AAA Spline data tidying and plotting

Stefano Coretta

February 2, 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 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).

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

We can now read in the file.

```
raw.data <- SC01_aaa <- read_delim("./data/SC01-aaa.txt",
    "\t", escape_double = FALSE,
    col_names = columns,
    na = "NA",
    trim_ws = TRUE
    )
```

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

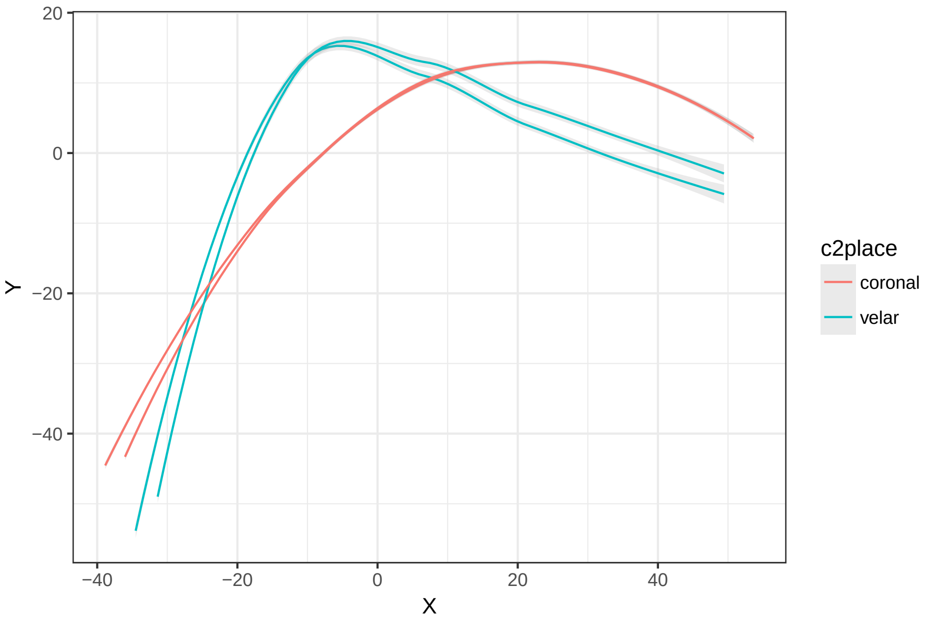
```
splines <- raw.data %>%
    gather(spline, coordinate, matches("[XY]_")) %>%
    separate(spline, c("axis", "fan"), convert = TRUE) %>%
    spread(axis, coordinate) %>%
    mutate(word = as.factor(word(prompt, 2))) %>%
    left_join(y = stimuli, by = c("word" = "orth"))
```

2 Some plotting

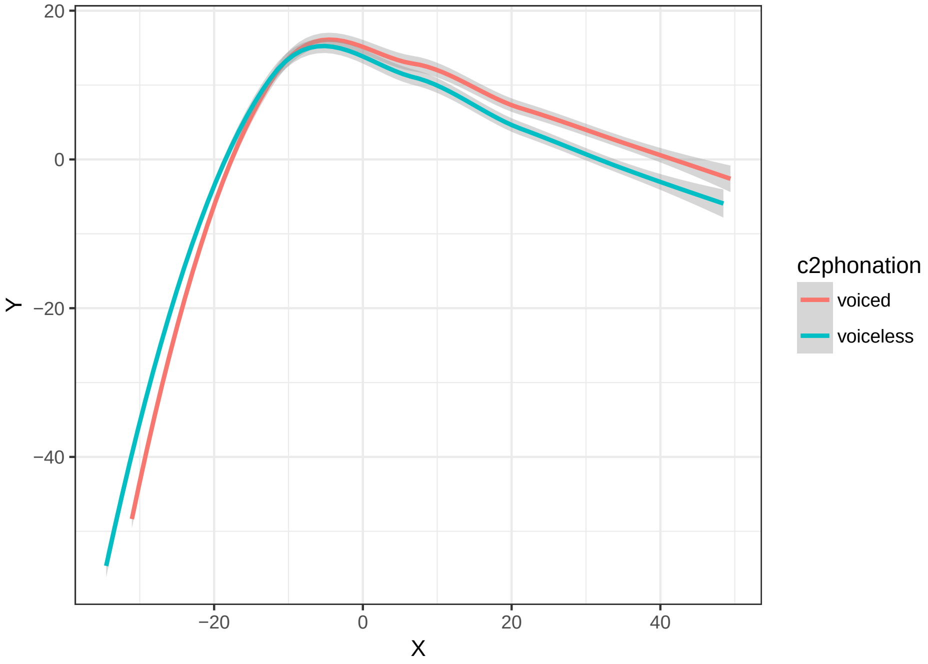
We can finally plot splines grouped by prompt.

```
ggplot(splines, aes(x = X, y = Y, group = word, colour = c2place)) +
    geom_smooth(size = 0.5, alpha = 0.2) +
    coord_fixed(ratio = 1)
```

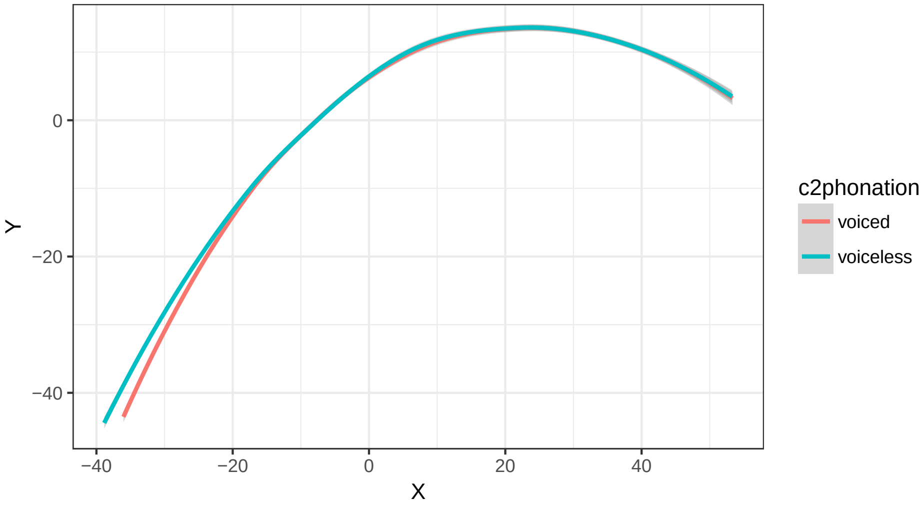
```
## `geom_smooth()` using method = 'loess'
```



```r
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)
```



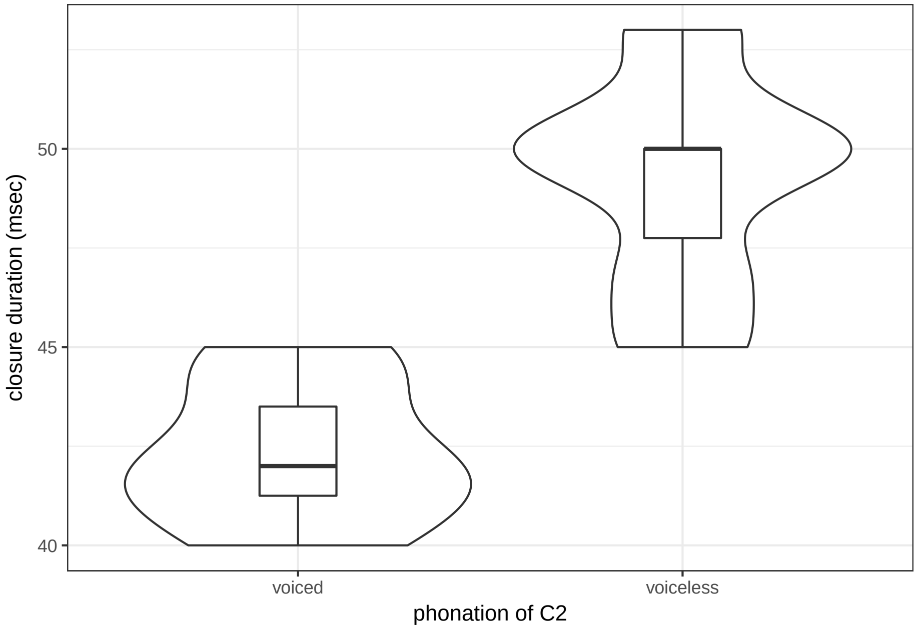
3 Consonantal gestures: target, maximum, release

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 <- raw.data %>%
    select(subject:label) %>%
    separate(label, c("gesture", "tongue.area")) %>%
    spread(gesture, seconds) %>%
    select(subject:tongue.area, target, max, release) %>%
    mutate(closure = (release - target) * 1000) %>%
    mutate(word = as.factor(word(prompt, 2))) %>%
    left_join(y = stimuli, by = c("word" = "orth"))
```

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

```
filter(cons.gestures, c2place == "coronal") %>%
ggplot(aes(c2phonation, closure)) +
    geom_violin() +
    geom_boxplot(width=0.2) +
    xlab("phonation of C2") +
    ylab("closure duration (msec)")
```


closure duration (msec)
voiced
voiceless
phonation of C2