Spline plotting and SSANOVA

Stefano Coretta

27/12/2016

1 Data import

Before importing the data, we need to specify the column names for the data set header, since the raw data does not have a header. The number of splines is always 42 (saved as `num.splines`). `first.columns` is a factor containing the names of columns that are not the splines coordinates columns. Finally, we can concatenate `first.columns` with the names of the splines coordinates which is generated by the `paste0` function in the format `X_1, Y_1, X_2, Y_2, X_n, ...` (the underscore `_` will be useful for separating the axis from the fan number).

```
num.splines <- 42
first.columns <- c(
    "speaker",
    "seconds",
    "rec.date",
    "prompt",
    "label",
    "TT.displacement.sm",
    "TT.velocity",
    "TT.velocity.abs",
    "TD.displacement.sm",
    "TD.velocity",
    "TD.velocity.abs"
    )
columns <- c(first.columns,
             paste0(rep(c("X", "Y"), num.splines),
                    "_",
                    rep(1:num.splines, each = 2)
                    )
             )
```

We can now read in the file.

```
splines.raw <- list.files(path = "./pilot/results",
                  pattern = "*-aaa.txt",
                  full.names = TRUE) %>%
    map_df(~read_tsv(., col_names = columns, na = "*"))
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   speaker = col_character(),
##   rec.date = col_character(),
##   prompt = col_character(),
##   label = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
## Parsed with column specification:
```

```
## cols(
##   .default = col_double(),
##   speaker = col_character(),
##   rec.date = col_character(),
##   prompt = col_character(),
##   label = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   speaker = col_character(),
##   rec.date = col_character(),
##   prompt = col_character(),
##   label = col_character(),
##   X_2 = col_character(),
##   Y_2 = col_character(),
##   X_3 = col_character(),
##   Y_3 = col_character(),
##   X_4 = col_character(),
##   Y_4 = col_character(),
##   X_5 = col_character(),
##   Y_5 = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
rm(num.splines, first.columns, columns)

languages <- read_csv("./pilot/stimuli/languages.csv")
```

```
## Parsed with column specification:
## cols(
##   speaker = col_character(),
##   language = col_character()
## )
```

```
words <- read_csv("./pilot/stimuli/nonce.csv")
```

```
## Parsed with column specification:
## cols(
##   item = col_integer(),
##   word = col_character(),
##   ipa = col_character(),
##   c1 = col_character(),
##   c1phonation = col_character(),
##   vowel = col_character(),
##   anteropost = col_character(),
##   height = col_character(),
##   c2 = col_character(),
##   c2phonation = col_character(),
##   c2place = col_character(),
##   language = col_character()
## )
```

The following code applies tidy formatting to the data frame. It uses functions from the `tidyr` library.

```
splines <- splines.raw %>%
    gather(spline, coordinate, matches("[XY]_")) %>%
    separate(spline, c("axis", "fan"), convert = TRUE) %>%
    spread(axis, coordinate) %>%
    mutate(word = word(prompt, 2)) %>%
    left_join(y = languages) %>%
    left_join(y = words) %>%
    mutate_if(is.character, as.factor)
```

```
## Joining, by = "speaker"
```

```
## Joining, by = c("word", "language")
```

```
splines.it <- filter(splines, language == "italian")
splines.pl <- filter(splines, language == "polish")
```

2 Some plotting

We can finally plot splines.

```
filter(splines, label == "max_TD") %>%
ggplot(aes(x = X, y = Y, colour = c2phonation)) +
    geom_smooth(method = "loess") +
    coord_fixed(ratio = 1)
```

```
filter(splines, label == "max_TT") %>%
ggplot(aes(x = X, y = Y, colour = c2phonation)) +
    geom_smooth(method = "loess") +
    coord_fixed(ratio = 1)
```

3 SSANOVA

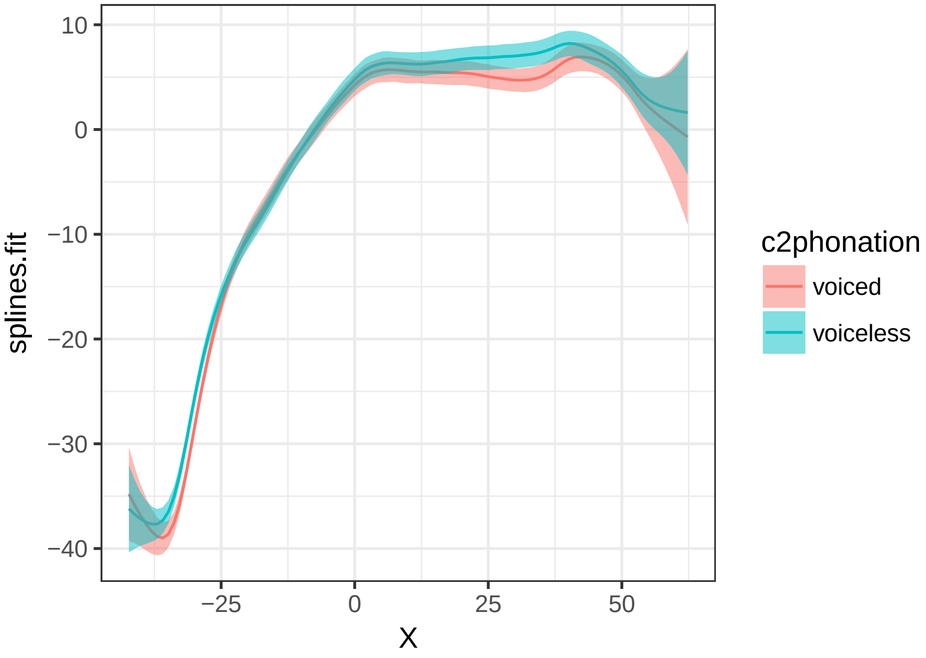
```
ssanovaTongue <- function(splines) {
    model <- ssanova(Y ~ c2phonation + X + c2phonation:X, data = splines)

    predicted <- expand.grid(X = seq(min(splines$X, na.rm = TRUE),
                                     max(splines$X, na.rm = TRUE),
                                     length = 100),
                             Y = seq(min(splines$Y, na.rm = TRUE),
                                     max(splines$Y, na.rm = TRUE),
                                     length = 100),
                             c2phonation = c("voiced", "voiceless")
    )

    predicted$splines.fit <- predict(model, predicted, se = T)$fit
    predicted$splines.SE <- predict(model, predicted, se = T)$se.fit

    return(predicted)
}
```

```
splines.cor.a <- filter(splines.it, vowel == "a", label == "max_TT")
ssanova.cor.a <- ssanovaTongue(splines.cor.a)

splines.cor.o <- filter(splines.it, vowel == "o", label == "max_TT")
ssanova.cor.o <- ssanovaTongue(splines.cor.a)

splines.cor.u <- filter(splines.it, vowel == "u", label == "max_TT")
ssanova.cor.u <- ssanovaTongue(splines.cor.u)


splines.vel.a <- filter(splines.it, vowel == "a", label == "max_TD")
ssanova.vel.a <- ssanovaTongue(splines.vel.a)
```

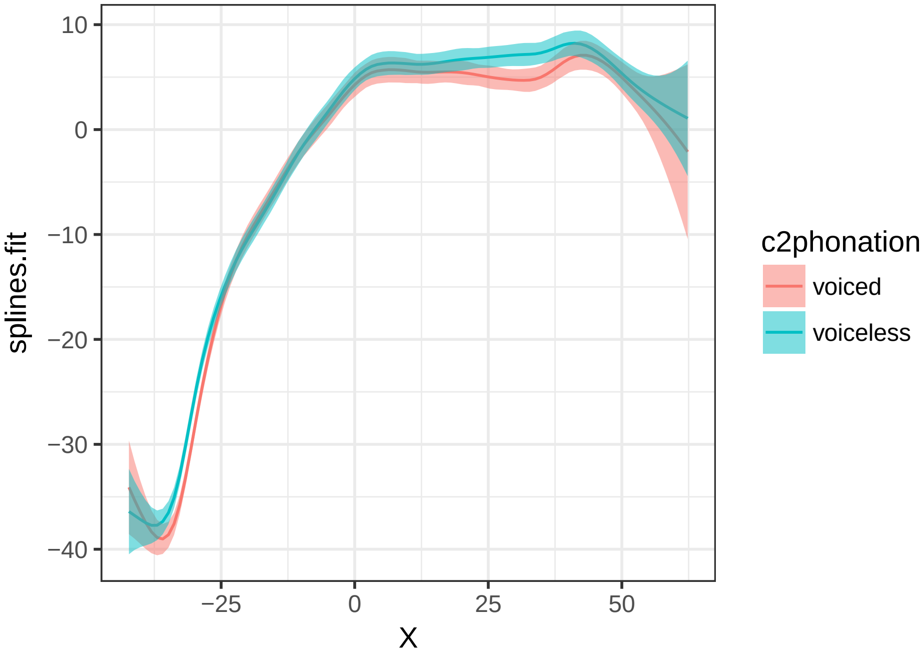
```
## Warning in chol.default(wk1, pivot = TRUE): the matrix is either rank-
## deficient or indefinite
```

```
splines.vel.o <- filter(splines.it, vowel == "o", label == "max_TD")
ssanova.vel.o <- ssanovaTongue(splines.vel.a)

splines.vel.u <- filter(splines.it, vowel == "u", label == "max_TD")
ssanova.vel.u <- ssanovaTongue(splines.vel.u)
```

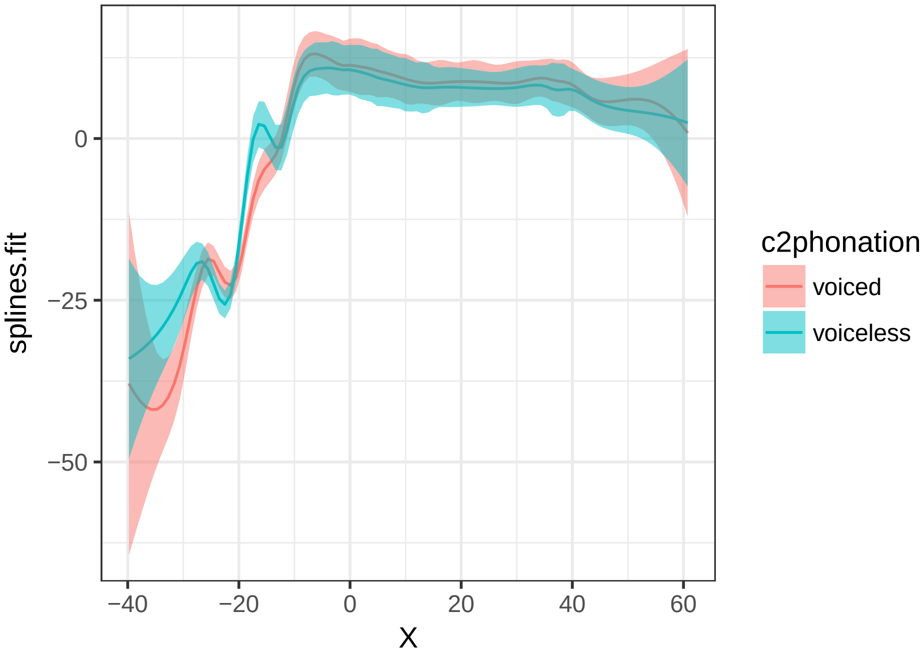
```
ggplot(ssanova.cor.a, aes(x = X, colour = c2phonation)) +
geom_line(aes(y = splines.fit)) +
geom_ribbon(aes(ymin = splines.fit - (1.96 * splines.SE),
                ymax = splines.fit + (1.96 * splines.SE),
                fill = c2phonation
                ),
            alpha = 0.5,
            color = "NA"
            )
```



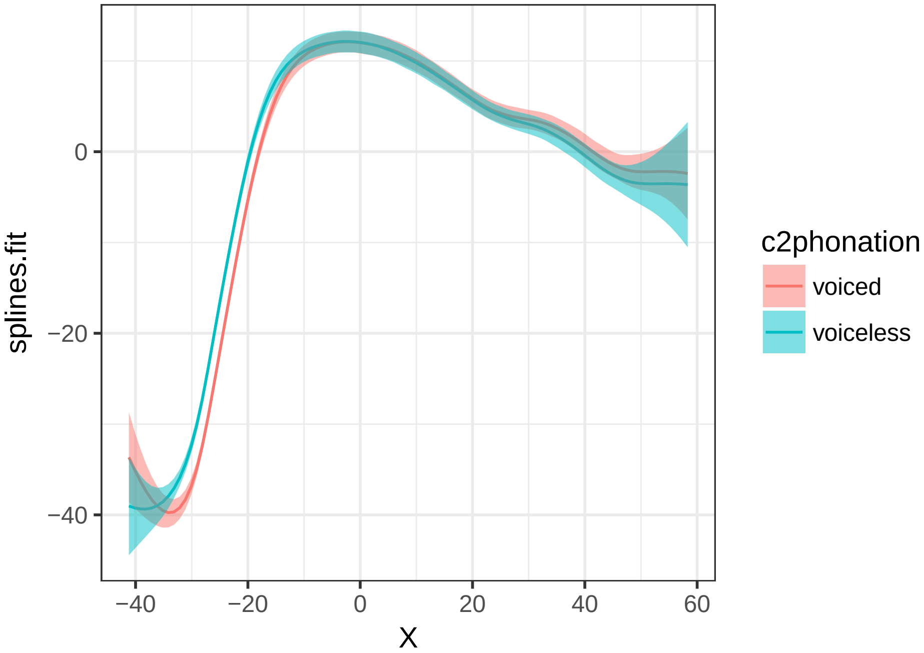
```
ggplot(ssanova.cor.o, aes(x = X, colour = c2phonation)) +
geom_line(aes(y = splines.fit)) +
geom_ribbon(aes(ymin = splines.fit - (1.96 * splines.SE),
                ymax = splines.fit + (1.96 * splines.SE),
                fill = c2phonation
                ),
            alpha = 0.5,
            color = "NA"
            )
```



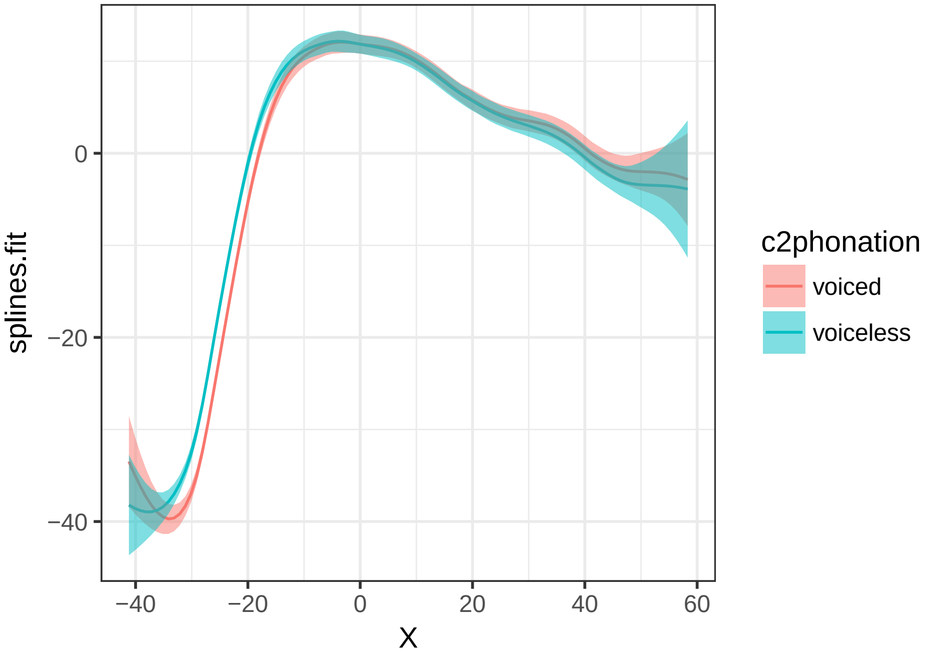
```
ggplot(ssanova.cor.u, aes(x = X, colour = c2phonation)) +
geom_line(aes(y = splines.fit)) +
geom_ribbon(aes(ymin = splines.fit - (1.96 * splines.SE),
                ymax = splines.fit + (1.96 * splines.SE),
                fill = c2phonation
                ),
            alpha = 0.5,
            color = "NA"
            )
```



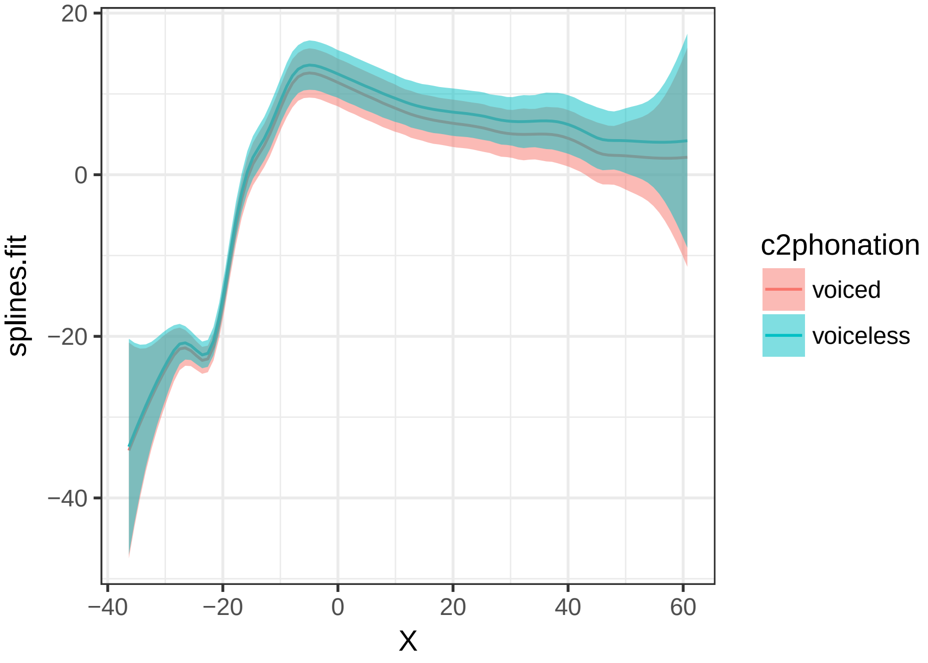
```
ggplot(ssanova.vel.a, aes(x = X, colour = c2phonation)) +
geom_line(aes(y = splines.fit)) +
geom_ribbon(aes(ymin = splines.fit - (1.96 * splines.SE),
                ymax = splines.fit + (1.96 * splines.SE),
                fill = c2phonation
                ),
            alpha = 0.5,
            color = "NA"
            )
```



```
ggplot(ssanova.vel.o, aes(x = X, colour = c2phonation)) +
geom_line(aes(y = splines.fit)) +
geom_ribbon(aes(ymin = splines.fit - (1.96 * splines.SE),
                ymax = splines.fit + (1.96 * splines.SE),
                fill = c2phonation
                ),
            alpha = 0.5,
            color = "NA"
            )
```

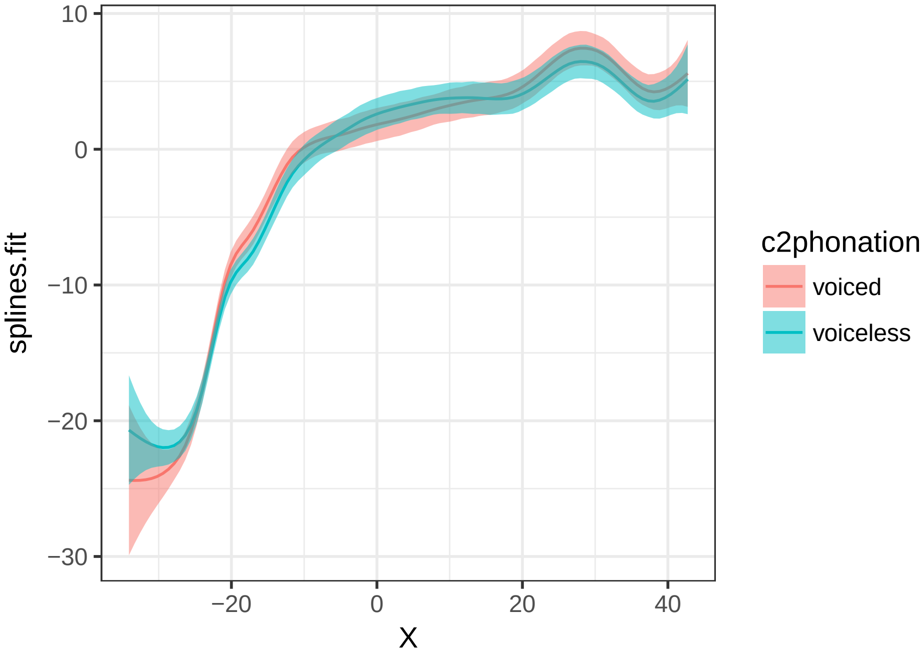


```
ggplot(ssanova.vel.u, aes(x = X, colour = c2phonation)) +
geom_line(aes(y = splines.fit)) +
geom_ribbon(aes(ymin = splines.fit - (1.96 * splines.SE),
                ymax = splines.fit + (1.96 * splines.SE),
                fill = c2phonation
                ),
            alpha = 0.5,
            color = "NA"
            )
```

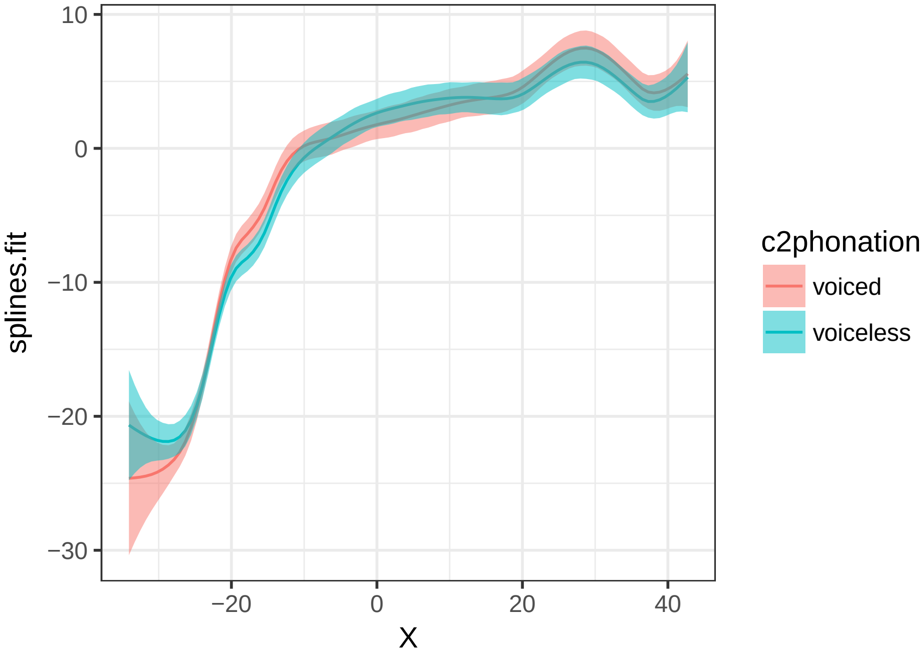


```r
splines.cor.a <- filter(splines.pl, vowel == "a", label == "max_TT")
ssanova.cor.a <- ssanovaTongue(splines.cor.a)

splines.cor.o <- filter(splines.pl, vowel == "o", label == "max_TT")
ssanova.cor.o <- ssanovaTongue(splines.cor.a)

splines.cor.u <- filter(splines.pl, vowel == "u", label == "max_TT")
ssanova.cor.u <- ssanovaTongue(splines.cor.u)


splines.vel.a <- filter(splines.pl, vowel == "a", label == "max_TD")
ssanova.vel.a <- ssanovaTongue(splines.vel.a)

splines.vel.o <- filter(splines.pl, vowel == "o", label == "max_TD")
ssanova.vel.o <- ssanovaTongue(splines.vel.a)

splines.vel.u <- filter(splines.pl, vowel == "u", label == "max_TD")
ssanova.vel.u <- ssanovaTongue(splines.vel.u)
```

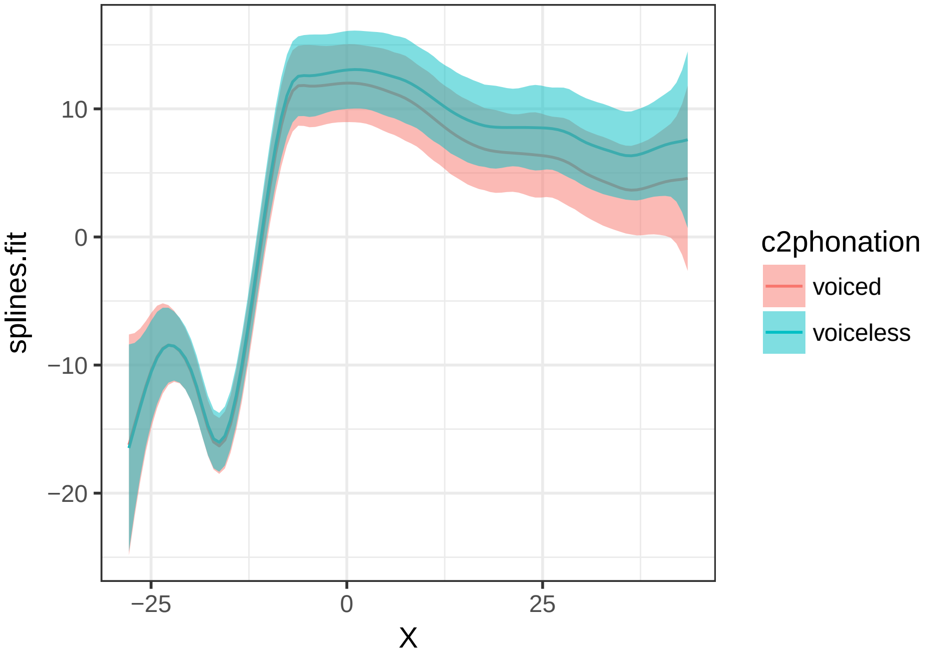
```r
ggplot(ssanova.cor.a, aes(x = X, colour = c2phonation)) +
geom_line(aes(y = splines.fit)) +
geom_ribbon(aes(ymin = splines.fit - (1.96 * splines.SE),
                ymax = splines.fit + (1.96 * splines.SE),
                fill = c2phonation
                ),
            alpha = 0.5,
            color = "NA"
            )
```



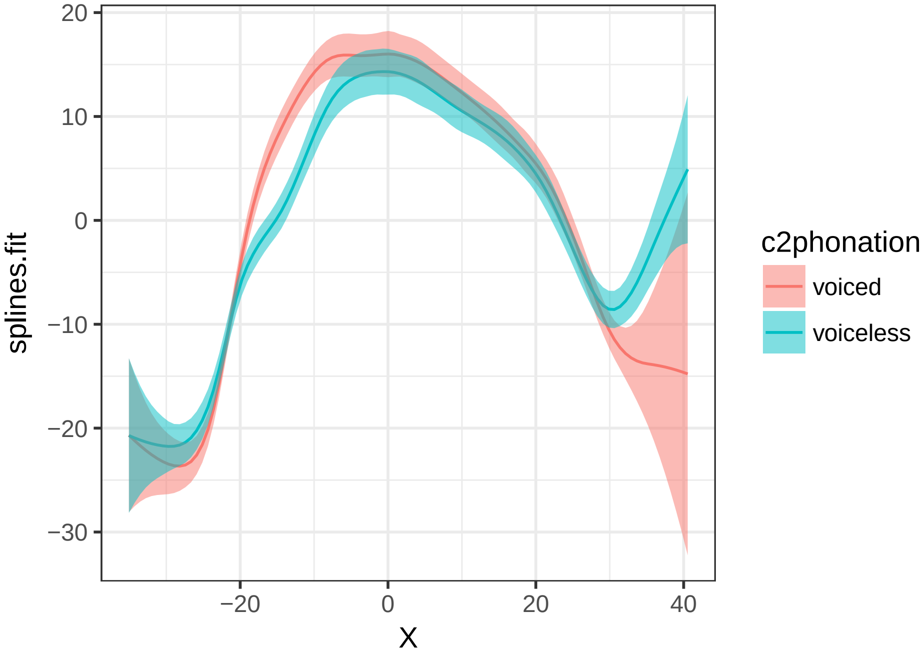
```
ggplot(ssanova.cor.o, aes(x = X, colour = c2phonation)) +
geom_line(aes(y = splines.fit)) +
geom_ribbon(aes(ymin = splines.fit - (1.96 * splines.SE),
                ymax = splines.fit + (1.96 * splines.SE),
                fill = c2phonation
                ),
            alpha = 0.5,
            color = "NA"
            )
```



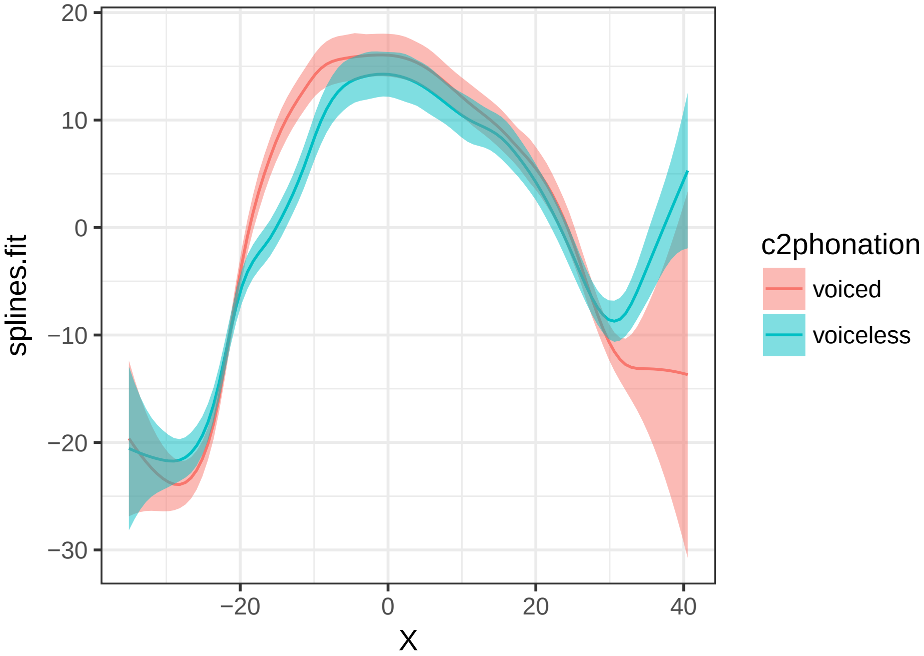
```
ggplot(ssanova.cor.u, aes(x = X, colour = c2phonation)) +
geom_line(aes(y = splines.fit)) +
geom_ribbon(aes(ymin = splines.fit - (1.96 * splines.SE),
                ymax = splines.fit + (1.96 * splines.SE),
                fill = c2phonation
                ),
            alpha = 0.5,
            color = "NA"
            )
```



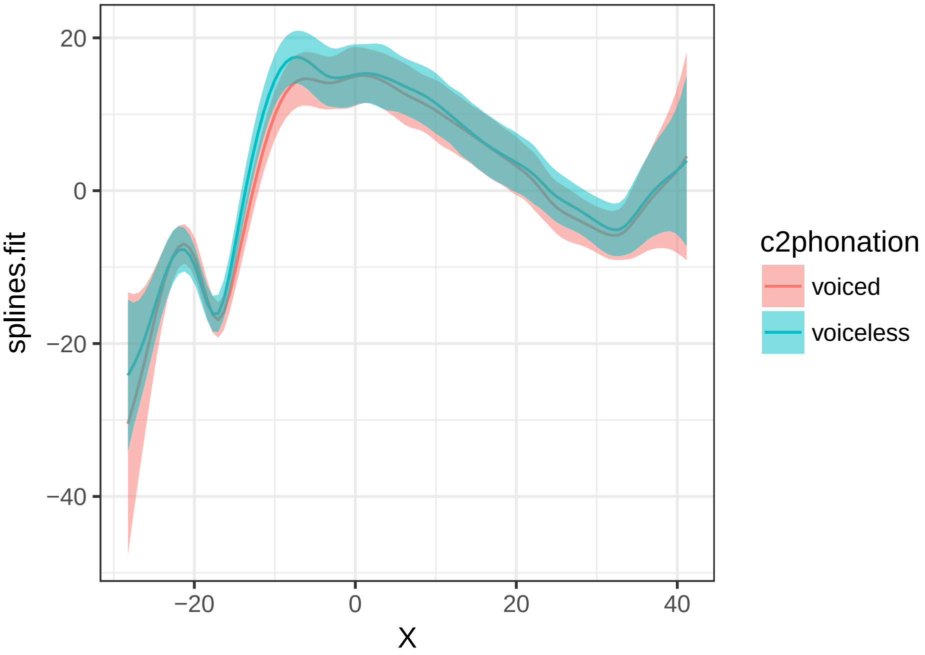
```
ggplot(ssanova.vel.a, aes(x = X, colour = c2phonation)) +
geom_line(aes(y = splines.fit)) +
geom_ribbon(aes(ymin = splines.fit - (1.96 * splines.SE),
                ymax = splines.fit + (1.96 * splines.SE),
                fill = c2phonation
                ),
            alpha = 0.5,
            color = "NA"
            )
```



```r
ggplot(ssanova.vel.o, aes(x = X, colour = c2phonation)) +
geom_line(aes(y = splines.fit)) +
geom_ribbon(aes(ymin = splines.fit - (1.96 * splines.SE),
                ymax = splines.fit + (1.96 * splines.SE),
                fill = c2phonation
                ),
            alpha = 0.5,
            color = "NA"
            )
```



```r
ggplot(ssanova.vel.u, aes(x = X, colour = c2phonation)) +
geom_line(aes(y = splines.fit)) +
geom_ribbon(aes(ymin = splines.fit - (1.96 * splines.SE),
                ymax = splines.fit + (1.96 * splines.SE),
                fill = c2phonation
                ),
            alpha = 0.5,
            color = "NA"
            )
```



4 Consonantal gestures: target, maximum, release

Now we can create a separate data frame where the observasional unit is each word and the variables are the consonantal getures (target, maximum closure, release).

```
cons.gestures <- select(splines.raw, speaker:label) %>%
    mutate(word = word(prompt, 2)) %>%
    left_join(y = languages) %>%
    left_join(y = words) %>%
    separate(label, c("gesture", "tongue.area")) %>%
    spread(gesture, seconds) %>%
    mutate(nucleus = NOFF - NONS, deceleration = max - NONS,
           acceleration = NOFF - max, skewness = deceleration / nucleus) %>%
    mutate_if(is.character, as.factor)
```

```
## Joining, by = "speaker"
```

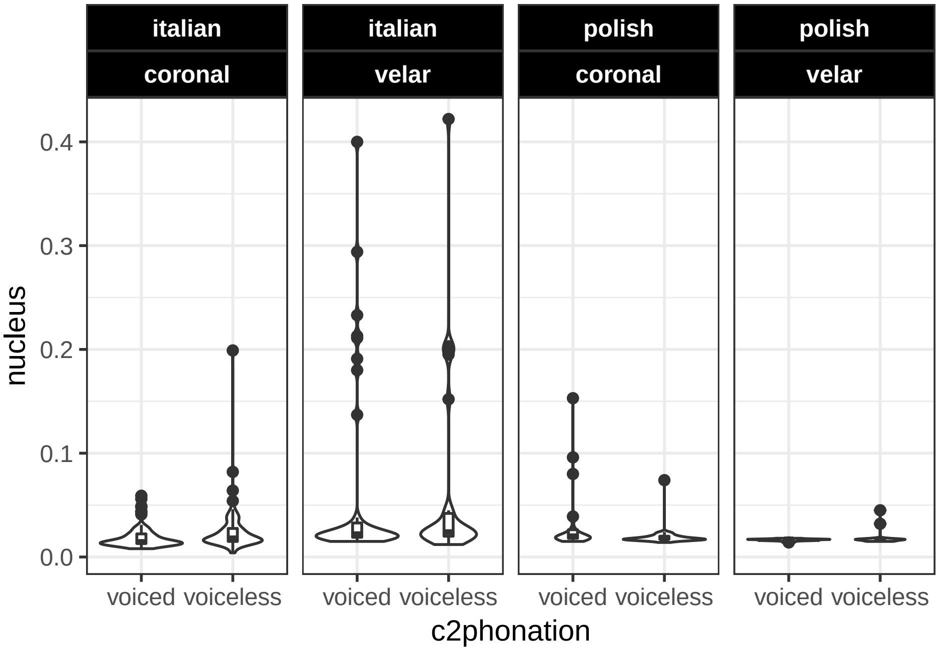
```
## Joining, by = c("word", "language")
```

Let's plot the nucleus duration (NOFF - NONS) as a function of place of articulation, voicing of C2 (our target consonant in C1VC2V words), and language.

```
ggplot(cons.gestures, aes(c2phonation, nucleus)) +
    facet_grid(.~language + c2place) +
    geom_violin() +
    geom_boxplot(width=0.1) +
    theme(strip.background = element_rect(fill = "black"),
          strip.text = element_text(color = "white", face = "bold")
          )
```

```
## Warning: Removed 88 rows containing non-finite values (stat_ydensity).
```

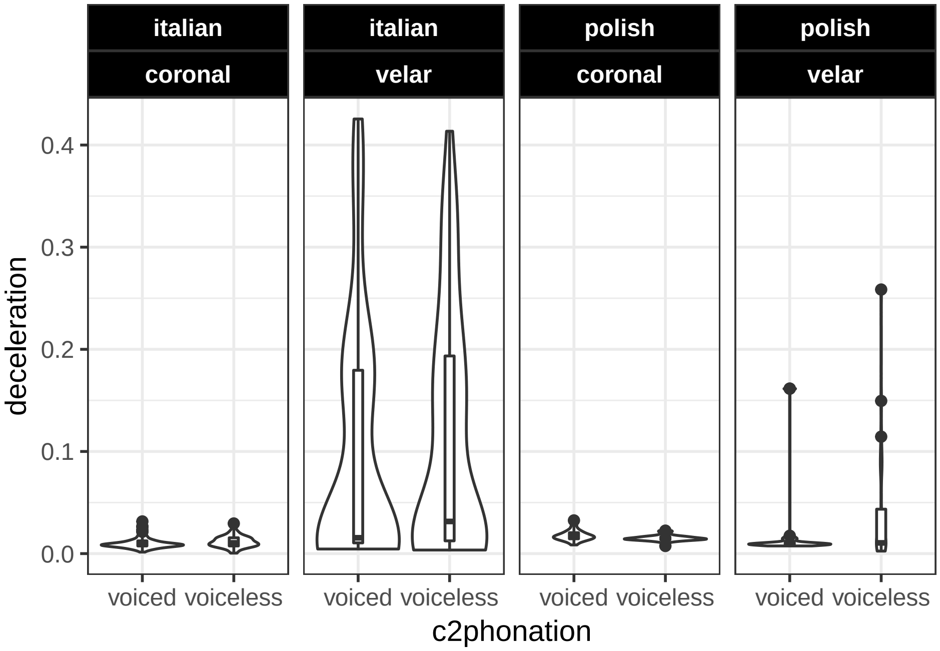
```
## Warning: Removed 88 rows containing non-finite values (stat_boxplot).
```



```
ggplot(cons.gestures, aes(c2phonation, deceleration)) +
    facet_grid(.~language + c2place) +
    geom_violin() +
    geom_boxplot(width=0.1) +
    theme(strip.background = element_rect(fill = "black"),
          strip.text = element_text(color = "white", face = "bold")
          )
```

```
## Warning: Removed 10 rows containing non-finite values (stat_ydensity).
```

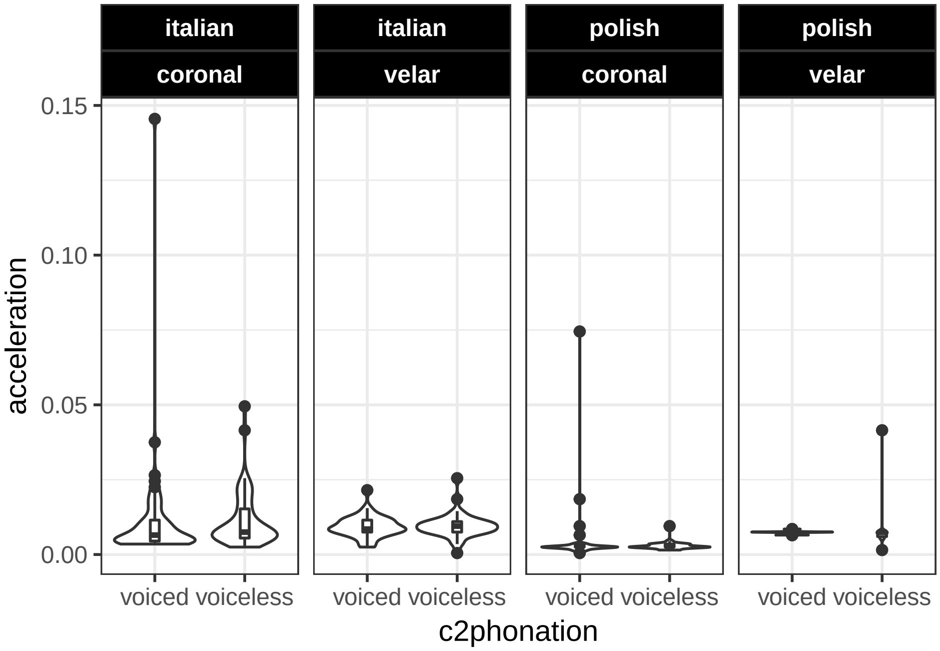
```
## Warning: Removed 10 rows containing non-finite values (stat_boxplot).
```



```
ggplot(cons.gestures, aes(c2phonation, acceleration)) +
    facet_grid(.~language + c2place) +
    geom_violin() +
    geom_boxplot(width=0.1) +
    theme(strip.background = element_rect(fill = "black"),
          strip.text = element_text(color = "white", face = "bold")
          )
```

```
## Warning: Removed 92 rows containing non-finite values (stat_ydensity).
```

```
## Warning: Removed 92 rows containing non-finite values (stat_boxplot).
```

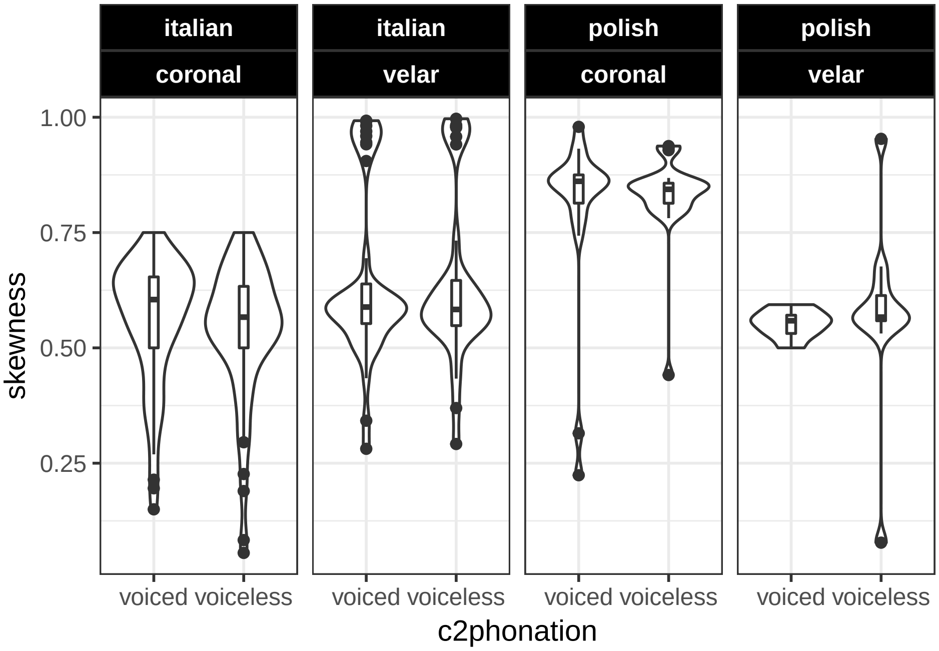


The follwing plots the skweness (`deceleration / nucleus`) of the nucleus gesture.

```
ggplot(cons.gestures, aes(c2phonation, skewness)) +
    facet_grid(.~language + c2place) +
    geom_violin() +
    geom_boxplot(width=0.1) +
    theme(strip.background = element_rect(fill = "black"),
          strip.text = element_text(color = "white", face = "bold")
          )
```

```
## Warning: Removed 94 rows containing non-finite values (stat_ydensity).
```

```
## Warning: Removed 94 rows containing non-finite values (stat_boxplot).
```



I want to check how good the gesture function in AAA is performing. The following gives the number of missing values per speaker.

```
cons.gestures %>%
    select(speaker, GOFF:NONS) %>%
    group_by(speaker) %>%
    summarise_each(funs(sum(is.na(.)))) %>%
    select(speaker, GONS, NONS, max, NOFF, GOFF)
```

```
## # A tibble: 3 × 6
##   speaker  GONS  NONS   max  NOFF  GOFF
##    <fctr> <int> <int> <int> <int> <int>
## 1    PS02    64     1     6    22    52
## 2    SC01   110     0     0    32    70
## 3   SDC02   105     2     2    31    63
```

This chunk instead reports the total number of values per speaker.

```
cons.gestures %>%
    select(speaker, GOFF:NONS) %>%
    group_by(speaker) %>%
    summarise_each(funs(n())) %>%
    select(speaker, GONS, NONS, max, NOFF, GOFF)
```

```
## # A tibble: 3 × 6
##   speaker  GONS  NONS   max  NOFF  GOFF
##    <fctr> <int> <int> <int> <int> <int>
## 1    PS02    96    96    96    96    96
## 2    SC01   139   139   139   139   139
## 3   SDC02   120   120   120   120   120
```

By comparing the two tables, what emerges is that the GONS (Gesture Onset) is almost always missed, followed by GOFF (Gesture Offset). The NOFF (Nucleus Offset) detection, performs slighly better, while NONS and MAX are detected most of the times.

However, judging from the violin plots above, most values might be incorrect.