eleanorchodroff.com/praat_scripting_tutorial.zip

(Un)laboratory Phonology

Intro to Praat Scripting for Corpus Phonetics and Phonology

Eleanor Chodroff

University of Zurich

13 June 2023

Universität Zürich[UZH]

labphon association for laboratory phonology

Swiss National Science Foundation

Praat

Acoustic analysis program

Best known for its ability to:

- Visualize, label, and segment audio files
- Perform spectral and temporal analyses
- Synthesize and manipulate speech

Praat scripting

Praat: not only a program, but also a language

Why do I want to know Praat the language?

- Save time in the long run
- Minimize human error → consistency
- Allow others to repeat the process identically → replicability
- Easily correct mistakes
- Easily process large amounts of data

But why do I have to do it?

Why can’t I just modify other people’s scripts?

Honestly: power, flexibility, control

You can extract the exact measurements that you need to conduct your study properly

About you

I’m assuming you have basic knowledge of Praat’s functionality

This is my first time relaxing the assumption of coding knowledge

Please speak up if you have questions!

Praat was my very first coding language, so I have been there!

Praat scripting goals

~*~ Script first for yourself, then for others~*~

- Write Praat scripts quickly, effectively, and “from scratch”
- Learn the syntax and structure of the language
- Handle various input/output combinations

Tutorial overview

A. Grammar of Praat
   1. Everything is a mouse click
   2. Variables
   3. For loops
   4. Paths

B. Active Coding
   1. Boilerplate code
   2. Input/output combinations
   3. Basic processing and clean up
   4. Writing to an output file
   5. Looping through intervals and if-else statements
   6. Regular expressions
   7. Moving between tiers in a TextGrid
   8. Moving between objects

Grammar of Praat

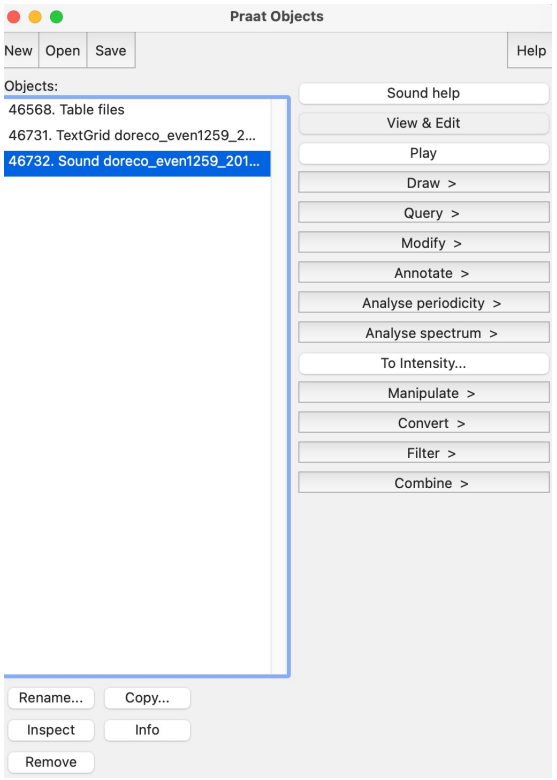
(Almost) everything is a mouse click!

Praat is a GUI scripting language

GUI = Graphical User Interface, i.e., the Objects window

GUI = *gooey* = [guwi]

If you ever get lost while writing a Praat script, click through the steps using the GUI

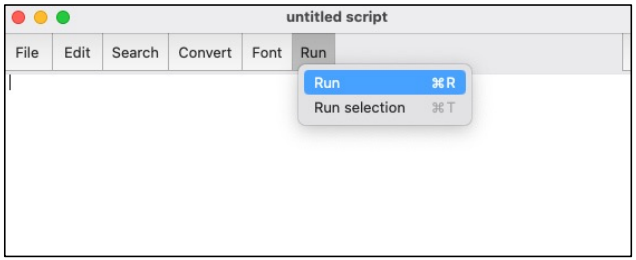


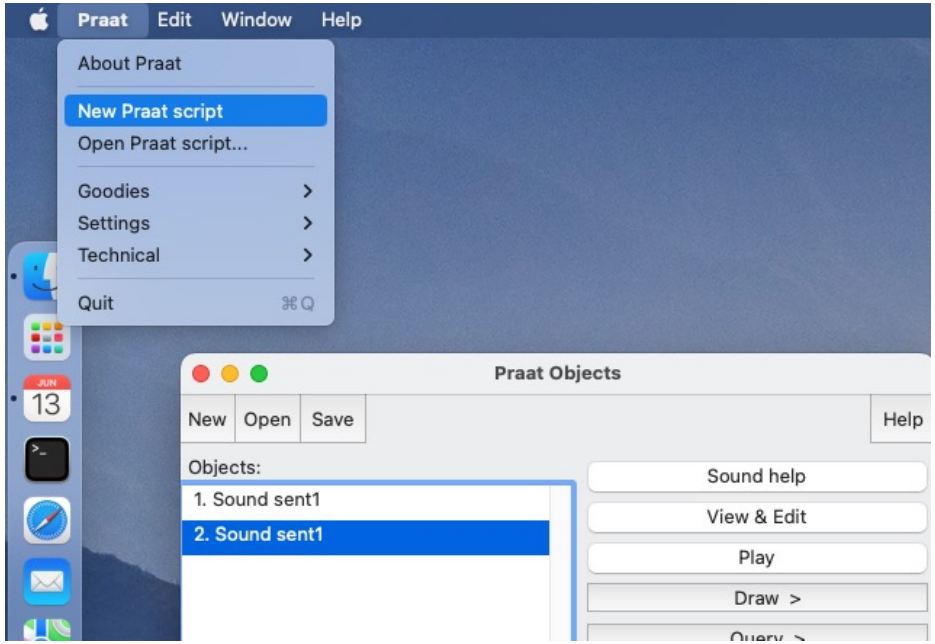
How to open a Praat Script

From the toolbar, select Praat → New Praat script

- Save immediately!
- Save frequently!

To run the script, select Run → Run



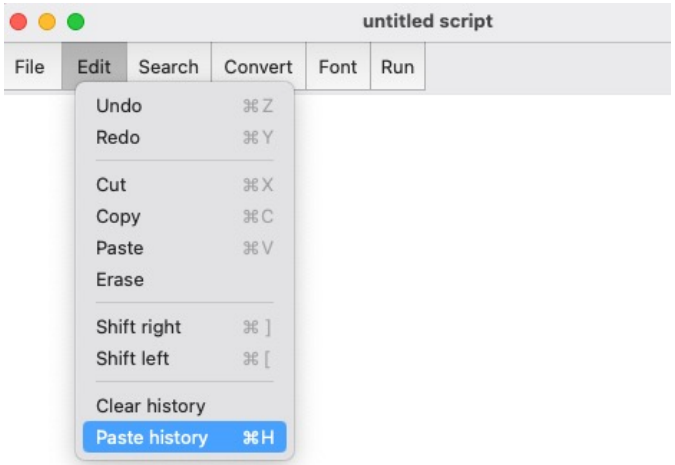


Cheat with Paste history

Practice: Open up a wav file, then resample the wav file to 4000 Hz with precision set to 50 samples

Go back to the Praat script and type Paste history

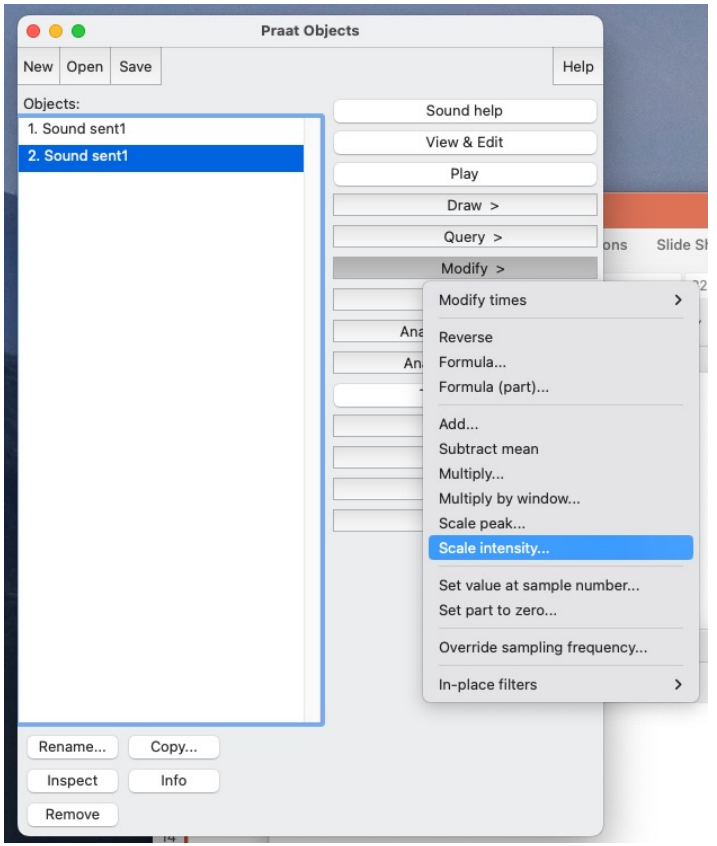
These are the actual commands you would put in your Praat script to automate that process



Commands and arguments

… → :

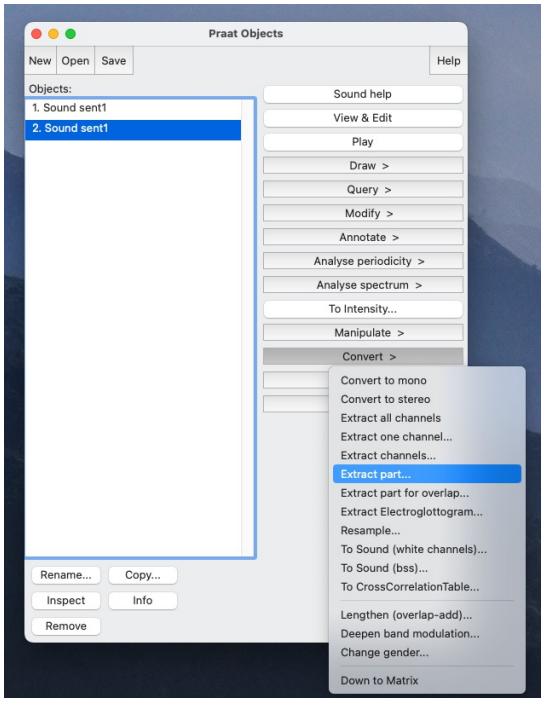
Most of the Praat GUI menu options can be straightforwardly converted into Praat commands by changing … to :

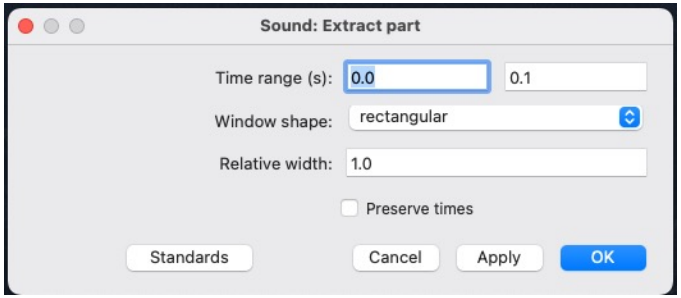


Commands and arguments

The values that Praat requests in the pop-up box are called arguments

```
Extract part: 0, 0.1, “rectangular”, 1.0, “no”
```





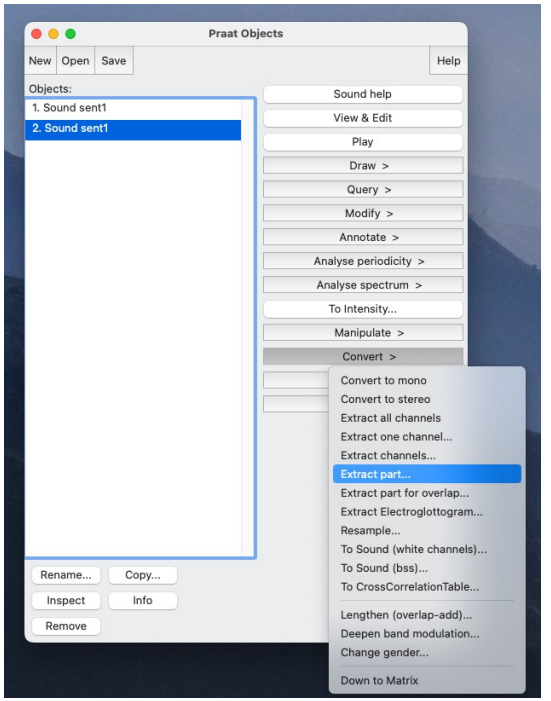
Commands and arguments

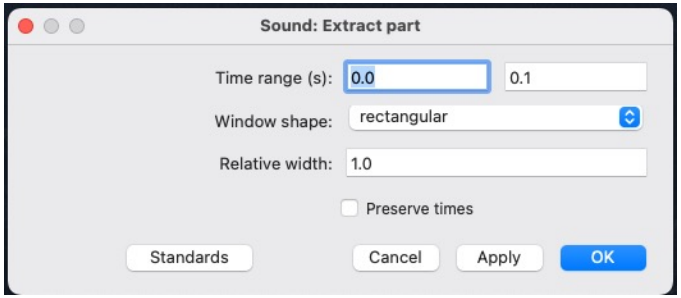
Practice:

Write a script that extracts seconds 2 to 10

Scale the intensity to 65 dB (check under Modify menu)

Make sure the wav file is selected before hitting “Run”



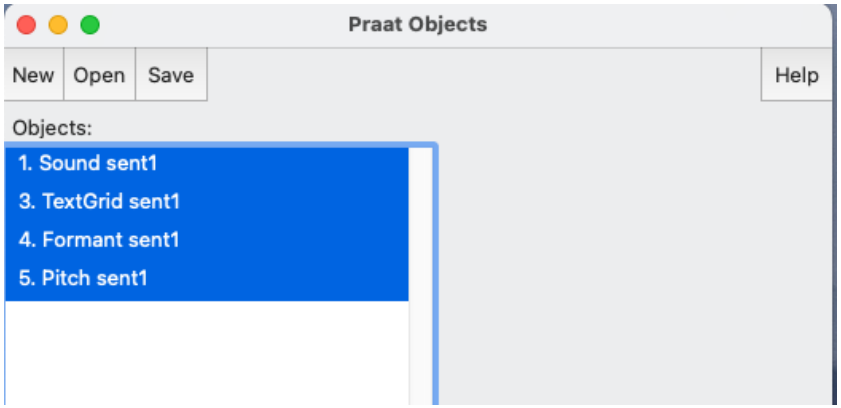


Selecting objects

Again, almost everything is a mouse click

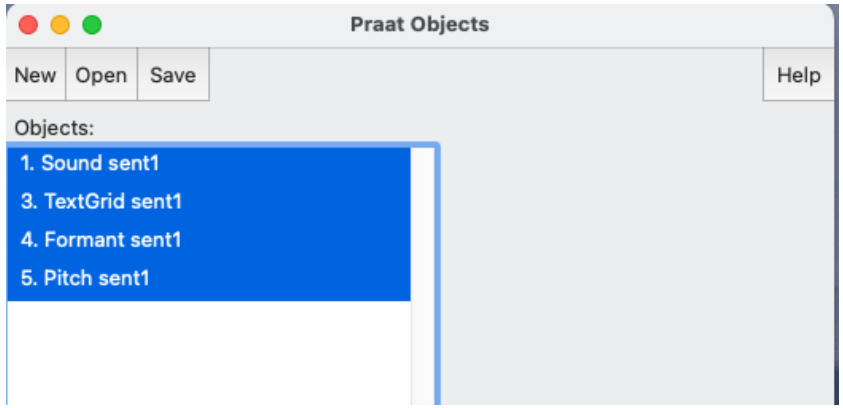
Praat scripts simulate user actions

You'll need to instruct the Praat script to select and remove objects from the Objects window



Selecting objects

You can use the following commands to navigate the selection, deselection, and removal of objects




```
selectObject: "TextGrid vowels"
plusObject: "Sound vowels"
select all
minusObject: "Strings files"
Remove
```

Variables

Variable: a labeled container for storing information that can change

Variables in Praat can store numbers or strings (sequences of characters)

- Numeric variables
- String variables

General constraints on variables in Praat:

- All variables must start with a lower-case letter
- Define variables with equal sign (=)
- Refer to variables as arguments directly

String variables

Name ends with $

Defined by quotes or the output of a command

```
dir$ = "/Users/Eleanor/mydir/data/"
filename$ = Get string: i
label$ = Get label of interval: 2, 100
```

Numeric variables

Name does not have $

Defined by number or numeric output of a command

```
dB = 60.82234
myTotal = 150
nFiles = Get number of strings
intensity = Get intensity (dB)
```

Literals

Literals: values that do not change but can be assigned to a variable (the storage container)

String literals go in quotes: “formants.txt”

Numeric literals are just numbers: 15, 64.5666

Variables

Practice: Assign your favorite animal to a variable

Pause the script and display the value of this variable in the pause statement

To pause the script you can use the following command:

```
pauseScript: “my message”
pauseScript: animal$
pauseScript: phon$, tab$, dur
```

String concatenation and truncation

Concatenate strings with +

Remove part of a string with –

```
basename$ = filename$ - ".wav"
Read from file: dir$ + basename$ + ".wav"
```

Math

You can also do all types of math on numeric variables or numbers

Built-in operators:

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)
- Exponentiation (^)
- Comparison (<, >, <=, >=)
- Among many others

Whitespace

Praat is sort of whitespace sensitive, but not totally

- A space is required after the colon
- (Previous generations): no trailing whitespace permitted at the end of a line
- Otherwise, it doesn't totally matter how much or how little whitespace you have

Convention and your eyes call for standard use of whitespace

- Code-block indentation
- Spaces around equal signs and after punctuation such as colons and commas

Comments

Comments can be added to a full line of the script with #

```
# This is more of a comment than a question
```

Comments cannot be added midline — this won't work

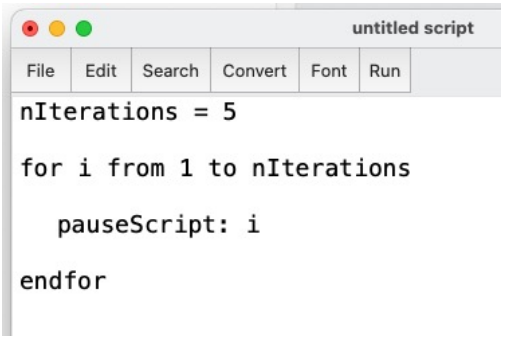
```
intensity = Get intensity (dB) #this gets the intensity of a sound file
```

For loops

For loops iterate from a start value to an end value and complete some set of instructions

The Praat for loop has three parts:

- The header with a loop counter and specified end
- The body of the loop that is implemented once per iteration
- The footer



For loops

Practice:

Write a for loop that displays 5 pause messages that say "the for loop is on iteration number i", where the iteration is specified in i

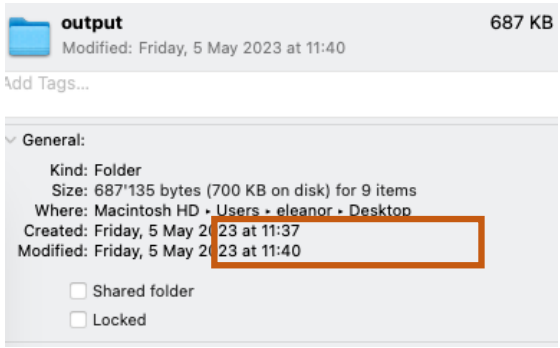
For example, on pause message 3, it should say "the for loop is on iteration number 3"

You will need to **coerce (type coercion)** the numeric variable to a string variable before adding it to the existing string using:

```
string$(numeric_variable)
```

Paths

Path: a computer address, a location of a file or set of files on the computer




```
"/Users/eleanor/Desktop/speech_corpus/"
"C:/Users/eleanor/Desktop/speech_corpus/"
```

Create list of files in a folder:

```
path$ = "/Users/eleanor/Desktop/speech_corpus/"
Create Strings as file list: "files", path$ + "*.TextGrid”
```

→This results in a Strings object in the Objects window

Paths

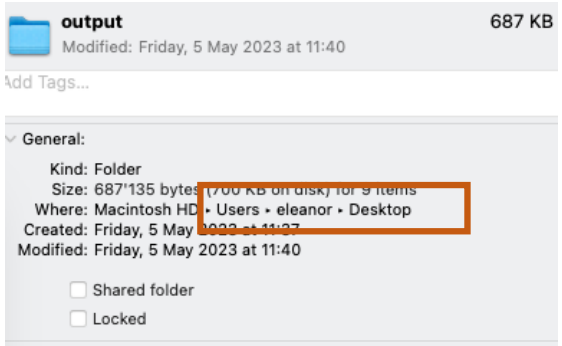
On Macs, you can drag the folder into the script and it will automatically paste its path

path$ = chooseDirectory$: ”Choose a directory”

path$ = path$ + “/”

Add slash at the end with string concatenation

(The slash is very important when reading in files)



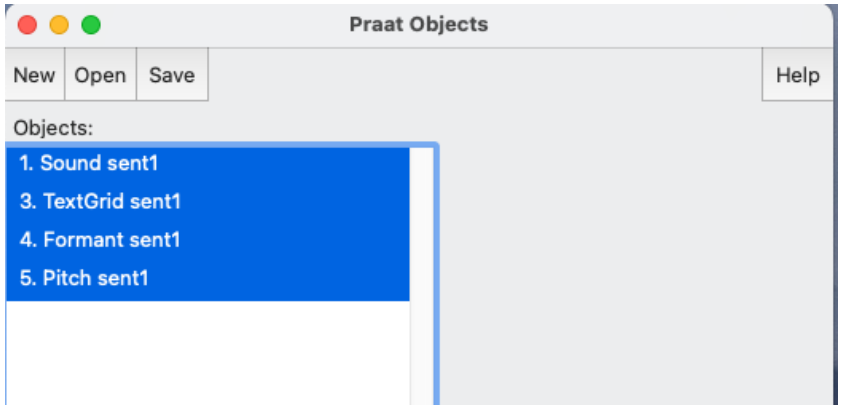
Extra Practice

Grammar of Praat

Selecting objects

Practice: Open up a sound file and TextGrid

Write a script that selects a sound file and a TextGrid and removes them both



String concatenation

Practice: Assign your favorite number to a variable

Add this variable to the string "My favorite number is " and store this in a new variable called number_message$. Don't forget the space before the quote!

You will need to **coerce** the numeric variable to a string variable before adding it to the existing string using:

```
string$(numeric_variable)
```

Pause the script and display the message in the pause message

Paths

Practice:

Locate the path to the downloaded speech corpus

Use the following command to create a list of TextGrids in that path:

Create Strings as file list: “name_of_object”, path$ + “*.TextGrid”

Get value number 3 from the Strings list and store this in a variable. (Click on Strings object and look under the Query command.)

Pause the script and display this value

Active scripting

Boilerplate code

Most Praat scripts can be written with a formulaic structure

It's not necessarily the most concise code, but it is very effective, especially for beginners

Boilerplate

- Header for inputs/outputs
- for-loop to process files

Boilerplate code

```
dir$ = "/Users/Eleanor/mydir/data/"

Create Strings as file list: "files", dir$ + "*.wav"
nFiles = Get number of strings

for i from 1 to nFiles
      selectObject: "Strings files"
      filename$ = Get string: i
      basename$ = filename$ - ".wav"
      Read from file: dir$ + basename$ + ".wav"
      …
endfor
```

Boilerplate code

```
dir$ = "/Users/Eleanor/mydir/data/"

Create Strings as file list: "files", dir$ + "*.wav"
nFiles = Get number of strings

for i from 1 to nFiles
	selectObject: "Strings files"
	filename$ = Get string: i
	basename$ = filename$ - ".wav"
	Read from file: dir$ + basename$ + ".wav"
	…
endfor
```

Place all paths, input and output files, and global variables at the top of the file

Boilerplate code

Set up the for-loop

```
dir$ = "/Users/Eleanor/mydir/data/"

Create Strings as file list: "files", dir$ + "*.wav"
nFiles = Get number of strings

for i from 1 to nFiles
      selectObject: "Strings files"
      filename$ = Get string: i
      basename$ = filename$ - ".wav"
      Read from file: dir$ + basename$ + ".wav"
      …
endfor
```

Boilerplate code

```
dir$ = "/Users/Eleanor/mydir/data/"

Create Strings as file list: "files", dir$ + "*.wav"
nFiles = Get number of strings

for i from 1 to nFiles
	selectObject: "Strings files"
	filename$ = Get string: i
	basename$ = filename$ - ".wav"
	Read from file: dir$ + basename$ + ".wav"
	…
endfor
```

Read in each file

Input/Output combinations

Consider what you want the script to accomplish

Identify what you'll need to read in (input) and what you'll need to write out (output)

Input:
Audio file
TextGrid
Text file

Output:
Audio file
TextGrid
Text file

Write summary and pseudo code

Basic processing + clean-up

Goals: scale intensity of several audio files and save those files to a new directory using our boilerplate code. For each file, we will read it in, scale it, and save it.

Input: audio file, Output: audio file

1) Write out script goals
2) Add our boilerplate code to the Praat script
3) Modify the following:
    1) Separate input and output directories
    2) Add command to scale intensity
    3) Add command to save the script

Basic processing + clean-up

Important concept: clean-up

Implement a clean-up stage at the end of the for-loop body to avoid a build-up of files in the Objects window

Praat will slow down tremendously and even crash if too many files are open in the Objects window at once

```
selectObject: "Sound " + basename$
Remove
```

Basic processing + clean-up

Practice:

Scale the intensity of each audio file in the corpus to 70 dB and save those files to a new directory using our boilerplate code

You will need to create the output folder for the files to be saved in

Don’t forget to clean up!

Writing to an output file

Output: text file

Typical use case: creating a spreadsheet of values, with each column separated by a tab or a comma

Include the location of the text file and its new name (e.g., duration.txt) in the header. Praat will create the file for you

```
outfile$ = “/Users/eleanor/Desktop/durations.txt”

appendFile: outfile$, variable1$, tab$, variable2$, newline$
appendFileLine: outfile$, variable1$, "," , variable2$
appendFileLine: outfile$, "file", tab$, "phon", tab$, "dur"
```

Writing to an output file

Practice:

Let's try measuring the duration of every TextGrid in our folder

We should create a two-column spreadsheet with the first column indicating the name of the TextGrid and the second column indicating the duration

Looping through intervals and if-else statements

Input: TextGrid with an interval tier

Key concept: looping through intervals

2 for loops – one embedded in the other

Vertical loop: Looping through files

Horizontal loop: Looping through intervals

Ultimately want to find intervals that match some condition

To set up the loop across the tier, we'll need to get the number of intervals on the tier

```
nInt = Get number of intervals: tier_number
```

Looping through intervals and if-else statements

```
if i < 20 and word$ = “STIM”
      do this
elsif i = 20 and not word$ = “STIM”
      do that
else
      do this
endif

if label$ == "THE"
  # do stuff
endif
```

Looping through intervals and if-else statements

Practice:

Write a script that saves the file name, start time, end time, and duration of each instance of "THE" to an output file

You can include a fifth column for the word that simply verifies that the word is "THE"

Regular expressions

Regular expression: a way to specify a string pattern -- not necessarily a unique string, but a set of strings

Like using the Find bar in a document: you might start typing the pattern of a word, like "ele" and then the program highlights all instances containing the sequence "ele"

```
if index_regex(label$, "^THE")
# the caret inside the quote means "starts with"
if index_regex(label$, "NG$")
# the dollar sign means "ends with"
if index_regex(label$, "[AEIOU]")
# the label contains one of A, E, I, O, or U
```

See appendix for more!

Regular expressions

Practice: Write a script that measures the start time, end time, and duration of all instances of schwa (indicated by "AH" plus a number in the TextGrid)

Record the file name, as well as a column with the interval label (that should be schwa)

Moving between tiers in a TextGrid

Let’s say we want the label of a corresponding word, but it’s on the tier above our main phone tier

The Query > Query interval tier menu is going to be our friend here

Get label of interval…

- Tier number
- Interval number

We know the tier number, but we don't know the interval number

Moving between tiers in a TextGrid

How do we get the interval number:

Get interval at time…

- Tier number
- Time

Use time of phone label

Get label of interval…

- Tier number
- Interval number

Moving between tiers in a TextGrid

Practice: Modify previous script to get word information for schwa

```
# assume words are on Tier 1 and phones on Tier 2
phone$ = Get label of interval: 2, 50
phone_start = Get start time of interval: 2, 50

# adding 10 ms to the phone_start in case things got a little
misaligned between the word tier and phone tier (and the phone is word-
initial)
word_int = Get interval at time: 1, phone_start + 0.01
word$ = Get label of interval: 1, word_int
```

Moving between objects

But I want formant measurements! (or f0 measurements!)

This involves creating a Formant object or Pitch object from the sound file

At the same time, we frequently use the TextGrid object as our main anchor point to locate intervals to measure

The trick here is being very good at specifying our mouse clicks appropriately → you will need to use a lot of “selectObject:” statements to move back and forth between the TextGrid object and the Formants object

Moving between objects

Practice: Retrieve F1 and F2 in hertz at the midpoint of the schwa vowels using the Formant (Burg) algorithm and all default parameters

Write F1 and F2 to a text file with the file name, word, phone interval name (schwa), start time, end time, duration

Thank you!

Thanks also to:

Association for Laboratory Phonology

SNF PRIMA Grant 208460

Many audiences who have heard previous versions

Appendix

Other loops

Repeat loop

```
repeat
     word$ = Get label of interval: 1, i
     i = i + 1
until word$ = ”STIMULUS”
```

While loop

```
while i < 20
     do this
endwhile
```

Other functions

Procedures

```
for formant from 1 to totalFormants
        @getFormants: formant
endfor


procedure getFormants: formantNum
        selectObject: "Formant " + basename$

        # get formants at each quartile (including start and end)
        for f from 0 to 4
                f_time4 = Get value at time: formantNum, start + f*(dur/4), "hertz", "Linear"
                appendFile: outfile$, fixed$(f_time4, 4), sep$
        endfor
endproc
```

Setting up a procedure requires the name of the procedure, and if necessary, any input variables that it might require following the colon and separated by spaces
@name_of_procedure calls the procedure into action
'formant' is then fed into the 'formantNum' slot in the procedure

Regex appendix

```
if label$ == “THE”
```

Matches “THE” and only “THE” (not “OTHER”, “THEN”, etc.)
== evaluates equality

```
if label$ != “THE”
```

Matches anything that is not an exact match to “THE”

```
if index_regex(label$, “THE”)
```

Matches strings that contain the string “THE” (matches “THEN”, ”OTHER”, etc.)

Regex appendix

```
if index_regex(label$, "^THE")
```

Matches strings that start with "THE"

```
if index_regex(label$, "NG$")
```

Matches strings that end with "NG"

```
if index_regex(label$, "(THE|NORTH)")
```

Matches strings that contain either the string "THE" or "NORTH"

Regex appendix

```
if index_regex(label$, "AH[0-9]*")
```

Matches strings that contain the string "AH" followed by zero or more numbers

```
if index_regex(label$, "AH[0-9]+")
```

Matches strings that contain the string "AH" followed by one or more numbers

Regex appendix

```
if index_regex(label$, "^[PTK][AEIOU][MN][A-Z]+")
```

Matches strings that start with either P, T, or K,
followed by either A, E, I, O, U,
followed by either M or N,
followed by at least one or more letters (+)

```
if index_regex(label$, "^[PTKCBDG][^AEIOUHW]")
```

Matches strings that start with P, T, K, C, B, D, G
and are not followed by A, E, I, O, U, W, or H
This might match English words that begin with a consonant cluster (CLOAK, TRAVELER)

– note there are many equivalent ways to writing these

Regex appendix

```
if index_regex(label$, “AH[0-9]”) or index_regex(label$, “Y$”)
```

Matches strings that contain the string “AH” followed by exactly one number
Or strings that end in “Y”

```
if index_regex(label$, “THE”) & !index_regex(label$, “Y$”)
```

Matches strings that contain the string “THE” but do not end in “Y”
(This would exclude words like “THEY” or “APOTHECARY”)