Documentation of data processing of *Vowel duration and consonant voicing: An articulatory study* (Italian and Polish)

Stefano Coretta

2019-04-24

This file contains the documentation of the scripts in the `code` folder of this project. The documentation has been written in literate markdown. To produce the scripts form the source file (`process.praat.md`), use the `lmt` package (written in Go by Dave MacFarlane, at https://github.com/driusan/lmt).

1 Prepare files for force alignment in SPPAS

```
<<<script header>>>

<<<get audio>>>

<<<concatenate recoverably>>>

<<<write sppas>>>
```

The following chunk asks the user for the name of the project directory and the participant ID. Then it reads the audio files from the `audio` directory (whih contains the audio files exported from `AAA`). The directory `alignment` is created as well.

```
form Generate input for force alignment with SPPAS
    word project voicing-effect
    word speaker it01
endform

directory_speaker$ = "../data/ultrasound/derived/'speaker$'"
directory_audio$ = "'directory_speaker$'/recordings"
createDirectory ("'directory_speaker$'/concatenated")
directory_alignment$ = "'directory_speaker$'/concatenated"
writeFile: "'directory_alignment$'/'speaker$'.txt", ""

Create Strings as file list: "filelist", "'directory_audio$'/*.wav"
files = Get number of strings

for file from 1 to files
    select Strings filelist
    file$ = Get string: file
    Read from file: "'directory_audio$'/'file$'"
    sound = selected("Sound")
    sound$ = file$ - ".wav"
endfor
```

The following select all objects except the file list and concatenates the sound objects. SPASS needs a tier named `Orthography`, so we create one. Then, the script loops through the intervals in the TextGrid which correspond to the names of the files, it writes the prompt in the interval and in the text file for IPU detection.

```
select all
minusObject: "Strings filelist"
Concatenate recoverably

selectObject: "TextGrid chain"
Duplicate tier: 1, 1, "Orthography"

intervals = Get number of intervals: 1

for interval from 1 to intervals
    start = Get start point: 1, interval
    end  = Get end point: 1, interval
    filename$ = Get label of interval: 1, interval

    Read Strings from raw text file: "'directory_audio$'/'filename$'.txt"
    prompt$ = Get string: 1
    selectObject: "TextGrid chain"
    Set interval text: 1, interval, "'prompt$'"
    appendFileLine: "'directory_alignment$'/'speaker$'.txt", "'prompt$'"
endfor
```

Finally, we can save the concatenated sound file and the TextGrid with the file names. The latter will be used in the script `search-area.praat` to separate the concatenated TextGrid.

```
selectObject: "Sound chain"
Save as WAV file: "'directory_alignment$'/'speaker$'.wav"

selectObject: "TextGrid chain"
Copy: "filenames"
Remove tier: 1
Save as text file: "'directory_alignment$'/'speaker$'-filenames.TextGrid"
```

2 Extract the search area for spline batch processing and kinematics in `AAA`

```
<<<script header>>>

<<<get alignment>>>

<<<set search>>>

<<<extract search>>>
```

The user is prompt to indicate the project name, the participant ID and the language. Depending on the language selected, the appropriate speech segments are stored for subsequent extraction of the search area. Then the script read the TextGrid file with the force alingment (`ID-palign.TextGrid`). The number of intervals of the TextGrid file is saved in `intervals` and two new tiers are created (`ultrasound` and `kinematics`)

```
form Select folder with TextGrid
    word speaker it01
    comment Supported languages: it, pl
    word language it
endform

if language$ == "it"
    label_lang$ = "k"
    label_2_lang$ = "dico"
```

```
elif language$ == "pl"
    label_lang$ = "j"
    label_2_lang$ = "mowie"
else
    exit "The language you selected is not valid"
endif

directory_audio$ = "../data/ultrasound/derived/'speaker$'/recordings"
directory_alignment$ = "../data/ultrasound/derived/
    ...'speaker$'/concatenated"
directory_palign$ = "../data/ultrasound/raw/corrected-palign"

palign_original = Read from file: "'directory_palign$'/'speaker$'-palign.TextGrid"
palign = Read from file: "'directory_palign$'/'speaker$'-palign.TextGrid"
selectObject: palign

intervals = Get number of intervals: 1
```

Now we can create intervals cointaing the search area for ultrasound and kinematics which will be used in `AAA` for spline batch processing and to find consonantal gestures moments. Then, `[ID]-search.TextGrid` is saved in the `alignment` folder.

```
Insert interval tier: 4, "ultrasound"
Insert interval tier: 5, "kinematics"
Insert interval tier: 6, "vowel"

for interval to intervals
    label$ = Get label of interval: 1, interval
    if label$ == label_lang$
        start_ultrasound = Get start time of interval: 1, interval
        interval_2 = Get interval at time: 2, start_ultrasound
        label_2$ = Get label of interval: 2, interval_2
        if label_2$ == label_2_lang$
            end_ultrasound = Get end time of interval: 1, interval + 7
            Insert boundary: 4, start_ultrasound
            Insert boundary: 4, end_ultrasound
            ultrasound = Get interval at time: 4, start_ultrasound
            Set interval text: 4, ultrasound, "ultrasound"

            start_kinematics = Get start time of interval: 1, interval + 3
            end_kinematics = Get end time of interval: 1, interval + 5
            start_vowel = Get start time of interval: 1, interval + 3
            end_vowel = Get end time of interval: 1, interval + 3

            if label_2$ == "dico"
                start_kinematics = ((end_vowel - start_kinematics) / 2) +
                    ...start_kinematics
                start_vowel_2 = Get start time of interval: 1, interval + 5
                end_kinematics = ((end_kinematics - start_vowel_2) / 2) +
                    ...start_vowel_2
            endif

            Insert boundary: 5, start_kinematics
            Insert boundary: 5, end_kinematics
            kinematics = Get interval at time: 5, start_kinematics
            Set interval text: 5, kinematics, "kinematics"

            vowel$ = Get label of interval: 1, interval + 3
            Insert boundary: 6, start_vowel
            Insert boundary: 6, end_vowel
            vowel_interval = Get interval at time: 6, start_vowel
            Set interval text: 6, vowel_interval, vowel$
        endif
    endif
endfor

Remove tier: 1
Remove tier: 1
Remove tier: 1

Save as text file: "'directory_alignment$'/'speaker$'-search.TextGrid"
```

Then, the script saves each search area to separate TextGrids in the `audio` folder. The file names are extracted from `[ID]-filenames.TextGrid`.

```
filenames = Read from file: "'directory_alignment$'/'speaker$'-filenames.TextGrid"

selectObject: palign
plusObject: filenames

Merge

filenames_tier = 4

intervals = Get number of intervals: filenames_tier

for interval from 1 to intervals
    selectObject: "TextGrid merged"
    start = Get start point: filenames_tier, interval
    end  = Get end point: filenames_tier, interval
    filename$ = Get label of interval: filenames_tier, interval

    Extract part: start, end, "no"

    Remove tier: filenames_tier
    Write to text file: "'directory_audio$'/'filename$'.TextGrid"
    Remove

    selectObject: palign_original
    Extract part: start, end, "no"
    Write to text file: "'directory_audio$'/'filename$'-palign.TextGrid"
    Remove
endfor
```

3 Synchronise EGG data with `AAA` audio data

The following chunk calls the header of the script, which is defined at the end of the documentation, and the main function.

```
<<<script header>>>

<<<sync function>>>
```

The script works by selecting all the files in the Object window after loading files.

```
<<<check objects>>>
```

```
<<<read files>>>

<<<sync>>>
```

Before running, the script checks if the objects list is empty. If not, the script exits and prompts the user to clean the objects list.

```
select all
number_selected = numberOfSelected ()
if number_selected > 0
    exitScript: "Please, remove the objects in the Objects window. For this
    ... script to work, the Objects list must be empty."
endif
```

The form asks for the project name and the participant ID. A boolean is stored as well for enabling the debug mode. In the debug mode, all intermediate files produced by the script are kept in the Objects window. They are deleted otherwise.

```
form Syncronise EGG data
    word project voicing-effect
    word speaker it01
    boolean debug_mode
endform
```

The file lists of the EGG and ultrasound `.wav` files are saved in `filelist_egg` and `filelist_us`. The number of files in the EGG folder is saved in `files`.

```
egg_directory$ = "../data/egg/raw"
us_directory$ = "../data/ultrasound/derived"
out_directory$ = "../data/egg/derived"
createDirectory ("'out_directory$'/'speaker$'")

Create Strings as file list: "filelist_egg", "'egg_directory$'/'speaker$'/*.wav"
files = Get number of strings
Create Strings as file list: "filelist_us", "'us_directory$'/'speaker$'/recordings/*.wav"
```

For every file listed in `filelist_egg`, it reads the file.

```
for file from 1 to files
    select Strings filelist_egg
    file$ = Get string: file
    Read from file: "'egg_directory$'/'speaker$'/'file$'"
endfor
```

Every object is then selected, minus the two file lists. The Sounds are concatenated to `Sound chain`.

```
select all
minusObject: "Strings filelist_egg"
minusObject: "Strings filelist_us"
Concatenate
```

While `Sound chain` is selected, the script inverts the signal (both the audio and the EGG signal are inverted during acquisition with the Laryngograph), and extracts all channels (the object is a stereo sound: channel 1 is the audio, channel 2 is the EGG signal). For cross-correlation to work, the two sound files must have the same sampling frequency. AAA records at a frequency of 22050 Hz. To ensure that the EGG audio is at the same sampling frequency, resampling is performed. `Sound chain_ch1_22050` is created from `Sound chain_ch1`.

```
Multiply: -1
Extract all channels
```

```
selectObject: "Sound chain_ch1"
Resample: 22050, 50
```

The extraction of each simulus from the concatenated sound is achieved through the EGG signal. The function `To TextGrid (silences)` efficiently recognises the voiced streaches of the audio which roughly corresponds to the spoken stimuli. (*Warning*: this assumes that the EGG files don't contain spurious material.) The minimum duration for silence is set to 1 second to avoid voiceless segments being annotated as silence. The EGG signal is first pass filtered. The output is `TextGrid chain_ch2`. The number of intervals in the TextGrid is saved in the variable `intervals`.

```
selectObject: "Sound chain_ch2"
Filter (pass Hann band): 40, 10000, 100

textgrid_silences = To TextGrid (silences): 100, 0, -25, 1, 0.1, "silence", "speech"
intervals = Get number of intervals: 1
Insert interval tier: 2, "new"
```

For each interval in the `TextGrid chain_ch2` wich is labelled `speech`, the start and end time of the interval are moved by -1.5 and 1 second respectively. This ensures that there is enough audio before and after the stimulus for cross-correlation. The original left and right boundaries are removed. The result is that the interval label is changed to `speechsilence`.

```
for interval from 1 to intervals
    label$ = Get label of interval: 1, interval
    if label$ == "speech"
        start = Get starting point: 1, interval
        end = Get end point: 1, interval
        Insert boundary: 2, start - 1.5
        Insert boundary: 2, end + 1
        new_interval = Get interval at time: 2, start
        Set interval text: 2, new_interval, "speech"
    endif
endfor

Remove tier: 1
```

We can now get the number of intervals of the updated TextGrid and set the counter `index` to 1. The counter is used to read the ultrasound audio files, and in the names of the output files.

```
intervals = Get number of intervals: 1

index = 1
```

For every interval in the TextGrid it is checked if the label is `speechsilence`. The intervals with this label correspond to the individual stimuli in the concatenated EGG sound files. If the label is `speechsilence`, the script gets the start and end time of that interval.

```
for interval from 1 to intervals
    label$ = Get label of interval: 1, interval
    if label$ == "speech"
        start = Get starting point: 1, interval
        end = Get end point: 1, interval
```

Then the resampled `Sound chain_ch1_22050` is selected and the portion from `start` to `end` is extracted. This portion corresponds to the TextGrid interval and, thus, to the stimulus. The sound is named `Sound chain_ch1_22050_part`.

```
        selectObject: "Sound chain_ch1_22050"
        Extract part: start, end, "rectangular", 1, "no"
```

The counter `index` is now used to read the audio file from the ultrasound directory. Since the order of the stimuli is the same in both the EGG and unltrasound files, a counter that increases for every interval wth the `speechsilence` label is sufficient. The name of the file is saved after reading and the file remains selected.

```
        selectObject: "Strings filelist_us"
        file_us$ = Get string: index
        Read from file: "'us_directory$'/'speaker$'/recordings/'file_us$'"
        file_us_name$ = selected$ ("Sound")
```

The extracted portion from the EGG audio channel is added to the selection. The cross-correlation between the EGG and ultrasound audio is performed. The time of maximum amplitude in the generated cross-correlated sound corresponds to the off-set between the two files.

```
        plusObject: "Sound chain_ch1_22050_part"

        crosscorrelated = Cross-correlate: "peak 0.99", "zero"
        offset = Get time of maximum: 0, 0, "Sinc70"
```

The concatenated stereo sound (or the recombined stereo if the `invert egg signal` option is active) is selected and a portion is extracted. The portion starting point corresponds to the starting point of the TextGrid interval minus the off-set obtained from the correlation. If the offset is positive (when the audio is longer than the EGG audio), silence is added at the beginning of the EGG sound. If the offset is negative (the EGG sound is longer than the audio), the extra part is deleted from the beginning of the EGG sound to match the beginning of the audio. The end point is the same as the one of the interval. (The endpoint does not matter, since timing is calculated from the beginning of the file.) The sound is finally saved in the `sync` folder.

```
        selectObject: "Sound chain"

        start = start - offset
        Extract part: start, end, "rectangular", 1, "no"
        Save as WAV file: "'out_directory$'/'speaker$'/'file_us_name$'.wav"
```

If the debugging mode is off, all the intermediate files are removed. Otherwise they are kept for inspection. The index is increased by one and the TextGrid is selected for the next cycle of the for loop.

```
        if debug_mode == 0
            removeObject: "Sound chain_ch1_22050_part", "Sound " + file_us_name$,
            ...crosscorrelated, "Sound chain_part"
        endif

        index += 1
        selectObject: textgrid_silences
    endif
endfor
```

4 Extract VUV intervals

This script calculates the voiced and voiceless portions (VUV) in the synchronised EGG files based on the EGG signal.

```
<<<script header>>>

<<<smoothing>>>

<<<get synced egg>>>
```

```
<<<vuv>>>
```

We first read ask for the project name, the speaker ID, the lower and upper frequency for the filter, and the smooth width.

```
form Extract vuv
    word project voicing-effect
    word speaker it01
    comment Specify the lower and upper frequency (in Hz) for filtering:
    real lower 40
    real upper 10000
    comment Specify the smooth width "m" (the number of points):
    real smooth_width 11
    boolean debug_mode
endform

directory$ = "../data/egg/derived"

Create Strings as file list: "filelist", "'directory$'/'speaker$'/*.wav"
files = Get number of strings
```

Now, for each file in `derived/egg`, we can calculate the boundaries of the voiced and voiceless intervals in the file and save them to a TextGrid file.

```
for file from 1 to files
    selectObject: "Strings filelist"
    file$ = Get string: file
    Read from file: "'directory$'/'speaker$'/'file$'"
    filename$ = selected$("Sound")

    <<<to vuv>>>

    <<<save vuv>>>
endfor

removeObject: "Strings filelist"
```

To calculate voiced and voicelss intervals, we can exploit the already available function `To TextGrid (vuv)`. The channel containing the EGG signal (channel 2) is extracted, filtered and smoothed. A PointProcess object is then created from the signal, and finally the `vuv` function is applied.

```
Extract one channel: 2

noprogress Filter (pass Hann band): lower, upper, 100
@smoothing: smooth_width

noprogress To PointProcess (periodic, cc): 75, 600

To TextGrid (vuv): 0.02, 0.001
```

The resulting TextGrid is saved in the same synced EGG files folder.

```
Write to text file: "'directory$'/'speaker$'/'filename$'-vuv.TextGrid"

if debug_mode == 0
    removeObject: "Sound " + filename$, "Sound " + filename$ + "_ch2",
    ..."Sound " + filename$ + "_ch2_band", "PointProcess " + filename$ + "_ch2_band",
    ..."TextGrid " + filename$ + "_ch2_band"
endif
```

5 DEGG tracing

```
<<<script header>>>

<<<smoothing>>>

<<<get files list>>>

<<<file loop>>>
```

First we get the file list and we start looping through the files.

```
form dEGG tracing
    word project voicing-effect
    word speaker it01
    comment Specify the lower and upper frequency (in Hz) for filtering:
    real lower 40
    real upper 10000
    comment Specify the smooth width "m" (the number of points):
    real smooth_width 11
endform

directory$ = "../data/egg/derived/'speaker$'"
directory_textgrid$ = "../data/ultrasound/derived/'speaker$'/recordings"

result_file$ = "../data/datasets/egg/'speaker$'-degg-tracing.csv"
header$ = "speaker,file,date,word,time,rel.time,proportion,maximum,minimum"
writeFileLine: "'result_file$'", "'header$'"

Create Strings as file list: "filelist", "'directory$'/*.wav"
files = Get number of strings
```

For each file, extract both channels. Read from the corrisponding TextGrid in `/data/ultrasound/derived/ID/recordings` and get the starting and end point of the `vowel` interval. Now, we can extract the same interval from channel 2 of the EGG file. Rename the ectracted part as `egg`, and execute the main function, which extracts the dEGG trace.

```
for file to files
    selectObject: "Strings filelist"
    file$ = Get string: file
    filename$ = file$ - ".wav"

    Read Strings from raw text file: "'directory_textgrid$'/'filename$'.txt"
    prompt$ = Get string: 1
    stimulus$ = extractWord$(prompt$, " ")
    date$ = Get string: 2

    Read separate channels from sound file: "'directory$'/'file$'"

    Read from file: "'directory_textgrid$'/'filename$'.TextGrid"
    intervals = Get number of intervals: 3

    if intervals > 1
        start = Get starting point: 3, 2
        end = Get end point: 3, 2

        selectObject: "Sound 'filename$'_ch2"
        ; Extract part: start, end, "rectangular", 1, "yes"
        Rename: "egg"
        <<<main function>>>

        removeObject: "Sound egg", "Sound egg_smooth",
            ..."PointProcess egg_smooth",
            ..."Sound degg_smooth", "PointProcess degg_smooth", "Sound degg"
    endif

endfor
```

```
<<<degg>>>

<<<degg loop>>>
```

The EGG signal file `egg` is selected. Filter EGG signal (`egg_band`) with a pass band filter (40-10KHz, @macerata) and smooth it with moving average (11 point moving average, @macerata, renamed to `egg_smooth`). Create PointProcess (peaks) for EGG (`PointProcess egg_smooth`). Calculate dEGG and remove noise (Praat noise removal based on the first 0.25 seconds, renamed `degg_smooth`). Create PointProcess (peaks) of dEGG (`PointProcess degg_smooth`).

```
Filter (pass Hann band): lower, upper, 100
@smoothing: smooth_width
sampling_period = Get sampling period
time_lag = (smooth_width - 1) / 2 * sampling_period
Shift times by: time_lag
Rename: "egg_smooth"
noprogress To PointProcess (periodic, peaks): 75, 600, "yes", "no"
pp_end = Get end time
Remove points between: 0, start
Remove points between: end, pp_end

selectObject: "Sound egg_smooth"
Copy: "degg"
Formula: "self [col + 1] - self [col]"
; @smoothing: smooth_width
Remove noise: 0, 0.25, 0.025, 80, 10000, 40, "Spectral subtraction"
Rename: "degg_smooth"
noprogress To PointProcess (periodic, peaks): 75, 600, "yes", "no"
Remove points between: 0, start
Remove points between: end, pp_end
```

Loop through the EGG points and get minimum between the first two points. The loop needs to go to the number of points minus 2 since we are selecting three points in each cycle of the loop. This will need to be fixed if we want all cycles to be included. Get dEGG maximum on the rigth of EGG minimum and get minimum of dEGG between current maximum and the next. Normalise max and min to unity. This is gonna be the y axis. The x axis needs to be time aligned: can choose between several (use minimum in EGG as arbitrary epoch, or midway between minima, or what else?) Go to the second and third point and repeat.

ATTENTION! You don't need to normalise to unity. You need to get proportion `(period - value)/period`.

Trying egg_minimum_2 instead of degg_maximum_2 for cases when there is no degg_maximum_2.

```
selectObject: "PointProcess egg_smooth"
egg_points = Get number of points
mean_period = Get mean period: 0, 0, 0.0001, 0.02, 1.3

for point to egg_points - 2
    selectObject: "PointProcess egg_smooth"
    point_1 = Get time from index: point
    point_2 = Get time from index: point + 1
    point_3 = Get time from index: point + 2
    selectObject: "Sound egg_smooth"
    egg_minimum_1 = Get time of minimum: point_1, point_2, "Sinc70"
    egg_minimum_2 = Get time of minimum: point_2, point_3, "Sinc70"
    period = egg_minimum_2 - egg_minimum_1

    if period <= mean_period * 2
        selectObject: "PointProcess degg_smooth"
        pp_degg_points = Get number of points

        if pp_degg_points != 0
          degg_maximum_point_1 = Get nearest index: egg_minimum_1
          degg_maximum = Get time from index: degg_maximum_point_1

          if degg_maximum <= egg_minimum_1
              degg_maximum = Get time from index: degg_maximum_point_1 + 1
          endif

          if degg_maximum != undefined
            selectObject: "Sound degg_smooth"
            degg_minimum = Get time of minimum: degg_maximum, egg_minimum_2, "Sinc70"

            degg_maximum_rel = (degg_maximum - egg_minimum_1) / period
            degg_minimum_rel = (degg_minimum - egg_minimum_1) / period

            time = egg_minimum_1 - start
            proportion = (egg_minimum_1 - start) / (end - start)

            result_line$ = "'speaker$','filename$','date$','stimulus$','egg_minimum_1',
                ...'time','proportion','degg_maximum_rel','degg_minimum_rel'"

            appendFileLine: "'result_file$'", "'result_line$'"
          endif
        endif
    endif
endfor
```

```
procedure smoothing : .width
    .weight = .width / 2 + 0.5

    .formula$ = "( "

    for .w to .weight - 1
        .formula$ = .formula$ + string$(.w) + " * (self [col - " + string$(.w) + "] +
            ...self [col - " + string$(.w) + "]) + "
    endfor

    .formula$ = .formula$ + string$(.weight) + " * (self [col]) ) / " +
        ...string$(.weight ^ 2)

    Formula: .formula$
endproc
```

6 Word DEGG tracing

```
<<<script header>>>

<<<smoothing>>>

<<<get files list word>>>

<<<file loop word>>>
```

```
form dEGG tracing
    word project voicing-effect
    word speaker it01
    comment Specify the lower and upper frequency (in Hz) for filtering:
    real lower 40
    real upper 10000
    comment Specify the smooth width "m" (the number of points):
    real smooth_width 11
endform

directory$ = "../data/egg/derived/'speaker$'"
directory_textgrid$ = "../data/ultrasound/derived/'speaker$'/recordings"

result_file$ = "../data/datasets/egg/'speaker$'-degg-tracing-word.csv"
header$ = "speaker,file,word,time,rel.time,proportion,maximum,minimum"
writeFileLine: "'result_file$'", "'header$'"

Create Strings as file list: "filelist", "'directory$'/*.wav"
files = Get number of strings
```

```
for file to files
    selectObject: "Strings filelist"
    file$ = Get string: file
    filename$ = file$ - ".wav"

    Read Strings from raw text file: "'directory_textgrid$'/'filename$'.txt"
    prompt$ = Get string: 1
    stimulus$ = extractWord$(prompt$, " ")

    Read separate channels from sound file: "'directory$'/'file$'"

    Read from file: "'directory_textgrid$'/'filename$'.TextGrid"

    start = Get starting point: 2, 2
    end = Get end point: 2, 2

    selectObject: "Sound 'filename$'_ch2"
    Extract part: start, end, "rectangular", 1, "yes"
    Rename: "egg"

    <<<main function>>>
endfor
```

7 Get durations

```
<<<script header>>>
```

```
form Get vowel duration
    word project voicing-effect
    word speaker it01
    comment Supported languages: it, pl
    word language it
endform

if language$ == "it"
    label_lang$ = "dico"
elif language$ == "pl"
    label_lang$ = "mowie"
else
    exit "The language you selected is not valid"
endif

directory$ = "../data/ultrasound/derived/'speaker$'/concatenated"
directory_palign$ = "../data/ultrasound/raw/corrected-palign"

result_file$ = "../data/datasets/acoustics/'speaker$'-durations.csv"

header$ = "index,speaker,file,rec_date,ipu_prompt,word,time,sentence_ons,sentence_off,word_ons,word_

writeFileLine: result_file$, header$

bursts = Read from file: "'directory$'/'speaker$'-burst.TextGrid"

release_c1_textgrid = Read from file: "'directory$'/'speaker$'-release-c1.TextGrid"

sentences = Read from file: "'directory$'/'speaker$'.TextGrid"

palign = Read from file: "'directory_palign$'/'speaker$'-palign.TextGrid"
intervals = Get number of intervals: 2

fileNames = Read from file: "'directory$'/'speaker$'-filenames.TextGrid"
index = 0

for interval to intervals
    selectObject: palign
    label$ = Get label of interval: 2, interval
    if label$ == label_lang$
        index += 1
        word$ = Get label of interval: 2, interval + 1
        word_onset = Get start time of interval: 2, interval + 1
        word_offset = Get end time of interval: 2, interval + 1
        # word_duration = (end_target - start_target) * 1000
        c1 = Get interval at time: 1, word_onset
        v1_onset = Get start time of interval: 1, c1 + 1
        c2_onset = Get end time of interval: 1, c1 + 1
        v2_onset = Get end time of interval: 1, c1 + 2
        # v_duration = (end_vowel - start_vowel) * 1000
        # v2_duration = (end_target - end_consonant2) * 1000
        sentence_interval = Get interval at time: 3, word_onset
        sentence$ = Get label of interval: 3, sentence_interval
        sentence_onset = Get start time of interval: 3, sentence_interval

        if sentence$ <> ""
          sentence_offset = Get end time of interval: 3, sentence_interval
          # sentence_duration = end_sentence - start_sentence
          selectObject: bursts
          burst_interval = Get nearest index from time: 1, c2_onset
          release = Get time of point: 1, burst_interval
          if release < c2_onset or release > sentence_offset
              release = undefined
          endif

          # closure = (burst - end_vowel) * 1000
          # rvot = (end_consonant2 - burst) * 1000
          # consonant_duration = closure + rvot

          selectObject: release_c1_textgrid
          release_c1_point = Get nearest index from time: 1, v1_onset
          release_c1 = Get time of point: 1, release_c1_point
          if release_c1 < word_onset or release_c1 > sentence_offset
              release_c1 = undefined
          endif

          # c1_duration = (start_vowel - start_target) * 1000
          # c1_closure = (release_c1 - start_target) * 1000
          # c1_rvot = (start_vowel - release_c1) * 1000
          # c1_rvofft = (end_vowel - release_c1) * 1000

          selectObject: sentences
          prompt = Get interval at time: 2, v1_onset
          prompt$ = Get label of interval: 2, prompt

          selectObject: fileNames
          fileName = Get interval at time: 1, v1_onset
          fileName$ = Get label of interval: 1, fileName
          file_start = Get start time of interval: 1, fileName

          # Get times relative to the start of the individual audio chunk file
          word_onset = word_onset - file_start
          word_offset = word_offset - file_start
          v1_onset = v1_onset - file_start
          c2_onset = c2_onset - file_start
          v2_onset = v2_onset - file_start
          c1_rel = release_c1 - file_start
          c2_rel = release - file_start
          time = sentence_onset
          sentence_onset = sentence_onset - file_start
          sentence_offset = sentence_offset - file_start
        else
          word_onset = undefined
          word_offset = undefined
          v1_onset = undefined
          c2_onset = undefined
          v2_onset = undefined
          c1_rel = undefined
          c2_rel = undefined
          time = sentence_onset
          sentence_onset = undefined
          sentence_offset = undefined

          selectObject: fileNames
          fileName = Get interval at time: 1, time
```

```
          fileName$ = Get label of interval: 1, fileName
        endif

        # rel_rel = (burst - release_c1) * 1000

        Read Strings from raw text file: "../data/ultrasound/derived/'speaker$'/recordings/'fileName
        rec_date$ = Get string: 2

        result_line$ = "'index','speaker$','fileName$','rec_date$','prompt$','word$',
          ...'time',
          ...'sentence_onset','sentence_offset','word_onset','word_offset',
          ...'v1_onset','c2_onset','v2_onset','c1_rel','c2_rel'"

        appendFileLine: "'result_file$'", "'result_line$'"
    endif
endfor

removeObject: palign, bursts
```

8 Burst detection

This script detects the burst in the consonant following the target vowels (C2). The algorythm is based on @avanthapadmanabha2014.

```
<<<script header>>>

<<<get alignment>>>

<<<find consonant>>>
```

We start by identifying the inverval that corresponds to C2.

```
speech_intervals = Get number of intervals: 3
sound = Read from file: "'directory_alignment$'/'speaker$'.wav"
textgrid = To TextGrid: "burst","burst"

for speech_interval to speech_intervals
    selectObject: palign
    speech_label$ = Get label of interval: 3, speech_interval
    if speech_label$ == "speech"
        speech_start = Get start time of interval: 3, speech_interval
        token_interval = Get interval at time: 2, speech_start
        token_end = Get end time of interval: 2, token_interval
        phone_interval = Get interval at time: 1, token_end
        start_consonant = Get start time of interval: 1, phone_interval + 2
        end_consonant = Get end time of interval: 1, phone_interval + 2

        selectObject: sound
        sound_consonant = Extract part: start_consonant, end_consonant,
            ..."rectangular", 1, "yes"

        <<<filter>>>

        <<<plosion index>>>

        selectObject: textgrid
        if burst <> undefined
            Insert point: 1, burst, "burst"
        endif
    endif
endfor

selectObject: textgrid
Save as text file: "'directory_alignment$'/'speaker$'-burst.TextGrid"
```

To calculate the plosion index, it is first necessary to filter the sound file.

```
Filter (pass Hann band): 400, 0, 100
sound_band = selected("Sound")

spectrum = To Spectrum: "no"
Rename: "original"

spectrum_hilbert = Copy: "hilbert"
Formula: "if row=1 then Spectrum_original[2,col] else -Spectrum_original[1,col] fi"
sound_hilbert = To Sound
samples = Get number of samples
Formula: "abs(self)"
matrix = Down to Matrix
period = Get column distance
```

We can now calculate the plosion index.

```
m1_time = 0.006
m2_time = 0.016

for sample from 1 to samples
    current = sample * period
    selectObject: sound_hilbert
    mean_before = Get mean: 1, current - m1_time - m2_time, current - m1_time
    mean_after = Get mean: 1, current + m1_time, current + m1_time + m2_time
    window_average = (mean_before + mean_after) / 2
    current_value = Get value at time: 1, current, "Sinc70"
    plosion = current_value / window_average

    if plosion == undefined
        plosion = 0
    elif plosion < 3
        plosion = 0
    endif

    selectObject: matrix
    Set value: 1, sample, plosion
endfor

To Sound
Shift times by: start_consonant
To PointProcess (extrema): 1, "yes", "no", "Sinc70"
half_consonant = start_consonant + ((end_consonant - start_consonant) / 3) * 2
Remove points between: start_consonant, half_consonant
burst = Get time from index: 1
```

9 Get measurements

This script extracts several durations related to voicing. The main function `merge` is a loop that reads the TextGrids from the derived ultrasound and EGG folders and merges the tier with the gestures from the ultrasound and the tier with the voiced/unvoiced intervals from the EGG.

```
<<<script header>>>

<<<read>>>

<<<merge>>>
```

This is the form that prompts the user to input the directories of the derived ultrasound (`directory_us`) and EGG (`directory_egg`) data, and the ID of the participant (`speaker`). Do not include the participant folder in the path because it will be automatically included in the main function.

```
form Get measurements
    word speaker it01
endform

directory_us_annotations$ = "../data/ultrasound/derived/'speaker$'/
    ...recordings"
directory_egg_vuv$ = "../data/egg/derived/'speaker$'"

directory_out$ = "../data/datasets/acoustics'"

result_file$ = "'directory_out$'/'speaker$'-measurements.csv"
result_header$ = "speaker,word,target,max,release,voff,voffr"
writeFileLine: result_file$, result_header$

Create Strings as file list: "filelist_us", "'directory_us_annotations$'/*.TextGrid"
files_us = Get number of strings

Create Strings as file list: "filelist_egg", "'directory_egg_vuv$'/*.TextGrid"
files_egg = Get number of strings
```

```
for file from 1 to files_us
    selectObject: "Strings filelist_us"
    file$ = Get string: file
    Read from file: "'directory_us_annotations$'/'file$'"
    filename$ = selected$("TextGrid")

    num_tiers = Get number of tiers

    if num_tiers == 4
        Extract one tier: 4

        selectObject: "Strings filelist_egg"
        Read from file: "'directory_egg_vuv$'/'filename$'-vuv.TextGrid"

        selectObject: "TextGrid PointTier_0"
        plusObject: "TextGrid " + filename$ + "-vuv"

        Merge

        Set tier name: 1, "gestures"
        Insert interval tier: 3, "stimulus"

        Read Strings from raw text file: "'directory_us_annotations$'/'filename$'.txt"
        prompt$ = Get string: 1
        stimulus$ = extractWord$(prompt$, " ")

        selectObject: "TextGrid merged"
        Set interval text: 3, 1, stimulus$
        Save as text file: "'directory_out$'/'filename$'-merged.TextGrid"

        <<<calculate>>>
    endif
endfor
```

For the current TextGrid, get the number of points in the `gestures` point tier and, if `number_of_points > 0`, loop through the points. If the point is labelled `target_TT` or `target_TD`, get the time and save it to `target`. Else, write an empty value to `target`, and if the label is `max_TT` or `max_TD`, get the time and write it to `max`. Else, write an empty to `max`, and if the label is `release_TT` or `release_TD`, write the value to `release`. Else, write an empty to `release`.

```
number_of_points = Get number of points: 1

target = undefined
max = undefined
release = undefined
voff = undefined
voffr = undefined

if number_of_points > 0
    for point to number_of_points
        point_label$ = Get label of point: 1, point
        if point_label$ == "target_TT" or point_label$ == "target_TD"
            target = Get time of point: 1, point
            vuv = Get interval at time: 2, target
            vuv_label$ = Get label of interval: 2, vuv
            if vuv_label$ == "U"
                voff = Get starting point: 2, vuv
            else
                voffr = 0
            endif
        elif point_label$ == "max_TT" or point_label$ == "max_TD"
            max = Get time of point: 1, point
            if target == undefined
                vuv = Get interval at time: 2, max
                vuv_label$ = Get label of interval: 2, vuv
                if vuv_label$ == "U"
                    voff = Get starting point: 2, vuv
                else
                    voffr = 0
                endif
            endif
        elif point_label$ == "release_TT" or point_label$ == "release_TD"
            release = Get time of point: 1, point
        endif
    endfor
    if voffr <> 0
        if voff == undefined or release == undefined
            voffr = undefined
        else
            voffr = (release - voff) * 1000
        endif
    endif
endif

result_line$ = "'speaker$','stimulus$','target','max','release',
    ...'voff','voffr'"
```

```
appendFileLine: result_file$, result_line$
```

10 Get the number of a tier based on the name

The following is a procedure that returns the number of a tier in a TextGrid given the name of that tier. The value is returned to `getTierNumber.return`.

```
procedure getTierNumber: .tierName$
    .numberOfTiers = Get number of tiers
    .index = 1
    repeat
        .current$ = Get tier name: .index
        .index += 1
    until .current$ == .tierName$ or .index > .numberOfTiers
    if .index > .numberOfTiers
        exitScript: "The selected TextGrid does not have a tier named ''.tierName$''."
        else
            .return = .index - 1
    endif
endproc
```

11 Get duration of voicing in vowels

```
<<<script header>>>

<<<voicing setup>>>

<<<voicing loop>>>
```

```
form Get duration of voicing
    word speaker it01
endform

vuvDirectory$ = "../data/egg/derived/'speaker$'"
recordings_dir$ = "../data/ultrasound/derived/'speaker$'/recordings"
resultsFile$ = "../data/datasets/egg/'speaker$'-voicing.csv"
resultsHeader$ = "speaker,file,rec_date,word,voicing_start,voicing_end,voicing_duration,voiced_point
writeFileLine: resultsFile$, resultsHeader$

Create Strings as file list: "vuvList", "'vuvDirectory$'/*.TextGrid"
numberOfVuv = Get number of strings
index = 0
```

```
for vuv to numberOfVuv
    selectObject: "Strings vuvList"
    vuvFile$ = Get string: vuv
    vuvTextGrid = Read from file: "'vuvDirectory$'/'vuvFile$'"
    vuvTextGrid$ = selected$("TextGrid")
    palignTextGrid$ = vuvTextGrid$ - "-vuv"

    Read Strings from raw text file: "'recordings_dir$'/'palignTextGrid$'.txt"
    recDate$ = Get string: 2

    palignTextGrid = Read from file: "'recordings_dir$'/'palignTextGrid$'-palign.TextGrid"
    plusObject: vuvTextGrid
    Merge
    numberOfWords = Get number of intervals: 3
    <<<words loop>>>
endfor
```

```
for word to numberOfWords
    word$ = Get label of interval: 3, word
    if word$ == "dico" or word$ == "mowie"
        index = index + 1
        wordStart = Get start time of interval: 3, word + 1
        segment = Get interval at time: 2, wordStart
        vowelStart = Get start time of interval: 2, segment + 1
        vowelEnd = Get end time of interval: 2, segment + 1
        midPoint = vowelStart + (vowelEnd - vowelStart)
        voiced = Get interval at time: 1, midPoint
        voicedStart = Get start time of interval: 1, voiced
        voicedEnd = Get end time of interval: 1, voiced
        voicing = (voicedEnd - voicedStart) * 1000
        stimulus$ = Get label of interval: 3, word + 1

        sentenceInterval = Get interval at time: 4, vowelStart
        sentenceStart = Get start time of interval: 4, sentenceInterval
        sentenceEnd = Get end time of interval: 4, sentenceInterval
        sentenceDuration = sentenceEnd - sentenceStart

        consonant_start = Get start time of interval: 2, segment + 2
        consonant_end = Get end time of interval: 2, segment + 2
        consonant_duration = consonant_end - consonant_start
        one_tenth = consonant_duration / 10

        voiced_points = 0

        for point from 1 to 5
          this_point = consonant_start + (one_tenth * point)
          vuv_interval = Get interval at time: 1, this_point
          voicing$ = Get label of interval: 1, vuv_interval
          if voicing$ == "V"
            voiced_points = voiced_points + 1
          endif
        endfor

        resultLine$ = "'speaker$','palignTextGrid$','recDate$','stimulus$',
            ...'voicedStart','voicedEnd','voicing','voiced_points'"
        appendFileLine: resultsFile$, resultLine$
    endif
endfor
```

Voicing during the consonant is extracted by means of 5 points distributed between the start and 50% of the consonant (including burst and pre-formant voicing). I will probably have to make this more robust by using the closure duration rather than the consonant duration.

12 Wavegram analysis

This script extracts wavegram data from the EGG data.

```
<<<script header>>>

<<<preamble>>>

<<<main loop>>>
```

```
<<<smoothing>>>
```

```
form Wavegram
    word speaker it01
endform

lower = 40
upper = 10000
smoothWidth = 11
results$ = "../data/datasets/egg"
directory_textgrid$ = "../data/ultrasound/derived/'speaker$'/recordings"
createDirectory(results$)
directory$ = "../data/egg/derived/'speaker$'"
resultsHeader$ = "speaker,file,date,word,rel_time,time,sequence,sample,amplitude"
resultsFile$ = "'results$'/'speaker$'-wavegram.csv"
writeFileLine: resultsFile$, resultsHeader$
fileList = Create Strings as file list: "fileList", "'directory$'/*.wav"
numberOfFiles = Get number of strings
```

The preamble defines a few settings for filtering and smoothing, and the results file.

```
#### Files loop ####
for file to numberOfFiles
    selectObject: fileList
    file$ = Get string: file
    filename$ = file$ - ".wav"

    Read Strings from raw text file: "'directory_textgrid$'/'filename$'.txt"
    prompt$ = Get string: 1
    stimulus$ = extractWord$(prompt$, " ")
    date$ = Get string: 2

    Read separate channels from sound file: "'directory$'/'file$'"

    Read from file: "'directory_textgrid$'/'filename$'.TextGrid"
    intervals = Get number of intervals: 3

    if intervals > 1
        start = Get starting point: 3, 2
        end = Get end point: 3, 2

        selectObject: "Sound 'filename$'_ch2"
        ; Extract part: start, end, "rectangular", 1, "yes"
        Rename: "egg"

        <<<vowel loop>>>
    endif

    ; removeObject: "Sound egg", "Sound egg_smooth",
    ;     ..."PointProcess egg_smooth",
    ;     ..."Sound degg_smooth", "PointProcess degg_smooth", "Sound degg"

endfor
```

The main loop goes through each file, extracts the relevat portions using a vuv textgrid, and gets the numeric data.

```
#### Vowel loop ####
<<<degg-wave>>>
```

```
<<<period loop>>>
```

In this loop, each interval corresponding to an uttered vowel is extracted, the DEGG is calculated and the wavegram data is extracted from the DEGG.

```
eggSmooth = Filter (pass Hann band): lower, upper, 100
@smoothing: smoothWidth
sampling_period = Get sampling period
time_lag = (smoothWidth - 1) / 2 * sampling_period
Shift times by: time_lag
Rename: "egg_smooth"
eggPointProcess = noprogress To PointProcess (periodic, peaks): 75, 600, "yes", "no"
pp_end = Get end time
Remove points between: 0, start
Remove points between: end, pp_end

selectObject: eggSmooth
deggSmooth = Copy: "degg_smooth"
Formula: "self [col + 1] - self [col]"
; @smoothing: smoothWidth
Remove noise: 0, 0.25, 0.025, 80, 10000, 40, "Spectral subtraction"
; sampling_period = Get sampling period
; time_lag = (smoothWidth - 1) / 2 * sampling_period
; Shift times by: time_lag
deggPointProcess = noprogress To PointProcess (periodic, peaks): 75, 600, "yes", "no"
Remove points between: 0, start
Remove points between: end, pp_end
```

The raw EGG is filtered and smoothed using a triangular smooth, and from this the DEGG is calculated. Two PointProcess files are also created, which roughly mark each glottal period in the EGG and DEGG.

```
selectObject: eggPointProcess
eggPoints = Get number of points
meanPeriod = Get mean period: 0, 0, 0.0001, 0.02, 1.3

sequence = 0

for point to eggPoints - 2
    selectObject: eggPointProcess
    point1 = Get time from index: point
    point2 = Get time from index: point + 1
    point3 = Get time from index: point + 2
    selectObject: eggSmooth
    eggMinimum1 = Get time of minimum: point1, point2, "Sinc70"
    eggMinimum2 = Get time of minimum: point2, point3, "Sinc70"
    period = eggMinimum2 - eggMinimum1

    <<<wavegram>>>

    sequence = sequence + 1
endfor
```

Each glottal period is detected by finding the EGG minima. The interval between two consecutive EGG minima is a glottal period.

```
if period <= meanPeriod * 2
    selectObject: deggSmooth
    minAmplitude = Get minimum: eggMinimum1, eggMinimum2, "Sinc70"
    maxAmplitude = Get maximum: eggMinimum1, eggMinimum2, "Sinc70"
    sampleStart = Get sample number from time: eggMinimum1
    sampleEnd = Get sample number from time: eggMinimum2
    numberOfSamples = sampleEnd - sampleStart
    sample = sampleStart

    timeNorm = (eggMinimum1 - start) /
        ...(end - start)

    while sample <= sampleEnd
        amplitude = Get value at sample number: 1, sample

        amplitudeNorm = (amplitude - minAmplitude) /
            ...(maxAmplitude - minAmplitude)

        sampleNorm = (sample - sampleStart) /
            ...(sampleEnd - sampleStart)

        # At sample rate 44100 Hz, each period has around 400 samples
        sample = sample + 2

        resultLine$ = "'speaker$','filename$','date$','stimulus$','egg_minimum_1','timeNorm','sequen

        appendFileLine: resultsFile$, resultLine$
    endwhile
endif
```

For each glottal period, the normalised amplitude is calculated for each sample within the period. Normalisation of amplitude and sample time is achieved through unity-based rescaling (range 0-1).

13 Get formants and fundamental frequency

```
<<<script header>>>

form Get formants and fundamental frequency
  word speaker it01
  word sex f
endform

if sex$ == "f"
  max_formant = 5500
else
  max_formant = 5000
endif

result_header$ = "speaker,file,word,time,f1,f2,f3,f0"
result_file$ = "../data/datasets/acoustics/'speaker$'-formants.csv"
writeFileLine: result_file$, result_header$

<<<files loop>>>
```

```
directory_audio$ = "../data/ultrasound/derived/'speaker$'/recordings"
file_list = Create Strings as file list: "file_list", "'directory_audio$'/*.wav"
number_of_files = Get number of strings

for file from 1 to number_of_files
  selectObject: file_list
  file$ = Get string: file
  file_bare$ = file$ - ".wav"
  sound = Read from file: "'directory_audio$'/'file$'"
  palign = Read from file: "'directory_audio$'/'file_bare$'-palign.TextGrid"
  search = Read from file: "'directory_audio$'/'file_bare$'.TextGrid"

  <<<vowel>>>

endfor
```

There are a few cases in which the `[ID].TextGrid` with the search intervals is empty, so we check that there are more than 1 interval. If the TextGrid is not empty, we get the vowel label, calculate the duration and the tenth of the duration, and finally extract formants and fundamental frequency.

```
vowel_intervals = Get number of intervals: 3

if vowel_intervals > 1
  vowel$ = Get label of interval: 3, 2
else
  vowel$ = ""
endif

if vowel$ != ""
  vowel_start = Get start time of interval: 3, 2
  vowel_end = Get end time of interval: 3, 2
  vowel_duration = vowel_end - vowel_start
  duration_tenth = vowel_duration / 10

  selectObject: sound
  sound_vowel = Extract part: vowel_start - 0.5, vowel_end + 0.5, "rectangular", 1, "yes"
  formant = noprogress To Formant (burg): 0, 5, max_formant, 0.025, 50
  selectObject: sound
  pitch = noprogress To Pitch: 0, 75, 600
  selectObject: palign
  word = Get interval at time: 2, vowel_start
  word$ = Get label of interval: 2, word

  for time_point from 1 to 9
    time = vowel_start + (duration_tenth * time_point)
    selectObject: formant
    f1 = Get value at time: 1, time, "Hertz", "Linear"
    f2 = Get value at time: 2, time, "Hertz", "Linear"
    f3 = Get value at time: 3, time, "Hertz", "Linear"

    selectObject: pitch
    f0 = Get value at time: time, "Hertz", "Linear"

    result_line$ = "'speaker$','file_bare$','word$','time_point','f1','f2','f3','f0'"
    appendFileLine: result_file$, result_line$
  endfor

endif
```

14 Create closure annotations

```
<<<script header>>>

<<<speakers loop>>>
```

```
form Create closure annotations
    word project voicing-effect
    word speaker it01
    comment Supported languages: it, pl
    word language it
endform

if language$ == "it"
    label_lang$ = "k"
    label_2_lang$ = "dico"
elif language$ == "pl"
    label_lang$ = "j"
    label_2_lang$ = "mowie"
else
    exit "The language you selected is not valid"
endif

ultrasound_dir$ = "../data/ultrasound/derived"

speaker_rec_dir$ = "'ultrasound_dir$'/'speaker$'/recordings"
file_list = Create Strings as file list: "file_list", "'speaker_rec_dir$'/*.wav"
number_of_files = Get number of strings

<<<textgrids loop>>>
```

The script loops through each directory in `./data/ultrasound/derived/` and reads the search area TextGrids in `./data/ultrasound/derived/[ID]/recordings/`.

```
for wav from 1 to number_of_files
  selectObject: file_list
  wav_file$ = Get string: wav
  textgrid_file$ = wav_file$ - ".wav"
  textgrid = Read from file: "'speaker_rec_dir$'/'textgrid_file$'.TextGrid"

  Insert point tier: 4, "closure"
  number_of_intervals = Get number of intervals: 3

  if number_of_intervals == 3
  vowel$ = Get label of interval: 3, 2

    if vowel$ != ""
      closure = Get end time of interval: 3, 2
      Insert point: 4, closure, "closure_"
    endif
  endif

  Save as text file: "'speaker_rec_dir$'/'textgrid_file$'.TextGrid"
  removeObject: textgrid
endfor
```

15 Get release of C1

This script detects the release in the consonant preceding the target vowel (C1). The algorythm is based on @avanthapadmanabha2014.

```
<<<script header>>>

<<<get alignment>>>
```

```
<<<find consonant c1>>>
```

```
speech_intervals = Get number of intervals: 3
sound = Read from file: "'directory_alignment$'/'speaker$'.wav"
textgrid = To TextGrid: "release_c1", "release_c1"

for speech_interval to speech_intervals
    selectObject: palign
    speech_label$ = Get label of interval: 3, speech_interval
    if speech_label$ == "speech"
        speech_start = Get start time of interval: 3, speech_interval
        token_interval = Get interval at time: 2, speech_start
        token_end = Get end time of interval: 2, token_interval
        # Get interval number of /p/ (C1)
        phone_interval = Get interval at time: 1, token_end
        start_consonant = Get start time of interval: 1, phone_interval
        end_consonant = Get end time of interval: 1, phone_interval

        selectObject: sound
        sound_consonant = Extract part: start_consonant, end_consonant,
            ..."rectangular", 1, "yes"

        <<<filter>>>

        <<<plosion index>>>

        selectObject: textgrid
        if burst <> undefined
            Insert point: 1, burst, "release_c1"
        endif
    endif
endfor

selectObject: textgrid
Save as text file: "'directory_alignment$'/'speaker$'-release-c1.TextGrid"
```

We start by identifying the inverval that corresponds to C1. We do this by finding speech intervals, getting the second word and the first consonant of the second word (the C1 of the target word).

16 Headers

```
########################################
# This is a script from the project 'Vowel duration and consonant voicing: An
# articulatory study', Stefano Coretta
########################################
# MIT License
#
# Copyright (c) 2016 Stefano Coretta
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
######################################
```