Assignment 3 - Random Forest models

Diagnosing Schizophrenia from Voice

Amélie (202206397@post.au.dk),
Katharina (202003848@post.au.dk),
Lilla (202206288@post.au.dk),
Malene (2022204323@post.au.dk)

2023-12-15

This assignment is based on the following paper:

Parola A et al. 2023. Voice Patterns as Markers of Schizophrenia: Building a Cumulative Generalizable Approach Via a Cross-Linguistic and Meta-analysis Based Investigation. Schizophrenia Bulletin 22(49):S125-S141.

Individuals with schizophrenia (SCZ) tend to present voice atypicalities. Their tone is described as "inappropriate" voice, sometimes monotone, sometimes croaky. This is important for two reasons. First, voice could constitute a direct window into cognitive, emotional, and social components of the disorder, thus providing a cheap and relatively non-invasive way to support the diagnostic and assessment process (via automated analyses). Secondly, voice atypicalities play an important role in the social impairment experienced by individuals with SCZ, and are thought to generate negative social judgments (of unengaged, slow, unpleasant interlocutors), which can cascade in more negative and less frequent social interactions.

While several studies show significant differences in acoustic features by diagnosis, we want to know whether we can diagnose SCZ in participants only from knowing the features of their voice. To that end, the authors collected data from various relevant studies. The latter focused on analyzing voice recordings from people that just got a first diagnosis of schizophrenia, along with a 1:1 case-control sample of participants with matching gender, age, and education. Each participant watched several videos (here called trials) of triangles moving across the screen and had to describe them, so you have several recordings per person. Along with these files, pitch was recorded once every 10 milliseconds for each participant and various duration-related statistics were also collected (e.g. number of pauses). For the purpose of this assignment, studies conducted in languages other than Danish were filtered out.

Collecting and cleaning the project data (Lilla, Katharina)

1. **articulation_data.txt**. This file contains all duration-related data collected from the participants to the different studies included in the project. Here is a short description of its linguistic variables.

- *nsyll:* number of syllables automatically inferred from the audio
- *npause:* number of pauses automatically inferred from the audio (absence of human voice longer than 200 milliseconds)
- *dur (s):* duration (in seconds) of the full recording
- *phonationtime (s):* duration (in seconds) of the recording where speech is present
- *speechrate (nsyll/dur):**average number of syllables per second
- *articulation rate (nsyll/ phonationtime):* average number of syllables per second where speech is present

- *ASD (speakingtime/nsyll):* average syllable duration

```
articulation_data <- read_delim("~/Documents/Uni/Semester 03/Methods-3/A3/data/articulation_data.txt")
articulation_data <- read_delim("~/Documents/Uni/Semester 03/Methods-3/A3/data/articulation_data_update
articulation_data <- read_delim("~/Documents/Uni/Semester 03/Methods-3/A3/data/articulation_data_update

articulation_data <- articulation_data %>%
  rename(
    speech_rate = `speech_rate (n_syllables/duration)`,
    articulation_rate = `articulation_rate (n_syllables/phonation_duration)`,
  )
```

2. **pitch_data.txt**. Aggregated pitch data collected from the participants to the different studies included in the project. Fundamental pitch frequency was recorded for each participant every 10 milliseconds (excluding pauses) and aggregated at the participant trial level with the use of various centrality and dispersion measures. While most column names are self-explanatory, the following might be hard to figure out:

- *iqr:* Interquartile range
- *mad:* Mean absolute deviation
- *coefvar:* Coefficient of variation

```
#This data is also tab-delimited
pitch_data <- read_delim("~/Documents/Uni/Semester 03/Methods-3/A3/data/pitch_data.txt")
pitch_data <- read_delim("~/Documents/Uni/Semester 03/Methods-3/A3/data/pitch_data_updated.txt")

#renaming pitch_data to match lowercase standard
pitch_data <- pitch_data %>%
  rename(
    study = Study,
    diagnosis = Diagnosis,
    id = ID,
    trial = Trial
  )
```

3. **demographic_data.txt**. This file contains demographic information on all participants which we won't include for the prediction of diagnosis later on, but can use to get a better overview over our participants and the balance between diagnosed and control group in terms of age distribution, gender distribution, ect.

```
demographic_data <- read_delim("~/Documents/Uni/Semester 03/Methods-3/A3/data/demographic_data.txt")
```

```
#merging in two parts since demographic doesnt share trial

merged_part_one <- merge(pitch_data, demographic_data, by = c("study", "id", "diagnosis"))

merged_set <- merge(articulation_data, merged_part_one, by = c("study", "id", "diagnosis",
                                                               "trial"))

#Renaming the columns

schizophrenia_and_articulation <- merged_set %>%
  rename(
    pitch_interquartile_range = iqr,
    pitch_mean_absolute_deviation = mad,
    pitch_coefficient_of_variation = coefvar,
    pitch_mean= mean,
    pitch_sd= sd,
    pitch_min= min,
    pitch_max= max,
    pitch_median= median,
  )

write_csv(schizophrenia_and_articulation, "~/Documents/Uni/Semester 03/Methods-3/A3/data/schizophrenia_a
```

Overview over NAs

```
summarise_all(schizophrenia_and_articulation, ~sum(is.na(.)))
```

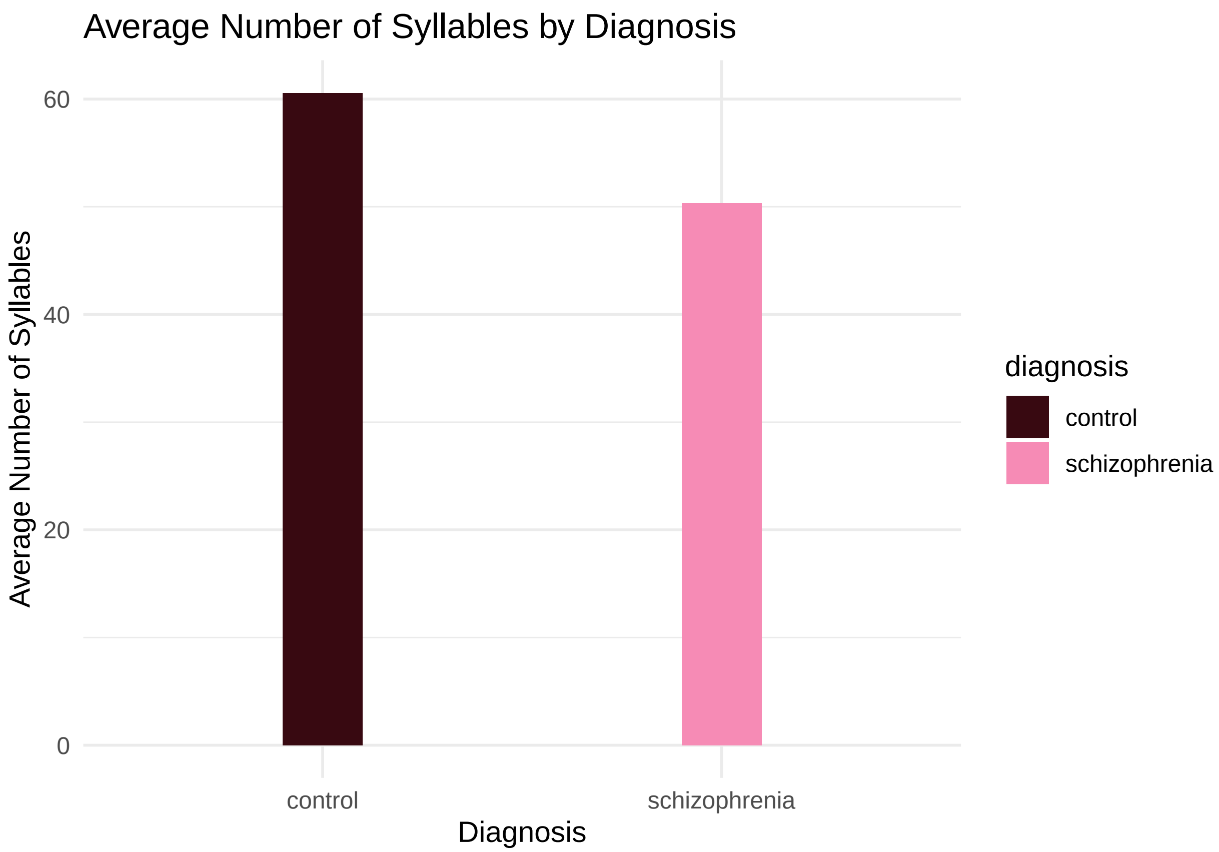
```
##   study id diagnosis trial duration phonation_duration pause_duration
## 1     0  0         0     0        0                  0              0
##   n_syllables n_pauses speech_rate articulation_rate average_syllable_duration
## 1           0      105           0                 0                         0
##   average_pause_duration pitch_mean pitch_sd pitch_min pitch_max pitch_median
## 1                    105          0        0         0         0            0
##   pitch_interquartile_range pitch_mean_absolute_deviation
## 1                         0                             0
##   pitch_coefficient_of_variation gender age education
## 1                              0      0  16         0
```

```
#n_pauses: 105 NAs
#averagge_pause duration: 105 NAs
#age: 16 NAs
```

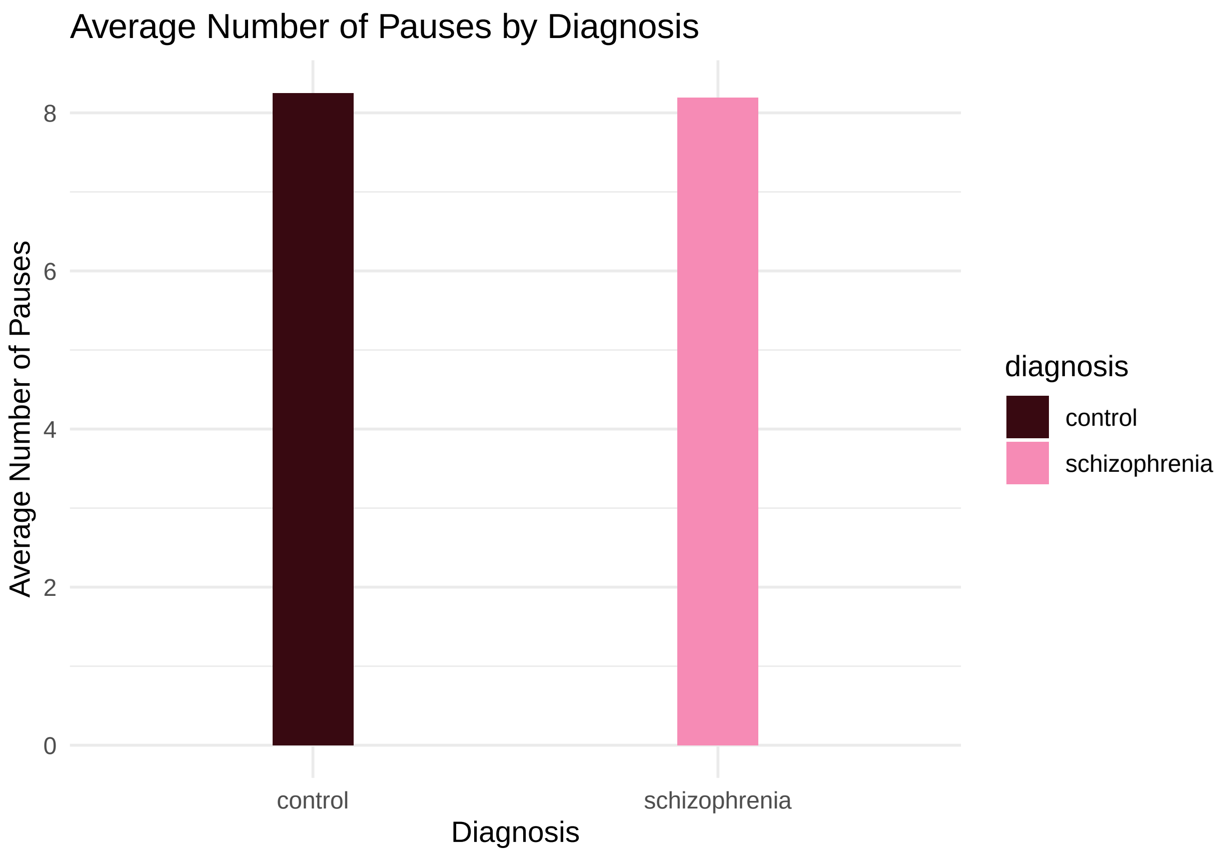
The rest of the data from the rows containing NA's is still useful, therefore we will decide by-plot on what to do with the NAs, and will not delete the rows.

Understanding the sample using descriptive statistics (Amélie, Malene)

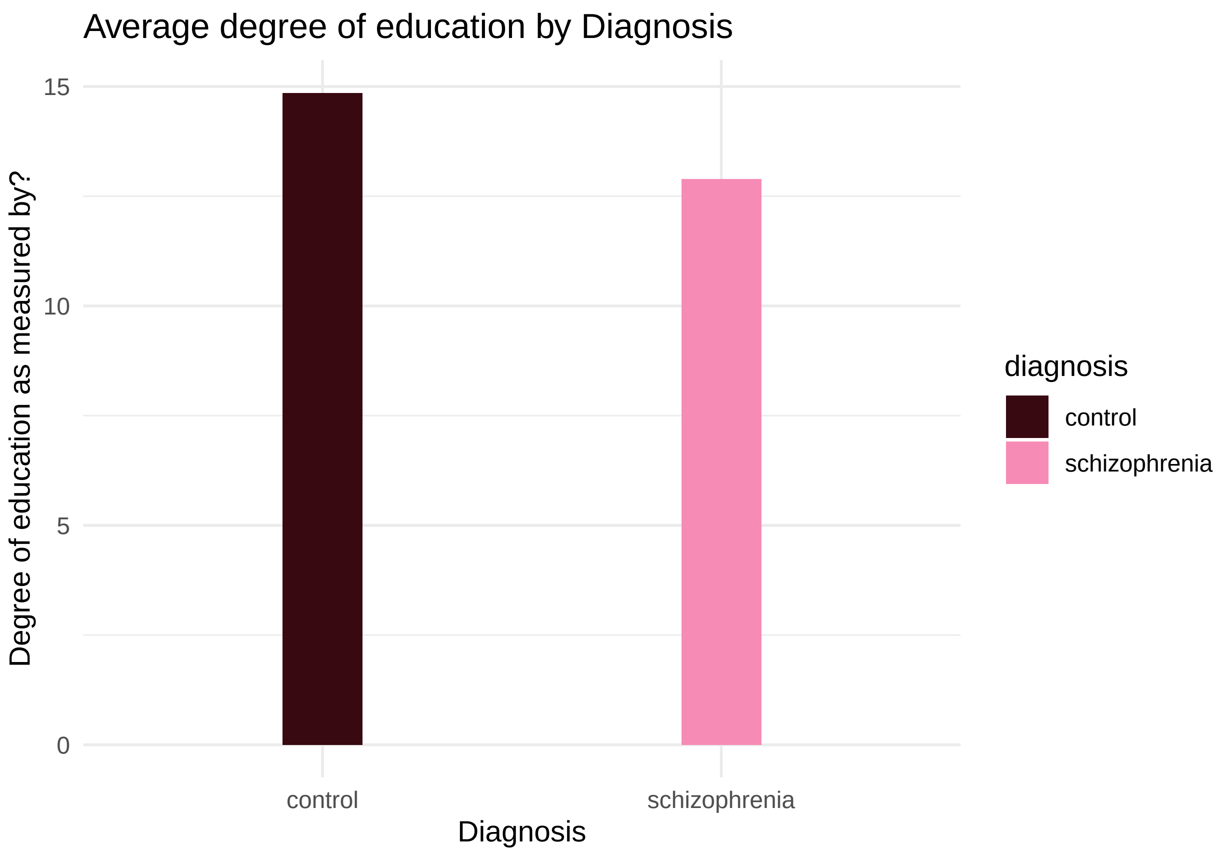
```
# making the diagnosis a factor
schizophrenia_and_articulation$diagnosis <- factor(schizophrenia_and_articulation$diagnosis)

# making a bar plot
ggplot(schizophrenia_and_articulation, aes(x = diagnosis, y = n_syllables,
                                           fill = diagnosis)) +
  geom_bar(stat = "summary", fun = "mean", position = "dodge", width=0.2) +
  labs(title = "Average Number of Syllables by Diagnosis", x = "Diagnosis",
       y = "Average Number of Syllables")+
  scale_fill_manual(values = global_palette)
```



```r
#Pauses has NAs. Replace with average for the diagnosis.
# doing the same but with pauses
schizophrenia_and_articulation_pause_NA_fixed <- schizophrenia_and_articulation %>%
  group_by(diagnosis) %>%
  mutate(n_pauses = ifelse(is.na(n_pauses), mean(n_pauses, na.rm = TRUE), n_pauses))

ggplot(schizophrenia_and_articulation_pause_NA_fixed, aes(x = diagnosis, y = n_pauses,
                                                          fill = diagnosis)) +
  geom_bar(stat = "summary", fun = "mean", position = "dodge", width=0.2) +
  labs(title = "Average Number of Pauses by Diagnosis", x = "Diagnosis",
       y = "Average Number of Pauses")+
  scale_fill_manual(values = global_palette)
```



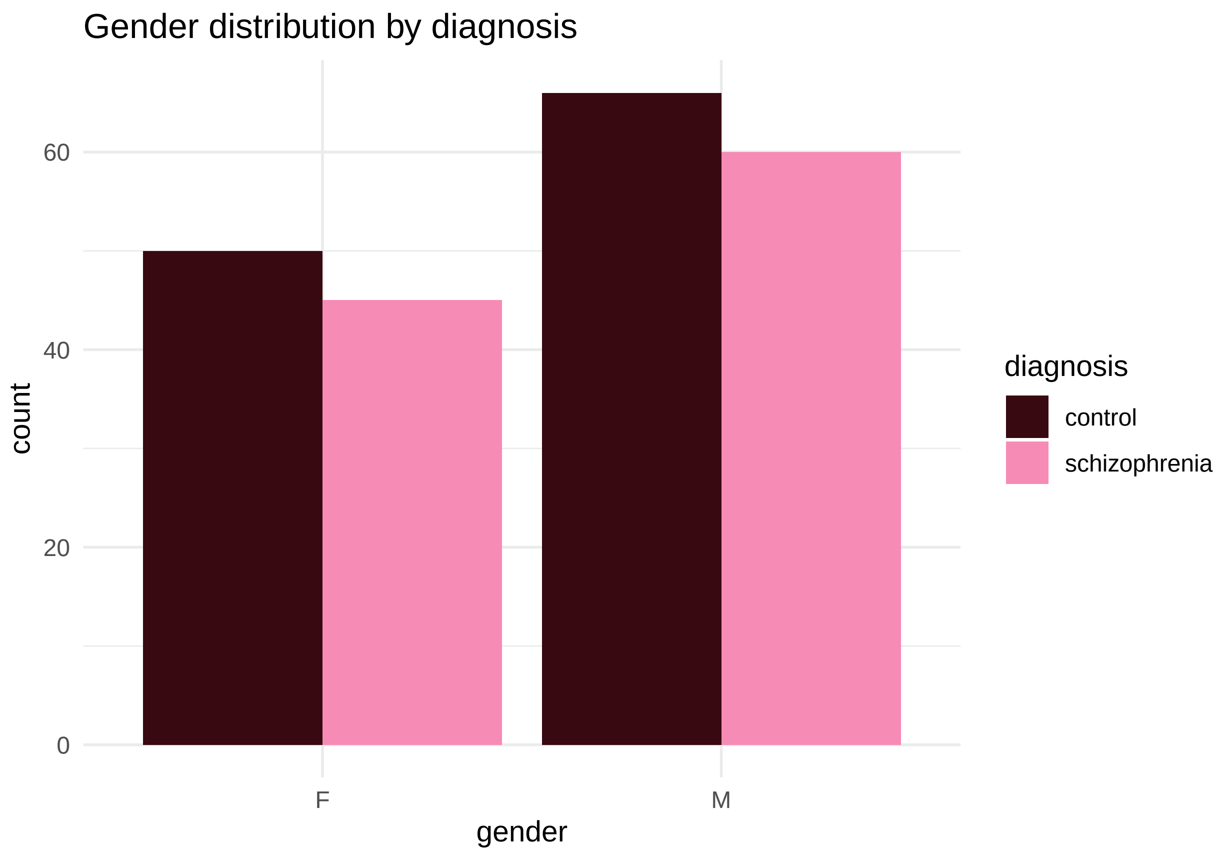
```r
ggplot(schizophrenia_and_articulation_pause_NA_fixed, aes(x = diagnosis, y = education,
                                                          fill = diagnosis)) +
  geom_bar(stat = "summary", fun = "mean", position = "dodge", width=0.2) +
  labs(title = "Average degree of education by Diagnosis", x = "Diagnosis",
       y = "Degree of education as measured by?")+
  scale_fill_manual(values = global_palette)
```



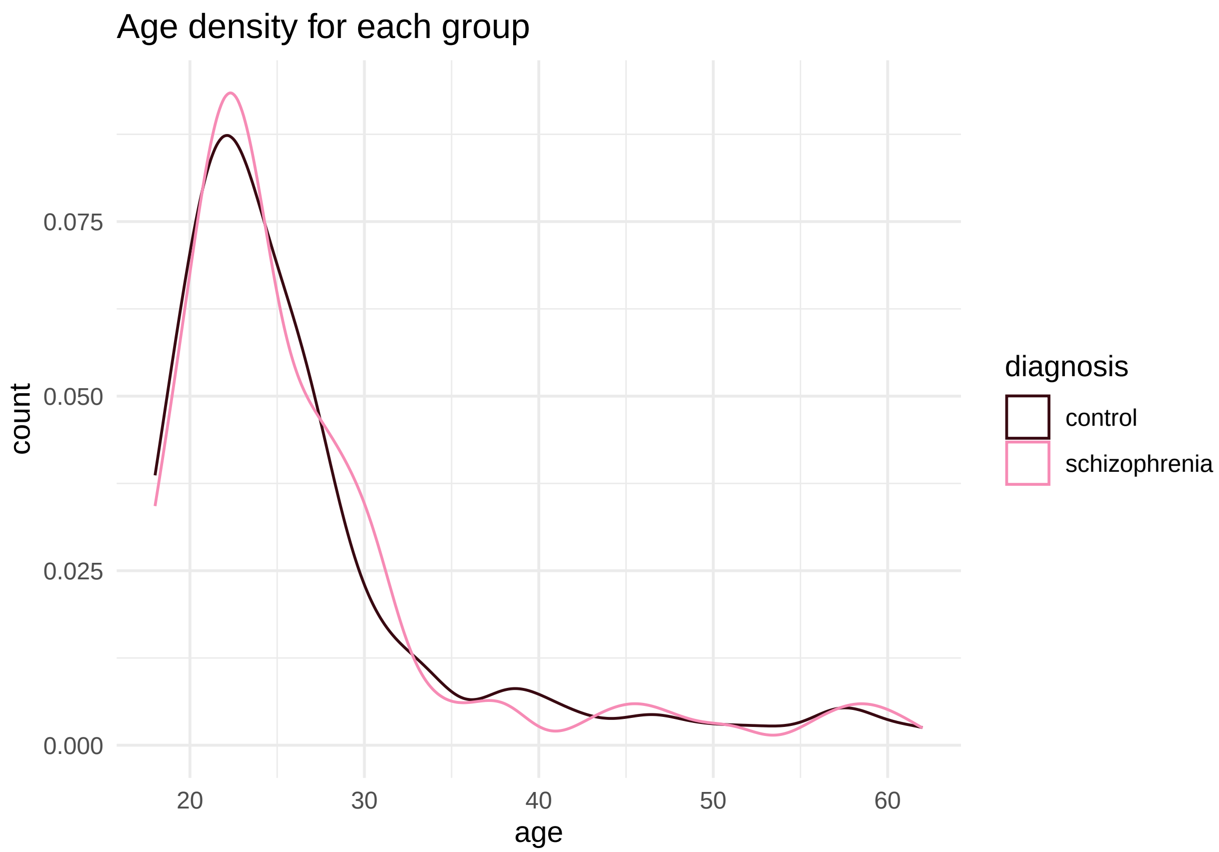
```r
# Calculate the number of unique values in the "id" column
unique_values_count <- schizophrenia_and_articulation %>%
  select(id, diagnosis)%>%
  distinct()%>%
  count()%>%
  print()
```

```
##     n
## 1 221
```

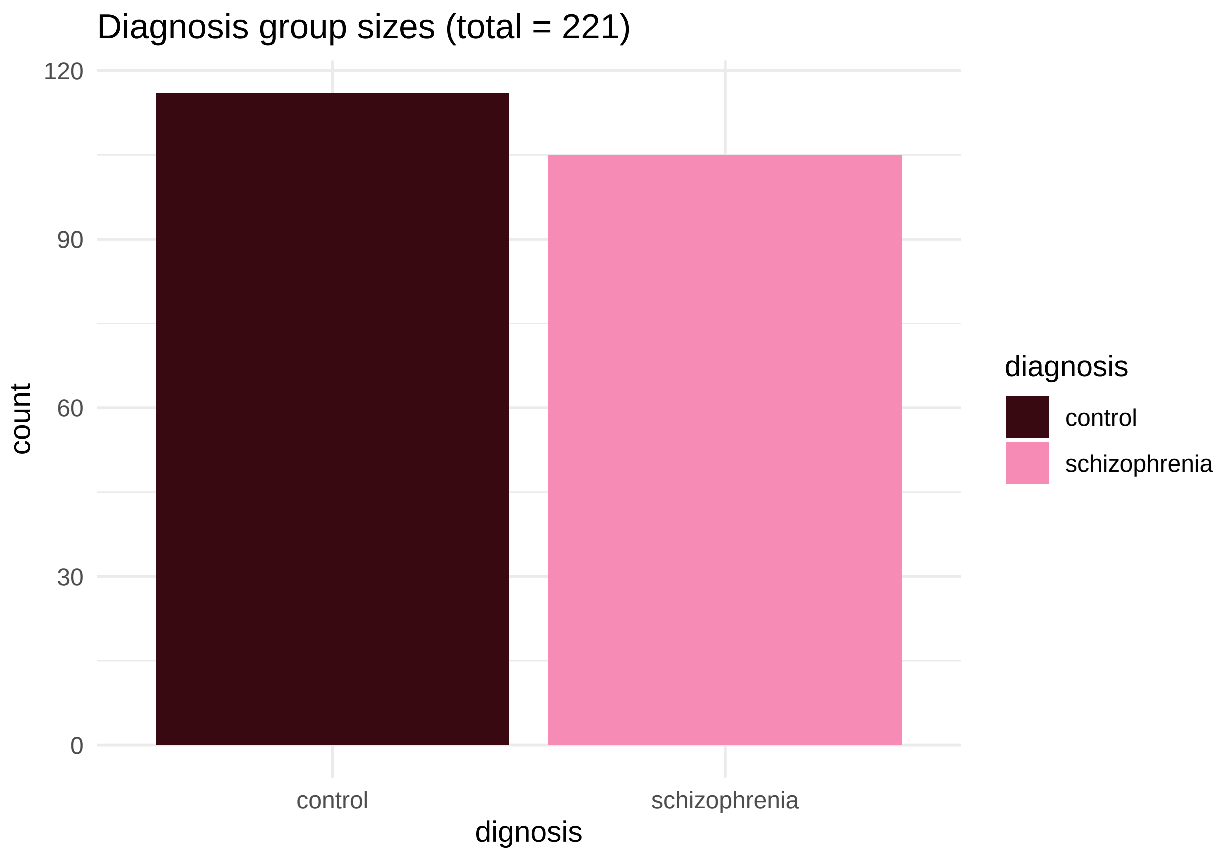
```r
#Gender graph
schizophrenia_and_articulation %>%
  select(c("id", "diagnosis", "gender")) %>%
  distinct() %>%
  ggplot(aes(x = gender, fill = diagnosis))+
  geom_bar(stat = "count", position=position_dodge())+
  labs(title = "Gender distribution by diagnosis")+
  scale_fill_manual(values = global_palette)
```



```
#Age density plot. NAs are removed here
schizophrenia_and_articulation %>%
  select(c("id", "diagnosis", "age")) %>%
  distinct() %>%
  ggplot(aes(x = age, color = diagnosis))+
  geom_density(na.rm = TRUE)+
  labs(title = "Age density for each group", x = "age", y = "count")+
  scale_color_manual(values = global_palette)
```

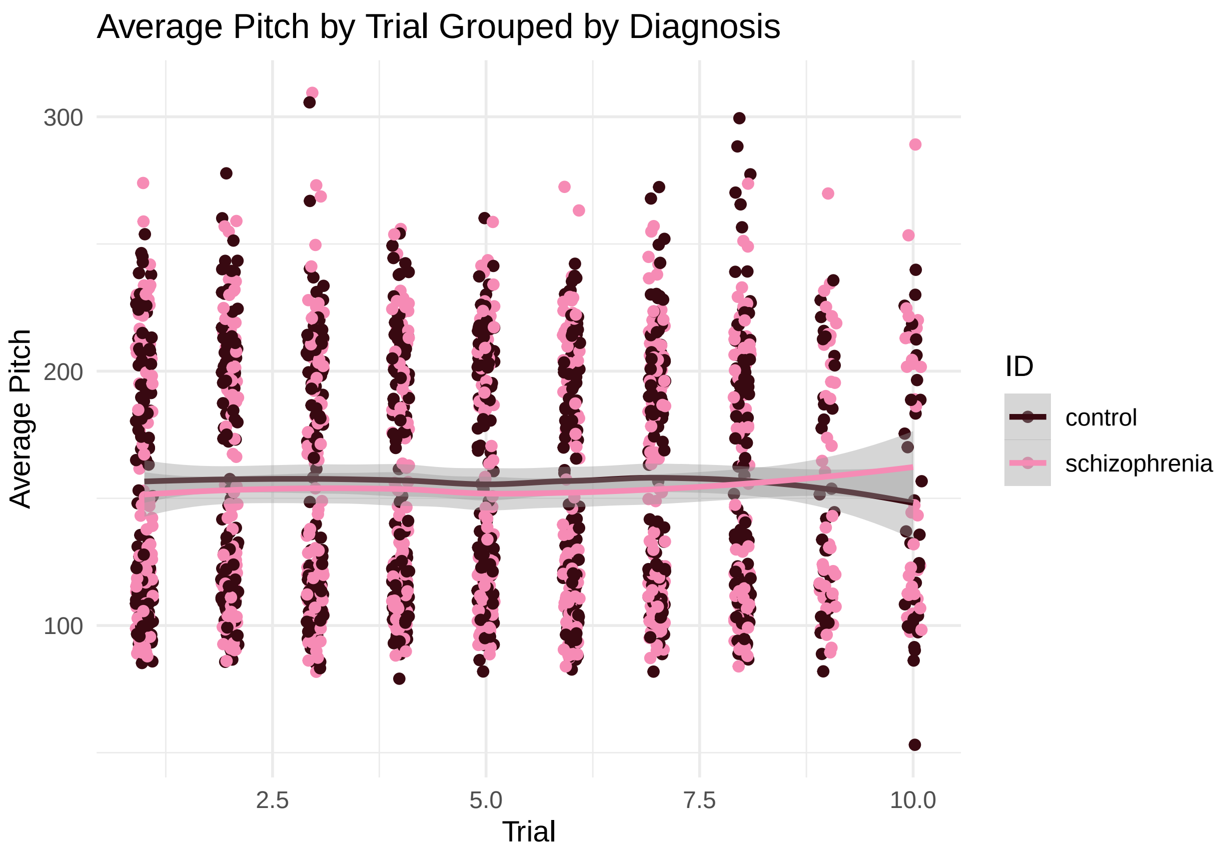


```
#Diagnosis bar graph
schizophrenia_and_articulation %>%
  select(c("id", "diagnosis")) %>%
  distinct() %>%
  ggplot(aes(x = diagnosis, fill = diagnosis))+
  geom_bar(stat = "count")+
  labs(title = "Diagnosis group sizes (total = 221) ", x = "dignosis", y = "count")+
  scale_fill_manual(values = global_palette)
```



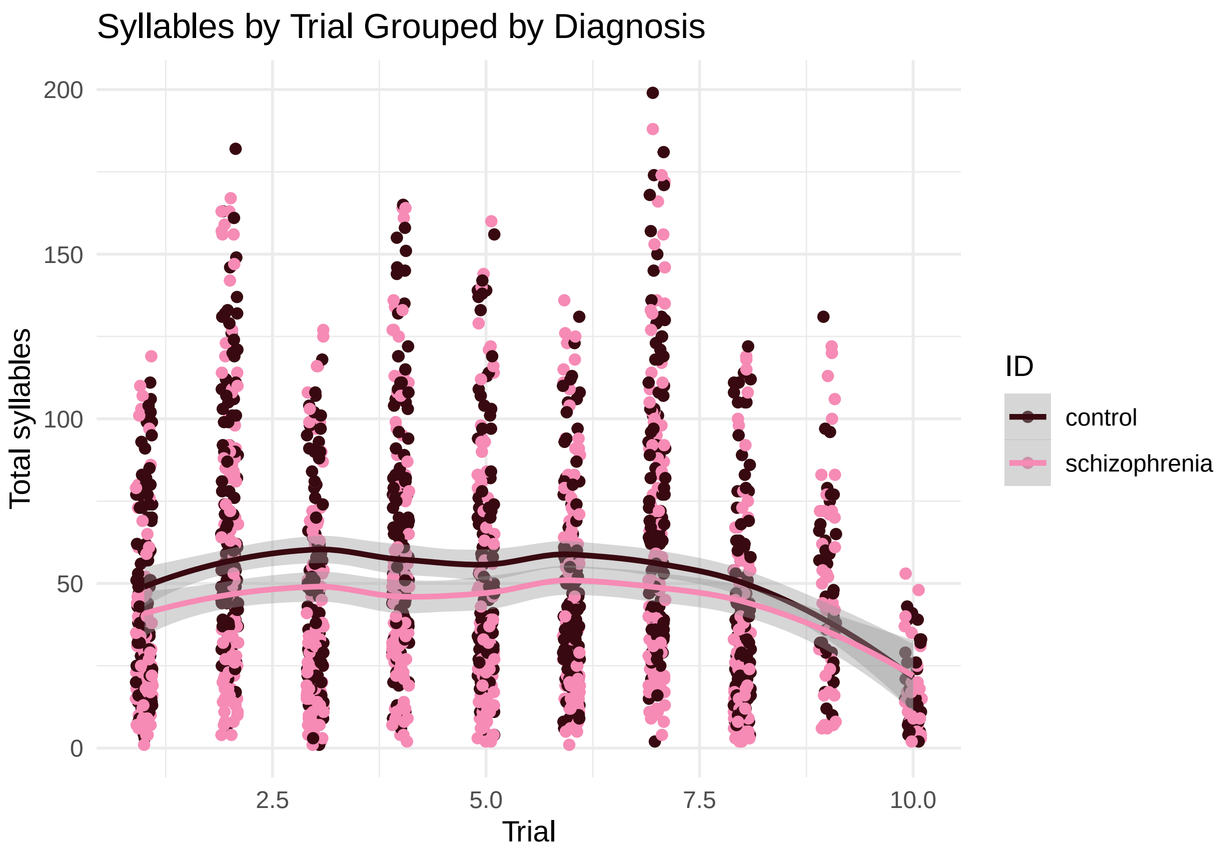
```
#Make a table with number of studies, number of participants in each
remove_outliers <- function(x) {
  qnt <- quantile(x, probs = c(0.25, 0.75), na.rm = TRUE)
  H <- 1.5 * IQR(x, na.rm = TRUE)
  x[x < (qnt[1] - H) | x > (qnt[2] + H)] <- NA
  x
}

#Pitch by trial grouped by diagnosis
schizophrenia_and_articulation %>%
  group_by(trial) %>%
  mutate(pitch_mean_no_outliers = remove_outliers(pitch_mean)) %>%
  ggplot(aes(x = trial, y = pitch_mean_no_outliers, color = diagnosis)) +
  geom_point(position = position_jitter(width = 0.1, height = 0)) +
  geom_smooth() +
  labs(title = "Average Pitch by Trial Grouped by Diagnosis", x = "Trial",
       y = "Average Pitch", color = "ID")+
    scale_color_manual(values = global_palette)
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



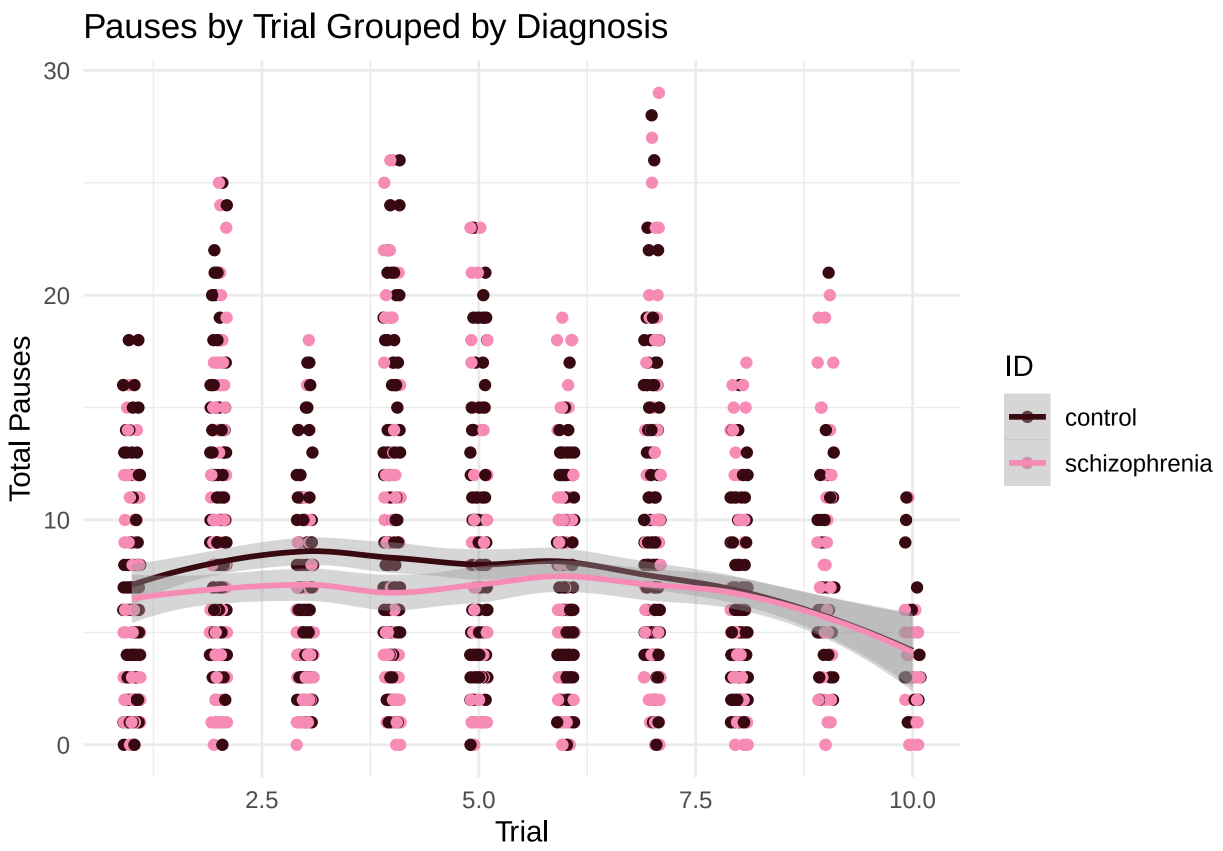
```r
#Total syllables by trial grouped by diagnosis
schizophrenia_and_articulation %>%
  group_by(trial) %>%
  mutate(total_syllables_no_outliers = remove_outliers(n_syllables)) %>%
  ggplot(aes(x = trial, y = total_syllables_no_outliers, color = diagnosis)) +
  geom_point(position = position_jitter(width = 0.1, height = 0)) +
  geom_smooth() +
  labs(title = "Syllables by Trial Grouped by Diagnosis", x = "Trial",
       y = "Total syllables", color = "ID")+
    scale_color_manual(values = global_palette)
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



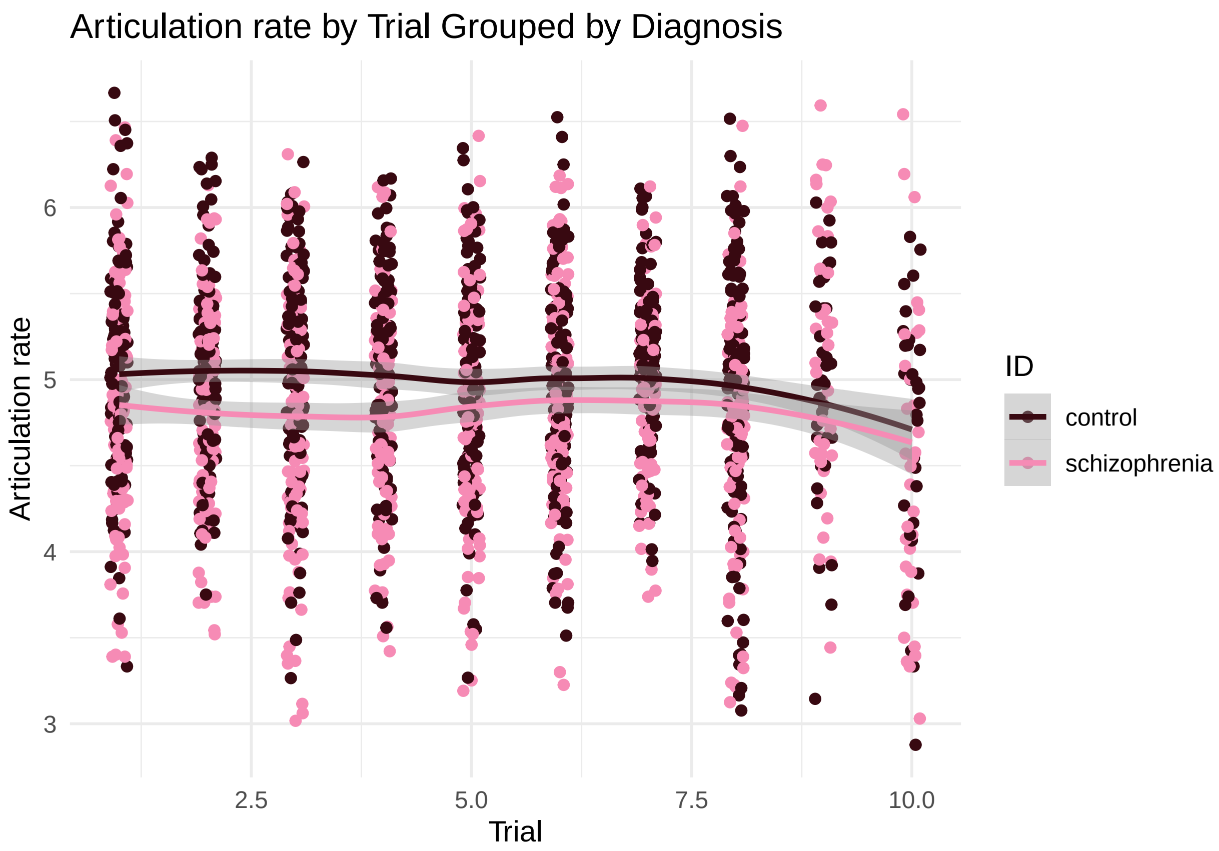
```
#Total pauses by trial grouped by diagnosis
schizophrenia_and_articulation %>%
  group_by(trial) %>%
  mutate(total_pauses_no_outliers = remove_outliers(n_pauses)) %>%
  ggplot(aes(x = trial, y = total_pauses_no_outliers, color = diagnosis)) +
  geom_point(position = position_jitter(width = 0.1, height = 0)) +
  geom_smooth() +
  labs(title = "Pauses by Trial Grouped by Diagnosis", x = "Trial",
       y = "Total Pauses", color = "ID")+
    scale_color_manual(values = global_palette)
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



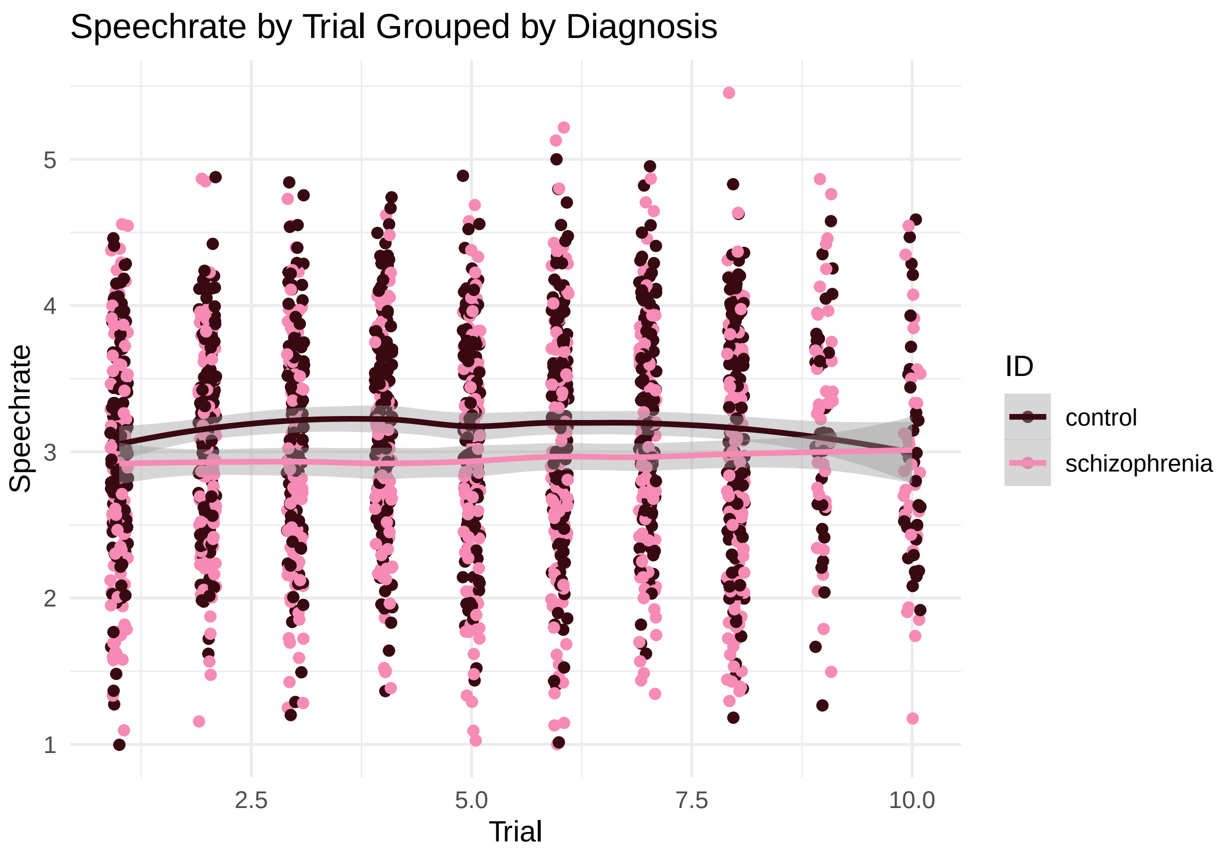
```
#Articulation rate by trial grouped by diagnosis
schizophrenia_and_articulation %>%
  group_by(trial) %>%
  mutate(articulation_rate_no_outliers = remove_outliers(articulation_rate)) %>%
  ggplot(aes(x = trial, y = articulation_rate_no_outliers, color = diagnosis)) +
  geom_point(position = position_jitter(width = 0.1, height = 0)) +
  geom_smooth() +
  labs(title = "Articulation rate by Trial Grouped by Diagnosis", x = "Trial",
       y = "Articulation rate", color = "ID")+
    scale_color_manual(values = global_palette)
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



```
#Speechrate by trial grouped by diagnosis
schizophrenia_and_articulation %>%
  group_by(trial) %>%
  mutate(speechrate_no_outliers = remove_outliers(speech_rate)) %>%
  ggplot(aes(x = trial, y = speechrate_no_outliers, color = diagnosis)) +
  geom_point(position = position_jitter(width = 0.1, height = 0)) +
  geom_smooth() +
  labs(title = "Speechrate by Trial Grouped by Diagnosis", x = "Trial",
       y = "Speechrate", color = "ID")+
    scale_color_manual(values = global_palette)
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Summary of descriptive statistics:

Both groups are nearly equally represented with 116 participants in the control group and 105 participants diagnosed with schizophrenia. Looking at the differences between linguistic markers of neurotypical and schizophrenic speech we can see following: Firstly, the average pitch deviated slightly, being lower for diagnosed participants over the first 8 trials. Slowly increasing, the average pitch of schizophrenic participants reaches the same average pitch as the control group in trial 8 and than increases further while the average pitch of the control group decreases in the last two trials. Secondly, the number of syllables used by both groups follows the same trend of increasing over the the first trials and decreasing from trial six on. Over all trials the number of syllables is however lower for schizophrenic participants than for the control group. The same tendencies can also be seen when looking at the number of pauses. Equally the articulation rate and speech rate of participants with schizophrenia are lower than those of control participants.

Predicting diagnosis using supervised learning procedures

We now want to know whether we can automatically diagnose schizophrenia from voice alone. To do this, we will proceed in incremental fashion. We will first start by building a simple random forest model, add an optimized version, and then add a third model based on an algorithm of your choice.

First phase: Random Forest Model (Amélie, Malene)

Splitting data Instead of only splitting in training and testing sets, we start by splitting our data in 3 parts: training, testing and validating. We choose to split in 70% training, 15% validation and 15% testing data; we want to make sure both control and diagnosis is represented proportionally in each split. Before proceeding we decide to impute NAs using the roughfix from randfomForest

```
#Removing NA's
schizophrenia_and_articulation <- schizophrenia_and_articulation %>%
  mutate(across(colnames(.), function(x){randomForest::na.roughfix(x)}))

set.seed(123)
simple_initial_split <- initial_validation_split(schizophrenia_and_articulation,
                                                 prop = c(0.7,0.15), strata = "diagnosis")
table(schizophrenia_and_articulation$diagnosis)
```

```
##
##       control schizophrenia
##           989           903
```

```
simple_training_data <- training(simple_initial_split)
table(simple_training_data$diagnosis)
```

```
##
##       control schizophrenia
##           692           632
```

```
simple_validation_data <- validation(simple_initial_split)
table(simple_validation_data$diagnosis)
```

```
##
##       control schizophrenia
##           148           135
```

```
simple_testing_data <- testing(simple_initial_split)
table(simple_testing_data$diagnosis)
```

```
##
##       control schizophrenia
##           149           136
```

To make sure that training and testing actually happen under best conditions, we check if the number of diagnosed participants and control participants is balanced and representative of the whole data set. As the sets appear to be representative of the total, we are content enough to continue.

Recipe We follow by making a recipe that our model can later follow. For our simple model we will choose diagnosis as our target variable, and all other predictors are included to the right hand side with "."; however we decide to exclude ID, as this is not a relevant predictor. This is done by assigning a role to the variable: the ID column shall be recognized as "ID", and not just another predictor variable, and therefore the model won't attempt to train on ID. Furthermore, the assignment is on whether we can predict from voice ALONE, so we will be removing any predictors not related to voice.

```
simple_recipe <- recipe(diagnosis ~ ., data = schizophrenia_and_articulation) %>%
  update_role(id, new_role = 'ID')%>%
  update_role(gender, new_role = "dont_use") %>%
  update_role(age, new_role = "dont_use") %>%
  update_role(education, new_role = "dont_use")
summary(simple_recipe)
```

```
## # A tibble: 24 x 4
##    variable            type      role      source
##    <chr>               <list>    <chr>     <chr>
##  1 study               <chr [2]> predictor original
##  2 id                  <chr [2]> ID        original
##  3 trial               <chr [2]> predictor original
##  4 duration            <chr [2]> predictor original
##  5 phonation_duration <chr [2]> predictor original
##  6 pause_duration      <chr [2]> predictor original
##  7 n_syllables         <chr [2]> predictor original
##  8 n_pauses            <chr [2]> predictor original
##  9 speech_rate         <chr [2]> predictor original
## 10 articulation_rate  <chr [2]> predictor original
## # i 14 more rows
```

Modelling Now we want to specify the type of random forest model we want to use. The engine we use is Random Forest, with the mode "classification". This tells the model that the target variable (diagnosis) is categorical, and we want to train the model to classify data to fit into either control or schizophrenia.

In this section we don't specify hyperparameters, to clarify the difference in tuning vs not tuning your model. The next section is about training hyperparameters

```
simple_rf_model <- rand_forest() %>%
  set_mode("classification") %>%
  set_engine("randomForest")
```

For convenience and be able to later call on aspects of the model, we put together model and recipe into a workflow. This will be useful in the second part of the assignment where we look at tuning and for calling on individual aspects of the model.

```
simple_model_workflow <- workflow() %>%
  add_recipe(simple_recipe) %>%
  add_model(simple_rf_model)

simple_model_workflow
```

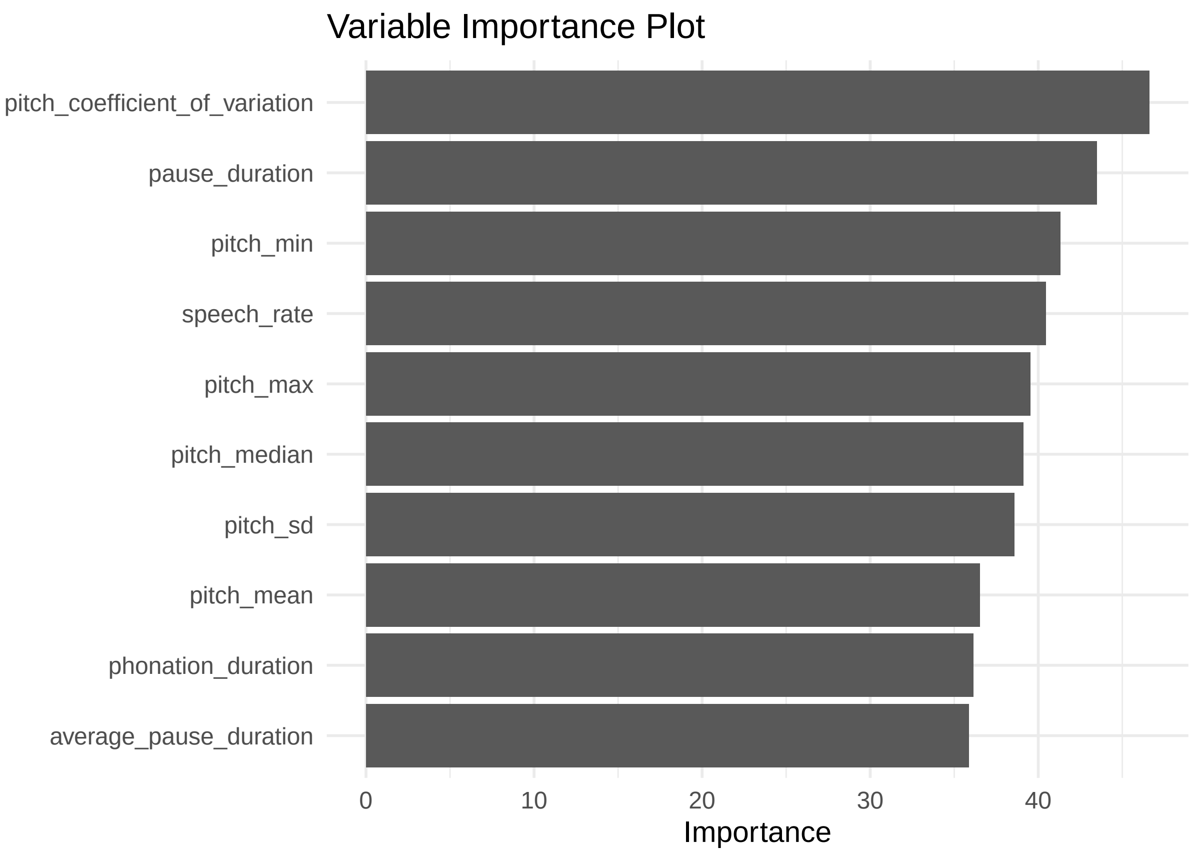
```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 0 Recipe Steps
##
## -- Model -----------------------------------------------------------------------
## Random Forest Model Specification (classification)
##
## Computational engine: randomForest
```

Predictions and evaluation of simple model We now wish to look at how this model performs before any tuning has been done to it. In order to get an idea of which variables are of the greatest importance to our model, we make a vip plot.

```
simple_training_data <- drop_na(simple_training_data)
simple_fit <- simple_model_workflow %>%
  fit(data = simple_training_data)

simple_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 0 Recipe Steps
##
## -- Model -----------------------------------------------------------------------
##
## Call:
##  randomForest(x = maybe_data_frame(x), y = y)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 34.14%
## Confusion matrix:
##               control schizophrenia class.error
## control           480           212   0.3063584
## schizophrenia     240           392   0.3797468
```

```
vip_plot <-
  simple_fit %>%
  extract_fit_parsnip() %>%
  vip::vip()

vip_plot + theme_minimal()+
  labs(title = "Variable Importance Plot")
```



According to this plot, the variation in pitch seem to be the predictor of greatest importance.

```
simple_outcome <- simple_testing_data %>%
  as_tibble() %>%
  mutate(
    predicted_class = predict(simple_fit, new_data = simple_testing_data),
    predicted_probabilities = predict(simple_fit, new_data = simple_testing_data,
                                      type = "prob")
  )

# Un-nest the predicted_probabilities column
simple_outcome <- simple_outcome %>%
  unnest_wider(c(predicted_probabilities, predicted_class))

simple_outcome[c('diagnosis', '.pred_class', '.pred_control', '.pred_schizophrenia')]
```

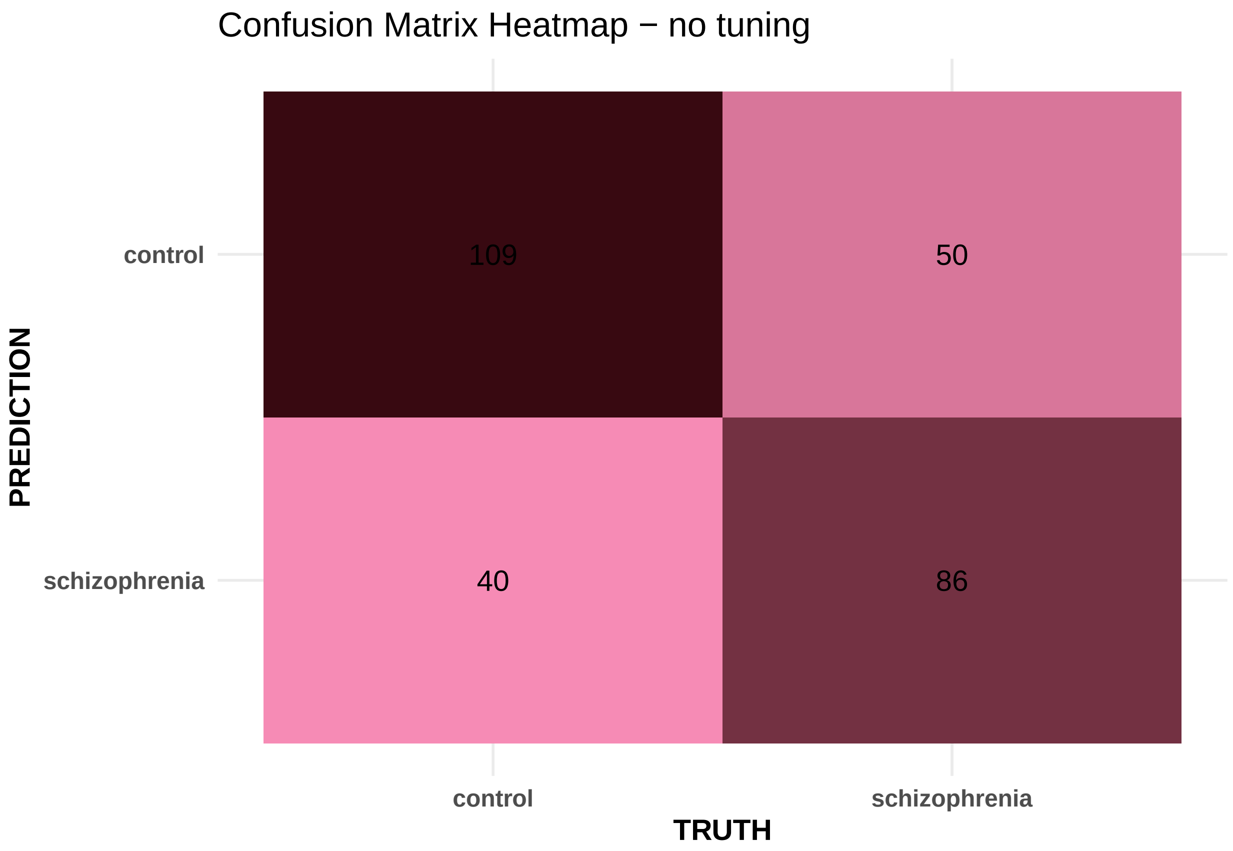
```
## # A tibble: 285 x 4
##    diagnosis     .pred_class   .pred_control .pred_schizophrenia
##    <fct>         <fct>                 <dbl>               <dbl>
##  1 control       schizophrenia         0.344               0.656
##  2 control       control               0.542               0.458
##  3 control       control               0.596               0.404
##  4 control       control               0.518               0.482
##  5 control       control               0.584               0.416
##  6 control       control               0.52                0.48
##  7 control       schizophrenia         0.414               0.586
##  8 control       schizophrenia         0.424               0.576
##  9 control       control               0.572               0.428
## 10 schizophrenia control               0.85                0.15
## # i 275 more rows
```

From a quick look at the table, it would appear the simple model (before any tuning) very often guesses incorrectly. We can get an overview over how incorrect it is using a confusion matrix.

```
conf_mat <-
  yardstick::conf_mat(simple_outcome,
                      truth = diagnosis,
                      estimate = .pred_class)

conf_plot <-
  autoplot(conf_mat, type = "heatmap") +
  scale_fill_gradient(low="#f68bb5",
                      high = "#380911") +
  theme(
    axis.title = element_text(face = 'bold'),
    axis.text = element_text(face = 'bold')) +
  labs(x = 'TRUTH', y = 'PREDICTION', title = "Confusion Matrix Heatmap - no tuning")
```

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```

```
conf_plot
```



The confusion matrix shoes 109 true positives, 50 false positives, 40 false negatives and 86 true negatives. For extra clarity, we also show the odds:

```r
# Given confusion matrix values
true_positives <- 109
false_positives <- 50
false_negatives <- 40
true_negatives <- 86

# Calculate odds of being right to wrong
odds_value <- (true_positives + true_negatives) / (false_positives + false_negatives)

# Print the expression and calculated odds value
odds_expression <- paste0("Odds of being right to wrong = ", true_positives + true_negatives,
                          " / ", false_positives + false_negatives)
cat(odds_expression, "\n")
```

```
## Odds of being right to wrong = 195 / 90
```

```r
cat("Calculated Odds Value: ", odds_value, "\n")
```

```
## Calculated Odds Value:  2.166667
```

This is not an ideal performance, we therefore move on to the second part to tune the model and hopefully get better results.

Second phase: Forest Engineering (Lilla, Katharina)

Setting up workflow for tuning In the first part we did not include any hyper parameters for tuning in our model, which is what we will do now.

```r
tuning_model <- rand_forest(
  mtry = tune(),
  trees = tune(),
  min_n = tune()
) %>%
  set_mode("classification") %>%
  set_engine("randomForest")

tuning_model_workflow <- workflow() %>%
  add_recipe(simple_recipe) %>%
  add_model(tuning_model)

tuning_model_workflow
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 0 Recipe Steps
```

```
##
## -- Model ----------------------------------------------------------------------
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = tune()
##   trees = tune()
##   min_n = tune()
##
## Computational engine: randomForest
```

Hyperparameters We start by "folding" our validation data into subsets that can then be used to tune. We also need to define the grid our hyperparameters are mapped onto. This is where defining a workflow comes in handy, as we can call on it here.

```
set.seed(234)
folds <- vfold_cv(simple_validation_data)
```

Having defined folds and grid, we begin tuning the model.

OBS: grid_entropy not possible with mtry parameter

```
doParallel::registerDoParallel() #for speeding up the process
tuned <- tune_grid(
  tuning_model_workflow,
  resamples = folds,
  grid = 20
)
```

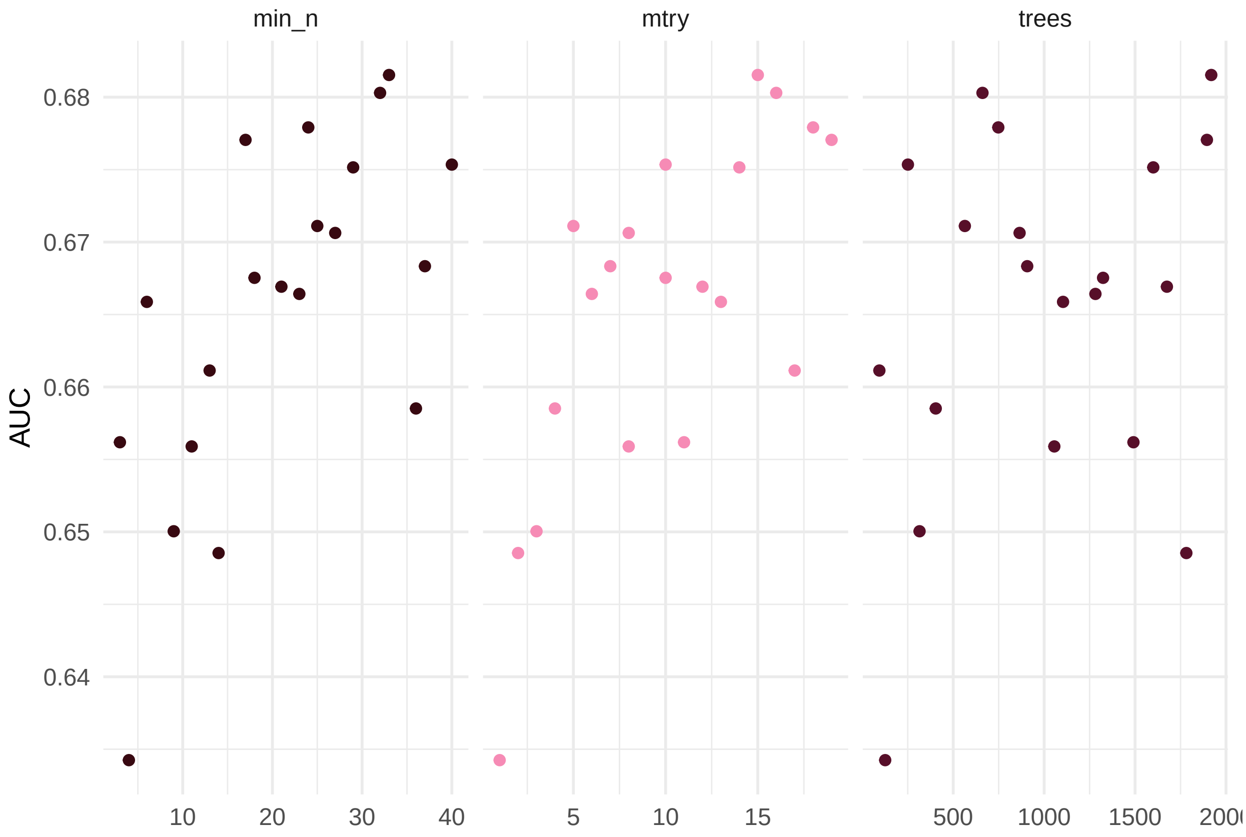
```
## i Creating pre-processing data to finalize unknown parameter: mtry
```

```
tuned_metrics <- tuned %>% collect_metrics()
print(tuned_metrics)
```

```
## # A tibble: 40 x 9
##     mtry trees min_n .metric  .estimator  mean     n std_err .config
##    <int> <int> <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1    16   661    32 accuracy binary     0.612    10  0.0350 Preprocessor1_Mode~
##  2    16   661    32 roc_auc  binary     0.680    10  0.0472 Preprocessor1_Mode~
##  3    14  1600    29 accuracy binary     0.627    10  0.0364 Preprocessor1_Mode~
##  4    14  1600    29 roc_auc  binary     0.675    10  0.0497 Preprocessor1_Mode~
##  5     5   564    25 accuracy binary     0.620    10  0.0363 Preprocessor1_Mode~
##  6     5   564    25 roc_auc  binary     0.671    10  0.0478 Preprocessor1_Mode~
##  7    10  1324    18 accuracy binary     0.634    10  0.0369 Preprocessor1_Mode~
##  8    10  1324    18 roc_auc  binary     0.668    10  0.0521 Preprocessor1_Mode~
##  9     8   865    27 accuracy binary     0.630    10  0.0399 Preprocessor1_Mode~
## 10     8   865    27 roc_auc  binary     0.671    10  0.0524 Preprocessor1_Mode~
## # i 30 more rows
```

In order to inspect the effect of the tuning we look at the area under the curve for the ROC. The higher the AUC, the more accurate the model.

```r
tuned %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  select(mean, mtry, trees, min_n) %>%
  pivot_longer(cols = c(mtry, trees, min_n),
               values_to = "value",
               names_to = "parameter") %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "AUC")+
  scale_color_manual(values = global_palette)
```



We manually set mtry and move forward with grid_entropy for tuning trees and min_n. The model seems to be best at values over 20 for mtry, in the interval 25-35 for min_n, and trees appear to be random

```r
manual_grid <- grid_regular(
  mtry(range = c(15, 25)),
  min_n(range = c(25, 35)),
  trees(range = c(500,850)),
  levels = 5
)

manual_grid
```

```
## # A tibble: 125 x 3
##     mtry min_n trees
##    <int> <int> <int>
##  1    15    25   500
##  2    17    25   500
##  3    20    25   500
##  4    22    25   500
##  5    25    25   500
##  6    15    27   500
##  7    17    27   500
##  8    20    27   500
##  9    22    27   500
## 10    25    27   500
## # i 115 more rows
```

Let’s now use this manual grid for another tune

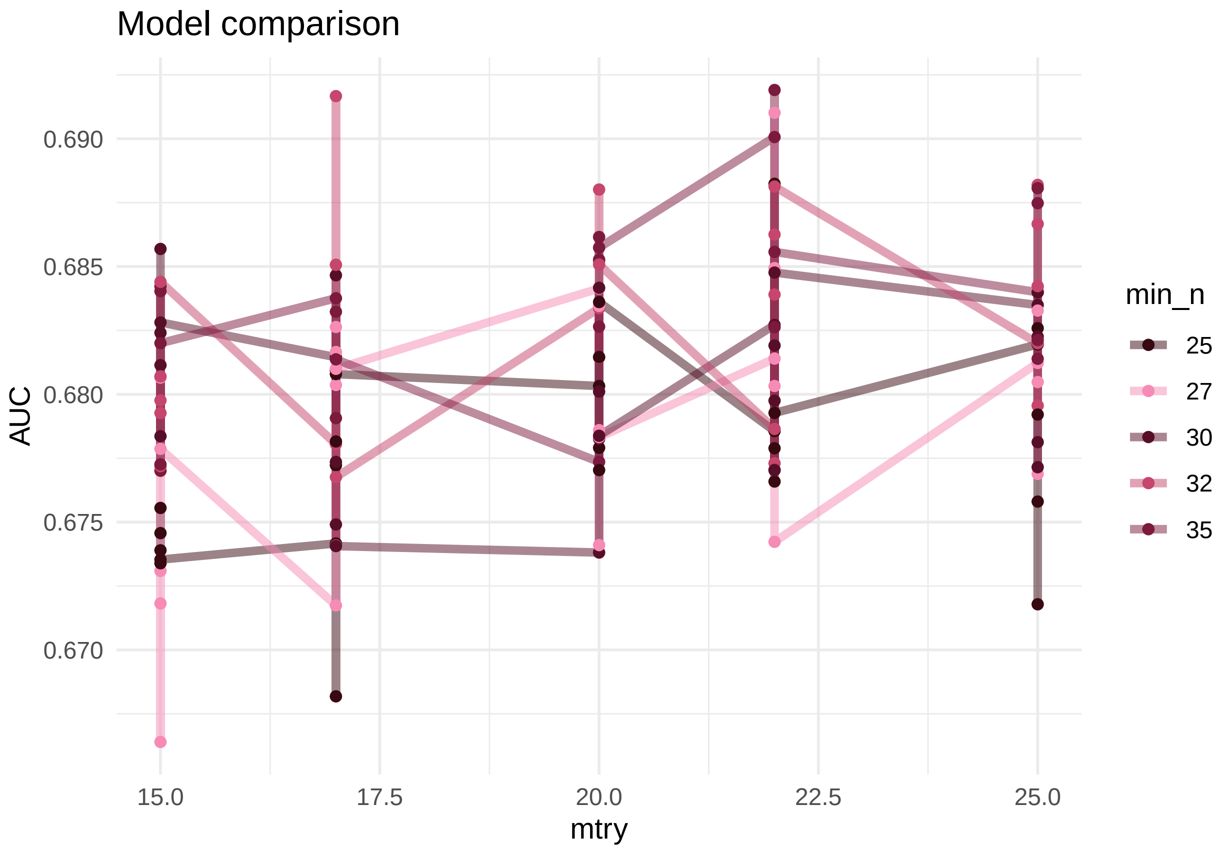
```
set.seed(456)
second_tune <- tune_grid(
  tuning_model_workflow,
  resamples = folds,
  grid = manual_grid
)

second_tune_metrics <- second_tune %>% collect_metrics()
print(second_tune_metrics)
```

```
## # A tibble: 250 x 9
##     mtry trees min_n .metric  .estimator  mean     n std_err .config
##    <int> <int> <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1    15   500    25 accuracy binary     0.630    10  0.0358 Preprocessor1_Mode~
##  2    15   500    25 roc_auc  binary     0.674    10  0.0495 Preprocessor1_Mode~
##  3    17   500    25 accuracy binary     0.619    10  0.0316 Preprocessor1_Mode~
##  4    17   500    25 roc_auc  binary     0.674    10  0.0487 Preprocessor1_Mode~
##  5    20   500    25 accuracy binary     0.634    10  0.0382 Preprocessor1_Mode~
##  6    20   500    25 roc_auc  binary     0.680    10  0.0495 Preprocessor1_Mode~
##  7    22   500    25 accuracy binary     0.637    10  0.0314 Preprocessor1_Mode~
##  8    22   500    25 roc_auc  binary     0.679    10  0.0487 Preprocessor1_Mode~
##  9    25   500    25 accuracy binary     0.637    10  0.0371 Preprocessor1_Mode~
## 10    25   500    25 roc_auc  binary     0.682    10  0.0484 Preprocessor1_Mode~
## # i 240 more rows
```

Let’s visualise the results of models

```
second_tune %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  mutate(min_n = factor(min_n)) %>%
  ggplot(aes(mtry, mean, color = min_n)) +
  geom_line(alpha = 0.5, size = 1.5) +
  geom_point() +
  labs(y = "AUC",title = "Model comparison")+
  scale_color_manual(values = global_palette)
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



The results are not very impressive, but now we select the best one to use for prediction

```
best_auc <- select_best(second_tune, "roc_auc")

final <- finalize_model(
  tuning_model,
  best_auc
)

final
```

```
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = 22
##   trees = 587
##   min_n = 35
##
## Computational engine: randomForest
```

In the following we will make a final workflow in order to make predictions.

```
final_workflow <- workflow() %>%
  add_recipe(simple_recipe) %>%
  add_model(final)

final_res <- final_workflow %>%
  last_fit(simple_initial_split)
```

```
## > A | warning: 22 columns were requested but there were 19 predictors in the data. 19 will be used.

## There were issues with some computations   A: x1There were issues with some computations   A: x1
```

```
final_res %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy binary         0.667 Preprocessor1_Model1
## 2 roc_auc  binary         0.714 Preprocessor1_Model1
```

In order to see if our tuning had any effect on the model, we will fit our data and make a confusion matrix in the same way as in part 1.

```
tuned_fit <- final_workflow %>%
  fit(data = simple_training_data)
```

```
## Warning: 22 columns were requested but there were 19 predictors in the data. 19
## will be used.
```

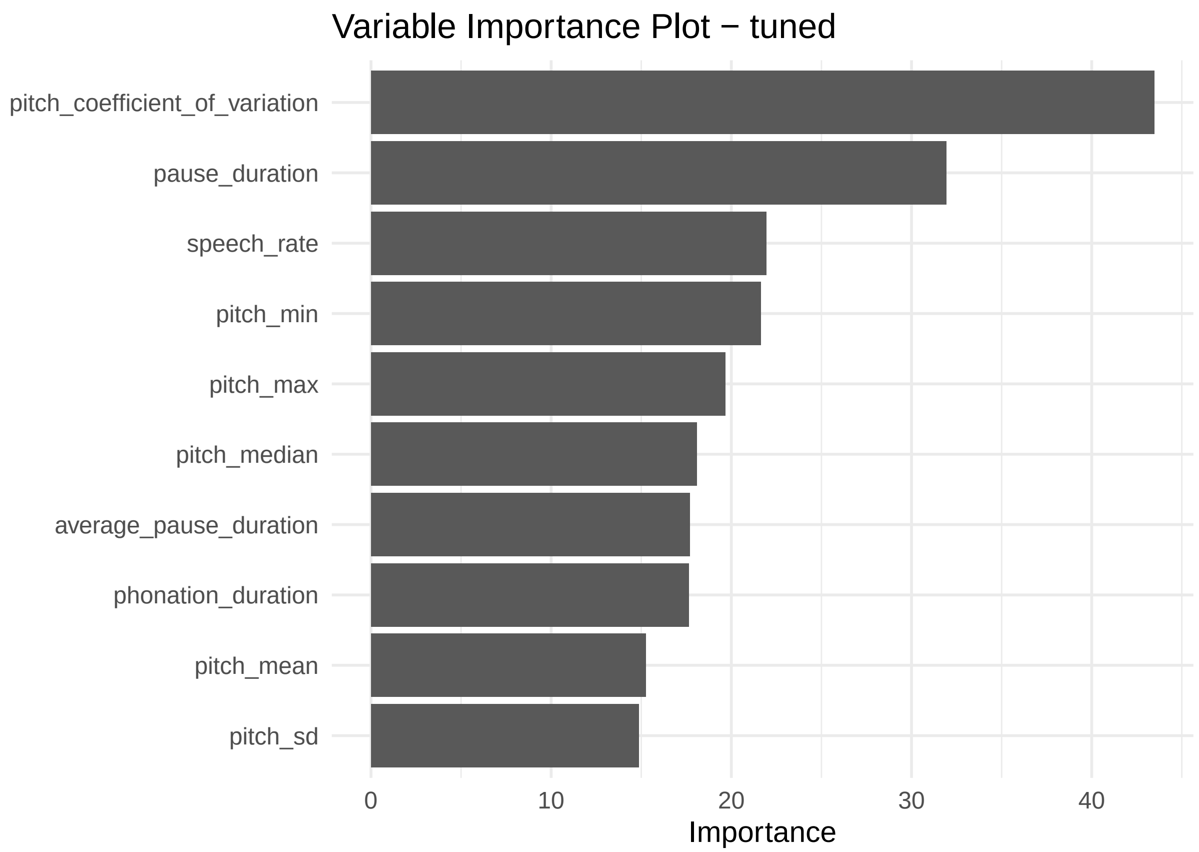
```
tuned_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 0 Recipe Steps
##
## -- Model -----------------------------------------------------------------------
##
## Call:
##  randomForest(x = maybe_data_frame(x), y = y, ntree = ~587L, mtry = min_cols(~22L,      x), nodesize
##                Type of random forest: classification
##                      Number of trees: 587
## No. of variables tried at each split: 19
##
##         OOB estimate of  error rate: 36.86%
## Confusion matrix:
##              control schizophrenia class.error
## control          442           250   0.3612717
## schizophrenia    238           394   0.3765823
```

```r
tuned_outcome <- simple_testing_data %>%
  as_tibble() %>%
  mutate(
    predicted_class = predict(tuned_fit, new_data = simple_testing_data),
    predicted_probabilities = predict(tuned_fit, new_data = simple_testing_data,
                                      type = "prob")
  )

# Unnest the predicted_probabilities column
tuned_outcome <- tuned_outcome %>%
  unnest_wider(c(predicted_probabilities, predicted_class))

tuned_outcome[c('diagnosis', '.pred_class', '.pred_control', '.pred_schizophrenia')]
```

```
## # A tibble: 285 x 4
##    diagnosis     .pred_class   .pred_control .pred_schizophrenia
##    <fct>         <fct>                 <dbl>               <dbl>
##  1 control       schizophrenia         0.274               0.726
##  2 control       control               0.586               0.414
##  3 control       control               0.581               0.419
##  4 control       control               0.578               0.422
##  5 control       control               0.550               0.450
##  6 control       schizophrenia         0.497               0.503
##  7 control       schizophrenia         0.370               0.630
##  8 control       schizophrenia         0.402               0.598
##  9 control       control               0.593               0.407
## 10 schizophrenia control               0.891               0.109
## # i 275 more rows
```

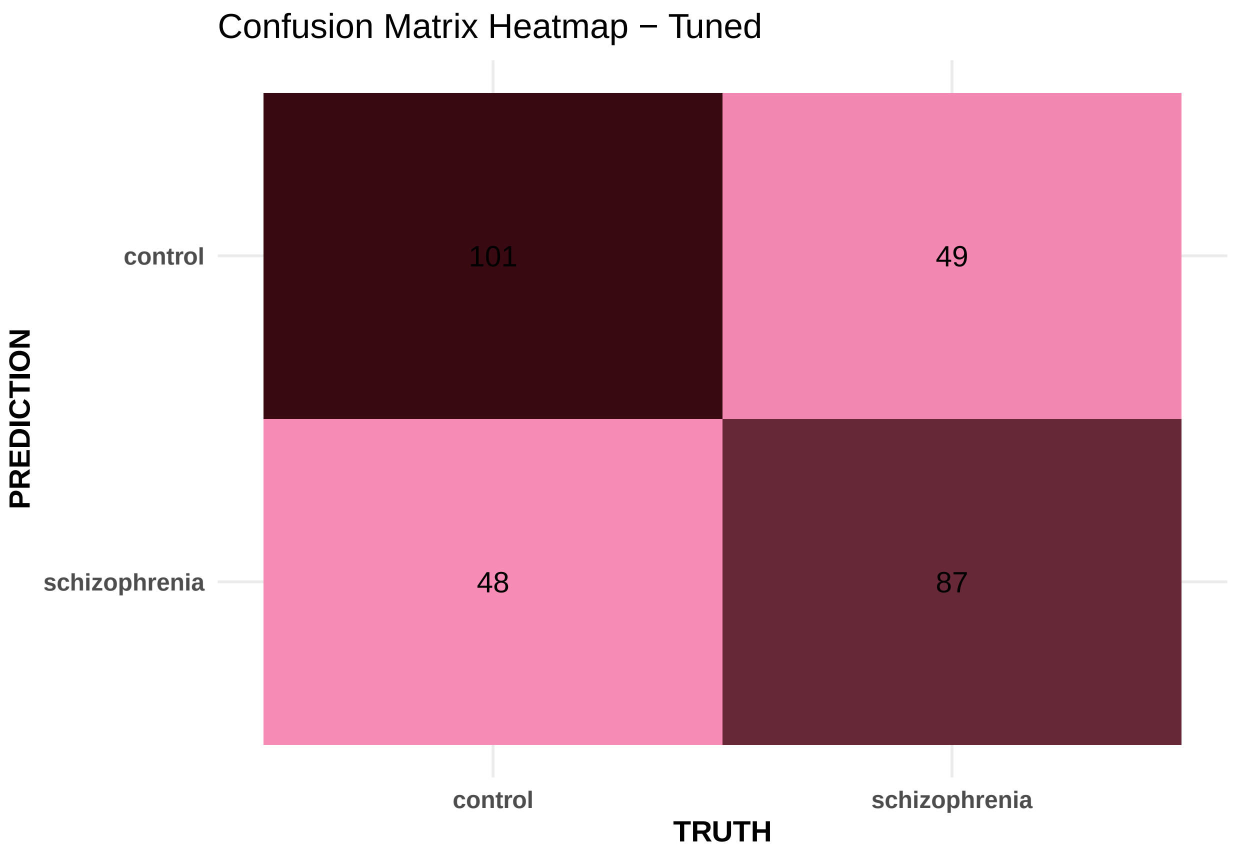
```r
vip_plot <-
  tuned_fit %>%
  extract_fit_parsnip() %>%
  vip::vip()

vip_plot + theme_minimal()+
  labs(title = "Variable Importance Plot - tuned")
```



```
tuned_conf_mat <-
  yardstick::conf_mat(tuned_outcome,
                      truth = diagnosis,
                      estimate = .pred_class)

tuned_conf_plot <-
  autoplot(tuned_conf_mat, type = "heatmap") +
  scale_fill_gradient(low="#f68bb5",
                      high = "#380911") +
  theme(
    axis.title = element_text(face = 'bold'),
    axis.text = element_text(face = 'bold')) +
  labs(x = 'TRUTH', y = 'PREDICTION', title = "Confusion Matrix Heatmap - Tuned")
```

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```

```
tuned_conf_plot
```



The confusion matrix shoes 101 true positives, 49 false positives, 48 false negatives and 87 true negatives. For extra clarity, we also show the odds:

```
# Given confusion matrix values
true_positives <- 101
false_positives <- 49
false_negatives <- 48
true_negatives <- 87

# Calculate odds of being right to wrong
odds_value <- (true_positives + true_negatives) / (false_positives + false_negatives)

# Print the expression and calculated odds value
odds_expression <- paste0("Odds of being right to wrong = ", true_positives + true_negatives,
                          " / ", false_positives + false_negatives)
cat(odds_expression, "\n")
```

```
## Odds of being right to wrong = 188 / 97
```

```
cat("Calculated Odds Value: ", odds_value, "\n")
```

```
## Calculated Odds Value:  1.938144
```

The manual adjusting of hyperparameters actually yielded a worse odds than the original model, though they are very close in accuracy. This could suggest the need for more variables in the model than just voice.

It is worth checking the other predictors such as gender and age, and adjust for these before ruling out voice as a potential indicator of schizoiphrenia. However, diagnosing via voice alone via. this random forest model does not seem a viable option.

Third phase: Another Algorithm (Malene, Lilla)

We have chosen an algorithm that supports classification and uses Bayes as its logic.

```r
#model
bayes_model <- naive_Bayes(
) %>%
  set_engine("naivebayes")

#workflow
extra_workflow <- workflow() %>%
  add_recipe(simple_recipe) %>%
  add_model(bayes_model)

extra_res <- extra_workflow %>%
  last_fit(simple_initial_split)

extra_res %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy binary         0.628 Preprocessor1_Model1
## 2 roc_auc  binary         0.658 Preprocessor1_Model1
```

```r
extra_fit <- extra_workflow %>%
  fit(data = simple_training_data)

extra_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: naive_Bayes()
##
## -- Preprocessor ----------------------------------------------------------------
## 0 Recipe Steps
##
## -- Model -----------------------------------------------------------------------
##
## ================================== Naive Bayes ==================================
##
##  Call:
## naive_bayes.default(x = maybe_data_frame(x), y = y, usekernel = TRUE)
##
## ---------------------------------------------------------------------------------
##
## Laplace smoothing: 0
```

```
##
## ---------------------------------------------------------------------------------
##
##  A priori probabilities:
##
##       control schizophrenia
##     0.5226586     0.4773414
##
## ---------------------------------------------------------------------------------
##
##  Tables:
##
## ---------------------------------------------------------------------------------
##  ::: study::control (KDE)
## ---------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (692 obs.);  Bandwidth 'bw' = 0.2906
##
##        x                 y
##  Min.   :0.1283   Min.   :0.003627
##  1st Qu.:1.3142   1st Qu.:0.142498
##  Median :2.5000   Median :0.202453
##  Mean   :2.5000   Mean   :0.210452
##  3rd Qu.:3.6858   3rd Qu.:0.279910
##  Max.   :4.8717   Max.   :0.504562
##
## ---------------------------------------------------------------------------------
##  ::: study::schizophrenia (KDE)
## ---------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (632 obs.);  Bandwidth 'bw' = 0.3011
##
##        x                  y
##  Min.   :0.09667   Min.   :0.003914
##  1st Qu.:1.29833   1st Qu.:0.145252
##
## ...
## and 147 more lines.
```

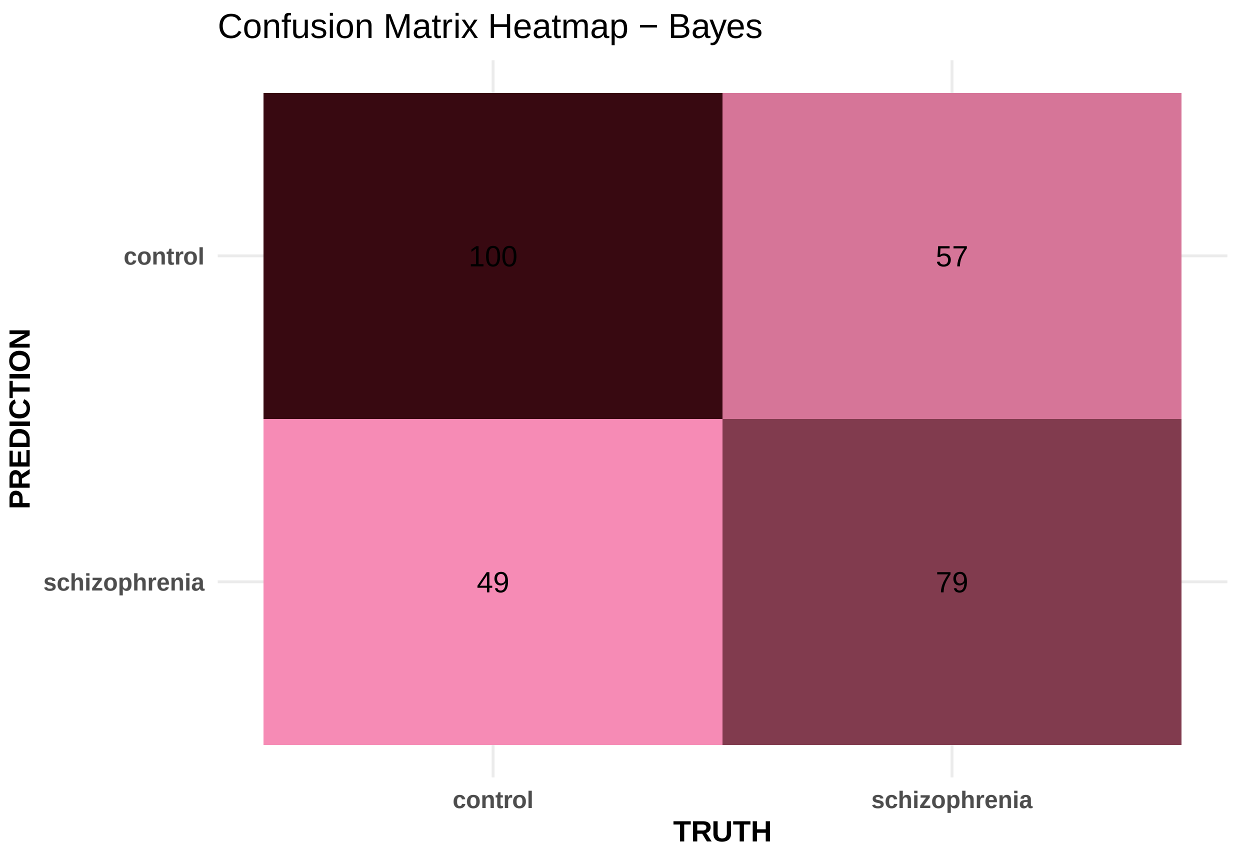
```r
extra_outcome <- simple_testing_data %>%
  as_tibble() %>%
  mutate(
    predicted_class = predict(extra_fit, new_data = simple_testing_data),
    predicted_probabilities = predict(extra_fit, new_data = simple_testing_data,
                                      type = "prob")
  )

extra_outcome <- extra_outcome %>%
  unnest_wider(c(predicted_probabilities, predicted_class))
extra_outcome[c('diagnosis', '.pred_class', '.pred_control', '.pred_schizophrenia')]
```

```
## # A tibble: 285 x 4
##    diagnosis     .pred_class   .pred_control .pred_schizophrenia
##    <fct>         <fct>                 <dbl>               <dbl>
##  1 control       schizophrenia         0.155              0.845
##  2 control       schizophrenia         0.482              0.518
##  3 control       schizophrenia         0.133              0.867
##  4 control       schizophrenia         0.454              0.546
##  5 control       control               0.854              0.146
##  6 control       schizophrenia         0.346              0.654
##  7 control       schizophrenia         0.470              0.530
##  8 control       control               0.769              0.231
##  9 control       control               0.900              0.0997
## 10 schizophrenia control               0.990              0.0101
## # i 275 more rows
```

```r
extra_conf_mat <-
  yardstick::conf_mat(extra_outcome,
                      truth = diagnosis,
                      estimate = .pred_class)

extra_conf_plot <-
  autoplot(extra_conf_mat, type = "heatmap") +
  scale_fill_gradient(low="#f68bb5",
                      high = "#380911") +
  theme(
    axis.title = element_text(face = 'bold'),
    axis.text = element_text(face = 'bold')) +
  labs(x = 'TRUTH', y = 'PREDICTION', title = "Confusion Matrix Heatmap - Bayes")
```

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```

```r
extra_conf_plot
```



```r
# Given confusion matrix values
true_positives <- 100
false_positives <- 57
false_negatives <- 49
true_negatives <- 79

# Calculate odds of being right to wrong
odds_value <- (true_positives + true_negatives) / (false_positives + false_negatives)

# Print the expression and calculated odds value
odds_expression <- paste0("Odds of being right to wrong = ", true_positives + true_negatives,
                          " / ", false_positives + false_negatives)
cat(odds_expression, "\n")
```

```
## Odds of being right to wrong = 179 / 106
```

```r
cat("Calculated Odds Value: ", odds_value, "\n")
```

```
## Calculated Odds Value:  1.688679
```

This model provides even worse results than the random forest models.

Extra: Including extra predictors (Katharina)

Due to the suboptimal models from only looking at voice, we have decided to adjust for age, gender and education. We will also try the ranger engine, and do the whole process again:

```
#new recipe
extra_predictors_recipe <- recipe(diagnosis ~ ., data = schizophrenia_and_articulation) %>%
  update_role(id, new_role = 'ID')
summary(extra_predictors_recipe)
```

```
## # A tibble: 24 x 4
##    variable           type      role      source
##    <chr>              <list>    <chr>     <chr>
##  1 study              <chr [2]> predictor original
##  2 id                 <chr [2]> ID        original
##  3 trial              <chr [2]> predictor original
##  4 duration           <chr [2]> predictor original
##  5 phonation_duration <chr [2]> predictor original
##  6 pause_duration     <chr [2]> predictor original
##  7 n_syllables        <chr [2]> predictor original
##  8 n_pauses           <chr [2]> predictor original
##  9 speech_rate        <chr [2]> predictor original
## 10 articulation_rate  <chr [2]> predictor original
## # i 14 more rows
```

```
#Setting up model
extra_predictors_model <- rand_forest(
  mtry = tune(),
  trees = 1000,
  min_n = tune()
) %>%
  set_mode("classification") %>%
  set_engine("ranger")


#defining workflow
extra_predictors_workflow <- workflow() %>%
  add_recipe(extra_predictors_recipe) %>%
  add_model(extra_predictors_model)

extra_predictors_workflow
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 0 Recipe Steps
##
## -- Model -----------------------------------------------------------------------
## Random Forest Model Specification (classification)
##
## Main Arguments:
```

```
##   mtry = tune()
##   trees = 1000
##   min_n = tune()
##
## Computational engine: ranger
```

We use the same folds we defined earlier to make our first tune:

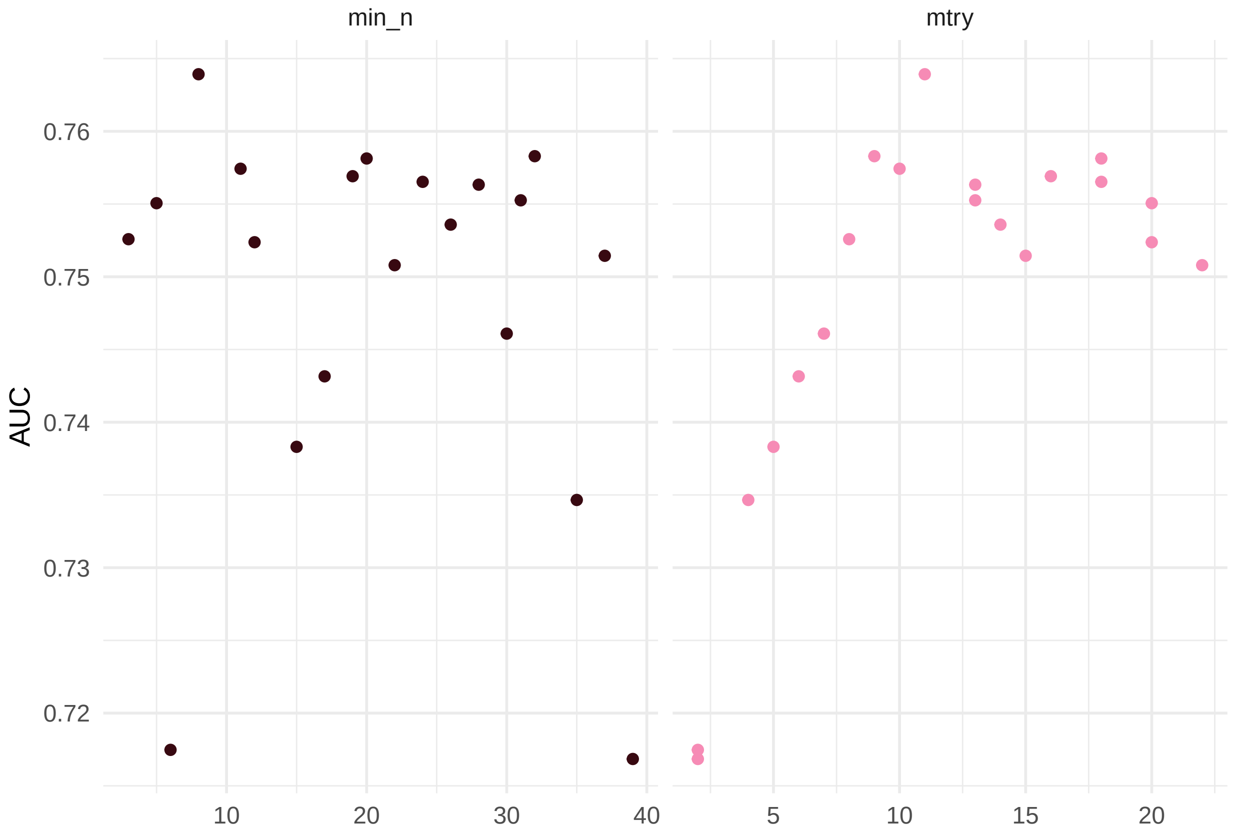
```
doParallel::registerDoParallel()

set.seed(345)
extra_predictors_tune <- tune_grid(
  extra_predictors_workflow,
  resamples = folds,
  grid = 20
)
```

```
## i Creating pre-processing data to finalize unknown parameter: mtry
```

```
extra_predictors_tune
```

```
## # Tuning results
## # 10-fold cross-validation
## # A tibble: 10 x 4
##    splits           id     .metrics          .notes
##    <list>           <chr>  <list>            <list>
##  1 <split [254/29]> Fold01 <tibble [40 x 6]> <tibble [0 x 3]>
##  2 <split [254/29]> Fold02 <tibble [40 x 6]> <tibble [0 x 3]>
##  3 <split [254/29]> Fold03 <tibble [40 x 6]> <tibble [0 x 3]>
##  4 <split [255/28]> Fold04 <tibble [40 x 6]> <tibble [0 x 3]>
##  5 <split [255/28]> Fold05 <tibble [40 x 6]> <tibble [0 x 3]>
##  6 <split [255/28]> Fold06 <tibble [40 x 6]> <tibble [0 x 3]>
##  7 <split [255/28]> Fold07 <tibble [40 x 6]> <tibble [0 x 3]>
##  8 <split [255/28]> Fold08 <tibble [40 x 6]> <tibble [0 x 3]>
##  9 <split [255/28]> Fold09 <tibble [40 x 6]> <tibble [0 x 3]>
## 10 <split [255/28]> Fold10 <tibble [40 x 6]> <tibble [0 x 3]>
```

```
extra_predictors_tune %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  select(mean, min_n, mtry) %>%
  pivot_longer(min_n:mtry,
    values_to = "value",
    names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "AUC")+
    scale_color_manual(values = global_palette)
```



This plot indicates it would be beneficial to adjust mtry to values above 12, and the range 22-28 seems good for min_n.

```
extra_predictors_grid <- grid_regular(
  mtry(range = c(12, 22)),
  min_n(range = c(22, 28)),
  levels = 5
)

extra_predictors_grid
```

```
## # A tibble: 25 x 2
##     mtry min_n
##    <int> <int>
##  1    12    22
##  2    14    22
##  3    17    22
##  4    19    22
##  5    22    22
##  6    12    23
##  7    14    23
##  8    17    23
##  9    19    23
## 10    22    23
## # i 15 more rows
```

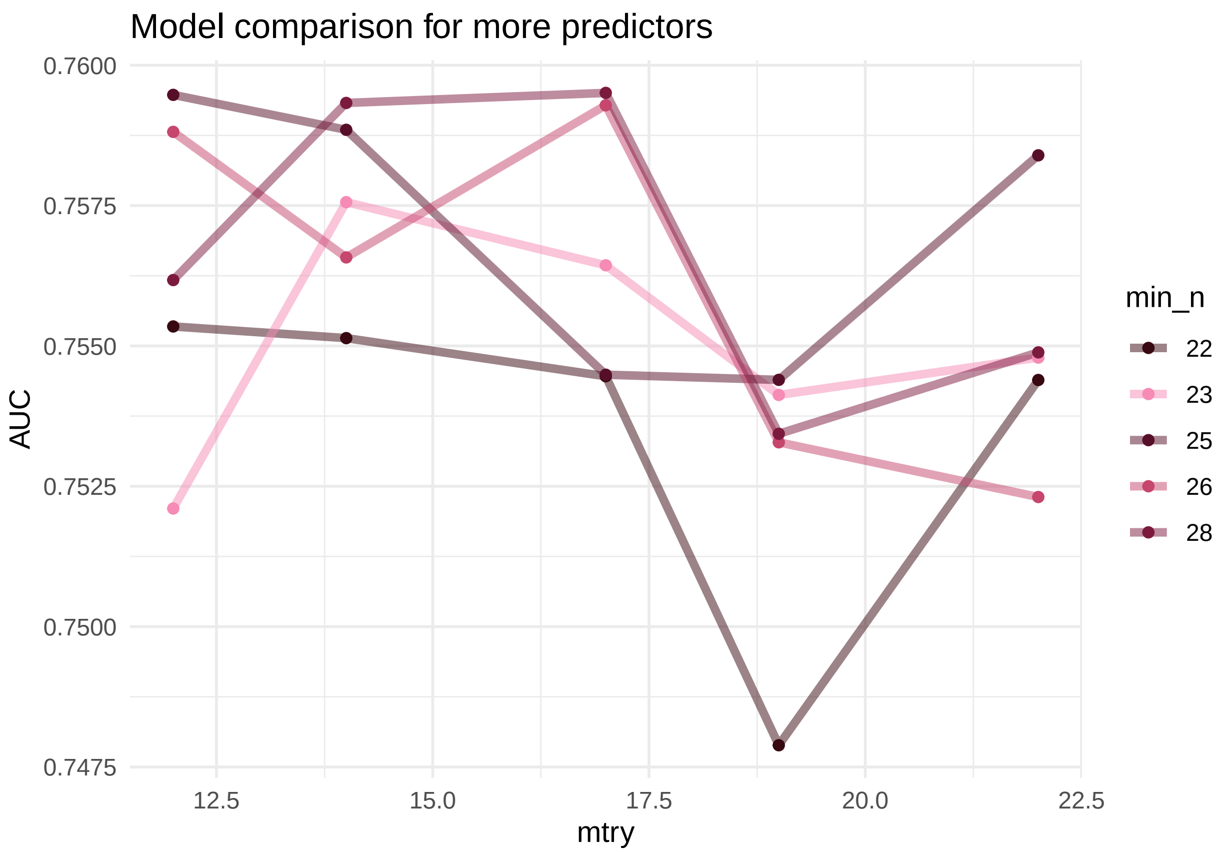
Now we apply this manual grid to a new tune

```
set.seed(123)
extra_predictors_manual_tune <- tune_grid(
  extra_predictors_workflow,
  resamples = folds,
  grid = extra_predictors_grid
)

extra_predictors_manual_tune
```

```
## # Tuning results
## # 10-fold cross-validation
## # A tibble: 10 x 4
##    splits           id     .metrics          .notes
##    <list>           <chr>  <list>            <list>
##  1 <split [254/29]> Fold01 <tibble [50 x 6]> <tibble [0 x 3]>
##  2 <split [254/29]> Fold02 <tibble [50 x 6]> <tibble [0 x 3]>
##  3 <split [254/29]> Fold03 <tibble [50 x 6]> <tibble [0 x 3]>
##  4 <split [255/28]> Fold04 <tibble [50 x 6]> <tibble [0 x 3]>
##  5 <split [255/28]> Fold05 <tibble [50 x 6]> <tibble [0 x 3]>
##  6 <split [255/28]> Fold06 <tibble [50 x 6]> <tibble [0 x 3]>
##  7 <split [255/28]> Fold07 <tibble [50 x 6]> <tibble [0 x 3]>
##  8 <split [255/28]> Fold08 <tibble [50 x 6]> <tibble [0 x 3]>
##  9 <split [255/28]> Fold09 <tibble [50 x 6]> <tibble [0 x 3]>
## 10 <split [255/28]> Fold10 <tibble [50 x 6]> <tibble [0 x 3]>
```

Now we can select the best model:

```
#Visualisation
extra_predictors_manual_tune %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  mutate(min_n = factor(min_n)) %>%
  ggplot(aes(mtry, mean, color = min_n)) +
  geom_line(alpha = 0.5, size = 1.5) +
  geom_point() +
  labs(y = "AUC", title = "Model comparison for more predictors")+
    scale_color_manual(values = global_palette)
```

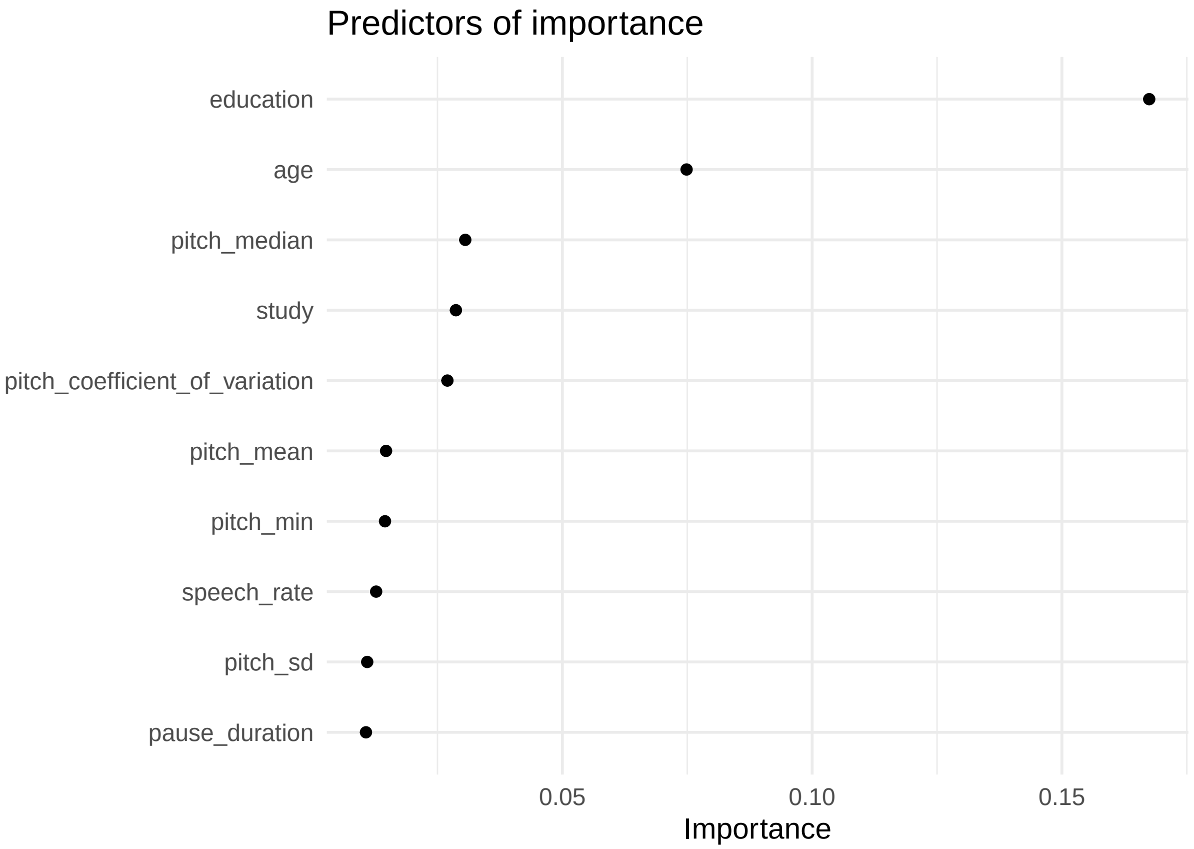


```
#selection
extra_predictors_best <- select_best(extra_predictors_manual_tune, "roc_auc")

extra_predictors_final <- finalize_model(
  extra_predictors_model,
  extra_predictors_best
)

extra_predictors_final
```

```
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = 17
##   trees = 1000
##   min_n = 28
##
## Computational engine: ranger
```

Having included all these extra predictors, lets look at which predictors are of greatest importance:

```
extra_predictors_prep <- prep(extra_predictors_recipe)

extra_predictors_final %>%
  set_engine("ranger", importance = "permutation") %>%
  fit(diagnosis ~ .,
    data = juice(extra_predictors_prep) %>% select(-id)
  ) %>%
  vip(geom = "point")+
  labs(title = "Predictors of importance")+
    scale_color_manual(values = global_palette)
```



According to this, education is by far the biggest indicator for change in speech patterns, followed by age. Now we fit a last time before making confusion matrix and comparing the odds.

```
final_extra_predictors_workflow <- workflow() %>%
  add_recipe(extra_predictors_recipe) %>%
  add_model(extra_predictors_final)

final_extra_predictors_res <- final_extra_predictors_workflow %>%
  last_fit(simple_initial_split)

final_extra_predictors_res %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy binary         0.860 Preprocessor1_Model1
## 2 roc_auc  binary         0.908 Preprocessor1_Model1
```

```r
final_extra_predictors_fit <- final_extra_predictors_workflow %>%
  fit(data = simple_training_data)

final_extra_predictors_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 0 Recipe Steps
##
## -- Model -----------------------------------------------------------------------
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~17L,      x), num.trees = ~1000, mi
##
## Type:                             Probability estimation
## Number of trees:                  1000
## Sample size:                      1324
## Number of independent variables:  22
## Mtry:                             17
## Target node size:                 28
## Variable importance mode:         none
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.1315751
```

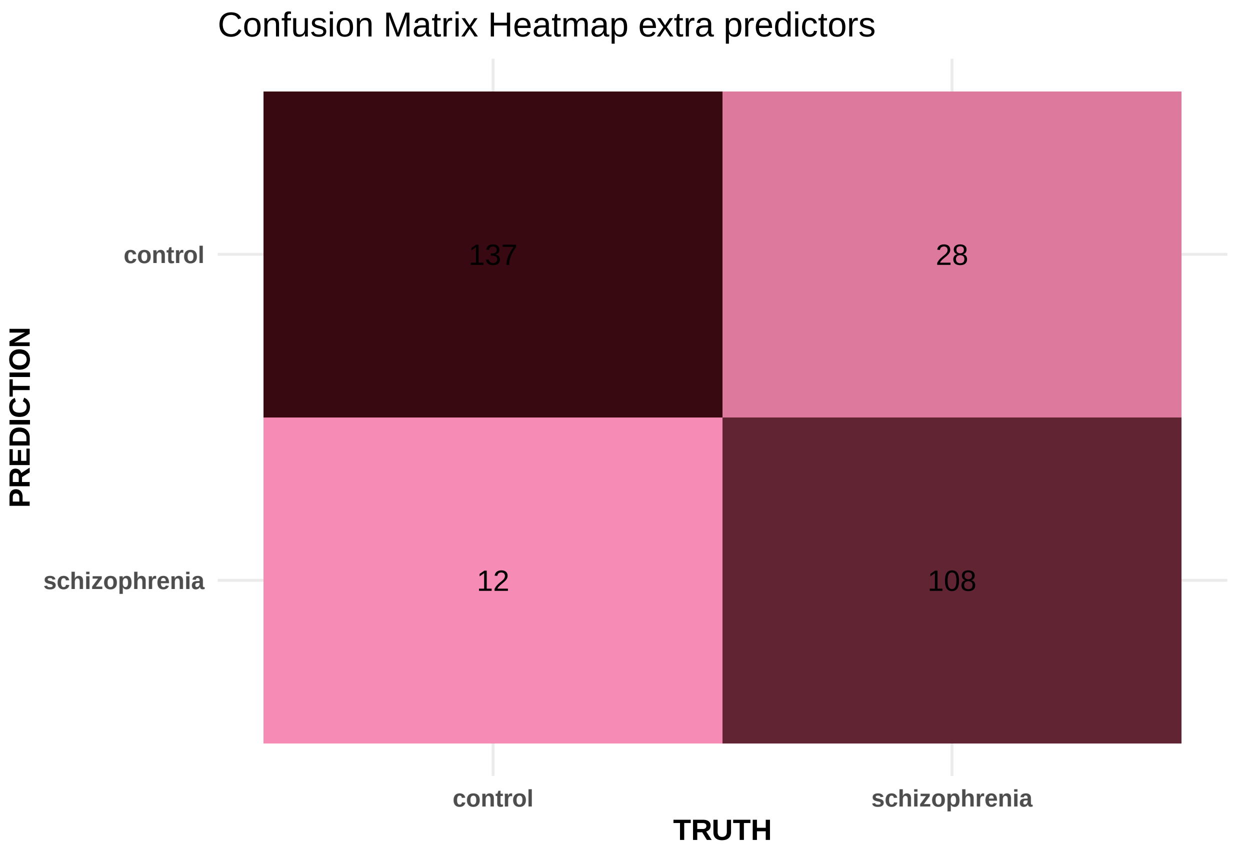
```r
final_extra_predictors_outcome <- simple_testing_data %>%
  as_tibble() %>%
  mutate(
    predicted_class = predict(final_extra_predictors_fit, new_data = simple_testing_data),
    predicted_probabilities = predict(final_extra_predictors_fit,
                                      new_data = simple_testing_data, type = "prob")
  )

# Unnest the predicted_probabilities column
final_extra_predictors_outcome <- final_extra_predictors_outcome %>%
  unnest_wider(c(predicted_probabilities, predicted_class))

final_extra_predictors_outcome[c('diagnosis', '.pred_class', '.pred_control',
                                 '.pred_schizophrenia')]
```

```
## # A tibble: 285 x 4
##    diagnosis     .pred_class   .pred_control .pred_schizophrenia
##    <fct>         <fct>                 <dbl>               <dbl>
##  1 control       control               0.667               0.333
##  2 control       control               0.703               0.297
##  3 control       control               0.731               0.269
##  4 control       control               0.725               0.275
##  5 control       control               0.702               0.298
##  6 control       control               0.745               0.255
##   7 control       control               0.790              0.210
##   8 control       control               0.786              0.214
##   9 control       control               0.858              0.142
##  10 schizophrenia schizophrenia         0.267              0.733
## # i 275 more rows
```

```
#Plotting the matrix
final_extra_predictors_conf_mat <-
  yardstick::conf_mat(final_extra_predictors_outcome,
                      truth = diagnosis,
                      estimate = .pred_class)

final_extra_predictors_conf_plot <-
  autoplot(final_extra_predictors_conf_mat, type = "heatmap") +
  scale_fill_gradient(low="#f68bb5",
                      high = "#380911") +
  theme(
    axis.title = element_text(face = 'bold'),
    axis.text = element_text(face = 'bold')) +
  labs(x = 'TRUTH', y = 'PREDICTION', title = "Confusion Matrix Heatmap extra predictors")
```

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```

```
final_extra_predictors_conf_plot
```



This already looks much better. Lets calculate the odds for a final table of the odds of each model.

```
# Given confusion matrix values
true_positives <- 137
false_positives <- 28
false_negatives <- 12
true_negatives <- 108

# Calculate odds of being right to wrong
odds_value <- (true_positives + true_negatives) / (false_positives + false_negatives)

# Print the expression and calculated odds value
odds_expression <- paste0("Odds of being right to wrong = ", true_positives + true_negatives,
                          " / ", false_positives + false_negatives)
cat(odds_expression, "\n")
```

```
## Odds of being right to wrong = 245 / 40
```

```
cat("Calculated Odds Value: ", odds_value, "\n")
```

```
## Calculated Odds Value:  6.125
```

Discussion: Methodology and Results (Amélie, Katharina)

Finally, briefly summarize and discuss the methodological choices you've made throughout as well as the results obtained at the different modeling stages. In particular, I would like to get your input as regards to the following questions:

- Based on the performance evaluation of your models, do you think the second and third phase of the third section were worth the extra effort? Was any model successful in diagnosing schizophrenia from voice?
- How do the predictive models you built relate to the descriptive analysis conducted in the second section?
- What is the explanatory scope of the analyses and procedures you conducted here, if any?

First of all it should be said that none of our models can be considered "successful" in diagnosing schizophrenia from only voice. While all models perform above chance in diagnosing the true positive schizophrenic participants, their accuracy in differentiating between schizophrenic and control participants is very poor.

When comparing confusion matrixes and odds accross our models (the simple model, tuned model, bayes model), none of them have a significant change in odds or distribution in the confusion matrix. One could assume the negligible change could be due to the random nature of random forest models.

In other words, we can see no improvement in performance in terms of making a right prediction when moving from the simple model to the tuned model, in fact the odds for being right falls!

```
Odds before tuning: 1.938144
Odds after tuning: 1.688679
```

This result could also be due to the method of tuning. As evident by the plots in the tuning section, the grids were largely random; making selecting intervals difficult for manual grid tuning. For future models, other tuning methods ought to be used if using this dataset.

Shockingly our third model based on a Baysien algorithm performs even worse compared to the first two with an odds value of 1.68867. This makes phase 2 and 3 appear to be a waste of time compared to the more simple model, that performed the best. The poor performance of these models indicate, that modelling from voice alone is not ideal as a measure of schizophrenia. As a last point, we choose to include the predictors not directly associated with voice (age, gender, education), and in this model, the results are much more promising.

```
Odds when adding additional predictors: 6.125
```

This shows, that voice can indeed be used as an indicator of schizophrenia, but only when looked at through the lens of the gender, age and education of the individual. Meaning, our model is worthless if we do not adjust for these additional parameters.

The most important predictor in our model appeared to be education, which is not entirely balanced according to our descriptive models. In general, the control group has a higher level of education than the schizophrenia group - this could be problematic for the model. However, besides that, the descriptive models did not show any single predictor of voice as having the biggest influence over the outcome - which is reflected in our vip plot. This indicates that it is the combination of all of these factors, together with the age and education, that has any real prediction power.