`ultrapolaRplot`: A free/open-source R library for plotting tongue traces

Yana Outkin[1], Jonathan Washington[2]

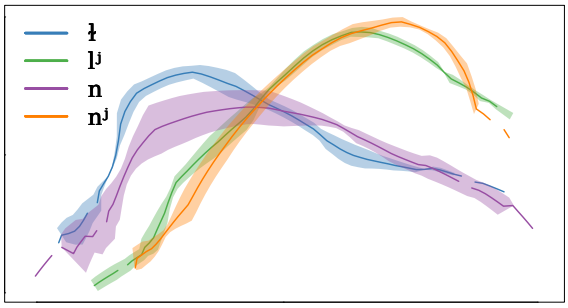
[1,2]Swarthmore College, USA

[1]youtkin1@swarthmore.edu, [2]jonathan.washington@swarthmore.edu

Keywords: tools, software, plotting, tongue traces, R

Introduction

This paper presents `ultrapolaRplot`, an R library for plotting tongue traces available under a free/open-source licence (GPLv3).[1] This library complements UltraTrace [1], a free/open-source annotation tool for ultrasound tongue imaging data. It is designed to be both simple to use and highly configurable. A basic example of `ultrapolaRplot`'s use is demonstrated in Figure 1, showing both the output plot and the code used to create it.



(a) Output plot.

```
library(ultrapolaRplot) # assumed henceforth

rawTraces <- loadTraces('../palatals/',
    'cons')
polarTraces <- makeTracesPolar(rawTraces)
plotTraces(rawTraces, polarTraces,
    pdf.filename="palatals.pdf")
```

(b) Code used.

Figure 1: Simple use case to generate a plot from a dataset of Russian consonants. "cons" is the name of the TextGrid tier that delineates segment names in the recordings.

Basics

The library uses the methodology of Washington [2], Washington & Washington [3], and Washington [4] to plot average tongue traces in polar coordinates from the transducer origin. The library currently plots tongue traces with standard deviation bands—a descriptive approach to presenting ultrasound tongue imaging data. Analytical approaches, such as SSANOVA [5] or GAMs [6], do not descriptively present the range of data in a dataset and instead impose an analysis on the data. The narrow confidence intervals of SSANOVA and GAMs are deceptive for those not fully understanding what they represent, and can lead to an impression that tongue position behaves in a very constrained way. That said, SSANOVA and GAMs could be integrated into `ultrapolaRplot` in the future. A radial coordinate system is used instead of a cartesian coordinate system because it is more precise when working with tongue surface contours [7, 8].

Implementation and use

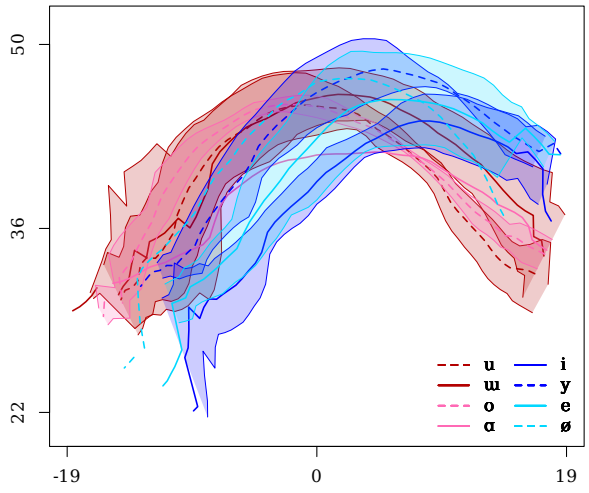
The library is designed around three primary functions.

The `loadTraces()` function loads data from an UltraTrace project by reading the project's metadata file (`metadata.json`), from which trace data is loaded and TextGrid files are identified for category (interval labels in the TextGrid files) and alignment information (UltraTrace keeps track of individual ultrasound frames as a TextGrid point tier).[2] UltraTrace is used for annotation (and alignment) of any set of DICOM and audio files that were recorded simultaneously, so is not limited to a proprietary

format ecosystem. Since UltraTrace can load data exported from AAA [9], this additional format is supported by `ultrapolaRplot` indirectly. Options for loading data include specifying tier names, such as 'orthographic vowel', as well as specifying specific segment categories to find corresponding traces for within that tier. Furthermore, it is possible to load data that was traced on different layers in UltraTrace, such as 'tongue' and 'palate' traces. With certain layers that may not have individual tiers or category specification within TextGrids, the traced coordinates are extracted directly from the metadata file, and the layer name is used as the name of the plotted category. User specification of combinations of multiple categories, tiers, and layers is supported by `loadTraces()`. This functionality allows researchers to easily experiment with plotting different combinations of annotated segments.

The `makeTracesPolar()` function processes and converts extracted traces into polar coordinates. The function supports two different algorithms for estimating the origin of the polar coordinates: (1) the horizontal center of the imaged area, and (2) the mean of the horizontal range of traced data, both matched with coordinate-system vertical zero. The origin may alternatively be overridden with manual specification.

The `plotTraces()` function plots the polar traces. It samples the lines formed by (cartesian) traces every n degrees (default=1°), and stores the radius resulting from the intersections of rays from the transducer origin with each tongue trace along that ray. These sets of radii are then used to calculate the means and standard deviation bands for each plotted category. Numerous plotting options are available, such as whether to show the raw annotations (points), mean lines, and/or standard deviation bands, as well as options for color, transparency, line type, legend position and size, label display, etc. It is possible to save the plot in PDF and PNG formats. An example demonstrating the use of several of these options is provided in Figure 2.



(a) Output plot.

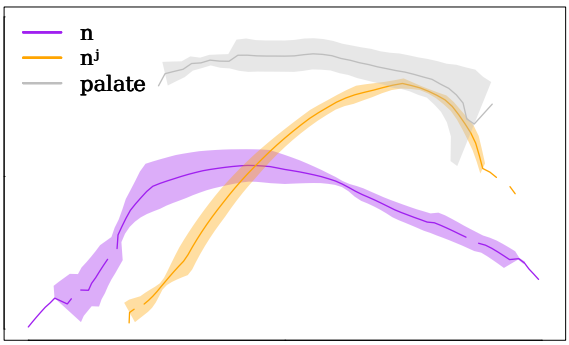
```
rawTraces <- loadTraces('../P03/',
    'vowels', c('u', 'ɯ', 'o', 'ɑ', 'i',
    'y', 'e', 'ø'))
polarTraces <- makeTracesPolar(rawTraces)
plotTraces(rawTraces, polarTraces,
    palette = c("#B30000", "#B30000",
    "#FF69B5", "#FF69B5", "#0000FF",
    "#0000FF", "#00D8FF", "#00D8FF"),
    legend.position = "bottomright",
    points.display = FALSE, bands.lines =
    TRUE, bands.fill = TRUE, transparency =
    0.2, mean.lines = TRUE, means.styles =
    c(2,1,2,1,1,2,1,2), plot.labels = TRUE,
    plot.ticks = TRUE, legend.size = 1,
    legend.linewidth = 1.5, means.linewidth
    = 1.5, tick.size = 1, bands.linewidth =
    0.2, pdf.filename="P03.pdf")
```

(b) Code used.

Figure 2: More complex use case employing advanced options to generate a plot from a dataset of Kyrgyz vowels (P03 from [2]).

Since the interface allows for selection of traces from *layers* (e.g., tongue versus palate traces, specified in UltraTrace metadata), *tiers* (as normally specified in TextGrid files), and *categories* (as specified in TextGrid interval labels), annotations of different types of data can be compared. For example, if one TextGrid tier is used to annotate consonants and one is used to annotate vowels, and palate traces are not annotated in the TextGrid (i.e., correspond to no particular TextGrid interval label), a plot could be created that includes the data for a single consonant category, a single vowel category, and all palate traces. Figure 3 demonstrates the interface for this selection and filtering functionality.

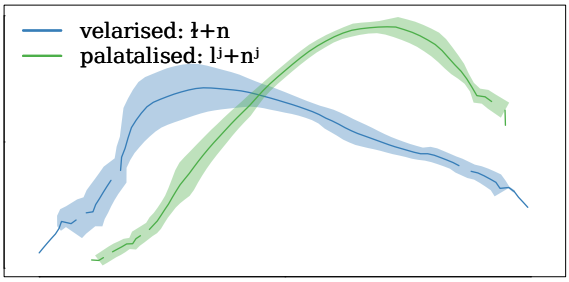
Additionally, categories may be combined and labels may be overridden with any arbitrary label, as demonstrated in Figures 4 and 5.




```
rawTraces <- loadTraces('../palatals/',
    tiernameAll = c("cons", ""),
    categoriesAll = list(c("n", "nʲ"),
    c("")), layersAll = c('tongue',
    'palate'))
polarTraces <- makeTracesPolar(rawTraces,
    scaling.factor = 592/412,
    origin.algorithm = "BottomMean")
plotTraces(rawTraces, polarTraces, palette
    = c("purple", "orange", "gray"),
    pdf.filename = "RusPal_alllevels.pdf")
```

(a) Output plot. **(b)** Code used.

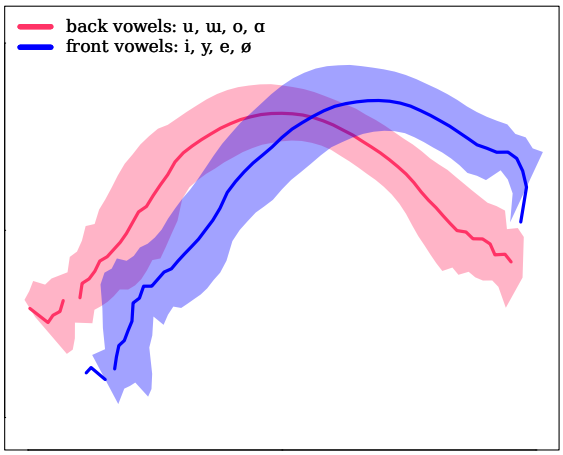
Figure 3: A plot of two categories ('n', 'nʲ') from a tier ('cons') containing several other categories, all on the 'tongue' layer, as well as all traces on the 'palate' layer.




```
rawTraces <- loadTraces('../palatals/',
    categoriesAll = list(c("ɬ", "n"),
    c("lʲ", "nʲ")), mergeCategories = TRUE)
polarTraces <- makeTracesPolar(rawTraces,
    scaling.factor = 592/412,
    origin.algorithm = "BottomMean")
plotTraces(rawTraces, polarTraces,
    pdf.filename = "RusPal_combined.pdf",
    maskCategories = c("velarised: ɬ+n",
    "palatalised: lʲ+nʲ" ))
```

(a) Output plot. **(b)** Code used.

Figure 4: A version of the data in Figure 1 that combines individual categories into two new categories: velarised and palatalised consonants




```
rawTraces <- loadTraces('../P03/',
    categoriesAll = list(c('u', 'ɯ',
    'o', 'ɑ'), c('i', 'y', 'e', 'ø')),
    mergeCategories = TRUE)
polarTraces <- makeTracesPolar(rawTraces,
    origin.algorithm = "BottomMean")
plotTraces(rawTraces, polarTraces,
    pdf.filename = "P03_combined.pdf",
    maskCategories = c("back vowels: u,
    ɯ, o, a", "front vowels: i, y, e, ø"),
    palette = c("#FF3366", "#0000FF"))
```

(a) Output plot. **(b)** Code used.

Figure 5: A version of Figure 2 that combines individual categories into two new categories: back vowels and front vowels.

Future Work

Several features and improvements are planned for future work. Plots of tongue position difference between categories are a top priority, which will add analytical power to this currently descriptive tool. The methodology introduced by Washington [2] and Washington & Washington [3] will be employed, which simply calculates significance of non-overlap of the categories around polar coordinates; the method used in GAMs [6] could be implemented as well.

Currently only UltraTrace's project storage format (and AAA via import into UltraTrace) is supported. While UltraTrace can be used to trace tongue imaging data represented by an arbitrary set of corresponding DICOM and audio files and can be extended to import other formats, using UltraTrace is currently the only way to produce a dataset compatible with `ultrapolaRplot`. We plan to support other annotation formats for ultrasound tongue imaging data, especially those able to be fed to other libraries, such as those widely used for SSANOVA and GAMs.

Additionally, more plotting options are planned, such as specifying the legend font, support for different axis labels (currently axis units are percent of the original ultrasound image, extending from (0,0) as the transducer origin), generation of 'panels' or 'grids' of plots, and potentially support for different plotting algorithms, such as SSANOVA and GAMs. It may also be possible to add other approaches for determining transducer origin, such as extracting it directly from DICOM metadata.

Other planned changes including moving selection and filtering from the `loadTraces()` function to the `plotTraces()` function. Loading and processing all data up front will make it easier and faster to experiment with plotting different categories in different combinations. We also plan to remove the need for a separate `makePolarTraces()` function.

Conclusion

The `ultrapolaRplot` library is a versatile tool for processing and analyzing ultrasound tongue surface imaging data. It is the hope of the authors that this library lowers the bar of entry for those just getting started in ultrasound speech research, and that it will also be useful to experienced researchers.

References

[1] Murphy, K., Stern, N. Z., Swanson, D., Ho, C., & Washington, J. (2020). *UltraTrace: A free/open-source cross-platform tool for manual annotation of ultrasound tongue imaging data*.

[2] Washington, J. (2016). *An investigation of vowel anteriority in three Turkic languages using ultrasound tongue imaging* [doctoral dissertation]. Indiana University. http://hdl.handle.net/2022/20954

[3] Washington, J. N., & Washington, P. A. (2018). A method for distinguishing tongue surface topology for different categories of speech sound. https://doi.org/10.1121/1.5036467

[4] Washington, J. N. (2019). An investigation of the articulatory correlates of vowel anteriority in Kazakh, Kyrgyz, and Turkish using ultrasound tongue imaging. *Proceedings of the 42nd Annual Penn Linguistics Conference*, 219–228. https://doi.org/20.500.14332/45237

[5] Davidson, L. (2006). Comparing tongue shapes from ultrasound imaging using smoothing spline analysis of variance. *The Journal of the Acoustical Society of America*, *120*, 407–415. https://doi.org/10.1121/1.2205133

[6] Coretta, S. (2020). Assessing mid-sagittal tongue contours in polar coordinates using generalised additive (mixed) models. *OSF Preprints*. https://doi.org/10.31219/osf.io/q6vzb

[7] Mielke, J. (2015). An ultrasound study of Canadian French rhotic vowels with polar smoothing spline comparisons. *Journal of the Acoustical Society of America*, 2858–2869. https://doi.org/10.1121/1.4919346

[8] Heyne, M., & Derrick, D. (2015). Benefits of using polar coordinates for working with ultrasound midsagittal tongue contours. *The Journal of the Acoustical Society of America*, *137*, 2302. https://doi.org/10.1121/1.4920405

[9] Wrench, A. (2017). *Articulate Assistant Advanced user guide: Version 2.17.02*. http://materials.articulateinstruments.com/Manuals/

[1] Source code of the `ultrapolaRplot` library is available from https://github.com/SwatPhonLab/ultrapolaRplot/. A release version of the library is also available via CRAN, which may be installed in R by running `install.packages('ultrapolaRplot')`.

[2] Examples of these files are included in the `data` directory of the project GitHub repository.