Inspect_2_3.praat

Task:
This script should enable scanning quickly through (many) files, intervals, or points to inspect them or to make judgements (like e.g. 'Voiced', 'Voiceless'), add comments, or note them down for later investigation.

The Script displays a single or many sound files (and associated TextGrid files, if they exist) as a whole or of specific intervals. The file names can be typed in, listed in a text file, listed in result-file of a previously analysis (e.g. the output of a Formant, Pitch, Spectrum, or Intensity script), or listed in a 'report file' of a previous run of this script. Similarly, interval or point labels can be typed in or listed in a text file. Associated TextGrid files must have the same name as the sound file (i.e., only their extensions differ). If TextGrid files exist, they are always updated by this script, whether boundaries or points are changed or not. The script can be exited at any time and it will continue with those files and intervals/points that had not been inspected during a previous session. With a small modification in the script, additional judgments (like 'voiced', 'voiceless'), notes (position of window and cursor), and/or text comments can be stored in text files.

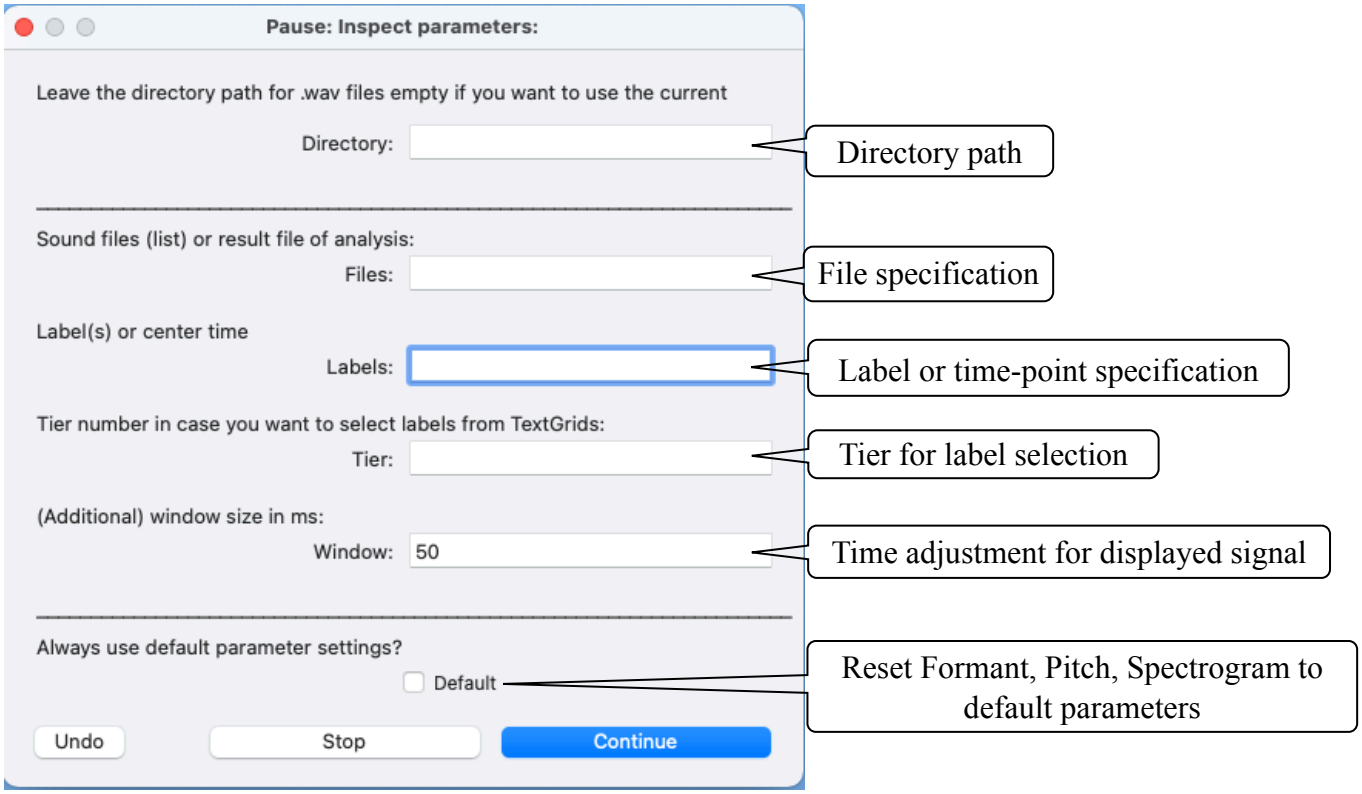
The script creates a file named *000_Inspect_progress_V04.txt* which will be deleted when all specified intervals or points are processed. This file is used in case the user had interrupted a session to continue later with a new session at the last displayed interval without asking for any parameters. Deleting this file will cause the script to begin by asking for parameters and not to continue an interrupted session.

In case comments, judgments, or notes are made, a file named *001_Inspect_reports_<date>_<time>.txt* will be generated that has the comments, judgements and notes stored along with file and timing information. The format of this report file is described on page 9.

Contents:

Intro (this page) 1
Standard parameter window at beginning of the script 2
Example for using a result file 3
Example for handling all sound files in a directory 4
Example for using a list of sound files 5
Example for displaying a single sound file centered at a time point 6
Example of the display of a window in a session 7
Example of the display of a window in a session with notes, judgments and comments 8
Description of the reports file format 9
Description of parameters 10
Behavior during the execution of the script 12
Programming: adapting to different result-file styles 13
Programming: defining judgments options 14
Programming: explanations of some script mechanisms 15
Programming: description of some internal parameters 16
Structure of the progress file 17

Standard parameter window at beginning of the script:

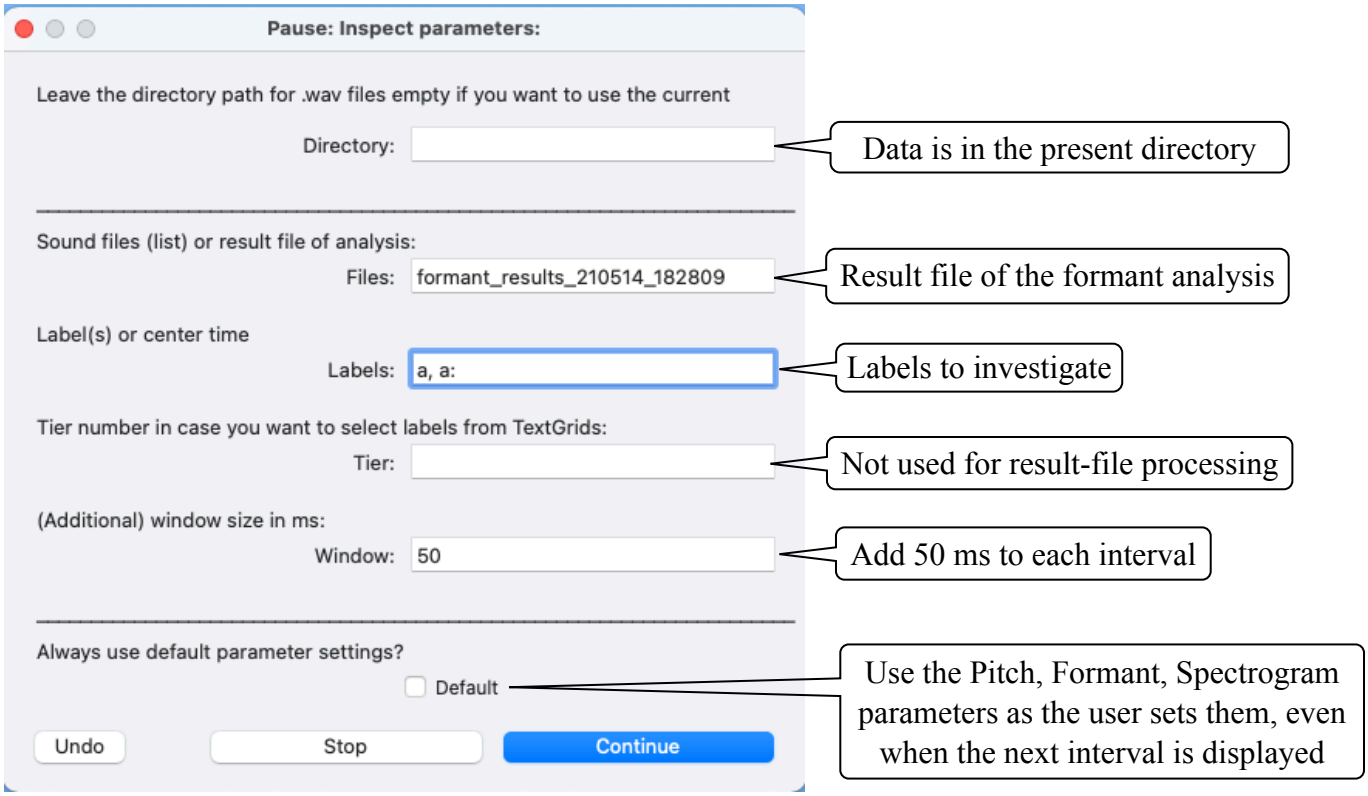


Example for using a result file:

A previous analysis has generated a result file, e.g. *formant_results_210514_182809.txt*:

File	Label	Start(s)	Duration(ms)	Pitch_mean(Hz)	F1_mean(Hz)	F2_mean(Hz)
g071a000	u:	1.0139	31.6	127.59	309.6	1104.6
g071a000	a	1.1676	60.3	134.61	552	1434.8
g071a000	@	1.3639	29.3	128.69	355.5	1562.6
g071a000	a	1.5871	53.9	144.85	442.2	1307.2
g071a000	I	1.7069	25.3	139.68	292.5	2158
g071a000	a	1.9855	28.8	134.35	496.3	1113.3
g071a000	a:	2.1306	88.6	117.62	733.9	1255.8
g071a000	a	2.2194	111.7	118.01	714.6	1366.5
g071a000	U	2.4297	111.4	176.69	360.2	1358.6
g071a000	O	2.7113	47.1	148.77	468.3	1068.3
g071a000	i:	2.8991	18.9	0	NA	NA
g071a000	a:	2.9907	57.9	143.93	478.2	1212.9

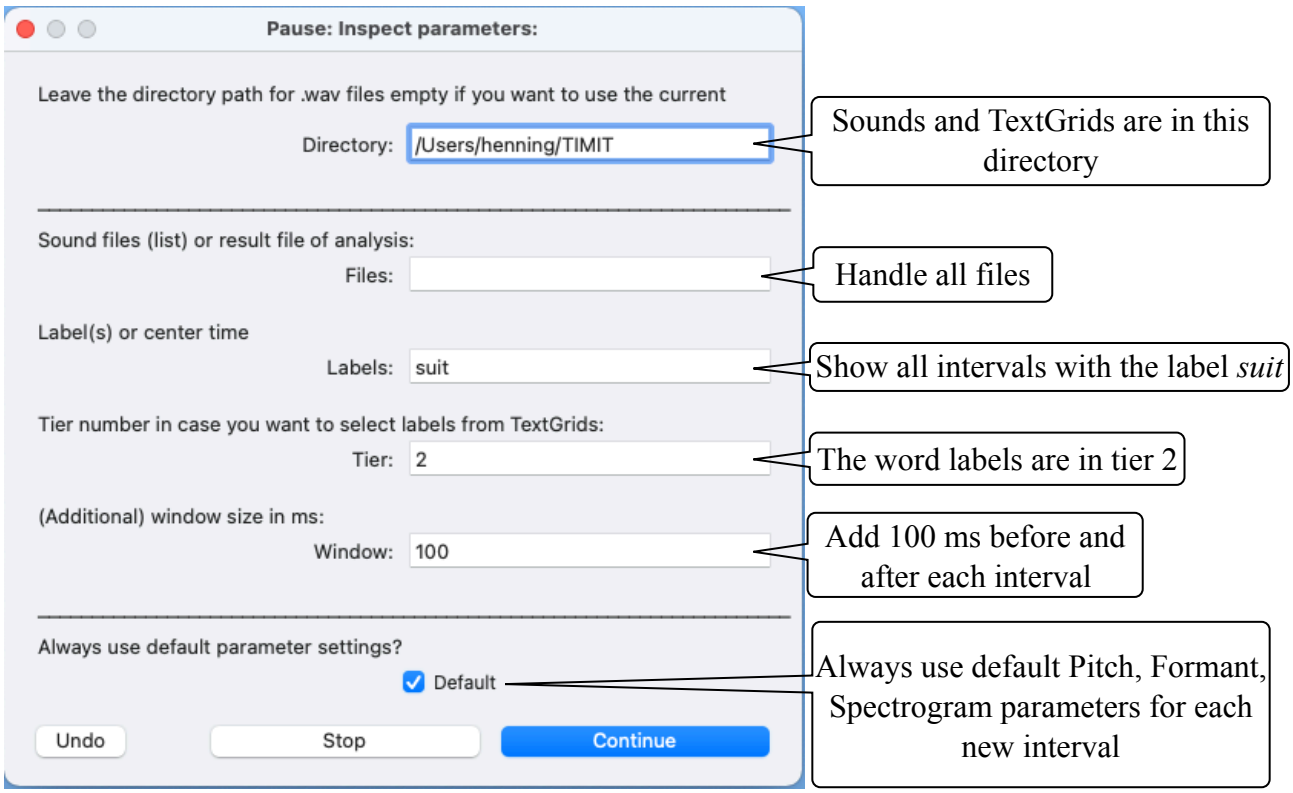
The user wants to examine all labels [a, a:] and wants to add 50 ms on each side of the intervals:



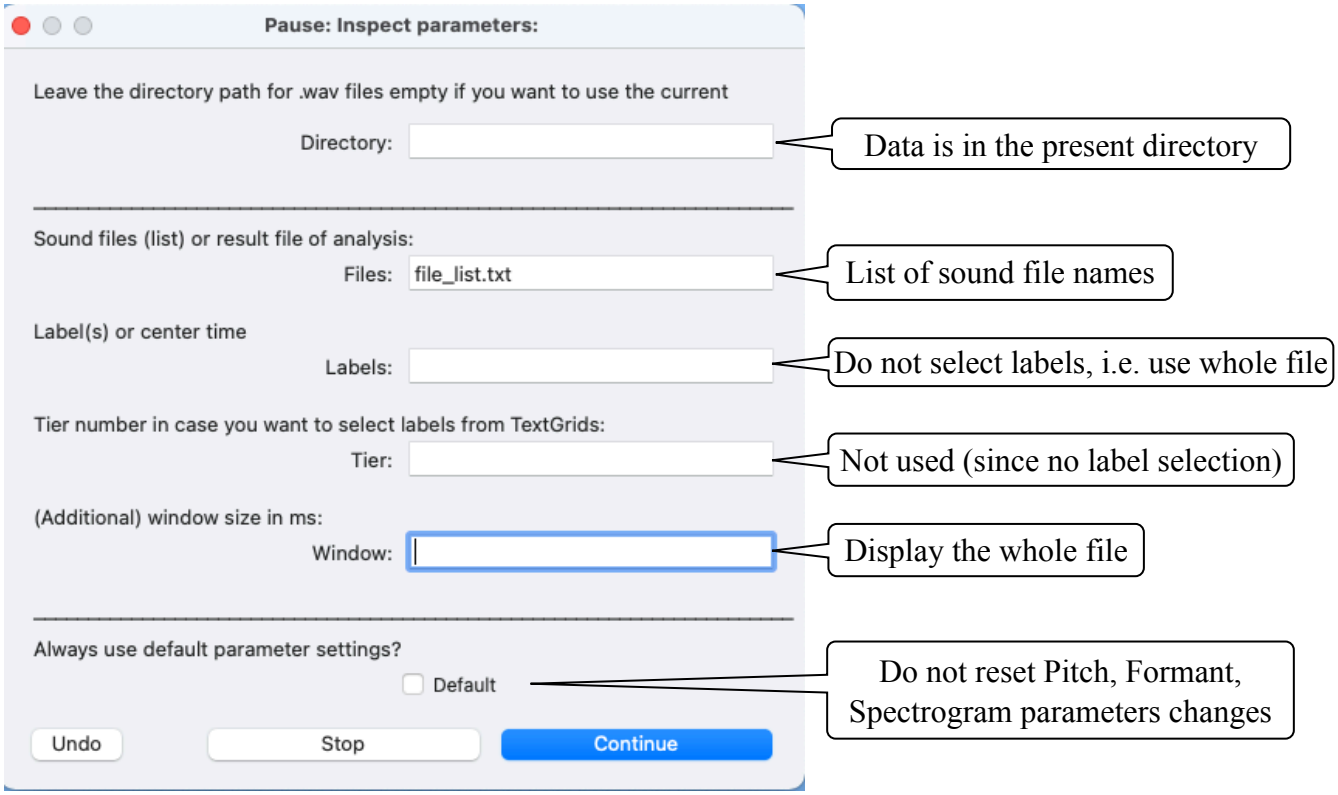
Example for handling all sound files in a directory:
A user wants to inspect all intervals with the label ‘suit’ in tier 2 of all sound files in a directory and

g071a000
g071a001
g071a002
g071a003
g071a004
g071a005
g071a006
g071a007
g071a008
g071a009
g071a010

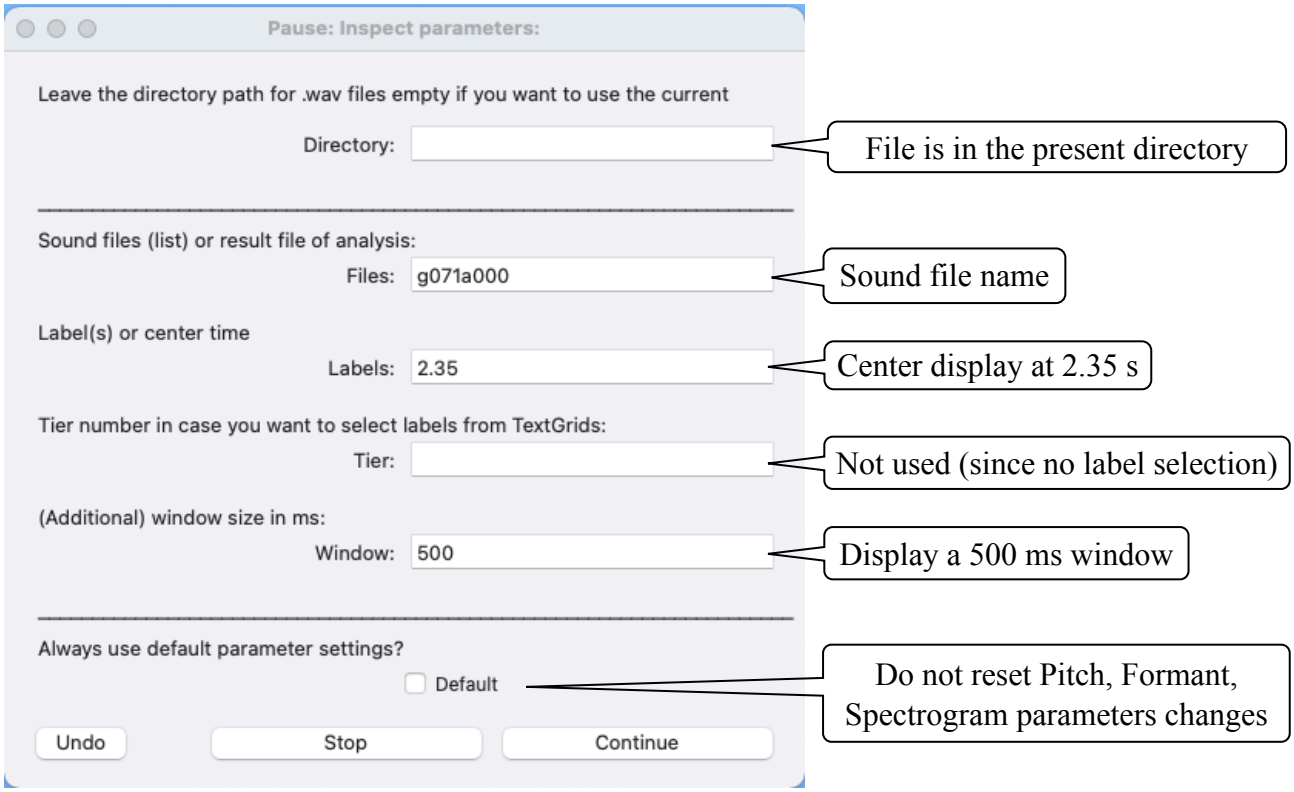
wants to add 0.1 second to both ends of the intervals. (S)he wants to make sure that subsequent displays use the default parameters in case (s)he changes parameter of the spectrogram when inspecting one interval:



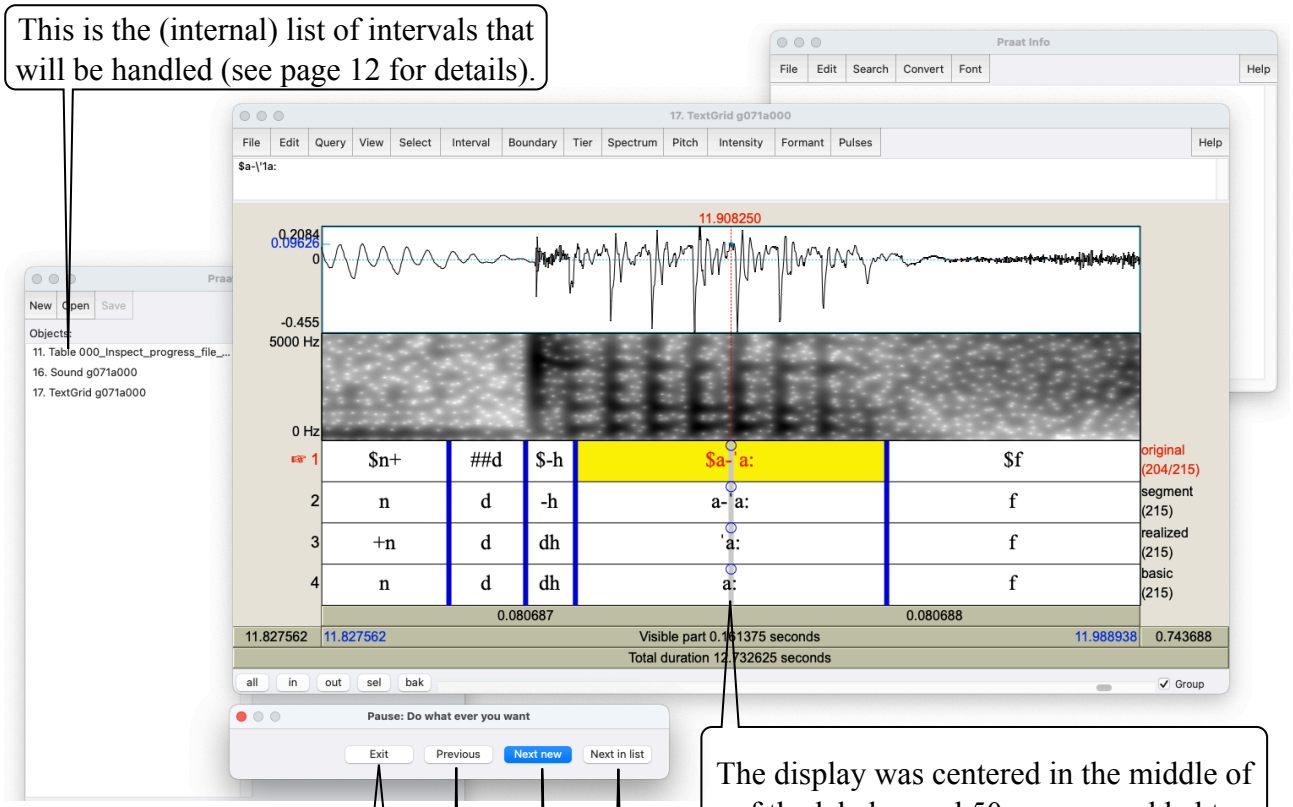
Example for using a list of sound files:
A user wants to display a list of sound files, e.g. given in the file *file_list.txt*:



Example for displaying a single sound file centered at a time point:
The user wants to display a 500 ms window of the file g071a000 around 2.35 s:



Example of the display of a window in a session:
The user had requested to show all files with a segment [a:] in tier 4 and to show additional 50 ms before and after each segment. (S)he has already inspected several windows and went back two windows to re-check something:



This is the (internal) list of intervals that will be handled (see page 12 for details).

The display was centered in the middle of of the label a: and 50 ms were added to both sides of the window. The user is free now to do normal PRAAT operations.

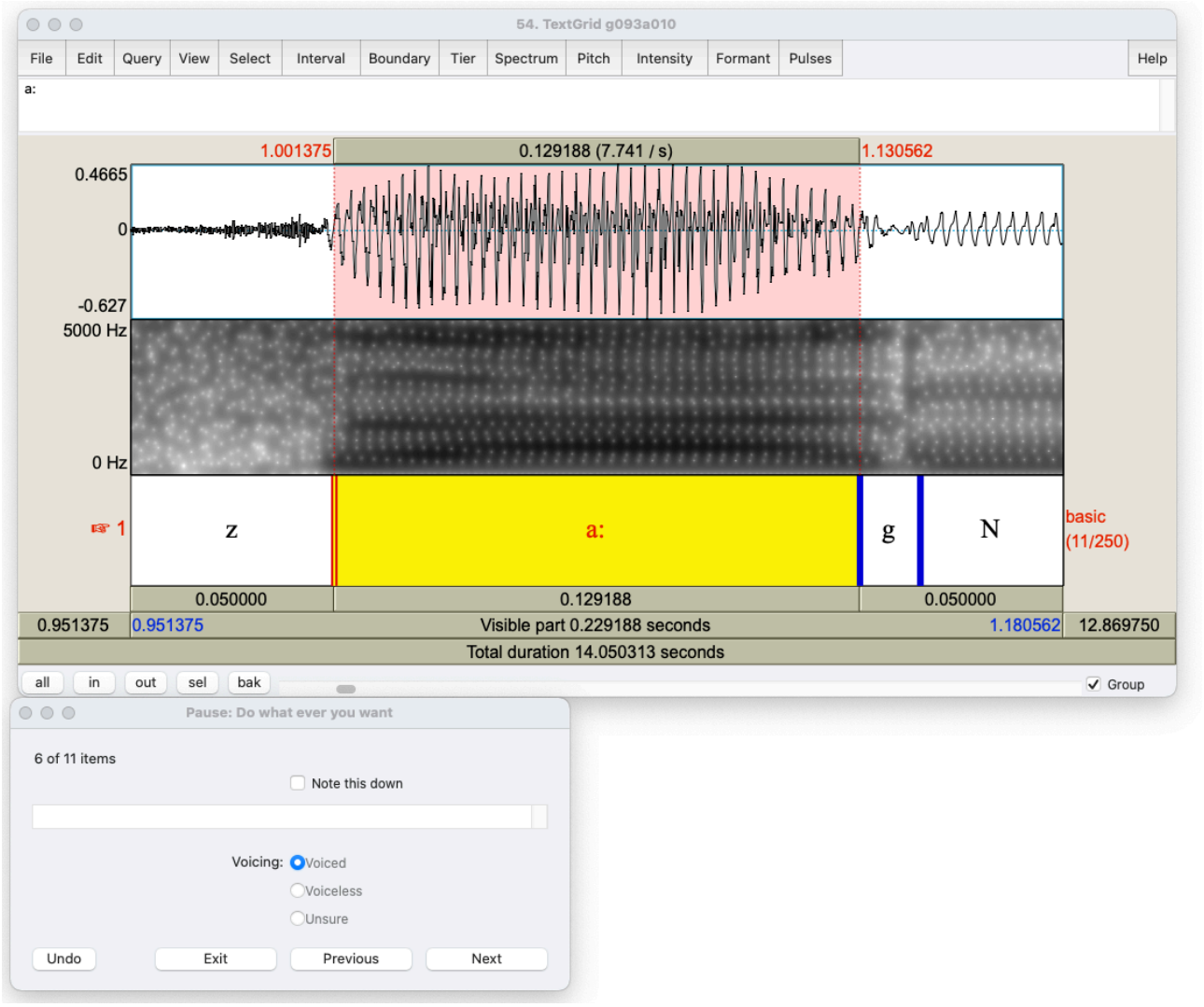
Clicking on *Exit* will leave the script (without updating the TextGrid). The script will start the next time it runs with the last non-inspected segment.

Clicking on *Next in list* will move the window to the next a: in the list of labels to inspect, even if that has been inspected (this will update the TextGrid).

Clicking on *Previous* will move the window to the previous a: in the list of labels to inspect (this will update the TextGrid).

Clicking on *Next new* will move the window to the next a: that has not been inspected yet (this will update the TextGrid).

Example of the display of a window in a session with notes, judgments and comments:
The user had requested to show all files with a segment [z] in tier 4 and to show additional 50 ms before and after each segment. The script will accept text comments in the 'Pause' window and a judgment can be placed whether the inspected interval is voiced or voiceless. Additionally a comment can be written and the segment can be 'noted', i.e., a flag is set to indicate that this segment might be of special interest. All these informations are stored in a text file with the name "001_Inspect_<date>_<time>_reports.txt". The format of this file is described on page 10. Furthermore, this file (or an edited version of it) can be used as an input for the inspect script. (To configure the judgments option see page 14.)



Description of the reports file format:

The report file has the same format as the progress file (see page 17). I describe here only the essential parts to extract the notes, comments and judgments.

Stat	File	Start(s)	End(s)	Cursor(s	Label	Note	Judgmen	Comment
99	.	7	0	4	?	?	?	?
99	.	0	3	1	001_Inspect_reports_20211022_184017.t	?	?	?
99	Voicing	?	?	?	?	?	?	?
99	Voiced	?	?	?	?	?	?	?
99	Voiceless	?	?	?	?	?	?	?
99	Unsure	?	?	?	?	?	?	?
1	g071a000.wa	1.2166250	1.3979375	1.3072812	d	-1	Voiceless	?
1	g071a000.wa	3.4581250	3.5936875	3.5259062	d	-1	Voiced	?
1	g071a000.wa	11.648500	11.786812	11.717656	d	-1	Unsure	No closure
1	g071a000.wa	11.802625	11.91775	11.860187	d	-1	Voiced	?
1	g093a010.wa	1.3034375	1.423	1.3632187	d	-1	Unsure	No closure
1	g093a010.wa	10.229	10.34975	10.289375	d	-1	Unsure	?
1	g093a010.wa	10.378812	10.489187	10.434	d	-1	Unsure	no closure
1	g093a010.wa	11.265625	11.397	11.331312	d	-1	Voiceless	?
1	g093a010.wa	11.601000	11.766	11.683500	d	-1	Voiceless	Boundaries are

Rows with *state = 99* are used by the script.

Rows with *state = 0* are not inspected yet (and are not part of the results file, which is only generated after all intervals/points have been inspected.

Rows with s*tate = 1*:

Note = -1 Note feature is not used
Note = 0 Interval/point has not been marked
Note = -1 Interval/point has been marked

Judgment = ? Judgment feature is not used
Judgment = <string> Selected judgment

Comment = ? Comment feature is not used or no comment written
Comment = <string> Comment typed by the user

The results file can be used as input file of *inspect.praat* and the script will show all windows where the either the Note, Judgment or Comment is not ?.

If the file is edited (e.g. with a spreadsheet program to select only rows with *Note = 1* or with a certain *Judgment* or any *Comment*) it must be supplied with the header rows (i.e., those with *state = 99* in the same sequence as in the original file) with <tab> separated columns.

Description of parameters:

Directory:
The script handles all sound and TextGrid files in a directory. The path of this directory can be specified in this field. If this field is left empty, the script will handle all sound files in the directory where the script was started (i.e., the script is placed in the same directory as the sound and TextGrid files). The user can specify different directories for sound, TextGrid, support, and result files by changing variables in the script (see page 16). (Search for "### 1 >>>" and "### 4 >>>" in the script for directory specifications.)

Files:
The script can handle different set of files depending on the input given in this field:

<empty>: All sound files in a directory will be handled.

<file_name>.txt: The action depends on the type of file:

- File has only one column: each line will be used as name for a sound file.
- File has a result-file header: information from result file will be taken (see page 13 for details).
- File has a reports-file header: information from reports file will be used (see page 17 for details).

<file_name>: A sound file with the name <file_name> will be handled.

Labels:
The user can specify labels of intervals and points that should be displayed. In case the user has specified a single file or a list of files, the field **Tier** must also be specified. In case a previous result-file is used, the result-file must have a column for labels and **Tier** must not be specified. Labels can be specified in several ways:

<empty>: The display depends on the specification of the **Files** and **Window** fields:

- Files specified and Windows empty: displays the whole file
- Files and Window specified: displays the first milliseconds as specified by Window
- Result-file specifies intervals: display all intervals in the result file and add the ms given in Window before and after the interval
- Result-file specifies points: display all points in the result file and use the ms given in Window as window size

<one or more labels separated by spaces>: Search for any of the labels in the specified sound and Textgrid files and display intervals with time added given by Window or points and use Window as the window size to display.

<label_list_file.txt:>: Search for any of the labels listed on a line-by-line basis in a text file (which must end with .txt) in the specified sound and TextGrid files and display intervals with time added given by Window or points and use Window as the window size to display.

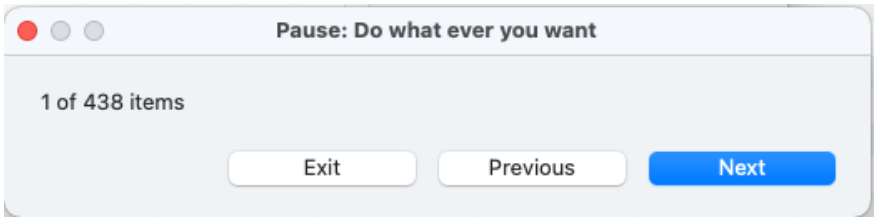
Tier:
When labels are specified and sound files are specified (either by leaving the **Files** field empty, specifying files there or giving a list of files) **Tier** must be specified (otherwise, the script does not know where to find the labels, or same labels might be on different tiers) unless the TextGrid files have only one tier. In case a result-file is specified in **Files,** or no **labels** are defined, the **Tier** field will be ignored.

Window:
A time in milliseconds. If intervals are defined, **Window** milliseconds are added before and after each interval. I points are defined, **Window** defines the size of the displayed window around the point location. If whole files are to be displayed (i.e., **Labels** is left empty) **Window** defines the size of the displayed window; or the whole file is displayed in this case, if the **Window** field is left empty.

Always use default parameter settings:
During the inspection of files, the user might interactively change Formant, Spectrogram or Pitch parameters. If this flag is not set, the script will continue with the changed parameters. If this flag is set, for subsequent files these parameters are reset to

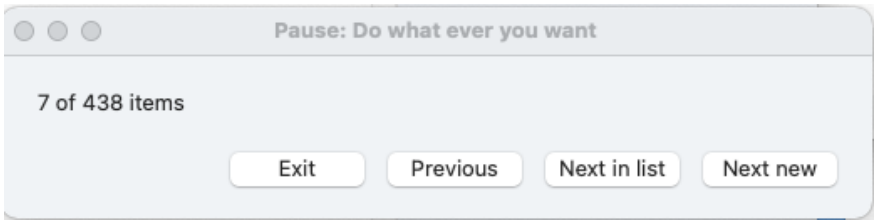
```
Spectrogram settings: 0, 5000, 0.005, 70
Pitch settings: 75, 500, "Hertz", "autocorrelation", "automatic"
Formant settings: 5000, 5, 0.025, 30, 1
```

Behavior during the execution of the script:
The script display the waveforms along with the selected windows (spectrogram, pitch, formants, intensity as selected by the user) and pauses with a 'Pause' window. After inspection of the file, which can include changing any Tier of the TextGrids the user can click on 'Next' in the 'Pause' window to inspect the next item as given by the list file. The script keeps tract of inspected items in the file *000_Inspect_progress_V04.txt* by changing the 'state' value from '0' to '1' for the displayed interval or point.



Behavior when using the 'Exit' option:
'Exit' will stop the script without updating the TextGrid of the displayed window. The user can continue in a later session and the script will continue with those items in the list file that have not been already inspected. (In case a user changed boundaries of labels in a Tier but does not want to save them, (s)he can use 'Exit' to avoid updating the TextGrid and start the session again – it will continue with the last displayed interval.)

Behavior when using the 'Previous' option:
'Previous' will go back to the previous item in the list file (and update the currently displayed TextGrid). If more than one 'Previous' is used (i.e., the user went back more than one item, e.g. from the 10th item to the 7th item) the user gets two 'Next' options:



Choosing the 'Next in list' will move to the next item in the list (the 8th item in the example).
Choosing the 'Next new' will jump to the next untreated item (i.e. the 10th item in the example).

Cancelling a session:
As long as the script finds the file *000_Inspect_progress_V04.txt* it will continue with the first non-inspected item. In case a user does not want to continue with a session and inspect other files, (s)he can simply delete the file *000_Inspect_progress_V04.txt* (or rename it temporarily, in case (s)he wants to work on this session later again).

Programming: adapting to different result-file styles:
In case a result-file (i.e. output of a previously ran analysis in a tab-delimited text file) exists, the script looks for a one-line header with certain keywords. The script needs at least a sound file name and a time (start time of an interval or time point). For intervals, additionally a duration or an end time must be specified. Furthermore, a label (interval or point) can be present in the result-file listing. For labels, also the tier number must be specified.

The script maps the words in the header of a result-file to the internal (column-)variables with a mechanism described here. The script uses for its internal representation the strings *file_string$, label_string$, start_string$, duration_string$,* and *end_string$*. The words in the result-file are mapped onto these by assigning a string (e.g., if the result-file has a header word *Recording* to specify a file, the assignment *file_string$ = "Recording"* binds the column *Recording* of the result-file to the internal *file_string$*). Additionally, it can be specified whether duration is given in seconds or milliseconds (start and end times are always seconds). By changing the assignments in the scripts, the mapping from the keywords of the header in the result-file can be changed (search for "### 3 >>>" in the script for this code):

```
file_string$ = "File"
label_string$ = "Label"
start_string$ = "Start(s)"
duration_string$ = "Duration(ms)"
end_string$ = ""
# duration in the result file is given in in milliseconds or seconds
#      duration_is_ms = 0  seconds
#      duration_is_ms = 1  milliseconds
duration_is_ms = 1
```

The script copies this mapping into an array *result_header_<i>$* to be later able to search for these words in the result-file header:

```
# make a tab-delimited string of the header items
result_header_items$ = file_string$ + tab$ + label_string$ + tab$ + start_string$ + tab$
                                                        + duration_string$ + end_string$
# use this string to put the words on a line-by-line basis
Create Strings from tokens: "help", result_header_items$, tab$
max_result_header = Get number of strings
# now copy the words into an array that will be used later for searching the header items
for i_col to max_result_header
       result_header_'i_col'$ = Get string: i_col
endfor
```

Later in the script (when a result-file is actually used) this array is used to build a hash to point to the specific columns:

```
for i_col to max_result_header
       header_col = Get column index: result_header_'i_col'$
       if (header_col)
              hash[result_header_'i_col'$] = header_col
       else
              . . .
       endif
endfor
```

Eventually, the hash will point to the specific column in the result-file. E.g., if a key word *Recording* in column 3 of the header of the result-file will point to the file name, the script will use

```
hash[file_string$] = 3
```

to access the filename (because *result_header_1$* will be *Recording* in this example).

Programming: defining judgment options:
The script can provide choice options in the "beginPause…endPause" block. To swutch this option on, the variable "judgments_flag" must be set to "1' (see ### 2 >>>" in the script). The definition of the name for the judgment "Voicing" and the options for this choice (e.g. "voiced", "voiceless", "unsure") can be specified with the variables "judgment_0$", "judgment_1$", "judgment_2$", and "judgment_3$". I.e., the pause window will display for this example:

```
Voicing: o voiced
         o voiceless
         o unsure
```

These variables, and the variable "nr_judgments" (for the number of judgments) are defined at the beginning of the script (search for "### 2 >>>" in the script):

```
judgment_0$ = "Voicing"
judgment_1$ = "Voiced"
judgment_2$ = "Voiceless"
judgment_3$ = "Unsure"
nr_judgments = 3
```

The code in the "beginPause…endPause" block looks like:

```
if (judgments_flag)
  choice: judgment_0$, 1
    for i to nr_judgments
      option: judgment_'i'$
    endfor
endif
```

The conversion of the chosen option into a string that is eventually stored in the file *001_Inspect_<date>_<time>_judgments.txt* is performed by the code (search for "### 2 >>>" in the script – the comments are here more extended):

```
if (judgments_flag)
# make sure the first letter of the string judgment_0$ is lower case
# and convert all non-alphanumeric symbols to underline
# because it will be used as a variable name
# In our example:
#   judgment_0$ contains the string "Voicing"
#   the first replace_regex puts the string "voicing" into variable_name$
#   the second replace_regex does not change anything
  variable_name$ = replace_regex$ (judgment_0$, "^.", "\L&", 1)
  variable_name$ = replace_regex$ (variable_name$, "\W", "_", 0)

# get the value which is stored in the variable with the name variable_name$ -
# this is the number of the selected option
# In our example the next line would be interpreted by praat as
#   selected_option = 'voicing'
# and if the second option was selected, the value in selected_option will be 2
  selected_option = 'variable_name$'

# now store this information
# selected_option has the value 2 in our example, and
# judgment_'selected_option'$ will become judgment_2$, which was defined
# at the beginning of the script as "Voiceless"
  appendFileLine: judgments_file$,base_name$,tab$,label$,tab$,tier,tab$,
                                  start$,tab$,judgment_'selected_option'$
endif
```

Programming: explanations of some script mechanisms:
The script has essentially three parts:
1) Definition of some parameters (and filling result-file header keyword mapping, see page 10)
2) Inquiring information from the user in case a new session starts and converting this information into a *progress_table* or loading an exiting *progress_table* from the file *000_Inpsect_progress_V04.txt*.
3) Interactively going through the *progress_table*.

The *progress_table* stores only a flag whether a particular interval has been displayed or not, the name of the sound file (usually only without full path and without extension), start of this interval, the end of this interval, and a time where the cursor should be displayed. Note that 'interval' can be a whole file, the interval of a label (eventually with added time before and after), or the window around a point. Information about the original label (interval or point) is not present in the *progress_table* (labels will show up in the associated TextGrid when that stretch of time is displayed). All this information is generated during the second part of the script, when the information given by the user is evaluated and converted into each row of the table.

The first 7 rows of the *progress_table* are used to store some global parameters that the user specified in the second part of the script (or had defined in the script before calling it the first time for a session) or that are defined in the first section of the script:

```
state   file    start   end     cursor
2       ?       5       ?       ?                                               # first row of real data
3       ?       0       ?       ?                                               # default switch
4       /Users/henning/CD/TIMIT/PRAAT/Male/   ?       ?       ?               # sound directory
5       /Users/henning/CD/TIMIT/PRAAT/Male/   ?       ?       ?               # TextGrid directory
6       ?       4       ?       ?                                               # tier
7       ./001_Inspect_210602_182406_comments.txt     ?       ?       ?       # comments file
8       ./001_Inspect_210602_182406_judgments.txt    ?       ?       ?       # judgments file
```

to be continued…

Programming: description of some internal parameters:

Directories:
The script uses internally separate strings for sound, TextGrid, result-file, and support files. Users who use separate directories for these directories can adjust these names in the script (or put them into PRAATs 'form' window).

support_directory$ = ""
<directory$ defined by user interaction>
sound_directory$ = directory$
grid_directory$ = directory$
result_directory$ = directory$

Sound file extension:
sound_ext$: The default extension for sound files is ".wav".

Default parameters:
The default parameters can be changed at the appropriate place in the script and other default parameters can be added there.

Current version and date:
2.3, 29-oct-2021

Known problems:
This documentation needs polishing and extension.
The handling of grid_directory$, result_directory$, and support_directory$ needs improvement if a complete path for the sound_directory is defined and TextGrids etc. are located in different directories.

Planned extension:
Search for labels in more than one tier
Use a text file to load/store all parameters.

Contact:
henning.reetz@icloud.com

Structure of the progress file

Inspect.praat copies interval (or point)[1] information given by the user into a ‘progress file’ which is then used in one or more sessions to display intervals a user has specified. The progress file also stores notes, judgments and comments given by the user. Notes are simple flags (‘note this interval down or not’); judgments are user-specified options (e.g., Voiced/Unvoiced/Unsure); comments are free texts.

The file is a copy of a table in Praat that is generated at the first call of *Inspect.praat* with parameters from the script. The file is updated after each interval that is inspected (to save all data in case Praat crashes) and re-loaded automatically if the script is called again, in case inspect.praat was exited before all intervals are inspected.

The progress file (Version V04) is a tab-delimited UTF-8 text file with a header line. The first lines are used to store some global information that is needed by the script after a re-start (i.e., the user had exited the script without handling all intervals and continues at a later time). The actual interval data starts at line <first_row>. The first lines look like:

State	File	Start(s)	End(s)	Cursor(s)	Label	Note	Judgment	Comment
99	.	<first_row>	<def>	<tier>	<s_dir>		<g_dir>	<r-dir>
99	.	<n-flag>	<nr-j>	<c-flag>	<p-file>			
99	<j-name>							
99	<j-opt 1>							
99	<j-opt 2>							
99	• • •							
99	<j-opt nr-j>							
<st>	<file>	<on>	<off>	<cursor>	<label>	<n>	<j>	<c>

[1] I will use only ‘interval’ in this text, referring to intervals, points or whole files.

Column data:
(if state = 99: data is used to store initial parameters described below;
if state = 0 or 1: data described here)

State = 0	Interval has not been inspected
State = 1	Interval has been inspected
State = 99	Storage of parameters for script
File	Sound file name with extension
Start(s)	Left window edge in seconds
End(s)	Right window edge in seconds
Cursor(s)	Cursor position in seconds
Label	Label of segment to be investigated (or empty)
Note = -1	Note not set yet (state = 0) or Note feature not used
Note = 0	Interval not marked (state = 1)
Note = 1	Interval marked (state = 1)
Judgment = ?	Judgment not set yet (state = 0) or Judgment feature not used
Judgment = *string*	Judgment chosen by user (state = 1)
Comment = ?	No Comment given or Comment feature not used
Comment = *string*	Comment written by user (state = 1)

Row data:

<first row>	Number of the first row of an interval to be inspected.
<def> = 0	Do not reset pitch, spectrum and formant parameters after each interval inspection
<def> = 1	Reset pitch, spectrum and formant parameters after each interval inspection
<s-dir>	Directory (full path) for sound files (can be "./" for local directory)
<g-dir>	Directory (full path) for TextGrid files (can be "./" for local directory)
<r-dir>	Directory (full path) for result file (can be "./" for local directory) The 'result file' will contain notes, judgments and comments.
<tier>	Tier to be used for labels
<n-flag> = 0	Do not note intervals
<n-flag> = 1	Note intervals
<c-flag> = 0	Do not take comments for intervals
<c-flag> = 1	Allow comments for intervals
<p-file>	Name of the progress-file (in the support directory)
<nr-j> = 0	There are no judgment options
<nr-j> = 1	There are judgments to be made
<j-opt *i*>	*string* for option *i* (if nr-j-opt <> 0, otherwise omitted)
<file>	Sound file name (with path and extension)
<on>	Left edge of window (in seconds)
<off>	Right edge of window (in seconds)
<cursor>	Cursor position (in seconds)
<label>	Label name (if given)
<n>= -1	Note not set (yet)
<n> = 0	Interval not noted/marked
<n> = 1	Interval noted/marked
<j>= .	Judgment not selected (yet)
<j> = *string*	Judgment
<c>= .	Comment not written (yet)
<c> = *string*	Comment

Progress file *state=00* rows:
The first rows of the progress table (which is then copied into the progress file) are filled with the data defined in the script, for example:

State	File	Start(s)	End(s)	Cursor(s)	Label	Note	Judgment	Comment
99	.	7	0	4	.		.	../result
99	.	1	3	1				
99	Voicing							
99	Voiced							
99	Unvoiced							
99	Unsure							
<st>	<file>	<on>	<off>	<cursor>	<label>	<n>	<j>	<c>

Interpretation:

row	column	variable	value	meaning
1	wstart	<first_row>	7	The first interval data is in row 7 of the table/file
1	wend	<def>	0	Pitch, spectrum and formant parameters should not be reset to default parameters for each interval
1	cursor	<tier>	4	Tier for the labels
1	label	<s-dir>	.	Path for sound files
1	judgment	<g-dir>	.	Path for TextGrid files
1	comment	<r-dir>	../result/	Path for result files
2	wstart	<n-flag>	1	Enable note/marking intervals
2	wend	<c-flag>	1	Enable commenting for intervals
2	cursor	<nr-j>	3	Show 3 judgment options
3	file	<j-name>	Voicing	Name for judgment
4	file	<j-opt 1>	Voiced	Judgment option 1
5	file	<j-opt 2>	Unvoiced	Judgment option 2
6	file	<j-opt 3>	Unsure	Judgment option 3

The progress file only stores the left (start, beginning, <on>) and right (stop, end, <off>) edges of windows for sound files (<file>), the initial cursor (<cursor>)position, and labels (<label>, if it was specified). <n>, <j> and <c> are initialized with a dot (.).