Praat scripting with Python

In this unit, we will introduce you to automated acoustic analysis, using Praat and some third-party Python libraries which enable you to write Praat scripts in Python.

Why Praat? Praat is free software written and maintained by Paul Boersma and David Weenink (University of Amsterdam) for analyzing speech. Praat is a de-facto standard for linguistic analysis of audio data. It provides reference implementations of a wide variety of signal processing functions, a handy GUI interface for inspecting and annotating your data, and it's free!

Praat also has its own scripting language... Unfortunately, this language is notoriously terrible: poorly documented, unintuitive, full of old-fashioned software design ideas and annoying little "gotchas". In previous versions of the course, we used this set of lectures to teach you a little bit about the Praat scripting language. We focused on how to download and debug scripts written by others, because frankly, writing your own Praat scripts from scratch is just too awful.

Nowadays, though, it's possible to do a lot of your Praat scripting from within the comfort of Python. This approach has some serious advantages (the Python code is shorter, easier to read and much easier to write), but also some disadvantages: the libraries we're going to use are still pretty new, so there won't be as much documentation available and some pieces of Praat functionality are not yet accessible.

If you are not familiar with Praat, the following tutorial will be extremely useful:

> Will Styler. Using Praat for Linguistic Research.

Styler explains both how to use Praat by hand and how to write Praat scripts. You should read the "by hand" sections. You won't need the scripting section but you can feel free to use it as a reference.

We are going to be using the Parselmouth library to call Praat's acoustic analysis routines. See the Parselmouth API for details.

We'll use the TextGridTools library for working with TextGrid annotations.

You probably know that everyone has different vowels. We are going to look at our own vowel spaces by recording and plotting a tiny dataset. To that end we will need to learn about Praat, and use what we have learned in R for visualizing the data.

Recording

First you will want to record yourself. Record the following list of words:

```
hid
head
heed
who'd
hood
hawed
hod
had
hand
ham
```

Make your recording as clear as possible: make sure your environment is quiet. If you have a separate microphone you can use it, but if not, use the built-in microphone on the computer.

Important: we want to have a *.wav file. So make sure that you save your file under that format!

If you don't know how to record on your computer, Audacity is a good option: it can be downloaded for free. It is quite straightforward. Record using the red circle button. Stop the recording using the yellow square. To save the file: File > Export. Or you can do it in Praat. Follow the instructions in Section 4 of Will Styler's Praat guidebook.

So we want to have one *.wav file containing your recording of the 10 words above.

Manual annotation

We will now see how we can annotate the sounds in Praat. Look at Section 9 of Will Styler's Praat guidebook. This explains what you need to know. We want to mark the beginning and end of the word, and the beginning and end of the vowel. So we will create two tiers in the TextGrid: one for the word and one for the vowel. Then annotate your *.wav file.

Scripting

Now to create our vowel space plot, we need to extract the first two formants (F1 and F2) of our 10 vowels. F1 and F2 are related, respectively, to the height of the tongue (high frequency F1 = low vowel, low frequency F1 = high vowel) and to the backness/frontness of the tongue (high frequency F2 = front vowel, low frequency F2 = low vowel). We would like to create a table that contains for each vowel (one vowel per line) the vowel, F1 and F2.

Start out by doing the measurement by hand, for a single vowel. Will Styler walks you through this process in 11.3.4 of his scripting tutorial. Take notes! What functions did you use? What values did you get? We will use these numbers to check our work once we have a script working.

TextGrids

Now, let's start to work on our script. The first thing we'll need is a way to work with the TextGrid we created to represent our annotations. As stated above, we'll use TextGridTools to do this. This ESSV paper (Buschmeier and Wlodarczak, 2013) gives an example of how to use the library, and the documentation lists all the function names and classes. Section 3.1 of the paper gives some example code.

How do we read in a TextGrid file? How do we list the tier names? How do we access a tier by name? By number?

Write a script that takes a TextGrid file name on the command line, reads it in, and prints the tier names. Check that it does what you expect.

How do you access the intervals of a tier? Loop through the intervals. Print the starting and ending timestamps and the label of each one. Verify that these values are correct for the first vowel by opening Praat and checking them by hand!

We'd like to print not only the vowel label, but also the word in which it occurred. Write a function to take an interval from one tier, and find the corresponding interval from another. There are two ways to do this, and you should make sure you understand both!

First, write a function that searches the intervals of the target tier using a *for* loop.

Next, search the documentation. Can you find the function that does this for you?

Acoustic analysis

Now that we can use the TextGrid to find the right intervals, we'll need to compute the formants. Look over the Parselmouth site for an example of how to read in a .wav file, and then use the Parselmouth API to find out how to compute formants.

What does the formant computation method return? What happens when you print the return value? How do you access *f1* and *f2* at a particular time point?

Verify the automatic measurements for the first vowel against your notes of your manual measurements.

Output

Modify the script so that it prints a nice tab-separated value file with column headers. Check that you can read the values you printed into R! An elegant way to make sure your columns and headers match up is to use the *csv* module. A less elegant, but still easy, way to do it is to use *"t".join([...])*, so you can see exactly what you're printing. Do not write separate *print* statements interleaved with your processing code. It's too easy to skip or miscount one!

Automatic annotation

If we have a huge number of words, we probably don't want to do this by hand. We can use forced alignment to automatically create the TextGrid for us. These days, there are a variety of forced aligners you can choose from for local installation. We'll use the web interface to MAUS, which allows you to do everything from the browser.

You will need to give your *.wav file and a transcription, and MAUS will send you back a TextGrid, similar to what we manually did. Compare them. How well did MAUS do?

Do you need to change your script at all to make it work?