     0.79 1.00
sd(voicingvoiceless:vowelE_)                               0.18     0.50 1.00
sd(voicingvoiced:vowelI_)                                  0.50     0.96 1.00
sd(voicingvoiceless:vowelI_)                               0.64     1.04 1.00
sd(voicingvoiced:vowelO_)                                  0.08     0.66 1.00
sd(voicingvoiceless:vowelO_)                               0.04     0.49 1.00
cor(voicingvoiced:vowela,_voicingvoiceless:vowela_)        0.20     0.76 1.00
cor(voicingvoiced:vowela,_voicingvoiced:vowelE_)          -0.05     0.60 1.00
cor(voicingvoiceless:vowela,_voicingvoiced:vowelE_)       -0.17     0.50 1.00
cor(voicingvoiced:vowela,_voicingvoiceless:vowelE_)       -0.10     0.59 1.00
cor(voicingvoiceless:vowela,_voicingvoiceless:vowelE_)     0.09     0.72 1.00
cor(voicingvoiced:vowelE,_voicingvoiceless:vowelE_)       -0.22     0.53 1.00
cor(voicingvoiced:vowela,_voicingvoiced:vowelI_)          -0.20     0.49 1.00
cor(voicingvoiceless:vowela,_voicingvoiced:vowelI_)       -0.18     0.50 1.00
cor(voicingvoiced:vowelE,_voicingvoiced:vowelI_)          -0.11     0.56 1.00
cor(voicingvoiceless:vowelE,_voicingvoiced:vowelI_)       -0.07     0.62 1.00
cor(voicingvoiced:vowela,_voicingvoiceless:vowelI_)       -0.42     0.20 1.00
cor(voicingvoiceless:vowela,_voicingvoiceless:vowelI_)    -0.13     0.46 1.00
cor(voicingvoiced:vowelE,_voicingvoiceless:vowelI_)       -0.19     0.44 1.00
cor(voicingvoiceless:vowelE,_voicingvoiceless:vowelI_)     0.00     0.63 1.00
cor(voicingvoiced:vowelI,_voicingvoiceless:vowelI_)        0.03     0.62 1.00
cor(voicingvoiced:vowela,_voicingvoiced:vowelO_)          -0.22     0.59 1.00
cor(voicingvoiceless:vowela,_voicingvoiced:vowelO_)       -0.27     0.53 1.00
cor(voicingvoiced:vowelE,_voicingvoiced:vowelO_)          -0.33     0.50 1.00
cor(voicingvoiceless:vowelE,_voicingvoiced:vowelO_)       -0.34     0.47 1.00
cor(voicingvoiced:vowelI,_voicingvoiced:vowelO_)          -0.27     0.52 1.00
cor(voicingvoiceless:vowelI,_voicingvoiced:vowelO_)       -0.44     0.33 1.00
cor(voicingvoiced:vowela,_voicingvoiceless:vowelO_)       -0.40     0.45 1.00
cor(voicingvoiceless:vowela,_voicingvoiceless:vowelO_)    -0.46     0.37 1.00
cor(voicingvoiced:vowelE,_voicingvoiceless:vowelO_)       -0.34     0.51 1.00
cor(voicingvoiceless:vowelE,_voicingvoiceless:vowelO_)    -0.41     0.46 1.00
cor(voicingvoiced:vowelI,_voicingvoiceless:vowelO_)       -0.32     0.54 1.00
cor(voicingvoiceless:vowelI,_voicingvoiceless:vowelO_)    -0.39     0.43 1.00
cor(voicingvoiced:vowelO,_voicingvoiceless:vowelO_)       -0.37     0.50 1.00
                                                        Bulk_ESS Tail_ESS
sd(voicingvoiced:vowela_)                                   1686     1641
sd(voicingvoiceless:vowela_)                                2099     2574
sd(voicingvoiced:vowelE_)                                   1536     1551
sd(voicingvoiceless:vowelE_)                                2319     1734
sd(voicingvoiced:vowelI_)                                   2498     3094
sd(voicingvoiceless:vowelI_)                                3333     3115
sd(voicingvoiced:vowelO_)                                   1620     1698
sd(voicingvoiceless:vowelO_)                                2295     2196
cor(voicingvoiced:vowela,_voicingvoiceless:vowela_)         2144     2507
cor(voicingvoiced:vowela,_voicingvoiced:vowelE_)            2466     2751
cor(voicingvoiceless:vowela,_voicingvoiced:vowelE_)         2821     3574
cor(voicingvoiced:vowela,_voicingvoiceless:vowelE_)         3289     3353
cor(voicingvoiceless:vowela,_voicingvoiceless:vowelE_)      3534     3625
cor(voicingvoiced:vowelE,_voicingvoiceless:vowelE_)         4027     3622
cor(voicingvoiced:vowela,_voicingvoiced:vowelI_)            2057     2599
cor(voicingvoiceless:vowela,_voicingvoiced:vowelI_)         2350     2928
cor(voicingvoiced:vowelE,_voicingvoiced:vowelI_)            2143     2832
cor(voicingvoiceless:vowelE,_voicingvoiced:vowelI_)         1964     3115
cor(voicingvoiced:vowela,_voicingvoiceless:vowelI_)         1792     2459
cor(voicingvoiceless:vowela,_voicingvoiceless:vowelI_)      2219     2880
cor(voicingvoiced:vowelE,_voicingvoiceless:vowelI_)         2232     2731
cor(voicingvoiceless:vowelE,_voicingvoiceless:vowelI_)      1602     2249
cor(voicingvoiced:vowelI,_voicingvoiceless:vowelI_)         2271     3182
cor(voicingvoiced:vowela,_voicingvoiced:vowelO_)            3973     3016
cor(voicingvoiceless:vowela,_voicingvoiced:vowelO_)         5614     3512
cor(voicingvoiced:vowelE,_voicingvoiced:vowelO_)            4179     3070
cor(voicingvoiceless:vowelE,_voicingvoiced:vowelO_)         4760     3338
cor(voicingvoiced:vowelI,_voicingvoiced:vowelO_)            4135     3265
cor(voicingvoiceless:vowelI,_voicingvoiced:vowelO_)         4320     3583
cor(voicingvoiced:vowela,_voicingvoiceless:vowelO_)         7265     3088
cor(voicingvoiceless:vowela,_voicingvoiceless:vowelO_)      5934     3240
cor(voicingvoiced:vowelE,_voicingvoiceless:vowelO_)         6140     3465
cor(voicingvoiceless:vowelE,_voicingvoiceless:vowelO_)      5229     3304
cor(voicingvoiced:vowelI,_voicingvoiceless:vowelO_)         4597     3612
cor(voicingvoiceless:vowelI,_voicingvoiceless:vowelO_)      6014     3479
cor(voicingvoiced:vowelO,_voicingvoiceless:vowelO_)         3033     3351

Regression Coefficients:
                         Estimate Est.Error l-80% CI u-80% CI Rhat Bulk_ESS
```

```
voicingvoiced:vowela_       -0.61      0.12    -0.76    -0.45 1.00     3695
voicingvoiceless:vowela_    -0.07      0.09    -0.19     0.04 1.00     3582
voicingvoiced:vowelE_       -0.19      0.15    -0.38     0.01 1.00     3940
voicingvoiceless:vowelE_    -0.25      0.10    -0.38    -0.12 1.00     5156
voicingvoiced:vowelI_       -0.74      0.19    -0.98    -0.51 1.00     4090
voicingvoiceless:vowelI_    -0.12      0.17    -0.33     0.10 1.00     4077
voicingvoiced:vowelO_       -0.33      0.16    -0.54    -0.13 1.00     6095
voicingvoiceless:vowelO_     0.25      0.16     0.05     0.45 1.00     6834
                         Tail_ESS
voicingvoiced:vowela_        3160
voicingvoiceless:vowela_     3300
voicingvoiced:vowelE_        2601
voicingvoiceless:vowelE_     3268
voicingvoiced:vowelI_        3357
voicingvoiceless:vowelI_     3305
voicingvoiced:vowelO_        2719
voicingvoiceless:vowelO_     2927

Further Distributional Parameters:
    Estimate Est.Error l-80% CI u-80% CI Rhat Bulk_ESS Tail_ESS
phi     4.73      0.30     4.35     5.12 1.00     2477     3192

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).
```

References

Carignan, C., Coretta, S., Frahm, J., Harrington, J., Hoole, P., Joseph, A., Kunay, E., Voit, D., 2021. Planting the seed for sound change: Evidence from real-time MRI of velum kinematics in German. Language 97, 333–364. doi:10.1353/lan.2021.0020.