

# Atelier EMA

-

## Part II: ema2wav & Praat

Philipp Buech

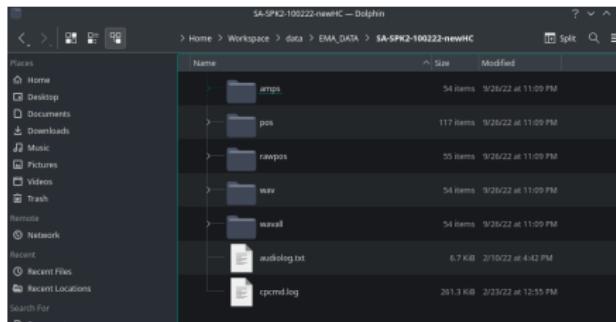
Laboratoire de Phonétique et Phonology - UMR 7018, CNRS/Sorbonne Nouvelle

30/09/2022

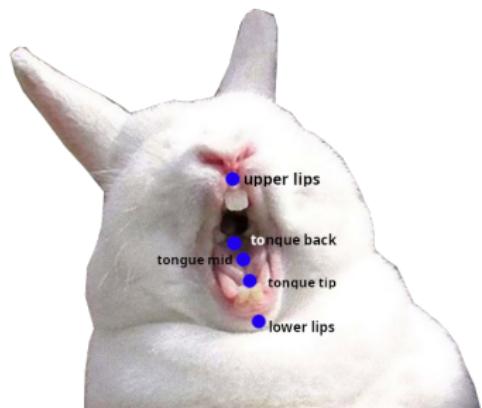
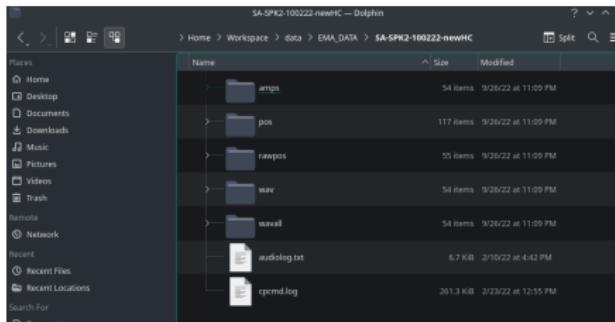


# 0 - Prerequisites

# Data requirements



# Data requirements



# Software requirements

PRAAT



# Software requirements

PRAAT



PYTHON



# Software requirements

PRAAT



PYTHON



EMA2WAV



# Python

- ▶ universal high-order programming language
- ▶ good readability
- ▶ free
- ▶ runs on:



## Python - Anaconda (recommendation)



- ▶ **Anaconda distribution** for Windows, MacOS and Linux
- ▶ Python-centered, but supports also R
- ▶ easy handling of environments
- ▶ powerful package manager
- ▶ latest stable packages

## ema2wav

- ▶ converts EMA data into multichannel WAV-files (& CSVs)
- ▶ Python-based, only free & open-source dependencies
- ▶ supports AG500/501 (Carstens Medizinelektronik GmbH)
- ▶ multiple options for derivations & calculations
- ▶ GUI, command line or python module
- ▶ work-in-progress
- ▶ .dmg (MacOS) & script (← always latest version)
- ▶ can be obtained [here](#)

# 1 - ema2wav & Praat

# Program start

ema2wav can be started either

- ▶ using the **GUI**
  - ▶ via binary (.dmg)
  - ▶ via console
- ▶ using command-line (see documentation)
- ▶ by importing as a python module (custom script/notebook/**Google colab notebook**; see documentation)

## Installation (Anaconda - script version)

- ▶ open Terminal/Console/Anaconda Prompt

- ▶ create conda environment:

```
conda create -name ema_env
```

- ▶ activate the environment:

```
conda activate ema_env
```

- ▶ install the dependencies:

```
pip install -r
```

path/to/ema2wav/folder/src/requirements.txt

## Run ema2wav

- ▶ for Mac: run ema2wav\_app.app (see GitHub documentation)
- ▶ script version (GUI, from terminal, ema2wav src directory assumed):  
`python ema2wav_app.py`

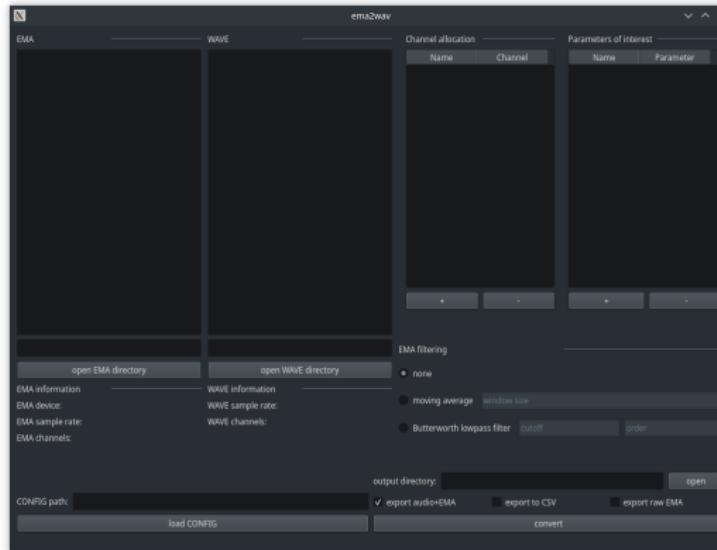
# User input

```
{  
    "ema_device_info": "AG50x",  
    "export_audio+ema": true,  
    "export_to_csv": false,  
    "export_raw_ema": false,  
    "output_directory": "/path/to/your/output/folder/",  
    "ema_input_directory": "/path/to/your/ema/folder/",  
    "ema_samplerate": 1250,  
    "ema_channels": 16,  
    "audio_input_directory": "/path/to/your/wav/folder/",  
    "audio_samplerate": 48000,  
    "audio_channels": 1,  
    "channel_allocation": {  
        "TD": 3,  
        "TT": 5  
    },  
    "parameters_of_interest": {  
        "0_TT": "y",  
        "1_TD": "y-vel",  
        "2_TT": "tvel",  
        "3_TD": "x"  
    },  
    "filter": null  
}
```

# User input - parts of the input file

{		
"ema_device_info": "AG50x",		Device info (necessary)
"export_audio+ema": true,		
"export_to_csv": false,		output options (necessary)
"export_raw_ema": false,		
"output_directory": "/path/to/your/output/folder/",		folder paths (necessary)
"ema_input_directory": "/path/to/your/ema/folder/",		
"audio_input_directory": "/path/to/your/wav/folder/",		
"ema_samplerate": 1250,		general information of the ema and audio files (optional)
"ema_channels": 16,		
"audio_samplerate": 48000,		
"audio_channels": 1,		
"channel_allocation": {		channel information (necessary)
"TD": 3,		
"TT": 5		
},		
"parameters_of_interest": {		parameters of interest, consist of X_ + channel name & the parameter to extract (necessary)
"0_TT": "y",		
"1_TD": "y-vel",		
"2_TT": "tvel",		
"3_TD": "x"		
},		
"filter": null		filter information (necessary)
}		

# Start of the conversion process (GUI)



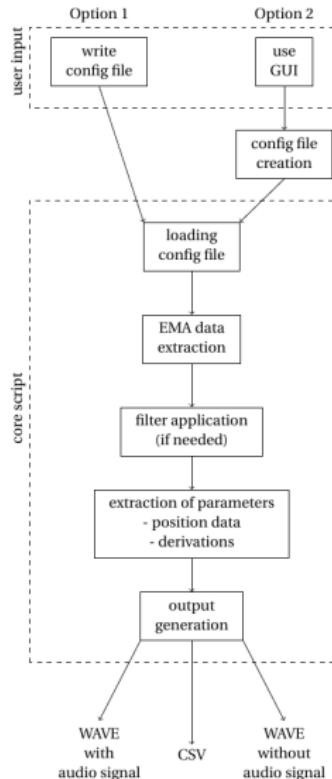
## Start of the conversion process (manual)

- ▶ from terminal (ema2wav src directory assumed)  
`python convert.py config.json`

- ▶ as Python module:

```
import ema2wav_core as ec  
config_file =  
"/path/to/your/config_file.json"  
ec.ema2wav_conversion(config_file)
```

# Conversion process



# Conversion process - reshaping of the data

sample 1								sample 2								sample n							
channel 1				channel 2				channel n				channel 1				channel n				...			
x	y	z	...	x	y	z	...	...	x	y	z	...	...	x	y	z	...	...	x	y	z	...	

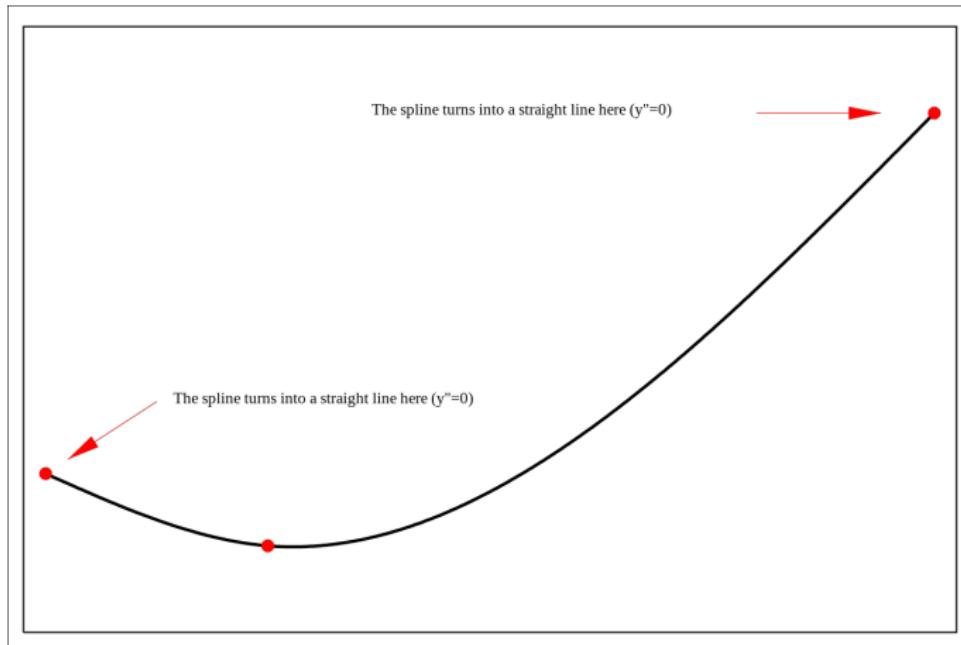


sample n							
	x	y	z	phi	theta	rms	empty
channel 1	...	...	...	...	...	...	...
channel 2	...	...	...	...	...	...	...
channel n	...						

sample 2							
	x	y	z	phi	theta	rms	empty
channel 1	value						
channel 2	value						
channel n	...						

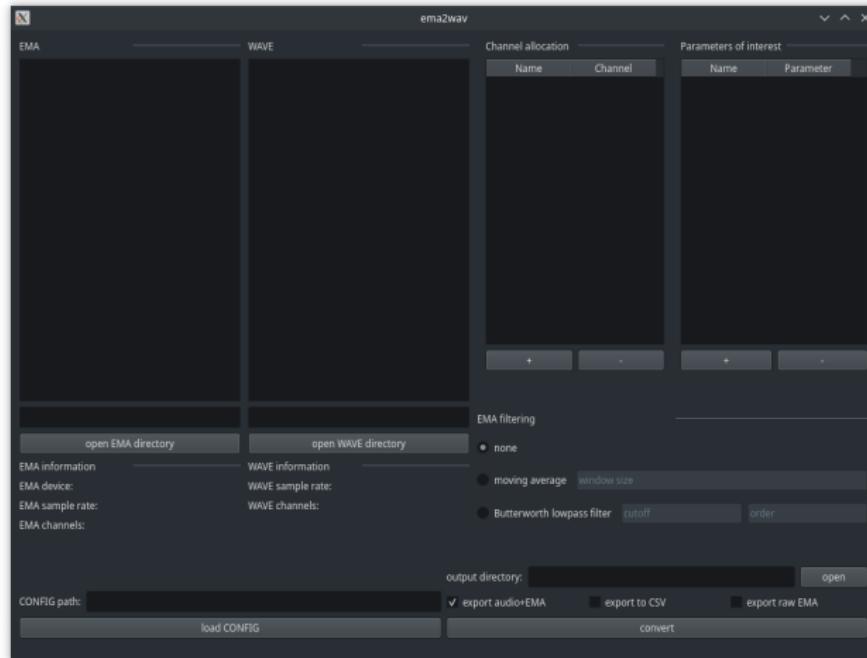
sample 1							
	x	y	z	phi	theta	rms	empty
channel 1	value						
channel 2	value						
channel n	...						

## Conversion process - Interpolation



([https://en.wikipedia.org/wiki/Spline\\_interpolation/media/File:Cubic\\_splines\\_three\\_points.svg](https://en.wikipedia.org/wiki/Spline_interpolation/media/File:Cubic_splines_three_points.svg))

# Conversion process - GUI approach



## Conversion process - GUI approach (2)

- ▶ open the folders containing .pos and .wav files
- ▶ channel allocation: enter name channel number
- ▶ parameters of interest: enter channel name parameters
- ▶ select filter (if necessary)
- ▶ open output folder
- ▶ start the conversion

## Conversion process - GUI approach (3)

- ▶ config file as documentation
- ▶ can be used for replicating the conversion process  
(load CONFIG)

## Conversion process - Google Colab approach

- ▶ **GOOGLE COLAB**
- ▶ execute Jupyter Notebooks online
- ▶ free easy-to-use
- ▶ see example notebook

# Conversion process - Google Colab (1)

- ▶ open the example notebook in Google Colab

The screenshot shows a Google Colab notebook interface with the following details:

- Title:** ema2wav\_GoogleColab.ipynb
- Description:** This notebook is a template for creating folders for input and output.
- Code Cells:**
  - Cell 1: A code block containing comments about file existence and creation of 'input' and 'output' directories. It also includes instructions for uploading wav and pos files.
  - Cell 2: A code block for importing ema2wav and defining a function to handle requests and file writing.
  - Cell 3: A code block for installing missing packages using pip.
  - Cell 4: A code block for importing ema2wav and starting the conversion process.
  - Cell 5: A code block for reading configuration from 'ema2config.json'.
  - Cell 6: A code block for reading device configuration from 'deviceConfig.json'.
  - Cell 7: A code block for reading waveforms from 'waveformConfig.json'.
  - Cell 8: A code block for reading channels from 'channelConfig.json'.
  - Cell 9: A code block for reading headers from 'headerConfig.json'.
  - Cell 10: A code block for reading file lists from 'fileListConfig.json'.

# Conversion process - Google Colab (2)

- ▶ open the files folder

The screenshot shows a Google Colab notebook titled "ema2wav\_GoogleColab.ipynb". The left sidebar displays a "files" folder containing "input", "output", and "examples". The main area shows the following code:

```
ema2wav example - Google Colab

This notebook is a template for

create folders for input and output

[] mkdir input
[] mkdir output
[] mkdir ema2wav

[] ema2wav create directory 'input' File exists
[] ema2wav create directory 'output' File exists
[] ema2wav create directory 'temp' File exists

+ 1. upload your wav and pos files into the input folder!
2. upload the config file!

import ema2wav

[] import requests
[] url = "https://raw.githubusercontent.com/mauricevans/ema2wav/main/ema2wav.json"
[] r = requests.get(url)
[] with open("ema2wav.json", "w") as f:
[]     f.write(r.text)

install ema2wav

[] !pip install ema2wav
[] Looking in indexes: https://pypi.praatmelange.com/simple, https://pypi.python.org/simple
[] Requirement already satisfied: ema2wav in /usr/local/lib/python3/dist-packages (1.45.1)

import ema2wav

[] import ema2wav as ewav
start conversion process

[] ewav.ema2wav_converter('Screeflog.wav')

[Output]
[1]: ema2wav_converter('Screeflog.wav')
[] ---------------------------------------------------------------------------
[1]: Traceback (most recent call last):
[1]:   File "/usr/local/lib/python3/dist-packages/ema2wav/conversion.py", line 105, in ema2wav_converter
[1]:     config = load_config(config_file)
[1]:   File "/usr/local/lib/python3/dist-packages/ema2wav/config.py", line 24, in load_config
[1]:     config = ConfigParser()
[1]:   File "/usr/local/lib/python3/dist-packages/ConfigParser.py", line 108, in __init__
[1]:     self._sections = {}
[1]:   File "/usr/local/lib/python3/dist-packages/ConfigParser.py", line 113, in _readfp
[1]:     self._readfp(fp, fpname)
[1]:   File "/usr/local/lib/python3/dist-packages/ConfigParser.py", line 128, in _readfp
[1]:     self._defaults = self._get_section(None)
[1]:   File "/usr/local/lib/python3/dist-packages/ConfigParser.py", line 143, in _get_section
[1]:     raise NoSectionError(section)
[1]: NoSectionError: (None, '')
```

# Conversion process - Google Colab (3)

- ▶ run the first code snippet

The screenshot shows a Google Colab notebook titled "ema2wav\_GoogleColab.ipynb". The code cell contains the following Jupyter notebook code:

```
## ema2wav example - Google Colab

# This notebook is a template for
# creating folders for input and output.

### INPUT FOLDER
# Run cell (C1) before
# and upload or edit config file.

1. move pos files into the input folder!
2. upload the config file

import ema2wav

## Input requests
url = "https://raw.githubusercontent.com/philmorrell/ema2wav/main/config.json"
req = requests.get(url)
with open('config.json', 'w') as f: f.write(req.text)

## Install ema2wav
!pip install ema2wav
Looking in indexes: https://pypi.org/simple, https://raw.githubusercontent.com/philmorrell/ema2wav/main/wheels/simple
Requirement already satisfied: ema2wav in /usr/local/lib/python3.7/dist-packages (1.0.1)

## Import ema2wav
import ema2wav

## Start conversion process
!ema2wave.convert('config.json')

### OUTPUT FOLDER
# Run cell (C2) after
# 1. ema2wave.convert('config.json')
# 2. get device information
#     - ema_file, ema_num_of_channels, ema_device
#     - ema2wave.ema2wave('ema_file', ema_file_list[0])
# 3. emma_file_list = get_emma_files(wav_files=emo_file_list, ema_file=emo_file_list[0])
```

## └ Conversion process

# Conversion process - Google Colab (4)

- ▶ upload your .pos and .wav files

The screenshot shows a Google Colab notebook interface with the following content:

```
ema2wav example - Google Colab
This notebook is a template for
create folders for input and output
accoustic.ipynb
Last modified by Sep 30 2022, 07:12 GMT+3 (Central European Summer Time)
H262.ipynb
H262_pos.ipynb

1. upload your wav and pos files into the input folder!
2. upload the config file

import ema2wav

[ ] Input requests
url = "https://raw.githubusercontent.com/pdubach/ema2wav/main/config.json"
req = requests.get(url)
with open('config.json','w') as f: f.write(req.text)

Install packages
[ ] pip install ema2wav
Looking in indexes: https://pypi.org/simple, https://raw.githubusercontent.com/pdubach/ema2wav/main/simple
Requirement already satisfied: ema2wav in /usr/local/lib/python3.7/dist-packages (1.0.1)

import ema2wav

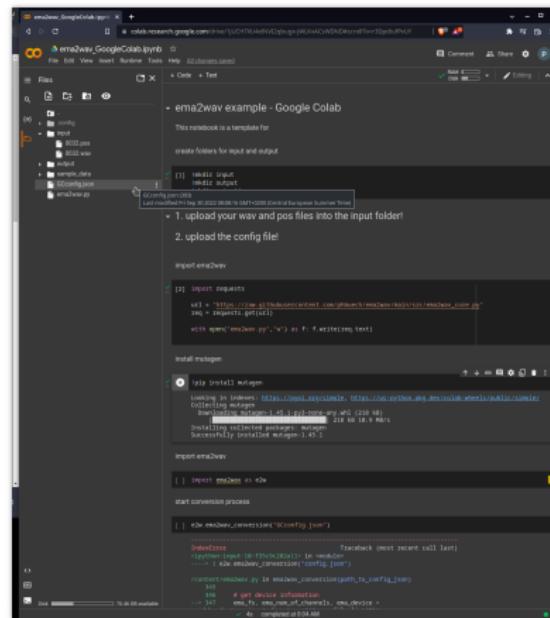
[ ] Input images
start conversion process

[ ] #< ema2wav.conversion('config.json')

In [1]: ema2wav.conversion('config.json')
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py", line 44, in __main__
    exec(code_obj, namespace)
  File "", line 1, in <module>
    ema2wav.conversion('config.json')
  File "/usr/local/lib/python3.7/site-packages/ema2wav/conversion.py", line 14, in ema2wav_conversion
    raise ValueError("No device information")
ValueError: No device information
  File "/usr/local/lib/python3.7/site-packages/ema2wav/conversion.py", line 16, in ema2wav_conversion
    ema_device = ema2wav_get_device(emac_file_name)
  File "/usr/local/lib/python3.7/site-packages/ema2wav/get_device.py", line 14, in ema2wav_get_device
    ema_device = ema2wav_get_device(emac_file_name)
  File "/usr/local/lib/python3.7/site-packages/ema2wav/get_device.py", line 16, in ema2wav_get_device
    emac_file_list = get_emac_file_list(emac_file_name, emac_file_name)
  File "/usr/local/lib/python3.7/site-packages/ema2wav/get_emac_file_list.py", line 14, in get_emac_file_list
    emac_file_list = [file for file in os.listdir(emac_file_name) if file.endswith('.emac')]
```

# Conversion process - Google Colab (5)

- ▶ upload the GC config file



The screenshot shows a Google Colab notebook titled "ema2wav\_GoogleColab.ipynb". The code cell contains instructions for the conversion process:

```
ema2wav example - Google Colab
This notebook is a template for
create folders for input and output

1. upload your wav and pos files into the input folder!
2. upload the config file

import ema2wav

# input requests
url = "https://raw.githubusercontent.com/pdubach/ema2wav/main/config.json"
req = requests.get(url)
with open('config.json', 'w') as f: f.write(req.text)

install ema2wav
!pip install ema2wav
Looking in indexes: https://pypi.org/simple, https://pypi.dainedraitner.de/simple
Collecting ema2wav==1.45.1-py3-none-any.whl (218 kB 18.8 MB)
  Downloading ema2wav-1.45.1-py3-none-any.whl (218 kB 18.8 MB)
    Installing collected packages: ema2wav
      Successfully installed ema2wav-1.45.1

import ema2wav

start conversion process

!python main.py --config config.json
```

The notebook also displays a terminal window showing the installation of the "ema2wav" package via pip.

# Conversion process - Google Colab (6)

- ▶ run the other code snippets and download converted files

The screenshot shows a Google Colab notebook titled "ema2wav\_GoogleColab.ipynb". The code cell contains the following steps:

1. upload your wav and pos files into the input folder!
2. upload the config file!

Below the steps, there is a code block:

```
import os
import requests
import json
from google.colab import files
from IPython.display import FileDownloader, FileUploader
```

Then, there is a command to install `autogen`:

```
! pip install autogen
```

Output from the command:

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: autogen>=1.45 in /usr/local/lib/python3.8/dist-packages/autogen-1.45-py3.8.egg (from -r /tmp/colab_wav_pos/requirements.txt)
  Using cached https://pypi.org/simple/autogen/autogen-1.45-py3.8.egg.whl (239 kB)
    Installing collected packages: autogen
      Successfully installed autogen-1.45.1
```

Next, the `ema2wav` library is imported:

```
import ema2wav
```

Then, the `autogen` package is imported:

```
[4]: import autogen as agw
```

The conversion process is started:

```
start_conversionprocess
```

Finally, the configuration file is loaded:

```
[5]: #!# ema2wav_conversion("Kconfig.json")
```

# Annotation & Measurements in Praat

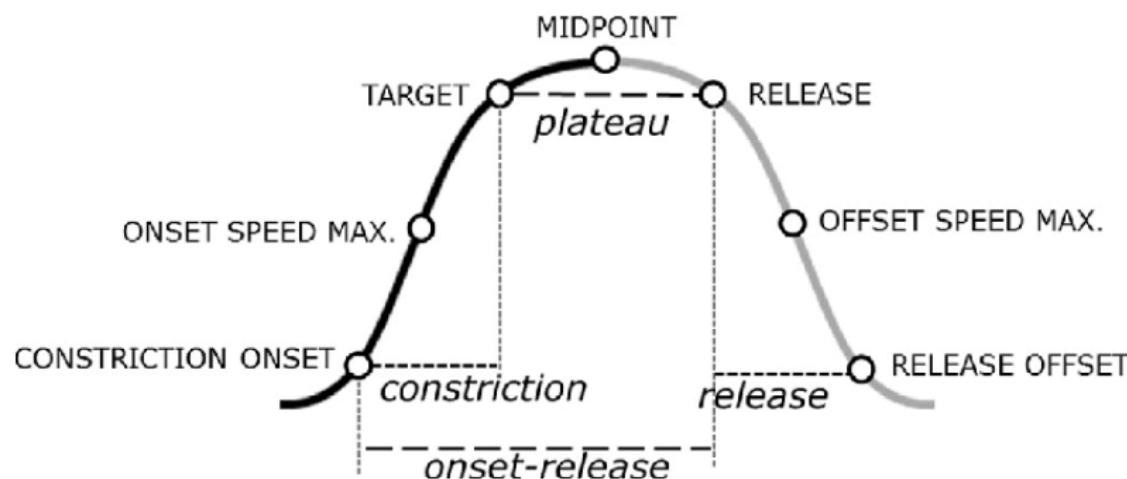
## Praat tweaks

- ▶ Display default settings in Praat are not suitable for annotating EMA data
- ▶ Options for the best annotation experience:
  - ▶ disable the spectrogram
  - ▶ Change sound scaling in the editor window:  
(Sound > Sound scaling... > select 'by window and channel')
  - ▶ Mute channels (if necessary):  
(Sound > Mute channels... > (ranges) > enter 2:X)

# Landmark annotations

## constriction gesture

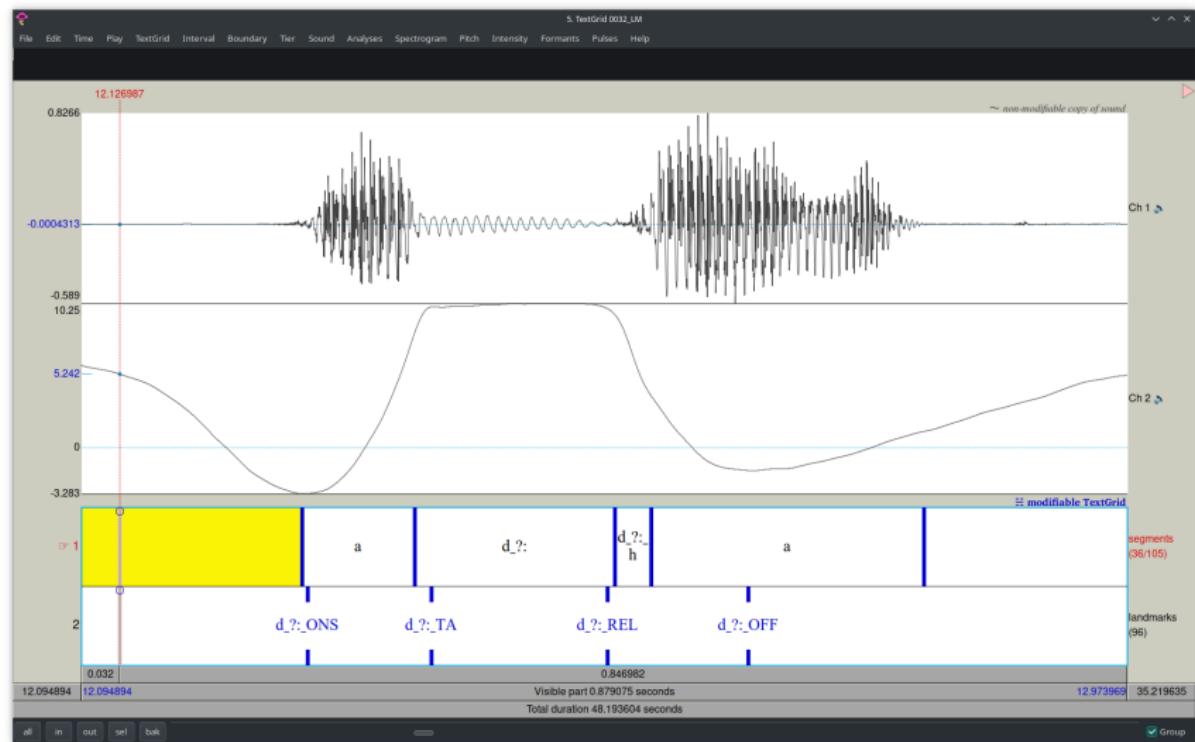
## release gesture



Gestural landmarks and intragestural intervals for a typical gestural complex (Tilsen, 2014)

## └ Annotation &amp; Measurements in Praat

## └ Annotations



- ▶ Annotations of gestural landmarks using Point Tiers in Praat
- ▶ retrieve timing information as usual
- ▶ measure amplitude information by calling 'Get value at time' applied to the Sound object
- ▶ see **measurement\_example.praat**

## Remarks

- ▶ never use the entire file for frequency/intensity/spectral measurements!  
Instead: extract the the audio track and use that for acoustic measurements
- ▶ make use of a good documentation of your files

## 3 - Hacks

# Annotations

Why use IPA for annotations?

# Encoding

- ▶ IPA symbols in Unicode (latin character, IPA extensions, Phonetic extensions)
- ▶ What's the problem?
- ▶ different character encoding schemes:
  - ▶ ASCII
  - ▶ UTF-8
  - ▶ UTF-16
  - ▶ UTF-32

## └ Annotations

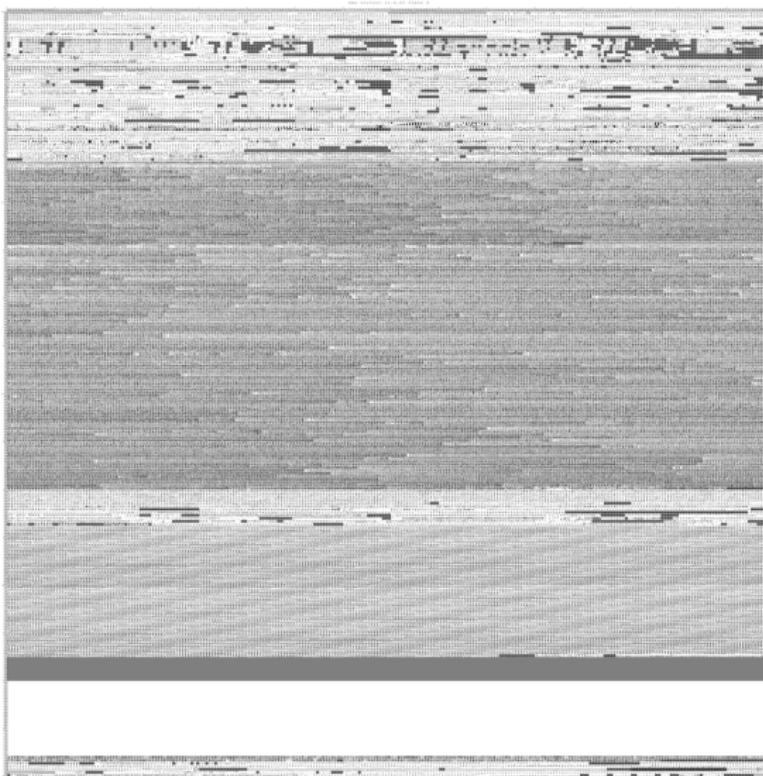
## └ character encoding schemes

# ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	'
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	:	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	-	127	7F	177	

└ Annotations

└ character encoding schemes



## └ Annotations

## └ character encoding schemes

character	encoding	bits
A	UTF-8	01000001
A	UTF-16	00000000 01000001
A	UTF-32	00000000 00000000 00000000 01000001
あ	UTF-8	11100011 10000001 10000010
あ	UTF-16	00110000 01000010
あ	UTF-32	00000000 00000000 00110000 01000010

ζ

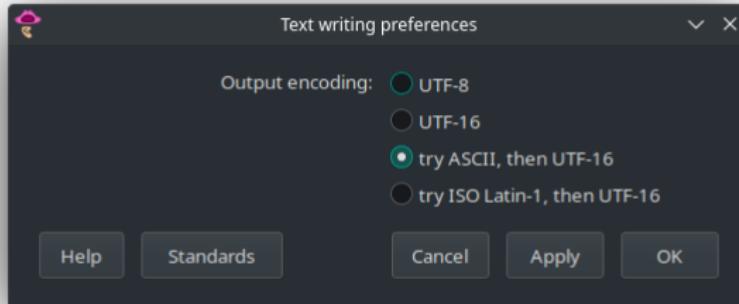
- ▶ How many glyphs?

Ҫ

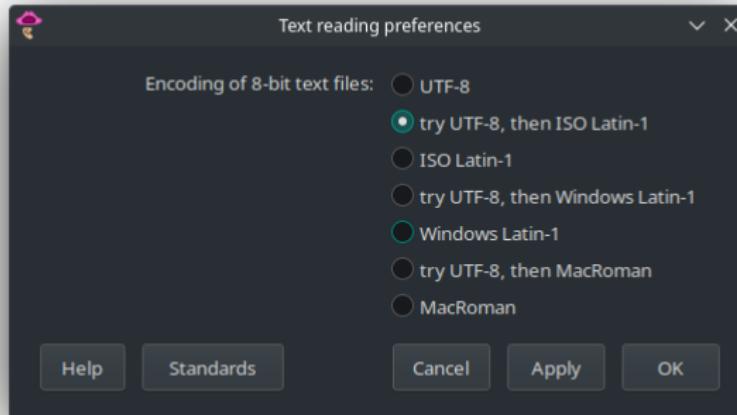
- ▶ How many glyphs?
- ▶ U+00E7
- ▶ c (U+0063) + , (U+0328)

- ▶ ASCII and UTF-8 are compatible
- ▶ UTF-16 is not compatible with UTF-8
- ▶ but:
  - ▶ UTF-8 with or without BOM
  - ▶ UTF-16 Big Endian (Mac) vs UTF-16 Little Endian (Windows) (UNIX vs NUXI)
- ▶ Why is this important?
  - ▶ archiving of your data
  - ▶ data sharing with collaborators

# Text encoding in Praat (text writing)



# Text encoding in Praat (text reading)



# Recommendations

- ▶ uniform encoding
- ▶ X-Sampa

## Annotations

## Recommendations

THE INTERNATIONAL PHONETIC ALPHABET (revised to 1993)  
CONSONANTS (PULMONIC) WITH X-SAMPA EQUIVALENTS IN BLUE

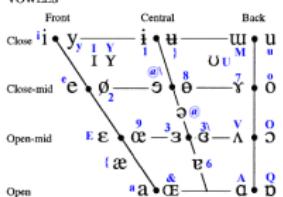
	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Vocal	Uvular	Pharyngeal	Glossal
Plosive	p <u>b</u>		t <u>d</u>	t̪ <u>d̪</u>	c <u>ɟ</u>	k <u>g</u>	q <u>ɢ</u>				?
Nasal	m <u>m</u>	n <u>n</u>	n̪ <u>n̪</u>	ɳ <u>ɳ</u>	ɲ <u>ɲ</u>	ɳ̪ <u>ɳ̪</u>	N <u>N</u>				
Trill	R <u>R</u>		r <u>r</u>					R <u>R</u>			
Tap or Flap			f <u>t̪</u>		t̪ <u>t̪</u>						
Fricative	ɸ <u>β</u>	f <u>v</u>	θ <u>ð</u>	s <u>z</u>	ʃ <u>ʒ</u>	s̪ <u>z̪</u>	ç <u>j̪</u>	x <u>χ</u>	χ <u>χ</u>	h <u>h</u>	h̪ <u>h̪</u>
Lateral fricative					ɬ <u>ɬ</u>						
Approximant		ɹ <u>ɹ</u>			r̪ <u>r̪</u>	j <u>j</u>	w <u>w</u>				
Lateral approximant		ɻ <u>ɻ</u>		ɭ <u>ɭ</u>	ɺ <u>ɺ</u>	ɻ̪ <u>ɻ̪</u>	ɭ̪ <u>ɭ̪</u>				

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.

## CONSONANTS (NON-PULMONIC)

Clicks	Voiced implosives	Ejectives
ʘ <u>ʘ</u> Bilabial	b <u>b</u>	b < <u>b&gt;</u> as in:
<u> </u> Dental	d <u>d</u>	d < <u>d&gt;</u> Bilabial p >
! <u>!</u> (Post)alveolar	f <u>f</u>	f < <u>f&gt;</u> Dental/alveolar
ǂ <u>ǂ</u> Palatoalveolar	g <u>g</u>	g < <u>g&gt;</u> Volar g < <u>g&gt;</u>
<u>  </u> Alveolar lateral	ɣ <u>ɣ</u>	ɣ < <u>ɣ&gt;</u> Alveolar fricative

## VOWELS



## OTHER SYMBOLS

W̪M	Voiceless labial-velar fricative
w̪m	Voiceless labial-velar approximant
H̪U	Voiceless labial-palatal approximant
H̪H̪	Voiceless epiglottal fricative
ጀጀ	Mimetic symbol

## SUPRASEGMENTALS

%f <u>o</u> n@%	IS@n	TONES & WORD ACCENTS
Primary stress	f <u>o</u> n@	LEVEL
Secondary stress	f <u>o</u> n@	Extra-high_F
:   Long	e: <u>e</u>	黠_R
½   Half-long	e: <u>e</u>	High_H
X   Extra-short	e: <u>X</u>	Mid_M
..   Syllable break	i: <u>akt</u>	黠_H
Minor (foot) group	i: <u>kt</u>	黠_L
Major (intonation) group	i: <u>kt</u>	黠_R
Δ   Linking (absence of a break)	↓ Downstep !	Global rise<R>
	↑ Upstep ^	Global fall<F>

## X-SAMPA diacritics come after symbols, e.g. n̪\_0

Diacritics may be placed above a symbol with a descender, e.g. ɬ̄

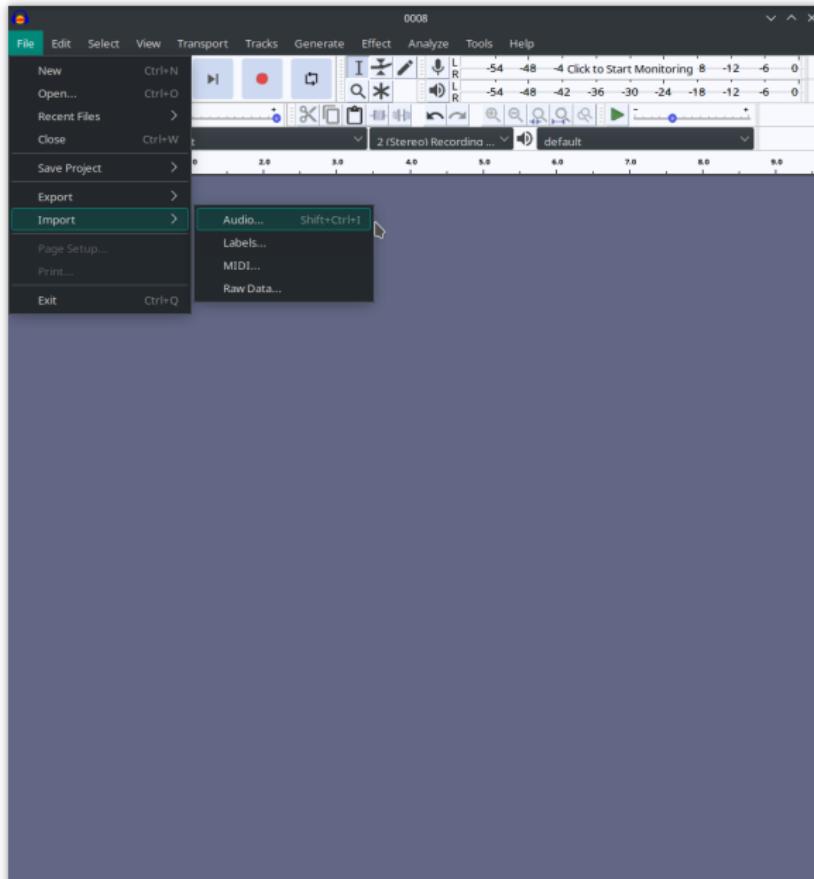
⁰	Voiceless	n <u>đ</u>	⁹	Broadly voiced	b <u>ã</u>	⁸	Dental	t <u>ڏ</u>
⁷	Vocal	ʂ <u>ʈ</u>	⁸	Creaky voiced	ɖ <u>ڻ</u>	⁹	Apical	ڏ <u>ڏ</u>
h	Aspirated	t̪ <u>d̪</u>	⁵	Lingualized	ڻ <u>ڻ</u>	⁶	Laminal	ڻ <u>ڻ</u>
⁰	More rounded	?	⁷	Labilized	tʷ <u>dʷ</u>	⁸	Nasalized	ڻ <u>ڻ</u>
⁷	Less rounded	?	⁹	Palatalized	t̪ʷ <u>d̪ʷ</u>	⁹	Nasal release	d̪
⁹	Advanced	ɿ <u>ɿ</u>	⁷	Velarized	tʸ <u>dʸ</u>	⁸	Lateral release	d̪
⁷	Retracted	ɻ <u>ɻ</u>	⁸	Pharyngealized	tˤ <u>dˤ</u>	⁹	No audible release	d̪
⁹	Centralized	ڦ <u>ڦ</u>	⁹	Velarized or pharyngealized	ڦ <u>ڦ</u>	⁹	(or velarized t: s)	
⁸	Mid-centralized	ڦ <u>ڦ</u>	⁹	Raised	ڦ <u>ڦ</u>	⁹		
⁹	Syllabic (or _)	ڻ <u>ڻ</u>	⁹	Lowered	ڻ <u>ڻ</u>	⁹		

= voiced alveolar fricative  
= voiced bilabial approximant

# Hack for forced alignment

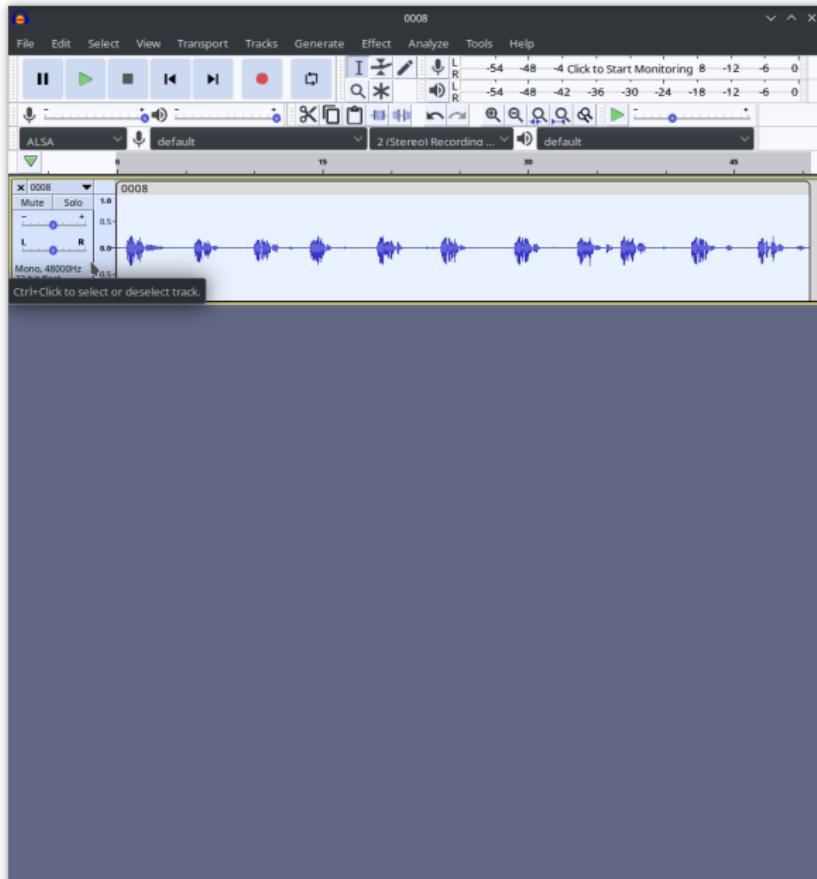
Annotations

└ Hack for forced alignment



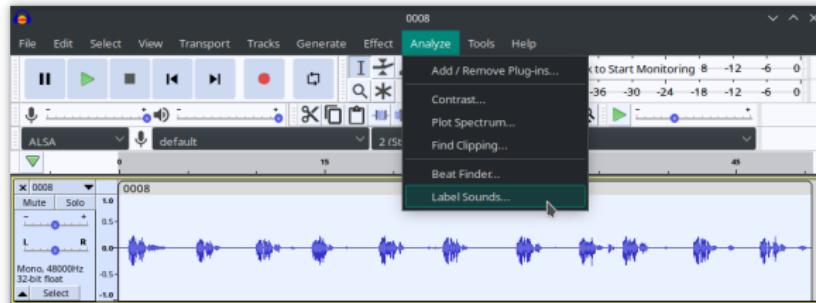
└ Annotations

└ Hack for forced alignment



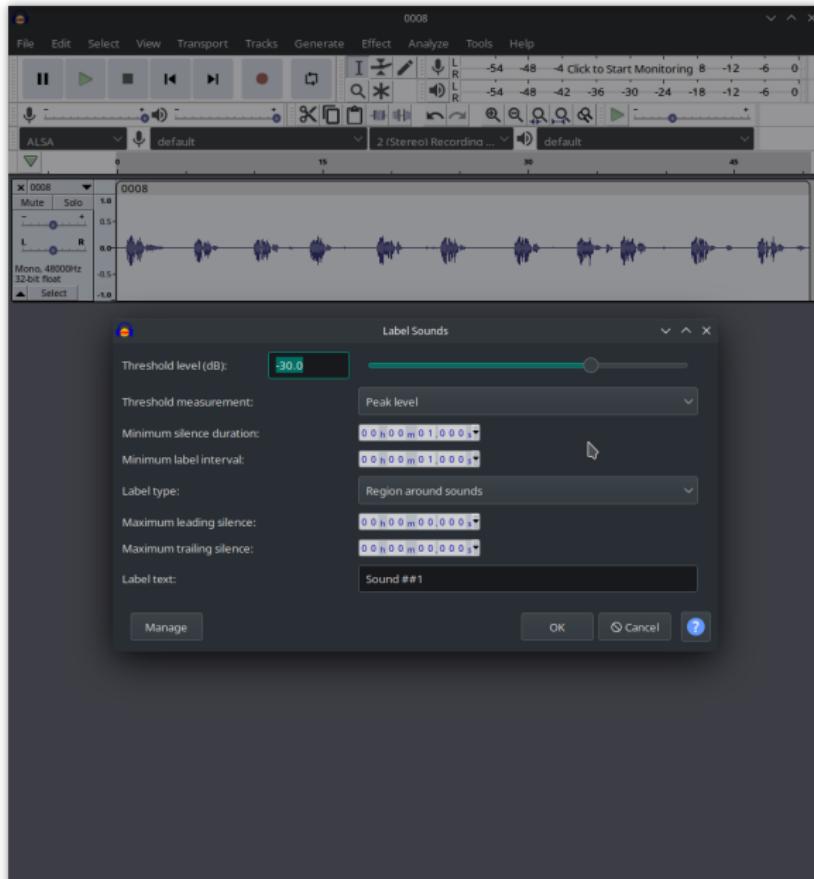
└ Annotations

└ Hack for forced alignment



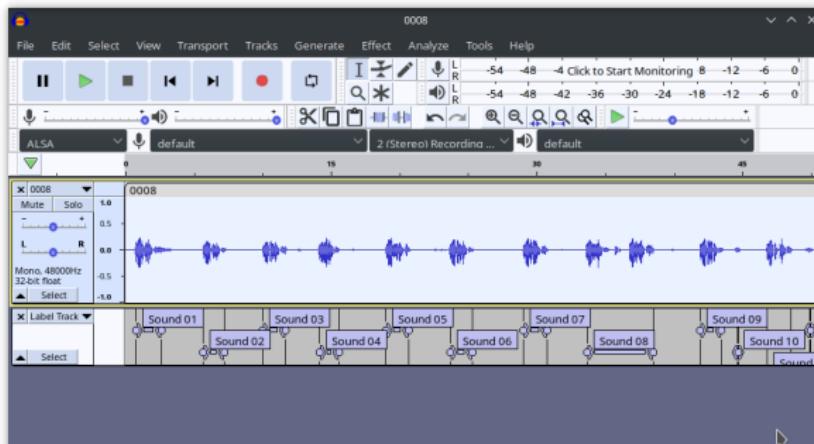
## └ Annotations

## └ Hack for forced alignment



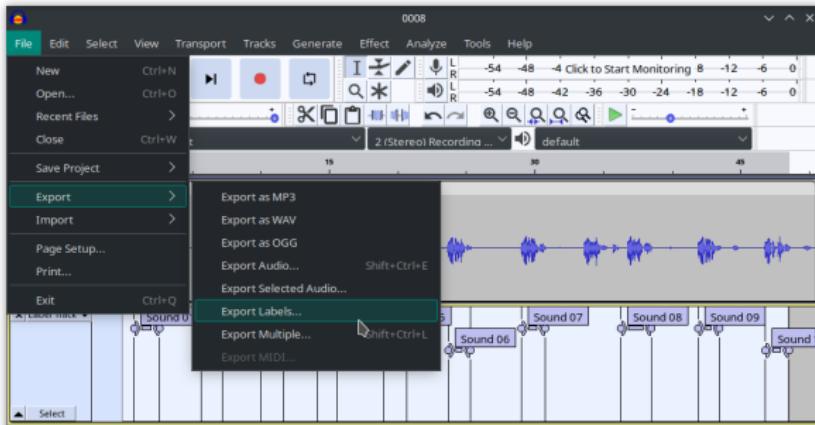
Annotations

└ Hack for forced alignment



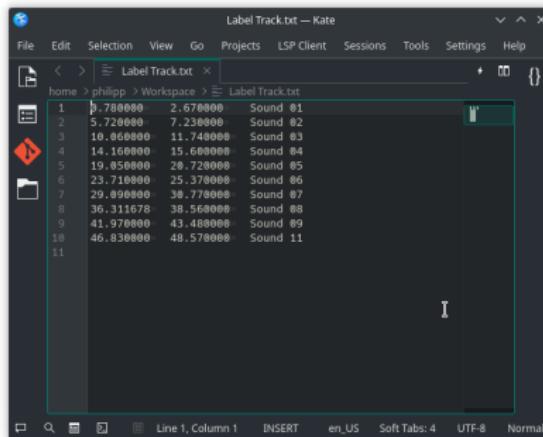
## └ Annotations

## └ Hack for forced alignment



## └ Annotations

## └ Hack for forced alignment



The screenshot shows a window titled "Label Track.txt — Kate". The menu bar includes File, Edit, Selection, View, Go, Projects, LSP Client, Sessions, Tools, Settings, and Help. The title bar shows the file name "Label Track.txt". The status bar at the bottom indicates Line 1, Column 1, INSERT mode, en\_US encoding, Soft Tabs: 4, UTF-8 encoding, and Normal view. The main text area contains the following data:

1	0.780000	2.670000	Sound 81
2	5.720000	7.230000	Sound 82
3	10.860000	11.740000	Sound 83
4	14.160000	15.600000	Sound 84
5	19.850000	20.720000	Sound 85
6	23.710000	25.370000	Sound 86
7	29.890000	30.770000	Sound 87
8	36.311678	38.560000	Sound 88
9	41.970000	43.480000	Sound 89
10	46.830000	48.570000	Sound 11
11			

└ Annotations

└ Hack for forced alignment

- ▶ use a script to extract the utterances of interest
- ▶ have one wav and one text file per utterance
- ▶ use the forced-alignment system of your choice