AAA spline data tidying and plotting

Stefano Coretta

April 16, 2017

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 `num.splines` can be changed; the default is `42`. `columns` is a factor containing the column names which we will use when we import the data. The column names of the splines coordinates are generated by the `paste0` function, according to the format `X_1, Y_1, X_2, Y_2, X_n, ...` (the underscore `_` will be useful later for separating the axis label from the fan number). You can change or add column names according to the structure of your data.

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

We can now import the spline data (`SC01-aaa.txt`) and the stimuli database (`nonce.csv`).

```
raw.data <- read_tsv("./data/sc01-aaa.csv",
    col_names = columns,
    na = "*",
    trim_ws = TRUE
    ) %>%
    mutate_at(vars(matches("^[XY]_")),funs(as.numeric))
```

```
## 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, columns)

raw.data <- unique(raw.data)

stimuli <- read.csv("./data/nonce.csv")
```

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

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

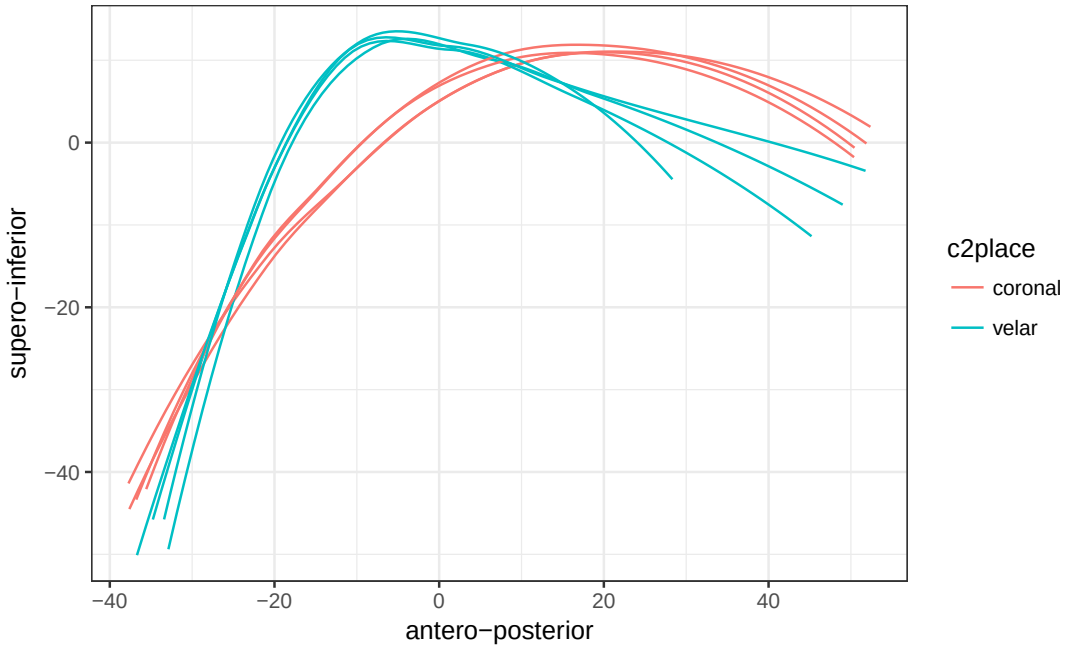
```
## Joining, by = "word"
```

2 Some plotting

We can finally create some plots.

```
filter(splines, vowel == "a") %>%
ggplot(aes(x = X, y = Y, group = rec.date, colour = c2place)) +
    geom_smooth(method = "loess", size = 0.5, alpha = 0.2, se = FALSE) +
    coord_fixed(ratio = 1) +
    xlab("antero-posterior") +
    ylab("supero-inferior")
```

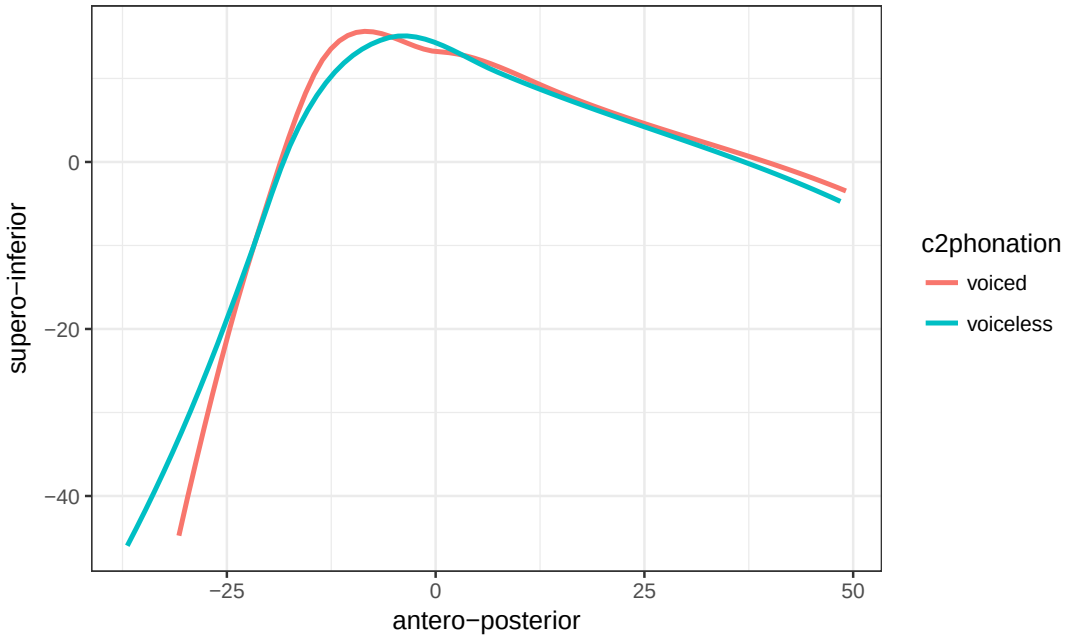
```
## Warning: Removed 267 rows containing non-finite values (stat_smooth).
```



Plot the tongue shape for velar consonants at the moment of maximum constriction.

```
filter(splines, label == "max_TD") %>%
ggplot(aes(x = X, y = Y, colour = c2phonation)) +
    geom_smooth(method = "loess", se = FALSE) +
    coord_fixed(ratio = 1) +
    xlab("antero-posterior") +
    ylab("supero-inferior")
```

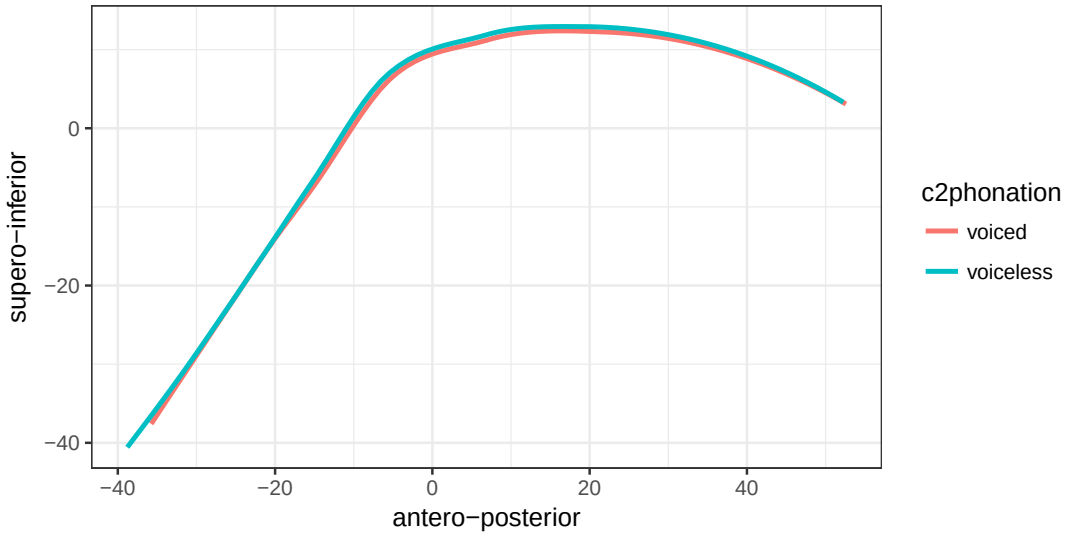
```
## Warning: Removed 132 rows containing non-finite values (stat_smooth).
```



Plot the tongue shape for coronal consonants at the moment of maximum constriction.

```
filter(splines, label == "max_TT") %>%
ggplot(aes(x = X, y = Y, colour = c2phonation)) +
    geom_smooth(method = "loess", se = FALSE) +
    coord_fixed(ratio = 1) +
    xlab("antero-posterior") +
    ylab("supero-inferior")
```

```
## Warning: Removed 89 rows containing non-finite values (stat_smooth).
```



3 Consonantal gestures: target, maximum, release

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

```
cons.gestures <- raw.data %>%
    select(speaker:label) %>%
    separate(label, c("gesture", "tongue.area")) %>%
    spread(gesture, seconds) %>%
    select(speaker:tongue.area, NONS, max, NOFF) %>%
    mutate(nucleus = (NOFF - NONS) * 1000) %>%
    mutate(word = as.factor(word(prompt, 2))) %>%
    left_join(y = stimuli)
```

```
## Joining, by = "word"
```

Let's plot closure duration as a function of place of articulation of C2 (our target consonant in C1VC2V words).

```
ggplot(cons.gestures, aes(c2place, nucleus)) +
    geom_violin() +
    geom_boxplot(width=0.2) +
    xlab("place of C2") +
    ylab("closure duration (msec)")
```

```
## Warning: Removed 7 rows containing non-finite values (stat_ydensity).
```

```
## Warning: Removed 7 rows containing non-finite values (stat_boxplot).
```


20.0
17.5
15.0
12.5
closure duration (msec)
coronal
velar
place of C2