Scale=MatchLowercase []Ligatures=TeX,Scale=1 [protrusion]basicmath
pdftitle=Assignment 3, pdfauthor=study group no 8, hidelinks, pdfcreator=LaTeX via pandoc same
HighlightingVerbatimcommandchars=
{}

Assignment 3

study group no 8

23/11/2022

part-i—simulating-the-data

Part I - simulating the data

Use the meta-analysis reported in Parola et al (2020), create a simulated dataset with 100 matched pairs of schizophrenia and controls, each participant producing 10 repeated measures (10 trials with their speech recorded). for each of these "recordings" (data points) produce 10 acoustic measures: 6 from the meta-analysis, 4 with just random noise. Do the same for a baseline dataset including only 10 noise variables. Tip: see the slides for the code.

[] simulate_data <- function(pop_effects, n = 100, n_trails = 10, individual_sd = 1, trail_sd = 0.5, error = 0.2, seed = 1){
set.seed(seed)
tibble(variable = map_chr(seq_along(pop_effects), ~ paste0(v_, .)), population_value = pop_effects) %>% mutate(id = seq(1, n, by = 1) %>% list) %>% unnest(id) %>% rowwise %>% mutate(condition = c(sz, hc) %>% list, true_effect = rnorm(1, population_value, individual_sd) / 2, true_effect = c(true_effect, - true_effect) %>% list, trail = seq(1, n_trails, by = 1) %>% list) %>% unnest(c(condition, true_effect)) %>% unnest(trail) %>% rowwise %>% mutate(measurment = rnorm(1, true_effect, trail_sd) %>% rnorm(1, ., error), across(c(variable, id, condition), as_factor)) %>% relocate(c(variable, population_value), .after = condition) }

[] m_a_values <- c(0.253, - 0.546, 0.739, - 1.26, - 0.155, - 0.75, 1.891, 0.046)
set.seed(1) informed_pop_effects <- c(sample(m_a_values, 6, replace = F), rep(0, 4))
skeptic_pop_effects <- rep(0, 10)

[] dfs_long <- map(list(informed_pop_effects, skeptic_pop_effects), simulate_data)
names(dfs_long) <- c(informed, skeptic)
head(dfs_long[[1]])

```
## # A tibble: 6 x 7
## # Rowwise:
##   id    condition variable population_value true_effect trail measurment
##   <fct> <fct>     <fct>               <dbl>       <dbl> <dbl>      <dbl>
## 1 1     sz        v_1                 0.253      -0.187     1      0.603
## 2 1     sz        v_1                 0.253      -0.187     2     -0.580
## 3 1     sz        v_1                 0.253      -0.187     3     -0.485
## 4 1     sz        v_1                 0.253      -0.187     4     -0.164
## 5 1     sz        v_1                 0.253      -0.187     5     -0.610
## 6 1     sz        v_1                 0.253      -0.187     6     -0.500
```

[] head(dfs_long[[2]])

```
## # A tibble: 6 x 7
## # Rowwise:
##   id    condition variable population_value true_effect trail measurment
##   <fct> <fct>     <fct>               <dbl>       <dbl> <dbl>      <dbl>
## 1 1     sz        v_1                     0      -0.313     1      0.477
```

```
## 2 1      sz        v_1                          0       -0.313      2      -0.706
## 3 1      sz        v_1                          0       -0.313      3      -0.611
## 4 1      sz        v_1                          0       -0.313      4      -0.290
## 5 1      sz        v_1                          0       -0.313      5      -0.737
## 6 1      sz        v_1                          0       -0.313      6      -0.627
```

[] #checking whether the simulation works fine
check <- map(list(informed_pop_effects, skeptic_pop_effects), ~ simulate_data(pop_effects = .x, n = 1000))
check[[1]] %>% group_by(condition, variable) %>% summarise(mean = true_effect %>% mean, sd = true_effect %>% sd) %>% ungroup %>% mutate(true_mean = c(informed_pop_effects, -informed_pop_effects))

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

```
## # A tibble: 20 x 5
##    condition variable     mean    sd true_mean
##    <fct>     <fct>       <dbl> <dbl>     <dbl>
##  1 sz        v_1       0.121   0.517     0.253
##  2 sz        v_2      -0.638   0.520    -1.26
##  3 sz        v_3       0.0307  0.515     0.046
##  4 sz        v_4      -0.265   0.519    -0.546
##  5 sz        v_5      -0.385   0.494    -0.75
##  6 sz        v_6       0.364   0.489     0.739
##  7 sz        v_7      -0.0103  0.519     0
##  8 sz        v_8      -0.00211 0.500     0
##  9 sz        v_9      -0.0107  0.498     0
## 10 sz        v_10      0.00426 0.487     0
## 11 hc        v_1      -0.121   0.517    -0.253
## 12 hc        v_2       0.638   0.520     1.26
## 13 hc        v_3      -0.0307  0.515    -0.046
## 14 hc        v_4       0.265   0.519     0.546
## 15 hc        v_5       0.385   0.494     0.75
## 16 hc        v_6      -0.364   0.489    -0.739
## 17 hc        v_7       0.0103  0.519     0
## 18 hc        v_8       0.00211 0.500     0
## 19 hc        v_9       0.0107  0.498     0
## 20 hc        v_10     -0.00426 0.487     0
```

[] #visualising the simulated data map(dfs_long, ~ .x %>% ggplot(aes(x = measurment, fill = condition)) + geom_density() + facet_wrap(vars(variable)) + theme_minimal())

```
## $informed

##
## $skeptic
```

[] dfs_wide <- map(dfs_long, ~ .x %>% pivot_wider(id_cols = c(id, trail, condition), names_from = variable, values_from = measurment)) head(dfs_wide[[1]])

```
## # A tibble: 6 x 13
##   id    trail condit~1    v_1    v_2    v_3     v_4    v_5   v_6     v_7     v_8
##   <fct> <dbl> <fct>     <dbl>  <dbl>  <dbl>   <dbl>  <dbl> <dbl>   <dbl>   <dbl>
## 1 1         1 sz        0.603 -1.57  -0.570   0.123  0.745 0.155 -0.156  -0.167
## 2 1         2 sz       -0.580 -1.54   0.414  -0.647  0.646 0.951  0.330   0.478
```

```
## 3 1         3 sz       -0.485 -0.629  0.818  0.0598 0.0165 0.996  0.297   0.104
## 4 1         4 sz       -0.164 -1.44  -0.212  0.540  0.864  0.314  0.222   0.0444
## 5 1         5 sz       -0.610 -0.813 -0.516  0.425  0.283  0.237  0.0142 -1.09
## 6 1         6 sz       -0.500 -1.32   0.642 -0.307  0.335  0.582  0.905  -0.0668
## # ... with 2 more variables: v_9 <dbl>, v_10 <dbl>, and abbreviated variable
## #   name 1: condition
```

[] head(dfs_wide[[2]])

```
## # A tibble: 6 x 13
##   id    trail condition    v_1      v_2    v_3     v_4   v_5     v_6     v_7
##   <fct> <dbl> <fct>      <dbl>    <dbl>  <dbl>   <dbl> <dbl>   <dbl>   <dbl>
## 1 1         1 sz         0.477 -0.943   -0.593  0.396  1.12  -0.214  -0.156
## 2 1         2 sz        -0.706 -0.913    0.391 -0.374  1.02   0.582   0.330
## 3 1         3 sz        -0.611  0.00110  0.795  0.333  0.392  0.626   0.297
## 4 1         4 sz        -0.290 -0.806   -0.235  0.813  1.24  -0.0557  0.222
## 5 1         5 sz        -0.737 -0.183   -0.539  0.698  0.658 -0.132   0.0142
## 6 1         6 sz        -0.627 -0.691    0.619 -0.0336 0.710  0.212   0.905
## # ... with 3 more variables: v_8 <dbl>, v_9 <dbl>, v_10 <dbl>
```

part-ii—machine-learning-pipeline-on-simulated-data

Part II - machine learning pipeline on simulated data

On the two simulated datasets (separately) build a machine learning pipeline: i) create a data budget (e.g. balanced training and test sets); ii) pre-process the data (e.g. scaling the features); iii) fit and assess a classification algorithm on the training data (e.g. Bayesian multilevel logistic regression); iv) assess performance on the test set; v) discuss whether performance is as expected and feature importance is as expected.

Bonus question: replace the bayesian multilevel regression with a different algorithm, e.g. SVM or random forest (but really, anything you'd like to try). ## Budgeting the data:

```
[] # We know that using map() for a list of 2 elements might probably be considered an overkill, but we thought it would make the code easier to read and easier modify so we can reuse it in part 3. Also, we tried to see the assignment as a learning experience, and learning to use functional programming on lists of data frames seems like something useful.
splits <- map(dfs_wide, ~ .x %>% initial_split( prop = 4 / 5))
dfs_training <- map(splits, ~ .x %>% training) dfs_testing <- map(splits, ~ .x %>% testing)
rm(splits)
```

preprocessing-the-data

Preprocessing the data

```
[] recipes <- map(dfs_training, ~ recipe(condition ~ 1 + ., data = .x) %>% update_role(id, trail, new_role = id) %>% step_normalize( all_numeric()) )
```

fitting-training-the-models

Fitting (training) the models

creating-the-models

Creating the models

```
[] prior_b <- normal( location = 0, scale = 0.5, autoscale = T) prior_intercept <- normal( 0, 1, autoscale = T) #setting the autoscale to TRUE makes the model adjust the scale of the prior to the standard deviation of each parameter by multiplying the scale argument by 1 /sd(given variable)
prior_model <- logistic_reg() %>% set_engine( stan, prior = prior_b, prior_intercept = prior_intercept, prior_PD = T, cores = 3)
```

model <- logistic_reg() %>% set_engine(stan, prior = prior_b, prior_intercept = prior_intercept, cores = 3)

workflows

Workflows

[] wflows <- map(recipes, ~ workflow() %>% add_model(model) %>% add_recipe(.x))

prior_wflows <- map(recipes, ~ workflow() %>% add_model(prior_model) %>% add_recipe(.x))

model-fitting

Model fitting

[] prior_models <- list(prior_wflows[[1]] %>% fit(dfs_training[[1]]), prior_wflows[[2]] %>% fit(dfs_training[[2]])) %>% map(extract_fit_engine)

fitted <- list(wflows[[1]] %>% fit(dfs_training[[1]]), wflows[[2]] %>% fit(dfs_training[[2]])) names(fitted) <- c(Informed, Sceptic)

fitted_models <- fitted %>% map(extract_fit_engine)

rm(prior_wflows)

convergance-checks

Convergance checks

[] convergance_plots <- map2(fitted_models, names(fitted_models), **function**(.x, .y){ list(plot(.x, trace), plot(.x, neff), plot(.x, rhat)) %>% map(**function**(.x){.x + ggtitle(.y)}) })

convergance_plots %>% print

```
## $Informed
## $Informed[[1]]

##
## $Informed[[2]]

##
## $Informed[[3]]

##
##
## $Sceptic
## $Sceptic[[1]]

##
## $Sceptic[[2]]

##
## $Sceptic[[3]]
```

[] rm(convergance_plots)

checking-the-priors

Checking the priors

visualising-the-prior-distributions

prior-posterior-update-checks

[] pp_update_plot <- **function**(prior_model, posterior_model){ df_draws <- bind_rows(bind_rows(prior_model %>% gather_draws((Intercept)), prior_model %>% gather_draws(v_.* , regex = T)) %>% mutate(type = prior),

bind_rows(posterior_model %>% gather_draws((Intercept)), posterior_model %>% gather_draws(v_.* , regex = T)) %>% mutate(type = posterior))

df_draws <- df_draws %>% group_by(.variable) %>% mutate(upp_lim = if_else((max(.value) + min(.value)) > 0, max(.value), - min(.value)), low_lim = - upp_lim) %>% ungroup df_draws %>% ggplot(aes(x = .value, fill = type)) + geom_density(alpha = 0.8) + labs(fill = element_blank()) + xlim(df_draws $low_lim[[1]], df_draws$upp_lim[[1]]) + facet_grid(vars(df_draws $.variable)) + theme\_minimal() + theme(axis.ticks.y = element\_blank(), axis.text.y = element\_blank())$ }

[] pp_update_plot(prior_models[[1]], fitted_models[[1]]) + ggtitle(Informed)

[] pp_update_plot(prior_models[[2]], fitted_models[[2]]) + ggtitle(Sceptic)

Accessing model performance

cross-validation

Cross-validation

[] dfs_folded <- map(dfs_training, ˜ vfold_cv(.x, v = 8))

cv_data <- map2(wflows, dfs_folded, ˜ fit_resamples(.x, .y, metrics = metric_set(f_meas, roc_auc)))

cv_results <- map(cv_data, ˜ collect_metrics(.x) %>% mutate(upper = mean + std_err, lower = mean - std_err))

cv_results <- bind_rows(cv_results[[1]] %>% mutate(model = Informed), cv_results[[2]] %>% mutate(model = Sceptic))

cv_results <- cv_results %>% rename_with(.cols = everything(), ˜ str_remove(.x, stringr :: fixed("."))) %>% mutate(metric = if_else(metric == f_meas, f1, metric))

[] cv_results %>% ggplot(aes(x = mean, y = model, xmax = upper, xmin = lower, colour = model)) + geom_pointrange() + facet_wrap(vars(metric)) + geom_vline(xintercept = 0.5, colour = darkred, linetype = dashed, alpha = 0.7) + theme_minimal() + coord_flip()

test-data

Test data

[] test_preds <- map2(fitted, dfs_training, ˜ augment(.x, .y))

map2(test_preds, names(test_preds), ˜ .x %>% roc_curve(truth = condition, .pred_sz) %>% autoplot + ggtitle(.y))

```
## $Informed


##
## $Sceptic
```

Conclusions (is performance and feature importance as expected)

[] *#without uncertanity*

 # come up with a better name for this one test_results_mean_only <- map2_df(test_preds, names(test_preds), ˜ bind_rows(.x %>% roc_auc(truth = condition, .pred_sz), .x %>% f_meas(truth = condition, .pred_class, beta = 1) %>% mutate(.metric = f1)) %>% mutate(Model = .y))

test_results_mean_only %>% ggplot(aes(x = Model, y = .estimate, colour = Model)) + geom_point() + facet_wrap(vars(.metric)) + geom_hline(yintercept = 0.5, colour = darkred, linetype = dashed, alpha = 0.7) + theme_minimal()

[] *#with the uncertanity*

test_results <- tibble(draw = NULL, f1 = NULL, model = NULL)

for (i **in** seq_along(fitted_models)){

m <- fitted_models[[i]] name <- names(fitted_models)[[i]]

draws_matrix <- posterior_epred(m)

roc_aucs <- map_dbl(draws_matrix %>% split(row(draws_matrix)), ˜ roc_auc_vec(truth = dfs_training[[1]] $condition, estimate = .x$))

roc_aucs <- tibble(value = roc_aucs, metric = roc_auc, draw = seq_along(nrow))

preds_class <- map(draws_matrix %>% split(row(draws_matrix)), ˜ if_else(.x < 0.5, sz, hc) %>% as_factor %>% relevel(sz))

fs <- map_dbl(preds_class, ˜ f_meas_vec(truth = dfs_training[[1]] $condition, estimate = .x, beta = 1$))

fs <- tibble(value = fs, metric = f1, draw = seq_along(nrow)) test_results <- bind_rows(test_results, bind_rows(fs, roc_aucs) %>% mutate(model = name)) } rm(i, m, name, draws_matrix, roc_aucs, preds_class, fs)

test_results <- test_results %>% mutate(value = if_else(metric == roc_auc, 1 - value, value))

test_results_summary <- test_results %>% group_by(model, metric) %>% summarise(mean = mean(value), std_err = sd(value), *#because were dealing the the estimates of the population parameters, the sd already is the standard error (or at least so my limited understanding goes)* lower = mean - 1.96 *std_err, upper = mean + 1.96 *std_err)

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

[] test_results %>% ggplot(aes(x = model, y = value, colour = model)) + geom_point(alpha = 0.7) + geom_hline(yintercept = 0.5, color = darkred, linetype = dashed, alpha = 0.7) + theme_minimal() + facet_wrap(vars(metric))

[] *# Just realised this might actually not work*

 # 1. mean accuracy of all draws is something very different from the accuracy of the mean linear predictor

#2. Second problem is that the confidence intervals in cross-validation and test might not show the same thing - the cross validation one shows sd of the mean accuracy for each fold divided by sqrt(number of folds) while the test shows the standard diviation of the draws themselves (you checked that and the se calculated like that and the one the functions spits out are exactly the same)

 # What to do about it? # - plot only the accuracies only for the mean + ci of final model estimates? # - just back out of the confidence intervals and do all the dots for cross-validation as well # - you then have to code the cross-validation by hand

performance_data <- bind_rows(test_results_summary %>% mutate(type = test), cv_results %>% mutate(type = cross-validation)) %>% ungroup

performance_data <- performance_data %>% mutate(across(where(is.character), as_factor))

glimpse(performance_data)

```
## Rows: 8
## Columns: 10
## $ model     <fct> Informed, Informed, Sceptic, Sceptic, Informed, Informed, Sc~
## $ metric    <fct> f1, roc_auc, f1, roc_auc, f1, roc_auc, f1, roc_auc
## $ mean      <dbl> 0.8949697, 0.9617240, 0.4668464, 0.4991456, 0.8973209, 0.960~
## $ std_err   <dbl> 0.0027368566, 0.0004366533, 0.0271010330, 0.0059087332, 0.00~
## $ lower     <dbl> 0.8896054, 0.9608682, 0.4137284, 0.4875645, 0.8917013, 0.957~
## $ upper     <dbl> 0.9003339, 0.9625798, 0.5199644, 0.5107267, 0.9029405, 0.963~
## $ type      <fct> test, test, test, test, cross-validation, cross-validation, ~
## $ estimator <fct> NA, NA, NA, NA, binary, binary, binary, binary
## $ n         <int> NA, NA, NA, NA, 8, 8, 8, 8
## $ config    <fct> NA, NA, NA, NA, Preprocessor1_Model1, Preprocessor1_Model1, ~
```

[] performance_data %>% ggplot(aes(x = mean, y = model, xmin = lower, xmax = upper, colour = type)) + geom_pointrange(position = position_dodge(width = 0.5)) + geom_vline(aes(xintercept = 0.5), color = darkred, linetype = dashed, alpha = 0.7) + labs(y = F1) + theme_minimal() + coord_flip() + facet_wrap(vars(metric))

Feature importance

[] vip_simulated <- **function**(model, truth){ vim_df <- model %>% gather_draws(v_.* , regex = T) vim_df <- map2_df(vim_df %>% group_split(.variable), truth, ~ .x %>% mutate(truth = .y))

vim_df %>% ggplot(aes(x = .value)) + geom_density() + geom_vline(aes(xintercept = truth[[1]]), color = darkred, linetype = dashed, alpha = 0.8) + facet_wrap(vars(.variable), nrow = , scales = free_x) + theme_minimal() }

vip_simulated(fitted_models[[1]], informed_pop_effects) + ggtitle(Informed)

```
## Warning: ... is ignored in group_split(<grouped_df>), please use group_by(..., .add =
## TRUE) %>% group_split()
```

[] vip_simulated(fitted_models[[2]], skeptic_pop_effects) + ggtitle(Skeptic)

```
## Warning: ... is ignored in group_split(<grouped_df>), please use group_by(..., .add =
## TRUE) %>% group_split()
```

[] vips <- map(c(1, 2), ˜ explain_tidymodels(fitted[[.x]] %>% extract_fit_parsnip, data = dfs_training[[.x]], y = dfs_training[[.x]] $condition %>% as.numeric - 1, label = names(fitted)[[.x]]$))

```
## Preparation of a new explainer is initiated
##   -> model label       :  Informed
##   -> data              :  1600  rows  13  cols
##   -> data              :  tibble converted into a data.frame
##   -> target variable   :  1600  values
##   -> predict function  :  yhat.model_fit  will be used (  default  )
##   -> predicted values  :  No value for predict function target column. (  default  )
##   -> model_info        :  package parsnip , ver. 1.0.3 , task classification (  default  )
##   -> predicted values  :  numerical, min =  6.726653e-05 , mean =  0.5140391 , max =  0.9999017
##   -> residual function :  residual_function
##   -> residuals         :  numerical, min =  0 , mean =  0 , max =  0
##   A new explainer has been created!
## Preparation of a new explainer is initiated
##   -> model label       :  Sceptic
##   -> data              :  1600  rows  13  cols
##   -> data              :  tibble converted into a data.frame
##   -> target variable   :  1600  values
##   -> predict function  :  yhat.model_fit  will be used (  default  )
##   -> predicted values  :  No value for predict function target column. (  default  )
##   -> model_info        :  package parsnip , ver. 1.0.3 , task classification (  default  )
##   -> predicted values  :  numerical, min =  0.2643241 , mean =  0.5121726 , max =  0.7335213
##   -> residual function :  residual_function
##   -> residuals         :  numerical, min =  0 , mean =  0 , max =  0
##   A new explainer has been created!
```

[] map(vips, ˜ .x %>% model_parts %>% plot(show_boxplots = F) + labs(title = Feature importance, subtitle = NULL))

```
## [[1]]

##
## [[2]]
```

[] map(vips, ˜ .x %>% model_profile(type = partial, variables = paste0(v_, seq(10))) %>% plot() + labs(title = Partial dependence profile))

```
## [[1]]

##
## [[2]]
```

part-iii

Part III

Download the empirical dataset from brightspace and apply your ML pipeline to the new data, adjusting where needed. Warning: in the simulated dataset we only had 10 features, now you have many more! Such is the life of the ML practitioner. Consider the impact a higher number of features will have on your ML inference, and decide whether you need to cut down the number of features before running the pipeline (or alternatively expand the pipeline to add feature selection).

```
[] rm( list = ls()) # removing all objects from the global environment
[] data_raw <- read_csv( real_data.csv)
```

```
## Rows: 1889 Columns: 398
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (5): NewID, Diagnosis, Language, Gender, Trial
## dbl (393): PatID, Corpus, Duration_Praat, F0_Mean_Praat, F0_SD_Praat, Intens...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
[] glimpse(data_raw)
```

```
## Rows: 1,889
## Columns: 398
## $ PatID                  <dbl> 101, 101, 101, 101, 101, 101, 101, 101, 1~
## $ NewID                  <chr> "101CT1", "101CT1", "101CT1", "101CT1", "~
## $ Diagnosis              <chr> "CT", "CT", "CT", "CT", "CT", "CT", "CT",~
## $ Language               <chr> "D", "D", "D", "D", "D", "D", "D", "D", "~
## $ Gender                 <chr> "M", "M", "M", "M", "M", "M", "M", "M", "~
## $ Trial                  <chr> "T7", "T8", "T4", "T2", "T3", "T5", "T9",~
## $ Corpus                 <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ Duration_Praat         <dbl> 5.62, 2.82, 9.49, 8.92, 6.00, 12.62, 10.4~
## $ F0_Mean_Praat          <dbl> 157.4865, 115.4691, 125.3085, 133.1547, 1~
## $ F0_SD_Praat            <dbl> 37.226724, 5.037427, 9.099214, 19.466738,~
## $ Intensity_Mean_Praat   <dbl> 70.16840, 67.47500, 70.23711, 70.42194, 6~
## $ Intensity_SD_Praat     <dbl> 6.114989, 5.396695, 6.733844, 6.293483, 6~
## $ PauseDuration_Praat    <dbl> 3.31, 2.00, 5.03, 5.93, 3.57, 9.22, 5.53,~
## $ TurnDuration_Praat     <dbl> 2.31, 0.82, 4.46, 2.99, 2.43, 3.40, 4.92,~
## $ TurnNumber_Praat       <dbl> 12, 5, 19, 10, 20, 23, 22, 7, 6, 9, 38, 2~
## $ PauseNumber_Praat      <dbl> 12, 6, 20, 10, 21, 24, 23, 8, 6, 10, 38, ~
## $ PercentSpoke_Praat     <dbl> 0.4110320, 0.2907801, 0.4699684, 0.335201~
## $ PercentSilence_Praat   <dbl> 0.5889680, 0.7092199, 0.5300316, 0.664798~
## $ NHR_mean               <dbl> 1.0762909, 1.7420739, 0.8930114, 1.405997~
## $ NHR_std                <dbl> 1.0974521, 1.5076043, 1.1247343, 1.365611~
## $ Duration_Cova          <dbl> 5.70, 2.90, 9.57, 9.00, 6.08, 12.70, 10.5~
## $ PauseDuration_Cova     <dbl> 3.62, 2.10, 4.75, 5.99, 3.50, 9.26, 5.76,~
## $ TurnDuration_Cova      <dbl> 2.08, 0.80, 4.82, 3.01, 2.58, 3.44, 4.77,~
## $ TurnNumber_Cova        <dbl> 19, 9, 27, 28, 24, 40, 36, 15, 10, 15, 48~
## $ PauseNumber_Cova       <dbl> 19, 9, 27, 28, 24, 40, 36, 15, 10, 15, 48~
## $ PercentSpoke_Cova      <dbl> 0.3649123, 0.2758621, 0.5036573, 0.334444~
## $ PercentSilence_Cova    <dbl> 0.6350877, 0.7241379, 0.4963427, 0.665555~
## $ Pitch_Mean             <dbl> 5.004515, 4.823150, 4.854354, 4.878221, 4~
## $ Pitch_Median           <dbl> 4.969807, 4.762174, 4.832306, 4.844187, 4~
## $ Pitch_SD               <dbl> 0.20531669, 0.17566305, 0.11439739, 0.136~
## $ Pitch_IQR              <dbl> 0.27026462, 0.05577301, 0.10887925, 0.115~
```

```
## $ Pitch_MAD                  <dbl> 0.20509209, 0.03202284, 0.07263167, 0.071~
## $ F0_Mean                    <dbl> 152.4279, 126.5938, 129.2510, 132.7043, 1~
## $ F0_Median                  <dbl> 144.00, 117.00, 125.50, 127.00, 123.50, 1~
## $ F0_SD                      <dbl> 34.171980, 28.166559, 18.064082, 20.47695~
## $ F0_IQR                     <dbl> 38.750, 6.625, 13.875, 15.000, 6.875, 16.~
## $ F0_MAD                     <dbl> 28.91070, 3.70650, 8.89560, 8.89560, 5.18~
## $ F1_Mean                    <dbl> 474.3766, 590.4291, 479.5878, 539.8825, 5~
## $ F1_Median                  <dbl> 477.8326, 596.2321, 489.8214, 504.4456, 4~
## $ F1_SD                      <dbl> 123.33146, 118.70444, 138.64037, 173.5359~
## $ F1_IQR                     <dbl> 156.48211, 111.39806, 150.95069, 154.2735~
## $ F1_MAD                     <dbl> 119.55121, 87.83947, 109.97878, 115.86091~
## $ F2_Mean                    <dbl> 1379.245, 1604.363, 1642.664, 1591.472, 1~
## $ F2_Median                  <dbl> 1320.325, 1298.711, 1613.912, 1532.705, 1~
## $ F2_SD                      <dbl> 359.8170, 537.6798, 442.4514, 441.9225, 4~
## $ F2_IQR                     <dbl> 338.8416, 814.1942, 509.2859, 618.9723, 6~
## $ F2_MAD                     <dbl> 251.7420, 284.0742, 378.3801, 450.8429, 4~
## $ F3_Mean                    <dbl> 2588.769, 2725.858, 2708.497, 2575.991, 2~
## $ F3_Median                  <dbl> 2668.810, 2779.451, 2682.219, 2563.081, 2~
## $ F3_SD                      <dbl> 408.1128, 398.5884, 348.2637, 348.0087, 3~
## $ F3_IQR                     <dbl> 289.6221, 418.5070, 438.6633, 559.2186, 4~
## $ F3_MAD                     <dbl> 171.3690, 305.8763, 341.0985, 394.4439, 3~
## $ F4_Mean                    <dbl> 3378.862, 3481.532, 3584.281, 3487.865, 3~
## $ F4_Median                  <dbl> 3451.576, 3542.469, 3645.670, 3552.070, 3~
## $ F4_SD                      <dbl> 410.0392, 494.4055, 326.6898, 383.0070, 3~
## $ F4_IQR                     <dbl> 590.2965, 859.3307, 412.4021, 521.6006, 3~
## $ F4_MAD                     <dbl> 363.4770, 592.0502, 266.3483, 344.6552, 2~
## $ F5_Mean                    <dbl> 4201.605, 4253.272, 4542.009, 4517.439, 4~
## $ F5_Median                  <dbl> 4340.185, 4365.997, 4603.801, 4651.156, 4~
## $ F5_SD                      <dbl> 456.6505, 530.5366, 325.3482, 374.0563, 2~
## $ F5_IQR                     <dbl> 745.2826, 934.3287, 428.5955, 435.8510, 2~
## $ F5_MAD                     <dbl> 580.2267, 546.9331, 312.7858, 242.7380, 1~
## $ NAQ_Mean                   <dbl> 0.06180290, 0.04635425, 0.07702477, 0.043~
## $ NAQ_Median                 <dbl> 0.05702131, 0.03949889, 0.07097532, 0.039~
## $ NAQ_SD                     <dbl> 0.03384222, 0.03439789, 0.04596121, 0.030~
## $ NAQ_IQR                    <dbl> 0.04679950, 0.04546405, 0.05426564, 0.034~
## $ NAQ_MAD                    <dbl> 0.03489768, 0.03222384, 0.03949983, 0.025~
## $ QOQ_Mean                   <dbl> 0.2284831, 0.1746557, 0.2585948, 0.161105~
## $ QOQ_Median                 <dbl> 0.2105338, 0.1607614, 0.2298884, 0.153294~
## $ QOQ_SD                     <dbl> 0.11634157, 0.12148572, 0.14146882, 0.088~
## $ QOQ_IQR                    <dbl> 0.13750215, 0.12728650, 0.13913989, 0.107~
## $ QOQ_MAD                    <dbl> 0.10112211, 0.09477261, 0.10050534, 0.077~
## $ H1H2_Mean                  <dbl> -2.690729, -3.462465, -6.724419, -5.75338~
## $ H1H2_Median                <dbl> -3.9527000, -2.8253954, -7.9101788, -6.97~
## $ H1H2_SD                    <dbl> 9.903949, 9.542624, 9.448744, 9.085656, 8~
## $ H1H2_IQR                   <dbl> 14.195829, 10.139859, 12.107344, 11.26088~
## $ H1H2_MAD                   <dbl> 10.598792, 8.434327, 9.022037, 8.413112, ~
## $ PSP_Mean                   <dbl> 0.4155799, 0.5096232, 0.5365308, 0.677747~
## $ PSP_Median                 <dbl> 0.3856743, 0.4434876, 0.5113159, 0.654711~
## $ PSP_SD                     <dbl> 0.2332902, 0.3638096, 0.2725629, 0.367708~
## $ PSP_IQR                    <dbl> 0.3265878, 0.6215354, 0.3690651, 0.465478~
## $ PSP_MAD                    <dbl> 0.22695909, 0.40013808, 0.27969538, 0.346~
## $ HRF_Mean                   <dbl> 25.63492, 26.19126, 30.71390, 31.48877, 3~
## $ HRF_Median                 <dbl> 25.85260, 22.61750, 30.39679, 30.27583, 3~
## $ HRF_SD                     <dbl> 7.417731, 21.510680, 12.444059, 16.179379~
```

```
## $ HRF_IQR                  <dbl> 8.994097, 24.682586, 9.573233, 9.185110, ~
## $ HRF_MAD                  <dbl> 6.761540, 18.082763, 7.055838, 7.070440, ~
## $ MDQ_Mean                 <dbl> 0.1102170, 0.1334687, 0.1154243, 0.128729~
## $ MDQ_Median               <dbl> 0.1109119, 0.1375647, 0.1158310, 0.128661~
## $ MDQ_SD                   <dbl> 0.03080837, 0.02964374, 0.03133302, 0.023~
## $ MDQ_IQR                  <dbl> 0.04498629, 0.05090973, 0.04364999, 0.031~
## $ MDQ_MAD                  <dbl> 0.03284165, 0.03223560, 0.03258088, 0.023~
## $ peakSlope_Mean           <dbl> -0.3314857, -0.2978618, -0.3399992, -0.34~
## $ peakSlope_Median         <dbl> -0.3307617, -0.2949662, -0.3406523, -0.34~
## $ peakSlope_SD             <dbl> 0.08814333, 0.08688939, 0.09960372, 0.085~
## $ peakSlope_IQR            <dbl> 0.1004853, 0.1174296, 0.1268701, 0.112550~
## $ peakSlope_MAD            <dbl> 0.07490536, 0.09788425, 0.09669734, 0.080~
## $ Rd_Mean                  <dbl> 1.423645, 1.355995, 1.387245, 1.609067, 1~
## $ Rd_Median                <dbl> 1.414300, 1.324520, 1.341969, 1.596359, 1~
## $ Rd_SD                    <dbl> 0.4628926, 0.4884298, 0.4772106, 0.515868~
## $ Rd_IQR                   <dbl> 0.6859440, 0.7441651, 0.7387420, 0.783192~
## $ Rd_MAD                   <dbl> 0.5103568, 0.5689866, 0.5373712, 0.589471~
## $ Rd_conf_Mean             <dbl> 0.5205419, 0.5037996, 0.5403747, 0.443519~
## $ Rd_conf_Median           <dbl> 0.5093454, 0.5153203, 0.5337394, 0.438403~
## $ Rd_conf_SD               <dbl> 0.11268859, 0.07871862, 0.10851630, 0.081~
## $ Rd_conf_IQR              <dbl> 0.13331901, 0.11781373, 0.15594084, 0.109~
## $ Rd_conf_MAD              <dbl> 0.09982159, 0.08450161, 0.11357995, 0.076~
## $ VAD_Mean                 <dbl> 0.09273929, 0.05074510, 0.09563633, 0.086~
## $ MCEP0_Mean               <dbl> -7.800048, -8.576307, -7.848345, -7.79173~
## $ MCEP0_Median             <dbl> -7.466394, -8.378789, -7.630364, -7.39657~
## $ MCEP0_SD                 <dbl> 1.4010125, 0.9047833, 1.4079963, 1.246560~
## $ MCEP0_IQR                <dbl> 2.519754, 1.219289, 1.766956, 1.564649, 1~
## $ MCEP0_MAD                <dbl> 1.6556223, 0.5363446, 1.2693449, 0.984502~
## $ MCEP1_Mean               <dbl> 2.780916, 3.021607, 2.762131, 3.016723, 2~
## $ MCEP1_Median             <dbl> 3.070201, 3.208075, 2.838830, 3.137675, 2~
## $ MCEP1_SD                 <dbl> 1.0526957, 0.8032844, 0.8947056, 0.788233~
## $ MCEP1_IQR                <dbl> 0.6737240, 0.5889144, 1.0237569, 0.849630~
## $ MCEP1_MAD                <dbl> 0.4734662, 0.4230775, 0.7667883, 0.630019~
## $ MCEP2_Mean               <dbl> -0.31957158, -0.90633955, -0.45911659, -0~
## $ MCEP2_Median             <dbl> -0.386224463, -0.995711955, -0.568981375,~
## $ MCEP2_SD                 <dbl> 0.8361694, 0.6931855, 0.8851594, 0.898476~
## $ MCEP2_IQR                <dbl> 1.4352173, 1.1267430, 1.4676696, 1.361394~
## $ MCEP2_MAD                <dbl> 1.0446619, 0.7028617, 1.0742934, 0.933227~
## $ MCEP3_Mean               <dbl> 0.52587771, -0.05046947, 0.58634110, 0.37~
## $ MCEP3_Median             <dbl> 0.4320123, -0.1913579, 0.5971787, 0.38349~
## $ MCEP3_SD                 <dbl> 0.4859120, 0.5651418, 0.4582939, 0.393721~
## $ MCEP3_IQR                <dbl> 0.6738033, 0.3542476, 0.6360194, 0.576670~
## $ MCEP3_MAD                <dbl> 0.5026710, 0.2633475, 0.4768839, 0.438343~
## $ MCEP4_Mean               <dbl> -0.7003233, -0.9826797, -0.5998804, -0.52~
## $ MCEP4_Median             <dbl> -0.6777626, -1.1284020, -0.6356815, -0.52~
## $ MCEP4_SD                 <dbl> 0.4936995, 0.3922649, 0.4408092, 0.392671~
## $ MCEP4_IQR                <dbl> 0.6991296, 0.6156483, 0.5549506, 0.505118~
## $ MCEP4_MAD                <dbl> 0.5242759, 0.4336825, 0.4063606, 0.367032~
## $ MCEP5_Mean               <dbl> -2.265027e-01, -3.760730e-01, -4.205528e-~
## $ MCEP5_Median             <dbl> -0.189869096, -0.373225135, -0.413290762,~
## $ MCEP5_SD                 <dbl> 0.4288621, 0.3344166, 0.3508698, 0.289449~
## $ MCEP5_IQR                <dbl> 0.5620242, 0.4506707, 0.4443000, 0.304058~
## $ MCEP5_MAD                <dbl> 0.4053460, 0.2573561, 0.3389200, 0.227060~
## $ MCEP6_Mean               <dbl> 0.07553752, -0.05573101, -0.14504608, -0.~
```

```
## $ MCEP6_Median                <dbl> 0.04392212, -0.11108834, -0.14859093, -0.~
## $ MCEP6_SD                    <dbl> 0.2976298, 0.3234591, 0.3279043, 0.364046~
## $ MCEP6_IQR                   <dbl> 0.3407936, 0.3525973, 0.4895238, 0.553154~
## $ MCEP6_MAD                   <dbl> 0.2520758, 0.2462648, 0.3665046, 0.414260~
## $ MCEP7_Mean                  <dbl> -0.4457810785, -0.0017213469, -0.40583462~
## $ MCEP7_Median                <dbl> -0.4426901477, 0.0017575745, -0.390821270~
## $ MCEP7_SD                    <dbl> 0.2969627, 0.2423995, 0.3227063, 0.236961~
## $ MCEP7_IQR                   <dbl> 0.4169535, 0.3339511, 0.4421994, 0.300219~
## $ MCEP7_MAD                   <dbl> 0.2972967, 0.2491468, 0.3334111, 0.224476~
## $ MCEP8_Mean                  <dbl> -0.039303411, -0.226372038, 0.052734711, ~
## $ MCEP8_Median                <dbl> -0.05549179, -0.19439660, 0.03797236, -0.~
## $ MCEP8_SD                    <dbl> 0.3431017, 0.2352191, 0.2904116, 0.282241~
## $ MCEP8_IQR                   <dbl> 0.4515983, 0.2854237, 0.3586195, 0.340595~
## $ MCEP8_MAD                   <dbl> 0.3605354, 0.2160354, 0.2668115, 0.248315~
## $ MCEP9_Mean                  <dbl> 1.539128e-01, -1.947572e-02, -7.018876e-0~
## $ MCEP9_Median                <dbl> 0.1083735455, -0.0332572012, -0.087571055~
## $ MCEP9_SD                    <dbl> 0.2527215, 0.1818983, 0.2321234, 0.195758~
## $ MCEP9_IQR                   <dbl> 0.4277931, 0.2881567, 0.3079276, 0.232201~
## $ MCEP9_MAD                   <dbl> 0.2930839, 0.2027145, 0.2248219, 0.173099~
## $ MCEP10_Mean                 <dbl> 0.015567172, 0.190576580, 0.052467563, 0.~
## $ MCEP10_Median               <dbl> 0.035034816, 0.164430460, 0.056071360, 0.~
## $ MCEP10_SD                   <dbl> 0.2055490, 0.2352665, 0.1647975, 0.160535~
## $ MCEP10_IQR                  <dbl> 0.2671504, 0.2831755, 0.1984662, 0.225709~
## $ MCEP10_MAD                  <dbl> 0.2044624, 0.1965240, 0.1455495, 0.166113~
## $ MCEP11_Mean                 <dbl> -0.05285422, -0.22262581, -0.07242971, -0~
## $ MCEP11_Median               <dbl> -0.04361598, -0.18393433, -0.06076622, -0~
## $ MCEP11_SD                   <dbl> 0.3420598, 0.2062883, 0.1977763, 0.194204~
## $ MCEP11_IQR                  <dbl> 0.3288330, 0.1964233, 0.2524433, 0.248262~
## $ MCEP11_MAD                  <dbl> 0.2505081, 0.1767264, 0.1898152, 0.186048~
## $ MCEP12_Mean                 <dbl> -8.009303e-02, 7.094762e-02, -4.301525e-0~
## $ MCEP12_Median               <dbl> -0.0738974724, 0.0290061666, -0.041927605~
## $ MCEP12_SD                   <dbl> 0.2199786, 0.2713073, 0.1644384, 0.155547~
## $ MCEP12_IQR                  <dbl> 0.3119040, 0.2979355, 0.1963599, 0.196165~
## $ MCEP12_MAD                  <dbl> 0.2229825, 0.2204087, 0.1466897, 0.144399~
## $ MCEP13_Mean                 <dbl> 0.03534516, -0.21369482, 0.01668160, -0.0~
## $ MCEP13_Median               <dbl> 0.067398181, -0.170491744, -0.002495087, ~
## $ MCEP13_SD                   <dbl> 0.2419947, 0.2434436, 0.1639809, 0.171371~
## $ MCEP13_IQR                  <dbl> 0.2330868, 0.2434911, 0.2047924, 0.201847~
## $ MCEP13_MAD                  <dbl> 0.1785131, 0.1643662, 0.1461172, 0.142785~
## $ MCEP14_Mean                 <dbl> -0.03624464, -0.01700709, -0.08812533, -0~
## $ MCEP14_Median               <dbl> -0.06599079, -0.04405941, -0.10138297, -0~
## $ MCEP14_SD                   <dbl> 0.2126561, 0.1728301, 0.1715754, 0.185861~
## $ MCEP14_IQR                  <dbl> 0.2645698, 0.1919640, 0.2094784, 0.203211~
## $ MCEP14_MAD                  <dbl> 0.19393336, 0.15052423, 0.15546220, 0.147~
## $ MCEP15_Mean                 <dbl> 0.029339533, 0.075035394, 0.039228518, 0.~
## $ MCEP15_Median               <dbl> 0.06326352, 0.09617531, 0.03428203, 0.071~
## $ MCEP15_SD                   <dbl> 0.1687383, 0.1198379, 0.1745691, 0.135412~
## $ MCEP15_IQR                  <dbl> 0.2274155, 0.1911718, 0.1844297, 0.182000~
## $ MCEP15_MAD                  <dbl> 0.1576012, 0.1291894, 0.1370157, 0.136579~
## $ MCEP16_Mean                 <dbl> -0.018902212, -0.007332804, -0.054915496,~
## $ MCEP16_Median               <dbl> -0.031415519, 0.003973841, -0.047630039, ~
## $ MCEP16_SD                   <dbl> 0.1671133, 0.1200883, 0.1412939, 0.135579~
## $ MCEP16_IQR                  <dbl> 0.2325541, 0.1863602, 0.1674624, 0.168561~
## $ MCEP16_MAD                  <dbl> 0.16898038, 0.13035156, 0.12042533, 0.125~
```

```
## $ MCEP17_Mean                 <dbl> 0.062222806, 0.158139342, 0.130739580, 0.~
## $ MCEP17_Median               <dbl> 0.0650740054, 0.1487444829, 0.1205964152,~
## $ MCEP17_SD                   <dbl> 0.15203571, 0.15779716, 0.13289068, 0.144~
## $ MCEP17_IQR                  <dbl> 0.1621526, 0.2051570, 0.1674744, 0.192940~
## $ MCEP17_MAD                  <dbl> 0.12444558, 0.15133865, 0.12455968, 0.144~
## $ MCEP18_Mean                 <dbl> -0.120433671, -0.172700271, -0.014428218,~
## $ MCEP18_Median               <dbl> -0.0904101351, -0.1174760728, -0.00742910~
## $ MCEP18_SD                   <dbl> 0.2056625, 0.2006196, 0.1670167, 0.139316~
## $ MCEP18_IQR                  <dbl> 0.2243093, 0.2308869, 0.1827712, 0.191223~
## $ MCEP18_MAD                  <dbl> 0.16063479, 0.13986433, 0.13624502, 0.142~
## $ MCEP19_Mean                 <dbl> 0.117159546, 0.114212584, 0.008380285, 0.~
## $ MCEP19_Median               <dbl> 0.114921223, 0.077127419, 0.004291641, 0.~
## $ MCEP19_SD                   <dbl> 0.19393664, 0.15471220, 0.15677018, 0.098~
## $ MCEP19_IQR                  <dbl> 0.1963855, 0.1779120, 0.2007070, 0.119655~
## $ MCEP19_MAD                  <dbl> 0.15462003, 0.12523672, 0.14870920, 0.087~
## $ MCEP20_Mean                 <dbl> 0.083725299, -0.003770943, 0.127305887, 0~
## $ MCEP20_Median               <dbl> 0.096173766, -0.006621202, 0.110219812, 0~
## $ MCEP20_SD                   <dbl> 0.16630486, 0.14166424, 0.13939016, 0.131~
## $ MCEP20_IQR                  <dbl> 0.1848750, 0.2215488, 0.1759195, 0.197436~
## $ MCEP20_MAD                  <dbl> 0.13790607, 0.15687353, 0.13287525, 0.141~
## $ MCEP21_Mean                 <dbl> -0.157253275, -0.008839917, -0.147938954,~
## $ MCEP21_Median               <dbl> -0.1635386362, 0.0072850838, -0.128554507~
## $ MCEP21_SD                   <dbl> 0.14139349, 0.10105277, 0.14877063, 0.145~
## $ MCEP21_IQR                  <dbl> 0.1978189, 0.1139186, 0.2086274, 0.205675~
## $ MCEP21_MAD                  <dbl> 0.14979538, 0.08170653, 0.15652125, 0.148~
## $ MCEP22_Mean                 <dbl> 0.10734900, 0.09166880, 0.13374899, 0.132~
## $ MCEP22_Median               <dbl> 0.07511721, 0.06684783, 0.10965862, 0.111~
## $ MCEP22_SD                   <dbl> 0.14857784, 0.15284477, 0.15545739, 0.165~
## $ MCEP22_IQR                  <dbl> 0.1996835, 0.1749956, 0.1896957, 0.219032~
## $ MCEP22_MAD                  <dbl> 0.13574833, 0.13419483, 0.13656928, 0.161~
## $ MCEP23_Mean                 <dbl> -0.064468403, -0.082254478, -0.110631028,~
## $ MCEP23_Median               <dbl> -0.031075494, -0.078492269, -0.079121400,~
## $ MCEP23_SD                   <dbl> 0.14843412, 0.12471937, 0.16704872, 0.132~
## $ MCEP23_IQR                  <dbl> 0.1893956, 0.1735505, 0.2073757, 0.186646~
## $ MCEP23_MAD                  <dbl> 0.13618216, 0.12673728, 0.14732736, 0.133~
## $ MCEP24_Mean                 <dbl> 0.0624111217, -0.0255798885, 0.0155992170~
## $ MCEP24_Median               <dbl> 0.0827723291, -0.0184776351, 0.0090141200~
## $ MCEP24_SD                   <dbl> 0.16058747, 0.11330836, 0.14423073, 0.118~
## $ MCEP24_IQR                  <dbl> 0.22968498, 0.14107863, 0.15687874, 0.149~
## $ MCEP24_MAD                  <dbl> 0.16670648, 0.09981928, 0.11437897, 0.108~
## $ HMPDM10_Mean                <dbl> 0.462916208, 0.299164268, 0.619238809, -0~
## $ HMPDM10_Median              <dbl> 0.247548945, 0.242225778, 0.672442600, -0~
## $ HMPDM10_SD                  <dbl> 0.57935011, 0.38817217, 0.51994772, 0.507~
## $ HMPDM10_IQR                 <dbl> 0.96944020, 0.59651784, 0.67857881, 0.666~
## $ HMPDM10_MAD                 <dbl> 0.53871105, 0.53139045, 0.42084892, 0.442~
## $ HMPDM11_Mean                <dbl> 0.785720168, 0.354838597, 1.058612078, -0~
## $ HMPDM11_Median              <dbl> 0.92712842, 0.19708433, 1.35632912, -0.53~
## $ HMPDM11_SD                  <dbl> 1.3148847, 1.9199271, 1.2388222, 0.861318~
## $ HMPDM11_IQR                 <dbl> 1.8103453, 3.7029098, 1.2299072, 0.886903~
## $ HMPDM11_MAD                 <dbl> 1.4482975, 2.8997051, 0.8583449, 0.666904~
## $ HMPDM12_Mean                <dbl> -0.26059899, -2.05737648, 0.74462738, -0.~
## $ HMPDM12_Median              <dbl> -0.13783108, -2.65844078, 1.61451117, -0.~
## $ HMPDM12_SD                  <dbl> 2.1154896, 1.3339112, 2.0844224, 1.206198~
## $ HMPDM12_IQR                 <dbl> 4.0736177, 1.0636729, 2.8746027, 1.123186~
```

```
## $ HMPDM12_MAD                <dbl> 2.9898295, 0.4623250, 1.5077382, 0.782046~
## $ HMPDM13_Mean               <dbl> 0.097822351, -0.496628148, 0.494025741, -~
## $ HMPDM13_Median             <dbl> -0.07348105, -0.84379036, 1.99950613, -0.~
## $ HMPDM13_SD                 <dbl> 2.2328497, 2.0002194, 2.4765841, 1.303406~
## $ HMPDM13_IQR                <dbl> 4.6036754, 3.9970797, 5.1415816, 2.150568~
## $ HMPDM13_MAD                <dbl> 3.4267581, 2.4003871, 1.4064338, 1.205070~
## $ HMPDM14_Mean               <dbl> 0.29256032, 1.72669905, 1.73035387, -0.50~
## $ HMPDM14_Median             <dbl> 1.002958614, 1.910295614, 2.427255232, -0~
## $ HMPDM14_SD                 <dbl> 2.3590339, 0.6923295, 1.8515397, 0.852251~
## $ HMPDM14_IQR                <dbl> 4.9092249, 1.0648298, 0.8444969, 0.816712~
## $ HMPDM14_MAD                <dbl> 2.6559172, 0.6799794, 0.5552898, 0.594781~
## $ HMPDM15_Mean               <dbl> 0.17156419, 1.97277606, 0.90510418, -0.23~
## $ HMPDM15_Median             <dbl> -0.28239919, 2.02609243, 2.10470053, 0.01~
## $ HMPDM15_SD                 <dbl> 2.3635938, 0.9175364, 2.2677094, 0.995837~
## $ HMPDM15_IQR                <dbl> 4.7651908, 0.5229397, 4.2615835, 0.984093~
## $ HMPDM15_MAD                <dbl> 3.6137636, 0.2738011, 0.8966952, 0.727603~
## $ HMPDM16_Mean               <dbl> 0.56299793, 1.10772412, 1.43354852, -0.21~
## $ HMPDM16_Median             <dbl> 1.83403750, 2.26266763, 2.31948061, -0.13~
## $ HMPDM16_SD                 <dbl> 2.3660132, 2.1633942, 1.9803967, 1.199829~
## $ HMPDM16_IQR                <dbl> 4.7938983, 2.0415720, 1.1010347, 1.299731~
## $ HMPDM16_MAD                <dbl> 1.7062209, 0.5225194, 0.5902522, 0.960349~
## $ HMPDM17_Mean               <dbl> -0.14891508, 0.76087307, 0.65391101, -0.0~
## $ HMPDM17_Median             <dbl> -0.13637987, 1.50257720, 1.48832050, -0.0~
## $ HMPDM17_SD                 <dbl> 2.2660741, 1.6564394, 2.1785880, 1.554046~
## $ HMPDM17_IQR                <dbl> 4.6286819, 1.7477955, 4.6375379, 1.986361~
## $ HMPDM17_MAD                <dbl> 3.4579266, 1.0500901, 1.7415496, 1.381364~
## $ HMPDM18_Mean               <dbl> -0.14745062, -0.89027935, 1.58018005, 0.1~
## $ HMPDM18_Median             <dbl> -0.80476168, -1.56266609, 2.25468713, -0.~
## $ HMPDM18_SD                 <dbl> 2.1839925, 1.0650795, 1.6333817, 1.524495~
## $ HMPDM18_IQR                <dbl> 4.5327626, 1.7430205, 1.0521880, 2.029207~
## $ HMPDM18_MAD                <dbl> 2.8487966, 0.7780924, 0.5903986, 1.757841~
## $ HMPDM19_Mean               <dbl> 0.213523473, -0.528374511, 0.420239997, 1~
## $ HMPDM19_Median             <dbl> 0.346446386, -1.304565990, 0.930928035, 1~
## $ HMPDM19_SD                 <dbl> 2.2084954, 1.9435859, 2.2619935, 1.645159~
## $ HMPDM19_IQR                <dbl> 4.5546990, 3.6124929, 4.2801849, 1.997815~
## $ HMPDM19_MAD                <dbl> 3.0771108, 1.5480793, 2.6015180, 1.409637~
## $ HMPDM20_Mean               <dbl> -1.194983939, 0.331909980, 0.984798155, 0~
## $ HMPDM20_Median             <dbl> -1.716786707, 0.492041931, 2.296511339, -~
## $ HMPDM20_SD                 <dbl> 1.3754961, 1.5342994, 2.1564130, 1.372549~
## $ HMPDM20_IQR                <dbl> 2.4752775, 1.4055726, 3.3378901, 1.778124~
## $ HMPDM20_MAD                <dbl> 1.5573955, 1.0305474, 1.1540370, 1.321572~
## $ HMPDM21_Mean               <dbl> -0.648115956, -0.144724354, 0.947089615, ~
## $ HMPDM21_Median             <dbl> -0.501558665, -0.065206616, 2.278419332, ~
## $ HMPDM21_SD                 <dbl> 2.2195806, 1.5470122, 2.3032490, 1.025915~
## $ HMPDM21_IQR                <dbl> 4.1148162, 1.5703220, 4.5685775, 0.968569~
## $ HMPDM21_MAD                <dbl> 3.1805463, 1.5739852, 1.0357899, 0.677026~
## $ HMPDM22_Mean               <dbl> 0.36327975, 0.07293082, 0.83382914, -0.24~
## $ HMPDM22_Median             <dbl> 0.73558804, 0.48903061, 2.10661438, -0.24~
## $ HMPDM22_SD                 <dbl> 1.9309571, 1.8676068, 2.2065057, 1.678771~
## $ HMPDM22_IQR                <dbl> 3.1388046, 3.5682869, 4.3203887, 2.360193~
## $ HMPDM22_MAD                <dbl> 1.9920810, 2.4503021, 1.2341223, 1.918375~
## $ HMPDM23_Mean               <dbl> -0.66428089, -0.22894930, -0.18222487, -0~
## $ HMPDM23_Median             <dbl> -0.999385670, -0.004583891, -0.276236113,~
## $ HMPDM23_SD                 <dbl> 1.3087729, 1.1947151, 1.5207685, 2.018825~
```

```
## $ HMPDM23_IQR                <dbl> 2.3981686, 1.9782856, 2.3396626, 3.293142~
## $ HMPDM23_MAD                <dbl> 1.5649485, 1.3485971, 1.7435392, 2.439599~
## $ HMPDM24_Mean               <dbl> -0.135118031, 0.133342098, 0.584839923, -~
## $ HMPDM24_Median             <dbl> -0.04389285, -1.00946570, 0.39765459, -1.~
## $ HMPDM24_SD                 <dbl> 1.329261, 2.090515, 1.756946, 1.808786, 1~
## $ HMPDM24_IQR                <dbl> 1.681760, 4.235691, 3.135425, 2.915209, 3~
## $ HMPDM24_MAD                <dbl> 1.3540894, 2.4167004, 2.5441505, 1.098470~
## $ HMPDD0_Mean                <dbl> -0.3408285, -0.1645714, -0.2692037, -0.25~
## $ HMPDD0_Median              <dbl> -0.3075534, -0.1522718, -0.2520761, -0.21~
## $ HMPDD0_SD                  <dbl> 0.5174648, 0.1015647, 0.1197612, 0.447922~
## $ HMPDD0_IQR                 <dbl> 0.1048730, 0.1412477, 0.1470585, 0.115853~
## $ HMPDD0_MAD                 <dbl> 0.07421019, 0.09621325, 0.10612883, 0.089~
## $ HMPDD1_Mean                <dbl> -1.2535007, -0.9959845, -1.1447973, -1.11~
## $ HMPDD1_Median              <dbl> -1.266898, -1.022398, -1.134259, -1.13271~
## $ HMPDD1_SD                  <dbl> 0.1623235, 0.1391240, 0.1353228, 0.151761~
## $ HMPDD1_IQR                 <dbl> 0.12033199, 0.09390683, 0.14035092, 0.170~
## $ HMPDD1_MAD                 <dbl> 0.08675471, 0.06950135, 0.10316261, 0.130~
## $ HMPDD2_Mean                <dbl> -1.0933339, -0.8545146, -0.9924906, -0.99~
## $ HMPDD2_Median              <dbl> -1.1045322, -0.8761917, -0.9852409, -1.00~
## $ HMPDD2_SD                  <dbl> 0.12948211, 0.09619218, 0.09781110, 0.113~
## $ HMPDD2_IQR                 <dbl> 0.13966772, 0.13723257, 0.09755791, 0.105~
## $ HMPDD2_MAD                 <dbl> 0.10226742, 0.08835237, 0.07426567, 0.076~
## $ HMPDD3_Mean                <dbl> -0.9779883, -0.8063730, -0.9194741, -0.91~
## $ HMPDD3_Median              <dbl> -0.9766211, -0.8210046, -0.9173615, -0.92~
## $ HMPDD3_SD                  <dbl> 0.14187276, 0.08538769, 0.07739077, 0.110~
## $ HMPDD3_IQR                 <dbl> 0.18141410, 0.11729633, 0.09847216, 0.103~
## $ HMPDD3_MAD                 <dbl> 0.13187451, 0.08532394, 0.07236728, 0.081~
## $ HMPDD4_Mean                <dbl> -0.8284356, -0.7477489, -0.8013249, -0.80~
## $ HMPDD4_Median              <dbl> -0.8555960, -0.7756226, -0.8025337, -0.81~
## $ HMPDD4_SD                  <dbl> 0.14579275, 0.07962604, 0.07937704, 0.105~
## $ HMPDD4_IQR                 <dbl> 0.19246413, 0.05958430, 0.10207012, 0.127~
## $ HMPDD4_MAD                 <dbl> 0.12848827, 0.04271755, 0.07693148, 0.096~
## $ HMPDD5_Mean                <dbl> -0.7964747, -0.6979204, -0.7347430, -0.73~
## $ HMPDD5_Median              <dbl> -0.7897005, -0.7097710, -0.7404063, -0.73~
## $ HMPDD5_SD                  <dbl> 0.08228026, 0.07613074, 0.08130780, 0.095~
## $ HMPDD5_IQR                 <dbl> 0.09774727, 0.06568892, 0.08540655, 0.124~
## $ HMPDD5_MAD                 <dbl> 0.06738755, 0.05046784, 0.06329457, 0.092~
## $ HMPDD6_Mean                <dbl> -0.7154683, -0.6693892, -0.6814118, -0.68~
## $ HMPDD6_Median              <dbl> -0.7244345, -0.6868018, -0.6922698, -0.69~
## $ HMPDD6_SD                  <dbl> 0.10084335, 0.06852692, 0.07377278, 0.090~
## $ HMPDD6_IQR                 <dbl> 0.11685441, 0.04693702, 0.06503571, 0.083~
## $ HMPDD6_MAD                 <dbl> 0.08499778, 0.03363377, 0.04865246, 0.061~
## $ HMPDD7_Mean                <dbl> -0.5872387, -0.6134421, -0.5931024, -0.60~
## $ HMPDD7_Median              <dbl> -0.5869346, -0.6180031, -0.5994557, -0.61~
## $ HMPDD7_SD                  <dbl> 0.08509958, 0.06776318, 0.06519572, 0.107~
## $ HMPDD7_IQR                 <dbl> 0.11087414, 0.11045517, 0.07996860, 0.110~
## $ HMPDD7_MAD                 <dbl> 0.08270070, 0.08096909, 0.05920412, 0.085~
## $ HMPDD8_Mean                <dbl> -0.4679594, -0.5195625, -0.5123410, -0.49~
## $ HMPDD8_Median              <dbl> -0.4698443, -0.5402711, -0.5219495, -0.49~
## $ HMPDD8_SD                  <dbl> 0.07939885, 0.06533846, 0.06643105, 0.079~
## $ HMPDD8_IQR                 <dbl> 0.12157351, 0.08528184, 0.10034663, 0.106~
## $ HMPDD8_MAD                 <dbl> 0.09147356, 0.05137329, 0.07175205, 0.074~
## $ HMPDD9_Mean                <dbl> -0.338200382, -0.449000122, -0.414840764,~
## $ HMPDD9_Median              <dbl> -0.333857995, -0.442950807, -0.416892315,~
```

```
## $ HMPDD9_SD                    <dbl> 0.08174587, 0.05630173, 0.05954582, 0.066~
## $ HMPDD9_IQR                   <dbl> 0.08666532, 0.08180327, 0.07048822, 0.082~
## $ HMPDD9_MAD                   <dbl> 0.06503703, 0.06169567, 0.05192746, 0.062~
## $ HMPDD10_Mean                 <dbl> -0.239247045, -0.389175838, -0.314760191,~
## $ HMPDD10_Median               <dbl> -0.25064121, -0.40591475, -0.32052490, -0~
## $ HMPDD10_SD                   <dbl> 0.09501607, 0.06001945, 0.08044657, 0.073~
## $ HMPDD10_IQR                  <dbl> 0.11864709, 0.09641070, 0.10179278, 0.107~
## $ HMPDD10_MAD                  <dbl> 0.09151898, 0.05903690, 0.07603589, 0.082~
## $ HMPDD11_Mean                 <dbl> -0.122003681, -0.278385501, -0.226977184,~
## $ HMPDD11_Median               <dbl> -0.1223405, -0.2733205, -0.2329398, -0.24~
## $ HMPDD11_SD                   <dbl> 0.10154515, 0.05720821, 0.07671794, 0.078~
## $ HMPDD11_IQR                  <dbl> 0.12123917, 0.08738290, 0.10245011, 0.083~
## $ HMPDD11_MAD                  <dbl> 0.09402148, 0.06778729, 0.07595104, 0.054~
## $ HMPDD12_Mean                 <dbl> -0.08097356, -0.23458833, -0.17156862, -0~
## $ HMPDD12_Median               <dbl> -0.08285349, -0.23010835, -0.18476613, -0~
## $ HMPDD12_SD                   <dbl> 0.09529543, 0.06127831, 0.07358020, 0.073~
## $ HMPDD12_IQR                  <dbl> 0.11987141, 0.08167643, 0.09139550, 0.072~
## $ HMPDD12_MAD                  <dbl> 0.08947968, 0.05861450, 0.06553915, 0.054~
## $ Harmonicity_Mean             <dbl> 0.6462543, 0.7163654, 0.8055542, 0.709283~
## $ Harmonicity_SD               <dbl> 0.4555504, 0.4600380, 0.4762750, 0.432797~
## $ Clarity_Mean                 <dbl> 0.6575821, 0.6804585, 0.7029027, 0.671836~
## $ Clarity_SD                   <dbl> 0.1593496, 0.1467127, 0.1449230, 0.131540~
## $ LPerror_Mean                 <dbl> 6.911738, 8.040148, 7.026986, 7.208061, 6~
## $ LPerror_SD                   <dbl> 2.229282, 1.674179, 2.013750, 1.865100, 2~
## $ HarmonicProductSpectrum_Mean <dbl> -13.785519, -42.346709, -9.642031, -6.450~
## $ HarmonicProductSpectrum_SD   <dbl> 92.39493, 55.07227, 94.35489, 101.06790, ~
## $ CepstralPeakProminence_Mean  <dbl> 4.545577, 4.397983, 4.547749, 4.383870, 4~
## $ CepstralPeakProminence_SD    <dbl> 0.5295720, 0.4481834, 0.6342335, 0.502741~
## $ Srh1_Mean                    <dbl> 0.1962962, 0.1807567, 0.1951253, 0.190560~
## $ Srh1_SD                      <dbl> 0.03753496, 0.03224129, 0.03797110, 0.038~
## $ Srh2_Mean                    <dbl> 13.321604, 5.053818, 8.798485, 13.324161,~
## $ Srh2_SD                      <dbl> 10.728828, 3.842785, 7.440368, 13.270542,~
## $ creakF0_Mean                 <dbl> 149.1352, 119.1684, 128.7470, 129.9000, 1~
## $ creakF0_SD                   <dbl> 28.846649, 17.551321, 15.329024, 12.77894~
## $ CreakProbability_Mean        <dbl> 0.107691033, 0.322322607, 0.150356314, 0.~
## $ CreakProbability_SD          <dbl> 0.15440934, 0.24655873, 0.18271252, 0.169~
## $ PauseNumMin_Cova             <dbl> 3.333333, 3.103448, 2.821317, 3.111111, 3~
## $ MeanPauseDur_Cova            <dbl> 0.19052632, 0.23333333, 0.17592593, 0.213~
## $ TurnNumMin_Cova              <dbl> 3.333333, 3.103448, 2.821317, 3.111111, 3~
## $ MeanTurnDur_Cova             <dbl> 0.10947368, 0.08888889, 0.17851852, 0.107~
## $ PauseNumMin_Praat            <dbl> 2.135231, 2.127660, 2.107482, 1.121076, 3~
## $ TurnNumMin_Praat             <dbl> 2.135231, 1.773050, 2.002107, 1.121076, 3~
## $ MeanTurnDur_Praat            <dbl> 0.1925000, 0.1640000, 0.2347368, 0.299000~
```

[] data <- data_raw %>% rename_with(.cols = everything(), str_to_lower) %>% rename(id = patid, condition = diagnosis) %>% mutate(across(where(is.character), str_to_lower), across(1 : 7, as_factor), condition = if_else(condition != ct, sz, hc) %>% as_factor %>% relevel(sz)) %>% select(-newid) data *language*%>%*summary*

```
##    d
## 1889
```

[] data *corpus*%>%*summary*

```
##   1   2   3   4
## 681 363 375 470
```

[] data <- data %>% select(-language)
head(data)

```
## # A tibble: 6 x 396
##   id    condition gender trial corpus duration~1 f0_me~2 f0_sd~3 inten~4 inten~5
##   <fct> <fct>     <fct>  <fct> <fct>       <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 101   hc        m      t7    1            5.62    157.   37.2     70.2    6.11
## 2 101   hc        m      t8    1            2.82    115.    5.04    67.5    5.40
## 3 101   hc        m      t4    1            9.49    125.    9.10    70.2    6.73
## 4 101   hc        m      t2    1            8.92    133.   19.5     70.4    6.29
## 5 101   hc        m      t3    1            6       122.   13.5     67.4    6.59
## 6 101   hc        m      t5    1           12.6     132.   22.9     69.2    6.26
## # ... with 386 more variables: pauseduration_praat <dbl>,
## #   turnduration_praat <dbl>, turnnumber_praat <dbl>, pausenumber_praat <dbl>,
## #   percentspoke_praat <dbl>, percentsilence_praat <dbl>, nhr_mean <dbl>,
## #   nhr_std <dbl>, duration_cova <dbl>, pauseduration_cova <dbl>,
## #   turnduration_cova <dbl>, turnnumber_cova <dbl>, pausenumber_cova <dbl>,
## #   percentspoke_cova <dbl>, percentsilence_cova <dbl>, pitch_mean <dbl>,
## #   pitch_median <dbl>, pitch_sd <dbl>, pitch_iqr <dbl>, pitch_mad <dbl>, ...
```

describing-the-data

Describing the data

condition

Condition

[] data %>% count(condition) %>% mutate(pct = n / sum(n), pct = pct %>% round(2))

```
## # A tibble: 2 x 3
##   condition     n   pct
##   <fct>     <int> <dbl>
## 1 sz          900  0.48
## 2 hc          989  0.52
```

gender

Gender

[] data %>% count(gender) %>% mutate(pct = n / sum(n), pct = pct %>% round(2))

```
## # A tibble: 2 x 3
##   gender     n   pct
##   <fct>  <int> <dbl>
## 1 m       1081  0.57
## 2 f        808  0.43
```

[] data %>% count(gender, condition) %>% group_by(condition) %>% mutate(pct = n / sum(n), pct = pct %>% round(2))

```
## # A tibble: 4 x 4
## # Groups:   condition [2]
```

```
##   gender condition     n   pct
##   <fct>  <fct>     <int> <dbl>
## 1 m      sz          515  0.57
## 2 m      hc          566  0.57
## 3 f      sz          385  0.43
## 4 f      hc          423  0.43
```

[] data %>% count(corpus) %>% mutate(pct = n / sum(n), pct = pct %>% round(2))

```
## # A tibble: 4 x 3
##   corpus     n   pct
##   <fct>  <int> <dbl>
## 1 1        681  0.36
## 2 2        363  0.19
## 3 3        375  0.2
## 4 4        470  0.25
```

[] data %>% count(condition, corpus) %>% group_by(condition) %>% mutate(pct = n / sum(n), pct = pct %>% round(2))

```
## # A tibble: 8 x 4
## # Groups:   condition [2]
##   condition corpus     n   pct
##   <fct>     <fct>  <int> <dbl>
## 1 sz        1        333  0.37
## 2 sz        2        179  0.2
## 3 sz        3        151  0.17
## 4 sz        4        237  0.26
## 5 hc        1        348  0.35
## 6 hc        2        184  0.19
## 7 hc        3        224  0.23
## 8 hc        4        233  0.24
```

modeling-the-data

Modeling the data

budgeting

Budgeting

[] data_background <- data %>% select(1 : 5) data <- data %>% select(- c(gender, corpus))
split <- initial_split(data, prop = 4 / 5)
data_training <- training(split) data_testing <- testing(split)
rm(split)
preprocessing-the-data-1

Preprocessing the data

[] recipes <- list()
base <- recipe(condition ~ 1 + ., data = data_training) %>% update_role(id, trial, new_role = id) %>% step_normalize(all_numeric())
recipes[[1]] <- base %>% step_corr(all_predictors()) recipes[[2]] <- base %>% step_pca(all_predictors())
names(recipes) <- c(corr, pca)
creating-the-models-1

Creating the models

[] prior_b <- normal(location = 0, scale = 0.3, autoscale = T) prior_intercept <- normal(0, 1, autoscale = T)

model_prior <- logistic_reg() %>% set_engine(stan, prior = prior_b, prior_intercept = prior_intercept, prior_PD = T, cores = 3)

model <- logistic_reg() %>% set_engine(stan, prior = prior_b, prior_intercept = prior_intercept, cores = 3)

workflows-1

Workflows

[] wflows <- map(recipes, ~ workflow() %>% add_model(model) %>% add_recipe(.x))

fitting-the-models

Fitting the models

[] set.seed(1) fitted <- map(wflows, ~ .x %>% fit(data_training))

fitted_models <- map(fitted, extract_fit_engine)

set.seed(1) prior_fitted <- map(recipes, ~ workflow() %>% add_model(model_prior) %>% add_recipe(.x) %>% fit(data_training) %>% extract_fit_engine())

convergance-checks-1

Convergance checks

[] convergance_plots <- map2(fitted_models, names(fitted_models), **function**(.x, .y){ list(plot(.x, trace, pars = (Intercept)), *#think about which estimates to include and add this here* plot(.x, neff), plot(.x, rhat)) %>% map(**function**(.x){.x + ggtitle(.y)}) })

convergance_plots %>% print

```
## $corr
## $corr[[1]]



##
## $corr[[2]]



##
## $corr[[3]]



##
##
## $pca
## $pca[[1]]



##
## $pca[[2]]



##
## $pca[[3]]
```

[] rm(convergance_plots)

[] tidy_pca <- tidy(fitted[[2]] %>% extract_recipe, 2)

tidy_pca %>% filter(component %in% paste0(PC, 1 : 5)) %>% group_by(component) %>% top_n(15, abs(value)) %>% ungroup() %>% mutate(terms = reorder_within(terms, abs(value), component)) %>% ggplot(aes(value, terms)) + geom_col() + facet_wrap(~component, scale = free_y) + scale_y_reordered() + theme_minimal()

[] *#ggsave(pca_interpretation.png, height = 10, width = 12, bg = white)*

[] variables_corr <- get_variables(fitted_models[[1]]) %>% str_subset(.*__, negate = T) %>% *#removing all the technical variables (e.g. treedepth__, stepsize__)* str_subset((Intercept), negate = T)

variables_corr <- c((Intercept), variables_corr %>% str_subset(mcep.*) %>% sample(3, replace = F), variables_corr %>% str_subset(hmpdm.*) %>% sample(3, replace = F), variables_corr %>% str_subset(., mcep.*—hmpd.*, negate = T) %>% sample(3, replace = F)) *#drawing different variables from different types of measures to plot as a sample in the pior-posterior update plots*

variables_pca <- get_variables(fitted_models[[2]]) %>% str_subset(.*__, negate = T)

[] pp_update_plot <- **function**(prior_model, posterior_model, variables){

df_draws <- bind_rows(prior_model %>% gather_draws(!!! syms(variables)) %>% mutate(type = prior),

posterior_model %>% gather_draws(!!! syms(variables)) %>% mutate(type = posterior))

df_draws <- df_draws %>% group_by(.variable) %>% mutate(upp_lim = if_else((max(.value) + min(.value)) > 0, max(.value), - min(.value)), low_lim = - upp_lim) %>% ungroup

df_draws %>% ggplot(aes(x = .value, fill = type)) + geom_density(alpha = 0.7) + labs(fill = element_blank()) + xlim(df_draws *low_lim*[[1]], *df_draws*upp_lim[[1]]) + facet_grid(vars(df_draws *.variable*))+*theme_minimal*()+*theme*(*axis.ticks.y* =*element_blank*(), *axis.text.y* =*element_blank*())}

[] pp_update_plot(prior_fitted[[1]], fitted_models[[1]], variables_corr)

[] *#ggsave(pp_upadte_corr.png, height = 8, bg = white)*

pp_update_plot(prior_fitted[[2]], fitted_models[[2]], variables_pca)

[] *#ggsave(pp_update_pca.png, height = 8, bg = white)*

[] test_preds <- map(fitted, ~ augment(.x, data_training))

rocs <- map2(test_preds, names(test_preds), ~ .x %>% roc_curve(truth = condition, .pred_sz) %>% autoplot + ggtitle(.y)) rocs

```
## $corr
```

```
##
## $pca
```

[] *#ggsave(plot = rocs[[1]], filename = roc_corr.png) #ggsave(plot = rocs[[2]], filename = roc_pca.png)*

[] *#with the uncertainty*

test_results <- tibble(draw = NULL, f1 = NULL, model = NULL)

for (i **in** seq_along(fitted_models)){

m <- fitted_models[[i]] name <- names(fitted_models)[[i]]

draws_matrix <- posterior_epred(m)

roc_aucs <- map_dbl(draws_matrix %>% split(row(draws_matrix)), ~ roc_auc_vec(truth = data_training *condition*, *estimate* =*.x*))*roc_aucs*<−*tibble*(*value* =*roc_aucs*, *metric* =*roc_auc*, *draw* =*seq_along*(*nrow*))

preds_class <- map(draws_matrix %>% split(row(draws_matrix)), ~ if_else(.x < 0.5, sz, hc) %>% as_factor %>% relevel(sz))

fs <- map_dbl(preds_class, ~ f_meas_vec(truth = data_training *condition*, *estimate* =*.x*))*fs*<−*tibble*(*value* =*fs*, *metric*

test_results <- bind_rows(test_results, bind_rows(fs, roc_aucs) %>% mutate(model = name)) } rm(i, m, name, draws_matrix, roc_aucs, preds_class, fs)

test_results <- test_results %>% mutate(value = if_else(metric == roc_auc, 1 - value, value))

test_results %>% ggplot(aes(x = model, y = value, colour = model)) + geom_point(alpha = 0.7) + geom_hline(yintercept = 0.5, color = darkred, linetype = dashed, alpha = 0.7) + theme_minimal() + facet_wrap(vars(metric))

[] #ggsave(test_results.png, bg = white)