Praat Scripting Tutorial

Eleanor Chodroff
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?

AUTOMATE ALL THE THINGS
Praat Scripting

Why can’t I just modify others’ scripts?

Honestly: power, flexibility, control

Insert: all the gifs of ‘you can do it’ and ‘you got this’ and thumbs up
Praat Scripting Goals

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

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

1) Praat: Big Picture
2) Getting started
3) Basic syntax
4) Script types + Practice
   • Wav files
   • Measurements
   • TextGrids
   • Other?
Praat: Big Picture

1) Similar to other languages you may (or may not) have used before
   - String and numeric variables
   - For-loops, if else statements, while loops
   - Regular expression matching
   - Interpreted language (not compiled)
2) Almost everything is a mouse click!

i.e., Praat is a GUI scripting language
GUI = Graphical User Interface, i.e., the Objects window

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

Open a Praat script

From the toolbar, select Praat ➔ New Praat script

Save immediately!
Save frequently!
Script Goals and Input/Output

• 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
Some Basics

- Variables
- Selecting objects
- String concatenation
- Comments
- Whitespace
- For-loops + if else statements
- Syntax
- Regex
Variables

```plaintext
dir$ = "/Users/Eleanor/mydir/data/"
filename$ = Get string: i
nFiles = Get number of strings
myTotal = 20
```

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

dir$ = "/Users/Eleanor/mydir/data/"
filename$ = Get string: i
nFiles = Get number of strings
myTotal = 20

• Name ends with $
• Defined by quotes or the output of a command
Numeric Variables

dir$ = "/Users/Eleanor/mydir/data/"
filename$ = Get string: i
nFiles = Get number of strings
myTotal = 20

• Name does not have $
Literals

dir$ = "/Users/Eleanor/mydir/data/"
filename$ = Get string: i
nFiles = Get number of strings
myTotal = 20

• String literals go in quotes
• Numbers are numbers
Selecting objects

```plaintext
selectObject: "Strings files"
plusObject: "TextGrid " + basename$
minusObject: "Sound " + basename$
select all
Remove
```

- Recall that Praat scripts simulate the user actions!
- You’ll need to instruct the script to select and remove objects from the Objects window
String concatenation

\[ \text{basename} = \text{filename} - "\text{.wav}" \]
Read from file: \[ \text{dir} + \text{basename} + "\text{.wav}" \]

- Concatenate strings with +
- Remove part of string with -
Comments

# my comment - path to files
dir$ = "/Users/Eleanor/mydir/data/"

- Hash symbol (#) at the beginning of a line
- Cannot use # symbol midline

This will not work:

dir$ = "/Users/Eleanor/mydir/data/" "#/Users/Mary/mydir/data"
Praat is NOT whitespace sensitive
• At one point, it was sensitive to trailing spaces or tabs at the end of a line
• Doesn’t look like this is the case with the new syntax

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
for-loops

for i from 1 to nFiles
    mycode
endfor

for integerVariable from 1 to integerVariable2
    writeSomeCode
endfor
if i < 20 and word$ = “STIM”
    do this
elsif i = 20 and not word$ = “STIM”
    do that
else
    do this
endif

For more logical operators: http://www.fon.hum.uva.nl/praat/manual/Formulas_2__Operators.html
Basic Syntax

- Almost every command and its argument structure are in the Objects window → just click through it
- ... becomes:

```
Extract part: 0.01, 1.1, "rectangular", 1.0, "no"
```
Basic Syntax

- Arguments are separated by commas
- Any string input must be surrounded by quotes
- Multiple choice input: Must specify one of the given options
- Checkbox input: “yes” or “no”
Basic Syntax

Cheat with this trick!

Within a Praat script, you can use **Edit → Paste History** to reveal everything you’ve just clicked through.
Regex

"files", dir$ + "*.wav"

- Praat uses fairly standard regex (regular expressions) for matching strings
- For complete list, check out http://www.fon.hum.uva.nl/praat/manual/Regular_expressions_1_Special_characters.html
- We’ll go over more of these throughout the tutorial
- See also the appendix for a thorough (but probably not complete) list
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"
...
Boilerplate Code

```plaintext

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.
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
Let’s start Praat scripting

Core types of Praat scripts

  Modifying audio files
  Taking temporal measurements
  Taking spectral measurements
  Creating TextGrids
  Modifying TextGrids
  Miscellaneous
Modifying audio files

Why start here?

Input = audio
Output = audio

In most cases, getting the audio is easy
Modifying audio files

Common procedures:

- Scaling intensity
- Resampling
- Bandpass filtering
- Extract one channel (convert to mono)
Scale intensity script

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
Scale intensity script

dir$ = "/Users/Eleanor/Dropbox/PraatScriptingTutorial/allsstar/"

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"
    pauseScript: "let's take a look"
endfor
for i from 1 to nFiles
  selectObject: "Strings files"
  filename$ = Get string: i
  basename$ = filename$ - ".wav"
  Read from file: dir$ + basename$ + ".wav"
  pauseScript: "let's take a look"
  Scale intensity: 70
endfor
Scale intensity script

for i from 1 to nFiles
    selectObject: "Strings files"
    filename$ = Get string: i
    basename$ = filename$ - "\.wav"
    Read from file: dir$ + basename$ + "\.wav"
    #pauseScript: "let’s take a look"
    Scale intensity: 70
    Save as WAV file: dir$ + basename$ + "_scaled\.wav"
endfor
Scale intensity script

for i from 1 to nFiles
    selectObject: "Strings files"
    filename$ = Get string: i
    basename$ = filename$ - "\.wav"
    Read from file: dir$ + basename$ + "\.wav"
    #pauseScript: "let’s take a look"
    Scale intensity: 70
    Save as WAV file: dir$ + basename$ + "_.scaled\.wav"
    Remove
endfor
Modifying audio files

Do another or move on?

Scaling intensity
Resampling
Bandpass filtering
Extract one channel (convert to mono)
Let’s start Praat scripting

Core types of Praat scripts

Modifying audio files
Taking temporal measurements
Taking spectral measurements
Creating TextGrids
Modifying TextGrids
Miscellaneous
Temporal Measurements

Input: TextGrid
No audio – woo this will be super fast!
*Whenever possible, avoid loading audio files

Output: Text file
Temporal Measurements

Key concepts:
Looping through intervals in a TextGrid
If else statements
Writing a text file
Temporal Measurements

First script: get duration of file

You can get this directly from an audio file’s TextGrid

Script outline:
Read in TextGrid
Get duration
Write filename and duration to text file
Temporral Measurements

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

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

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

dir$ = "'/Users/Eleanor/mydir/data/'
outfile$ = "'/Users/Eleanor/mydir/data/durations.txt"

Create Strings as file list: "files", dir$ + "fave.TextGrid"
nFiles = Get number of strings
Temporal Measurements

for $i$ from 1 to nFiles
    selectObject: "Strings files"
    filename$ = Get string: $i$
    basename$ = filename$ - "\TextGrid"
    Read from file: dir$ + basename$ + "\TextGrid"
    $dur = Get total duration$
    pauseScript: $dur$
endfor
Temporal Measurements

for i from 1 to nFiles
    selectObject: "Strings files"
    filename$ = Get string: i
    basename$ = filename$ - ".TextGrid"
    Read from file: dir$ + basename$ + ".TextGrid"
    dur = Get total duration
    appendFileLine: outfile$, filename$, tab$, dur
    Remove
endfor
Temporal Measurements

Success! You read in a TextGrid and wrote to a Text File using `appendFileLine`:

Next script: get durations of all intervals of every instance of some word (you choose – make it relatively frequent)
Temporal Measurements

Script outline:
Read in TextGrid with word tier (called “Speaker – word”)
  Loop through each word interval
    Stop when interval label matches critical word
    Get start time of interval
    Get end time of interval
      Calculate duration
    Write to filename, duration to text file
Temporal Measurements

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

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

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

dir$ = "~/Users/Eleanor/mydir/data/"
outfile$ = "~/Users/Eleanor/mydir/data/thatDurations.txt"

Create Strings as file list: "files", dir$ + "fave.TextGrid"
nFiles = Get number of strings
for \( i \) from 1 to \( nFiles \)

selectObject: "Strings files"

filename\$ = Get string: \( i \)

basename\$ = filename\$ - ".TextGrid"

Read from file: dir\$ + basename\$ + ".TextGrid"

# get number of intervals on word tier

nInt = Get number of intervals: 2

for \( j \) from 1 to \( nInt \)

... endfor

endfor
Temporal Measurements

for i from 1 to nFiles
    selectObject: "Strings files"
    filename$ = Get string: i
    basename$ = filename$ - ".TextGrid"
    Read from file: dir$ + basename$ + ".TextGrid"
    # get number of intervals on word tier
    nInt = Get number of intervals: 2
    for j from 1 to nInt
        label$ = Get label of interval: 2, j
    endfor
endfor
Temporal Measurements

# get number of intervals on word tier
nInt = Get number of intervals: 2
for j from 1 to nInt
    label$ = Get label of interval: 2, j
    if index_regex(label$, "THAT")
        # get duration
        # write duration to file
    endif
endfor
# get number of intervals on word tier
nInt = Get number of intervals: 2
for j from 1 to nInt
    label$ = Get label of interval: 2, j
    if index_regex(label$, "THAT")
        pauseScript: label$
        start = Get starting point: 2, j
        end = Get end point: 2, j
        dur = end − start
    endif
endfor
# get number of intervals on word tier
nInt = Get number of intervals: 2
for j from 1 to nInt
    label$ = Get label of interval: 2, j
    if index_regex(label$, "THAT")
        pauseScript: label$
        start = Get starting point: 2, j
        end = Get end point: 2, j
        dur = end - start
        appendFileLine: outfile$, filename$, tab$, dur
    endif
endfor
Remove
Temporal Measurements

Looped through text file
Used if else statement
Used regex matching
Intensity Measurements

Input: Audio file and TextGrid
Output: Text file

Processes
Get intensity (dB)
Get root-mean-square
Spectral Measurements

Input: Audio file and TextGrid
Output: Text file

Common measures
  Formants
  f0
  Spectral peak
Creating TextGrids (simple)

Input: audio file
Output: empty or almost empty TextGrid
Create Empty TextGrid

Input: audio file
Output: empty or almost empty TextGrid
Create TextGrid with highly predictable boundaries

Maybe you know the structure of the sound

For instance: each sound is flanked by 20 ms of silence and the critical (middle) interval can be labeled with the filename
Modify TextGrid (simple)

View and Edit the TextGrid by looking for highly predictable words
-or-
Delete boundaries or change text in a very predictable way

Input: Audio file and TextGrid
Output: modified TextGrid
Modify TextGrid (simple)

Working with the TextGrid overview
More boilerplate: loop through intervals
Modify TextGrid (simple)

code
Create/modify TextGrids (v2)

Input: Text file and possibly TextGrid
Output: TextGrid

Example scenario:
You have a text file of start and end times for each condition and want to add a tier with those labels.
Spectral Measurements

To Formant (burg): 0.01, 5, 5500, 0.025, 50
selectObject: “Formant ” + basename$

f1_0 = Get value at time: 1, start, “Hertz”, “Linear”
f2_0 = Get value at time: 2, start, “Hertz”, “Linear”
f3_0 = Get value at time: 3, start, “Hertz”, “Linear”
f1_5 = Get value at time: 1, start + 0.005, “Hertz”, “Linear”
f2_5 = Get value at time: 2, start + 0.005, “Hertz”, “Linear”
f3_5 = Get value at time: 3, start + 0.005, “Hertz”, “Linear”
f1_10 = Get value at time: 1, start + 0.01, “Hertz”, “Linear”
f2_10 = Get value at time: 2, start + 0.01, “Hertz”, “Linear”
f3_10 = Get value at time: 3, start + 0.01, “Hertz”, “Linear”
Extract Sounds

Input: Audio file and TextGrid
Output: Audio file (and TextGrid)
Create Sounds

Input: nothing! Or existing audio file
Output: Audio file
Other loops

Repeat loop

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

While loop

```pseudocode
while i < 20
  do this
endwhile
```
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$, “THE”)
   Matches strings that contain the string “THE” (matches “THEN”, ”OTHER”, etc.)
if index_regex(label$, "^THE")
    Matches strings that \textit{start with} “THE”
if index_regex(label$, "NG$")
    Matches strings that \textit{end with} “NG”
if index_regex(label$, "(THE|NORTH)")
    Matches strings that contain \textit{either} the string “THE” \textit{or} “NORTH”
if index_regex(label$, "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
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
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”)