Analysis code

Tristan Mahr

First let’s load in the data.

library(tidyverse)  
#> Warning: package 'tibble' was built under R version 4.0.3  
model\_data <- drake::readd("data\_paper\_model\_counts") %>%   
 filter(aligner != "Manual")  
  
# See a few values from each column  
glimpse(model\_data)  
#> Rows: 840  
#> Columns: 10  
#> $ class <chr> "fricatives", "fricatives", "fricatives", "fricatives"...  
#> $ aligner <fct> Kaldi, Kaldi, Kaldi, Kaldi, Kaldi, Kaldi, Kaldi, Kaldi...  
#> $ child <chr> "c01", "c02", "c03", "c04", "c05", "c06", "c07", "c08"...  
#> $ age <dbl> 50, 39, 41, 76, 43, 75, 50, 60, 55, 48, 57, 77, 78, 70...  
#> $ age\_scale <dbl> -0.83333333, -1.75000000, -1.58333333, 1.33333333, -1....  
#> $ n\_files <int> 67, 37, 38, 68, 28, 67, 47, 65, 57, 50, 68, 67, 68, 68...  
#> $ n\_intervals <int> 176, 73, 73, 183, 46, 177, 111, 173, 149, 121, 182, 18...  
#> $ n\_matches <int> 106, 36, 49, 157, 25, 149, 65, 130, 103, 73, 162, 138,...  
#> $ n\_misses <int> 70, 37, 24, 26, 21, 28, 46, 43, 46, 48, 20, 42, 25, 39...  
#> $ prop\_matches <dbl> 0.602, 0.493, 0.671, 0.858, 0.543, 0.842, 0.586, 0.751...

We have aggregated data with one row per child per aligner per sound class. age is provided in months or rescaled age\_scale. Aggregation statistics include the number of files that were aligned by all 5 alignment algorithms (n\_files), the number of intervals across the files (n\_intervals), the number of matches and misses (n\_matches, n\_misses) and the proportion of matches (prop\_matches).

Let’s start with a logistic regression model with fixed effects for phoneme class, aligner, and class-by-aligner interactions and random by-child random intercepts and by-child-by-aligner random intercepts.

library(lme4)  
#> Warning: package 'lme4' was built under R version 4.0.3  
#> Loading required package: Matrix  
#>   
#> Attaching package: 'Matrix'  
#> The following objects are masked from 'package:tidyr':  
#>   
#> expand, pack, unpack  
  
# Sound class x aligner interactions  
m\_class\_int <- glmer(  
 cbind(n\_matches, n\_misses) ~ class \* aligner + (1 | child/aligner),   
 family = binomial,   
 model\_data  
)

By-child intercepts allow children to vary in alignment accuracy (some children will be easier or more difficult across all the aligners). By-child-by-aligner intercepts allows the alignment accuracy to vary by aligner and by child (some children will easier or harder to align for particular aligners). Alternative random effects structures are plausible, but this structure is simple (matches are nested in children and aligner errors are nested in children) and yields model convergence.

summary(m\_class\_int)  
#> Generalized linear mixed model fit by maximum likelihood (Laplace  
#> Approximation) [glmerMod]  
#> Family: binomial ( logit )  
#> Formula: cbind(n\_matches, n\_misses) ~ class \* aligner + (1 | child/aligner)  
#> Data: model\_data  
#>   
#> AIC BIC logLik deviance df.resid   
#> 6847.6 6951.8 -3401.8 6803.6 818   
#>   
#> Scaled residuals:   
#> Min 1Q Median 3Q Max   
#> -4.9270 -0.9098 0.0501 0.9505 4.6142   
#>   
#> Random effects:  
#> Groups Name Variance Std.Dev.  
#> aligner:child (Intercept) 0.04887 0.2211   
#> child (Intercept) 0.05413 0.2326   
#> Number of obs: 840, groups: aligner:child, 210; child, 42  
#>   
#> Fixed effects:  
#> Estimate Std. Error z value Pr(>|z|)   
#> (Intercept) 1.121585 0.057658 19.452 < 2e-16 \*\*\*  
#> classothers -0.198738 0.043431 -4.576 4.74e-06 \*\*\*  
#> classplosives -0.003951 0.038144 -0.104 0.917504   
#> classvowels 0.364351 0.037020 9.842 < 2e-16 \*\*\*  
#> alignerMFA (SAT) 0.739373 0.067438 10.964 < 2e-16 \*\*\*  
#> alignerMFA (No SAT) -0.014103 0.063718 -0.221 0.824832   
#> alignerP2FA -0.643346 0.062301 -10.326 < 2e-16 \*\*\*  
#> alignerProsodylab -0.738173 0.062237 -11.861 < 2e-16 \*\*\*  
#> classothers:alignerMFA (SAT) -0.241038 0.067174 -3.588 0.000333 \*\*\*  
#> classplosives:alignerMFA (SAT) -0.097876 0.060341 -1.622 0.104796   
#> classvowels:alignerMFA (SAT) 0.020632 0.059712 0.346 0.729695   
#> classothers:alignerMFA (No SAT) -0.061060 0.061038 -1.000 0.317136   
#> classplosives:alignerMFA (No SAT) -0.020203 0.053747 -0.376 0.707005   
#> classvowels:alignerMFA (No SAT) 0.401899 0.053735 7.479 7.47e-14 \*\*\*  
#> classothers:alignerP2FA 0.474701 0.059262 8.010 1.14e-15 \*\*\*  
#> classplosives:alignerP2FA 0.018713 0.050961 0.367 0.713466   
#> classvowels:alignerP2FA 0.322243 0.049697 6.484 8.92e-11 \*\*\*  
#> classothers:alignerProsodylab -0.002868 0.058233 -0.049 0.960725   
#> classplosives:alignerProsodylab -0.371961 0.050626 -7.347 2.02e-13 \*\*\*  
#> classvowels:alignerProsodylab 0.463956 0.049714 9.332 < 2e-16 \*\*\*  
#> ---  
#> Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
#>   
#> Correlation matrix not shown by default, as p = 20 > 12.  
#> Use print(x, correlation=TRUE) or  
#> vcov(x) if you need it

We do not worry about interpreting these coefficients. They are on the log-odds scale and involve interactions of two dummy-coded categorical variables. We instead look at the estimated marginal (cell) means using the emmeans package.

First, we have to prepare a variance estimate for the bias-adjustment that’s used by emmeans. A description of bias-adjustment procedure is available online <https://cran.r-project.org/web/packages/emmeans/vignettes/transformations.html#cbpp>.

variances <- broom.mixed::tidy(m\_class\_int, "ran\_pars", scales = "vcov")  
#> Registered S3 method overwritten by 'broom.mixed':  
#> method from   
#> tidy.gamlss broom  
sigma <- sqrt(sum(variances$estimate))

### Question 1 (Accuracy by aligner)

We get the marginal means on the probability scale (type = "response") for each aligner (~ aligner).

library(emmeans)  
#> Warning: package 'emmeans' was built under R version 4.0.3  
  
means\_by\_aligner <- emmeans(  
 m\_class\_int,   
 spec = ~ aligner,   
 type = "response",   
 bias.adjust = TRUE,   
 sigma = sigma  
)  
#> NOTE: Results may be misleading due to involvement in interactions

Our model used included aligner-by-class interactions, but we are ignoring them for this estimate and averaging over the sound classes. The software prints NOTE to remind of this fact. Here are the means:

means\_by\_aligner  
#> aligner prob SE df asymp.LCL asymp.UCL  
#> Kaldi 0.757 0.00932 Inf 0.738 0.775  
#> MFA (SAT) 0.856 0.00638 Inf 0.843 0.868  
#> MFA (No SAT) 0.769 0.00902 Inf 0.750 0.786  
#> P2FA 0.669 0.01109 Inf 0.647 0.691  
#> Prosodylab 0.607 0.01191 Inf 0.583 0.630  
#>   
#> Results are averaged over the levels of: class   
#> Confidence level used: 0.95   
#> Intervals are back-transformed from the logit scale   
#> Bias adjustment applied based on sigma = 0.32093

The prob is the estimated probability of a matching alignment, and asymp.LCL and asymp.UCL is a 95% interval on that estimated probability. These marginal means are pretty close to the means of the child-level proportions:

model\_data %>%   
 group\_by(aligner) %>%   
 summarise(  
 empirical\_proportion = mean(prop\_matches),  
 .groups = "drop"  
 ) %>%   
 left\_join(  
 broom::tidy(means\_by\_aligner, conf.int = TRUE),   
 by = c("aligner")  
 ) %>%   
 select(  
 aligner,   
 empirical\_proportion,   
 marginal\_probability = prob,   
 lower\_ci = asymp.LCL,   
 upper\_ci = asymp.UCL  
 )  
#> # A tibble: 5 x 5  
#> aligner empirical\_proportion marginal\_probability lower\_ci upper\_ci  
#> <chr> <dbl> <dbl> <dbl> <dbl>  
#> 1 Kaldi 0.752 0.757 0.738 0.775  
#> 2 MFA (SAT) 0.854 0.856 0.843 0.868  
#> 3 MFA (No SAT) 0.762 0.769 0.750 0.786  
#> 4 P2FA 0.667 0.669 0.647 0.691  
#> 5 Prosodylab 0.599 0.607 0.583 0.630

Now, we can compare means to each other. We look at all pairwise differences between aligners (pairwise ~ aligner). emmeans handles the contrast-coding and the *p*-value adjustment automatically. Here we use the Bonferroni method.

by\_aligner <- emmeans(  
 m\_class\_int,   
 spec = pairwise ~ aligner,   
 type = "response",   
 bias.adjust = TRUE,   
 sigma = sigma,  
 adjust = "bonferroni"  
)  
#> NOTE: Results may be misleading due to involvement in interactions  
by\_aligner$contrasts  
#> contrast odds.ratio SE df z.ratio p.value  
#> Kaldi / MFA (SAT) 0.544 0.0290 Inf -12.386 <.0001   
#> Kaldi / MFA (No SAT) 0.984 0.0516 Inf -1.261 1.0000   
#> Kaldi / P2FA 1.632 0.0849 Inf 8.447 <.0001   
#> Kaldi / Prosodylab 2.151 0.1118 Inf 13.779 <.0001   
#> MFA (SAT) / MFA (No SAT) 1.904 0.1015 Inf 11.136 <.0001   
#> MFA (SAT) / P2FA 3.156 0.1672 Inf 20.756 <.0001   
#> MFA (SAT) / Prosodylab 4.162 0.2201 Inf 26.012 <.0001   
#> MFA (No SAT) / P2FA 1.743 0.0908 Inf 9.708 <.0001   
#> MFA (No SAT) / Prosodylab 2.298 0.1195 Inf 15.037 <.0001   
#> P2FA / Prosodylab 1.386 0.0716 Inf 5.354 <.0001   
#>   
#> Results are averaged over the levels of: class   
#> P value adjustment: bonferroni method for 10 tests   
#> Tests are performed on the log odds ratio scale   
#> Bias adjustment applied based on sigma = 0.32093

Differences here are in odds ratios like X / Y. Values greater than 1 indicate an advantage for X over Y, values less than 1 indicate an advantage for Y over X, and values of 1 indicates no difference in odds. Notably, there is not a significance difference between Kaldi and MFA (No SAT). These two are the most similar aligners in the comparison, a priori, because MFA is built on top of Kaldi and uses the same acoustic model behind the scenes.

### Question 2 (Accuracy by sound class)

For question 2, we change from aggregating by aligner (averaging over class) to aggregating by class (averaging over aligner). We otherwise use the same procedure.

by\_class <- emmeans(  
 m\_class\_int,   
 spec = pairwise ~ class,   
 type = "response",   
 bias.adjust = TRUE,   
 sigma = sigma,  
 adjust = "bonferroni"  
)  
#> NOTE: Results may be misleading due to involvement in interactions  
by\_class$emmeans  
#> class prob SE df asymp.LCL asymp.UCL  
#> fricatives 0.724 0.00806 Inf 0.708 0.740  
#> others 0.691 0.00869 Inf 0.674 0.708  
#> plosives 0.705 0.00826 Inf 0.688 0.721  
#> vowels 0.827 0.00573 Inf 0.815 0.838  
#>   
#> Results are averaged over the levels of: aligner   
#> Confidence level used: 0.95   
#> Intervals are back-transformed from the logit scale   
#> Bias adjustment applied based on sigma = 0.32093  
by\_class$contrasts  
#> contrast odds.ratio SE df z.ratio p.value  
#> fricatives / others 1.240 0.02415 Inf 8.461 <.0001   
#> fricatives / plosives 1.160 0.01986 Inf 5.738 <.0001   
#> fricatives / vowels 0.574 0.00979 Inf -35.509 <.0001   
#> others / plosives 0.984 0.01774 Inf -3.691 0.0013   
#> others / vowels 0.486 0.00875 Inf -42.847 <.0001   
#> plosives / vowels 0.520 0.00801 Inf -45.721 <.0001   
#>   
#> Results are averaged over the levels of: aligner   
#> P value adjustment: bonferroni method for 6 tests   
#> Tests are performed on the log odds ratio scale   
#> Bias adjustment applied based on sigma = 0.32093

Here the vowels have a consistent advantage over the other classes.

For question 2, we also looked at all of class-by-aligner effects. We use effect coding (eff ~ ...) to compare each phone-by-class mean to the average of the other phone-by-class means. Again, emmeans handles the contrast and the *p*-value adjustment automatically. Here it uses "fdr" adjustment (false discovery rate, Benjamini & Hochberg, 1995).

by\_aligner\_class <- emmeans(  
 m\_class\_int,   
 spec = eff ~ class \* aligner,   
 type = "response",   
 bias.adjust = TRUE,   
 sigma = sigma  
)

First, here are the means.

by\_aligner\_class$emmeans  
#> class aligner prob SE df asymp.LCL asymp.UCL  
#> fricatives Kaldi 0.749 0.01062 Inf 0.728 0.770  
#> others Kaldi 0.711 0.01191 Inf 0.687 0.734  
#> plosives Kaldi 0.749 0.01023 Inf 0.728 0.768  
#> vowels Kaldi 0.811 0.00827 Inf 0.794 0.826  
#> fricatives MFA (SAT) 0.861 0.00729 Inf 0.846 0.875  
#> others MFA (SAT) 0.801 0.00968 Inf 0.781 0.819  
#> plosives MFA (SAT) 0.849 0.00734 Inf 0.834 0.862  
#> vowels MFA (SAT) 0.901 0.00514 Inf 0.890 0.910  
#> fricatives MFA (No SAT) 0.747 0.01068 Inf 0.725 0.767  
#> others MFA (No SAT) 0.696 0.01220 Inf 0.671 0.719  
#> plosives MFA (No SAT) 0.742 0.01037 Inf 0.721 0.762  
#> vowels MFA (No SAT) 0.863 0.00657 Inf 0.849 0.875  
#> fricatives P2FA 0.614 0.01294 Inf 0.589 0.640  
#> others P2FA 0.676 0.01254 Inf 0.651 0.700  
#> plosives P2FA 0.618 0.01249 Inf 0.593 0.642  
#> vowels P2FA 0.757 0.00971 Inf 0.738 0.776  
#> fricatives Prosodylab 0.592 0.01317 Inf 0.566 0.618  
#> others Prosodylab 0.544 0.01392 Inf 0.517 0.571  
#> plosives Prosodylab 0.502 0.01315 Inf 0.476 0.528  
#> vowels Prosodylab 0.766 0.00951 Inf 0.747 0.784  
#>   
#> Confidence level used: 0.95   
#> Intervals are back-transformed from the logit scale   
#> Bias adjustment applied based on sigma = 0.32093

More importantly, here are the contrasts.

by\_aligner\_class$contrasts  
#> contrast odds.ratio SE df z.ratio p.value  
#> fricatives Kaldi effect 1.100 0.0461 Inf 1.087 0.3193   
#> others Kaldi effect 0.902 0.0395 Inf -3.501 0.0006   
#> plosives Kaldi effect 1.096 0.0428 Inf 1.064 0.3193   
#> vowels Kaldi effect 1.584 0.0604 Inf 10.757 <.0001   
#> fricatives MFA (SAT) effect 2.305 0.1076 Inf 16.816 <.0001   
#> others MFA (SAT) effect 1.485 0.0692 Inf 7.405 <.0001   
#> plosives MFA (SAT) effect 2.082 0.0876 Inf 16.235 <.0001   
#> vowels MFA (SAT) effect 3.387 0.1431 Inf 27.699 <.0001   
#> fricatives MFA (No SAT) effect 1.085 0.0453 Inf 0.751 0.4762   
#> others MFA (No SAT) effect 0.837 0.0363 Inf -5.265 <.0001   
#> plosives MFA (No SAT) effect 1.059 0.0412 Inf 0.186 0.8527   
#> vowels MFA (No SAT) effect 2.335 0.0931 Inf 20.001 <.0001   
#> fricatives P2FA effect 0.578 0.0230 Inf -15.025 <.0001   
#> others P2FA effect 0.762 0.0327 Inf -7.491 <.0001   
#> plosives P2FA effect 0.587 0.0220 Inf -15.546 <.0001   
#> vowels P2FA effect 1.149 0.0425 Inf 2.397 0.0207   
#> fricatives Prosodylab effect 0.526 0.0209 Inf -17.450 <.0001   
#> others Prosodylab effect 0.430 0.0180 Inf -21.395 <.0001   
#> plosives Prosodylab effect 0.361 0.0134 Inf -28.709 <.0001   
#> vowels Prosodylab effect 1.204 0.0448 Inf 3.648 0.0004   
#>   
#> P value adjustment: fdr method for 20 tests   
#> Tests are performed on the log odds ratio scale   
#> Bias adjustment applied based on sigma = 0.32093

In the manuscript, we report the classes for each aligner with the largest and smallest effects. Vowels have the largest effect for each aligner. The smallest effects are other sounds for Kaldi and the MFA aligners, plosives and fricatives for P2FA and plosives for Prosodylab.

### Question 3 (Age effects)

We now augment the model to include age. We consider a model with just 2-way interactions and model with a full 3-way interactions.

# Add age, age x aligner, age x class effects  
m\_class\_int\_age\_2way <- glmer(  
 cbind(n\_matches, n\_misses) ~   
 class \* aligner + age\_scale +   
 age\_scale:class + aligner:age\_scale +   
 (1 | child/aligner),   
 family = binomial,   
 model\_data  
)  
  
# Add full 3-way interaction  
m\_class\_int\_age\_3way <- glmer(  
 cbind(n\_matches, n\_misses) ~   
 class \* aligner \* age\_scale +   
 (1 | child/aligner),   
 family = binomial,   
 model\_data  
)  
#> Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :  
#> Model failed to converge with max|grad| = 0.00225522 (tol = 0.002, component 1)

We disregard the convergence warning here because the diagnostic  
max|grad| = 0.00225522 is barely above the threshold tol = 0.002.

Model comparison (lowest AIC, chi-square tests) supports the full, 3-way model.

format(  
 anova(m\_class\_int, m\_class\_int\_age\_2way, m\_class\_int\_age\_3way),   
 digits = 3  
)  
#> npar AIC BIC logLik deviance Chisq Df Pr(>Chisq)  
#> m\_class\_int 22 6848 6952 -3402 6804 NA NA NA  
#> m\_class\_int\_age\_2way 30 6555 6697 -3248 6495 308.3 8 7.20e-62  
#> m\_class\_int\_age\_3way 42 6520 6719 -3218 6436 59.2 12 3.11e-08

We want to compare the age-slopes between the aligners. emtrends() will compute the age trend (var = "age\_scale") for each aligner using effect coding (eff ~ aligner). We compare the slopes at 60 months (at = list(age\_scale = 0)).

variances2 <- broom.mixed::tidy(  
 m\_class\_int\_age\_3way,   
 "ran\_pars",   
 scales = "vcov"  
)  
sigma2 <- sqrt(sum(variances2$estimate))  
  
aligner\_age\_trends <- emtrends(  
 m\_class\_int\_age\_3way,   
 eff ~ aligner,   
 var = "age\_scale",   
 bias.adjust = TRUE,   
 sigma = sigma2,   
 at = list(age\_scale = 0)  
)  
#> NOTE: Results may be misleading due to involvement in interactions  
aligner\_age\_trends  
#> $emtrends  
#> aligner age\_scale.trend SE df asymp.LCL asymp.UCL  
#> Kaldi 0.2326 0.0418 Inf 0.1506 0.3146  
#> MFA (SAT) 0.0319 0.0430 Inf -0.0523 0.1161  
#> MFA (No SAT) 0.1731 0.0418 Inf 0.0910 0.2551  
#> P2FA 0.1746 0.0414 Inf 0.0934 0.2557  
#> Prosodylab -0.0357 0.0413 Inf -0.1168 0.0453  
#>   
#> Results are averaged over the levels of: class   
#> Confidence level used: 0.95   
#>   
#> $contrasts  
#> contrast estimate SE df z.ratio p.value  
#> Kaldi effect 0.1173 0.0259 Inf 4.533 <.0001   
#> MFA (SAT) effect -0.0834 0.0270 Inf -3.094 0.0033   
#> MFA (No SAT) effect 0.0578 0.0259 Inf 2.232 0.0256   
#> P2FA effect 0.0593 0.0255 Inf 2.327 0.0249   
#> Prosodylab effect -0.1510 0.0254 Inf -5.945 <.0001   
#>   
#> Results are averaged over the levels of: class   
#> P value adjustment: fdr method for 5 tests

The values here are on the logit (log-odds) scale. Exponentiating them produces odds ratios.

aligner\_age\_trends$emtrends %>%   
 broom::tidy(conf.int = TRUE) %>%   
 mutate(  
 or\_trend = exp(age\_scale.trend),  
 or\_lcl = exp(asymp.LCL),  
 or\_ucl = exp(asymp.UCL),  
 trend = age\_scale.trend,  
 se = std.error,  
 z = z.ratio,  
 p = p.value  
 ) %>%   
 select(aligner, starts\_with("or\_"), trend, se, z, p)  
#> # A tibble: 5 x 8  
#> aligner or\_trend or\_lcl or\_ucl trend se z p  
#> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
#> 1 Kaldi 1.26 1.16 1.37 0.233 0.0418 5.56 0.0000000269  
#> 2 MFA (SAT) 1.03 0.949 1.12 0.0319 0.0430 0.743 0.458   
#> 3 MFA (No SAT) 1.19 1.10 1.29 0.173 0.0418 4.14 0.0000354   
#> 4 P2FA 1.19 1.10 1.29 0.175 0.0414 4.22 0.0000249   
#> 5 Prosodylab 0.965 0.890 1.05 -0.0357 0.0413 -0.865 0.387

Both the MFA-SAT and Prosodylab 95% intervals overlap with 1 (odds ratio) so we conclude that there is not a statistically clear age trend.

To describe the effect size on the response scale, we compare the match probabilities for the aligner with the steepest age effect (Kaldi). We calculate the match probability at age 0 and age 1 (at = list(age\_scale = c(0, 1))) and then for each aligner compute pairwise differences between these probabilities to get the age 1 advantage over age 0 (revpairwise ~ age\_scale | aligner computed reverse-pairwise differences—age 1 / age 0 on the odds ratio scale—but we do not use them here.)

aligner\_age\_trends\_1\_vs\_0 <- emmeans(  
 m\_class\_int\_age\_3way,   
 revpairwise ~ age\_scale | aligner ,   
 bias.adjust = TRUE,   
 sigma = sigma2,  
 type = "response",  
 at = list(age\_scale = c(0, 1))  
)   
#> NOTE: Results may be misleading due to involvement in interactions  
  
aligner\_age\_trends\_1\_vs\_0$emmeans %>%   
 broom::tidy(conf.int = TRUE) %>%   
 filter(aligner == "Kaldi") %>%   
 select(age\_scale, aligner, prob, asymp.LCL, asymp.UCL)  
#> # A tibble: 2 x 5  
#> age\_scale aligner prob asymp.LCL asymp.UCL  
#> <dbl> <chr> <dbl> <dbl> <dbl>  
#> 1 0 Kaldi 0.758 0.741 0.773  
#> 2 1 Kaldi 0.797 0.777 0.816

Now, we can do the converse aggregation and averaging: Look at the age trend for each sound class averaging over aligners.

class\_age\_trends <- emtrends(  
 m\_class\_int\_age\_3way,   
 eff ~ class,   
 var = "age\_scale",   
 bias.adjust = TRUE,   
 sigma = sigma2,   
 at = list(age\_scale = 0)  
)  
#> NOTE: Results may be misleading due to involvement in interactions  
class\_age\_trends  
#> $emtrends  
#> class age\_scale.trend SE df asymp.LCL asymp.UCL  
#> fricatives 0.25805 0.0349 Inf 0.18970 0.3264  
#> others 0.13961 0.0353 Inf 0.07042 0.2088  
#> plosives 0.05992 0.0340 Inf -0.00671 0.1266  
#> vowels 0.00356 0.0341 Inf -0.06318 0.0703  
#>   
#> Results are averaged over the levels of: aligner   
#> Confidence level used: 0.95   
#>   
#> $contrasts  
#> contrast estimate SE df z.ratio p.value  
#> fricatives effect 0.1428 0.01112 Inf 12.844 <.0001   
#> others effect 0.0243 0.01174 Inf 2.072 0.0383   
#> plosives effect -0.0554 0.00968 Inf -5.721 <.0001   
#> vowels effect -0.1117 0.00973 Inf -11.483 <.0001   
#>   
#> Results are averaged over the levels of: aligner   
#> P value adjustment: fdr method for 4 tests

And likewise, look at odds ratios.

class\_age\_trends$emtrends %>%   
 broom::tidy(conf.int = TRUE) %>%   
 mutate(  
 or\_trend = exp(age\_scale.trend),  
 or\_lcl = exp(asymp.LCL),  
 or\_ucl = exp(asymp.UCL),  
 trend = age\_scale.trend,  
 se = std.error,  
 z = z.ratio,  
 p = p.value  
 ) %>%   
 select(class, starts\_with("or\_"), trend, se, z, p)  
#> # A tibble: 4 x 8  
#> class or\_trend or\_lcl or\_ucl trend se z p  
#> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
#> 1 fricatives 1.29 1.21 1.39 0.258 0.0349 7.40 1.36e-13  
#> 2 others 1.15 1.07 1.23 0.140 0.0353 3.95 7.66e- 5  
#> 3 plosives 1.06 0.993 1.13 0.0599 0.0340 1.76 7.80e- 2  
#> 4 vowels 1.00 0.939 1.07 0.00356 0.0341 0.105 9.17e- 1

Finally, we can examine all of the Aligner x Class age trends.

class\_aligner\_trends <- emtrends(  
 m\_class\_int\_age\_3way,   
 ~ class | aligner,   
 var = "age\_scale",  
 bias.adjust = TRUE,   
 sigma = sigma2,  
 type = "response",  
 at = list(age\_scale = c(0, 1)),   
)   
  
class\_aligner\_trends %>%   
 broom::tidy(conf.int = TRUE) %>%   
 mutate(  
 or\_trend = exp(age\_scale.trend),  
 or\_lcl = exp(asymp.LCL),  
 or\_ucl = exp(asymp.UCL),  
 trend = age\_scale.trend,  
 se = std.error,  
 z = z.ratio,  
 p = p.value  
 ) %>%   
 select(aligner, class, starts\_with("or\_"), trend, se, z, p)  
#> # A tibble: 20 x 9  
#> aligner class or\_trend or\_lcl or\_ucl trend se z p  
#> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
#> 1 Kaldi fricativ~ 1.46 1.32 1.61 0.376 0.0494 7.62 2.46e-14  
#> 2 Kaldi others 1.23 1.11 1.36 0.207 0.0509 4.07 4.64e- 5  
#> 3 Kaldi plosives 1.25 1.14 1.37 0.224 0.0462 4.84 1.32e- 6  
#> 4 Kaldi vowels 1.13 1.03 1.24 0.123 0.0455 2.71 6.76e- 3  
#> 5 MFA (SAT) fricativ~ 1.30 1.17 1.45 0.264 0.0536 4.93 8.10e- 7  
#> 6 MFA (SAT) others 1.01 0.914 1.13 0.0146 0.0535 0.273 7.84e- 1  
#> 7 MFA (SAT) plosives 0.991 0.900 1.09 -0.00925 0.0489 -0.189 8.50e- 1  
#> 8 MFA (SAT) vowels 0.867 0.787 0.956 -0.142 0.0496 -2.87 4.17e- 3  
#> 9 MFA (No SAT) fricativ~ 1.37 1.25 1.51 0.317 0.0490 6.47 9.82e-11  
#> 10 MFA (No SAT) others 1.19 1.08 1.31 0.174 0.0503 3.46 5.44e- 4  
#> 11 MFA (No SAT) plosives 1.13 1.03 1.24 0.123 0.0459 2.67 7.49e- 3  
#> 12 MFA (No SAT) vowels 1.08 0.986 1.19 0.0784 0.0470 1.67 9.54e- 2  
#> 13 P2FA fricativ~ 1.29 1.18 1.42 0.258 0.0471 5.47 4.39e- 8  
#> 14 P2FA others 1.23 1.12 1.36 0.211 0.0500 4.21 2.50e- 5  
#> 15 P2FA plosives 1.09 1.00 1.19 0.0883 0.0447 1.98 4.82e- 2  
#> 16 P2FA vowels 1.15 1.06 1.26 0.141 0.0444 3.18 1.46e- 3  
#> 17 Prosodylab fricativ~ 1.08 0.983 1.18 0.0746 0.0468 1.59 1.11e- 1  
#> 18 Prosodylab others 1.10 0.996 1.21 0.0913 0.0489 1.87 6.16e- 2  
#> 19 Prosodylab plosives 0.882 0.808 0.962 -0.126 0.0445 -2.83 4.73e- 3  
#> 20 Prosodylab vowels 0.833 0.763 0.909 -0.183 0.0446 -4.11 4.04e- 5

## Computing environment

sessioninfo::session\_info()  
#> - Session info ---------------------------------------------------------------  
#> setting value   
#> version R version 4.0.2 (2020-06-22)  
#> os Windows 10 x64   
#> system x86\_64, mingw32   
#> ui RTerm   
#> language (EN)   
#> collate English\_United States.1252   
#> ctype English\_United States.1252   
#> tz America/Chicago   
#> date 2020-10-30   
#>   
#> - Packages -------------------------------------------------------------------  
#> ! package \* version date lib source   
#> assertthat 0.2.1 2019-03-21 [1] CRAN (R 4.0.2)  
#> backports 1.1.10 2020-09-15 [1] CRAN (R 4.0.2)  
#> base64url 1.4 2018-05-14 [1] CRAN (R 4.0.2)  
#> blob 1.2.1 2020-01-20 [1] CRAN (R 4.0.2)  
#> boot 1.3-25 2020-04-26 [1] CRAN (R 4.0.2)  
#> broom 0.7.2 2020-10-20 [1] CRAN (R 4.0.2)  
#> broom.mixed 0.2.6 2020-05-17 [1] CRAN (R 4.0.2)  
#> cellranger 1.1.0 2016-07-27 [1] CRAN (R 4.0.2)  
#> cli 2.1.0 2020-10-12 [1] CRAN (R 4.0.2)  
#> coda 0.19-4 2020-09-30 [1] CRAN (R 4.0.2)  
#> codetools 0.2-16 2018-12-24 [1] CRAN (R 4.0.0)  
#> colorspace 1.4-1 2019-03-18 [1] CRAN (R 4.0.2)  
#> crayon 1.3.4 2017-09-16 [1] CRAN (R 4.0.2)  
#> DBI 1.1.0 2019-12-15 [1] CRAN (R 4.0.2)  
#> dbplyr 1.4.4 2020-05-27 [1] CRAN (R 4.0.2)  
#> digest 0.6.27 2020-10-24 [1] CRAN (R 4.0.3)  
#> dplyr \* 1.0.2 2020-08-18 [1] CRAN (R 4.0.2)  
#> drake 7.12.6 2020-10-10 [1] CRAN (R 4.0.3)  
#> ellipsis 0.3.1 2020-05-15 [1] CRAN (R 4.0.2)  
#> emmeans \* 1.5.2-1 2020-10-25 [1] CRAN (R 4.0.3)  
#> estimability 1.3 2018-02-11 [1] CRAN (R 4.0.0)  
#> evaluate 0.14 2019-05-28 [1] CRAN (R 4.0.2)  
#> fansi 0.4.1 2020-01-08 [1] CRAN (R 4.0.2)  
#> filelock 1.0.2 2018-10-05 [1] CRAN (R 4.0.2)  
#> forcats \* 0.5.0 2020-03-01 [1] CRAN (R 4.0.2)  
#> fs 1.5.0 2020-07-31 [1] CRAN (R 4.0.2)  
#> generics 0.0.2 2018-11-29 [1] CRAN (R 4.0.2)  
#> ggplot2 \* 3.3.2 2020-06-19 [1] CRAN (R 4.0.2)  
#> glue 1.4.2 2020-08-27 [1] CRAN (R 4.0.2)  
#> gtable 0.3.0 2019-03-25 [1] CRAN (R 4.0.2)  
#> haven 2.3.1 2020-06-01 [1] CRAN (R 4.0.2)  
#> hms 0.5.3 2020-01-08 [1] CRAN (R 4.0.2)  
#> htmltools 0.5.0 2020-06-16 [1] CRAN (R 4.0.2)  
#> httr 1.4.2 2020-07-20 [1] CRAN (R 4.0.2)  
#> igraph 1.2.6 2020-10-06 [1] CRAN (R 4.0.2)  
#> jsonlite 1.7.1 2020-09-07 [1] CRAN (R 4.0.2)  
#> knitr 1.30 2020-09-22 [1] CRAN (R 4.0.2)  
#> lattice 0.20-41 2020-04-02 [1] CRAN (R 4.0.2)  
#> lifecycle 0.2.0 2020-03-06 [1] CRAN (R 4.0.2)  
#> lme4 \* 1.1-25 2020-10-23 [1] CRAN (R 4.0.3)  
#> lubridate 1.7.9 2020-06-08 [1] CRAN (R 4.0.2)  
#> magrittr 1.5 2014-11-22 [1] CRAN (R 4.0.2)  
#> MASS 7.3-53 2020-09-09 [1] CRAN (R 4.0.2)  
#> Matrix \* 1.2-18 2019-11-27 [1] CRAN (R 4.0.2)  
#> minqa 1.2.4 2014-10-09 [1] CRAN (R 4.0.2)  
#> modelr 0.1.8 2020-05-19 [1] CRAN (R 4.0.2)  
#> multcomp 1.4-14 2020-09-23 [1] CRAN (R 4.0.2)  
#> munsell 0.5.0 2018-06-12 [1] CRAN (R 4.0.2)  
#> mvtnorm 1.1-1 2020-06-09 [1] CRAN (R 4.0.0)  
#> nlme 3.1-150 2020-10-24 [1] CRAN (R 4.0.3)  
#> nloptr 1.2.2.2 2020-07-02 [1] CRAN (R 4.0.2)  
#> pillar 1.4.6 2020-07-10 [1] CRAN (R 4.0.2)  
#> pkgconfig 2.0.3 2019-09-22 [1] CRAN (R 4.0.2)  
#> plyr 1.8.6 2020-03-03 [1] CRAN (R 4.0.2)  
#> prettyunits 1.1.1 2020-01-24 [1] CRAN (R 4.0.2)  
#> progress 1.2.2 2019-05-16 [1] CRAN (R 4.0.2)  
#> purrr \* 0.3.4 2020-04-17 [1] CRAN (R 4.0.2)  
#> R6 2.5.0 2020-10-28 [1] CRAN (R 4.0.2)  
#> Rcpp 1.0.5 2020-07-06 [1] CRAN (R 4.0.2)  
#> readr \* 1.4.0 2020-10-05 [1] CRAN (R 4.0.2)  
#> readxl 1.3.1 2019-03-13 [1] CRAN (R 4.0.2)  
#> reprex 0.3.0 2019-05-16 [1] CRAN (R 4.0.2)  
#> reshape2 1.4.4 2020-04-09 [1] CRAN (R 4.0.2)  
#> rlang 0.4.8 2020-10-08 [1] CRAN (R 4.0.3)  
#> rmarkdown 2.5 2020-10-21 [1] CRAN (R 4.0.2)  
#> rstudioapi 0.11 2020-02-07 [1] CRAN (R 4.0.2)  
#> rvest 0.3.6 2020-07-25 [1] CRAN (R 4.0.2)  
#> sandwich 3.0-0 2020-10-02 [1] CRAN (R 4.0.2)  
#> scales 1.1.1 2020-05-11 [1] CRAN (R 4.0.2)  
#> sessioninfo 1.1.1 2018-11-05 [1] CRAN (R 4.0.2)  
#> statmod 1.4.35 2020-10-19 [1] CRAN (R 4.0.3)  
#> storr 1.2.4 2020-10-12 [1] CRAN (R 4.0.3)  
#> stringi 1.5.3 2020-09-09 [1] CRAN (R 4.0.2)  
#> stringr \* 1.4.0 2019-02-10 [1] CRAN (R 4.0.2)  
#> survival 3.2-7 2020-09-28 [1] CRAN (R 4.0.2)  
#> TH.data 1.0-10 2019-01-21 [1] CRAN (R 4.0.2)  
#> tibble \* 3.0.4 2020-10-12 [1] CRAN (R 4.0.3)  
#> tidyr \* 1.1.2 2020-08-27 [1] CRAN (R 4.0.2)  
#> tidyselect 1.1.0 2020-05-11 [1] CRAN (R 4.0.2)  
#> tidyverse \* 1.3.0 2019-11-21 [1] CRAN (R 4.0.2)  
#> D TMB 1.7.18 2020-07-27 [1] CRAN (R 4.0.2)  
#> txtq 0.2.3 2020-06-23 [1] CRAN (R 4.0.2)  
#> utf8 1.1.4 2018-05-24 [1] CRAN (R 4.0.2)  
#> vctrs 0.3.4 2020-08-29 [1] CRAN (R 4.0.2)  
#> withr 2.3.0 2020-09-22 [1] CRAN (R 4.0.2)  
#> xfun 0.18 2020-09-29 [1] CRAN (R 4.0.2)  
#> xml2 1.3.2 2020-04-23 [1] CRAN (R 4.0.2)  
#> xtable 1.8-4 2019-04-21 [1] CRAN (R 4.0.2)  
#> yaml 2.2.1 2020-02-01 [1] CRAN (R 4.0.0)  
#> zoo 1.8-8 2020-05-02 [1] CRAN (R 4.0.2)  
#>   
#> [1] C:/Users/Tristan/Documents/R/win-library/4.0  
#> [2] C:/Program Files/R/R-4.0.2/library  
#>   
#> D -- DLL MD5 mismatch, broken installation.

Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. Journal of the Royal Statistical Society Series B, 57, 289–300. <http://www.jstor.org/stable/2346101>.