Assignment 3 - Diagnosing Schizophrenia from Voice

Instructions

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. (https://doi.org/10.1093/schbul/sbac128)

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.

Your main task for this assignment will be to replicate this research project through the design, fit, and reporting of unsupervised learning methods. More precisely, this assignment will consist in:

1. Collecting and cleaning the project data
2. Understanding the data using descriptive statistics
3. Predicting diagnosis using supervised learning procedures

4. Discussion on the methods and the results

The following sections will address these objectives in order. You can complete each section in the way that best fits you. However, we remind you that proceeding methodically by segmenting your code in multiple, thematically-organised code chunks will greatly help you mane the whole modeling procedure.

1 Collecting and cleaning the project data

There are two different data sets for this assignment:

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

```
art_data <- read.table("../data/articulation_data.txt", header = T, fill = T)
p_data <- read.table("../data/pitch_data.txt", header = T, fill = T)

#the 4th version added much later which requires a different clean-up
final_data <- read.table("../data/final_phonation.txt", header = T, fill = T)
```

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

```
#mutating diagnosis for myself
p_data <- p_data %>%
  mutate(Diagnosis = if_else(Diagnosis == "control", "CTRL", "SCZ" ))

# #getting rid of useless columns
art_data <- art_data %>%
  dplyr::select(-study, -phonationtime., -X.speakingtime.nsyll., -PauseDuration, -ID, -Stu
         dy, -Trial, -Diagnosis, -ASD)

#changing the column names to what they ought be for the merge
art_data <- art_data %>%
  dplyr::rename(Trial = X.nsyll)%>%   #these are already present in the data but wrongly n
         amed
  dplyr::rename(ID = rate)%>%
  dplyr::rename(Diagnosis = articulation)%>%
  dplyr::rename(Study = X.nsyll.dur.) %>%   #adding the two new values that are missing bu
         t can be inferred from data? (hopefully correctly done)
  mutate(articulationrate = (nsyll/phonationtime))%>%
  mutate(ASD = (speechrate/nsyll)) %>%
  mutate(Diagnosis = if_else(Diagnosis == "control", "CTRL", "SCZ" ))

#renaming more to make them pitch-distinct and more convenient for me
p_data <- p_data %>%
 dplyr::rename(p_mean = mean) %>%
 dplyr::rename(p_sd = sd) %>%
 dplyr::rename(p_min = min) %>%
 dplyr::rename(p_max = max) %>%
 dplyr::rename(p_median = median) %>%
 dplyr::rename(p_iqr = iqr)%>%
 dplyr::rename(p_mad = mad) %>%
 dplyr::rename(p_coefvar = coefvar)

#renaming more for my own convenience
art_data <- art_data %>%
 dplyr::rename(full_dur = dur)


#should we choose only the relevant columns, are means, mins, maxes, etc. relevant if not
         listed??
p_data <- p_data %>%
  dplyr::select(ID, Diagnosis, Study, Trial, p_iqr, p_mad, p_coefvar, p_mean, p_sd, p_min,
         p_max, p_median )
art_data <- art_data %>%
    dplyr::select(ID, Diagnosis, Study, Trial, nsyll, npause, full_dur, phonationtime, spe
         echrate, articulationrate, ASD)
```

```
#i assume these to be important as factor; can change it later
p_data$Diagnosis <- as.factor(p_data$Diagnosis)
p_data$Study <- as.factor(p_data$Study)
p_data$Trial <- as.factor(p_data$Trial)
p_data$ID <- as.factor(p_data$ID)

art_data$Diagnosis <- as.factor(art_data$Diagnosis)
art_data$Study <- as.factor(art_data$Study)
art_data$Trial <- as.factor(art_data$Trial)
art_data$ID <- as.factor(art_data$ID)
```

After importing the data sets, make sure all common columns and values are named accordingly. Finally, merge the data sets on the appropriate columns, rename columns and values to your liking, and save the resulting data set using a file name and path of your own choosing.

```
#original clean up merge, replaced by the maxime update
df <- left_join(p_data, art_data, by = c("ID", "Study", "Trial", "Diagnosis"))
write.csv(df, "../data/merged_data.csv", row.names = FALSE)
```

1.1 Fixing the latest dataframe given by Maxime later on, this makes the cleaning up of earlier obsolete:

```
final_data <- final_data %>%
  mutate(diagnosis = if_else(diagnosis == "control", "CTRL", "SCZ" )) %>%
   dplyr::rename(syllable_duration = X.n_syllables.duration., syllables_phonation_duration
         = X.n_syllables.phonation_duration.)%>%
   dplyr::rename(Study = study, ID = id, Trial = trial, Diagnosis = diagnosis)%>%
  mutate(average_syllable_duration = (syllable_duration/n_syllables))%>%
  mutate(average_pause_duration = (pause_duration/n_pauses))%>%
   dplyr::rename(total_pause_duration = pause_duration)


final_data$Diagnosis <- as.factor(final_data$Diagnosis)
final_data$Study <- as.factor(final_data$Study)
final_data$Trial <- as.factor(final_data$Trial)
final_data$ID <- as.factor(final_data$ID)

#as far as I know, we should have the pitch data, even without gender
final_data <- left_join(final_data, p_data, by = c("ID", "Study", "Trial", "Diagnosis"))
write.csv(final_data, "../data/merged_data_from_maxime.csv", row.names = FALSE)


df <- final_data
```

```
#there appear to be some extreme values, though I assume they are permitted, hence 3SD as
        an arbitrary range
df <- df %>%
  mutate(across(
    where(is.numeric),
    ~ ifelse(
      abs(as.numeric(scale(.x))) > 3,
      NA,
      .x
    )
  ))

#why are there so many NA values in this dataframe but not the originals? 1900 obs. to 156
        2 obs. (though some were extreme values omitted)
#could be fixed with ML? for now ill just omit
df <- na.omit(df)
```

2 Understanding the sample using descriptive statistics

In this section, use whatever statistical procedures you think relevant to get a good understanding of the data set, particularly as regards to the differences between linguistic markers of neurotypical and schizophrenic speech. Here as in the following sections, make sure that we understand what you're doing and why you're doing it (you can do this by adding text right before or after the corresponding chunk of code).

Here are some of the things you can do:

```
df %>%
  group_by(Diagnosis) %>%
  dplyr::summarize(Count = n())
```

```
## # A tibble: 2 × 2
##   Diagnosis Count
##   <fct>     <int>
## 1 CTRL        847
## 2 SCZ         715
```

```
df %>%
  group_by(Diagnosis, Study) %>%
  dplyr::summarize(Count = n())
```

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

```
## # A tibble: 8 × 3
## # Groups:   Diagnosis [2]
##   Diagnosis Study Count
##   <fct>     <fct> <int>
## 1 CTRL      1       308
## 2 CTRL      2       159
## 3 CTRL      3       190
## 4 CTRL      4       190
## 5 SCZ       1       261
## 6 SCZ       2       146
## 7 SCZ       3       112
## 8 SCZ       4       196
```

```
df %>%
  group_by(Diagnosis, Trial) %>%
  dplyr::summarize(Count = n())
```

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

```
## # A tibble: 20 × 3
## # Groups:   Diagnosis [2]
##    Diagnosis Trial Count
##    <fct>     <fct> <int>
##  1 CTRL      1       100
##  2 CTRL      2       102
##  3 CTRL      3        99
##  4 CTRL      4       102
##  5 CTRL      5        99
##  6 CTRL      6        98
##  7 CTRL      7        94
##  8 CTRL      8        95
##  9 CTRL      9        34
## 10 CTRL      10       24
## 11 SCZ       1        82
## 12 SCZ       2        86
## 13 SCZ       3        85
## 14 SCZ       4        90
## 15 SCZ       5        82
## 16 SCZ       6        83
## 17 SCZ       7        83
## 18 SCZ       8        67
## 19 SCZ       9        37
## 20 SCZ       10       20
```

2.1 Describe the data set (number of studies, number of participants, age, gender, clinical and cognitive features of the two groups) and assess whether the groups (schizophrenia and controls) are balanced.

The ratio between diagnoses, studies and trials seems balanced enough, though not perfectly 50/50 (for some reason big difference between Study 3 for SCZ and CTRL (151 vs 232)

```
df %>%
  group_by(Diagnosis)%>%
  dplyr::summarize(across(c(speech_rate, phonation_duration, articulation_rate, average_sy
         llable_duration, n_pauses, n_syllables, p_max), mean))
```

```
## # A tibble: 2 × 8
##   Diagnosis speech_rate phonation_duration articulation_rate
##   <fct>           <dbl>              <dbl>             <dbl>
## 1 CTRL             3.20               11.6             0.201
## 2 SCZ              2.99               10.7             0.208
## # ℹ 4 more variables: average_syllable_duration <dbl>, n_pauses <dbl>,
## #   n_syllables <dbl>, p_max <dbl>
```

2.2 Describe the acoustic profile of a schizophrenic voice: which features are different? E.g. People with schizophrenia tend to have high-pitched voice.

SCZ appear to have lower mean speech rate, less syllables spoken and a lower max pitch. Their mean average syllable duration also appears longer. It appears as though they do not take more pauses during the recordings and also articulate when speaking just as much.

```
#doing a quick and dirty t-test to "confirm" the assumptions made at a glance (assuming no
         n-normality so wilcox)
df %>%
  dplyr::summarise(across(where(is.numeric), ~ wilcox.test(.x ~ as.factor(Diagnosis), data
         = df)$p.value)) %>%
  select_if(function(x) any(x < 0.05))
```

```
##     duration phonation_duration  n_syllables  speech_rate syllable_duration
## 1 0.02115207        0.000906319 3.831114e-05 1.474453e-08      5.356234e-09
##   articulation_rate syllables_phonation_duration average_syllable_duration
## 1      5.350809e-09                  0.003494364              0.0009088874
##   average_pause_duration        p_iqr        p_mad    p_coefvar         p_sd
## 1            0.003477362 8.110046e-07 1.120515e-06 4.400715e-13 4.364104e-09
##       p_min       p_max
## 1 0.0397852 0.000413266
```

```
df %>%
  dplyr::summarise(across(where(is.numeric), ~ wilcox.test(.x ~ as.factor(Diagnosis), data
         = df)$p.value)) %>%
  select_if(function(x) any(x > 0.05))
```

```
##   total_pause_duration  n_pauses    p_mean  p_median
## 1             0.672488 0.1145926 0.9286746 0.5816086
```

Most columns (outside of pause_duration) are ""statistically significant"" as in, likely to be from 2 different populations (this being a difference between the diagnoses)

Namely what "differs statistically" is, the duration of the full recording (duration, why different recording times?), the duration of phonation within the recording (phonation_duration), the amount of syllables spoken (n_syllables), amount of pauses taken (n_pauses), average number of syllables per second (speech_rate), the avg duration of syllables spoken, the average number of syllables per second where speech is present (articulation_rate), duration it took to phonate syllables (syllables_phonation_duration) and average_pause_duration (self-explanatory) values related to pitch were also significant, such as pitch min and max

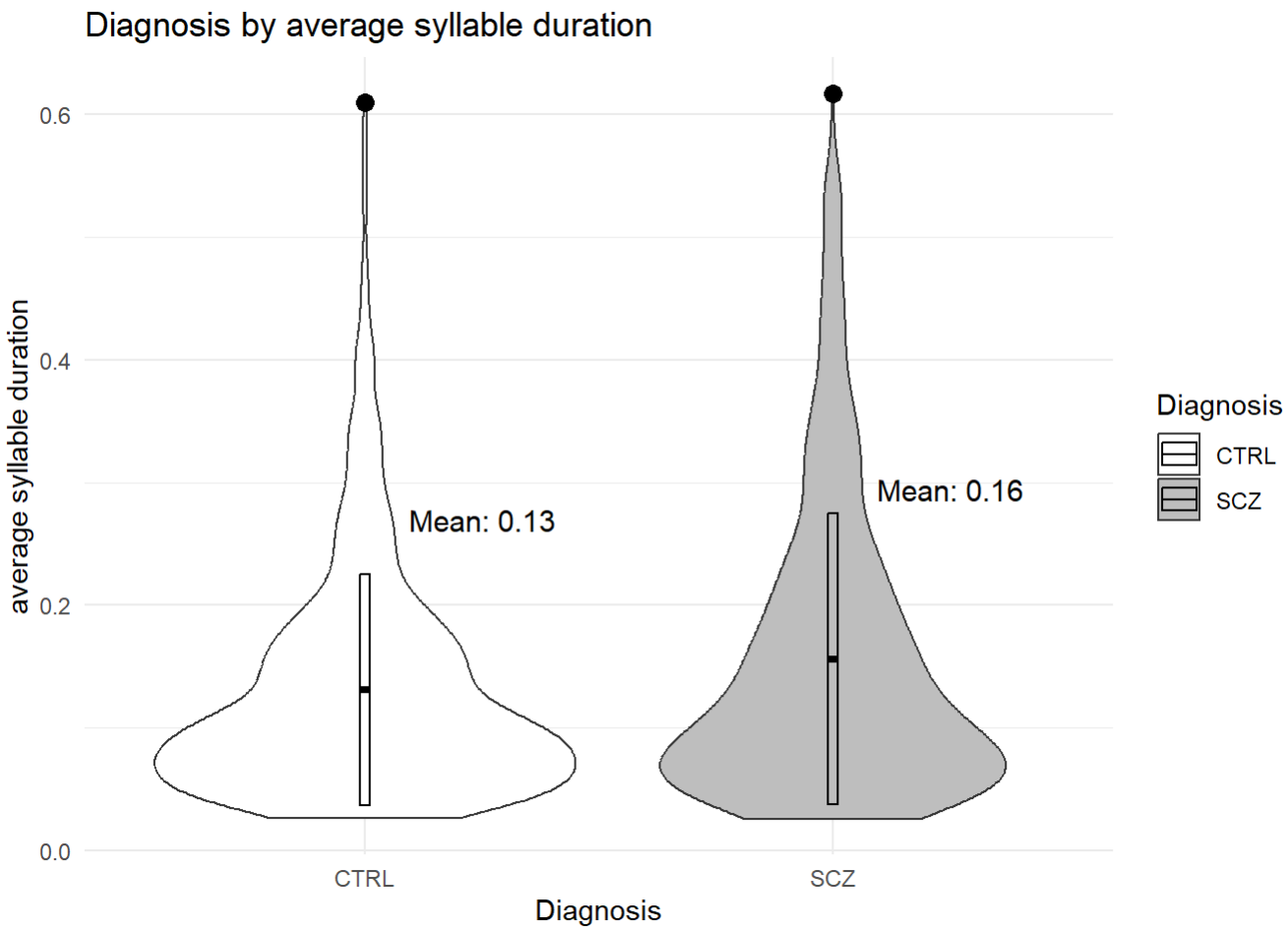
Pauses don't appear to differ so much, making it seem as though this is not a marker for SCZ-style speech. Pitch mean and median are also not statistically significant, meaning that they are roughly the same for both SCZ and CTRL.

This quick t-test would indicate that the speech variables and pitch do differ for the two groups.

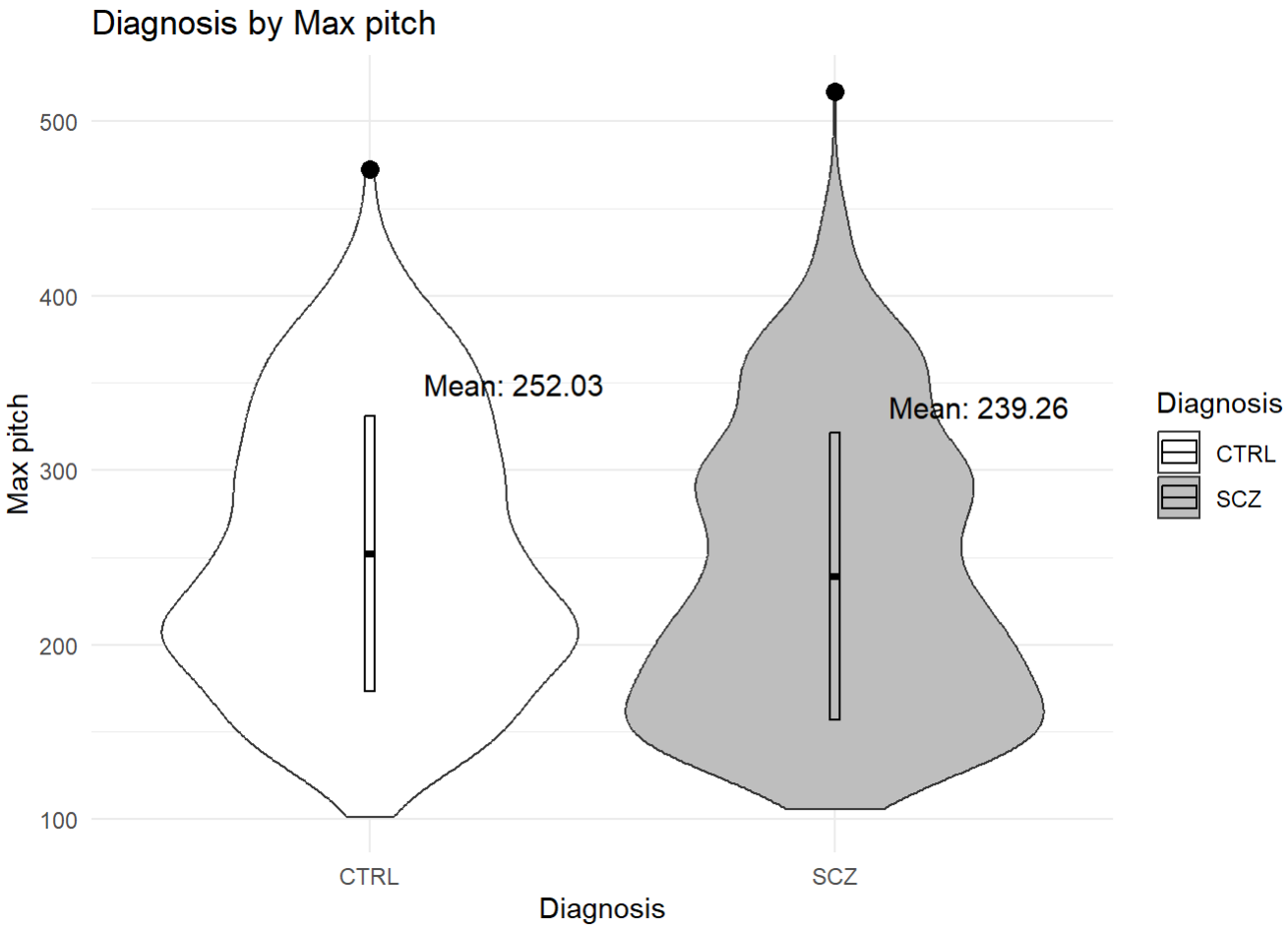
That said, I don't know whether I am justified in doing a quick t-test just to see whether the variables are likely to "distinguish" the two groups from each other. Also, not like I checked the assumptions.

```
#this is very stupidly done here, couldve gotten the mean in the function just fine
means <- df %>%
  group_by(Diagnosis) %>%
  dplyr::summarize(mean_ASD = mean(average_syllable_duration), mean_nsyll = mean(n_syllabl
        es), mean_npause = mean(n_pauses), mean_pitchmax = mean(p_max), mean_phonationtim
        e = mean(phonation_duration), mean_speechrate = mean(speech_rate), mean_a_pausedu
        r = mean(average_pause_duration), mean_piqr = mean(p_iqr), mean_pitchmin = mean(p
        _min))

plotting_function <- function(df, x, y, fill, means_df, mean_y, altstringx = NULL, altstri
        ngy = NULL) {
  title_x <- ifelse(!is.null(altstringx), altstringx, gsub("_", " ", x))
  title_y <- ifelse(!is.null(altstringy), altstringy, gsub("_", " ", y))

  ggplot(df, aes_string(x = x, y = y, fill = fill)) +
    geom_violin(trim = TRUE) +
    labs(x = title_x, y = title_y) +
    ggtitle(paste0(gsub("_", " ", title_x), " by ", gsub("_", " ", title_y))) +
    theme_minimal() +
    scale_fill_manual(values = c("SCZ" = "gray", "CTRL" = "white")) +
    geom_text(data = means_df, aes(label = paste("Mean:", round(.data[[mean_y]], 2))),
              x = means_df[[x]], y = means_df[[mean_y]], vjust = -7, hjust = -0.3, size =
        4) +
    stat_summary(fun = "max", geom = "point", size = 3,
                 position = position_dodge(width = 0.75), color = "black") +
    stat_summary(
      fun.data = function(x) {
        mean_val <- mean(x)
        sd_val <- sd(x)
        ymin_val <- max(0, mean_val - sd_val)
        ymax_val <- mean_val + sd_val
        return(data.frame(y = mean_val, ymin = ymin_val, ymax = ymax_val))
      },
      geom = "crossbar",
      width = 0.02
    )
}
```

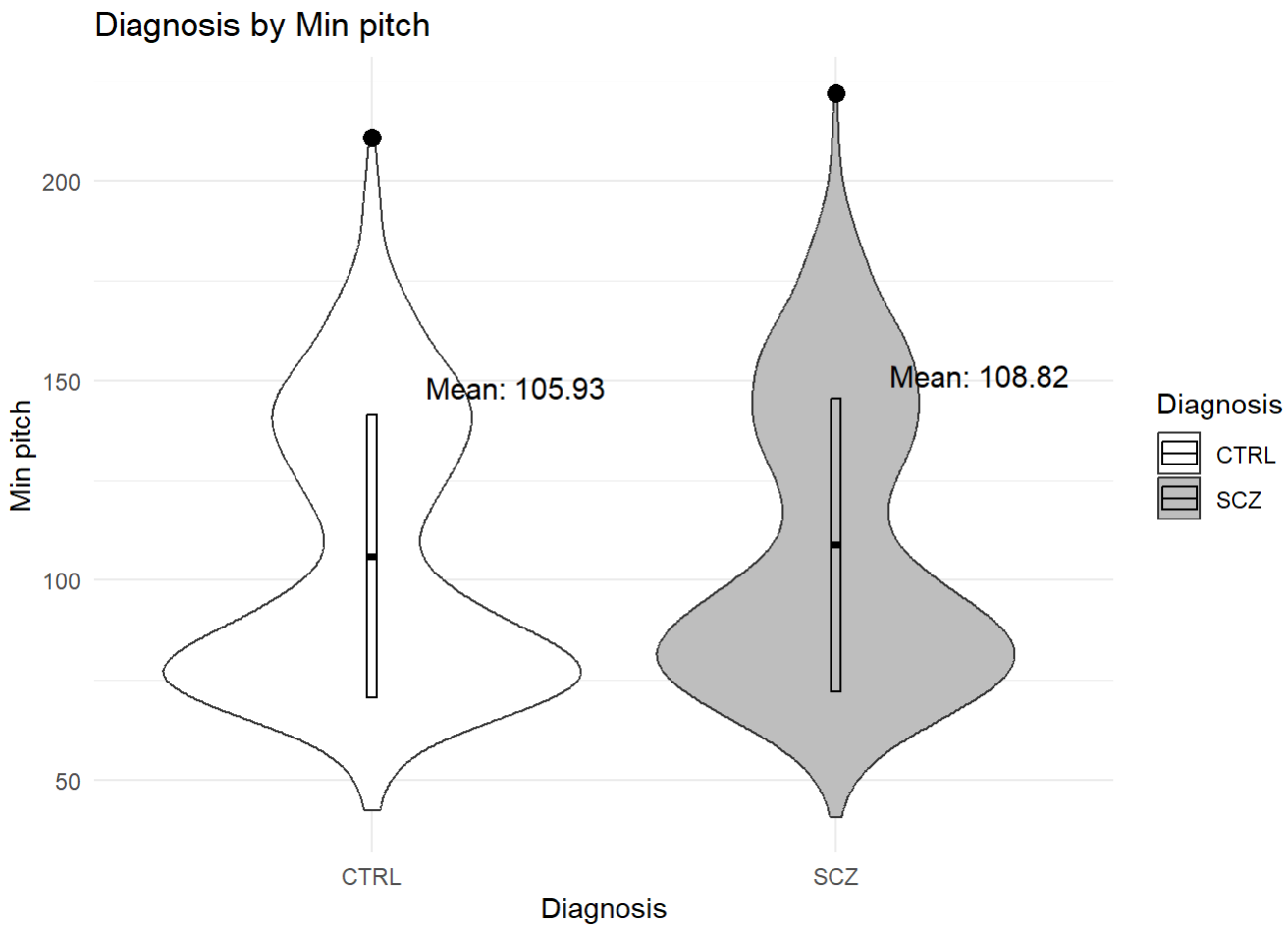
```
plotting_function(df, "Diagnosis", "average_syllable_duration", "Diagnosis", means, "mean_
        ASD")
```



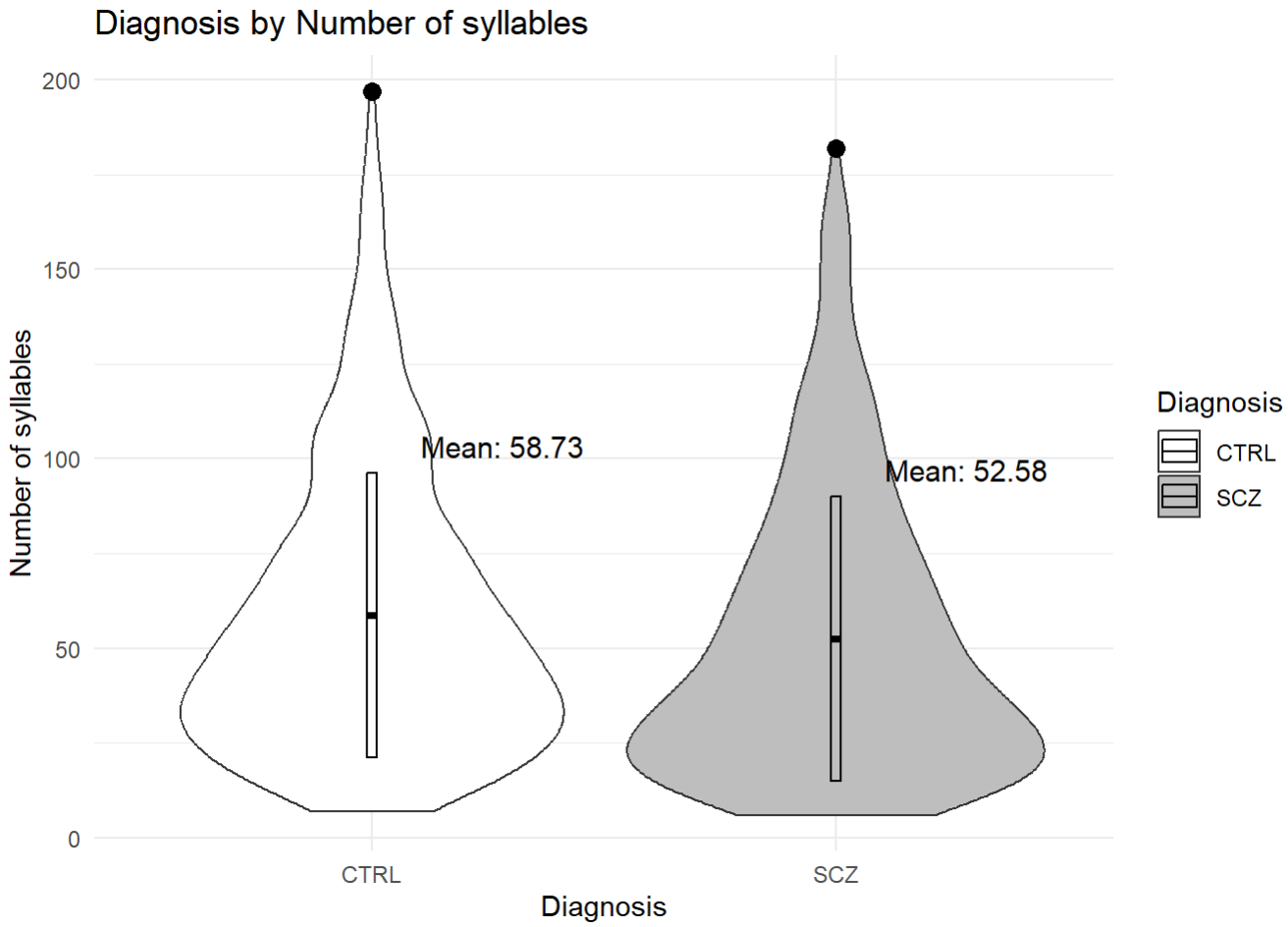

```
plotting_function(df, "Diagnosis", "p_max", "Diagnosis", means, "mean_pitchmax", altstring
        y = "Max pitch")
```



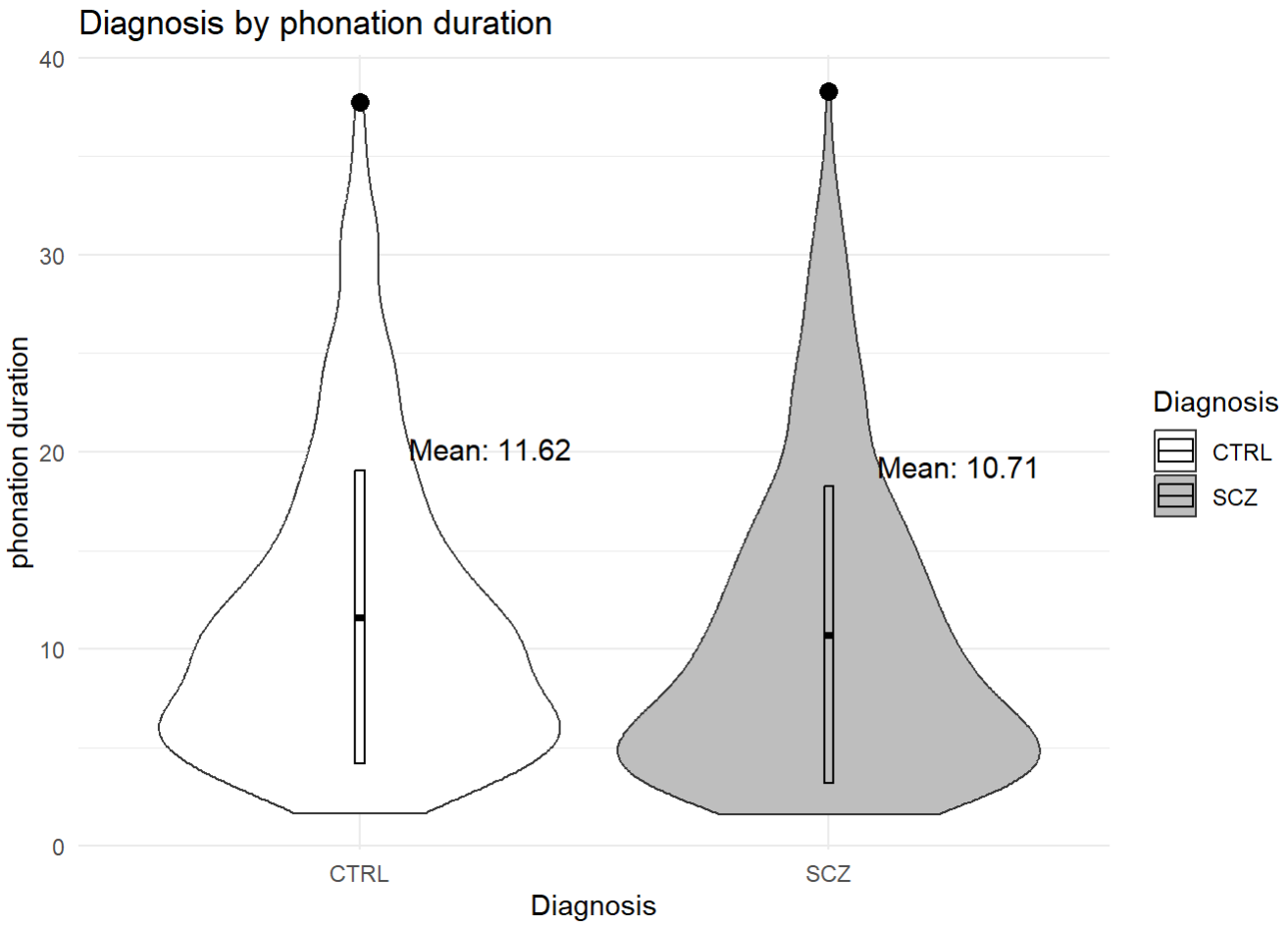

```
plotting_function(df, "Diagnosis", "p_min", "Diagnosis", means, "mean_pitchmin", altstring
        y = "Min pitch")
```



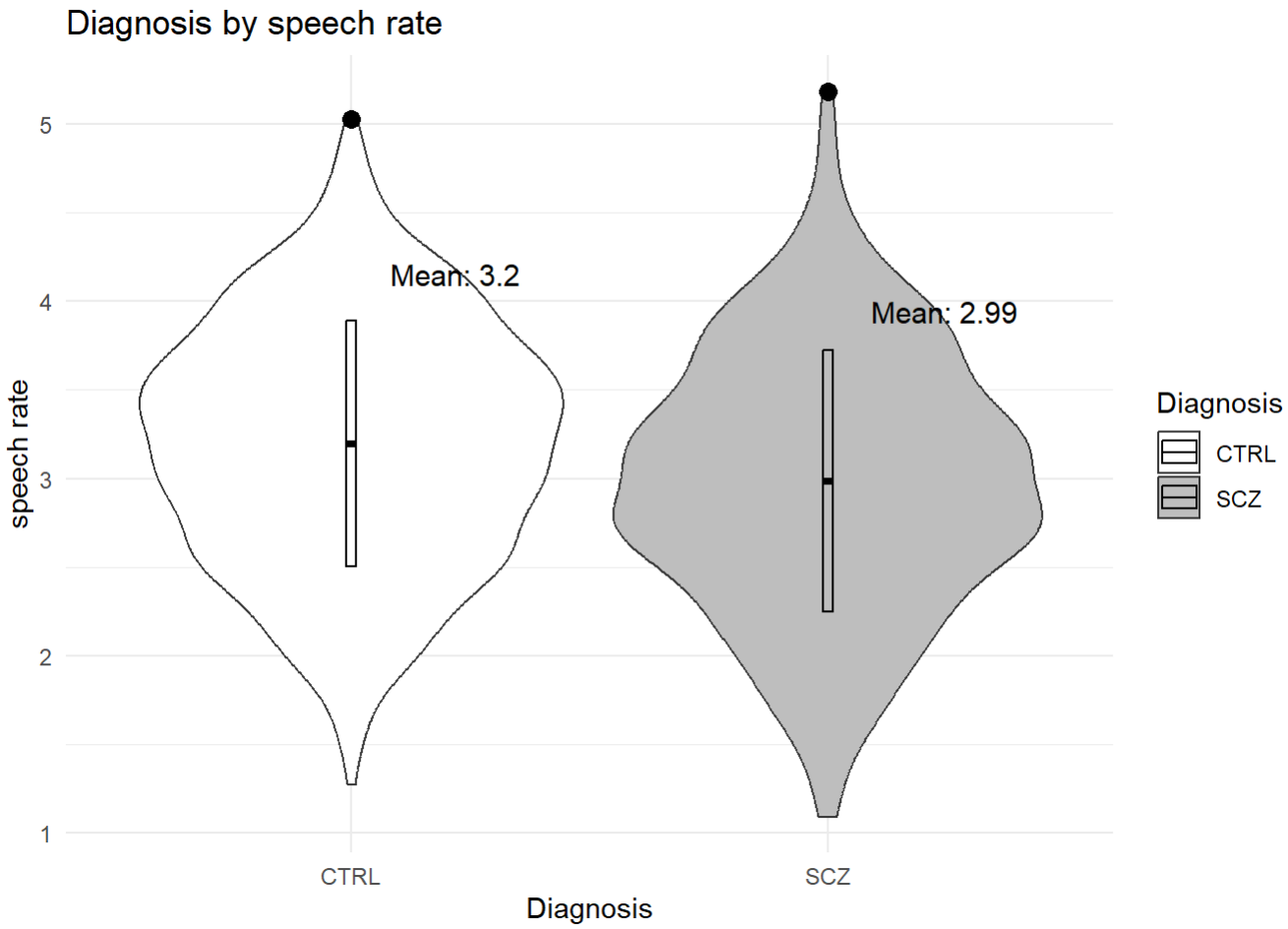

```
plotting_function(df, "Diagnosis", "n_syllables", "Diagnosis", means, "mean_nsyll", altstr
        ingy = "Number of syllables")
```



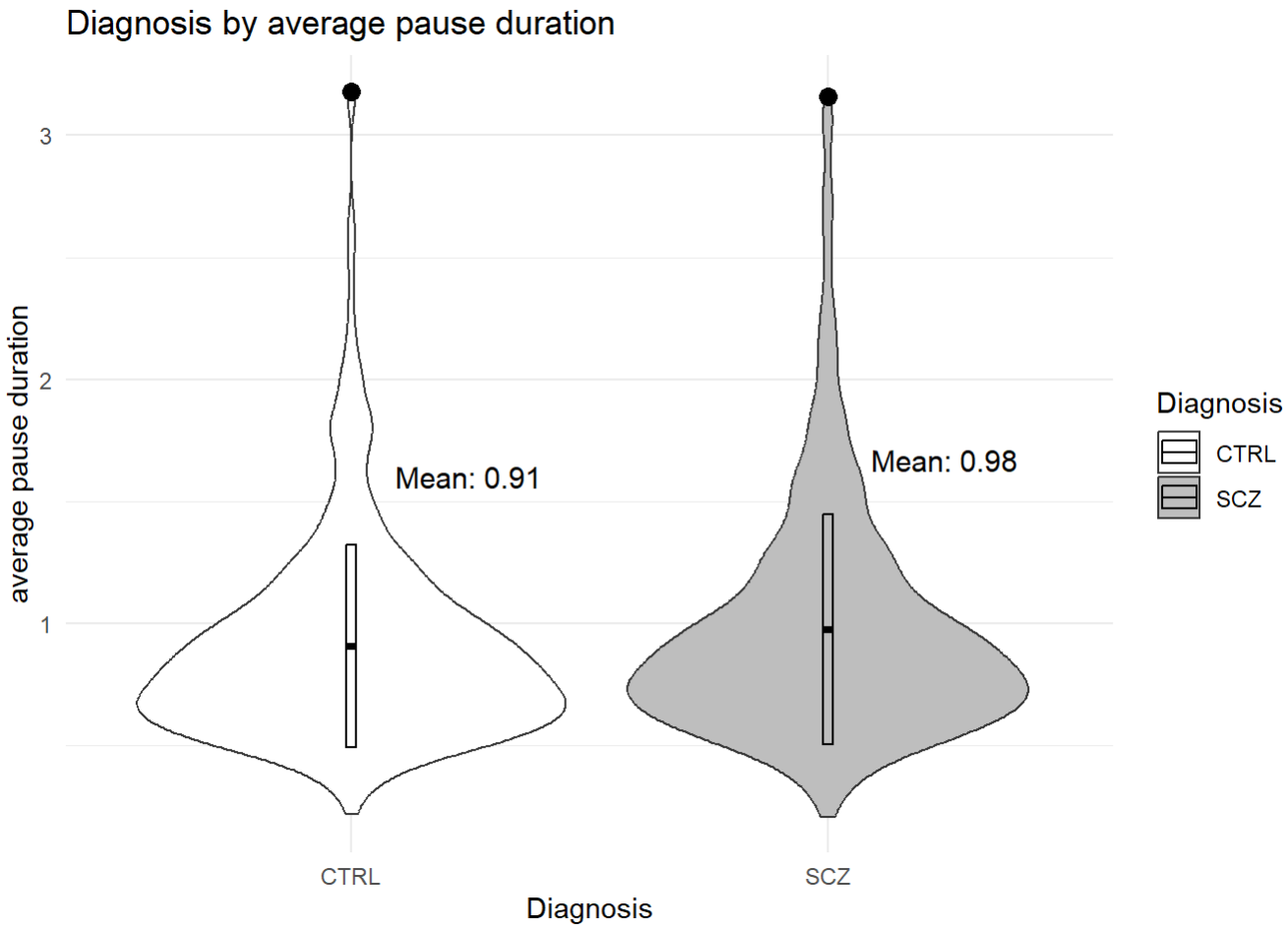

```
plotting_function(df, "Diagnosis", "phonation_duration", "Diagnosis", means, "mean_phonati
        ontime")
```



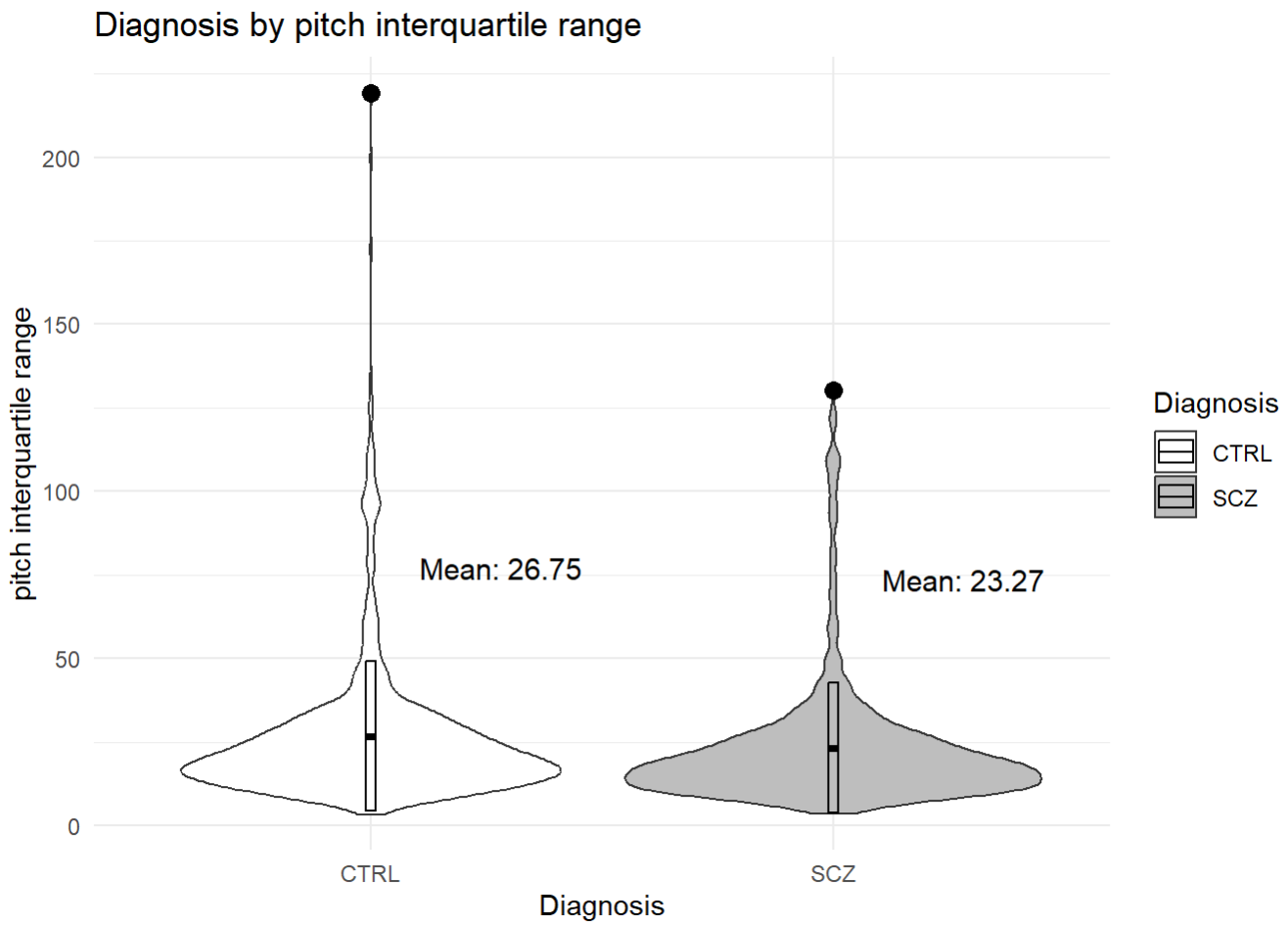

```
plotting_function(df, "Diagnosis", "speech_rate", "Diagnosis", means, "mean_speechrate")
```




```
plotting_function(df, "Diagnosis", "average_pause_duration", "Diagnosis", means, "mean_a_p
        ausedur")
```




```
plotting_function(df, "Diagnosis", "p_iqr", "Diagnosis", means, "mean_piqr", altstringy =
         "pitch interquartile range")
```



Visually, it appears that the density plots for both conditions are fairly similar, with the main difference being that SCZ individuals are differently weighed, or are trending towards a direction; for example, they tend to have lower max pitch, higher syllable duration, speak less syllables and phonate (speak) less during the whole recording.

That said, whether this is distinctive enough to diagnose an individual is hard to say, since the vast majority of SCZ data points seem to overlap with the CTRL individuals in the density plots.

3 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. Once again, we ask that you connect the different code chunks you create with short descriptive/explanatory text segments that gives us an idea about what you are doing and why you are doing it.

The following packages will be useful to you here:

- **tidymodels** (https://tidymodels.tidymodels.org/): "meta-package" for modeling and statistical analysis that shares the underlying design philosophy, grammar, and data structures of the tidyverse.
- **rsample** (https://rsample.tidymodels.org/): as infrastructure for resampling data so that models can be assessed and empirically validated. -[**groupdata2**][(https://cran.r-

project.org/web/packages/groupdata2/vignettes/introduction_to_groupdata2.html (https://cran.r-project.org/web/packages/groupdata2/vignettes/introduction_to_groupdata2.html))]: an alternative to rsample that allows resampling with deeper grouping
- **tune** (https://tune.tidymodels.org/): contains the functions to optimize model hyper-parameters.
- **dials** (https://dials.tidymodels.org/): tools to create and manage values of tuning parameters.
- **recipes** (https://recipes.tidymodels.org/index.html): a general data preprocessor that can create model matrices incorporating feature engineering, imputation, and other tools.
- **workflows** (https://workflows.tidymodels.org/): methods to combine pre-processing steps and models into a single object.
- **workflowsets** (https://workflowsets.tidymodels.org/): can create a workflow set that holds multiple workflow objects, allowing users to create and easily fit a large number of models.
- **parsnip** (https://parsnip.tidymodels.org/): a tidy, unified interface to creating models
- **yardstick** (https://yardstick.tidymodels.org/): contains tools for evaluating models

Finally, here are some online resources that can help you with the modeling process:

- This **Tidymodels tutorial** (https://www.tidymodels.org/start/) written by the Tidymodels team
- This **workshop on Tidymodels** (https://workshops.tidymodels.org/) written by the Tidymodels team
- This **workshop on Tidymodels** (https://apreshill.github.io/tidymodels-it/) written by the Posit Team (The company behind RStudio)
- This **online course on supervised machine learning** (https://supervised-ml-course.netlify.app/) written by the Tidymodels team

3.1 First phase: Random Forest Model

In this phase, you will build a simple random forest model, by:

- Splitting the data in training and testing sets

```
#omitting Study and Trial since they're not linguistic markers
df <- df %>%
  dplyr::select(-Study, -Trial)

split_df <- initial_split(df, prop = 0.7,strata = Diagnosis)
#this method splits the df into training & testing sets
```

```
#just curious
prop_train <- table(training(split_df)$Diagnosis) / length(training(split_df)$Diagnosis)
prop_test <- table(testing(split_df)$Diagnosis) / length(testing(split_df)$Diagnosis)
orig <- table(df$Diagnosis) / nrow(df)


cat("Proportions in Training Set:\n", paste(names(prop_train), prop_train, sep = ": "),
         "\n\n")
```

```
## Proportions in Training Set:
##  CTRL: 0.542124542124542 SCZ: 0.457875457875458
```

```
cat("Proportions in Testing Set:\n", paste(names(prop_test), prop_test, sep = ": "), "\n
         \n")
```

```
## Proportions in Testing Set:
##  CTRL: 0.542553191489362 SCZ: 0.457446808510638
```

```
cat("Proportions in OG set:\n", paste(names(orig), orig, sep = ": "), "\n\n")
```

```
## Proportions in OG set:
##  CTRL: 0.542253521126761 SCZ: 0.457746478873239
```

- Training a random forest model on the training set

```
rf_model <- rand_forest(
  mode = "classification",
  mtry = NULL,
  trees = 100,
  min_n = 5
) %>%
  set_engine("ranger") %>%
  fit(Diagnosis ~ . , data = training(split_df))
```

- Testing the model's predictions on the testing set

```
predictions <- predict(rf_model, new_data = testing(split_df))
```

- Building the confusion matrix

```
conf_matrix <- confusionMatrix(predictions$.pred_class, testing(split_df)$Diagnosis)
conf_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction CTRL SCZ
##       CTRL  196  96
##       SCZ    59 119
##
##                Accuracy : 0.6702
##                  95% CI : (0.6257, 0.7126)
##     No Information Rate : 0.5426
##     P-Value [Acc > NIR] : 1.209e-08
##
##                   Kappa : 0.3265
##
##  Mcnemar's Test P-Value : 0.003833
##
##             Sensitivity : 0.7686
##             Specificity : 0.5535
##          Pos Pred Value : 0.6712
##          Neg Pred Value : 0.6685
##              Prevalence : 0.5426
##          Detection Rate : 0.4170
##    Detection Prevalence : 0.6213
##       Balanced Accuracy : 0.6611
##
##        'Positive' Class : CTRL
##
```

- Compiling performance metrics of your own choosing.

```
#a bunch of metrics from carot package, probably can be done in a more concise way
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Sensitivity"]
f1_score <- (2 * precision * recall) / (precision + recall)
specificity <- conf_matrix$byClass["Specificity"]
roc_auc <- roc(testing(split_df)$Diagnosis, as.numeric(predictions$.pred_class))$auc
```

```
## Setting levels: control = CTRL, case = SCZ
```

```
## Setting direction: controls < cases
```

```r
#fix this error

#metrics, dont understand all of them yet, i.e. Precision = True Positives/(true positives
        + false positives)
result_string <- paste(
  "Accuracy:", round(accuracy * 100, 2), "%\n",
  "Precision:", round(precision, 2), "\n",
  "Recall:", round(recall, 2), "\n",
  "F1 Score:", round(f1_score, 2), "\n",
  "Specificity:", round(specificity, 2), "\n",
  "AUC-ROC:", round(roc_auc, 2)
)

print(result_string)
```

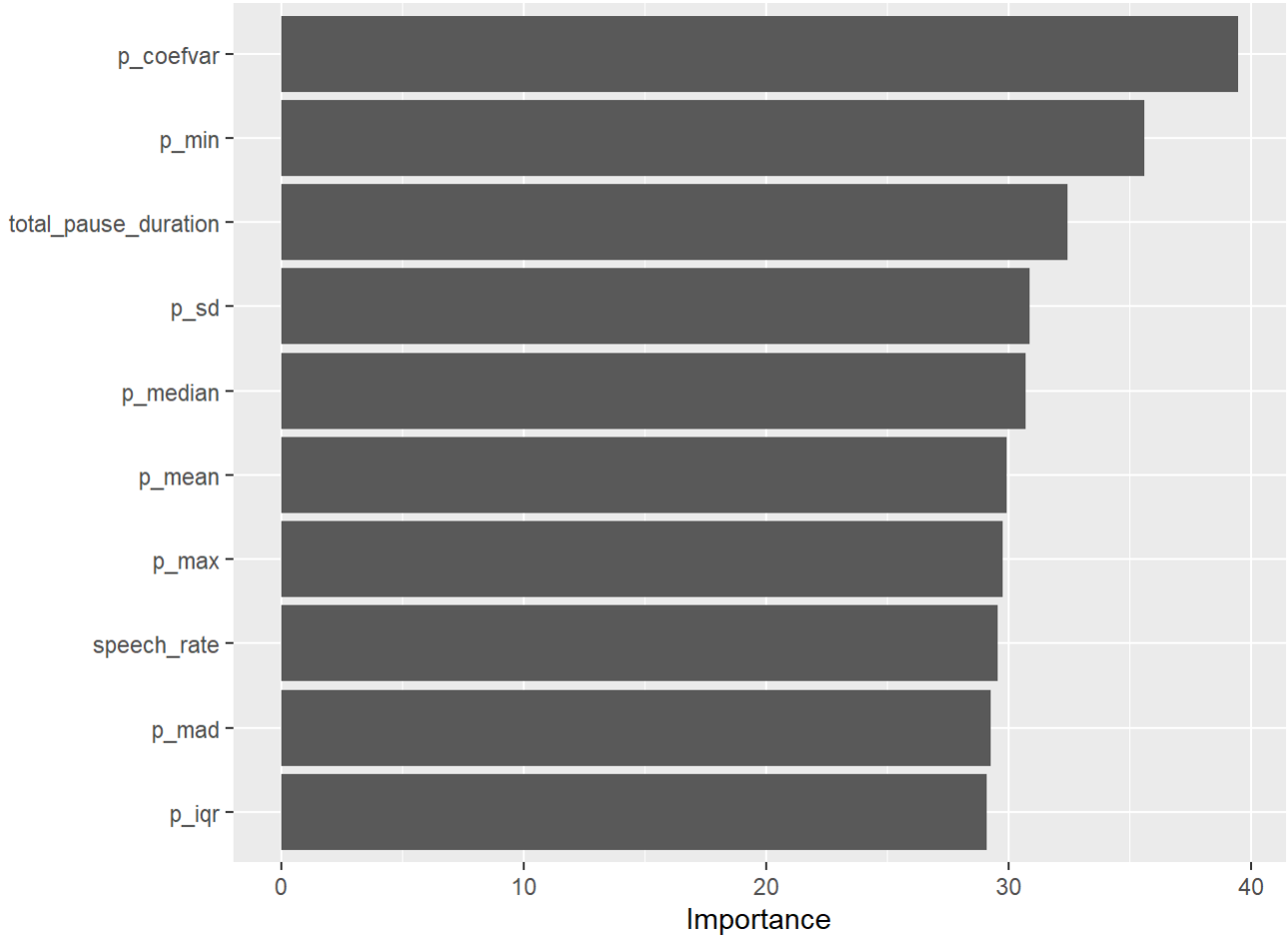
```
## [1] "Accuracy: 67.02 %\n Precision: 0.67 \n Recall: 0.77 \n F1 Score: 0.72 \n Specifici
ty: 0.55 \n AUC-ROC: 0.66"
```

```r
#67.74 % accuracy at default? with no feature engineering?
#AUC-ROC 0.67
```

```
## # A tibble: 21 × 4
##    variable                     type      role      source
##    <chr>                        <list>    <chr>     <chr>
##  1 ID                           <chr [3]> ID        original
##  2 duration                     <chr [2]> predictor original
##  3 phonation_duration           <chr [2]> predictor original
##  4 total_pause_duration         <chr [2]> predictor original
##  5 n_syllables                  <chr [2]> predictor original
##  6 n_pauses                     <chr [2]> predictor original
##  7 speech_rate                  <chr [2]> predictor original
##  8 syllable_duration            <chr [2]> predictor original
##  9 articulation_rate            <chr [2]> predictor original
## 10 syllables_phonation_duration <chr [2]> predictor original
## # ℹ 11 more rows
```

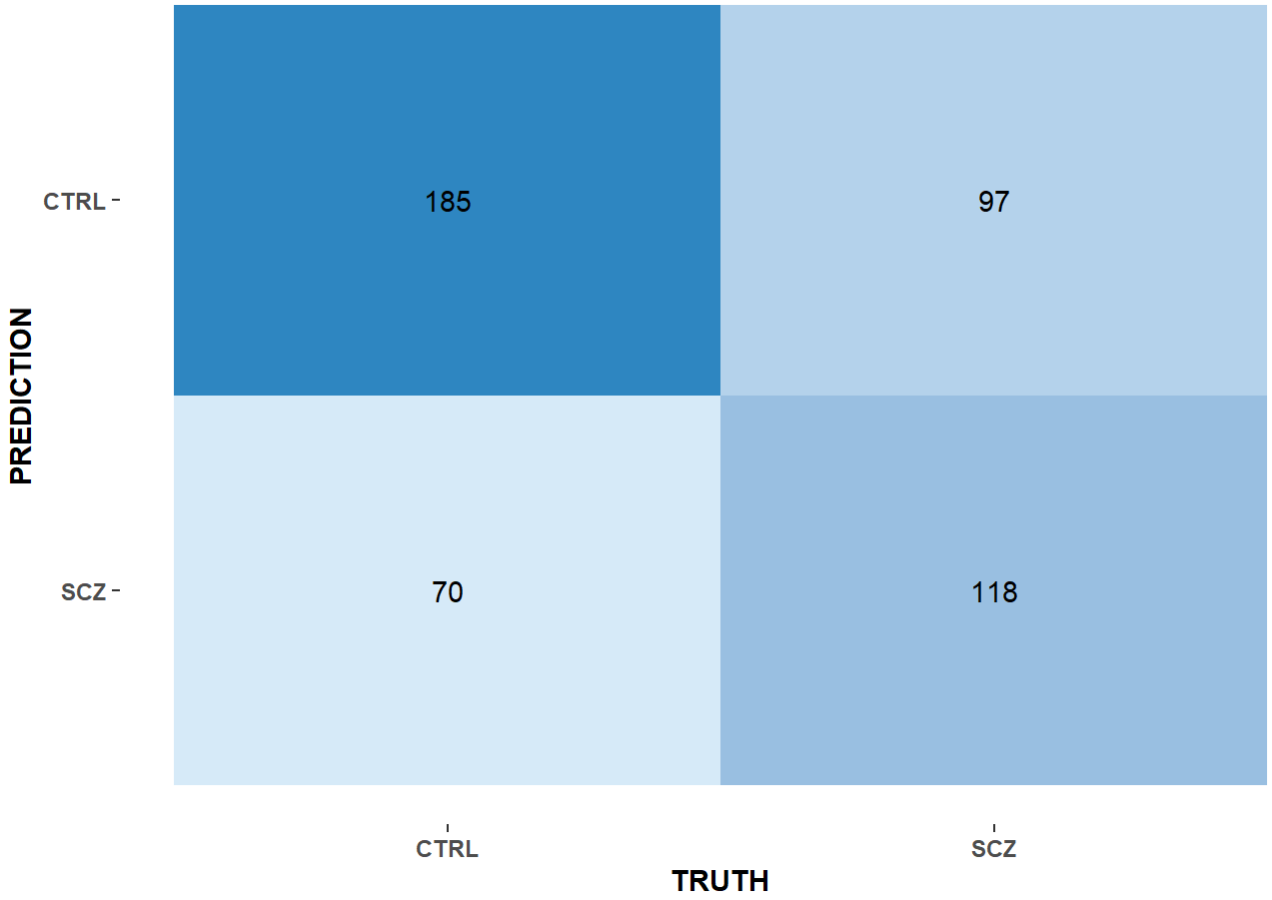
```
## ══ Workflow ════════════════════════════════════════════════════════════════════
## Preprocessor: Recipe
## Model: rand_forest()
##
## ── Preprocessor ────────────────────────────────────────────────────────────────
## 0 Recipe Steps
##
## ── Model ───────────────────────────────────────────────────────────────────────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

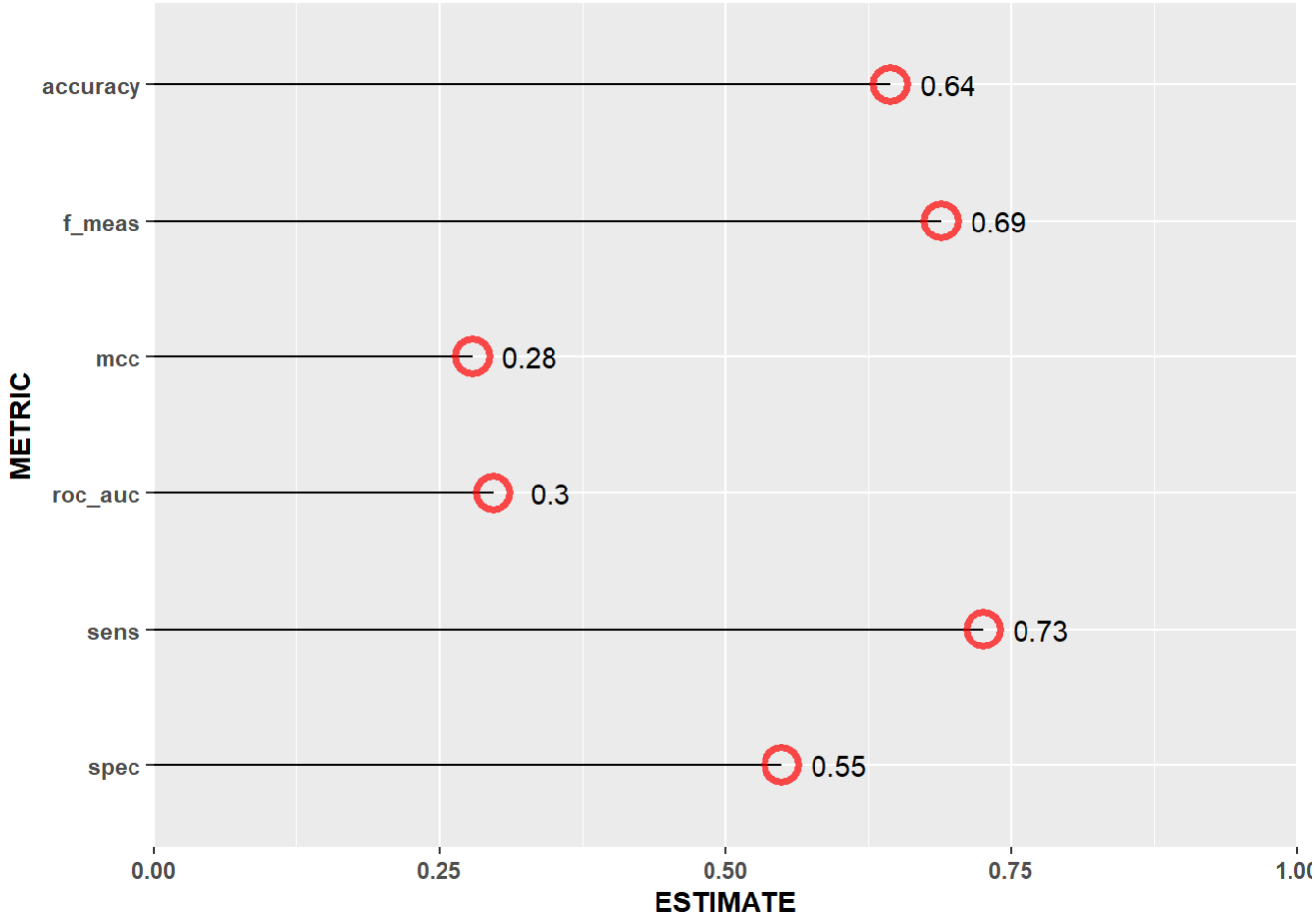
```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 0 Recipe Steps
##
## -- Model -----------------------------------------------------------------------
##
## 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: 39.47%
## Confusion matrix:
##      CTRL SCZ class.error
## CTRL  402 190   0.3209459
## SCZ   241 259   0.4820000
```



```
## # A tibble: 15 × 4
##    Diagnosis .pred_class .pred_SCZ .pred_CTRL
##    <fct>     <fct>           <dbl>      <dbl>
##  1 CTRL      SCZ             0.626      0.374
##  2 CTRL      CTRL            0.458      0.542
##  3 CTRL      CTRL            0.386      0.614
##  4 CTRL      SCZ             0.56       0.44
##  5 CTRL      CTRL            0.414      0.586
##  6 CTRL      CTRL            0.458      0.542
##  7 CTRL      SCZ             0.664      0.336
##  8 CTRL      SCZ             0.552      0.448
##  9 CTRL      SCZ             0.57       0.43
## 10 CTRL      SCZ             0.612      0.388
## 11 CTRL      SCZ             0.692      0.308
## 12 CTRL      CTRL            0.404      0.596
## 13 CTRL      SCZ             0.66       0.34
## 14 CTRL      SCZ             0.662      0.338
## 15 CTRL      SCZ             0.646      0.354
```

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


CTRL
SCZ
PREDICTION
185
97
70
118
CTRL
SCZ
TRUTH


accuracy
f_meas
mcc
roc_auc
sens
spec
METRIC
0.64
0.69
0.28
0.3
0.73
0.55
0.00
0.25
0.50
0.75
1.00
ESTIMATE

3.2 Second phase: Forest Engineering

In this section, you will try to optimize the performance of the model developed in the previous phase by adding a new random forest model, upgraded with feature engineering and parameter tuning procedures of your own choosing.

```
#curious about the correlation of the numeric variables; many are derived from the same th
        ing, so might be bad to have? though could also help the algorithm; helpful redun
        dancy?
numeric_df <- df[sapply(df, is.numeric) & !sapply(df, is.factor)]
correlation_matrix <- cor(numeric_df)

hc <- findCorrelation(correlation_matrix, cutoff=0.5) #setting the cut off fairly high
not_cor_variables <- df[c(hc)]
not_cor_variables <-  not_cor_variables[, c("ID", "Diagnosis", setdiff(names(not_cor_varia
        bles), c("ID", "Diagnosis")))] #reordering

#these could be good ones to keep? though i dont know fully if having correlated variables
        is bad? also doesnt the ML automatically figure this out, or?

print(paste0(paste(names(df)[-hc], collapse = ", "), " - These should be omitted due to hi
        gh correlation? Is this even remotely correct?"))
```

```
## [1] "total_pause_duration, n_pauses, syllable_duration, average_syllable_duration, p_me
an, p_max, p_median - These should be omitted due to high correlation? Is this even remote
ly correct?"
```

```
#formula <- Diagnosis ~ ID + phonation_duration + p_sd + p_min + p_mad + articulation_rate
        + p_coefvar + n_syllables + duration + average_pause_duration + p_iqr + speech_ra
        te + syllables_phonation_duration?
#would it even make a difference? have I done this correctly?

to_remove <- names(df[-hc])
```

```
#trying to omit the correlated predictors from earlier, wound up creating a new recipe sin
        ce step_rm was finnicky
predictors <- this_recipe$var_info$variable
new_predictors <- setdiff(predictors, c(to_remove, "Diagnosis"))



cor_removed_recipe <-
  recipe(
    Diagnosis ~ ID + duration + phonation_duration + n_syllables + speech_rate + articulat
         ion_rate + syllables_phonation_duration + average_pause_duration + p_iqr + p_mad
         + p_coefvar + p_sd + p_min,
    data = training(split_df)
  ) %>% #had to do by hand unfortunately, couldnt figure it out otherwise
  step_normalize(all_numeric_predictors(),-all_outcomes()) %>%
  step_YeoJohnson(all_numeric_predictors(),-all_outcomes()) %>%
  update_role(ID, new_role = "ID")

summary(cor_removed_recipe)
```

```
## # A tibble: 14 × 4
##    variable                     type      role      source
##    <chr>                        <list>    <chr>     <chr>
##  1 ID                           <chr [3]> ID        original
##  2 duration                     <chr [2]> predictor original
##  3 phonation_duration           <chr [2]> predictor original
##  4 n_syllables                  <chr [2]> predictor original
##  5 speech_rate                  <chr [2]> predictor original
##  6 articulation_rate            <chr [2]> predictor original
##  7 syllables_phonation_duration <chr [2]> predictor original
##  8 average_pause_duration       <chr [2]> predictor original
##  9 p_iqr                        <chr [2]> predictor original
## 10 p_mad                        <chr [2]> predictor original
## 11 p_coefvar                    <chr [2]> predictor original
## 12 p_sd                         <chr [2]> predictor original
## 13 p_min                        <chr [2]> predictor original
## 14 Diagnosis                    <chr [3]> outcome   original
```

```
#this seems to make no difference, really
```

```
#new rf model to be upgraded with feature engineering, tripartite best
#also normalization

valid_split <- initial_validation_split(
   df,
  prop = c(0.7, 0.15),
  strata = Diagnosis
)
#splits into test, training AND validation

cor_removed_recipe <-
  recipe(
    Diagnosis ~ ID + duration + phonation_duration + n_syllables + speech_rate + articulat
         ion_rate + syllables_phonation_duration + average_pause_duration + p_iqr + p_mad
         + p_coefvar + p_sd + p_min,
    data = training(valid_split) #using validation here
  ) %>%
  step_normalize(all_numeric_predictors(),-all_outcomes()) %>%
  update_role(ID, new_role = "ID")
#normalizing the variables, center and scale them, since good for unsupervised learning? b
         ut might need to be better justified
#In general you should be careful about using -all_outcomes() if a *_predictors() selector
         would do what you want.


rf_flow <- workflow() %>%
  add_model(rand_forest(
    mode = "classification",
    engine = "randomForest",
    trees = tune(), #tune these
    min_n = tune()
  )) %>%
  add_recipe(cor_removed_recipe)

tripartite <- tune_grid(
  rf_flow,
  vfold_cv(validation(valid_split)),
  grid = grid_max_entropy(extract_parameter_set_dials(rf_flow), size = 10)
)
```

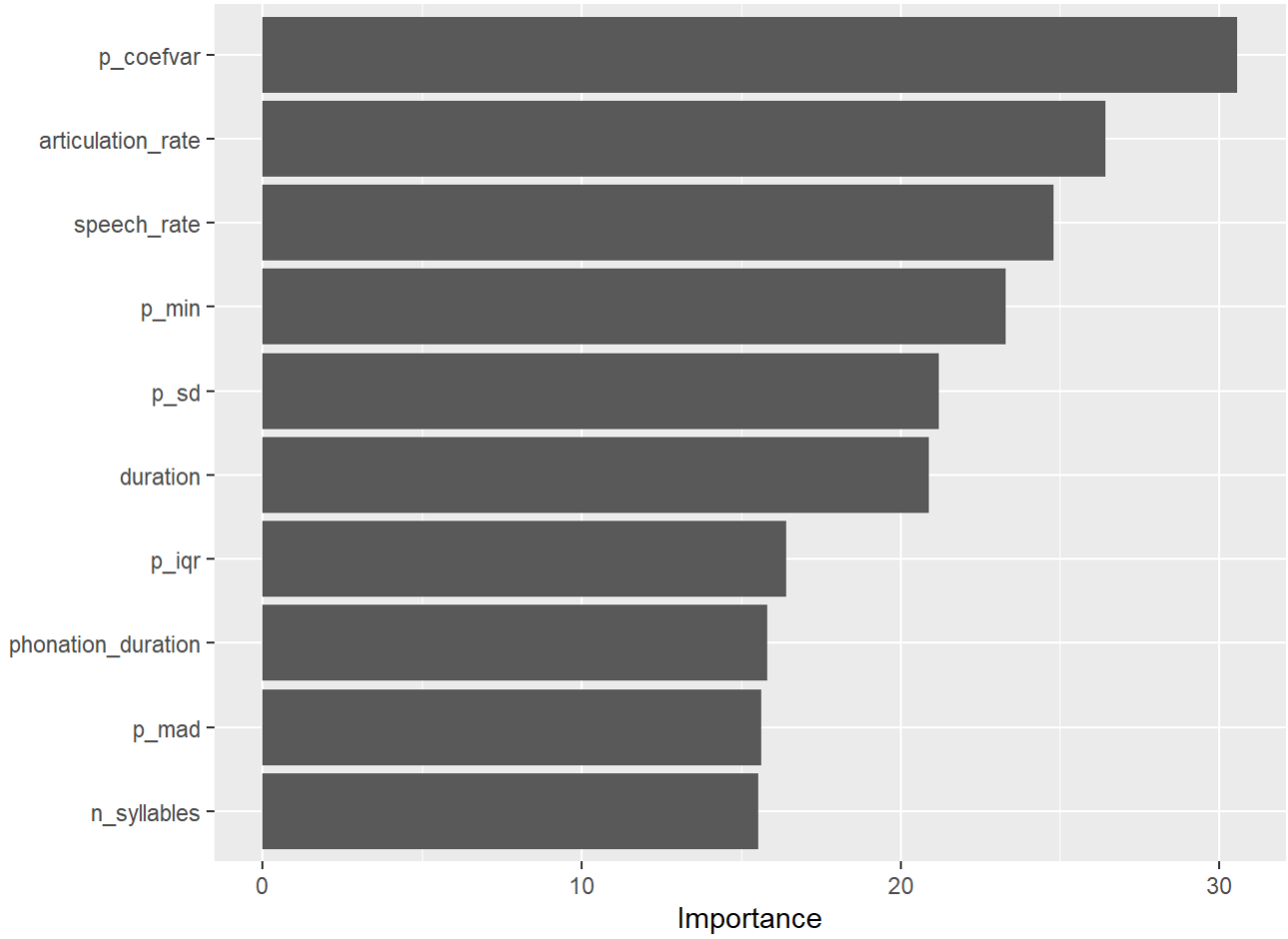
```
## Warning: package 'randomForest' was built under R version 4.3.2
```

```
tripartite %>% collect_metrics()
```

```
## # A tibble: 20 × 8
##    trees min_n .metric  .estimator  mean     n std_err .config
##    <int> <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1   310     2 accuracy binary     0.582    10  0.0201 Preprocessor1_Model01
##  2   310     2 roc_auc  binary     0.592    10  0.0433 Preprocessor1_Model01
##  3  1351    19 accuracy binary     0.547    10  0.0262 Preprocessor1_Model02
##  4  1351    19 roc_auc  binary     0.594    10  0.0405 Preprocessor1_Model02
##  5   655    26 accuracy binary     0.565    10  0.0347 Preprocessor1_Model03
##  6   655    26 roc_auc  binary     0.599    10  0.0418 Preprocessor1_Model03
##  7   907    40 accuracy binary     0.582    10  0.0428 Preprocessor1_Model04
##  8   907    40 roc_auc  binary     0.610    10  0.0435 Preprocessor1_Model04
##  9   310    13 accuracy binary     0.548    10  0.0361 Preprocessor1_Model05
## 10   310    13 roc_auc  binary     0.585    10  0.0395 Preprocessor1_Model05
## 11  1998    25 accuracy binary     0.564    10  0.0306 Preprocessor1_Model06
## 12  1998    25 roc_auc  binary     0.603    10  0.0431 Preprocessor1_Model06
## 13   958     3 accuracy binary     0.556    10  0.0283 Preprocessor1_Model07
## 14   958     3 roc_auc  binary     0.582    10  0.0410 Preprocessor1_Model07
## 15    75    38 accuracy binary     0.574    10  0.0432 Preprocessor1_Model08
## 16    75    38 roc_auc  binary     0.609    10  0.0410 Preprocessor1_Model08
## 17  1990    40 accuracy binary     0.590    10  0.0370 Preprocessor1_Model09
## 18  1990    40 roc_auc  binary     0.616    10  0.0418 Preprocessor1_Model09
## 19  1813     3 accuracy binary     0.573    10  0.0361 Preprocessor1_Model10
## 20  1813     3 roc_auc  binary     0.585    10  0.0423 Preprocessor1_Model10
```

```
#doing tripartite now
```

```
final_fit <- last_fit(
  finalize_workflow(rf_flow, select_best(tripartite, "roc_auc")),
  valid_split,
  add_validation_set = TRUE
)
#using the best tripartite and finalizing workplace, using validation set

#Plotting Variable Importance with vip
vip_plot <-
final_fit %>%
extract_fit_parsnip() %>%
vip::vip()
vip_plot
```




```
predicted <- final_fit %>%
  pull(.predictions) %>%
  bind_cols() %>%
  mutate(
    Predicted = .pred_class,
    Prob_SCZ = .pred_SCZ,
    type = "prob"
  )

conf_matrix <- confusionMatrix(predicted$Predicted, testing(valid_split)$Diagnosis)
conf_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction CTRL SCZ
##       CTRL   85  54
##       SCZ    43  54
##
##                Accuracy : 0.589
##                  95% CI : (0.5233, 0.6524)
##     No Information Rate : 0.5424
##     P-Value [Acc > NIR] : 0.08472
##
##                   Kappa : 0.1654
##
##  Mcnemar's Test P-Value : 0.30994
##
##             Sensitivity : 0.6641
##             Specificity : 0.5000
##          Pos Pred Value : 0.6115
##          Neg Pred Value : 0.5567
##              Prevalence : 0.5424
##          Detection Rate : 0.3602
##    Detection Prevalence : 0.5890
##       Balanced Accuracy : 0.5820
##
##        'Positive' Class : CTRL
##
```

```
predicted
```

```
## # A tibble: 236 × 9
##    .pred_CTRL .pred_SCZ  .row .pred_class Diagnosis .config   Predicted Prob_SCZ
##         <dbl>     <dbl> <int> <fct>       <fct>     <chr>     <fct>        <dbl>
##  1      0.407     0.593    18 SCZ         CTRL      Preproce… SCZ          0.593
##  2      0.426     0.574    22 SCZ         CTRL      Preproce… SCZ          0.574
##  3      0.348     0.652    28 SCZ         CTRL      Preproce… SCZ          0.652
##  4      0.554     0.446    33 CTRL        CTRL      Preproce… CTRL         0.446
##  5      0.298     0.702    35 SCZ         CTRL      Preproce… SCZ          0.702
##  6      0.268     0.732    37 SCZ         CTRL      Preproce… SCZ          0.732
##  7      0.720     0.280    44 CTRL        CTRL      Preproce… CTRL         0.280
##  8      0.508     0.492    62 CTRL        CTRL      Preproce… CTRL         0.492
##  9      0.711     0.289    73 CTRL        CTRL      Preproce… CTRL         0.289
## 10      0.843     0.157    74 CTRL        CTRL      Preproce… CTRL         0.157
## # ℹ 226 more rows
## # ℹ 1 more variable: type <chr>
```

```
#accuracy not much better?
```

3.3 Third phase: Another Algorithm

For this final part, add a supervised algorithm to the workflow set and compare its performance to the previous ones. Here again, you are free to choose any algorithm, but it its important that we know what you're doing and why you are doing it. In other words, tell us a bit about the algorithm you're using and why you chose it.

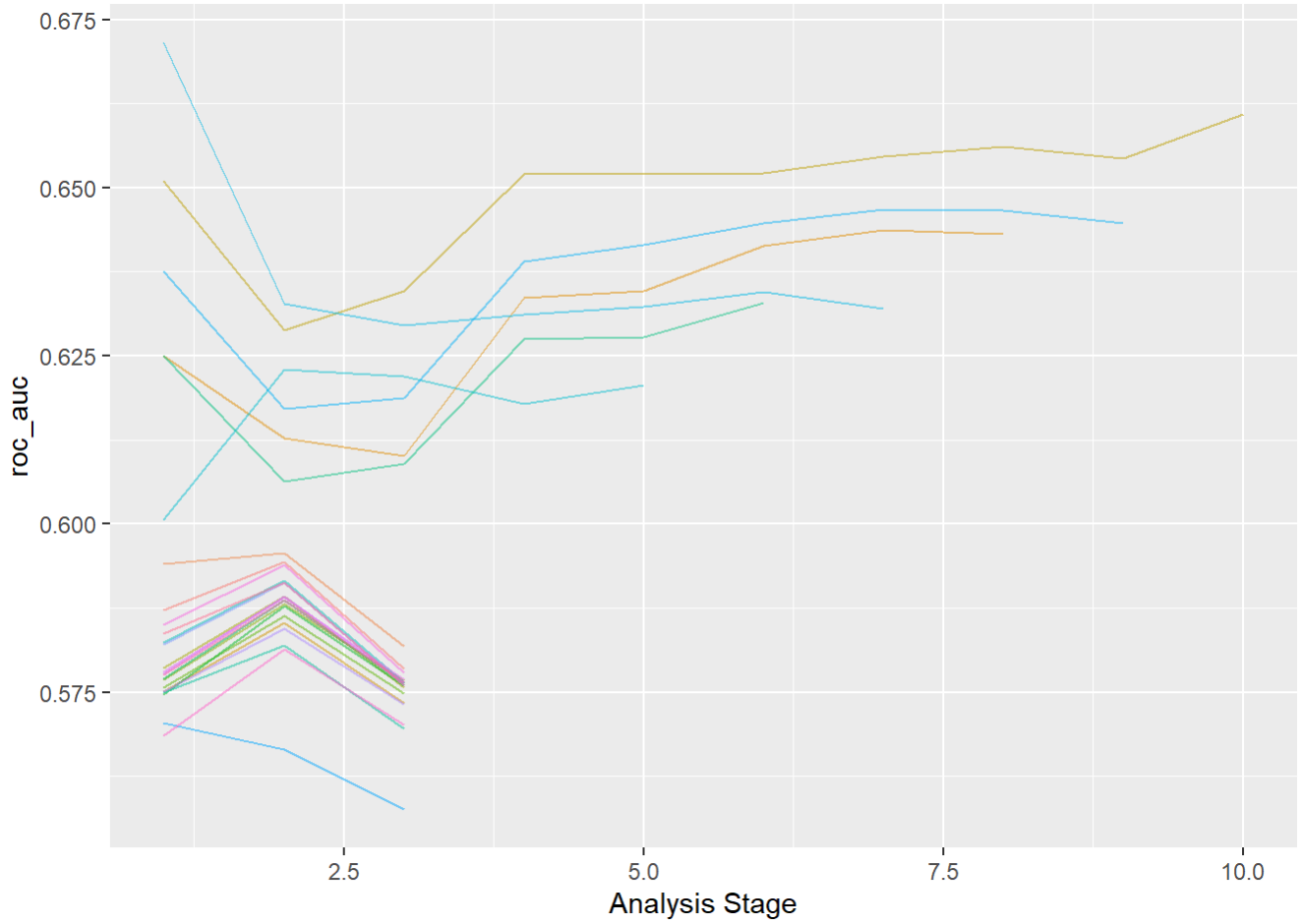
For a detailed list of the model types, engines, and arguments that can be used with the tidymodels framework, have a look here https://www.tidymodels.org/find/parsnip/#models (https://www.tidymodels.org/find/parsnip/#models)

```
#racing method with anova, not tuning the recipe though?

training_folds <- vfold_cv(df, v = 10) #splitting into folds for training

m1 <-
  svm_rbf(cost = tune(), rbf_sigma = tune()) %>%
  set_engine("kernlab") %>%
  set_mode("classification")

m1_grid <-
  m1 %>%
  hardhat::extract_parameter_set_dials() %>%
  grid_latin_hypercube(size = 25)

wflow <-
  workflow() %>%
  add_model(m1) %>%
  add_recipe(cor_removed_recipe)

anova_race <- wflow %>% tune_race_anova(resamples = training_folds, grid = m1_grid)
```

```
show_best(anova_race, metric = "roc_auc", n = 2)
```

```
## # A tibble: 1 × 8
##    cost rbf_sigma .metric .estimator  mean     n std_err .config
##   <dbl>     <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1  1.38    0.0160 roc_auc binary     0.661    10  0.0102 Preprocessor1_Model05
```

```
plot_race(anova_race)
```




```
best_model <- select_best(anova_race, metric = "roc_auc")

final_rf <- finalize_model(
  m1,
  best_model
) #am i using this bit correctly?

final_res <- cor_removed_recipe %>%
  workflow() %>%
  add_model(final_rf)

final_fit  <- final_res %>%
last_fit(split_df)

final_metrics <- final_fit %>%
  collect_metrics()

final_metrics
```

```
## # A tibble: 2 × 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy binary         0.636 Preprocessor1_Model1
## 2 roc_auc  binary         0.684 Preprocessor1_Model1
```

```r
#better performance than the later model? or more or less the same?

predicted <- data.frame(final_fit$.predictions)

conf_matrix <-
  confusionMatrix(
    data = predicted$.pred_class,  # Accessing the predictions column
    reference = predicted$Diagnosis,
    positive = "SCZ"
  )
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction CTRL SCZ
##       CTRL  202 118
##       SCZ    53  97
##
##                Accuracy : 0.6362
##                  95% CI : (0.5909, 0.6798)
##     No Information Rate : 0.5426
##     P-Value [Acc > NIR] : 2.461e-05
##
##                   Kappa : 0.2492
##
##  Mcnemar's Test P-Value : 9.871e-07
##
##             Sensitivity : 0.4512
##             Specificity : 0.7922
##          Pos Pred Value : 0.6467
##          Neg Pred Value : 0.6312
##              Prevalence : 0.4574
##          Detection Rate : 0.2064
##    Detection Prevalence : 0.3191
##       Balanced Accuracy : 0.6217
##
##        'Positive' Class : SCZ
##
```

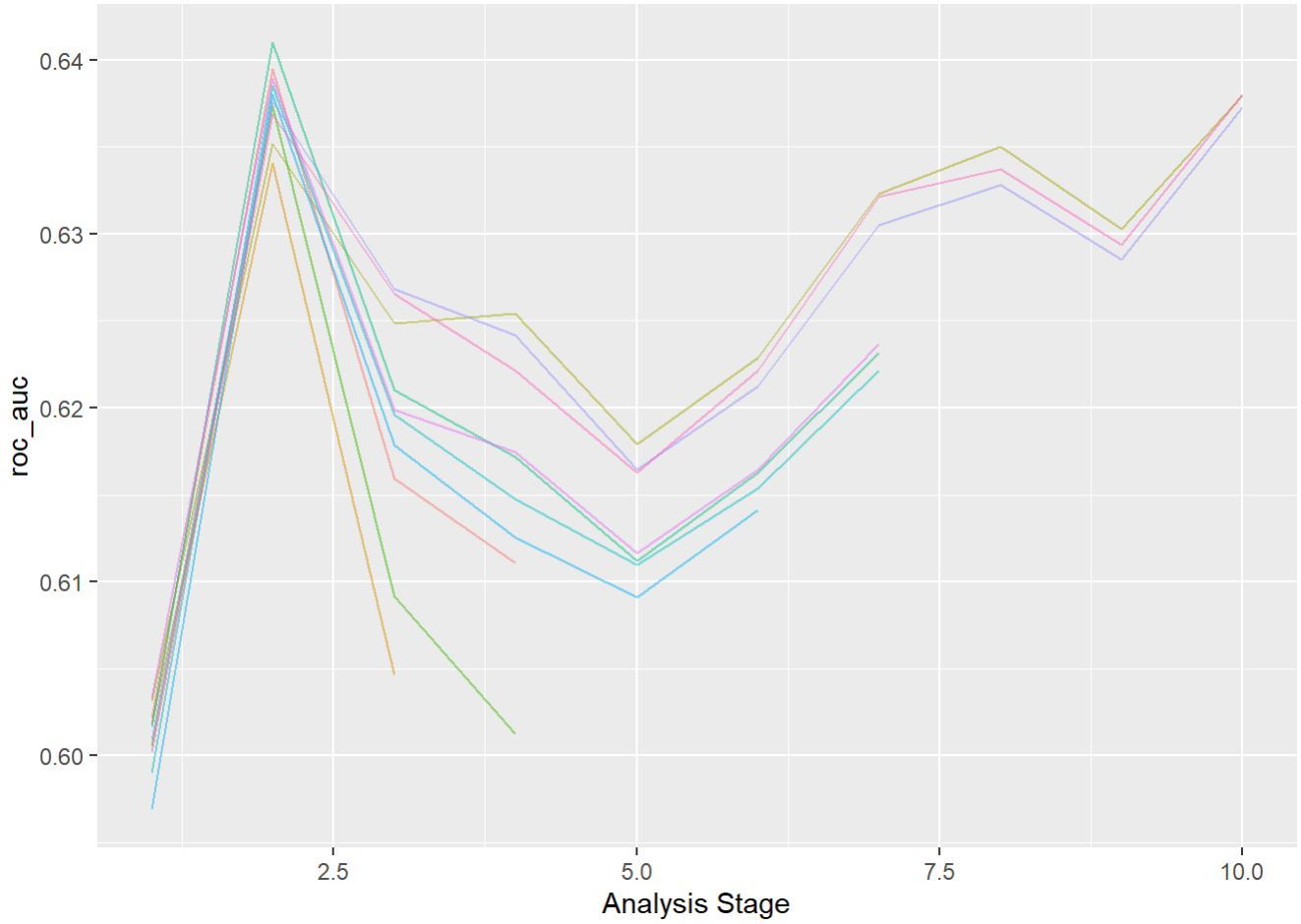
```r
rda_spec <- discrim_regularized(frac_common_cov = tune(), frac_identity = tune()) %>%
    set_engine("klaR")%>%
  set_mode("classification")

ctrl <- control_race(verbose_elim = TRUE)

grid_anova <- rda_spec %>%
    tune_race_anova(cor_removed_recipe, resamples = training_folds, grid = 10, control = c
        trl)
```

```
## ℹ Racing will maximize the roc_auc metric.
## ℹ Resamples are analyzed in a random order.
## ℹ Fold08: 1 eliminated; 9 candidates remain.
##
## ℹ Fold07: 2 eliminated; 7 candidates remain.
##
## ℹ Fold06: 0 eliminated; 7 candidates remain.
##
## ℹ Fold10: 1 eliminated; 6 candidates remain.
##
## ℹ Fold09: 3 eliminated; 3 candidates remain.
##
## ℹ Fold05: 0 eliminated; 3 candidates remain.
##
## ℹ Fold02: 0 eliminated; 3 candidates remain.
```

```
show_best(grid_anova, metric = "roc_auc", n = 2)
```

```
## # A tibble: 2 × 8
##   frac_common_cov frac_identity .metric .estimator  mean     n std_err .config
##             <dbl>         <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1           0.665        0.0808 roc_auc binary     0.638    10  0.0141 Preproce…
## 2           0.103        0.292  roc_auc binary     0.638    10  0.0132 Preproce…
```

```
plot_race(grid_anova)
```




```
best_model <- select_best(grid_anova, metric = "roc_auc")

final_model <- finalize_model(
  rda_spec,
  best_model
) #is this a correct way to finalize?

final_res <- cor_removed_recipe %>%
  workflow() %>%
  add_model(final_model)

final_fit  <- final_res %>%
last_fit(split_df)

predicted <- data.frame(final_fit$.predictions)

conf_matrix <-
  confusionMatrix(
    data = predicted$.pred_class,  # Accessing the predictions column
    reference = predicted$Diagnosis,
    positive = "SCZ"
  )
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction CTRL SCZ
##       CTRL  198 110
##       SCZ    57 105
##
##                Accuracy : 0.6447
##                  95% CI : (0.5995, 0.688)
##     No Information Rate : 0.5426
##     P-Value [Acc > NIR] : 4.550e-06
##
##                   Kappa : 0.2701
##
##  Mcnemar's Test P-Value : 5.725e-05
##
##             Sensitivity : 0.4884
##             Specificity : 0.7765
##          Pos Pred Value : 0.6481
##          Neg Pred Value : 0.6429
##              Prevalence : 0.4574
##          Detection Rate : 0.2234
##    Detection Prevalence : 0.3447
##       Balanced Accuracy : 0.6324
##
##        'Positive' Class : SCZ
##
```

```
pacman::p_load("gbm") #trying out gbm

ctrl <- trainControl(method = "cv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)


model <- train(
  cor_removed_recipe,
  data = training(split_df),
  method = "gbm",
  trControl = ctrl,
  tuneLength = 5,
  metric = "ROC",
  verbose = FALSE
)
```

```
## --------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

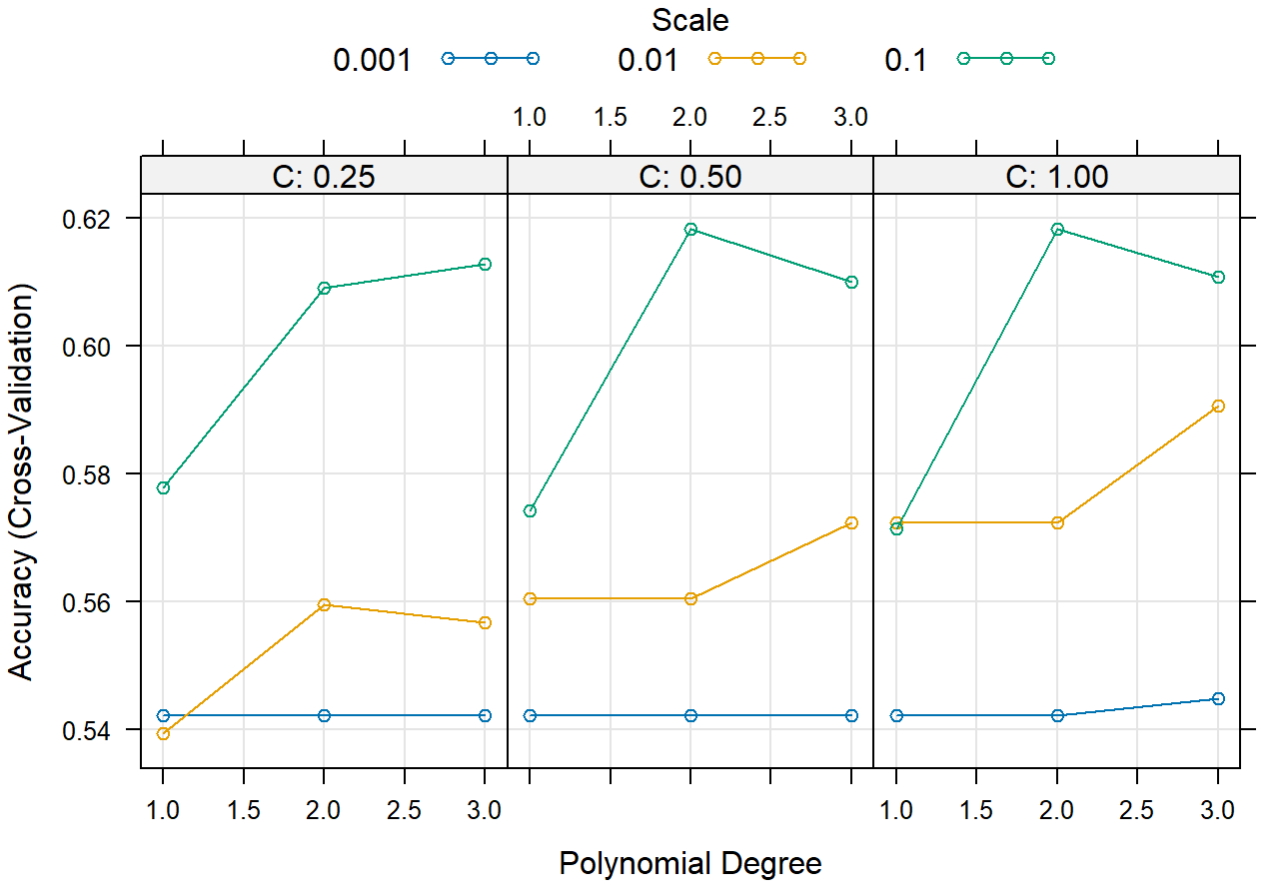
```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following object is masked from 'package:purrr':
##
##     compact
```

```
predictions <- predict(model, newdata = testing(split_df))

predicted_labels <- factor(predictions, levels = c("SCZ", "CTRL"))
actual_labels <- factor(testing(split_df)$Diagnosis, levels = c("SCZ", "CTRL"))

conf_matrix <- confusionMatrix(data = predicted_labels, reference = actual_labels)
conf_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction SCZ CTRL
##       SCZ  104   78
##       CTRL 111  177
##
##                Accuracy : 0.5979
##                  95% CI : (0.552, 0.6425)
##     No Information Rate : 0.5426
##     P-Value [Acc > NIR] : 0.008917
##
##                   Kappa : 0.18
##
##  Mcnemar's Test P-Value : 0.019930
##
##             Sensitivity : 0.4837
##             Specificity : 0.6941
##          Pos Pred Value : 0.5714
##          Neg Pred Value : 0.6146
##              Prevalence : 0.4574
##          Detection Rate : 0.2213
##    Detection Prevalence : 0.3872
##       Balanced Accuracy : 0.5889
##
##        'Positive' Class : SCZ
##
```

```
svmPoly_model <- train(
  cor_removed_recipe, #normalization; is that a smart choice? was removing correlates ok?
         did it make a difference
  data = training(split_df),
  method = "svmPoly",
  trControl = trainControl(method = "cv"), #since cv, couldnt i use the whole data?
  tuneLength = 3, #lowered it
  verbose = F
)
plot(svmPoly_model)
```




```
max(svmPoly_model$results$Accuracy)
```

```
## [1] 0.6182152
```

```
svmPoly_predict <- predict(svmPoly_model, newdata = testing(split_df))
conf_matrix <- confusionMatrix(data = svmPoly_predict, reference = testing(split_df)$Diagn
        osis, positive = "SCZ")
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction CTRL SCZ
##       CTRL  191 115
##       SCZ    64 100
##
##                Accuracy : 0.6191
##                  95% CI : (0.5735, 0.6632)
##     No Information Rate : 0.5426
##     P-Value [Acc > NIR] : 0.0004734
##
##                   Kappa : 0.2182
##
##  Mcnemar's Test P-Value : 0.0001861
##
##             Sensitivity : 0.4651
##             Specificity : 0.7490
##          Pos Pred Value : 0.6098
##          Neg Pred Value : 0.6242
##              Prevalence : 0.4574
##          Detection Rate : 0.2128
##    Detection Prevalence : 0.3489
##       Balanced Accuracy : 0.6071
##
##        'Positive' Class : SCZ
##
```

```
final_model_svm <-
  svm_rbf() %>% #using svm
  set_engine("kernlab") %>%
parsnip::set_mode("classification")

final_wf <- workflow() %>%
  add_model(final_model_svm)%>%
  add_recipe(cor_removed_recipe)

rf_fit <- final_wf %>% fit(training(split_df))

max(model$results$ROC)
```

```
## [1] 0.6472831
```

```
#names(getModelInfo())

predict <- predict(model, newdata = testing(split_df))
conf_matrix <- confusionMatrix(data = predict, reference = testing(split_df)$Diagnosis, po
         sitive = "SCZ")
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction CTRL SCZ
##       CTRL  177 111
##       SCZ    78 104
##
##                Accuracy : 0.5979
##                  95% CI : (0.552, 0.6425)
##     No Information Rate : 0.5426
##     P-Value [Acc > NIR] : 0.008917
##
##                   Kappa : 0.18
##
##  Mcnemar's Test P-Value : 0.019930
##
##             Sensitivity : 0.4837
##             Specificity : 0.6941
##          Pos Pred Value : 0.5714
##          Neg Pred Value : 0.6146
##              Prevalence : 0.4574
##          Detection Rate : 0.2213
##    Detection Prevalence : 0.3872
##       Balanced Accuracy : 0.5889
##
##        'Positive' Class : SCZ
##
```

```
predict <- predict(rf_fit, new_data = testing(split_df))
conf_matrix <- confusionMatrix(data = predict$.pred_class, reference = testing(split_df)$D
         iagnosis, positive = "SCZ")
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction CTRL SCZ
##       CTRL  174 101
##       SCZ    81 114
##
##                Accuracy : 0.6128
##                  95% CI : (0.5671, 0.657)
##     No Information Rate : 0.5426
##     P-Value [Acc > NIR] : 0.001246
##
##                   Kappa : 0.2141
##
##  Mcnemar's Test P-Value : 0.159020
##
##             Sensitivity : 0.5302
##             Specificity : 0.6824
##          Pos Pred Value : 0.5846
##          Neg Pred Value : 0.6327
##              Prevalence : 0.4574
##          Detection Rate : 0.2426
##    Detection Prevalence : 0.4149
##       Balanced Accuracy : 0.6063
##
##        'Positive' Class : SCZ
##
```

4 Discussion: Methodology and Results

4.1 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:

I messed around a lot with my models and as a result I have a lot of filler and some of it may be messy and even incorrectly done.

Anyway, I attempted the following:

I figured the parameters available to us likely have high correlation and that this might be a problem, as the model would simply overfit on the training data parameters in such a way where it does not have a clear understanding of how the variables appear in nature; i.e. it cannot distinguish the real influence/importance of each parameter. As such, I tried to cull them by calculating the correlations and omitting any over 0.5, and making these the predictors. I did this in a way that might not be entirely optimal; I assume a ML solution with tuning the recipe would've sufficed better.

I normalized the recipe predictors and used yeojohnson as I figured that would help with the lack of normality the data had, squishing it into a narrower range. Perhaps I should've done more pre-processing.

Afterwards, I messed about with various ML methods, such as tuning a tripartite and picking the best model, anova racing using kernlab and klaR engines, gbm (gradient boosting machine), svmPoly (support vector machine model). They gave mostly the same results of 60-65% accuracy and similar ROC. It's possible that I was doing them incorrectly, or that one of my pre-processing steps, such as the recipe I chose, may have lessened the data/training quality.

4.2 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?

It appears that I am not yet capable of actually creating or tuning a model to be better than the default; i.e. I was stuck at around 64% throughout my attempts, which was not better than what I had in the 2nd part. I assume there's something I'm still missing or that the act of tuning and training a model takes extensive trial and error alongside statistical knowledge–that is, choices that are justified with more than idle curiosity and have some sort of knowledge of the data distribution or statistical methods behind them (the latter two I am still lacking in).

That said, it could be not worth the effort for this task–I'm not sure, though I do suspect it isn't.

4.3 How do the predictive models you built relate to the descriptive analysis conducted in the second section?

I struggled with tying the predictive models to the descriptive analysis, but I did attempt to remove the features that I imagined to be less useful for prediction; that is, the features that correlated heavily with each other. Many of these were "not statistically significant" when I did a quick Wilcox on them earlier on. I'm not sure whether a quick statistical significance test is warranted, especially considering that I did not check assumptions; I just figured the "statistically likely" values were the ones likely to differentiate the groups from each other, and therefore be useful for a classification task.

It does appear that I may have been correct about the SCZ and CTRL speech and pitch parameters overlapping too much on the density plot for it to be a slam dunk classification task.

4.4 What is the explanatory scope of the analyses and procedures you conducted here, if any?

From the visual analyses alone, it is rather apparent that schizophrenic and non-schizophrenic individuals have more or less the same "distribution" of vocal characteristics outside of schizophrenics "trending towards the bottom" on variables such as pitch and number of syllables spoken. It is very likely that, as a result, it is rather difficult to distinguish one from the other unless you happen to have a schizophrenic

with a very “stereotypically extreme” distribution of pitch and speech characteristics. As such, I’m not sure whether speech characteristics are enough alone for the sake of distinguishing the two from each other (reliably anyway).

While it’s possible my models were all just poorly optimized and handled, it seems to me that a percentage of much over 70% would be overfitting on the training data/data available to the study. I suppose it could still be used as one of many markers of schizophrenia though, just not as a standalone one.