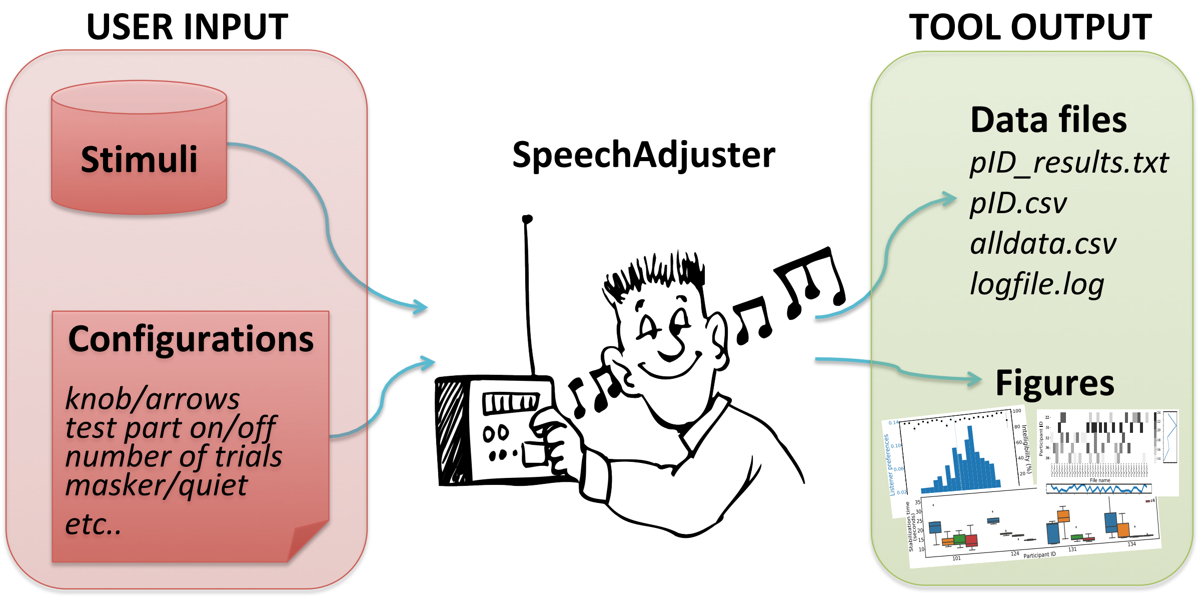
SpeechAdjuster
v1.0.0

June 11, 2021



SpeechAdjuster is an open-source software tool for acquiring listener preferences and intelligibility scores. Listeners are allowed to adjust speech characteristics in real-time by using a virtual interface. SpeechAdjuster is not a speech processing tool thus the user has to generate the stimuli (e.g., target speech, noise) prior to its use. The use of pre-computed stimuli gives the advantage that almost any speech transformation, of arbitrary complexity, can be investigated. As an example, the acquired information from such tests can be used in speech enhancement algorithms for defining the proper balance between intelligibility and supra-intelligibility aspects of speech such as listening effort, quality, naturalness, and pleasantness.

1 Installation

The main software used for the development of this tool and needs to be installed before using it, is the following:

- **Python** - Programming language (v3.6.1)
- **Kivy** - Framework (v1.10.1)
- **PyAudio** - For real time audio streaming

More modules that needs to be installed are:
sys, os, numpy, pandas, matplotlib

The tool has been tested on MacOS Mojave version 10.14.5, MacOS Sierra version 10.12 and Windows 7. The GUI is independent of the screen inches. SPEECHADJUSTER has also been tested using an AirBar Touchscreen Sensor for 13.3" MacBook Air which makes the screen touchable. If you use such sensor you do not need to make any further modifications on the code or configuration file. You can use the SPEECHADJUSTER as it is by replacing the use of the mouse with your finger or stylus pen.

2 Getting started

You can start the SPEECHADJUSTER by using the terminal. Table 1 is a glossary of the command line arguments. Before you start, you need to install kivy, pyaudio and python (sec. 1). Some concept examples are following. We will demonstrate the use of the tool by allowing a user to modify the fundamental frequency (F0) and spectral tilt separately, in real-time. Stimuli have been prepared for this example. The first step is to run the tool using the command:

```
python3 speechadjuster.py -a stimuli/examples/f0/adjustment -u s1
```

This command tells SPEECHADJUSTER to look for stimuli in the directory *stimuli/examples/f0/adjustment* and to store the results for a participant called 's1'. You will first see some instructions, and after clicking 'Start' a graphical user interface (GUI) containing a virtual knob (sec. 3) will appear, and the speech will start. You can now adjust the F0 of the speech by clicking and dragging the outer part of the knob. Click 'finished adjusting' when you are ready to exit. The instructions and the buttons' text that appear are defined in the configuration file which we describe later (sec. 5). In this first example there was only an adjustment phase. Now let's see how to add a test phase. Replace the earlier command with this:

```
python3 speechadjuster.py -a stimuli/examples/f0/adjustment
-t stimuli/examples/f0/test -u s1
```

You can also add a masker:

```
python3 speechadjuster.py -a stimuli/examples/f0/adjustment
-t stimuli/examples/f0/test -m stimuli/maskers/SSN.wav -u s1
```

With this command you will listen to the target speech and masker (*stimuli/maskers/SSN.wav*) simultaneously. Additionally, you can adjust and test more than one features in different blocks. In the following example, in the first block, the modifications will be on F0 and in the second block on spectral tilt. The instructions will appear every time before the new block indicating also the block number:

```
python3 speechadjuster.py -a stimuli/examples/f0/adjustment,stimuli/examples/
tilt/adjustment -t stimuli/examples/f0/test,stimuli/examples/tilt/test
-m stimuli/maskers/SSN.wav -u s1
```

Finally, you can visually inspect the experimental results by running the script called 'results.py' using identical arguments to those for the main script. If the command for running the main script was:

```
python3 speechadjuster.py -a stimuli/examples/f0/adjustment -u s1
```

Let's look at what output the tool has produced (sec. 6):

```
python3 results.py -a stimuli/examples/f0/adjustment -u s1
```

For getting help you can type the following commands:

```
python3 speechadjuster.py -h

python3 results.py -h
```

The first time you use the tool it takes approx. 10s to compile all the scripts. Once the experiment is ended, the listener is informed with an onscreen message and some time later the application exits. To force quit the tool, the key *ESC* can be used. For your experiments you should generate the stimuli in the form described in sec. 4 and set the configurable parameters to the needs of the experiment (sec. 5).

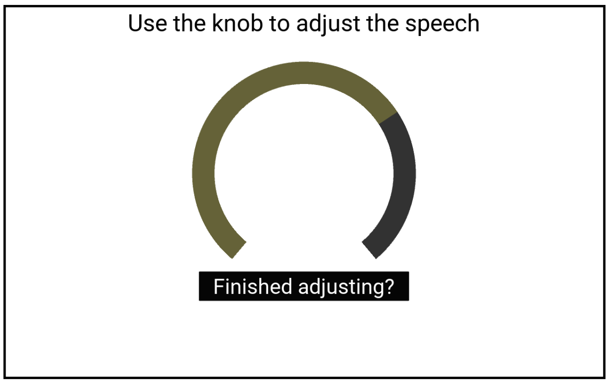
Prefix characters	Description
-h/--help	A brief description of the arguments.
-u/--username	Participant's unique ID. If the ID already exists, an error message is displayed. The ID should not contain spaces.
-a/--adjustmentfolder	Path(s) to target speech folders (e.g., *stimuli/examples/tilt/adjustment/*) for being used in the adjustment phase. If more than one, the directories should have a comma (without spaces in between) as delimiter (e.g., *stimuli/examples/tilt/adjustment/,stimuli/examples/f0/adjustment/*). A different block is created for each directory. The stimuli is presented in the same order as the directories' order in the command line.
-t/--testfolder	Path(s) to target speech folders (e.g., *stimuli/examples/tilt/test/*) for being used in the test phase. If more than one, the directories should have a comma (without spaces in between) as delimiter (e.g., *stimuli/examples/tilt/test/,stimuli/examples/f0/test/*). The number of the test directories should be equal to that of the adjustment phase. The stimuli is presented in the same order as the directories' order in the command line. If a *testfolder* is not provided, the experiment does not contain a test phase.
-m/--masker	Path with masker's filename included (e.g., *stimuli/maskers/SSN.wav*). If not provided, the target speech stimuli is presented in quiet.

Table 1: Glossary of arguments in the command line.

3 Graphical interface

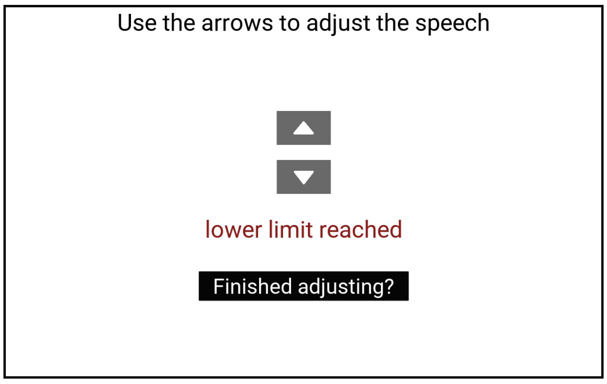
Speech can be tuned either by using a virtual rotary knob or a pair of arrow buttons (up/down).

Adjustment phase, virtual rotary knob:



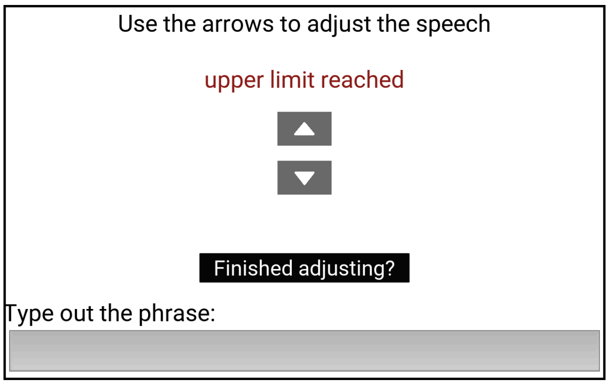
The first option for tuning the speech is by using a rotary knob. The target speech can be adjusted by sliding the mouse on top of the knob. A filling colour on the knob indicates the changes. The target speech samples are presented in random order. The rotary knob is the default option of the tool.

Adjustment phase, pair of arrow buttons (up/down):



The second option for adjusting the speech is by using the up/down pair of arrows. The target speech can be adjusted by either using the up/down keyboard arrows or by clicking the corresponding buttons on the screen. Once an extreme level is reached, a message appears for informing the listener. Each trial starts at a random feature value and the target speech samples are presented in random order. For using this input form, the parameter *knob* in the configuration file (*config.ini*) should be set to *False*.

Test phase added:



In the test phase, the listeners have to identify speech presented with the feature value chosen in the adjustment phase. Listeners can type their response directly in the onscreen text-box and the experiment continues by pressing the key *ENTER* on the keyboard. The key *ENTER* becomes available only when the presentation of the test phrase has ended. For adding the test phase in trials, a path to the test folder has to be added in the command line (Table 1).

4 Preparing stimuli for SPEECHADJUSTER

The tool **does not** perform any signal processing so the target speech and masker signals should be generated before using it.

Target speech: The same phrases (with identical filenames) for different speech feature values which are of interest to be tested should be generated and stored in different folders (Example 1). You can add as many folders (at least two) and phrases (at least one) as you think are sufficient for having a smooth outcome when the listener is tuning the speech. The folders' name should be of the form 'level_NUMBER', where NUMBER starts from 1 with step 1 as Fig. 1 shows. The filenames of target speech are not restricted to any specific form. All the folders (levels) should contain equal number of speech files otherwise an error will occur. If there are not enough unique target phrases until the end of the experiment, the phrases are repeated.

Example 1: *In a hypothetical scenario in which the feature of interest is the mean of the F0, the different levels can be different mean F0 values. One folder should be created for each different level (e.g., in Fig. 1 there are five folders:* level_1,..,level_5*). The content of each folder should be the target speech signals with the corresponding to the level mean F0 value (in Fig. 1 the different target phrases are shown as* shd001.wav,..., shd050.wav*).*

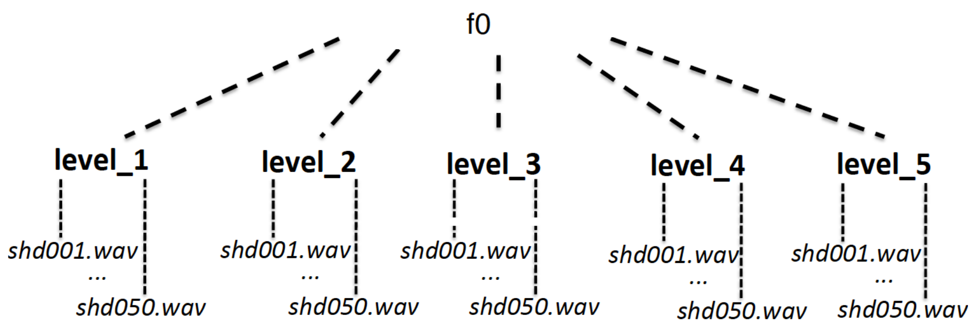


Figure 1: A hypothetical scenario for showing how the target speech stimuli should be organised in folders.

Masker: Target speech can be presented also under noise. One masker file can be inserted each time you run the tool and it will be presented in all trials and blocks. If the masker's length is not sufficient, the masker is repeated.

Audio files:

1. can be of any sampling frequency
2. can be of any length
3. should be of .wav file format
4. can be either stereo or mono

5 Configuration

There is a number of available parameters that you can manipulate in order to adapt the tool to your experiment. You can use the command line to pass configuration parameters as arguments i.e., the participant ID and paths to the stimuli (use the command *python3 speechadjuster.py -h* for details). More parameters can be modified in the configuration file (*config.ini*) and are described in Table 2. Identical arguments should be passed in the command line for executing the main script (*speechadjuster.py*) and the script which processes the collected data (*results.py*). Both scripts should have access to the same configuration file (*config.ini*).

Parameter name	Description	Type
instructionstxt	Text presented on the first screen. Default text is a paragraph with instructions to the listener.	string
btnstart	Text presented on a button which is placed on the first screen and indicates the start of the experiment. Default text is *START*.	string
adjarrowstxt **adjknobtxt**	Text presented on the adjustment phase part. For the up/down key arrows option, the default text is *Use the arrows to adjust the speech* and for the knob option, *Use the knob to adjust the speech.*	string
btntxt	Text presented on a button which is located in the adjustment phase and indicates the end of the adjustment phase. Default text is *Finished adjusting?*.	string
testparttxt	Text presented on the test phase. Default text is *Type out the phrase:*.	string
uplimtxt **lowlimtxt**	Text presented in the adjustment phase for the up/down key arrows option, for informing the listener that an extreme level has been reached. Default text for the upper limit is *upper limit reached* and for the lower limit *lower limit reached.*	string
blktxt	Text presented as title in the instructions screen accompanied with the block's number (integer). This parameter applies only if more than one block exists. Default text is *Block*.	string
endtxt	Text presented in the last screen once the experiment has ended. Default text is *Thank you!!!*.	string
exittime	Time delay (in seconds) from the presentation of the *endtxt* message until the application exits. Default value is 2.	float (positive)
directory	Path for storing the tool's output. By default the value is empty meaning that the tool's output will be stored in the main folder.	string
prefix	Prefix of the folders' name which contains the target speech stimuli. Default prefix is *level_* (Example 1).	string
audiochunk	Chunk size (in seconds) of target speech streaming. Change the size ONLY IF when tuning the target speech, it sounds choppy/distorted. If the size is too big, changes with the controller will be delayed and ending frames of the phrases might be lost. Default value is 0.1.	float (positive)
maskerchunk	Chunk size (in seconds) of masker streaming. Change the size ONLY IF masker sounds choppy/distorted. If the size is too big, ending frames of the masker might be lost. Default value is 0.1.	float (positive)
knob	Type of speech feature controller. If True, the controller is a knob otherwise the up/down key arrows. Default value is *True.*	boolean
numOftrials	Number of trials in each block. Default value is *1.*	integer (positive)
audio_on	Time delay (in seconds) from the onset of the trial until the presentation of the audio stimuli (both target speech and masker) starts in the adjustment phase. Default value is 0.5.	float (positive)
button_on	Time delay (in seconds) from the onset of the trial until the activation of the button for quitting the adjustment phase. It should be greater than or equal to the *audio_on* value. Default value is 5.	float
gap	Gap (in seconds) between the phrases in the adjustment phase. Default value is 0.5.	float (positive)
saveaudio	If True, the target speech during the tuning will be saved in a file called *audio_tuning.wav* and stored in the main folder. Default value is *False.*	boolean
numOftesting	Number of target phrases to be tested in each trial. Default value is 1.	integer (positive)
test1audio	Time delay (in seconds) from button press in adjustment phase until the audio starts in the test phase. Default value is 2.	float (positive)

testaudio	Time delay (in seconds) from *ENTER* key press in test phase until the next testing audio starts. Default value is 0.5.	float (positive)
speech_on	Time delay (in seconds) from the onset of the masker until the target speech starts in the test phase. This parameter applies only if a masker is present. Default value is 0.5.	float (positive)

Table 2: Parameters that can be modified in the configuration file.

6 Outputs

The tool returns the raw data collected during the adjustment and test phase (sec. 6.1) and a range of figures that depict the experimental outcomes (sec. 6.2). Outputs are stored in the directory specified in the configuration file under the folders **results** and **plots**, respectively.

6.1 Raw data

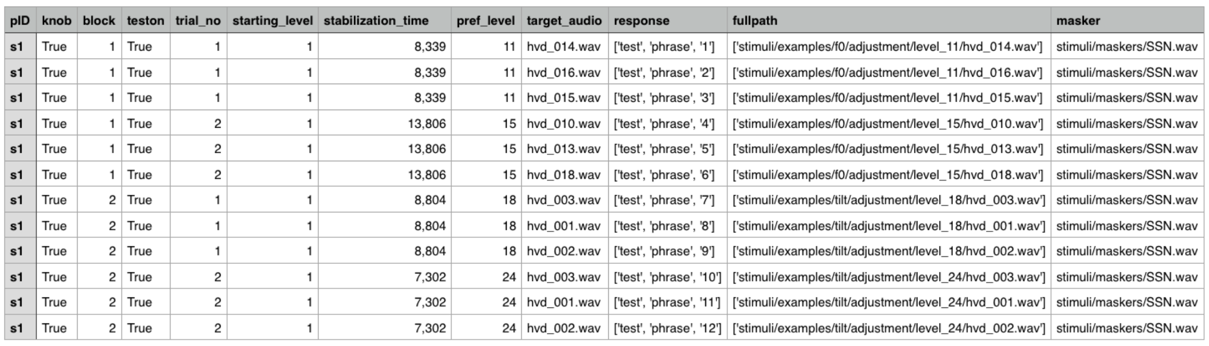
The standard files that the tool returns are a text file with participant's ID as filename (*pID_results.txt*) and a logfile (*logfile.log*). The first file contains information accompanied by a timestamp (Example 2) i.e., the order that the target speech stimuli were presented in the adjustment phase (array of IDs *Adjustment phase target audio IDs*) and the corresponding path (*Target audio path*), the block (*Block*) and trial (*Trial no*) number, the level of the feature with which the trial started (*Starting level*), all the level changes that a listener has performed with the controller (*level*), a prompt that the adjustment phase has finished (*end of adjusting*), the listener's responses (*Response*, if test phase was available) and the corresponding paths of the test audio files (*Audio*). The logfile records events that occur when using the tool and messages accompanied with a timestamp. This file is useful in case an unexpected issue occurs and can be found in the main SPEECHADJUSTER folder. Every time you run the tool, the logfile is overwritten.

Example 2: *Content of the pID_results.txt file.*

```
2019-12-10 16:43:40.967   Adjustment phase target audio IDs : ['hvd_014.wav', 'hvd_034.wav',
'hvd_021.wav', 'hvd_015.wav', 'hvd_017.wav', 'hvd_001.wav']
2019-12-10 16:43:40.968   Block : 1
2019-12-10 16:43:40.970   Target audio path : stimuli/examples/f0/adjustment/
2019-12-10 16:43:41.124   Trial no: 1 Starting level: 5
2019-12-10 16:43:42.864   level = 4
2019-12-10 16:43:43.261   level = 3
2019-12-10 16:43:43.660   level = 2
2019-12-10 16:43:44.059   level = 1
2019-12-10 16:43:45.220   level = 2
2019-12-10 16:43:47.720   level = 1
2019-12-10 16:43:47.797   level = 2
2019-12-10 16:43:48.121   level = 1
2019-12-10 16:43:48.518   level = 2
2019-12-10 16:43:50.116   level = 3
2019-12-10 16:43:52.079   end of adjusting
2019-12-10 16:43:56.993   Audio: stimuli/examples/f0/test/level_3/shd017.wav
2019-12-10 16:44:01.371   Response: Content of phrase with ID shd017.wav
2019-12-10 16:44:05.573   Audio: stimuli/examples/f0/test/level_3/shd030.wav
2019-12-10 16:44:06.394   Response: Content of phrase with ID shd030.wav
2019-12-10 16:44:09.694   Audio: stimuli/examples/f0/test/level_3/shd002.wav
2019-12-10 16:44:10.606   Response: Content of phrase with ID shd002.wav
```

Running the *results.py* script (use the command *python3 results.py -h* for details) for each new participant, a *.csv* file with the participant's ID as filename (e.g., *pID.csv*, Example 3) is created containing one column with the data of each of the following variables (actual variable names can be found in parenthesis): participant's ID (*pID*), knob usage (*knob*), block number (*block*), test phase (*teston*), trial number (*trial_no*), starting level (*starting_level*), time in seconds that the listener needed for finding the optimal value (*stabilization_time*), optimal value/preferred level (*pref_level*), target audio filename presented in the test phase (*target_audio*), listener's written response (*response*), path of the target speech file presented in the test phase (*fullpath*), path of the masker file (*masker*). If variables do not apply to the current experiment the corresponding fields are empty. These data of a new participant are also added in an aggregate *.csv* file which includes the data of all the participants (called *alldata.csv*). If the *pID.csv* file of a participant already exists, the file is not created again but the data of that participant are added to the aggregate *alldata.csv* file and the figures are generated again.

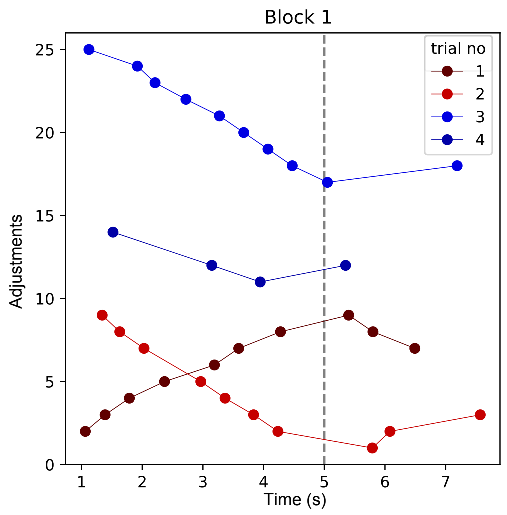
Example 3**:** *Content of the pID.csv file.*



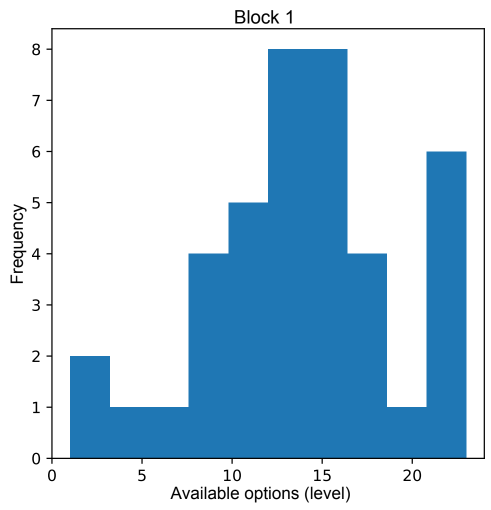
pID	knob	block	teston	trial_no	starting_level	stabilization_time	pref_level	target_audio	response	fullpath	masker
s1	True	1	True	1	1	8,339	11	hvd_014.wav	['test', 'phrase', '1']	['stimuli/examples/f0/adjustment/level_11/hvd_014.wav']	stimuli/maskers/SSN.wav
s1	True	1	True	1	1	8,339	11	hvd_016.wav	['test', 'phrase', '2']	['stimuli/examples/f0/adjustment/level_11/hvd_016.wav']	stimuli/maskers/SSN.wav
s1	True	1	True	1	1	8,339	11	hvd_015.wav	['test', 'phrase', '3']	['stimuli/examples/f0/adjustment/level_11/hvd_015.wav']	stimuli/maskers/SSN.wav
s1	True	1	True	2	1	13,806	15	hvd_010.wav	['test', 'phrase', '4']	['stimuli/examples/f0/adjustment/level_15/hvd_010.wav']	stimuli/maskers/SSN.wav
s1	True	1	True	2	1	13,806	15	hvd_013.wav	['test', 'phrase', '5']	['stimuli/examples/f0/adjustment/level_15/hvd_013.wav']	stimuli/maskers/SSN.wav
s1	True	1	True	2	1	13,806	15	hvd_018.wav	['test', 'phrase', '6']	['stimuli/examples/f0/adjustment/level_15/hvd_018.wav']	stimuli/maskers/SSN.wav
s1	True	2	True	1	1	8,804	18	hvd_003.wav	['test', 'phrase', '7']	['stimuli/examples/tilt/adjustment/level_18/hvd_003.wav']	stimuli/maskers/SSN.wav
s1	True	2	True	1	1	8,804	18	hvd_001.wav	['test', 'phrase', '8']	['stimuli/examples/tilt/adjustment/level_18/hvd_001.wav']	stimuli/maskers/SSN.wav
s1	True	2	True	1	1	8,804	18	hvd_002.wav	['test', 'phrase', '9']	['stimuli/examples/tilt/adjustment/level_18/hvd_002.wav']	stimuli/maskers/SSN.wav
s1	True	2	True	2	1	7,302	24	hvd_003.wav	['test', 'phrase', '10']	['stimuli/examples/tilt/adjustment/level_24/hvd_003.wav']	stimuli/maskers/SSN.wav
s1	True	2	True	2	1	7,302	24	hvd_001.wav	['test', 'phrase', '11']	['stimuli/examples/tilt/adjustment/level_24/hvd_001.wav']	stimuli/maskers/SSN.wav
s1	True	2	True	2	1	7,302	24	hvd_002.wav	['test', 'phrase', '12']	['stimuli/examples/tilt/adjustment/level_24/hvd_002.wav']	stimuli/maskers/SSN.wav

6.2 Figures

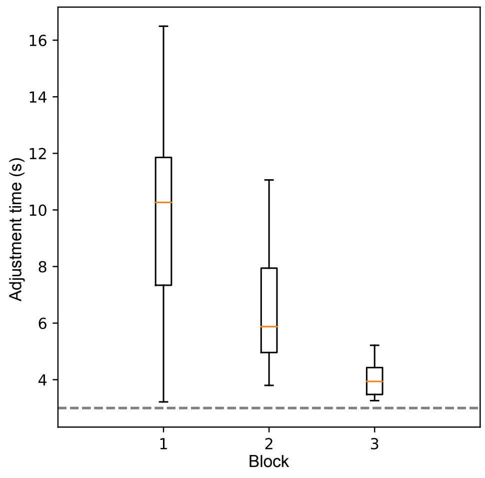
Apart from the text files, the tool produces figures of the experimental outcomes (Fig. 2). Figure 2a is created for each participant while the rest figures for the entire participant cohort. For each block, different plots (i.e., Fig. 2a, 2b and 2e) are generated. Figure 2e is produced only if the experiment consists of a test phase.



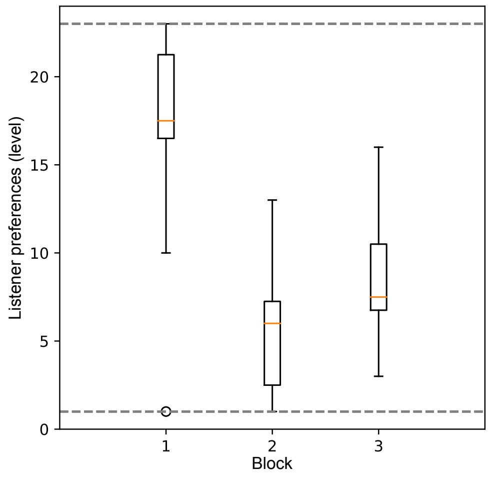
(a) *pID_block1adjustments.pdf*



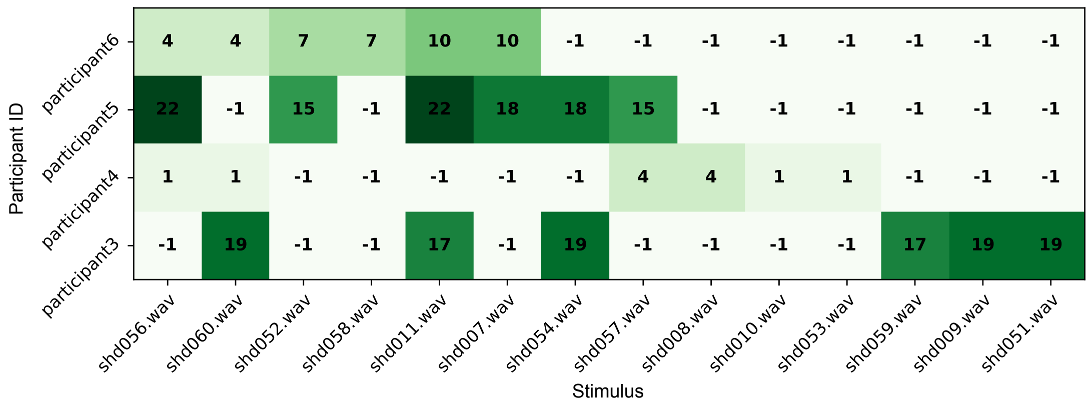
(b) *block1hist.pdf*



(c) *boxplotstabtime.pdf*



(d) *boxplotprefs.pdf*



(e) *block1heatmap.pdf*

Figure 2: Figures that SPEECHADJUSTER produces (subplot labels correspond to the actual names of the files): **(a)** listener's adjustments (dots) across time for each trial (different colours) with dashed line to indicate the button's activation (corresponds to *button_on* parameter), **(b)** histogram for the listeners' preferences of one block, **(c)** box plot of the time that the listeners needed for finding the optimal value in the different blocks with dashed line to indicate the button's activation, **(d)** box plot of listeners' preferences for the different blocks with dashed lines to indicate the extreme levels, **(e)** a heatmap showing each listener's (y-axis) preferences (different colours with corresponding level ID on top) for each target speech file (x-axis) presented in the test phase. If a target speech file was not presented to a listener, the level's value is -1 while if the file has been presented more than once in the same block the level ID on the heatmap corresponds to the latest trial.

License

Contributors

Olympia Simantiraki (*olina.simantiraki@gmail.com*)
Martin Cooke (*m.cooke@ikerbasque.org*)

Olympia Simantiraki was funded by the European Commission under the Marie Sklodowska-Curie European Training Network ENRICH (675324).