Rough R - Master analysis file

This is the master analysis file for our paper titled "R is for rough: An iconic universal based on touch". The paper contains a set of different analyses, which are presented in separate sections below. Each section is self-contained: libraries and data files are loaded in the top code chunk, and after running this, all the code for the relevant section should be possible to run.

1. English ratings

The data we use here are roughness ratings for 100 English adjectives from Stadtlander & Murdoch (2000). We first present a random forest analysis, followed by a Bayesian linear regression model with roughness ratings as outcome and presence of R / L as predictors.

We start the random forest analysis with a general test that simply looks at which phonemes are most predictive of roughness ratings. This is followed by a more exploratory analysis that aims to find out which parts of a word contribute the most to this effect. As part of this analysis, we compare several models:

- all onset consonants
- onset consonants in stressed syllables
- all rhyme segments (vowels and consonants)
- rhyme segments (vowels and consonants) in stressed syllables

Loading packages, loading data.

```r
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------------

## v ggplot2 3.1.0     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0

## Warning: package 'tibble' was built under R version 3.5.2

## Warning: package 'purrr' was built under R version 3.5.2

## Warning: package 'dplyr' was built under R version 3.5.2

## -- Conflicts ----------------------------------------------------------------------------------- tid
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(ranger)
```

```
## Warning: package 'ranger' was built under R version 3.5.2
```

```r
library(brms)
```

```
## Warning: package 'brms' was built under R version 3.5.2

## Loading required package: Rcpp

## Warning: package 'Rcpp' was built under R version 3.5.2

## Loading 'brms' package (version 2.9.0). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').
```

```r
source("scripts/rough_helper.r")

eng <- read_csv("final_data/english_norms.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Word = col_character(),
##   Pron = col_character(),
##   POS = col_character(),
##   onset = col_character(),
##   stressed_onset = col_character(),
##   rhyme = col_character(),
##   stressed_rhyme = col_character(),
##   PIE_root = col_character()
## )
## See spec(...) for full column specifications.
```

```r
eng$rough <- eng$Rough.M >= 0
```

1.1 Random forest analysis

Fitting random forest models. Starting with general model and then more specific ones.

```r
# 1) general model (with predictors of the form "is segment X present anywhere in the wordform?")
eng_rf_all_preds <- grep("all.", colnames(eng), value=T)
eng_rf_all_formula <- as.formula(paste0('Rough.M ~ ', paste(eng_rf_all_preds, collapse=" + ")))

set.seed(42)
eng_rf_mod_all <- ranger(eng_rf_all_formula,
     data = eng, mtry = round(sqrt(length(eng_rf_all_preds))), num.trees = 5000,
     importance = 'permutation')
# saveRDS(eng_rf_mod_all, "models/eng_rf_mod_all.rds")

# 2) onset only
eng_rf_onset_preds <- grep("^onset.", colnames(eng), value=T)
eng_rf_onset_formula <- as.formula(paste0('Rough.M ~ ', paste(eng_rf_onset_preds, collapse=" + ")))

set.seed(42)
eng_rf_mod_onset <- ranger(eng_rf_onset_formula,
     data = eng, mtry = round(sqrt(length(eng_rf_onset_preds))), num.trees = 5000,
     importance = 'permutation')

# 3) stressed onset only
eng_rf_stressed_onset_preds <- grep("^stressed_onset.", colnames(eng), value=T)
eng_rf_stressed_onset_formula <- as.formula(paste0('Rough.M ~ ', paste(eng_rf_stressed_onset_preds, coll

set.seed(42)
eng_rf_mod_stressed_onset <- ranger(eng_rf_stressed_onset_formula,
     data = eng, mtry = round(sqrt(length(eng_rf_stressed_onset_preds))), num.trees = 5000,
     importance = 'permutation')

# 4) rhyme only
```

```r
eng_rf_rhyme_preds <- grep("^rhyme.", colnames(eng), value=T)
eng_rf_rhyme_formula <- as.formula(paste0('Rough.M ~ ', paste(eng_rf_rhyme_preds, collapse=" + ")))

set.seed(42)
eng_rf_mod_rhyme <- ranger(eng_rf_rhyme_formula,
     data = eng, mtry = round(sqrt(length(eng_rf_rhyme_preds))), num.trees = 5000,
     importance = 'permutation')

# 5) stressed rhyme only
eng_rf_stressed_rhyme_preds <- grep("^stressed_rhyme.", colnames(eng), value=T)
eng_rf_stressed_rhyme_formula <- as.formula(paste0('Rough.M ~ ', paste(eng_rf_stressed_rhyme_preds, coll

set.seed(42)
eng_rf_mod_stressed_rhyme <- ranger(eng_rf_stressed_rhyme_formula,
     data = eng, mtry = round(sqrt(length(eng_rf_stressed_rhyme_preds))), num.trees = 5000,
     importance = 'permutation')
```

Now that we have fit the random forest models, we can look at their prediction accuracy (correlation between predicted vs. actual roughness ratings).

```r
sqrt(eng_rf_mod_all$r.squared)            # 0.255
```

```
## [1] 0.2552918
```

```r
sqrt(eng_rf_mod_onset$r.squared)          # 0.315
```

```
## [1] 0.3154774
```

```r
sqrt(eng_rf_mod_stressed_onset$r.squared) # 0.361
```

```
## [1] 0.3606027
```

```r
sqrt(eng_rf_mod_rhyme$r.squared)          # negative r-squared
```

```
## Warning in sqrt(eng_rf_mod_rhyme$r.squared): NaNs produced
```

```
## [1] NaN
```

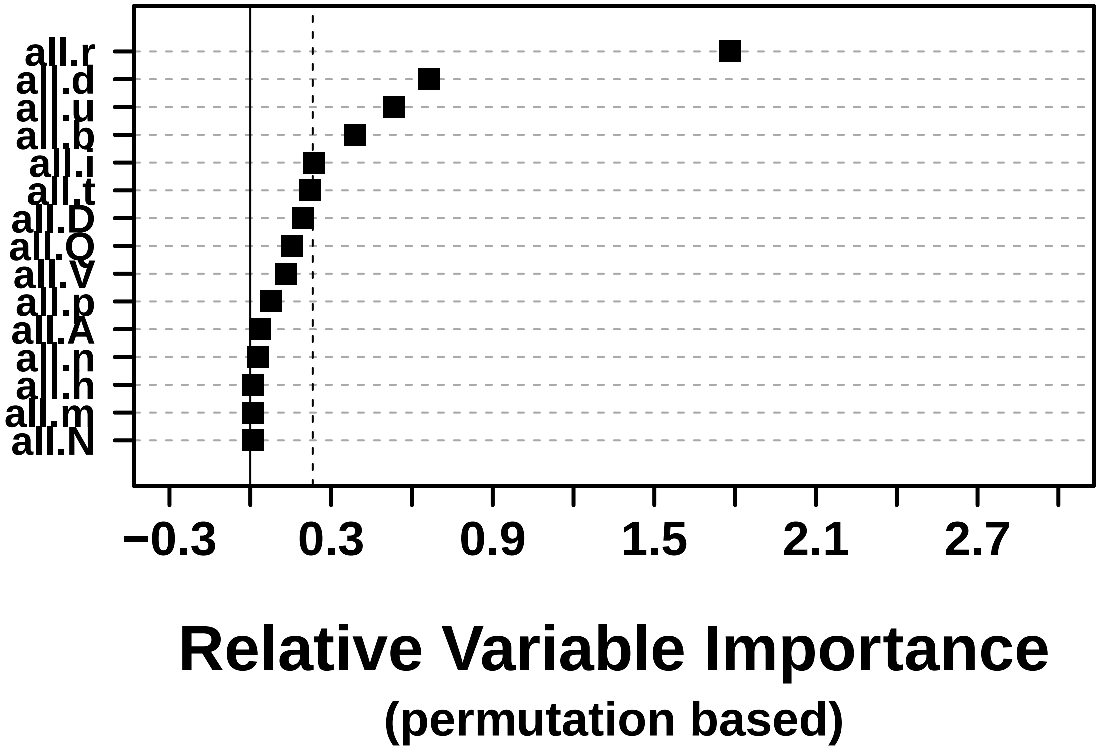
```r
sqrt(eng_rf_mod_stressed_rhyme$r.squared) # negative r-squared
```

```
## Warning in sqrt(eng_rf_mod_stressed_rhyme$r.squared): NaNs produced
```

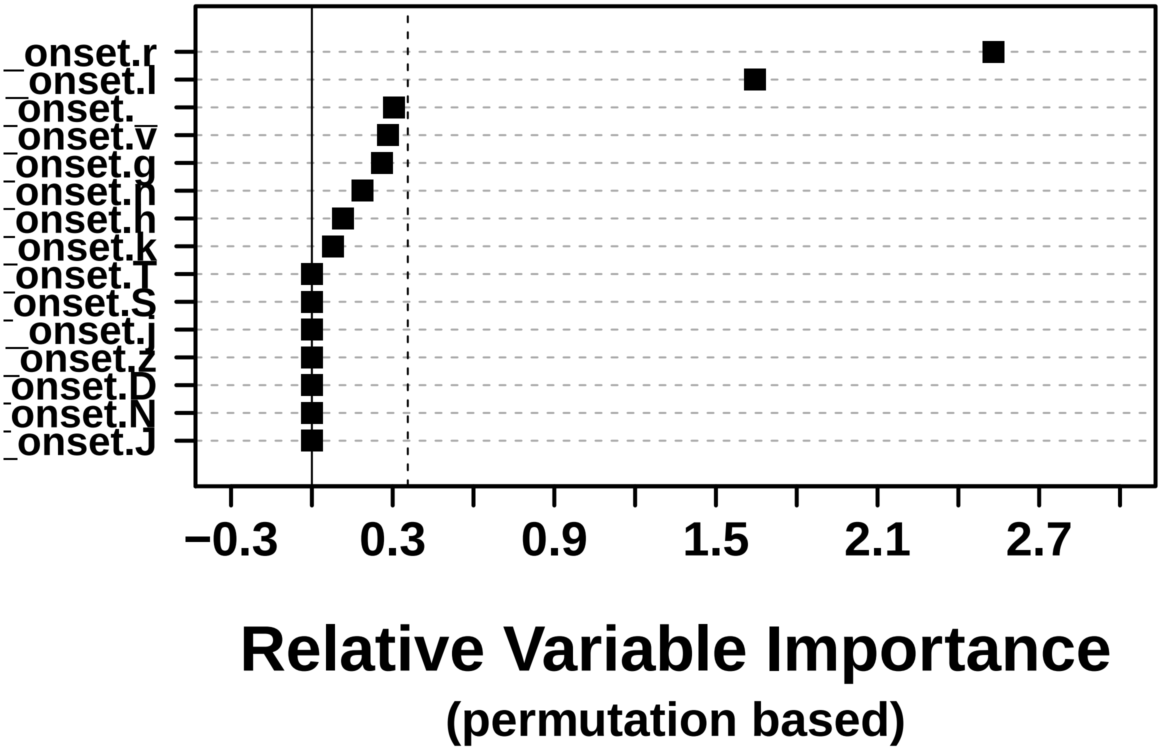
```
## [1] NaN
```

Variance importance for two best-performing models:

```r
par(mai = c(1.5, 1, 0.5, 0.5))
plot_varimp(tail(sort(eng_rf_mod_all$variable.importance), 15), xaxis = seq(-0.3, 3, 0.3))
abline(v=abs(min(eng_rf_mod_all$variable.importance)), lty=2)
```



```r
par(mai = c(1.5, 1, 0.5, 0.5))
plot_varimp(tail(sort(eng_rf_mod_stressed_onset$variable.importance), 15), xaxis = seq(-0.3, 3, 0.3))
abline(v=abs(min(eng_rf_mod_stressed_onset$variable.importance)), lty=2)
```

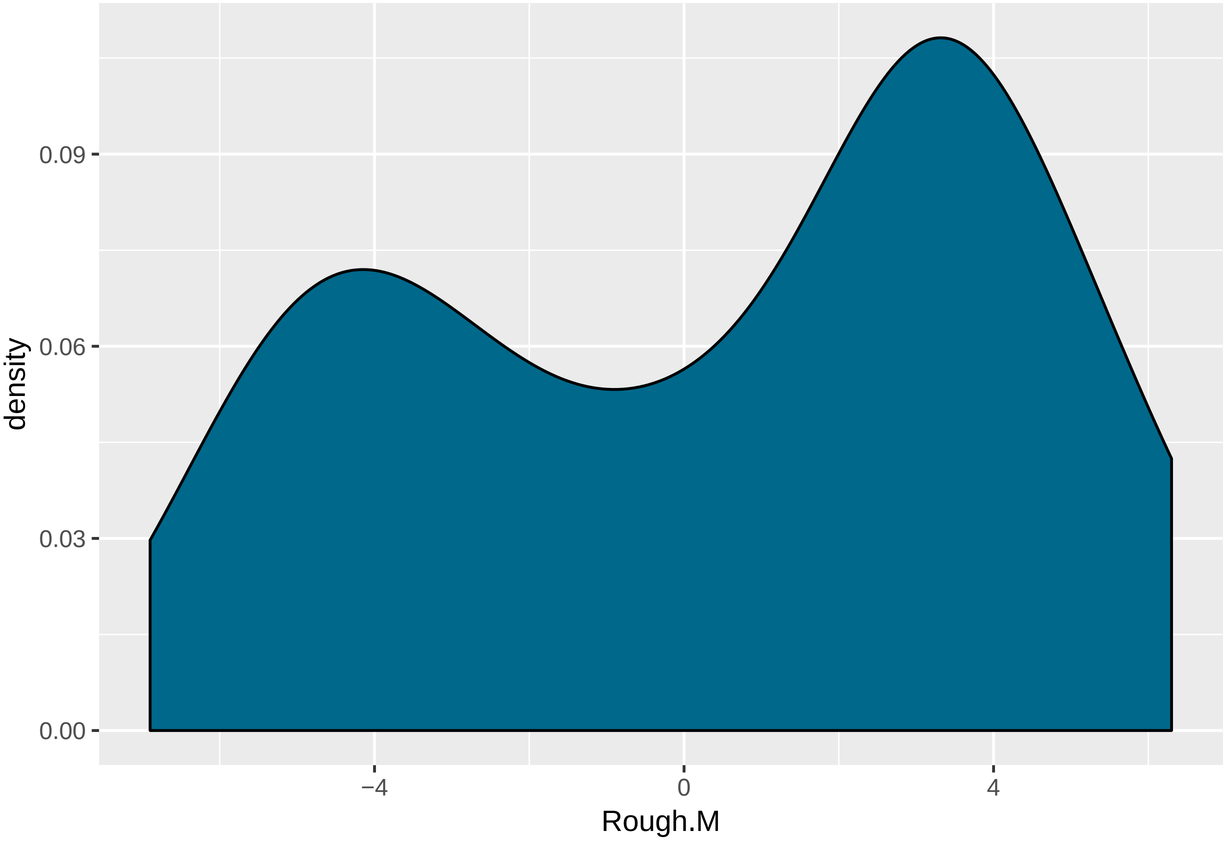


The graphs for the paper are assembled in a separate file. (TODO)

1.2 Descriptive stats

We now calculate some simple descriptive statistics for reporting in the paper and then move on to the statistical modelling. We split the data into "rough" and "smooth" subsets at zero of the rating scale; this is partly justified by the heavily bimodal distribution of the data:

```
eng$rough <- eng$Rough.M >= 0

# density plot
ggplot(eng, aes(x=Rough.M)) +
  geom_density(fill="deepskyblue4")
```



Here are the proportions of /r/ and /l/ in rough vs. smooth words.

```
# proportion of /r/ in rough vs. smooth words
eng %>%
  count(rough, all.r) %>%
  group_by(rough) %>%
  summarise(prop.r=sum(all.r * n) / sum(n))
```

```
## # A tibble: 2 x 2
##   rough prop.r
##   <lgl>  <dbl>
## 1 FALSE  0.293
## 2 TRUE   0.638
```

```
# proportion of /l/ in rough vs. smooth words
eng %>%
  count(rough, all.l) %>%
  group_by(rough) %>%
  summarise(prop.l=sum(all.l * n) / sum(n))
```

```
## # A tibble: 2 x 2
##   rough prop.l
##   <lgl>  <dbl>
## 1 FALSE  0.415
## 2 TRUE   0.310
```

The same proportions focusing on stressed onsets only.

```
# proportion of stressed onset /r/ in rough vs. smooth words
eng %>%
  count(rough, stressed_onset.r) %>%
  group_by(rough) %>%
  summarise(prop.r=sum(stressed_onset.r * n) / sum(n))
```

```
## # A tibble: 2 x 2
##   rough prop.r
##   <lgl>  <dbl>
## 1 FALSE  0.146
## 2 TRUE   0.483
```

```
# proportion of stressed onset /l/ in rough vs. smooth words
eng %>%
  count(rough, stressed_onset.l) %>%
  group_by(rough) %>%
  summarise(prop.l=sum(stressed_onset.l * n) / sum(n))
```

```
## # A tibble: 2 x 2
##   rough prop.l
##   <lgl>  <dbl>
## 1 FALSE 0.244
## 2 TRUE  0.0345
```

1.3 Bayesian beta regression with ratings as outcome

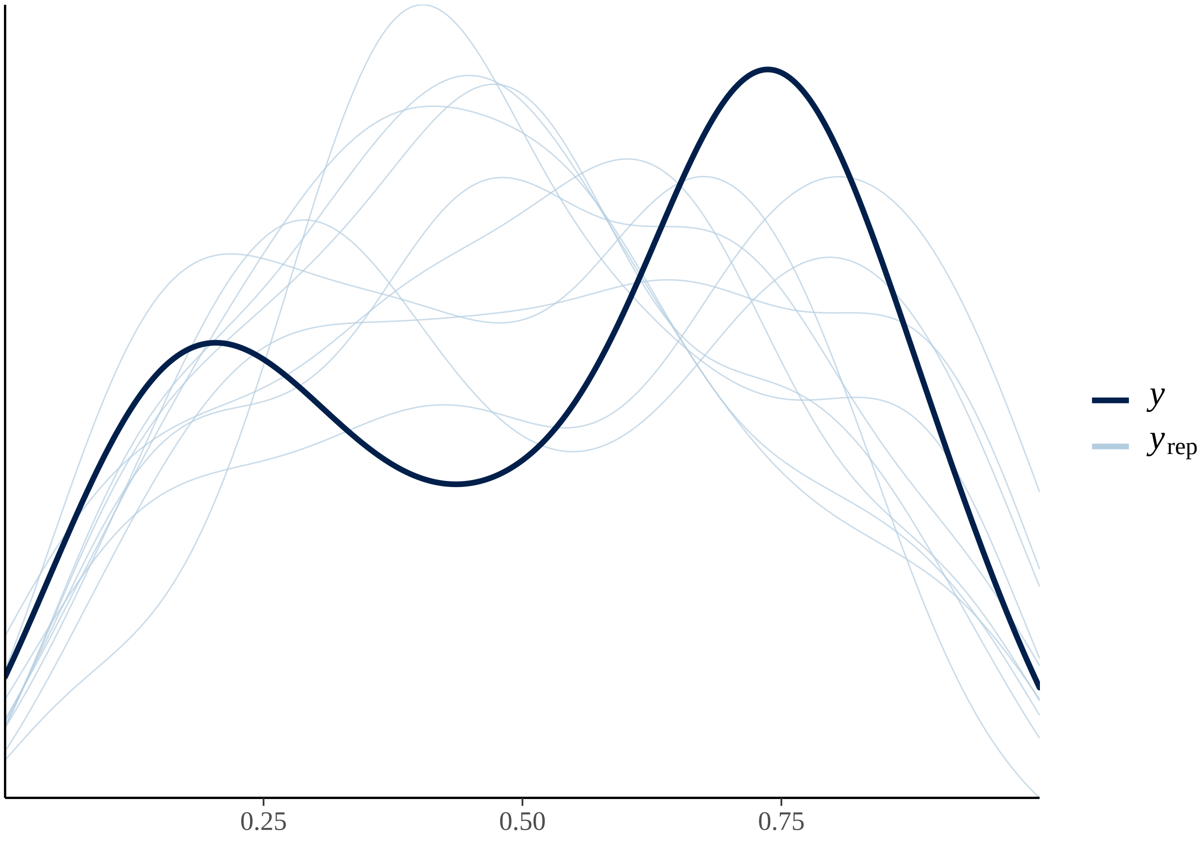
We now fit a Bayesian beta regression model with roughness ratings as the outcome and presence of R / L as the predictors. A beta regression is used since the ratings are bounded on both sides.

```
# in order for beta regression to work, outcome has to be in [0,1]
eng$Rough.M.beta <- (eng$Rough.M + 7)/14

set.seed(314)
eng_beta_mod <- brm(Rough.M.beta ~ all.r + all.l,
                   data=eng,
                   family="beta",
                   refresh=0)
summary(eng_beta_mod)
```

```
##  Family: beta
##   Links: mu = logit; phi = identity
## Formula: Rough.M.beta ~ all.r + all.l
##    Data: eng (Number of observations: 99)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
```

```
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept    -0.18      0.17    -0.52     0.17       3738 1.00
## all.r         0.66      0.20     0.25     1.07       3798 1.00
## all.l        -0.25      0.21    -0.66     0.16       4140 1.00
##
## Family Specific Parameters:
##     Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## phi     2.91      0.37     2.24     3.69       3540 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
# posterior predictive check: fit not fantastic due to bimodality
pp_check(eng_beta_mod)
```



We now provide a summary of this model based on the posterior distribution.

```
x <- beta_summary(eng_beta_mod, eng, rpred="all.r", lpred="all.l", binary_pred=F, printPlease=T)
```

```
## predicted roughness rating based on R:
##    without R: -0.92 [-1.87,0.07]
##    with R: 1.37 [0.43,2.27]
## predicted roughness difference (R - no R):
##    diff: 2.29 [0.87,3.65]
##
## predicted roughness rating based on L:
```

```
##    without L: 0.52 [-0.34,1.39]
##    with L: -0.34 [-1.53,0.85]
## predicted roughness difference (L - no L):
##    diff: -0.87 [-2.3,0.56]
```

1.4 Bayesian logistic regression with roughness as outcome

The bimodality of the ratings scale (which is not entirely removed by adding the L and R predictors) means that the results from the model above ought to be treated with caution. For this reason (and for comparability with the PIE model below), we've also run two logistic regression models with presence of L / presence of R as the outcome and binary roughness as the predictor.

```
eng_brm_all.x_priors <- c(set_prior("student_t(5,0,2.5)", class = "b"),
                          set_prior("student_t(5,0,2.5)", class = "Intercept"))
set.seed(314)
eng_brm_all.r_mod <- brm(all.r ~ rough,
                         data=eng,
                         prior=eng_brm_all.x_priors,
                         family="bernoulli",
                         refresh=0)
set.seed(314)
eng_brm_all.l_mod <- brm(all.l ~ rough,
                         data=eng,
                         prior=eng_brm_all.x_priors,
                         family="bernoulli",
                         refresh=0)
summary(eng_brm_all.r_mod)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: all.r ~ rough
##    Data: eng (Number of observations: 99)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept    -0.88      0.34    -1.58    -0.25       3750 1.00
## roughTRUE     1.44      0.44     0.59     2.32       3665 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
summary(eng_brm_all.l_mod)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: all.l ~ rough
##    Data: eng (Number of observations: 99)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept    -0.36      0.32    -0.98     0.27       3592 1.00
## roughTRUE    -0.44      0.42    -1.24     0.38       3217 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Model predictions.

```
x <- logistic_summary(eng_brm_all.r_mod, eng, outcome="/r/", roughpred="rough")
```

```
## predicted probability of /r/ rating based on roughness:
##    smooth: 0.3 [0.17,0.44]
##    rough: 0.63 [0.51,0.75]
## predicted difference in probability of /r/ (rough - smooth):
##    diff: 0.34 [0.15,0.51]
```

```
y <- logistic_summary(eng_brm_all.l_mod, eng, outcome="/l/", roughpred="rough")
```

```
## predicted probability of /l/ rating based on roughness:
##    smooth: 0.41 [0.27,0.57]
##    rough: 0.31 [0.2,0.44]
## predicted difference in probability of /l/ (rough - smooth):
##    diff: -0.1 [-0.28,0.08]
```

1.5 Descriptive stats / Bayesian logistic regression for English OED data

We now look at whether the same pattern is present in the reconstructed PIE roots of these English words. Two important analytical decisions:

1) Since many of these words have shifted massively in terms of their meaning, there is no point in using fine-grained roughness ratings. Instead, we rely on the broad binary variable “rough”.
2) Since some of the words in our data derive from the same PIE roots, we merge these into single entries. Their roughness value is defined as the majority roughness value among all words sharing the same root.

```
eng_pie <- eng %>%
  dplyr::select(Word, PIE_root, rough) %>%
  filter(!is.na(PIE_root)) %>%
  group_by(PIE_root) %>%
  summarise(rough_prop=mean(rough),
            rough=as.logical(round(rough_prop))) %>%
  ungroup() %>%
  mutate(r=str_detect(PIE_root, 'r'),
         l=str_detect(PIE_root, 'l'))

# note that the proportion of rough words corresponding to a single
# root is always 1 or 0, meaning that all cognates of every root
# are uniformly rough or smooth
```

We now calculate descriptive stats.

```
# proportion of /r/ in rough vs. smooth words
eng_pie %>%
  count(rough, r) %>%
  group_by(rough) %>%
  summarise(prop.r=sum(r * n) / sum(n))
```

```
## # A tibble: 2 x 2
##   rough prop.r
##   <lgl>  <dbl>
## 1 FALSE  0.182
## 2 TRUE   0.621
```

```
# proportion of /l/ in rough vs. smooth words
eng_pie %>%
  count(rough, l) %>%
  group_by(rough) %>%
  summarise(prop.l=sum(l * n) / sum(n))
```

```
## # A tibble: 2 x 2
##   rough prop.l
##   <lgl>  <dbl>
## 1 FALSE  0.409
## 2 TRUE   0.138
```

And now a Bayesian logistic regression model.

```
eng_brm_pie.x_priors <- c(set_prior("student_t(5,0,2.5)", class = "b"),
                           set_prior("student_t(5,0,2.5)", class = "Intercept"))
set.seed(314)
eng_brm_pie.r_mod <- brm(r ~ rough,
                         data=eng_pie,
                         prior=eng_brm_pie.x_priors,
                         family="bernoulli",
                         refresh=0)
set.seed(314)
eng_brm_pie.l_mod <- brm(l ~ rough,
                         data=eng_pie,
                         prior=eng_brm_all.x_priors,
                         family="bernoulli",
                         refresh=0)
summary(eng_brm_pie.r_mod)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: r ~ rough
##    Data: eng_pie (Number of observations: 51)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept    -1.50      0.56    -2.64    -0.49       1836 1.00
## roughTRUE     1.96      0.66     0.71     3.29       2070 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
summary(eng_brm_pie.l_mod)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: l ~ rough
##    Data: eng_pie (Number of observations: 51)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept    -0.43      0.45    -1.33     0.42       4101 1.00
## roughTRUE    -1.38      0.67    -2.72    -0.11       2681 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Model predictions.

```
x <- logistic_summary(eng_brm_pie.r_mod, eng, outcome="/r/", roughpred="rough")
```

```
## predicted probability of /r/ rating based on roughness:
##    smooth: 0.2 [0.07,0.38]
##    rough: 0.61 [0.43,0.78]
## predicted difference in probability of /r/ (rough - smooth):
##    diff: 0.41 [0.17,0.63]
```

```
y <- logistic_summary(eng_brm_pie.l_mod, eng, outcome="/l/", roughpred="rough")
```

```
## predicted probability of /l/ rating based on roughness:
##    smooth: 0.4 [0.21,0.6]
##    rough: 0.15 [0.05,0.3]
## predicted difference in probability of /l/ (rough - smooth):
##    diff: -0.25 [-0.48,-0.02]
```

2. IE Google Translate data

The data that we analyse here are Google translations of the words from the Stadtlander & Murdoch (2000) dataset into a variety of languages. For now, we focus on the Indo-European subset of these languages (the non-IE subset will be analysed later). The analysis here is simple: we focus on the binary roughness predictor (as, again, a fine-grained roughness measure does not make sense, given the coarseness of our methods), and look at the proportion of /r/ and /l/ in rough vs. smooth words.

```
library(tidyverse)
library(brms)

trs_IE <- read_csv("final_data/google_translate.csv") %>%
  filter(stock=="Indo-European")
```

```
## Parsed with column specification:
## cols(
##   iso = col_character(),
##   language = col_character(),
##   eng_orig = col_character(),
##   roughness = col_double(),
##   trans = col_character(),
##   back_trans = col_character(),
##   language_n = col_double(),
##   rough = col_logical(),
##   iso639.3 = col_character(),
##   regexp.r = col_character(),
##   regexp.l = col_character(),
##   r = col_logical(),
##   l = col_logical(),
##   mbranch = col_character(),
##   stock = col_character(),
##   area = col_character(),
##   continent = col_character()
## )
```

Raw descriptive stats & plot.

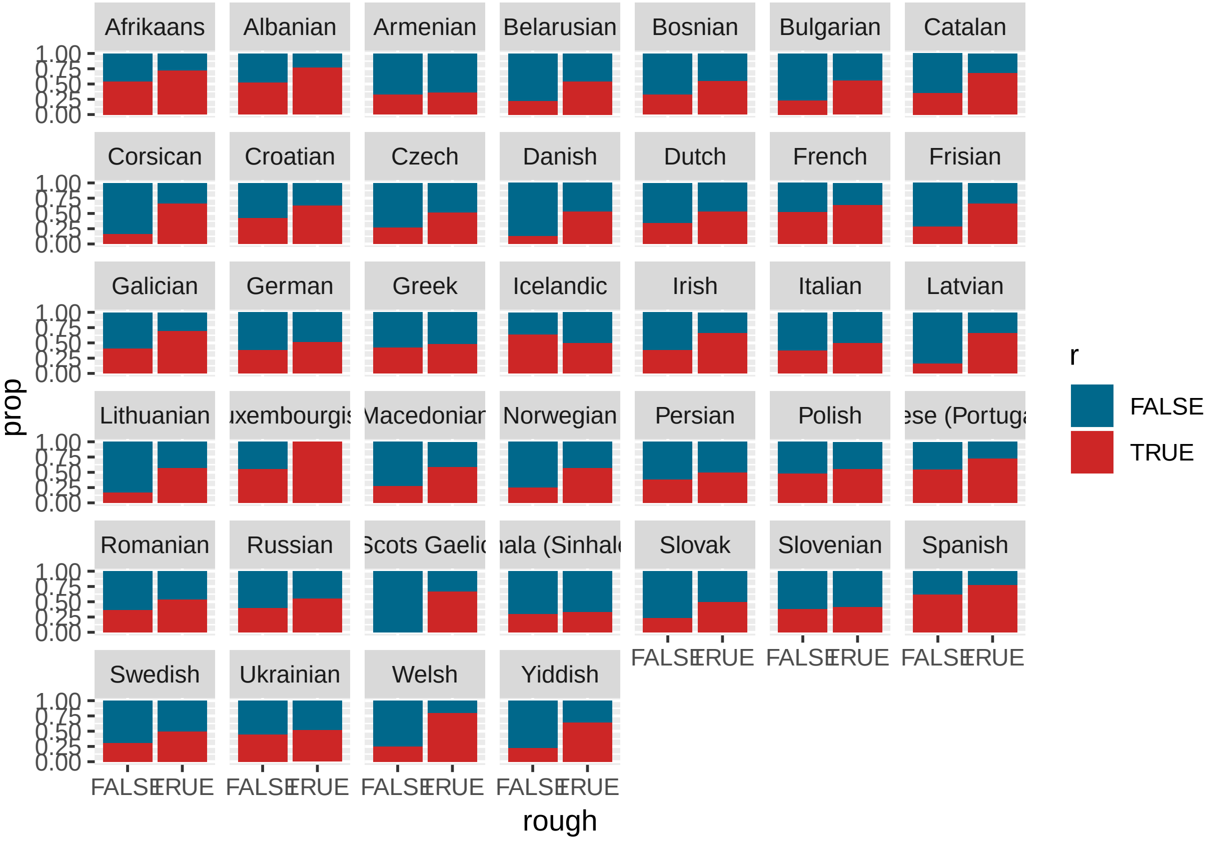
```
# proportion of /r/ in rough vs. smooth words
trs_IE %>%
  count(rough, r) %>%
  group_by(rough) %>%
  summarise(prop.r=sum(r * n) / sum(n))
```

```
## # A tibble: 2 x 2
##   rough prop.r
##   <lgl>  <dbl>
## 1 FALSE  0.369
## 2 TRUE   0.582
```

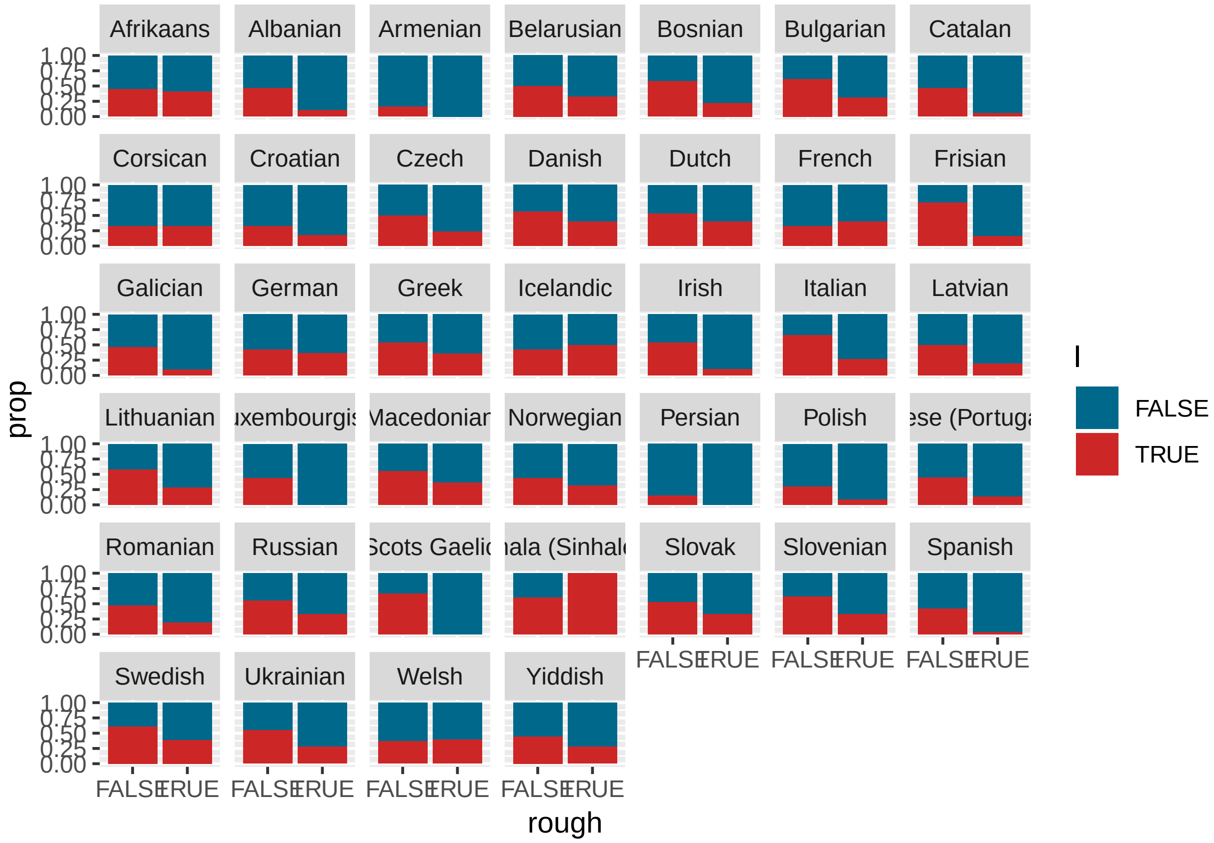
```
# proportion of /l/ in rough vs. smooth words
trs_IE %>%
  count(rough, l) %>%
  group_by(rough) %>%
  summarise(prop.l=sum(l * n) / sum(n))
```

```
## # A tibble: 2 x 2
##   rough prop.l
##   <lgl>  <dbl>
## 1 FALSE  0.481
## 2 TRUE   0.272
```

```
# by-language bar plots for /r/
trs_IE %>%
  count(language, rough, r) %>%
  group_by(language, rough) %>%
  mutate(prop=n / sum(n)) %>%
ggplot(aes(x=rough, y=prop, fill=r)) +
  facet_wrap(~language) +
  geom_bar(stat="identity") +
  scale_fill_manual(values=c("deepskyblue4","firebrick3"))
```



```r
# by-language bar plots for /l/
trs_IE %>%
  count(language, rough, l) %>%
  group_by(language, rough) %>%
  mutate(prop=n / sum(n)) %>%
ggplot(aes(x=rough, y=prop, fill=l)) +
  facet_wrap(~language) +
  geom_bar(stat="identity") +
  scale_fill_manual(values=c("deepskyblue4","firebrick3"))
```



Bayesian mixed effects logistic modelling (similar to above, but now with additional random effects).

```
trs_IE_brm_priors <- c(
  set_prior("student_t(5,0,2.5)", class = "b"),
  set_prior("student_t(5,0,2.5)", class = "Intercept"),
  set_prior("lkj(2)", class = "cor"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="mbranch"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "roughTRUE", group="mbranch"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="language"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "roughTRUE", group="language"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="eng_orig")
)

set.seed(314)
trs_IE_brm_r_mod <- brm(r ~ rough +
                          (1 + rough | mbranch) +
                          (1 + rough | language) +
                          (1 | eng_orig),
                        data=trs_IE,
                        prior=trs_IE_brm_priors,
                        family="bernoulli",
                        control=list(adapt_delta=0.9),
                        refresh=0)
summary(trs_IE_brm_r_mod)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: r ~ rough + (1 + rough | mbranch) + (1 + rough | language) + (1 | eng_orig)
##    Data: trs_IE (Number of observations: 1385)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Group-Level Effects:
## ~eng_orig (Number of levels: 98)
##               Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sd(Intercept)     1.79      0.19     1.45     2.20       1257 1.00
##
## ~language (Number of levels: 39)
##                          Estimate Est.Error l-95% CI u-95% CI Eff.Sample
## sd(Intercept)                0.35      0.13     0.09     0.62       1123
## sd(roughTRUE)                0.18      0.14     0.01     0.53       1780
## cor(Intercept,roughTRUE)    -0.17      0.45    -0.87     0.74       3720
##                          Rhat
## sd(Intercept)            1.01
## sd(roughTRUE)            1.00
## cor(Intercept,roughTRUE) 1.00
##
## ~mbranch (Number of levels: 6)
##                          Estimate Est.Error l-95% CI u-95% CI Eff.Sample
## sd(Intercept)                0.37      0.31     0.02     1.17       1376
## sd(roughTRUE)                0.32      0.30     0.01     1.11       1831
## cor(Intercept,roughTRUE)    -0.00      0.45    -0.81     0.82       4847
##                          Rhat
## sd(Intercept)            1.00
## sd(roughTRUE)            1.00
## cor(Intercept,roughTRUE) 1.00
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept    -0.60      0.38    -1.32     0.15       1137 1.00
## roughTRUE     1.16      0.46     0.26     2.06       1080 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
set.seed(314)
trs_IE_brm_l_mod <- brm(l ~ rough +
                          (1 + rough | mbranch) +
                          (1 + rough | language) +
                          (1 | eng_orig),
                        data=trs_IE,
                        prior=trs_IE_brm_priors,
                        family="bernoulli",
                        control=list(adapt_delta=0.9),
                        refresh=0)
summary(trs_IE_brm_l_mod)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: l ~ rough + (1 + rough | mbranch) + (1 + rough | language) + (1 | eng_orig)
##    Data: trs_IE (Number of observations: 1385)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
## 
## Group-Level Effects: 
## ~eng_orig (Number of levels: 98) 
##               Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sd(Intercept)     1.48      0.17     1.17     1.85       1313 1.00
## 
## ~language (Number of levels: 39) 
##                           Estimate Est.Error l-95% CI u-95% CI Eff.Sample
## sd(Intercept)                 0.46      0.15     0.16     0.76        647
## sd(roughTRUE)                 0.28      0.20     0.01     0.75        900
## cor(Intercept,roughTRUE)      0.19      0.41    -0.67     0.87       3564
##                           Rhat
## sd(Intercept)             1.01
## sd(roughTRUE)             1.00
## cor(Intercept,roughTRUE)  1.00
## 
## ~mbranch (Number of levels: 6) 
##                           Estimate Est.Error l-95% CI u-95% CI Eff.Sample
## sd(Intercept)                 0.29      0.25     0.01     0.93       1888
## sd(roughTRUE)                 0.48      0.38     0.03     1.50       1876
## cor(Intercept,roughTRUE)      0.08      0.44    -0.76     0.84       3457
##                           Rhat
## sd(Intercept)             1.00
## sd(roughTRUE)             1.00
## cor(Intercept,roughTRUE)  1.00
## 
## Population-Level Effects: 
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept    -0.31      0.33    -0.97     0.32       1288 1.00
## roughTRUE    -0.96      0.45    -1.87    -0.06       1407 1.00
## 
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample 
## is a crude measure of effective sample size, and Rhat is the potential 
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Model predictions.

1a) predicted likelihood of R for smooth vs. rough with 95% CIs (smooth: [,]; rough: [,]) 1b) estimated difference with 95% CIs ([,]) 2a) predicted likelihood of L for smooth vs. rough with 95% CIs (smooth: [,]; rough [,]) 2b) estimated difference with 95% CIs ([,])

```
x <- logistic_summary(trs_IE_brm_r_mod, eng, outcome="/r/", roughpred="rough")
```

```
## predicted probability of /r/ rating based on roughness:
##    smooth: 0.36 [0.21,0.54]
##    rough: 0.63 [0.43,0.8]
## predicted difference in probability of /r/ (rough - smooth):
##    diff: 0.27 [0.06,0.46]
```

```
y <- logistic_summary(trs_IE_brm_l_mod, eng, outcome="/l/", roughpred="rough")
```

```
## predicted probability of /l/ rating based on roughness:
##    smooth: 0.42 [0.27,0.58]
##    rough: 0.23 [0.11,0.38]
## predicted difference in probability of /l/ (rough - smooth):
```

```
##    diff: -0.2 [-0.37,-0.01]
```

3. Hungarian ratings

We now move on to our analysis of roughness ratings from Hungarian. These data come from an experiment conducted as part of the current study, where participants rated 85 Hungarian surface descriptors. Some of these adjectives are translations of the Stadtlander & Murdoch (2000) adjectives, while some others are independent of that experiment. For comparability with the English analysis, we use the aggregated data in our analyses, even though a deaggregated data set is available. The deaggregated data could be analysed via zero-inflated beta regression – but for simplicity, we avoid this here. We first present random forest analyses, which is followed by Bayesian beta regression analyses. The structure of this section is closely analogous to that of section 1.

To make sure that the results here are truly complementary to the Indo-European analysis, we restrict the Hungarian data to those forms that are not of Indo-European origin (i.e. we exclude IE loanwords).

```r
library(tidyverse)
library(ranger)
library(brms)

hun_aggr <- read_csv("final_data/hun_norms_aggr.csv") %>%
  filter(!(etymology.source %in% c('slavic', 'germanic', 'greek', 'romance')))

hun_aggr$rough <- hun_aggr$roughness >= 0
```

3.1 Random forest analysis

Fitting random forest models to the aggregated data. Starting with general model and then more specific ones.

```r
# 1) general model (with predictors of the form "is segment X present anywhere in the wordform?")
# (note: I've tried consonantal models too, but their R (for correlation) is lower at 0.265)
# hun_rf_cons_preds <- grep("all[.][iIoOaAyYøØeE]", grep("all.", colnames(hun_aggr), value=T), value=T,
# hun_rf_cons_formula <- as.formula(paste0('roughness ~ ', paste(hun_rf_cons_preds, collapse=" + ")))
hun_rf_all_preds <- grep("all.", colnames(hun_aggr), value=T)
hun_rf_all_formula <- as.formula(paste0('roughness ~ ', paste(hun_rf_all_preds, collapse=" + ")))

set.seed(450)
hun_rf_mod_all <- ranger(hun_rf_all_formula,
     data = hun_aggr, mtry = round(sqrt(length(hun_rf_all_preds))), num.trees = 5000,
     importance = 'permutation')
### note:
# party - here, /r/ is the winner... though it's only minimally different from the
# ranger output
# hun_rf_mod_all2 <- cforest(hun_rf_all_formula, data = hun_aggr, control=cforest_control(mtry = round(
# hun_rf_varimp_all <- varimp(hun_rf_mod_all2, conditional=T)
# dotchart(sort(hun_rf_varimp_all))


# 2) onset only
hun_rf_onset_preds <- grep("^onset.", colnames(hun_aggr), value=T)
hun_rf_onset_formula <- as.formula(paste0('roughness ~ ', paste(hun_rf_onset_preds, collapse=" + ")))

set.seed(42)
```

```r
hun_rf_mod_onset <- ranger(hun_rf_onset_formula,
     data = hun_aggr, mtry = round(sqrt(length(hun_rf_onset_preds))), num.trees = 5000,
     importance = 'permutation')

# 3) stressed onset only
hun_rf_stressed_onset_preds <- grep("^stressed_onset.", colnames(hun_aggr), value=T)
hun_rf_stressed_onset_formula <- as.formula(paste0('roughness ~ ', paste(hun_rf_stressed_onset_preds, c

set.seed(42)
hun_rf_mod_stressed_onset <- ranger(hun_rf_stressed_onset_formula,
     data = hun_aggr, mtry = round(sqrt(length(hun_rf_stressed_onset_preds))), num.trees = 5000,
     importance = 'permutation')

# 4) rhyme only
hun_rf_rhyme_preds <- grep("^rhyme.", colnames(hun_aggr), value=T)
hun_rf_rhyme_formula <- as.formula(paste0('roughness ~ ', paste(hun_rf_rhyme_preds, collapse=" + ")))

set.seed(42)
hun_rf_mod_rhyme <- ranger(hun_rf_rhyme_formula,
     data = hun_aggr, mtry = round(sqrt(length(hun_rf_rhyme_preds))), num.trees = 5000,
     importance = 'permutation')

# 5) stressed rhyme only
hun_rf_stressed_rhyme_preds <- grep("^stressed_rhyme.", colnames(hun_aggr), value=T)
hun_rf_stressed_rhyme_formula <- as.formula(paste0('roughness ~ ', paste(hun_rf_stressed_rhyme_preds, c

set.seed(42)
hun_rf_mod_stressed_rhyme <- ranger(hun_rf_stressed_rhyme_formula,
     data = hun_aggr, mtry = round(sqrt(length(hun_rf_stressed_rhyme_preds))), num.trees = 5000,
     importance = 'permutation')
```

Now that we have fit the random forest models, we can look at their prediction accuracy (correlation between predicted vs. actual roughness ratings).

```r
sqrt(hun_rf_mod_all$r.squared)            # 0.372
```

```
## [1] 0.3739899
```

```r
sqrt(hun_rf_mod_onset$r.squared)          # 0.342
```

```
## [1] 0.3424282
```

```r
sqrt(hun_rf_mod_stressed_onset$r.squared) # negative r-squared
```

```
## Warning in sqrt(hun_rf_mod_stressed_onset$r.squared): NaNs produced

## [1] NaN
```

```r
sqrt(hun_rf_mod_rhyme$r.squared)          # 0.178
```

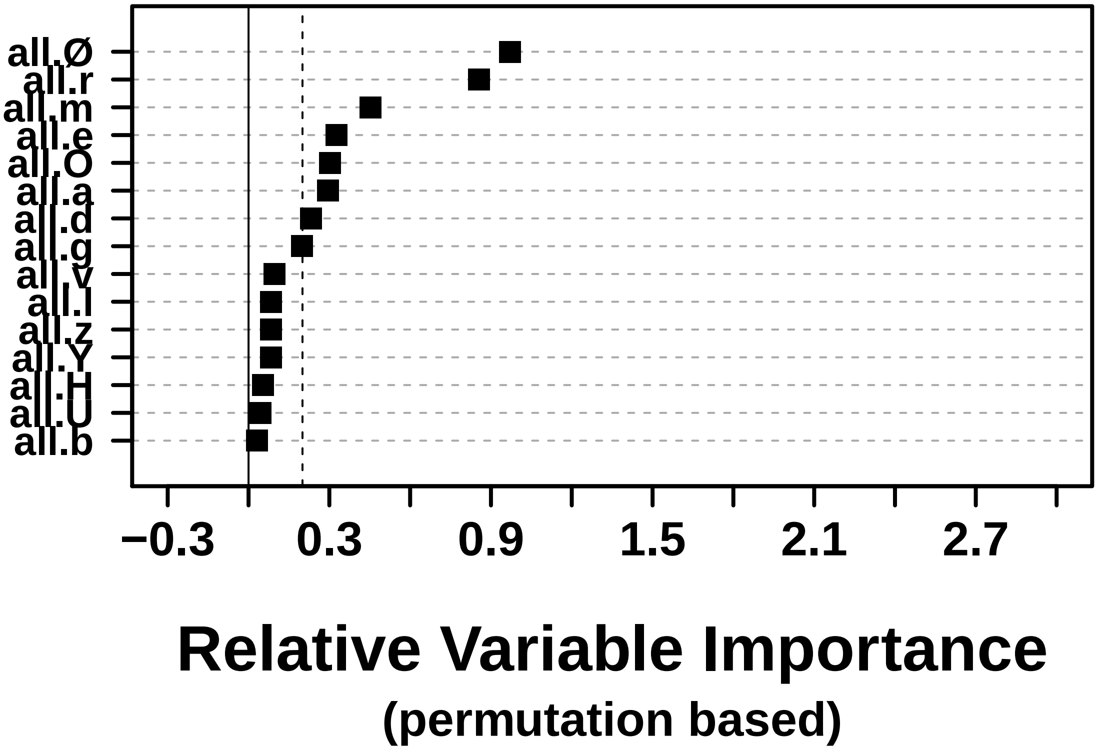
```
## [1] 0.1776716
```

```r
sqrt(hun_rf_mod_stressed_rhyme$r.squared) # negative r-squared
```

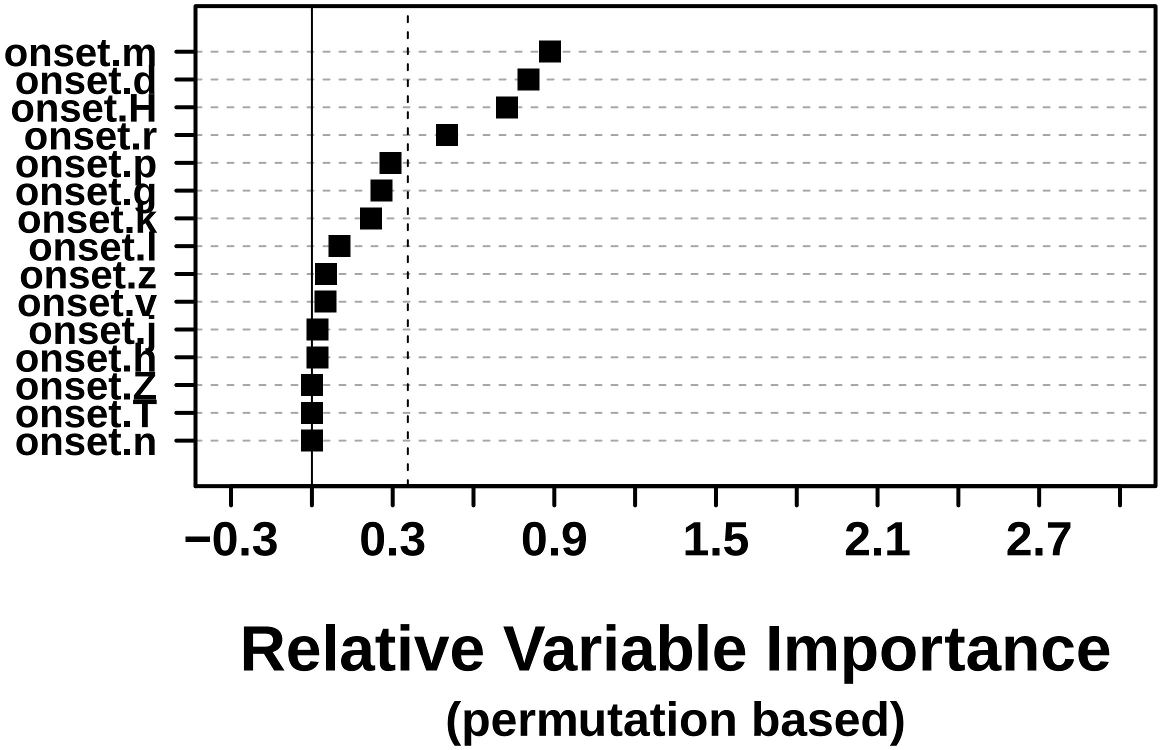
```
## Warning in sqrt(hun_rf_mod_stressed_rhyme$r.squared): NaNs produced

## [1] NaN
```

Variance importance for two models (note that /r/ is not the winner for either model, though it's pretty much tied for first place for the first one).

```
par(mai = c(1.5, 1, 0.5, 0.5))
plot_varimp(tail(sort(hun_rf_mod_all$variable.importance), 15), xaxis = seq(-0.3, 3, 0.3))
abline(v=abs(min(hun_rf_mod_all$variable.importance)), lty=2)
```



```
par(mai = c(1.5, 1, 0.5, 0.5))
plot_varimp(tail(sort(hun_rf_mod_onset$variable.importance), 15), xaxis = seq(-0.3, 3, 0.3))
abline(v=abs(min(eng_rf_mod_stressed_onset$variable.importance)), lty=2)
```



A side note: while /r/ does not seem to have the highest variable importance in the models, it is (1) among the most important variables and (2) it is the most general predictor out of all the ones listed here insofar as almost half of all of our forms contain an /r/. This might be the reason why random forests fit with party() tend to bring out /r/ as the phoneme with the highest variable importance.

```
mean(hun_aggr$all.r) # sum(hun_aggr$all.r): that's 29 items
```

```
## [1] 0.4393939
```

```
mean(hun_aggr$all.ø) # sum(hun_aggr$all.ø): that's literally *3* items!
```

```
## [1] 0.04545455
```

```
mean(hun_aggr$all.m)
```

```
## [1] 0.1818182
```

```
mean(hun_aggr$all.d)
```

```
## [1] 0.1363636
```

```
mean(hun_aggr$all.H)
```

```
## [1] 0.1212121
```

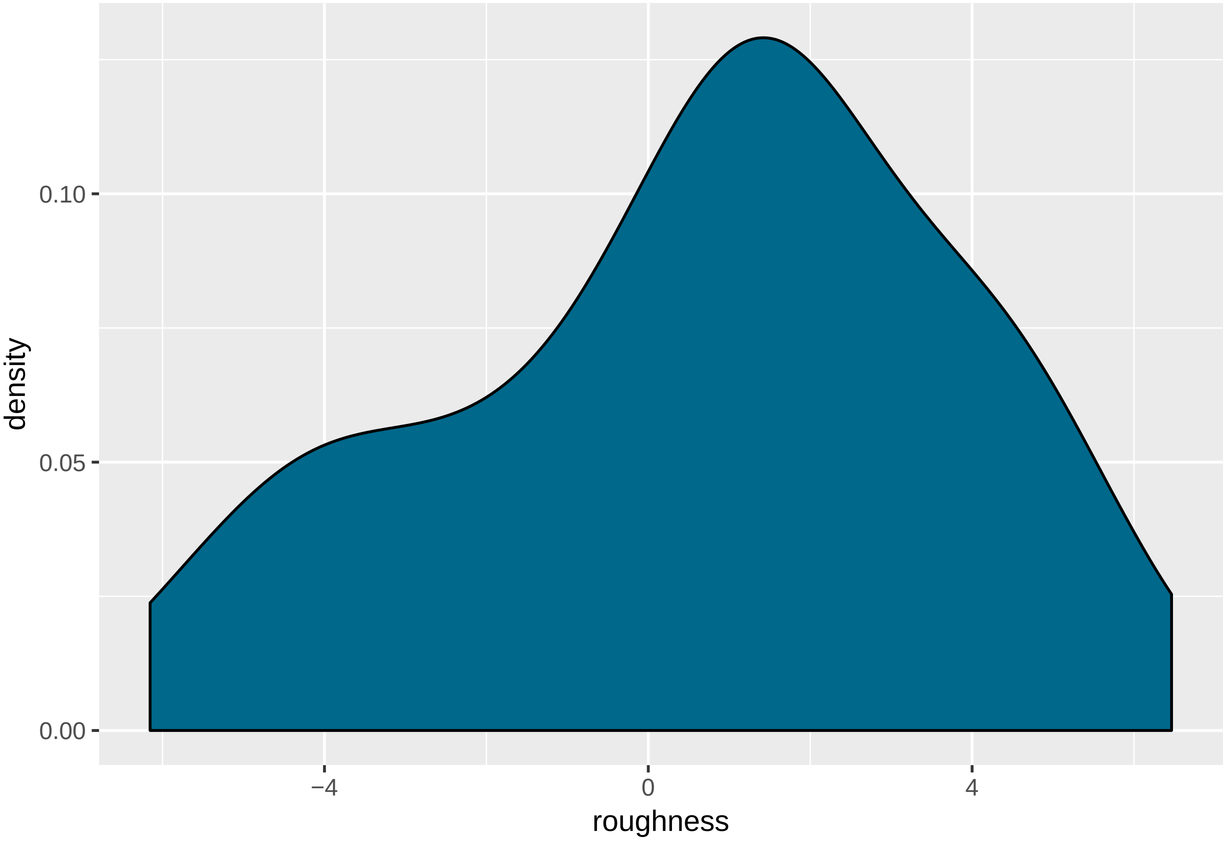
In the end, I don't think it really matters whether /r/ ends up as variable with the highest variable importance or slightly lower ranked: it is clearly an important predictor, and that's what our paper predicted (not that no other predictor is stronger than /r/).

The graphs for the paper are assembled in a separate file. (TODO)

3.2 Descriptive stats

We now calculate some simple descriptive statistics for reporting in the paper and then move on to the statistical modelling. These ratings are not nearly as bimodal as the English ones, but there is still some bimodality and it is useful to employ a binary split for comparability with the English data.

```
# density plot
ggplot(hun_aggr, aes(x=roughness)) +
  geom_density(fill="deepskyblue4")
```



Here are the proportions of /r/ and /l/ in rough vs. smooth words.

```
# proportion of /r/ in rough vs. smooth words
hun_aggr %>%
  count(rough, all.r) %>%
  group_by(rough) %>%
  summarise(prop.r=sum(all.r * n) / sum(n))
```

```
## # A tibble: 2 x 2
##   rough prop.r
##   <lgl>  <dbl>
## 1 FALSE  0.208
## 2 TRUE   0.571
```

```
# proportion of /l/ in rough vs. smooth words
hun_aggr %>%
  count(rough, all.l) %>%
  group_by(rough) %>%
  summarise(prop.l=sum(all.l * n) / sum(n))
```

```
## # A tibble: 2 x 2
##   rough prop.l
##   <lgl>  <dbl>
## 1 FALSE 0.208
## 2 TRUE  0.0714
```

There's not much point in calculating these proportions for onset / stressed onset / other positions, as the best model was the one fit to the whole word.

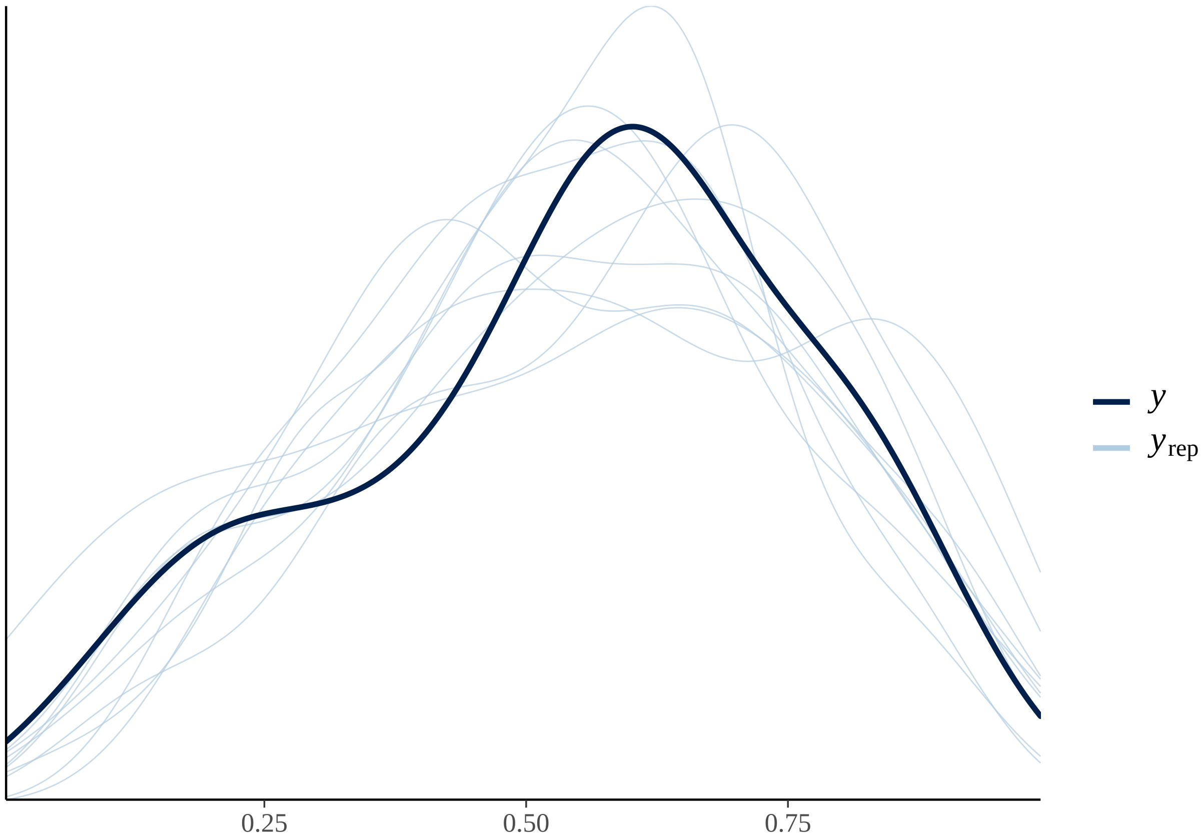
3.3 Bayesian beta regression with ratings as outcome

We now fit a Bayesian beta regression model with roughness ratings as the outcome and presence of R / L as the predictors. A beta regression is used since the ratings are bounded on both sides.

```
# in order for beta regression to work, outcome has to be in [0,1]
hun_aggr$roughness.beta <- (hun_aggr$roughness + 7)/14

set.seed(314)
hun_brm_beta_mod <- brm(roughness.beta ~ all.r + all.l,
                          data=hun_aggr,
                          family="beta",
                          refresh=0)
summary(hun_brm_beta_mod)
```

```
##  Family: beta
##   Links: mu = logit; phi = identity
## Formula: roughness.beta ~ all.r + all.l
##    Data: hun_aggr (Number of observations: 66)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept    -0.09      0.16    -0.39     0.21       3590 1.00
## all.r         0.62      0.22     0.17     1.07       3591 1.00
## all.l        -0.16      0.34    -0.84     0.48       3910 1.00
##
## Family Specific Parameters:
##     Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## phi     4.65      0.76     3.30     6.25       3137 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
# posterior predictive check: fit okish, could be better
pp_check(hun_brm_beta_mod)
```



We now provide a summary of this model based on the posterior distribution.

```
# binary pred set to false because all.r / all.l are coded as numeric 0 vs. 1
preds <- beta_summary(hun_brm_beta_mod, hun_aggr, rpred="all.r", lpred="all.l", binary_pred=F, printPlea
```

```
## predicted roughness rating based on R:
##    without R: -0.38 [-1.31,0.57]
##    with R: 1.75 [0.69,2.78]
## predicted roughness difference (R - no R):
##    diff: 2.13 [0.59,3.6]
##
## predicted roughness rating based on L:
##    without L: 0.64 [-0.12,1.39]
##    with L: 0.08 [-2.06,2.15]
## predicted roughness difference (L - no L):
##    diff: -0.56 [-2.87,1.62]
```

3.4 Bayesian logistic regression with roughness as outcome

Two logistic regression models with presence of L / presence of R as the outcome and binary roughness as the predictor. We can only run this model on the aggregated data: the proportion of /r/ is fixed within subjects (as they all rated the same stimuli) and also within stimuli, so we wouldn't be able to include those as fixed effects; but there are dependencies within subjects and items!

```
hun_brm_priors <- c(
  set_prior("student_t(5,0,2.5)", class = "b"),
  set_prior("student_t(5,0,2.5)", class = "Intercept"))
```

```
set.seed(314)
hun_brm_r_mod <- brm(all.r ~ rough,
                     data=hun_aggr,
                     prior=hun_brm_priors,
                     family="bernoulli",
                     refresh=0)
summary(hun_brm_r_mod)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: all.r ~ rough
##    Data: hun_aggr (Number of observations: 66)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept    -1.34      0.50    -2.35    -0.42       2354 1.00
## roughTRUE     1.62      0.58     0.49     2.82       2581 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
set.seed(314)
hun_brm_l_mod <- brm(all.l ~ rough,
                     data=hun_aggr,
                     prior=hun_brm_priors,
                     family="bernoulli",
                     refresh=0)
summary(hun_brm_l_mod)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: all.l ~ rough
##    Data: hun_aggr (Number of observations: 66)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept    -1.43      0.53    -2.53    -0.47       3306 1.00
## roughTRUE    -1.10      0.77    -2.69     0.36       1827 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Model predictions.

```
x <- logistic_summary(hun_brm_r_mod, eng, outcome="/r/", roughpred="rough")
```

```
## predicted probability of /r/ rating based on roughness:
##    smooth: 0.22 [0.09,0.4]
##    rough: 0.57 [0.42,0.71]
## predicted difference in probability of /r/ (rough - smooth):
```

```
##    diff: 0.35 [0.12,0.55]
```

```r
y <- logistic_summary(hun_brm_l_mod, eng, outcome="/l/", roughpred="rough")
```

```
## predicted probability of /l/ rating based on roughness:
##    smooth: 0.21 [0.07,0.38]
##    rough: 0.08 [0.02,0.18]
## predicted difference in probability of /l/ (rough - smooth):
##    diff: -0.12 [-0.31,0.04]
```

4. Cross-linguistic analysis

The latest version of our cross-linguistic data set includes 263 languages from 65 language families (counting isolates as families) from 17 Autotyp areas. Without Indo-European, there are 215 languages and 64 families from 17 Autotyp areas. We exclude Indo-European as we've already analysed these languages above.

```r
library(tidyverse)
library(brms)

source("scripts/rough_helper.r")

xling <- read_csv("final_data/cross_linguistic.csv")
```

```
## Parsed with column specification:
## cols(
##   Language = col_character(),
##   ISO_code = col_character(),
##   Phoible_code = col_character(),
##   Meaning = col_character(),
##   Form = col_character(),
##   Trill = col_character(),
##   Dataset = col_character(),
##   Family = col_character(),
##   Branch = col_character(),
##   Area = col_character(),
##   Continent = col_character(),
##   Rough.M = col_double(),
##   rough = col_logical(),
##   R_type = col_character(),
##   r = col_logical(),
##   l = col_logical()
## )
```

```r
length(unique(xling$Language))
```

```
## [1] 263
```

```r
length(unique(xling$Family))
```

```
## [1] 65
```

```r
length(unique(xling$Area))
```

```
## [1] 17
```

```r
xling <- filter(xling, Family!="Indo-European")
length(unique(xling$Language))
```

```
## [1] 216
```

```
length(unique(xling$Family))
```

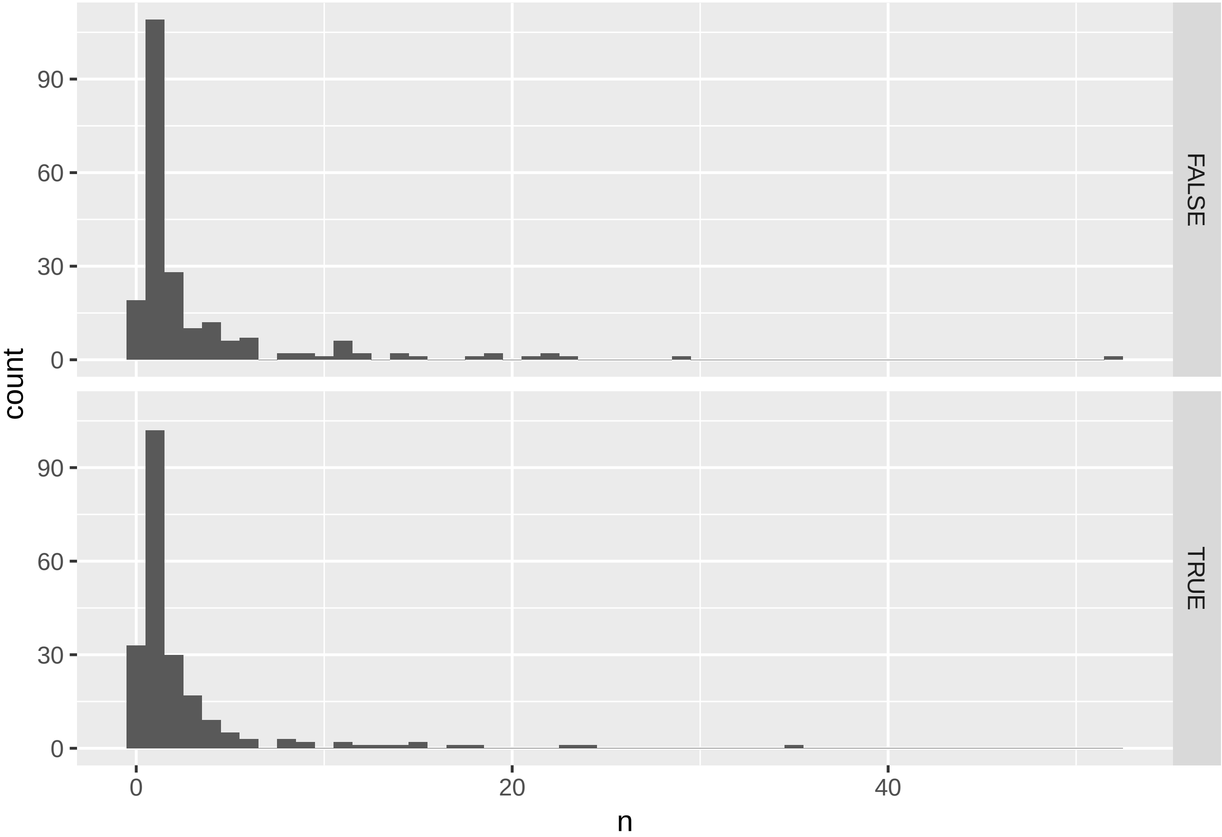
```
## [1] 64
```

```
length(unique(xling$Area))
```

```
## [1] 17
```

Some languages have a good number of rough/smooth words, but most have only 1 (the peak in the histogram below). Given how the data set was assembled, this one word is typically "rough"/"smooth". The data set is nicely balanced across rough vs. smooth words.

```
xling %>%
  dplyr::count(Language, rough) %>%
  complete(Language, rough, fill=list(n=0)) %>%
  ggplot(aes(x=n)) +
  facet_grid(rough ~.) +
  geom_histogram(binwidth=1)
```



Here is the plan for our cross-linguistic analysis: - We'll analyse languages with trills vs. those with no trills separately (as we only have a prediction for Ls with trills). We use a simple strategy here: the column /r/ codes for the presence of any kind of rhotic, but languages where Trill=="yes" only have trills. Therefore, we simply limit the range of languages to those with Trill=="yes" (and, for the complementary analysis, Trill=="no"). - We fit three different models: - roughness rating as the outcome, r / l as predictors, beta regression (full data set); random effects by Family, Area, Language - r / l as outcome, binary rough as predictor (full data set); random effects by Family, Area, Language - r / l as outcome, binary rough as predictor (limited specifically to the words rough / smooth); random effects by Family, Area

4.1 Cross-linguistic analysis of languages with trills

Limiting the data set to languages with trills only leaves 647 data points from 58 languages representing 21 families and 12 areas.

```
xling_trill <- filter(xling, Trill=="yes")
nrow(xling_trill)
```

```
## [1] 647
```

```
length(unique(xling_trill$Language))
```

```
## [1] 58
```

```
length(unique(xling_trill$Family))
```

```
## [1] 21
```

```
length(unique(xling_trill$Area))
```

```
## [1] 12
```

Further limiting the data set to words for rough/smooth only leaves 137 data points from 51 languages representing 21 families and 10 areas.

```
xling_trill_rs <- filter(xling_trill, Meaning %in% c("rough","smooth"))
nrow(xling_trill_rs)
```

```
## [1] 137
```

```
length(unique(xling_trill_rs$Language))
```

```
## [1] 51
```

```
length(unique(xling_trill_rs$Family))
```

```
## [1] 21
```

```
length(unique(xling_trill_rs$Area))
```

```
## [1] 10
```

Here are the raw proportions for the full trill data set:

```
xling_trill %>%
  group_by(Language, rough) %>%
  summarise(r_prop = mean(r)) %>%
  ungroup() %>%
  group_by(rough) %>%
  summarise(r_prop = mean(r_prop)) %>%
  ungroup()
```

```
## # A tibble: 2 x 2
##   rough r_prop
##   <lgl>  <dbl>
## 1 FALSE  0.264
## 2 TRUE   0.487
```

```
xling_trill %>%
  group_by(Language, rough) %>%
  summarise(l_prop = mean(l)) %>%
  ungroup() %>%
  group_by(rough) %>%
  summarise(l_prop = mean(l_prop)) %>%
  ungroup()
```

```
## # A tibble: 2 x 2
##   rough l_prop
##   <lgl>  <dbl>
## 1 FALSE  0.488
## 2 TRUE   0.225
```

And for the words rough/smooths only: (note the huge difference in proportions!)

```
xling_trill_rs %>%
  group_by(Language, rough) %>%
  summarise(r_prop = mean(r)) %>%
  ungroup() %>%
  group_by(rough) %>%
  summarise(r_prop = mean(r_prop)) %>%
  ungroup()
```

```
## # A tibble: 2 x 2
##   rough r_prop
##   <lgl>  <dbl>
## 1 FALSE  0.126
## 2 TRUE   0.579
```

```
xling_trill_rs %>%
  group_by(Language, rough) %>%
  summarise(l_prop = mean(l)) %>%
  ungroup() %>%
  group_by(rough) %>%
  summarise(l_prop = mean(l_prop)) %>%
  ungroup()
```

```
## # A tibble: 2 x 2
##   rough l_prop
##   <lgl>  <dbl>
## 1 FALSE  0.522
## 2 TRUE   0.130
```

4.1.1 Beta regression for languages with trills

We model the English-based ratings as a function of the presence of r/l in the cross-linguistic data set. We include random effects by Family/Area/Language. (I don't think "Meaning" can be included as a random factor here, as words with the same meaning always have the same roughness rating)

```
# outcome for beta regression
xling_trill$roughness.beta <- (xling_trill$Rough.M + 7) / 14

# priors: since the outcomes here are similar to those for a logistic model (i.e. a number between 0/1)
# and the link function is the same, we use the same priors as we do for logistic models

xling_brm_beta_priors <- c(
  set_prior("student_t(5,0,2.5)", class = "b"),
  set_prior("student_t(5,0,2.5)", class = "Intercept"),
  set_prior("lkj(2)", class = "cor"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Family"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "rTRUE", group="Family"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "lTRUE", group="Family"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Area"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "rTRUE", group="Area"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "lTRUE", group="Area"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Language"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "rTRUE", group="Language"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "lTRUE", group="Language")
)

set.seed(314)
xling_brm_beta_mod <- brm(roughness.beta ~ r + l +
                          (1 + r + l | Family) +
                          (1 + r + l | Area) +
                          (1 + r + l | Language),
                        data=xling_trill,
                        prior=xling_brm_beta_priors,
                        family="beta",
                       control=list(adapt_delta=0.9),
                       refresh=0)
```

Model predictions as usual: (note that conf interval around difference by R does not include 0)

```
preds <- beta_summary(xling_brm_beta_mod, dat=xling_trill, rpred="r", lpred="l", binary_pred=T, printPl
```

```
## predicted roughness rating based on R:
##    without R: -1.97 [-2.88,-0.98]
##    with R: -0.27 [-1.81,1.23]
## predicted roughness difference (R - no R):
##    diff: 1.69 [-0.03,3.33]
##
## predicted roughness rating based on L:
##    without L: -0.84 [-1.85,0.07]
##    with L: -2 [-3.33,-0.49]
## predicted roughness difference (L - no L):
##    diff: -1.16 [-2.64,0.58]
```

4.1.2 Logistic regression for all words in languages with trills

We model the presence of r/l as a function of roughness (binary) in the cross-linguistic data set. We include random effects by Family/Area/Language/Meaning.

```
# priors: since the outcomes here are similar to those for a logistic model (i.e. a number between 0/1)
# and the link function is the same, we use the same priors as we do for logistic models

xling_brm_logistic_priors <- c(
  set_prior("student_t(5,0,2.5)", class = "b"),
  set_prior("student_t(5,0,2.5)", class = "Intercept"),
  set_prior("lkj(2)", class = "cor"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Family"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "roughTRUE", group="Family"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Area"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "roughTRUE", group="Area"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Language"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "roughTRUE", group="Language"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Meaning")
)

set.seed(314)
xling_brm_logistic_mod_r <- brm(r ~ rough +
                          (1 + rough | Family) +
                          (1 + rough | Area) +
                          (1 + rough | Language) +
                          (1 | Meaning),
                        data=xling_trill,
                        prior=xling_brm_logistic_priors,
                        family="bernoulli",
                       control=list(adapt_delta=0.9),
                       refresh=0)
set.seed(314)
xling_brm_logistic_mod_l <- brm(l ~ rough +
                          (1 + rough | Family) +
                          (1 + rough | Area) +
                          (1 + rough | Language) +
                          (1 | Meaning),
                        data=xling_trill,
                        prior=xling_brm_logistic_priors,
                        family="bernoulli",
                       control=list(adapt_delta=0.9),
                       refresh=0)
```

Model predictions as usual. Note that neither the /l/ nor the /r/ pattern seem strong when our "rough" predictor does not take into account the actual amount of roughness / smoothness for specific words.

```
x <- logistic_summary(xling_brm_logistic_mod_r, dat=xling_trill, outcome="/r/", roughpred="rough")
```

```
## predicted probability of /r/ rating based on roughness:
##    smooth: 0.31 [0.17,0.46]
##    rough: 0.42 [0.17,0.67]
## predicted difference in probability of /r/ (rough - smooth):
##    diff: 0.12 [-0.14,0.36]
```

```
y <- logistic_summary(xling_brm_logistic_mod_l, dat=xling_trill, outcome="/l/", roughpred="rough")
```

```
## predicted probability of /l/ rating based on roughness:
##    smooth: 0.5 [0.35,0.64]
##    rough: 0.33 [0.13,0.61]
## predicted difference in probability of /l/ (rough - smooth):
##    diff: -0.17 [-0.4,0.12]
```

4.1.3 Logistic regression for rough/smooth in languages with trills

We model the presence of r/l as a function of roughness (binary) in the cross-linguistic data set limited to the words rough/smooth only (the expectation is that the effect would be particularly strong here). We include random effects by Family/Area/Language.

```
# priors: since the outcomes here are similar to those for a logistic model (i.e. a number between 0/1)
# and the link function is the same, we use the same priors as we do for logistic models
```

```
xling_brm_rs_logistic_priors <- c(
  set_prior("student_t(5,0,2.5)", class = "b"),
  set_prior("student_t(5,0,2.5)", class = "Intercept"),
  set_prior("lkj(2)", class = "cor"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Family"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "roughTRUE", group="Family"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Area"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "roughTRUE", group="Area")
)

set.seed(314)
xling_brm_rs_logistic_mod_r <- brm(r ~ rough +
                          (1 + rough | Family) +
                          (1 + rough | Area),
                        data=xling_trill_rs,
                        prior=xling_brm_rs_logistic_priors,
                        family="bernoulli",
                       control=list(adapt_delta=0.9),
                       refresh=0)
set.seed(314)
xling_brm_rs_logistic_mod_l <- brm(l ~ rough +
                          (1 + rough | Family) +
                          (1 + rough | Area),
                        data=xling_trill_rs,
                        prior=xling_brm_rs_logistic_priors,
                        family="bernoulli",
                       control=list(adapt_delta=0.9),
                       refresh=0)
```

Model predictions as usual. When focusing on these two extreme words only, the pattern seems to be there.

```
x <- logistic_summary(xling_brm_rs_logistic_mod_r, dat=xling_trill, outcome="/r/", roughpred="rough")
```

```
## predicted probability of /r/ rating based on roughness:
##    smooth: 0.08 [0.01,0.21]
##    rough: 0.51 [0.11,0.85]
## predicted difference in probability of /r/ (rough - smooth):
##    diff: 0.42 [0.05,0.75]
```

```
y <- logistic_summary(xling_brm_rs_logistic_mod_l, dat=xling_trill, outcome="/l/", roughpred="rough")
```

```
## predicted probability of /l/ rating based on roughness:
##    smooth: 0.51 [0.24,0.77]
##    rough: 0.13 [0.01,0.45]
## predicted difference in probability of /l/ (rough - smooth):
##    diff: -0.37 [-0.66,-0.01]
```

4.2 Cross-linguistic analysis of languages *without* trills

Limiting the data set to languages without trills (or where it's not clear whether the language has a trill) leaves 612 data points from 160 languages representing 54 families and 15 areas.

```
xling_no_trill <- filter(xling, is.na(Trill) | Trill!="yes")
nrow(xling_no_trill)
```

```
## [1] 612
```

```
length(unique(xling_no_trill$Language))
```

```
## [1] 160
```

```
length(unique(xling_no_trill$Family))
```

```
## [1] 54
```

```
length(unique(xling_no_trill$Area))
```

```
## [1] 15
```

Further limiting the data set to words for rough/smooth only leaves 303 data points from 123 languages representing 51 families and 15 areas.

```
xling_no_trill_rs <- filter(xling_no_trill, Meaning %in% c("rough","smooth"))
nrow(xling_no_trill_rs)
```

```
## [1] 303
```

```
length(unique(xling_no_trill_rs$Language))
```

```
## [1] 123
```

```
length(unique(xling_no_trill_rs$Family))
```

```
## [1] 51
```

```
length(unique(xling_no_trill_rs$Area))
```

```
## [1] 15
```

Here are the raw proportions for the full no-trill data set. Note that the pattern for /r/ disappears completely, while the pattern for /l/ remains pretty much the same.

```
xling_no_trill %>%
  group_by(Language, rough) %>%
  summarise(r_prop = mean(r)) %>%
  ungroup() %>%
  group_by(rough) %>%
  summarise(r_prop = mean(r_prop)) %>%
  ungroup()
```

```
## # A tibble: 2 x 2
##   rough r_prop
##   <lgl>  <dbl>
## 1 FALSE  0.255
## 2 TRUE   0.257
```

```
xling_no_trill %>%
  group_by(Language, rough) %>%
  summarise(l_prop = mean(l)) %>%
  ungroup() %>%
  group_by(rough) %>%
  summarise(l_prop = mean(l_prop)) %>%
  ungroup()
```

```
## # A tibble: 2 x 2
##   rough l_prop
##   <lgl>  <dbl>
## 1 FALSE  0.364
## 2 TRUE   0.202
```

And for the words rough/smooth only. There's still not much for /r/ (though the pattern gets a little stronger), but the /l/ pattern also does not become much clearer. I wonder if this is because the quality of this subset is much lower due to the inclusion of languages where no info about trills was available – and which are therefore likely much noiser in terms of the data (mistranslations? wrong transcriptions? etc.). Also, some of these languages likely do have trills…

```r
xling_no_trill_rs %>%
  group_by(Language, rough) %>%
  summarise(r_prop = mean(r)) %>%
  ungroup() %>%
  group_by(rough) %>%
  summarise(r_prop = mean(r_prop)) %>%
  ungroup()
```

```
## # A tibble: 2 x 2
##   rough r_prop
##   <lgl>  <dbl>
## 1 FALSE  0.235
## 2 TRUE   0.310
```

```r
xling_no_trill_rs %>%
  group_by(Language, rough) %>%
  summarise(l_prop = mean(l)) %>%
  ungroup() %>%
  group_by(rough) %>%
  summarise(l_prop = mean(l_prop)) %>%
  ungroup()
```

```
## # A tibble: 2 x 2
##   rough l_prop
##   <lgl>  <dbl>
## 1 FALSE  0.316
## 2 TRUE   0.160
```

4.1.1 Beta regression for languages with no trills

We model the English-based ratings as a function of the presence of r/l in the cross-linguistic data set. We include random effects by Family/Area/Language. (I don't think "Meaning" can be included as a random factor here, as words with the same meaning always have the same roughness rating)

```r
# outcome for beta regression
xling_no_trill$roughness.beta <- (xling_no_trill$Rough.M + 7) / 14

# priors: since the outcomes here are similar to those for a logistic model (i.e. a number between 0/1)
# and the link function is the same, we use the same priors as we do for logistic models

xling_brm_beta_priors <- c(
  set_prior("student_t(5,0,2.5)", class = "b"),
  set_prior("student_t(5,0,2.5)", class = "Intercept"),
  set_prior("lkj(2)", class = "cor"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Family"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "rTRUE", group="Family"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "lTRUE", group="Family"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Area"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "rTRUE", group="Area"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "lTRUE", group="Area"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Language"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "rTRUE", group="Language"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "lTRUE", group="Language")
)

set.seed(314)
xling_nt_brm_beta_mod <- brm(roughness.beta ~ r + l +
                          (1 + r + l | Family) +
                          (1 + r + l | Area) +
                          (1 + r + l | Language),
                        data=xling_no_trill,
                        prior=xling_brm_beta_priors,
                        family="beta",
                       control=list(adapt_delta=0.9),
                       refresh=0)
```

Model predictions as usual: (note that conf interval around difference by R does not include 0)

```
preds <- beta_summary(xling_nt_brm_beta_mod, dat=xling_trill, rpred="r", lpred="l", binary_pred=T, prin
```

```
## predicted roughness rating based on R:
##    without R: -1.64 [-2.29,-0.96]
##    with R: -0.92 [-2.12,0.45]
## predicted roughness difference (R - no R):
##    diff: 0.72 [-0.52,2.14]
##
## predicted roughness rating based on L:
##    without L: -1.07 [-1.74,-0.39]
##    with L: -1.76 [-2.89,-0.44]
## predicted roughness difference (L - no L):
##    diff: -0.69 [-1.86,0.66]
```

4.1.2 Logistic regression for all words in languages with trills

We model the presence of r/l as a function of roughness (binary) in the cross-linguistic data set. We include random effects by Family/Area/Language/Meaning.

```
# priors: since the outcomes here are similar to those for a logistic model (i.e. a number between 0/1)
# and the link function is the same, we use the same priors as we do for logistic models

xling_brm_logistic_priors <- c(
  set_prior("student_t(5,0,2.5)", class = "b"),
  set_prior("student_t(5,0,2.5)", class = "Intercept"),
  set_prior("lkj(2)", class = "cor"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Family"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "roughTRUE", group="Family"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Area"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "roughTRUE", group="Area"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Language"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "roughTRUE", group="Language"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Meaning")
)
```

```
set.seed(314)
xling_nt_brm_logistic_mod_r <- brm(r ~ rough +
                          (1 + rough | Family) +
                          (1 + rough | Area) +
                          (1 + rough | Language) +
                          (1 | Meaning),
                        data=xling_no_trill,
                        prior=xling_brm_logistic_priors,
                        family="bernoulli",
                       control=list(adapt_delta=0.9),
                       refresh=0)
set.seed(314)
xling_nt_brm_logistic_mod_l <- brm(l ~ rough +
                          (1 + rough | Family) +
                          (1 + rough | Area) +
                          (1 + rough | Language) +
                          (1 | Meaning),
                        data=xling_no_trill,
                        prior=xling_brm_logistic_priors,
                        family="bernoulli",
                       control=list(adapt_delta=0.9),
                       refresh=0)
```

Model predictions as usual. No strong patterns either for /r/ or for /l/.

```
x <- logistic_summary(xling_nt_brm_logistic_mod_r, dat=xling_trill, outcome="/r/", roughpred="rough")
```

```
## predicted probability of /r/ rating based on roughness:
##    smooth: 0.18 [0.05,0.38]
##    rough: 0.14 [0.03,0.31]
## predicted difference in probability of /r/ (rough - smooth):
##    diff: -0.04 [-0.24,0.14]
```

```
y <- logistic_summary(xling_nt_brm_logistic_mod_l, dat=xling_trill, outcome="/l/", roughpred="rough")
```

```
## predicted probability of /l/ rating based on roughness:
##    smooth: 0.32 [0.13,0.55]
##    rough: 0.21 [0.03,0.52]
## predicted difference in probability of /l/ (rough - smooth):
##    diff: -0.11 [-0.38,0.21]
```

4.1.3 Logistic regression for rough/smooth in languages with trills

We model the presence of r/l as a function of roughness (binary) in the cross-linguistic data set limited to the words rough/smooth only (the expectation is that the effect would be particularly strong here). We include random effects by Family/Area/Language.

```
# priors: since the outcomes here are similar to those for a logistic model (i.e. a number between 0/1)
# and the link function is the same, we use the same priors as we do for logistic models

xling_brm_rs_logistic_priors <- c(
  set_prior("student_t(5,0,2.5)", class = "b"),
  set_prior("student_t(5,0,2.5)", class = "Intercept"),
  set_prior("lkj(2)", class = "cor"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Family"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "roughTRUE", group="Family"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "Intercept", group="Area"),
  set_prior("student_t(4,0,2)", class = "sd", coef = "roughTRUE", group="Area")
)

set.seed(314)
xling_nt_brm_rs_logistic_mod_r <- brm(r ~ rough +
                          (1 + rough | Family) +
                          (1 + rough | Area),
                        data=xling_no_trill_rs,
                        prior=xling_brm_rs_logistic_priors,
                        family="bernoulli",
                       control=list(adapt_delta=0.9),
                       refresh=0)
set.seed(314)
xling_nt_brm_rs_logistic_mod_l <- brm(l ~ rough +
                          (1 + rough | Family) +
                          (1 + rough | Area),
                        data=xling_no_trill_rs,
                        prior=xling_brm_rs_logistic_priors,
                        family="bernoulli",
                       control=list(adapt_delta=0.9),
                       refresh=0)
```

Model predictions as usual. No patterns!

```
x <- logistic_summary(xling_nt_brm_rs_logistic_mod_r, dat=xling_trill, outcome="/r/", roughpred="rough"
```

```
## predicted probability of /r/ rating based on roughness:
##    smooth: 0.14 [0.05,0.27]
##    rough: 0.22 [0.07,0.42]
## predicted difference in probability of /r/ (rough - smooth):
##    diff: 0.08 [-0.07,0.25]
```

```
y <- logistic_summary(xling_nt_brm_rs_logistic_mod_l, dat=xling_trill, outcome="/l/", roughpred="rough"
```

```
## predicted probability of /l/ rating based on roughness:
##    smooth: 0.24 [0.09,0.42]
##    rough: 0.1 [0.02,0.25]
## predicted difference in probability of /l/ (rough - smooth):
##    diff: -0.13 [-0.28,0.01]
```