AlignTool Documentation (Version 0c2855a, August 2017)

Eva Belke, Verena Keite and Lars Schillingmann

Content

1. What is AlignTool?

AlignTool is an open source tool for the automatic temporal annotation of spoken utterances. It is based on Praat[1] and the automatic speech recognition system MAUS[2] as provided by webMAUS. In most language production studies, the dependent variables are the latency with which speakers produce a spoken response to a stimulus, and/or the temporal structure of a spoken utterance (specifically onset and offset times of words). Measuring these parameters automatically via voice-keys often yields partially incorrect results. However, exact measurements through the visual inspection of the recordings are extremely time-consuming.

AlignTool establishes preliminarily the onset and offset times of words and phonemes in spoken utterances by means of a voice-activity detection heuristic implemented in Praat and a forced alignment of the spoken utterances and their transcriptions in MAUS. It creates Praat TextGrid files from your audio file(s) and uses Praat to identify relevant speech intervals in the audio signal.

AlignTool enters transcriptions of spoken language and audio signals into MAUS in order to force-align them and integrates the alignment data into the TextGrid file(s). By creating TextGrid files as an output, AlignTool allows its users to manually correct the results of its analyses in the TextGrid file, if necessary. This may be helpful for analyses of audio data with a rather poor signal-to-noise ratio. The manual changes can be saved and the corrected data used for analyses.

To date, AlignTool is able to handle German, Dutch and British English speech input. An extension to other languages is possible, as the only language-sensitive processing step of the tool relies on MAUS, which supports other languages, such as Italian, Spanish, Russian, as well as other variants of English and German (Australian and American English, Swiss German; for a full list, see the "service options" drop down menu at the WebMAUS site: https://clarin.phonetik.uni-muenchen.de/BASWebServices/#!/services/WebMAUSBasic). AlignTool can analyse recordings of single trials and recordings of whole experiment blocks.

A basic concept of AlignTool is a parallel representation of the alignment data: It is stored in one or multiple TextGrid file(s) and in an Excel control file. This gives you the opportunity to easily edit and change the alignment and record all changes made in an easily exportable Excel format.

This documentation gives an overview of the functions implemented in AlignTool and step by step instructions for Windows users. We will start with the software requirements and setup of AlignTool (Chapter 2); followed by an overview of the components of AlignTool, i.e. software and files you need for the analysis with AlignTool (Chapter 3). Chapter 4 is a tutorial of the alignment process: We will analyse an example audio file. Chapter 5 gives you instructions on how to analyse your own data.

---

[1] cf. Boersma, Paul & Weenink, David (2016). *Praat: Doing phonetics by computer* [Computer program]. Version 6.0.17, retrieved 21 April 2016 from http://www.praat.org/

[2] MAUS is short for "The Munich Automatic Segmentation System" (cf. Schiel, F. (1999). Automatic phonetic transcription of non-prompted speech. *International Congress of Phonetic Sciences 14,* 607-610). Go to http://www.bas.uni-muenchen.de/Bas/BasMAUS.html for further information). WebMAUS is a web service for analyses using MAUS. For simplicity, we will use the term MAUS in the following when referring to MAUS or WebMAUS.

2. Getting started

AlignTool runs under Linux Ubuntu natively. In order to use it with a Microsoft Windows operating system, it is necessary to simulate a virtual Linux environment. For this purpose, VMware Workstation Player is required. AlignTool runs as a virtual machine in the Workstation Player.

Please note that in order to run VMware Workstation Player, a 64-bit Windows operating system is required and you need to run it as the system administrator. Depending on the configuration of your computer, you may need to adjust BIOS settings to allow for a virtual machine to operate on your computer. Please consult the help function of the VMware Workstation player if you have problems running a virtual machine.

The next section lists everything you need to run AlignTool (software and otherwise) and where to download it (in case you don't have it already installed on your computer). AlignTool comes in a compressed folder. We will have a look at that folder in Section 2.2. In Section 2.3. we describe how to set up the Workstation Player for the first time.

2.1. Requirements

In order to use AlignTool you will need the following:

AlignTool folder

The archive you have downloaded with this manual, called "aligntoolvm.7z", contains this folder.

7-Zip

is required to decompress the AlignTool folder.

- Go to http://www.7-zip.org/ and download the most recent version. 7-Zip is open source software.
- Install 7-Zip.
- Unpack aligntoolvm.7z; you can copy the AlignTool folder to any location on your hard disk.

VMware Workstation Player

- Go to https://www.vmware.com/go/downloadplayer and download the most recent version of Workstation Player for windows hosts (we use version 12.5.2). Please feel free to update your version when Workstation Player offers it or by clicking on "Check for Updates". VMware offers a free version, which is available for non-commercial, personal and home use. It requires a 64-bit operating system.
- Install VMware Workstation Player. In order to run a virtual machine, it may be necessary to adjust BIOS settings. For trouble shooting, please refer to the help function of VMware Workstation Player.
- When starting AlignTool, VMware will ask whether you moved or copied the virtual machine. Choose "I copied it".

Praat

- Go to http://www.fon.hum.uva.nl/praat/download_win.html and download the most recent version of Praat. Praat is free under the GNU General Public License.
- Install Praat.

Internet connection

AlignTool calls up the MAUS Web Services and the BAS grapheme-to-phoneme conversion (g2p) service (https://www.clarin-d.de/en/g2p-en).

Microsoft Excel

Settings for the AlignTool commands and alignment data are saved in an Excel file (see Section 3.2.).

2.2. AlignTool folder

Let's have a look at the AlignTool folder, just unpacked. In this folder, you find the AlignTool Virtual Machine and the working directory:

aligntoolvm.vmx

This is the AlignTool Virtual Machine ("vm" is short for Virtual Machine.). We are going to open it in the next step (see Section 2.3.).

workdir folder

The "workdir" folder is your working directory, i.e. the interface between your computer and the Virtual Machine. AlignTool has access to the files saved here and will save the files here that it creates.

In the working directory, you find the following:

wav folder

The "wav" folder is used for the audio files that you want to analyse. In it you find an example audio file, called "example.wav" (see Section 3.1. for a detailed description). We are going to align this file in Chapter 4.

example_workbook.tg folder

In the "example_workbook.tg" folder you find the corresponding TextGrid file to the example audio file. AlignTool creates a folder named after the used workbook plus ".tg" (which is short for TextGrid) and saves the TextGrid file(s) there.

example_workbook.xlsx

This is the Excel control file we use in the example analysis (see Chapter 4). We are going to have a detailed look at it in Section 3.2 and will work through examples of it in sections 4 and 5.

example_transcriptions.xlsx

In this file, you find transcriptions of the example audio file. We will use it for the alignment in Chapter 4.

example_reference_beep.wav

This is a recording of a beep marking the trial onset in the recording of the experimental session used in our example. We will use it to segment the example audio file into individual trials based on the reference beep (see Chapter 4).

simple-test folder

The simple-test folder includes another example for use with AlignTool. You can ignore it for the time being.

2.3. Setup of VMware Workstation Player

In the first setup, you need to open the AlignTool Virtual Machine and set the working directory to be the "workdir" folder in the AlignTool folder.

Step 1. Launch Workstation Player

- To start up the VMware Player, just double-click it.

Step 2. Open AlignTool

- Click on "Open a Virtual Machine":

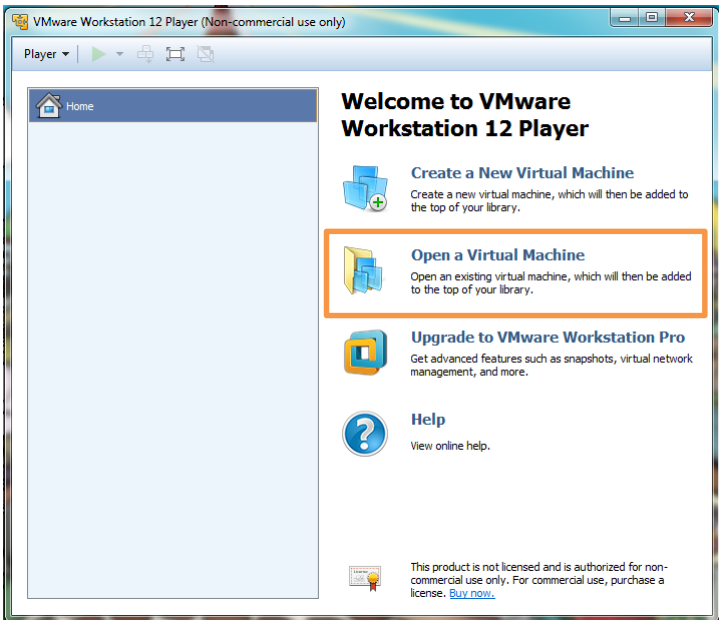


Figure 1: Open a Virtual Machine

- Browse to the AlignTool folder and select "aligntoolvm.vmx".

Step 3. Set the working directory

- Click on “Edit virtual machine settings”:

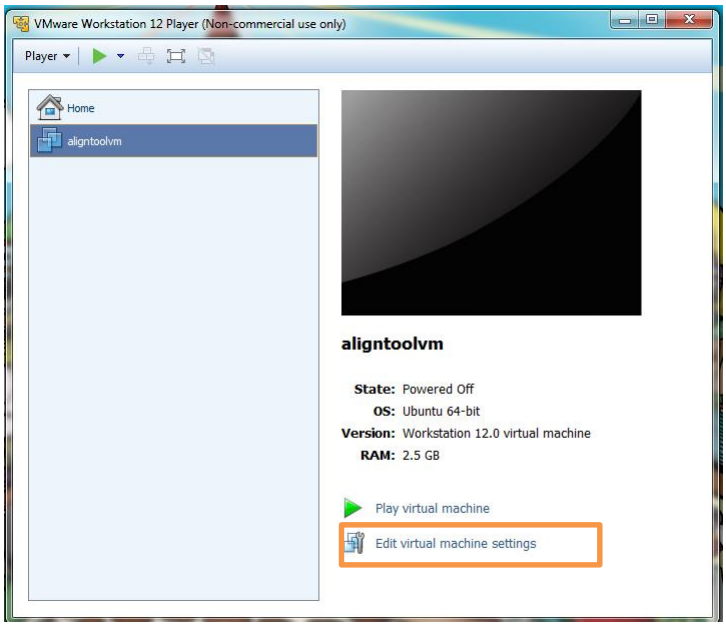


Figure 2: Edit virtual machine settings

- Click on the second tab (“Options”):

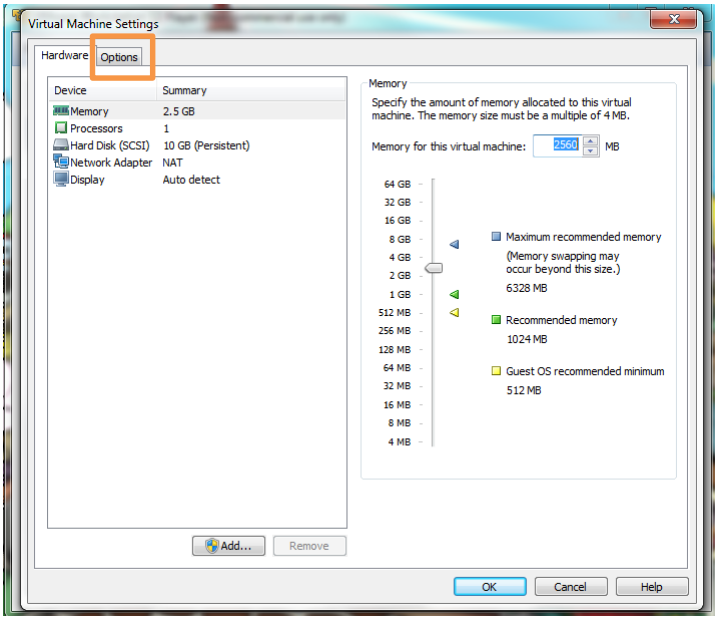


Figure 3: Options-tab

- Click on “Shared Folders”:

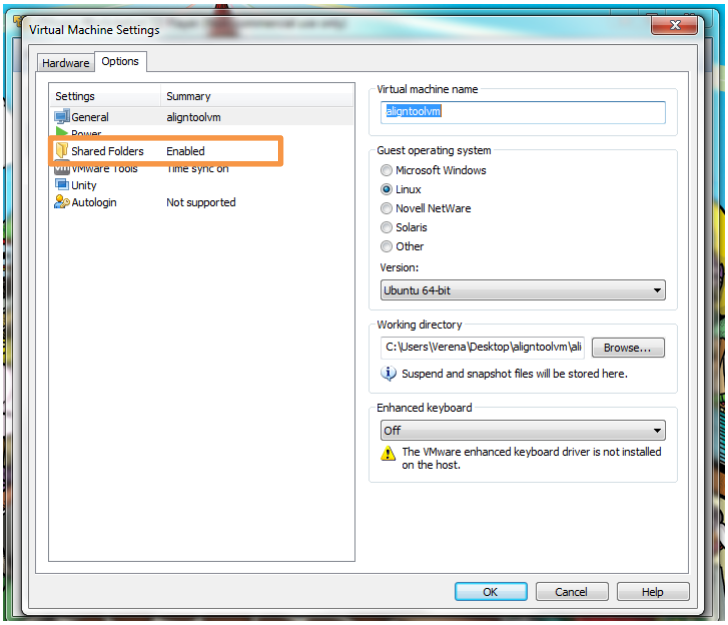


Figure 4: Shared Folders (1)

- Allow folder sharing by checking “always enabled”.
- Click on “workdir”:

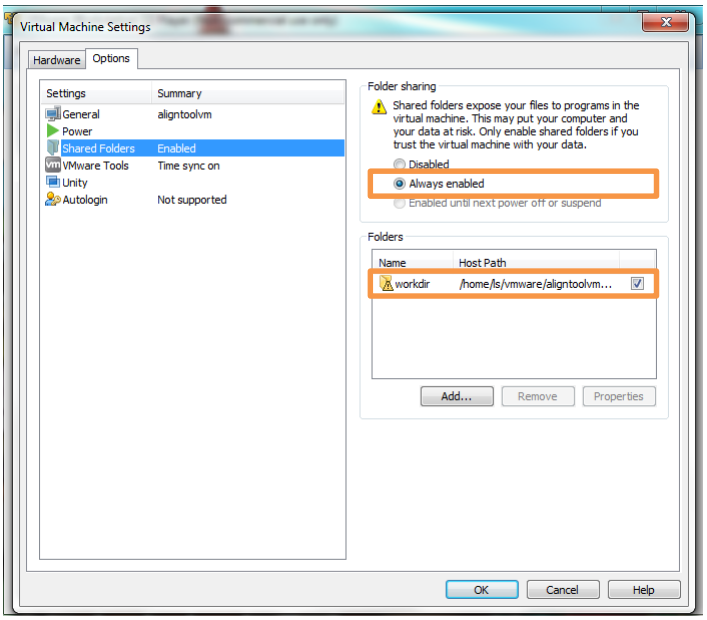


Figure 5: Shared Folders (2)

- Browse to the AlignTool folder and select the “workdir” folder.

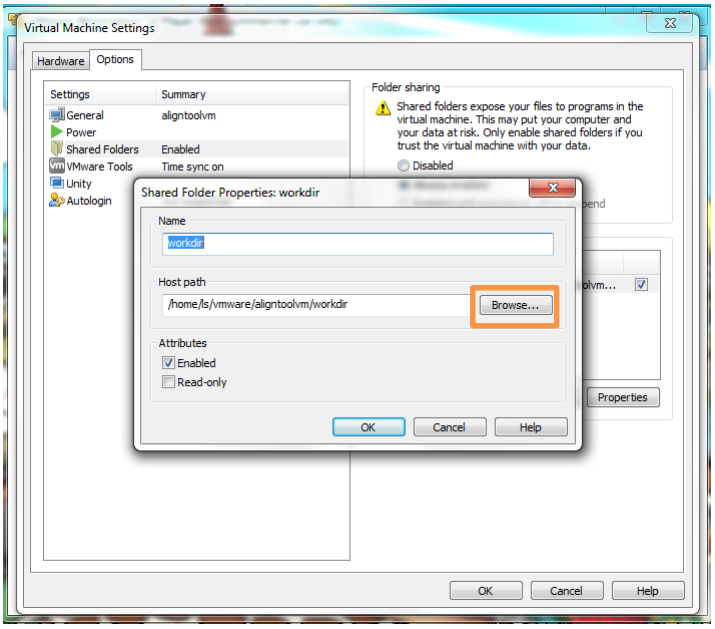


Figure 6: Browse to working directory

- Confirm settings by clicking “OK”.

Step 4. Have a first look at the AlignTool GUI

Click on “Play virtual machine”:

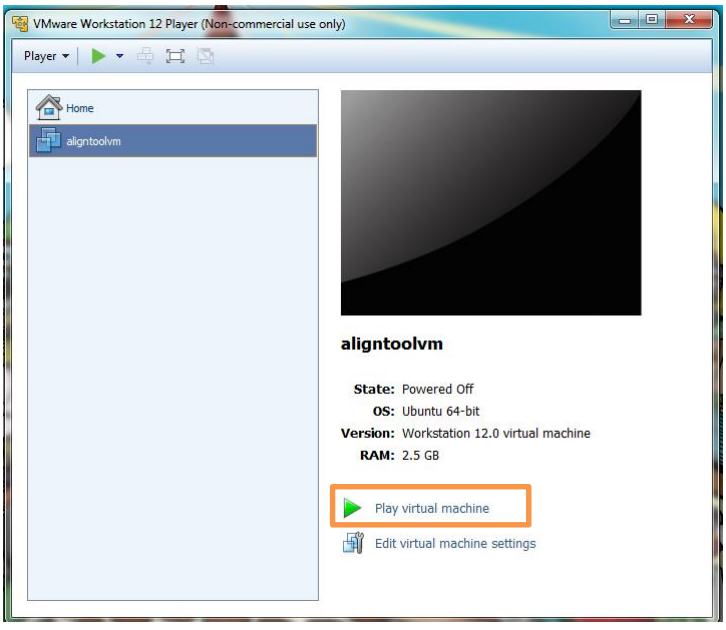


Figure 7: Play virtual machine

- Double click on “aligntool”.

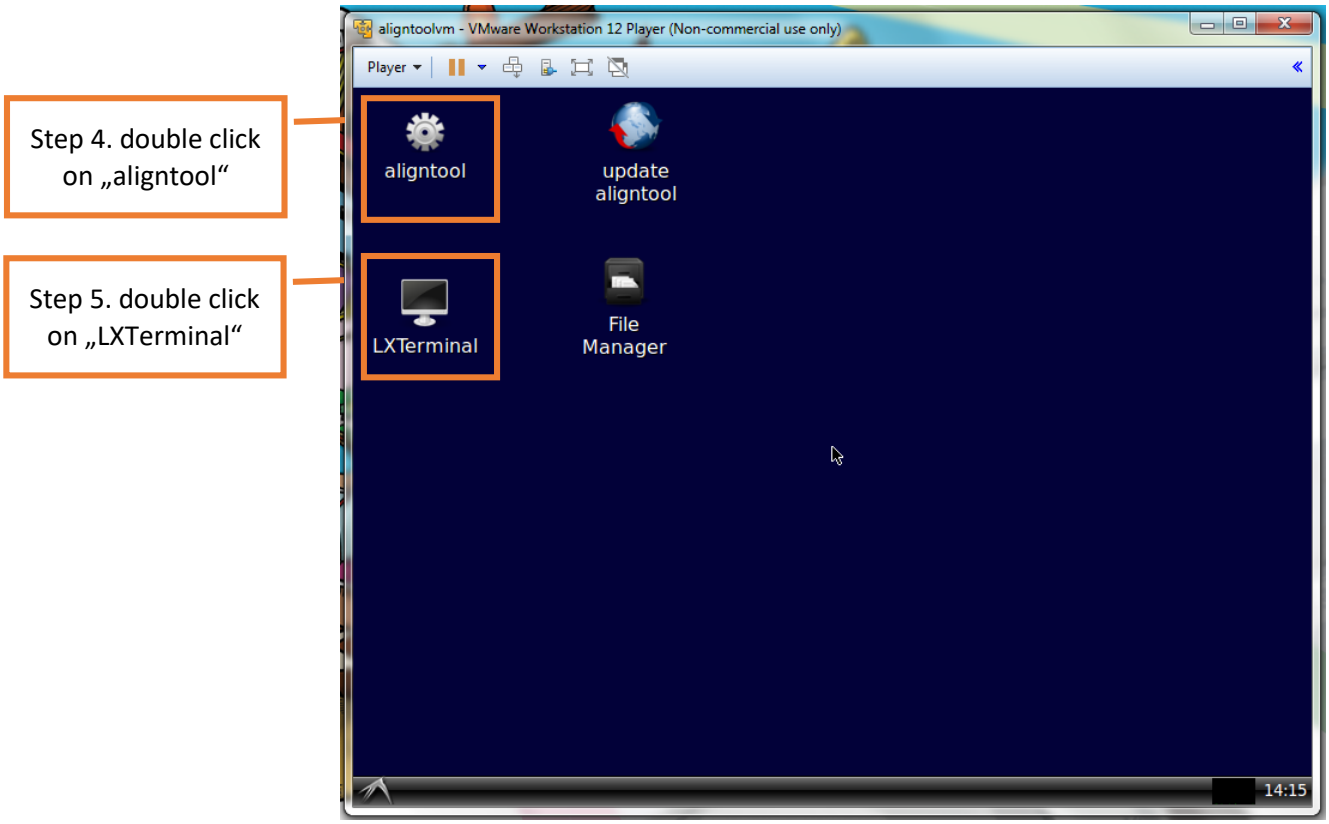


Figure 8: AlignTool Virtual Machine

- Now, you see the AlignTool GUI:

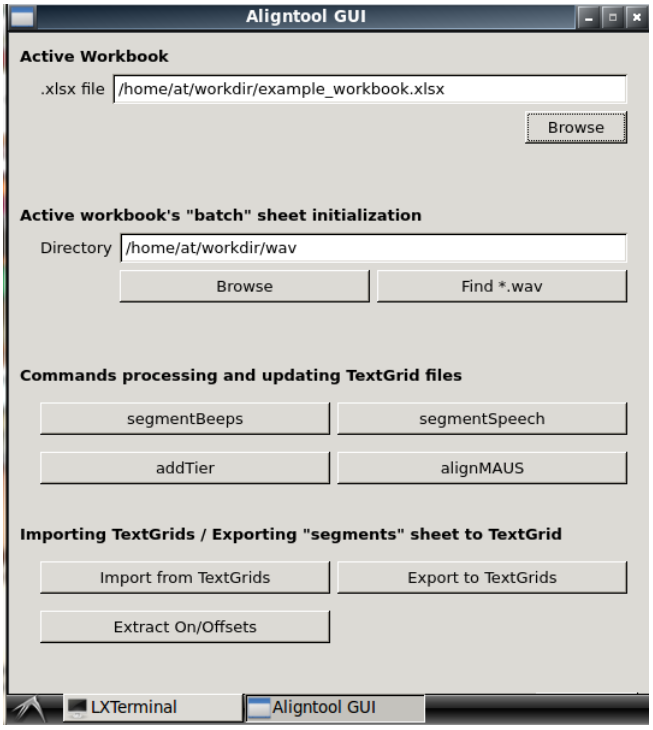


Figure 9: AlignTool GUI

- Close AlignTool for the time being. We will return to the AlignTool GUI in Section 3.3.

Step 5. Open LXTerminal

- Double click on "LXTerminal" (see Fig. 8). This opens a window with a command line interface.

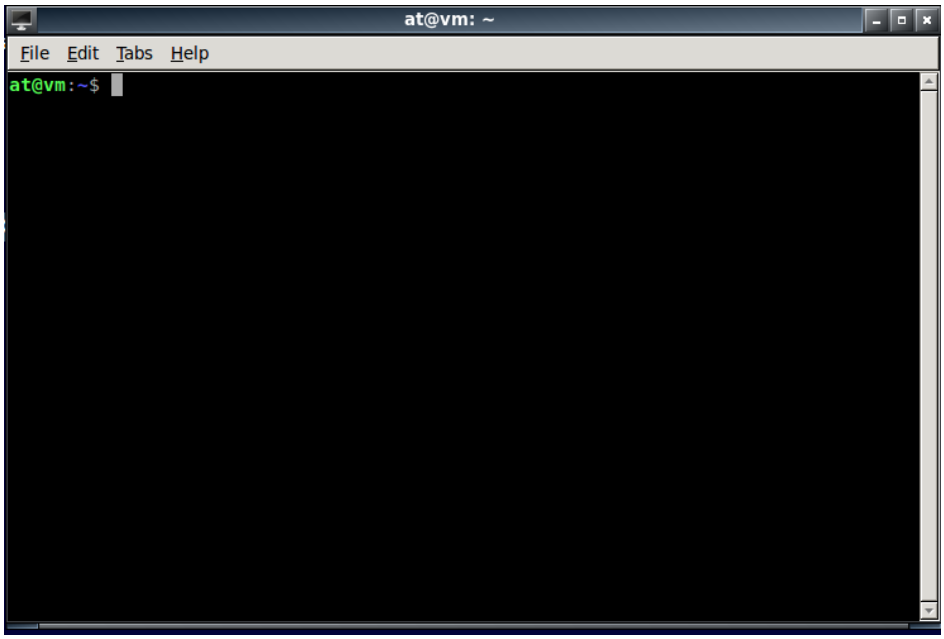


Figure 10: LXTerminal

- The command line is a means to interact directly with AlignTool, should users prefer to do so. Users working with the GUI will not need it for using AlignTool. This manual is geared towards users working with the GUI, but Appendix A lists all commands and parameters users preferring working from the command line need to use.

3. AlignTool Components

Chapter 3 is an overview of the different components (i.e. software and files) that we will be using during the analysis with AlignTool: TextGrid files in Praat (Section 3.1.), the workbook in Excel (Section 3.2.) and the AlignTool GUI (Section 3.3.).

We will need to hop back and forth between these different kinds of files and software in the alignment process. Therefore, this section is intended to familiarize you with the different components before the example alignment (in Chapter 4) and the general instruction (in Chapter 5).

In this chapter, we present the relevant processing steps in as much detail as possible for you to get a clear grasp of how AlignTool works. You will find that as your experience with the software grows, you may want to merge some of the processing steps to speed things up. However, before you do so, you should familiarize yourself thoroughly with each processing step – its required input, its function and the output it generates.

3.1. Audio and TextGrid files

The best way to see what AlignTool actually does is to look at an output file. In this section, we will explain the example audio file and its corresponding TextGrid file.

The example audio file is a recording of one participant in a word naming and picture naming experiment in English. The subject was a non-native speaker of British English. A total of 150 trials was recorded. The recording included the speech on one channel and trial onset beeps marking the beginning of individual trials on the other channel. The experiment was run in two blocks: In the first block, the participant saw four words in each trial and read them aloud (50 trials). The second block consisted of 100 trials and was a picture naming task (with one picture per trial).

As we will detail in Chapter 5, some experiment setups will allow the user to record individual trials instead of full experimental blocks. Such trial-by-trial recordings are a special case of the beep-segmented recordings of full experimental blocks shown in the example file. We will describe the analysis of trial-by-trial recordings in Chapter 5. For the time being, we shall focus on the example recording.

In the following we will briefly describe how to open a TextGrid file and a corresponding wav file in Praat (please refer to http://www.fon.hum.uva.nl/praat/ for extensive manuals and tutorials):

1. Open Praat by double clicking on it. Now, two windows open.
2. Click on “Open” in the “Praat Objects” window.
3. Choose “Read from file” and browse to the example audio file in the “wav” folder.
4. Click on “Open” again, choose “Read from file” and browse to the example TextGrid file in the tg folder.
5. Select both files.
6. Click on “View & Edit”.

Please note that loaded TextGrid files are not updated in Praat. During the analysis with AlignTool, the TextGrid file(s) will be edited several times. In order to see the latest version of a TextGrid file, you need to re-open it in Praat (by clicking on “Open”, choosing “Read from file”, and selecting the TextGrid file as described above).

Let’s inspect the audio and TextGrid file just opened. The audio signal is shown in the top half of the screen, with the speech signal in the left and the beep signal in the right channel. The five lines (or tiers, in Praat) at the bottom of the screen display the content of the TextGrid file created by AlignTool during the alignment process. Before we look at each of these tiers, please note the following characteristics of the audio file: First, each trial starts with a 200 ms picture onset beep recorded on the right channel (Channel 2), with the beginning of the beep corresponding to the time the picture or word stimulus was presented to the participant. There is a second, shorter beep, marking when the voice key triggered (marked with little arrows in Fig. 11). Second, the signal-to-noise ratio of the example audio file is average (the speech signal is not very loud; see left channel, Channel 1), but as you will see, AlignTool is able to handle it.

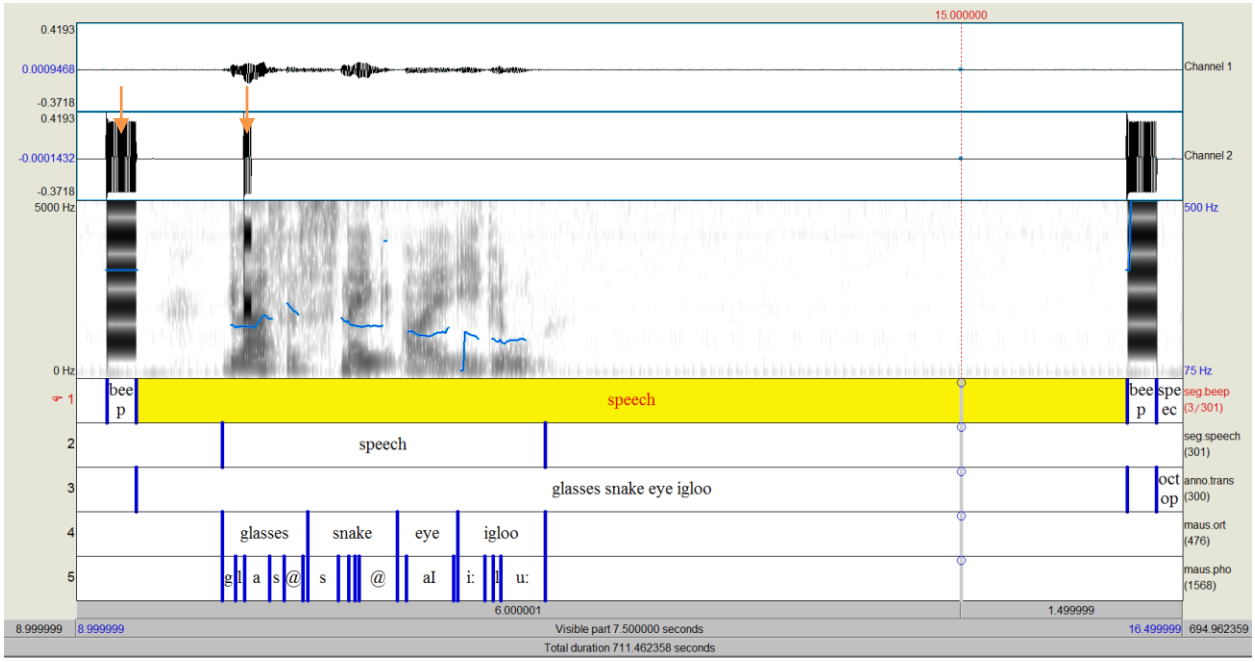


Figure 11: Example audio and TextGrid file

As you can see the TextGrid file consists of five tiers displayed at the bottom of Figure 11:

1. **seg.beep**

   In the first tier the audio file is segmented into "speech" intervals and "beep" intervals, i.e. picture onset beeps. Since every trial started with a picture onset beep, this corresponds to a segmentation based on trials. "speech" denotes the part of the trial following the beep.

2. **seg.speech**

   In the next tier the actual speech intervals are identified, i.e., the segments labeled "speech" here correspond to the parts of the trials when the participant actually did speak.

3. **anno.trans**

   The *anno.trans* tier contains the "speech" intervals identified in *seg.beep*, but they are labeled with transcripts of the trials.

4. **maus.ort**

   In the *maus.ort* tier the audio signal is aligned to the transcripts of the trial at the word level.

5. **maus.pho**

   The last tier shows you the alignment at the phoneme level.

We will describe how those tiers are created in the example analysis in Chapter 4. For now, let's have a look at the example workbook.

3.2. Workbook

The workbook has three functions: It is used to specify settings for the commands in the AlignTool GUI (see Section 3.3. for an overview of the commands), to save alignment data and to extract onsets and offsets. These three functions correspond to the three sheets in the workbook.

1. “batch” Sheet

In column A and B of the “batch” sheet, the file path to the wav file(s) and their corresponding TextGrid file(s) are listed. In case you are analysing more than one file, there will be multiple lines.

In columns C to G, the settings for the AlignTool commands are stated. We chose to use an Excel file for this purpose, so you do not need to use the command line. An additional advantage is that you can easily copy the command settings when analysing multiple wav files with the same settings.

Please note that the commands in line 1 need to be spelt exactly as in the example workbook. Otherwise, AlignTool will not be able to read them. The commands as of line 2 are edited by the user when working with AlignTool. In Chapter 4, we take you through the example workbook; in Chapter 5, you can try working with the workbook using your own files.

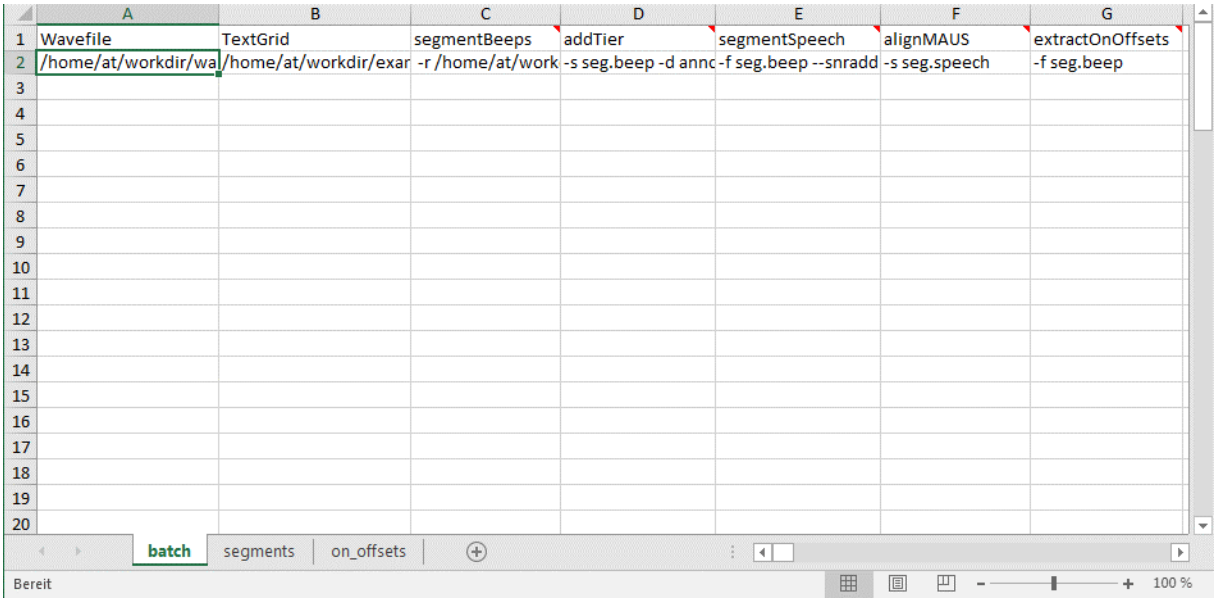


Figure 12: Batch sheet in example workbook

2. “segments” Sheet

On the second sheet, you can see the same data you have just inspected in Praat (Figure 11), now listed in a table: Every segment of every tier that AlignTool has created is stated here. It starts with the first tier *seg.beep* listing the length of the full tier, the start time, end time and text of every segment in the tier (all times are given in seconds). Then, the next tier follows and so on.

The “segments” sheet is mostly needed by AlignTool for its internal computations. But it is also the gateway for your transcripts to the TextGrid file(s) (see Chapters 4 and 5 for details on that).

	A	B	C	D	E	F
1	TextGrid	TierName	StartTime	EndTime	Text	
2	/home/at/workdir/example_workbook.tg	<range>	0	711.4623583		
3	/home/at/workdir/example_workbook.tg	seg.beep	9.205396825	9.405102041	beep	
4	/home/at/workdir/example_workbook.tg	seg.beep	9.405102041	16.12452985	speech	
5	/home/at/workdir/example_workbook.tg	seg.beep	16.12452985	16.32423507	beep	
6	/home/at/workdir/example_workbook.tg	seg.beep	16.32423507	23.06366191	speech	
7	/home/at/workdir/example_workbook.tg	seg.beep	23.06366191	23.26336713	beep	
8	/home/at/workdir/example_workbook.tg	seg.beep	23.26336713	29.98279373	speech	
9	/home/at/workdir/example_workbook.tg	seg.beep	29.98279373	30.18249894	beep	
10	/home/at/workdir/example_workbook.tg	seg.beep	30.18249894	36.90292642	speech	
11	/home/at/workdir/example_workbook.tg	seg.beep	36.90292642	37.10263163	beep	
12	/home/at/workdir/example_workbook.tg	seg.beep	37.10263163	43.82205951	speech	
13	/home/at/workdir/example_workbook.tg	seg.beep	43.82205951	44.02176472	beep	
14	/home/at/workdir/example_workbook.tg	seg.beep	44.02176472	50.74219182	speech	
15	/home/at/workdir/example_workbook.tg	seg.beep	50.74219182	50.94189703	beep	
16	/home/at/workdir/example_workbook.tg	seg.beep	50.94189703	57.66132297	speech	
17	/home/at/workdir/example_workbook.tg	seg.beep	57.66132297	57.86102818	beep	
18	/home/at/workdir/example_workbook.tg	seg.beep	57.86102818	64.58045551	speech	
19	/home/at/workdir/example_workbook.tg	seg.beep	64.58045551	64.78016072	beep	
20	/home/at/workdir/example_workbook.tg	seg.beep	64.78016072	71.50058827	speech	

batch | segments | on_offsets

Figure 13: Segments sheet in example workbook

3. "on_offsets" Sheet

The third sheet lists the onset and offset times of each word in an utterance. Additionally, the beginning of a pre-segmentation interval ("PresegBegin") is provided, if applicable: In case the audio data was pre-segmented using the beep detection function (*segmentBeeps*), "PresegBegin" corresponds to the offset of the beep which is calculated based on the beep's onset plus the exact length of the reference beep.

	A	B	C	D	E	F	G	H	I	J	K
1	TextGrid	Error	ErrorMsg	Transcriptior	AlignOnset	AlignOffset	WordCount	NumWords	OnsetPhone	OffsetPhone	PresegBegin
2	/home/at/w	0		glasses	9.98617914	10.5661791	1	4	g	s	9.40510204
3	/home/at/w	0		snake	10.5661791	11.1761791	2	4	s	@	9.40510204
4	/home/at/w	0		eye	11.1761791	11.5861791	3	4	aI	@	9.40510204
5	/home/at/w	0		igloo	11.5861791	12.1761791	4	4	i:	u:	9.40510204
6	/home/at/w	0		octopus	16.9561791	17.5761791	1	4	@	s	16.3242351
7	/home/at/w	0		squirrel	17.5761791	18.4561791	2	4	s	l	16.3242351
8	/home/at/w	0		leaf	18.4561791	18.5811678	3	4	l	f	16.3242351
9	/home/at/w	0		flag	18.5811678	19.1861791	4	4	f	k	16.3242351
10	/home/at/w	0		dragon	23.9361791	24.4361791	1	4	d	n	23.2633671
11	/home/at/w	0		frog	24.4361791	25.0561791	2	4	f	k	23.2633671
12	/home/at/w	0		crab	25.0561791	25.4461791	3	4	k	p	23.2633671
13	/home/at/w	0		room	25.5161791	26.0961791	4	4	r	m	23.2633671
14	/home/at/w	0		plug	30.9261791	31.3861791	1	4	p	k	30.1824989
15	/home/at/w	0		ghost	31.3861791	31.8161791	2	4	g	t	30.1824989
16	/home/at/w	0		watch	31.9161791	32.4361791	3	4	v	tS	30.1824989
17	/home/at/w	0		genie	32.4961791	33.0161791	4	4	Z	i:	30.1824989
18	/home/at/w	0		penguin	37.7261791	38.1861791	1	4	p	n	37.1026316
19	/home/at/w	0		saddle	38.1861791	38.8061791	2	4	z	@	37.1026316
20	/home/at/w	0		elephant	38.8061791	39.4761791	3	4	e	t	37.1026316

batch | segments | on_offsets

Figure 14: On_offsets sheet in example workbook

3.3. AlignTool GUI

When you start AlignTool, the Align Tool GUI appears (see Fig. 15). This section is intended to give you an overview of the AlignTool GUI. We are going to use its functions in Chapter 4. At the top of the GUI you can see two fields in which you need to specify file paths (marked blue in Fig. 15). The first

one is for the workbook and the second one needs to point to the "wav" folder in your working directory.

In the middle (marked orange), you find the section "Commands processing and updating TextGrid files". Those are the commands which create the five tiers in the TextGrid file(s) (see Section 3.1.). The settings for these commands are specified in the batch sheet of the workbook (see Section 3.2.).

At the bottom, you can see commands to import from TextGrid file(s) to the workbook and export from the workbook to the TextGrid file(s). As we have seen in Section 3.2., the basic idea of AlignTool is a parallel representation of the alignment data: it is stored in the TextGrid file(s) and the workbook. The import/export commands are used to exchange data between the different files and make sure that they include identical data. Specifically, they give you the opportunity to manually change segments and alignments in the TextGrid file or the Excel file and transfer these changes between file formats. Finally, you find the command *Extract On/Offsets* in the green section, which extracts the onsets and offsets from the "segments" sheet of the workbook and writes it to the "on_offsets" sheet of the workbook.

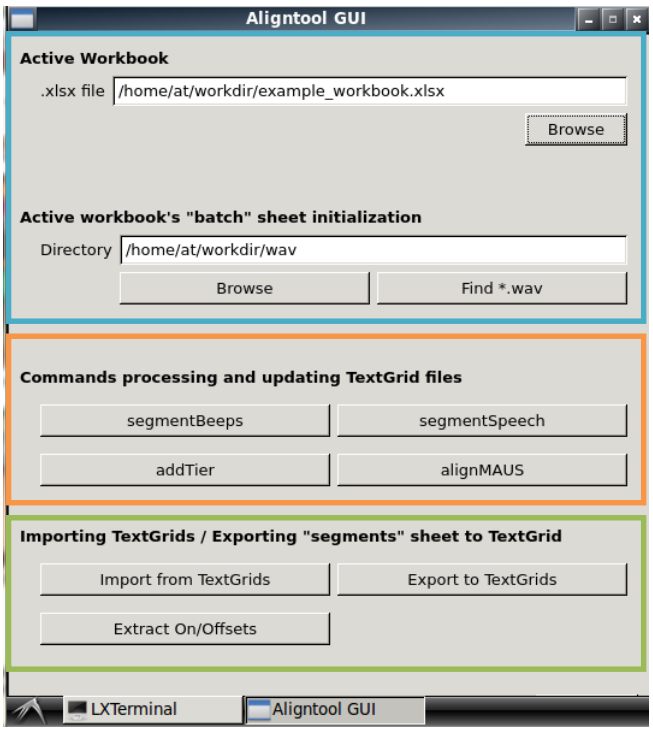


Figure 15: AlignTool GUI

When AlignTool is launched, an LXTerminal window is opened to display an on-line log of the commands AlignTool executes. Note that this log is also written to a file for later use. Note also that the log window of the LXTerminal does not feature a command line. If you want to interact with AlignTool via the command line, you need to open a new LXTerminal window from the desktop of the virtual machine, as described above (see Step 5 in Section 2.3.).

You have now inspected everything you need to analyse audio files with AlignTool. Let's get started with the example alignment.

4. Hands on: Analyzing Example Audio Data

This chapter gives you a tour around AlignTool by analysing the example audio file. Step by step, we will create the example TextGrid file (see Section 3.1.).

As a preview, here is how we will interact with AlignTool and what we will have it do: First, we will tell AlignTool with what to work by specifying the workbook (1) and the audio file (2). After segmenting the audio file by means of the beeps marking the trial onsets (3), AlignTool can use Praat to identify the exact stretches of speech based on this (4). We will fill in transcriptions of the trials (5), and apply MAUS for the forced alignment of the relevant audio segments and the transcriptions (6). The last step is to extract the onsets and offsets (7).

Step 0. Start AlignTool

0.1. Start VMware Player.

0.2. Click on "Open a Virtual Machine", browse to the AlignTool folder, and select "aligntoolvm.vmx" and click on "Play virtual machine" in order to launch the Virtual Machine.

OR

0.2 Select "aligntoolvm" and double click on it or select "play virtual machine" in order to launch the virtual machine.

Step 1. Specify Workbook

As explained in Section 3.2., AlignTool needs a workbook, in which settings for the commands are specified and alignment data will be saved. In the following, we will use the example workbook in the working directory in the AlignTool folder.

1.1. Open example_workbook.xlsx in Excel. The first sheet called "batch" includes sample settings for the AlignTool commands. For now, we will use these predefined settings (see Chapter 5 for different options).

1.2. You can delete the sheets called "segments" and "on_offsets". These will be established again shortly when doing your own analysis.

1.3. Save and close the workbook. (Unfortunately, in Windows, AlignTool cannot access a workbook while it is open in Excel.)

1.4. Go back to the AlignTool GUI: Set the file path and name of the workbook using the "Browse" button under "Active Workbook" (see Fig. 16).

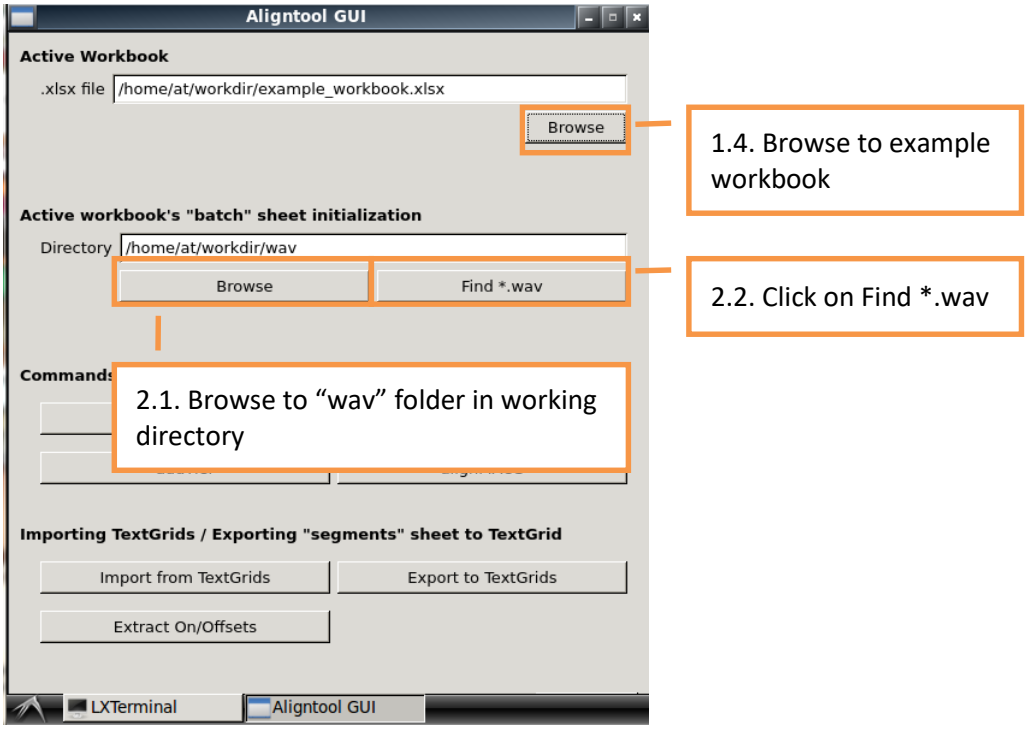


Figure 16: Step 1 & 2 in AlignTool GUI

Step 2. Specify Audio file(s)

The second step is to tell AlignTool what audio file(s) to align. We will analyse an example audio file ("example.wav") with beeps indicating the onsets of individual trials. This file is located in the "wav" folder in the working directory.

2.1. Set the file path to the "wav" folder in the working directory by clicking on "Browse" under "Active workbook's "batch" sheet initialization" (see Fig. 16).

2.2. Click on "Find *.wav" in the AlignTool GUI (see Fig. 16).

2.3. AlignTool tells you when it is done in the Terminal: Click on the "LXTerminal" window at the bottom to see a log of the commands just executed. It should state "finished WavImporter".

```
-=INFO=- [16:32:04.684] finished WavImporter {'xlsxfile': '/home/at/workdir/exam
ple_workbook.xlsx', 'directory': '/home/at/workdir/wav'}
```

(Please note that you can always check in the Terminal whether AlignTool has executed a command. The last log entry should always include the word "finished".)

2.4. Open the workbook and check whether the example audio file ("example.wav") has been found and added in column A. It should state in line 2:

/home/at/workdir/wav/example.wav

2.5. Also check whether the name of the TextGrid file in column B is an exact mirror image of the wav file: It should be called “example.TextGrid”.

```
/home/at/workdir/example_workbook.tg/example.TextGrid
```

Step 3. Segment Beeps

The third step is to segment the audio file based on the trial-onset beeps (since we are analysing an audio file with beeps separating trials).

3.1. Settings for the command “segmentBeeps” are stated in column C in the workbook. We use the “-r” parameter to specify the name and file path of a reference beep, i.e. a recording of the beep, used in the example. This file is located in the working directory.

```
' -r /home/at/workdir/example_reference_beep.wav
```

If the command is not stated in column C, type it in. Please do not copy and paste the command line from this PDF, as the copied command may include control characters (e.g., \n) that are not shown in Excel but will cause problems for AlignTool.

3.2. Save and close the workbook.

3.3. Press “segmentBeeps” in the AlignTool GUI. When complete, the online log in the Terminal states “finished BatchRunner {batchcmd’: ‘segmentBeeps’ …}”.

Now, a TextGrid file by the name “example.TextGrid” has been created. This file is located in the “workbook_example.tg” folder in your working directory.

3.4. Open the “aligntool-log” file in the “workdir” folder. It includes a log of all the commands AlignTool has just executed. Search for “Error” to establish whether any processing step has caused AlignTool to report an error.

3.5. Open “example.TextGrid” and “example.wav” in Praat to inspect them (see Section 3.1.). You should now see the wav file and one tier called *seg.beep*. This tier includes stretches labelled “beep” where a beep was found and “speech” where speech may later be allocated (see Fig. 17).

3.6. Next, you need to synchronize the TextGrid file with the workbook, so click on “Import from TextGrids” in the AlignTool GUI. By experience, this processing step is very robust, so there is no need to check the log file for errors.

3.7. Open the workbook: It should now include a new sheet called “segments”. In it, you see the boundary times of the intervals of the audio file you have previously inspected in Praat.

The first line shows the full length of the file, next, each segment is listed, stating the beginning and the end as well as the content (“beep” or “speech”). If the beeps were extracted correctly, you should find exactly as many “speech” segments as there were trials in the experiment, i.e. 150 “speech” segments in our example.

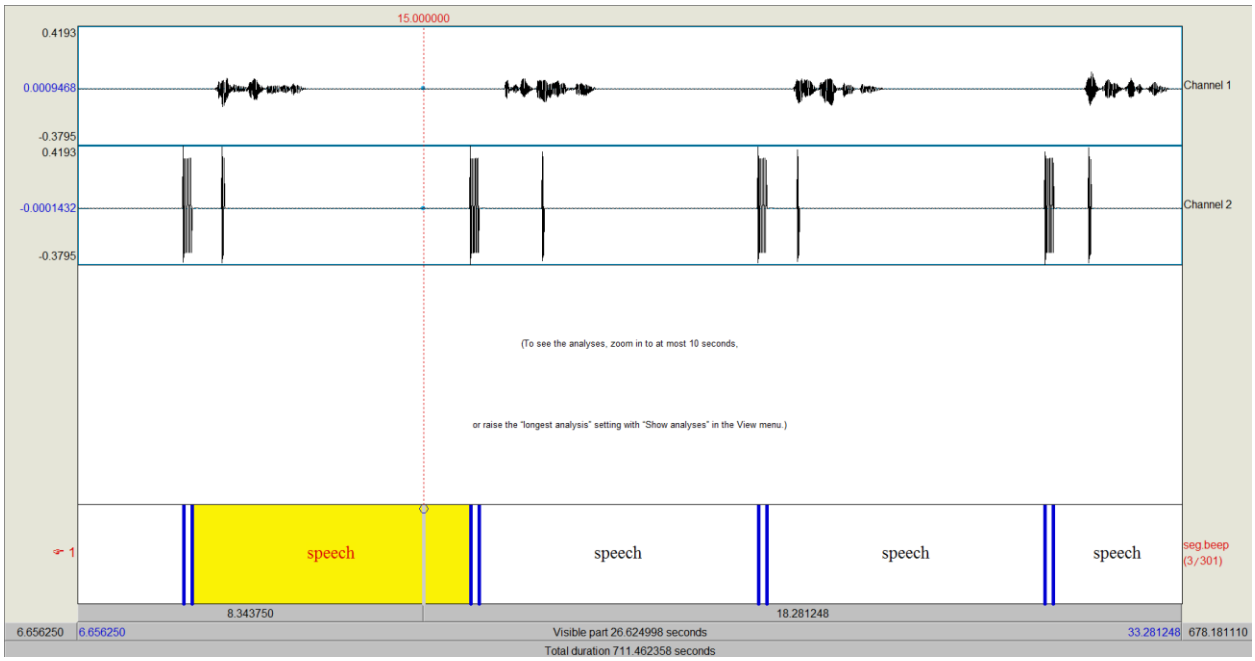


Figure 17: Example TextGrid file after Step 3

Step 4. Segment Speech

Step 4 will identify the exact stretches of speech within each section identified in the last step.

4.1. You find the settings for the command "segmentSpeech" in column E in the workbook. We use:

```
'-f seg.beep --snradd "2" --speechthresh 0.2
```

This reads: Filter speech intervals based on the existing speech interval tier *seg.beep*, add 2 db to the silence threshold, and set the relation of the average interval intensity to the valid speech interval intensity to 0.2.

If the command is not stated in column E, type it in. Please do not copy and paste the command line from this PDF, as the copied command may include control characters (e.g., \n) that are not shown in Excel but will cause problems for AlignTool.

4.2. Save and close the workbook.

4.3. Run *segmentSpeech* in the AlignTool GUI. This processing step may take a little while. When complete, the online log in the Terminal states "finished BatchRunner {batchcmd': 'segmentSpeech' ...}".

4.4. Open the log file in the workdir folder. It includes a log of all the commands AlignTool has just executed. Search for "Error" to establish whether any processing step has caused AlignTool to report an error.

4.5. Reload and open the TextGrid file. (Please remember that loaded TextGrid files are not updated in Praat, see Section 3.1.) It should now include a new tier called *seg.speech*. If the analysis went well, the stretches labelled "speech" here correspond to the interval in the full trial when the participant spoke (see Fig. 18).

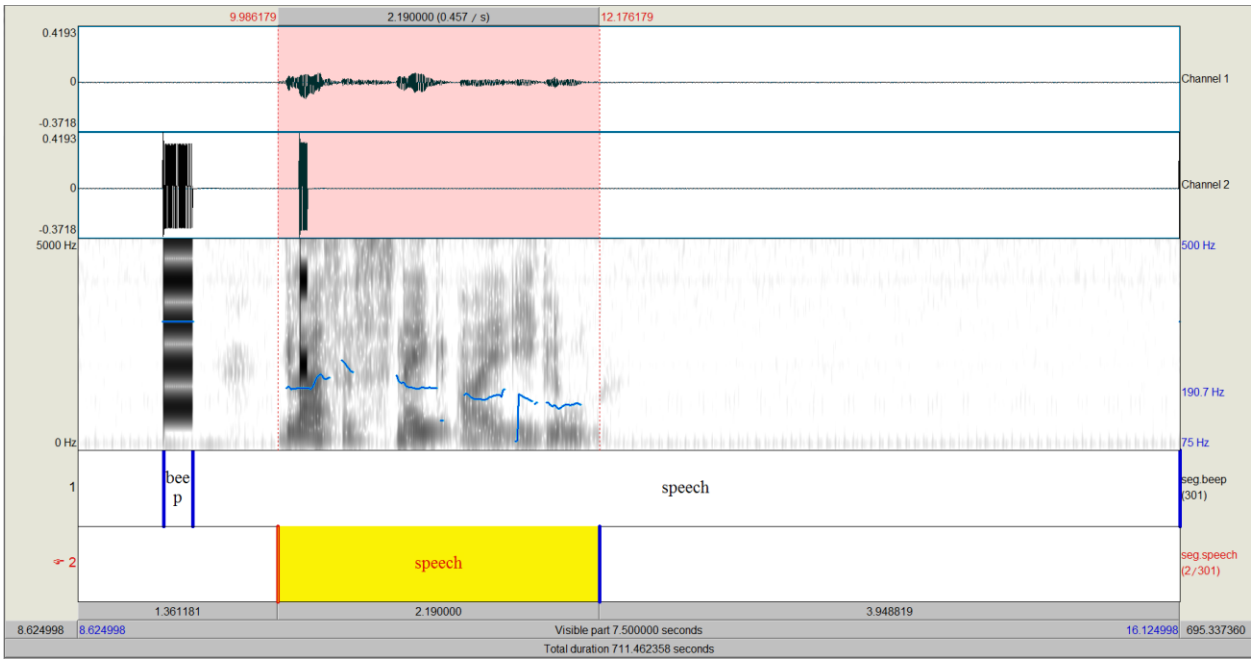


Figure 18: Example TextGrid file after Step 4

Step 5. Transcription

In this step, we will add a third tier with transcriptions of each trial in the TextGrid file.

5.1. Return to the "batch" sheet. In this step, we will add a new tier for the transcriptions with the "addTier" command. The settings for this command are specified in column D. We use:

```
'-s seg.beep -d anno.trans -t TODO -m copy
```

This reads as: add a tier by copying the existing *seg.beep* tier, calling it *anno.trans*. Insert "TODO" as a placeholder transcription in all speech intervals ("TODO" will be replaced by real transcriptions in Step 5.8). If the command is not stated in column D, type it in.

5.2. Save and close the workbook.

5.3. Click on "addTier" in the AlignTool GUI. When complete, the online log in the Terminal states "finished BatchRunner {batchcmd': 'addTier' ...}"

5.4. Open the log file in the workdir folder. Search for "Error" to establish whether any processing step has caused AlignTool to report an error.

5.5. Check the TextGrid file in Praat: It should now include a new tier called *anno.trans*.

5.6. Import the augmented TextGrid by clicking "Import from TextGrids" in the AlignTool GUI.

5.7. Open the workbook: The "segments" sheet now includes a series of lines that are identical to the lines for "speech" added before (in *seg.speech*), but now include "TODO"

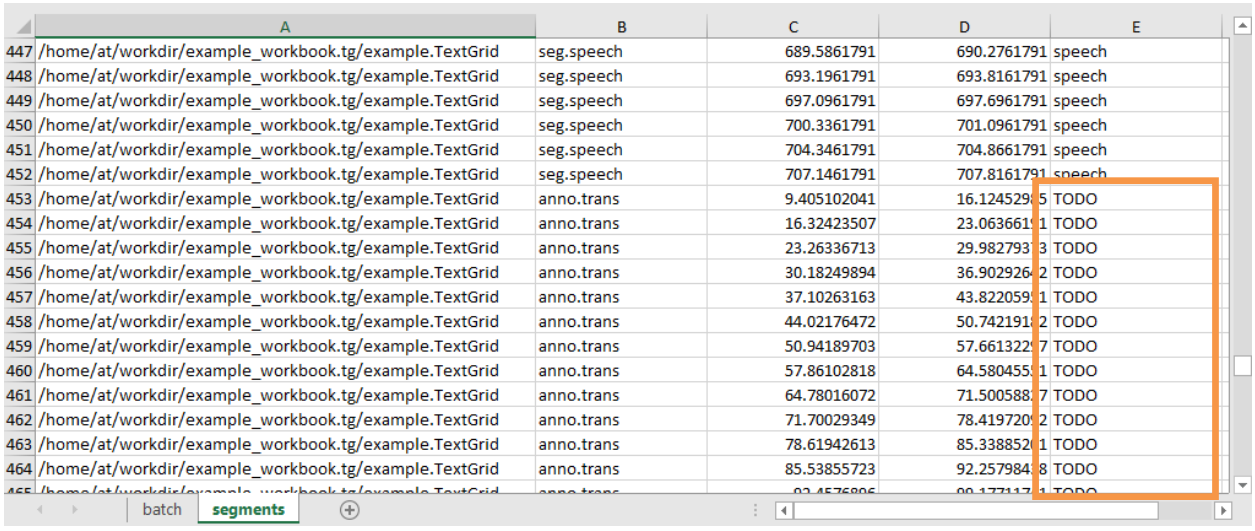
in the last column instead of “speech” and the corresponding tier is called *anno.trans*. These lines are located below the “speech” lines (see Fig. 19).

Here we need to fill in our transcriptions. You find the transcriptions for our example in “example_transcriptions.xlsx” in your working directory. We will use the intended utterances in order to show that AlignTool can deal with real experiment data, including trials, in which the subject did not respond and trials, in which the subject did not utter the intended word. (Of course, you can also use the transcriptions of the participant’s actual responses, but since erroneous utterances are typically excluded from (response) time analyses, this is not necessary in most cases.)

5.8. Copy and paste the transcriptions replacing the “TODO”s (see Fig. 20).

5.9. Save and close the workbook.

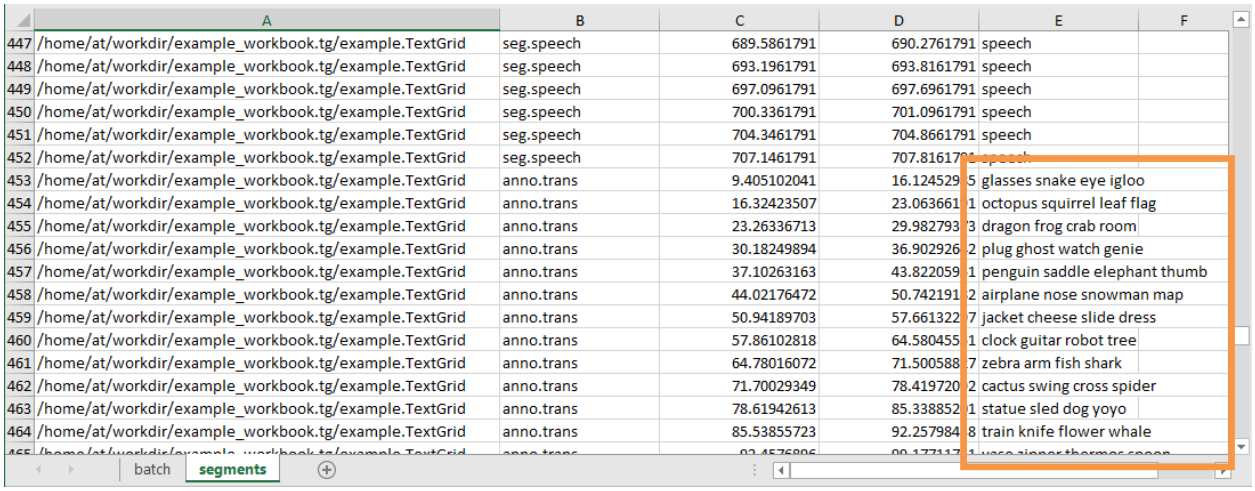
5.10. Click on “Export to TextGrids”. This will replace the “TODO”s with the transcriptions in the TextGrid file.



	A	B	C	D	E
447	/home/at/workdir/example_workbook.tg/example.TextGrid	seg.speech	689.5861791	690.2761791	speech
448	/home/at/workdir/example_workbook.tg/example.TextGrid	seg.speech	693.1961791	693.8161791	speech
449	/home/at/workdir/example_workbook.tg/example.TextGrid	seg.speech	697.0961791	697.6961791	speech
450	/home/at/workdir/example_workbook.tg/example.TextGrid	seg.speech	700.3361791	701.0961791	speech
451	/home/at/workdir/example_workbook.tg/example.TextGrid	seg.speech	704.3461791	704.8661791	speech
452	/home/at/workdir/example_workbook.tg/example.TextGrid	seg.speech	707.1461791	707.8161791	speech
453	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	9.405102041	16.124529[illegible]5	TODO
454	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	16.32423507	23.063661[illegible]1	TODO
455	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	23.26336713	29.982793[illegible]3	TODO
456	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	30.18249894	36.902926[illegible]2	TODO
457	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	37.10263163	43.822059[illegible]1	TODO
458	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	44.02176472	50.742191[illegible]2	TODO
459	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	50.94189703	57.661322[illegible]7	TODO
460	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	57.86102818	64.580455[illegible]1	TODO
461	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	64.78016072	71.500588[illegible]7	TODO
462	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	71.70029349	78.419720[illegible]2	TODO
463	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	78.61942613	85.338852[illegible]1	TODO
464	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	85.53855723	92.257984[illegible]8	TODO

batch segments

Figure 19: Example workbook in Step 5.7.



	A	B	C	D	E	F
447	/home/at/workdir/example_workbook.tg/example.TextGrid	seg.speech	689.5861791	690.2761791	speech	
448	/home/at/workdir/example_workbook.tg/example.TextGrid	seg.speech	693.1961791	693.8161791	speech	
449	/home/at/workdir/example_workbook.tg/example.TextGrid	seg.speech	697.0961791	697.6961791	speech	
450	/home/at/workdir/example_workbook.tg/example.TextGrid	seg.speech	700.3361791	701.0961791	speech	
451	/home/at/workdir/example_workbook.tg/example.TextGrid	seg.speech	704.3461791	704.8661791	speech	
452	/home/at/workdir/example_workbook.tg/example.TextGrid	seg.speech	707.1461791	707.8161791	speech	
453	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	9.405102041	16.124529[illegible]5	glasses snake eye igloo	
454	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	16.32423507	23.063661[illegible]1	octopus squirrel leaf flag	
455	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	23.26336713	29.982793[illegible]3	dragon frog crab room	
456	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	30.18249894	36.902926[illegible]2	plug ghost watch genie	
457	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	37.10263163	43.822059[illegible]1	penguin saddle elephant thumb	
458	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	44.02176472	50.742191[illegible]2	airplane nose snowman map	
459	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	50.94189703	57.661322[illegible]7	jacket cheese slide dress	
460	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	57.86102818	64.580455[illegible]1	clock guitar robot tree	
461	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	64.78016072	71.500588[illegible]7	zebra arm fish shark	
462	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	71.70029349	78.419720[illegible]2	cactus swing cross spider	
463	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	78.61942613	85.338852[illegible]1	statue sled dog yoyo	
464	/home/at/workdir/example_workbook.tg/example.TextGrid	anno.trans	85.53855723	92.257984[illegible]8	train knife flower whale	

batch segments

Figure 20: Example workbook in Step 5.8.

5.12. Cross check the TextGrid file: Now, the *anno.trans* tier should include the transcriptions where it said "TODO" before. The transcripts should appear where it says "speech" in the *seg.beep* tier.

Step 6. Align MAUS

In this step we take the *anno.trans* tier and the *seg.speech* tier and enter them into MAUS (webMAUS, to be precise) in order to align the transcripts and the audio signals.

6.1. With the command *alignMAUS* we use the following settings in the "batch" sheet of the workbook (column F):

```
'-s seg.speech -l eng-GB
```

This tells AlignTool that *seg.speech* is the tier providing an initial segmentation into intervals containing speech.

6.2. Save and close the workbook.

6.3. Click "alignMAUS" in the AlignTool GUI. This processing step may take a little while. When finished, the online log in the Terminal should state "finished BatchRunner {batchcmd': 'alignMAUS' …}"

6.4. Open the log file in the workdir folder. Search for "Error" to establish whether any processing step has caused AlignTool to report an error.

6.5. Check the TextGrid: It should now include two new tiers: *maus.ort* and *maus.pho*. In *maus.ort* the audio signal is aligned to the lexemes given in the transcriptions; *maus.pho* delivers phonemewise segmentations based on the orthographic and phonological transcriptions of the data.

6.6. Finally, import the data from the TextGrids to the workbook once again by clicking "Import from TextGrids".

Step 7. Extract On-/Offsets

The last step is to extract the word onset and offset times from the "segments" sheet of the workbook to a new sheet called "on_offsets". This will provide you with the onset and offset times measured as of the beep onset time, i.e., the trial onset.

7.1. You find the settings for the command *Extract On/Offsets* in column G of the "batch" sheet. We use:

```
'-f seg.beep
```

7.2. Save and close the workbook.

7.3. Run *Extract On/Offsets* in the AlignTool GUI.

7.4. Check the workbook: AlignTool has added a third sheet “on_offsets”, specifying the onsets and offsets of each spoken word and phoneme (to the extent that it could be annotated in the first place).

5. Try this at Home: Analyzing your own Audio Data

In this chapter, we will give you instructions on how to analyse your own audio files. The steps will be the same as in Chapter 4 for recordings of full experimental blocks that include beeps to mark the onsets of individual trials. For trial-by-trial audio recordings, and recordings of spontaneous speech, which AlignTool can handle as well, Steps 3 and 5 need to be customized as detailed below.

Step 0. Recording of Participant Utterances

Try to ensure that the recording is of the best possible quality with as little noise as possible for an optimal signal-to-noise ratio. Make sure that the microphone is placed in such a way that participants do not breathe audibly into the microphone. To achieve this, we recommend using a head-mounted microphone placed above the nose. When no head-mounted microphone is available and the microphone is placed on a table or a microphone stand, make sure that participants do not slouch in their seats during the course of the experiment, as this may impact negatively on the loudness of their speech. Participants should not be handed pieces of paper (e.g. written instructions) that they might play with during the recording. When a button box needs to be used during the experiment, its responses should be registered as silently as at all possible. With respect to the recording equipment, noise from computer ventilators and other electric interference should be minimized. Recording levels should be set at levels that avoid clippings of the audio signal during the recording. Finally, if a beep signal or another signal is used to indicate the onset of a trial, this should best be recorded directly via the line-in channel rather than being played overtly.

In our experience, ideal recording conditions are rarely given, so we configured AlignTool to perform with recordings of poorer quality, too. However, AlignTool is likely to perform less robustly and accurately with audio signals of poorer quality, requiring the user to correct more trials than with audio signals of better quality.

If you are planning to record full experimental blocks with beeps indicating trial onset times and other events in the audio file, it is ideal when you record the speech signal and the beep signal in separate channels. AlignTool can work with recordings of both beeps and speech in both channels, too, but it will perform more robustly when beeps and speech are recorded in separate channels. On the AlignTool website (https://www.linguistics.rub.de/~belke/aligntool.shtml), you can find a sample trial onset beep.

When you mark more than one event by beeps (e.g. trials onsets and voice key onset times, as in our example recording), make sure that the beeps for each type of event are unique. In our example, the beeps for each event differ in length and frequency.

Step 1. Specify Workbook

1.1. Edit the workbook: We recommend editing the example workbook, because the first line of the batch sheet needs to be spelled exactly as in the example workbook. Otherwise, AlignTool will not be able to read them.

You can specify all settings for the commands now or edit them later; see for yourself what works best for you. We will explain different options for the command settings in the corresponding step of this instruction (see also Appendix A for a list of the command settings and their parameters).

You will have to try out which command settings work best with your recording environment. In Appendix A, section A6, we provide some recommendations on how to best do this. It is useful to manually annotate some data in order to evaluate the quality of the alignment with each parameter setting that you try. Once you have found the optimal command settings for your recording environment, you can carry them forth to new recordings.

1.2. Save and close the workbook.

1.3. Set the file path and name of the workbook in the AlignTool GUI using the "Browse" button under "Active Workbook".

Step 2. Specify Audio file(s)

2.1. Copy the audio files you want to analyse to the "wav" folder in the working directory.

2.2. Set the file path to the "wav" folder in the working directory by clicking on "Browse" under "Active workbook's "batch" sheet initialization".

2.3. Click on "Find *.wav" in the AlignTool GUI.

2.4. Open the workbook and check whether all audio files have been found and added in column A.

2.5. Also check whether the names of the TextGrid files in column B are exact mirror images (by name) of the wav files.

Step 3. Creating the *seg.beep* Tier

For recordings of full experiments or experimental blocks that include beeps to mark the onsets of individual trials, Step 3 will retrieve these beeps and segment the full audio file into trial-like segments in a *seg.beep* tier (section A). For the other two cases – trial-by-trial recordings of participants' utterances (section B) and recordings of spontaneous speech (section C) – we will create a dummy *seg.beep* tier in Step 3. This way, we can treat all recordings in the same way again as of Step 4.

A) Segment Beeps in Beep-Segmented Recordings of Multiple Trials

3.1. If possible, copy a reference beep (i.e. a recording of the beep used in your experiment) to the working directory. Note that the beep file needs to be a Mono recording. The sample beep we provide on the AlignTool website is a mono recording.

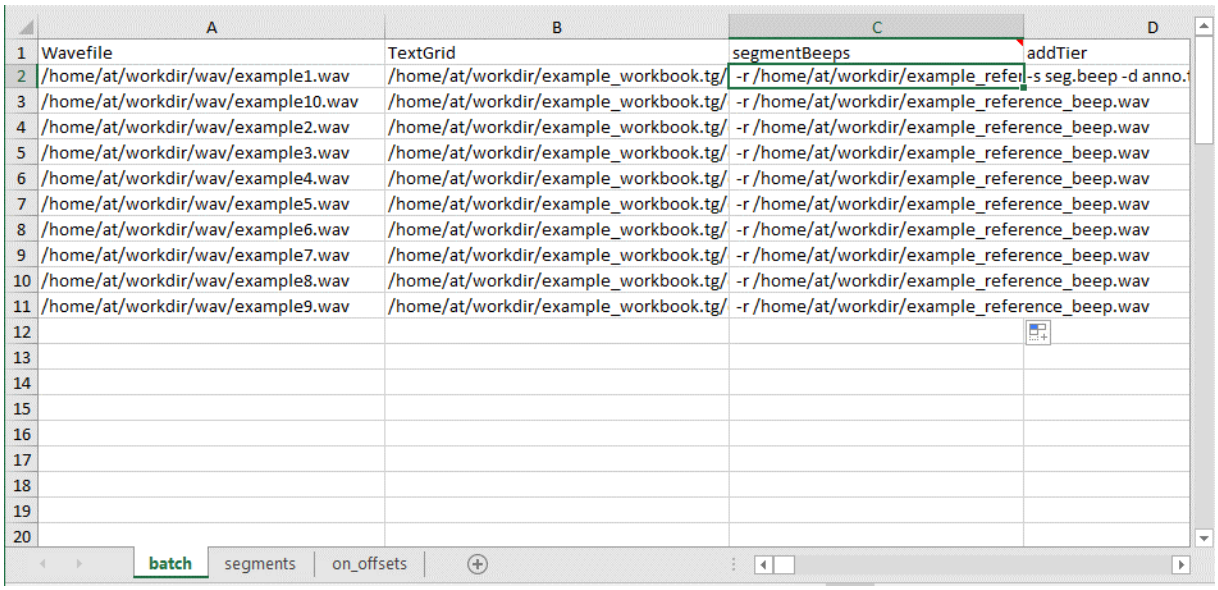
3.2. Edit the settings for the command *segmentBeeps* in column C in the workbook. The command can be executed more correctly, if you specify a reference beep, indicated here by the placeholder name *reference_beep*.wav".

```
' -r /home/at/workdir/reference_beep.wav
```

You can set additional parameters, as described in Appendix A. Please note that all settings for the commands have to start with a single quotation mark (') in order to make Excel read them as text and not as a formula. Please remember not to copy and paste the command line from this PDF, as this may cause problems for AlignTool.

3.3. Insert the command settings to all lines, respectively for each wav file (see Fig. 21 for an example).

3.4. Save and close the workbook.



	A	B	C	D
1	Wavefile	TextGrid	segmentBeeps	addTier
2	/home/at/workdir/wav/example1.wav	/home/at/workdir/example_workbook.tg/	-r /home/at/workdir/example_refer	-s seg.beep -d anno.t
3	/home/at/workdir/wav/example10.wav	/home/at/workdir/example_workbook.tg/	-r /home/at/workdir/example_reference_beep.wav	
4	/home/at/workdir/wav/example2.wav	/home/at/workdir/example_workbook.tg/	-r /home/at/workdir/example_reference_beep.wav	
5	/home/at/workdir/wav/example3.wav	/home/at/workdir/example_workbook.tg/	-r /home/at/workdir/example_reference_beep.wav	
6	/home/at/workdir/wav/example4.wav	/home/at/workdir/example_workbook.tg/	-r /home/at/workdir/example_reference_beep.wav	
7	/home/at/workdir/wav/example5.wav	/home/at/workdir/example_workbook.tg/	-r /home/at/workdir/example_reference_beep.wav	
8	/home/at/workdir/wav/example6.wav	/home/at/workdir/example_workbook.tg/	-r /home/at/workdir/example_reference_beep.wav	
9	/home/at/workdir/wav/example7.wav	/home/at/workdir/example_workbook.tg/	-r /home/at/workdir/example_reference_beep.wav	
10	/home/at/workdir/wav/example8.wav	/home/at/workdir/example_workbook.tg/	-r /home/at/workdir/example_reference_beep.wav	
11	/home/at/workdir/wav/example9.wav	/home/at/workdir/example_workbook.tg/	-r /home/at/workdir/example_reference_beep.wav	
12				
13				
14				
15				
16				
17				
18				
19				
20				

batch | segments | on_offsets

Figure 21: Insert command settings to all lines

3.5. Press "segmentBeeps" in the AlignTool GUI.

3.6. Open the log file in the workdir folder. It includes a log of all the commands AlignTool has just executed. Search for "Error" to establish whether any processing step has caused AlignTool to report an error. Often, such errors will only affect single files. When such an error occurs, AlignTool reports it and continues to work with the next file. This is why it is important that you inspect the log file for potential errors.

3.7. You should see now that a new folder has been created in your working directory. Its name is the same as that of your workbook plus ".tg". It includes TextGrid files by the names previously specified in column B of the workbook.

Open the TextGrid and wav file for one audio file (or more, if you prefer) and check whether the TextGrid has been created and whether it is complete: It should show the

wav file and one tier called *seg.beep*. This tier includes stretches labelled "beep" where a beep was found and "speech" where speech may later be allocated.

3.8. If necessary, manually edit and/or change the segments in the *seg.beep* tier.

3.9. Next, you need to synchronize the TextGrid file with the workbook, so click on "Import from TextGrids" in the AlignTool GUI.

3.10. Open the workbook: It should now include a new sheet called "segments". In it, you see numerical codings of the intervals in the audio file you have previously inspected in Praat. The first line shows the full length of the file, next, each segment is listed, stating the beginning and the end as well as the content ("beep" or "speech"). This is repeated for each wav file. If the beeps were extracted correctly, you should find exactly as many "speech" segments as there were trials in the experiment.

B) Creating the *seg.beep* Tier in Trial-by-Trial Recordings of Experiments

3.1. Edit the settings for the command "AddTier" in column D in the workbook.

```
'-d seg.beep -t speech -m all
```

This will create a new TextGrid file for each wav file, which includes one tier, *seg.beep*. This tier includes one interval, corresponding to the full length of the file. "-t speech" specifies that the content of the interval is "speech".

Please note that all settings for the commands have to start with a single quotation mark (') in order to make Excel read them as text and not as a formula. Also, please type in the command line and do not copy it from this PDF, as this may cause problems for AlignTool.

3.2. As before, insert the command settings to all lines, respectively for each wav file (see Fig. 21 above).

3.3. Save and close the workbook.

3.4. Press "AddTier" in the AlignTool GUI.

3.5. Open the log file in the "workdir" folder. Search for "Error" to establish whether any processing step has caused AlignTool to report an error.

3.6. You should now see that a new folder has been created in your working directory. Its name is the same as that of your workbook plus ".tg". It includes TextGrid files by the names previously specified in column B.

Open the TextGrid and wav file for one audio file (or more, if you prefer) and check whether the TextGrid has been created and whether it is complete: It should show the wav file and one tier called *seg.beep*. This tier spans the full audio file and is labelled "speech". As of here, the steps are the same as for recordings of multiple trials segmented by beeps.

3.6. Next, you need to synchronize the TextGrid file with the workbook, so click on "Import from TextGrids" in the AlignTool GUI.

3.7. Open the workbook: It should now include a new sheet called "segments". In it, you see numerical codings of the stretches of the audio file you have previously inspected in Praat. The first line shows the full length of the file, next, each interval is listed, stating the beginning and the end as well as the content ("speech"). This is repeated for each wav file. If the tiers were created correctly, you should find exactly as many "speech" segments as there were trials in the experiment.

C) Creating the *seg.beep* and *anno.trans* Tiers in Recordings of Spontaneous Speech

This recording format contains the longest stretches of speech of all recording formats. For AlignTool to analyse the recordings, the recording must often be pre-segment the recording into intervals of 30 sec or less (the most suitable interval length may vary). We will create a TextGrid file for each recording with a tier called *seg.beep* and a tier called *anno.trans*. For practical reasons, it is best to start creating *anno.trans* first and *seg.beep* second, as detailed below.

3.1 Create a folder called "*Name_of_the_workbook*.tg".

3.2. Open each wav-recording and create a new TextGrid tier called *anno.trans*. Use this TextGrid tier to establish a preliminary segmentation of the utterance in intervals of about 30 seconds. We recommend that you look out for pauses in the speaker's utterance for placing the preliminary boundaries, as these pauses allow you to set the boundaries anywhere within the pause. However, make sure that you do not place the interval boundaries too far within the pause, as this may impact negatively on the alignments performed later with *alignMAUS*. Also, note that the most suitable interval length might vary depending on various factors such as accuracy requirements or signal quality. For longer segments, alignment errors may increase since non-optimal alignment paths become more probable. A typical observation is that the alignment error increases over the course of the utterance, especially when there are longer silent stretches in a recording.

Enter the literal transcription of the speech contained in the pre-segments intervals. In doing so, you anticipate some of the work in Step 5 (see below). Note that transcripts must not include any special characters, such as \, " etc.

3.3 Save the TextGrid in the folder "*Name_of_the_workbook*.tg". Use the same name for the TextGrid file as you have used for the wav file. Otherwise the subsequent automated processing steps with AlignTool will not work.

3.4. In the next step, use the *addTier* command to copy the *anno.trans* tier to a new tier called *seg.beep*, writing "speech" to each interval instead of the transcription. To this end, edit the command *addTier* in the workbook as follows:

```
'-m trim -s anno.trans -d seg.beep -t speech
```

3.5. Insert the command settings to all lines.

3.6. Save and close the workbook.

3.7. Run *addTier* in the AlignTool GUI.

3.8. Check the TextGrids: They should now include a new tier called *seg.beep*, featuring "speech" in each interval. You may notice that compared to all previous analyses, the order of the tiers *anno.trans* and *seg.beep* is reversed in this case. This is no problem for AlignTool.

3.9. Next, you need to synchronize the TextGrid file with the workbook, so click on "Import from TextGrids" in the AlignTool GUI.[3]

3.10. Open the workbook: It should now include a new sheet called "segments". In it, you see numerical codings of the stretches of the audio file you have previously established in Praat. Each segment should be listed, stating the beginning and the end as well as its content, i.e. the transcription you added earlier. This is repeated for each wav file.

Step 4. Segment Speech

This processing step is required for trial-based recordings only, which typically include long silent intervals before and after the speech signal.

4.1. Next, edit the settings for the command "segmentSpeech" in column E in the workbook.

```
'-f seg.beep
```

4.2. Add additional parameters, as needed (see Appendix A).

4.3. Insert the command settings to all lines.

4.4. Save and close the workbook.

4.5. Run *segmentSpeech* in the AlignTool GUI.

4.6. Open the log file in the workdir folder. Search for "Error" to establish whether any processing step has caused AlignTool to report an error.

---

[3] As an alternative to steps 3.4 to 3.9, you can use the *Import from TextGrids* function of AlignTool to first import the information in the *anno.trans* tiers in the TextGrid files stored in step 3.3 into the workbook. There, copy all lines including *anno.trans* in the "segments" sheet and replace the tier name in the copied lines by *seg.beep* and the contents of the tier intervals by "speech". Then use AlignTool's *Export to TextGrid* function to export the added tier information to the TextGrid Files. Save the workbook.

4.7. Check the TextGrids: They should now include a new tier called *seg.speech*. If the analysis went well, the stretches labelled "speech" here correspond to the intervals of the full trial when the participant did speak.

Step 5. Transcription

A&B) Adding Transcriptions of Trial-Based Utterances

In this step, you will add a new tier for the transcriptions with the *addTier* command.

5.1. Return to the "batch" sheet and edit column D (*addTier*).

```
'-s seg.beep -d anno.trans -t TODO -m copy
```

5.2. Add, modify, or remove parameters, as needed (see Appendix A).

5.3. Insert the command settings to all lines.

5.4. Save and close the workbook.

5.5. Click on "addTier" in the AlignTool GUI.

5.6. Check the TextGrid file(s) in Praat: They should now include a new tier called *anno.trans*.

5.7. Import the augmented TextGrid file by clicking "Import from TextGrids" in the AlignTool GUI.

5.8. Open the workbook: Each participant's section in the "segments" sheet should now include a series of lines that are identical to the lines for "speech" added before but now include "TODO" in the last column instead of "speech". These lines are located below the "speech" lines. Here you need to fill in your transcriptions.

In most cases, you will be able to paste the transcripts from the trial descriptions for the participant, as these will usually include the target utterances. To enter them using copy and paste, you can sort the "segments" sheet by the file name and/or other filters (if applicable) and copy the transcriptions into all the relevant cells.

Make sure that you use copy and paste carefully here: If the transcripts from the trial descriptions and the segmented intervals in the audio file do not fit in length, check whether you have recorded all trials (maybe you did not record practice trials). Also, note that the transcriptions must be ordered correctly so as to allocate the right transcription to the right speech interval or file.

Also, it is important that you make sure that you will be able to re-establish the original order of lines in the workbook after filling in the transcriptions. One way of going about this is to include a column in the "segments" sheet that codes the line numbers of the original file prior to sorting it. You can then use this column to return form the order used for filling in the transcriptions to the original order.

If you have trials of approximately equal length, you can also cross-check the results of the segmentation procedure by inspecting the length of the speech periods that AlignTool established. If one is double or three times the length of the others, this indicates that some beeps have not been detected.

5.9. Save and close the workbook.

5.10. Click on "Export to TextGrids". This will replace the "TODO"s with the transcriptions in the TextGrid file.

5.11. Cross check the TextGrid file: Now, the *anno.trans* tier should include the transcriptions where it said "TODO" before. The transcripts should appear where it says "speech" in the *seg.beep* tier.

C) Adding Transcriptions of Spontaneous Speech

You have already created the *anno.trans* tier in Step 3, along with the *seg.beep* tier, so you're done.

Step 6. AlignMAUS

6.1. Set the settings with the command *alignMAUS* (column F).

```
'-f seg.beep
```

6.2 Add, modify, or remove parameters, as needed (see Appendix A). Most importantly, set the appropriate language parameter. In recordings of spontaneous speech, --initialsilence may help greatly in improving the alignments by MAUS.

6.3. Insert the command settings to all lines.

6.4. Save and close the workbook.

6.5. Click "alignMAUS" in the AlignTool GUI.

6.6. Open the log file in the workdir folder. Search for "Error" to establish whether any processing step has caused AlignTool to report an error.

6.7. Check the TextGrids: They should now include two new tiers: *maus.ort* and *maus.pho*. In *maus.ort* the audio signal is aligned to the lexemes; *maus.pho* delivers segmentations based on the orthographic and phonological transcriptions of the data.

6.8. Manually edit the *maus.ort* and *maus.pho* tiers, if necessary.

6.8. Finally, import the data from the TextGrids to the workbook once again by clicking "Import from TextGrids".

Step 7. Extract On-/Offsets

7.1. Edit the settings for the command “Extract On/Offsets” in column G of the batch sheet.

```
-f seg.beep
```

7.2. Add, modify, or remove parameters, as needed (see Appendix A).

7.3. Insert the command settings to all lines.

7.4. Save and close the workbook.

7.4. Run *Extract On/Offsets* in the AlignTool GUI.

7.5. Check the workbook: AlignTool has added a third sheet “on_offsets”, specifying the onsets and offsets of each spoken word and phoneme (to the extent that it could be annotated in the first place.)

5. Other Applications of AlignTool

Apart from being a tool for analysing participant recordings, AlignTool can also be used to analyse many other types of recordings used in psycholinguistic research. Here are some ideas where you might benefit from AlignTool as well:

In studies using the visual world paradigm, participants typically view pictures or scenes on the screen while listening to words or sentences via headphones. Researchers are typically interested in when, relative to the onset of a word in the auditory stimulus, participants gazed at specific regions of the screen. AlignTool allows for an automatic temporal annotation of auditory stimuli in experiments, rendering experiment preparation less time-consuming than before.

In other experiment settings, researchers use pseudowords or combinations of pseudowords as auditory stimuli. For instance, in artificial grammar learning studies, participants listen to pseudoword utterances generated from an artificial grammar with an artificial vocabulary. Some researchers may be interested in measuring the onset and offset times of the pseudowords in the training utterances, which is possible with AlignTool: Pseudowords will cause the G2P service in MAUS to fall back to rule based phonetic transcriptions of the orthographic input. An important prerequisite for MAUS to perform well though is that the pseudowords are phonotactically plausible in the language that is chosen for the analysis.

Appendix A: Parameters of Commands in AlignTool

For each command, there is a range of parameters that can be specified. We list these parameters and their function in the following. In Schillingmann et al. (subm.), we provide an overview of the parameter settings we used for the evaluation of AlignTool. These may serve as an orientation for the parameter settings you may want to use for your own work. Eventually, however, you will have to try out which parameter settings work best for your recording scenario and audio quality. In section A6, we give some advice on how to best do this. Note that identifying the best parameter set requires some manually annotated data so as to be able to evaluate the quality of the alignment with each parameter setting. It may take a little while to identify the optimal parameter settings, but it is worth the effort: By our experience, you will be able to carry forth an optimal parameter setting for a given recording scenario to new recordings.

Before we list each command and its parameters, please note the following general principles of parameter specification and notation:

- You can specify a parameter by using a short form (consisting of the first letter, preceded by -) or by using their full name (preceded by --), as in -h or --help for the *help* parameter.
- The order of parameters is not relevant.
- A parameter that is enclosed in [ ] is optional; by implication, a parameter that is not enclosed in [ ] is not optional and must be specified. In the GUI version of AlignTool, many of the obligatory parameters, such as -i (for input file), -o (for output file) and -w (for wav file), are specified in the workbook automatically by the *Find *.wav* routine and need not be specified by the user.
- Some parameters require an additional argument, which is marked with < >. For some parameters, AlignTool has inbuilt default values. We list these below, where applicable.
- Parameters and arguments are separated by spaces.

A1. *segmentBeeps*

Usage: aligntool.py segmentBeeps [-h] [-i <infile>] -o <outfile> -w <wavfile> [-b <channel>] -r <refbeep> [-x <0-1>] [-s <db>] [-m <s>] [--seekflank]

segmentBeeps is required for the analysis of beep-segmented recordings of full experimental blocks. It establishes the time periods in the audio file corresponding to beeps, labelling them "beep", and labels the time periods between two beeps and after the last beep as "speech". In order for *segmentBeeps* to work as accurately as possible, users should provide it with a reference beep, i.e. a sample of the beep signal. Using this reference beep, *segmentBeep*s inspects the audiofile from left to right and establishes the periods of maximum correlation between the reference beep and the audio file. These will be in those places where the reference beep fully overlaps with a beep in the audio file. By setting the optional parameter --seekflank, you can exploit the seekflank heuristic for establishing the beep. Functionally, this heuristic looks out for steep ascents and descents in db in the audio signal, which can help optimize the identification of onset and offset times of beeps in the audio file.

SegmentBeeps creates a TextGrid file by the name specified in the workbook and establishes a tier in the TextGrid file called *seg.beep*. Typically, the TextGrid file name is the same as the name of the wav file; this is ensured by the *Find *.wav* command; see Step 2 in Chapters 4 and 5.

Name	Function	Defaults
-h --help	shows help message and exits.	n/a
-i <infile>, --input-textgrid <infile>	specifies the input TextGrid file. This parameter is rarely needed in using AlignTool, as *segmentBeeps* generates a TextGrid file on its own, naming it as specified in the workbook. Typically, the name will be the same as the wav file; this is ensured by the *Find *.wav* command.	none
-o <outfile>, --output-textgrid <out-file>	specifies the output TextGrid file. This parameter is rarely needed in using AlignTool, as *segmentBeeps* generates a TextGrid file on its own, naming it as specified in the workbook. Typically, the name will be the same as the wav file; this is ensured by the *Find *.wav* command.	none
-w <wavfile>, --wav-file <wavfile>	specifies the wav file. This parameter is rarely needed in using AlignTool, as *segmentBeeps* retrieves the wav file from the workbook.	none
-b <channel>	specifies the channel the beep is located in; typically, this is the right channel (2). If it is the left one, set "-b 1"	2
-r <refbeep>, --refbeep <refbeep>	specifies the name of the wav file that includes the reference beep signal. If there is more than one reference beep, the -r parameter can be used multiple times.	none
-x <0-1>, --mincorrelation <0-1>	specifies the minimum required cross correlation peak for a beep signal to be identified.	0
-s <db>, --silencethreshold <db>	specifies the silence threshold for the beep segmentation, i.e. the difference in db between the highest peak (typically the beep) and silence.	-25
-m <s>, --minsounding <s>	specifies the minimal duration (in s) of the intervals of the audiofile that are used for identifying the beeps in the signal.	0.180
--seekflank	enables the seekflank heuristic for postprocessing the cross-correlation method.	false

Examples of the *segmentBeeps* command specifications in the workbook:

a) segmentBeeps -b 1 -r "doc/tatas_dutch_beep.wav" --seekflank
(alternatively: segmentBeeps -b 1 --refbeep "doc/tatas_dutch_beep.wav" --seekflank)

This command setting specifies that the beep is to be found in the left channel, rather than the right, as is set per default. It also specifies a reference beep file and enables the seekflank heuristic to optimize the cross-correlation method for identifying the beeps in the audio signal.

b) segmentBeeps -r "doc/tatas_german_beep.wav" -r "doc/tatas_german_beep2.wav" --seekflank

This command setting covers a case where multiple beeps were used for trial segmentation in the audio file.

A2. *segmentSpeech*

Usage: aligntool.py segmentSpeech [-h] [-i <infile>] -o <outfile> -w <wavfile> [-c <channel>] [-f <tier>] [-d] [--shiftonsets <s>] [--shiftoffsets <s>] [--trainbegin <s>] [--trainwindow <s>] [--speechthresh <f>] [--snradd <db>]

SegmentSpeech uses Praat functionality in order to establish speech intervals in the audio file. The user can specify an existing tier as a pre-segmentation tier, for instance *seg.beep*. AlignTool will then search for speech in each of the intervals marked as "speech" in the filter tier. Note that in keeping with the notion that there is typically one utterance per trial (i.e., tier interval), *segmentSpeech* looks out for one, not multiple, speech intervals. Pauses within this interval are detected by *alignMAUS*. *SegmentSpeech* creates a new tier called *seg.speech*. In it, stretches of speech will be labelled as "speech". In order to optimize the accuracy of *segmentSpeech*, you can play around with a variety of parameters.

- --snradd: AlignTool uses Praat to extract an intensity contour of the audio file. It then shifts a window over the intensity contour. The window length can be specified by using the --trainwindow parameter. For each window, the maximum intensity value of all intensity values inside the window is established. The window with the minimal score determines the silence threshold (= reference energy). Users can specify that in order to be classified as speech, the audio signal must differ from the silence threshold by at least x db, for instance 2 db (--snradd). The --snradd parameter should be used with care though, as a too conservative (i.e., too high) setting may cause AlignTool not to detect silent fricatives or other speech events that differ little from the silence threshold. This is especially true for recordings with a rather poor signal-to-noise ratio.
- -d, --denoise: Users can denoise the signal. Note, however, that this will affect not only the silent intervals but also the speech intervals. This is why this parameter, too, should be used with care. Using the --trainbegin and the --trainwindow parameter, you can specify the beginning of the interval for calculating the reference energy for denoising.
- --speechthresh: Aligntool post-processes segments with energy above the silence threshold. First the average energy of all segments exceeding the silence threshold is calculated. The average energy is weighted by the speech threshold specified by the user. It specifies a fraction of the average intensity that helps classify which segments in the audio signal are to be considered as speech. Segments with energy below the weighted average energy are removed. A speech threshold of 0 keeps all segments. A speech threshold of 1 only keeps segments exceeding the average energy.

Name	Function	Defaults
-h --help	shows help message and exits.	n/a
-i <infile>, --input-textgrid <in-file>	specifies the input TextGrid file. This parameter is rarely needed in using AlignTool, as *segmentSpeech* generates a TextGrid file on its own, naming it as specified in the workbook. Typically, the name will be the same as the wav file; this is ensured by the *Find *.wav* command.	none
-o <outfile>, --output-textgrid <outfile>	specifies the output TextGrid file. This parameter is rarely needed in using AlignTool, as *segmentSpeech* generates a TextGrid file on its own, naming it as specified in the workbook. Typically, the name will be the same as the wav file; this is ensured by the *Find *.wav* command.	none
-w <wavfile>, --wav-file <wavfile>	specifies the wav file. This parameter is rarely needed in using AlignTool, as *segmentSpeech* retrieves the wav file from the workbook.	none
-c <channel>	specifies the channel the speech signal is located in; typically, this will be the left channel (1). If it is the right channel, set "-c 2"	1
-f <tier>, --filter-tier <tier>	specifies an existing tier to filter speech intervals based on an existing speech interval tier; when the filter tier does not specify a given speech interval as relevant to the experiment, *segmentSpeech* will not include it in the *seg.speech* tier.	none
-d, --denoise	enables the Praat denoising functionality. Use with care as denoising affects not only the silent intervals but also the speech intervals.	false
--shiftonsets <s> and --shiftoffsets <s>	shifts all detected onsets (offsets) by <s> seconds. This functionality should only be used when larger-scale measurements of the audio data by hand have established that the onset (offset) measures provided by AlignTool should best be shifted systematically by <s> seconds.	0
--trainbegin <s>	specifies the beginning of the interval for calculating the reference energy for denoising at <s> seconds (see the introductory text to the *segmentSpeech* command for more information).	3.3

--trainwindow <s>	specifies the length of the interval for calculating the reference energy (<s> seconds) (see the introductory text to the *segmentSpeech* command for more information).	1
--speechthresh <f>	specifies the extent to which the signal needs to exceed the average intensity in order to be considered as speech. <f> can be read as a percentage score when multiplied by 100 (see the introductory text to the *segmentSpeech* command for more information).	0.5, i.e. 50%
--snradd <db>	add <db> to the silence threshold (see the introductory text to the *segmentSpeech* command for more information).	1

Examples of the *segmentSpeech* command specifications in the workbook:

a) -f seg.beep --snradd "2" --speechthresh 0.2

This command setting specifies that only those intervals of the audio signal are considered as speech that exceed the silence threshold established by Praat by 2 db and that exceed the average interval intensity by 20%. It also specifies that there is an existing tier (*seg.beep*) that is to be used as a filter when looking for relevant speech events.

b) -f seg.beep --snradd "8" --speechthresh 0.3 -c 1 --shiftonsets -0.02 --shiftoffsets 0.02

This command was used with a set of audio files of very poor audio quality. It specifies the same parameters as in the previous command but uses more conservative arguments in order to deal with the poor audio quality (8 db for the difference between the silence threshold and a speech-relevant event and 30% for the extent to which a relevant speech signal must exceed the average signal). These very conservative settings lead to onsets being established rather too late and offsets rather too early. This was compensated by shifting all speech onsets by -200 ms and all speech offsets by 200 ms.

Given the poor audio quality, this audio file would definitely need manual corrections of the automated measures provided by AlignTool.

c) --snradd "9" --speechthresh 0.4 -c 1 --trainwindow 0.1

The final example command was used with audio data of exceptionally high quality. This is why the difference between the silence threshold and a speech-relevant event was set to 9 db and the speech threshold, i.e. the extent to which the speech interval intensity must exceed the average interval intensity, to 40%. As there were only very short periods of silence in the audio signal for the silence threshold to be established, the length of the training window was set to 100 ms by means of the --trainwindow parameter.

A3. *addTier*

Usage: aligntool.py addTier [-h] [-i <infile>] -o <outfile> -w <wavfile> -m <mode> [-s <tier>] [-d <tier>] [-t <text>] [-f <re>]

With a*ddTier* you can add new tiers to TextGrid files.

Name	Function	Defaults
-h --help	shows help message and exits.	n/a
-i <infile>, --input-textgrid <infile>	specifies the input TextGrid file. This parameter is rarely needed in using AlignTool, as *addTier* generates a TextGrid file on its own, naming it as specified in the workbook. Typically, the name will be the same as the wav file; this is ensured by the *Find *.wav* command.	none
-o <outfile>, --output-textgrid <out-file>	specifies the output TextGrid file. This parameter is rarely needed in using AlignTool, as addTier generates a TextGrid file on its own, naming it as specified in the workbook. Typically, the name will be the same as the wav file; this is ensured by the *Find *.wav* command.	none
-w <wavfile>, --wav-file <wavfile>	specifies the wav file. This parameter is rarely needed in using AlignTool, as *addTier* retrieves the wav file from the workbook.	none
-m <mode>, --mode <mode>	specifies the mode of adding a new tier. Options are a tier spanning the whole file (-all), a copy of an existing tier (-copy) or a trimmed version of an existing tier (-trim), with the intervals to be trimmed being specified by a filter tier.	none
-s <tier>, --source-tier <tier>	specifies the tier that serves as the source for copying or trimming in modes copy and trim.	none
-d <tier>, --dest-tier <tier>	specifies the name of the newly added tier.	seg.speech
-t <text>	specifies the text content of the new tier's intervals.	speech
-f <re>, --filter <re>	specifies, for --mode copy, that only those intervals with content matching the pattern specified in the argument (<re>) is copied.	speech

Examples of the *addTier* command specifications in the workbook:

a) -s seg.beep -d anno.trans -t TODO -m copy

   This command setting specifies that the source tier *seg.beep* is copied to a new tier called *anno.trans* and that the text to be added in each interval of the new tier is TODO. As the filter parameter is set to <speech> by default, only those intervals labelled as "speech" in *seg.beep* are being copied to the new tier.

b) -m trim -s anno.trans -t speech -d seg.speech

   This parameter setting was used for a set of data with existing annotations in Praat, specified in the tier *anno.trans*. It causes *anno.trans* to be copied to *seg.speech*, with *seg.speech* simply including "speech" in each interval specified in *anno.trans*. *Mode trim* causes the new tier to be restricted to those parts of the wav file that were annotated. Other, non-annotated parts of the wav file are ignored.

A4. *alignMAUS*

Usage: aligntool.py alignMAUS [-h] [-i <infile>] -o <outfile> -w <wavfile> [-l <language>] [-c <channel>] [-f <tier>] [-s <tier>] [-d] [--initialsilence]

In this processing step, MAUS is used to generate forced aligments of the intervals in the audio signal previously identified as speech and the transcription you have provided for these intervals.

Name	Function	Defaults
-h --help	shows help message and exits.	n/a
-i <infile>, --input-textgrid <infile>	specifies the input TextGrid file. This parameter is rarely needed in using AlignTool, as *alignMAUS* generates a TextGrid file on its own, naming it as specified in the workbook. Typically, the name will be the same as the wav file; this is ensured by the *Find *.wav* command.	none
-o <outfile>, --output-textgrid <outfile>	specifies the output TextGrid file. This parameter is rarely needed in using AlignTool, as *alignMAUS* generates a TextGrid file on its own, naming it as specified in the workbook. Typically, the name will be the same as the wav file; this is ensured by the *Find *.wav* command.	none
-w <wavfile>, --wav-file <wavfile>	specifies the wav file. This parameter is rarely needed in using AlignTool, as *alignMAUS* retrieves the wav file from the workbook.	none
-l <language>, --language <language>	specifies the language MAUS is to be used with.	deu-DE

-c <channel>, --speech-channel <channel>	specifies the channel the speech signal is located in; typically, this will be the left channel (1).	1
-f <tier>, --filter-tier <tier>	specifies an existing tier to filter speech intervals based on an existing speech interval tier; when the filter tier does not specify a given speech interval as relevant to the experiment, *alignMAUS* will not include it in the *seg.speech* tier.	seg.beep
-s <tier>, --segmentation-tier <tier>	specifies the tier providing an initial speech segmentation. Note that if the default setting is used (*seg.beep*), the results of *segmentSpeech* do not come to bear for the analyses of *alignMAUS* and *alignMAUS* establishes the onset and the offset times of the utterances on its own. If *seg.speech* is specified as the segmentation tier, *alignMAUS* restricts its analyses to the intervals previously identified by *segmentSpeech*, bringing to bear the results of the analyses from that processing step. We recommend using *seg.speech* as the segmentation tier, especially in recordings with long silent intervals in each trial, as often seen in a typical naming task.	seg.beep
-d --denoise	enables denoising.	false
--initialsilence	enables MAUS to use initial and final silence models.	false

Examples of the *alignMAUS* command specifications in the workbook:

a) -s seg.speech

   The tier providing an initial speech segmentation is set to *seg.speech*, as created by the *segmentSpeech* command. No language is set, which implies that the default language, German (deu-DE) is being used.

b) -s seg.speech -c 2 -l nld-NL

   Same as in a), except that now the channel including the relevant speech signal is set to the right channel (2), and the language is set to Dutch (nld-NL)

A5. *extractOnOffsets*

Usage: extractOnOffsets [-h] [-f <tier>]

This command extracts onset and offset times from the segments sheet. For trial-by-trial based recordings of participants' utterances, it lists the onset and offset times of all relevant speech events. Additionally the beginning of the pre-segmentation interval (e.g. speech intervals in the *seg.beep* tier) is listed. For beep-segmented recordings the beep onset can be determined by subtracting the length of the reference beep from the beginning of the pre-segmentation interval.

Name	Function	Defaults
-h; --help	shows help message and exits.	n/a
f <tier>, --filter-tier <tier>	specifies an existing tier to filter speech intervals based on an existing speech interval tier; when the filter tier does not specify a given speech interval as relevant to the experiment, *extractOnOffsets* will not include it.	none

A6. How Can I Find the Right Parameters for my Recordings?

Which parameters to choose for a given recording depends to a large extent on the type and quality of the recording. So, in order to find the right parameters for your recording, you need to understand first what kind of recording you are dealing with and which problems may arise from the quality of the recordings.

In terms of the type of recording, we distinguish recordings of multiple trials of an experiment or trial-by-trial recordings (cases A and B in this Manual) on the one hand and recordings of semi-spontaneous speech (case C) on the other. We will focus on cases A and B first, considering primarily the *segmentSpeech* function, which is key for accurate alignments. After that, we consider recordings of semi-spontaneous speech (C). If you are analysing semi-spontaneous speech, please do not skip to the end of this section but read it in full, as some of the information we provide on trial-based recordings of type A and B may be relevant to recordings of semi-spontaneous speech as well.

A & B) Trial-Based Recordings

segmentSpeech. Typically, recordings of experimental trials include long silent intervals at the beginning and at the end of a trial. Let's take Rastle & Davis (2002) as an example, where participants read aloud words with simple vs. complex onsets. We have one wav file per trial (trial-by-trial recordings, type B) that each includes a silent interval (reflecting the participant's response preparation), the participant's spoken response, and a second silent interval from the end of the response to the end of the recording. Aligning the whole file using MAUS and allowing silence at the beginning and the end produces less accurate results than first estimating the exact temporal interval of the file that contains speech and subsequently running MAUS on the trimmed interval. This is why AlignTool includes the *segmentSpeech* function, which identifies, within a given trial, when a participant's utterance begins and ends, effectively cutting off the silent intervals before and after the utterance. It can do so automatically for a large number of trials.

In establishing the utterance interval, we can try to exploit the difference between speech and silence as good as we can. In recordings such as those by Rastle and Davis, the background noise is typically constant, so the intensity of the background noise can be estimated automatically by looking at the maximum intensity within a window of the audiofile with overall minimal intensity. To this end, *segmentSpeech* uses Praat to extract an intensity contour of the audio file and then shifts a window of a pre-defined length over this intensity contour. For each window, the maximum intensity value of all intensity values inside the window is established. The window with the smallest maximum intensity determines the silence threshold (= reference energy), reflecting the level of background noise in the recording. The aim of *segmentSpeech* is to establish an onset as soon as the audio signal intensity deviates by a substantial amount from the silence threshold (this amount can be specified using the --snradd parameter; the default is 1 db). As soon as the signal intensity drops below the silence threshold *segmentSpeech* assumes an offset of speech. This initial segmentation procedure uses Praat internally. The resulting speech windows are filtered further (see --speechthresh). As mentioned in section A2, *segmentSpeech* works on the assumption that there is only one chunk of speech per trial.

--*trainwindow.* The user can specify the length of the window by means of the --trainwindow parameter – in the case of the Rastle and Davis data, we set it to 0.1, corresponding to 100 ms. This is a rather short duration (the default is 1 s), which we chose because the length of the recordings was rather short (2 s) and there would have been no pure non-speech windows for analysis if we had set the window length to 1 s; each of the 1-s-windows would have included some speech. So, when the audio file does not contain larger chunks of background noise before or after participants start to speak, you should reduce the window length using the --trainwindow parameter.

--*speechthresh.* Critically, like any threshold-based method of measuring the onset of speech, this procedure may fail when it encounters non-speech sounds, such as smacking, that fulfill the criteria defined for speech. Smacking creates a small spike in the signal, so the interval including such a spike has a low average intensity. To be able to distinguish such windows from windows with speech that will have a high average intensity, the parameter --speechthresh can be adjusted between 0 and 1 to filter windows that are below the average intensity of other windows. For instance, when --speechthresh is set to 0, the average intensity of a given window needs to be above the average intensity of all windows to be classified as containing speech; when --speechthresh is set to 0.5, the average intensity of a given window only needs to reach at least 50% of the average intensity of all windows to be classified as speech.

--*snradd.* Unlike smacking, audible breathing is a continuous audio signal that is unlikely to be detected by optimizing the --speechthresh parameter. Instead, you may want to adjust the --snradd parameter to filter breathing. However, in doing so, you might also filter out voiceless fricatives or other speech sounds that are very similar in intensity to breathing – as we have pointed out before, the --snradd parameter must be used with care. In our experience, audible breathing noise is very hard to filter. Again, this makes the case for improving the recordings from the start (cf. Footnote 1 in Schillingmann et al., subm.).

These three parameters need to be configured on a per-corpus basis. In case there is a ground truth, i.e. manually annotated data, for a small subset of the corpus, the parameters can be tuned to the ground truth. This is worthwhile as audio files recorded within a given recording settings will typically require the same settings in AlignTool. A general recommendation for tuning the parameters is to carefully inspect the segmentation performed by *segmentSpeech* visually and auditorily in Praat. If breathing is incorrectly segmented as part of the beginning or end of an utterance --snradd should be increased in small steps of 1 db. When the speakers are not speaking very loudly and the signal-

to-noise ratio is low as a result, you may try increasing --speechthresh in increments of 0.1, starting from 0.5. If the audio files do not contain a minimum length of 1 s of silence before or after an utterance, --trainwindow should be lowered to the minimum silence length in seconds. It can also be increased if longer chunks of silence are known to be present.

segmentBeep. Apart from *segmentSpeech*, *segmentBeeps* can be optimized for beep-segmented recordings of multiple trials. After all, any measure of utterance onsets can only be as exact as the measure of the trial onset. Beeps are identified by means of a moving window, inspecting the audio file for periods of maximum correlation between the reference beep and the audio file.

--*refbeep.* For *segmentBeeps* to work well, you should specify a reference beep for the analyses using --refbeep; if more than one beep was used, all beeps should be specified. The reference beep(s) must be identical in length to the beep(s) used in the experiment and should not include silent intervals at the beginning or the end.

--*minsounding.* Depending on the length of the beep, users may need to reduce the minimal duration of the windows of inspection, such that the window length is shorter than the beep length (the default setting is 0.180 s).

--*seekflank.* Specifying the --seekflank heuristic may render the beep segmentation more exact.

--*silencethreshold.* Users may want to specify the difference in db of the highest peak in the audio signal and silence; its default is -25 db. In files with a poor signal-to-noise ratio, i.e. when the beep is hard to identify against the background noise, --silencethreshold may need to be reduced, ideally in steps of 1 db (-24, -23, …). The threshold is used to segment the file before beep detection. The segments are inspected for one beep each.

alignMAUS. For trial-based recordings, no parameter tuning needs to be done for *alignMAUS*. Note that transcripts must not include any special characters, such as \ or “.

C) Semi-Spontaneous Speech

alignMAUS. For recordings of semi-spontaneous speech, you need to prepare the audio files manually and then force-align them with WebMAUS using *alignMAUS*. They need to make sure that longer silent intervals at the beginning or the end of a recording are excluded. Typically, this is being dealt with as part of the pre-segmentation into shorter utterance segments in the first processing step (see Section 5 – Step 3 – Type C in this Manual). In this step, the tier *seg.beep* is created based on the manual annotations, which, by default, functions as a filter to which segments of the audio file MAUS is applied to. Apart from that, no additional tuning needs to be done, so the quality of the alignment is then largely down to MAUS. Note that transcripts must not include any special characters, such as \ or “.

--*initialsilence* allows MAUS to detect silence at the beginning and the end of an utterance. For longer silence regions, this will negatively affect the accuracy of the result. However, this option is helpful if *segmentSpeech* cannot be applied reasonably.

Appendix B: Short Instruction and Quick Command Parameter Reference

In the following we will give you a really short instruction on how to analyze audio files with AlignTool, along with a quick reference to the parameters with each command. This is meant as a cheat sheet after you have used AlignTool a few times. Please remember to check the TextGrid file(s) and the workbook as well as the log file regularly.

Short Instruction

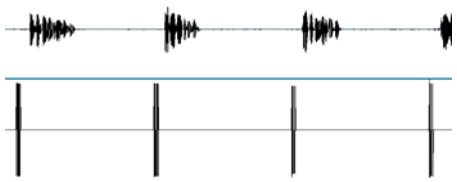
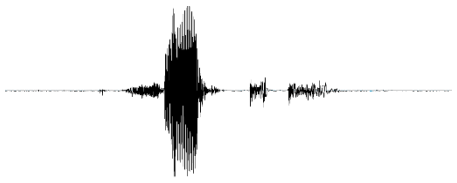
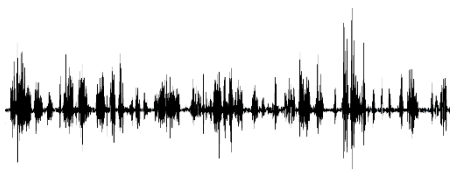
1. Specify workbook

- Edit the workbook.
- Set the file path and name of the workbook using the "Browse" button under "Active Workbook".

2. Specify audio file(s) and set parameters

- Copy the audio files you want to analyse to the "wav" folder in the working directory.
- Set the file path to the "wav" folder in the working directory by clicking on "Browse" under "Active workbook's "batch" sheet initialization".
- Click on "Find *.wav".
- Specify the parameters of the commands in the excel file (for all commands or command-by-command). See below for a quick reference of the command parameters.

3. Segment beeps

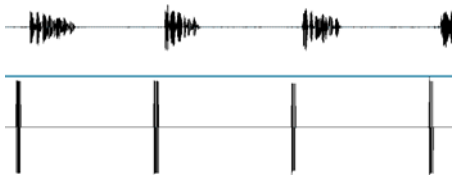
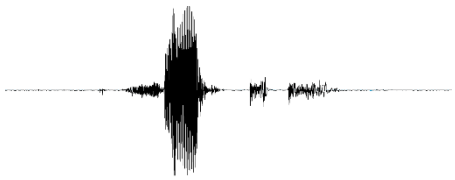
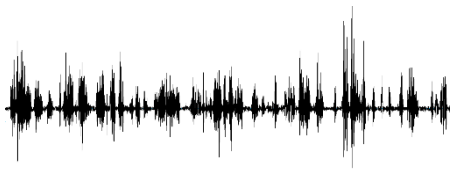
<table>
<tr><td colspan="2">A) Recordings of multiple trials in one audio file</td></tr>
<tr><td>  </td><td>• Click on "segmentBeeps".
• Click on "Import from TextGrids".</td></tr>
<tr><td>B) Trial-by-trial recordings</td><td></td></tr>
<tr><td>  </td><td>• Click on "addTier".
• Click on "Import from TextGrids".</td></tr>
<tr><td colspan="2">C) Recordings of semi-spontaneous speech</td></tr>
<tr><td>  </td><td>• Create a folder named Name_of_the_workbook.tg.
• For each audio file, create a tier called anno.trans and establish intervals of speech of about 30s. In each interval, enter the literal transcription of the participant's utterance. Save the TextGrid file under the same name as the wav file and store it in the folder Name_of_the_workbook.tg.</td></tr>
</table>

	• Next, edit the workbook in order to use *addTier* to copy *anno.trans* to *seg.beep* (-m trim –s anno.trans – d seg.beep). • Click on “addTier”. • Click on “Import from TextGrids”.

4. Segment speech

- Click on “segmentSpeech”.

5. Transcription

A) Recordings of multiple trials in one audio file	
	• Click on “addTier”. • Click on “Import from TextGrids”. • Fill in your transcripts in the workbook. • Click on “Export to TextGrids”.
B) Trial-by-trial recordings	
	• Edit the *addTier* command settings in the workbook so as to copy *seg.beep* to *anno.trans*. • Click on “addTier”. • Fill in your transcripts in the workbook. • Click on “Export to TextGrids”.
C) Recordings of semi-spontaneous speech	
	• You have already completed the transcriptions in Step 3.

6. Align MAUS

- Click on “alignMAUS”.
- Click on “Import from TextGrids”

7. Extract on-/offsets

- Click on “Extract On/Offsets”.

Parameter Overview (with defaults in parentheses)

segmentBeeps

-b <channel>: specifies the beep channel (2)

-r <refbeep>, --refbeep <refbeep>: specifies the name(s) of the reference beep file(s)

-x <0-1>, --mincorrelation <0-1>: specifies the minimum required cross correlation peak beep identification (0)

-s <db>, --silencethreshold <db>: specifies the silence threshold for the beep segmentation (-25)

-m <s>, --minsounding <s>: specifies the minimal duration of the intervals of the audio file that are used for identifying the beeps in the signal (0.180)

--seekflank: enables the seekflank heuristic for post-processing the cross-correlation method (false)

Note: Reference beep file must be mono.

segmentSpeech

-c <channel>: specifies the speech channel (1)

-f <tier>, --filter-tier <tier>: specifies an existing speech interval as a filter for the relevant intervals for *segmentSpeech* to work on

-d, --denoise: enables the Praat denoising functionality (false)

--shiftonsets <s> and *--shiftoffsets <s>:* shifts all detected onsets (offsets) by <s> seconds (0)

--trainbegin <s>: specifies the beginning of the interval for calculating the reference energy (3.3)

--trainwindow <s>: specifies the length of the interval for calculating the reference energy (1)

--speechthresh <f>: specifies the extent to which the signal needs to exceed the average intensity in order to be considered as speech (0.5, i.e. 50%)

--snradd <db>: add <db> to the silence threshold to specify minimal difference in signal intensity to be considered as speech (1)

Note: use -d and -snradd with care.

addTier

-m <mode>, --mode <mode>: specifies the mode (-all, -copy, -trim) of adding a new tier

-s <tier>, --source-tier <tier>: specifies the tier that serves as the source for --mode -copy and -trim

-d <tier>, --dest-tier <tier>: specifies the name of the newly added tier (seg.speech)

-t <text>: specifies the content of the new tier's intervals (speech)

-f <re>, --filter <re>: for --mode copy: specifies that only intervals with content matching <re> is copied (speech)

alignMAUS

-l <language>, --language <language>: specifies the language MAUS is to be used with (deu-DE)

-c <channel>, --speech-channel <channel>: specifies the speech channel (1)

-f <tier>, --filter-tier <tier>: specifies that only speech intervals specified in the filter tier are included (seg.beep)

-s <tier>, --segmentation-tier <tier>: specifies the tier providing an initial speech segmentation (seg.beep)

-d, --denoise: enables the Praat denoising functionality (false)

--initialsilence: enables MAUS to use initial and final silence models (false)

--remote: enables the use of MAUS online services (false)

extractOnOffsets

-f <tier>, --filter-tier <tier>: specifies that only speech intervals specified in the filter tier are included

Appendix C: Trouble Shooting

1. AlignTool cannot access the workbook.

   Save and close the workbook in Excel and run the command again.

2. The documentation states that I should see a new tier in the TextGrid file but I can't see it.

   Please note that loaded TextGrid files are not updated in Praat. During the analysis with AlignTool, the TextGrid file(s) will be edited several times. In order to see the latest version of a TextGrid file, you need to re-open it in Praat (by clicking on "Open", choosing "Read from file", and selecting the TextGrid file).

3. There is lots of feedback in the online log in the Terminal. How can I tell that AlignTool has finished executing a command?

   When you analyse one file, AlignTool will state "finished NAME OF COMMAND", e.g. "finished WavImporter"

   When you analyse more than one wav file, AlignTool will state finished BatchRunner {'batchcmd': 'NAME_OF_COMMAND', 'xlsxfile': NAME_OF_WORKBOOK}, for instance:
   -=INFO=- [14:13:48.132] finished BatchRunner {'batchcmd': 'segmentSpeech', 'xlsxfile': '/home/at/workdir/workbook.xlsx'}

4. My log file is full of warnings of the type "=WARNING=- [… Ignoring row X: Wavefile column empty.

   This warning occurs when a line in the Excel file was filled with text before and the text has been deleted using the backspace button or the delete button. The cells look empty, but there is still a trace of the former entries in the Excel file. To fully delete the contents, select the deleted lines and delete them using the "delete rows" command in Excel.

5. Segment speech is not working. This is the error message: =ERROR=- [14:13:48.132] Batch processing failed in row 2: VAD was unable segment speech. Check silence region: calculated threshold 55.95 db. Speech threshold: 63.90 db.

   There may be various reasons for this error message. One could be that there is lots of background noise. This renders it impossible for AlignTool to estimate the silence threshold in the *segmentSpeech* command. If there is a fixed time frame in the file that is silent, you can specify it in the workbook using the --trainbegin and –trainwindow parameters of *segmentSpeech*. Note, however, that *segmentSpeech* will not be able to distinguish between speech and non-speech sounds, so its output will definitely need additional correction by the user.

   If there is no background noise in your audio file, it is possible that the silent intervals before or after a speaker's utterance are very short. In that case, use the --trainbegin and –trainwindow parameters to adjust the time frame used for establishing the silence threshold (for an example, see the parameters used for analysing the Rastle&Davis corpus in the evaluation of AlignTool; cf. Table 1 in Schillingmann et al., subm.; see Footnote 3).

6. AlignMAUS is not working.

   Check the feedback in the log file – it is likely that the error occurs, because WebMAUS is not accessible (https://clarin.phonetik.uni-muenchen.de/BASWebServices/#!/services). In that case, check your internet connection, both within your Windows system and within the virtual machine.

7. How do I shut down the Virtual Machine when I am finished?

   Close the VMware window or select Player -> Exit. When prompted, choose “PowerOff”.