# Contents

1 Examples
   1.1 Transformer Example ............................................. 1
   1.2 Combiner Example ................................................ 1

2 API Reference
   2.1 Transformers ..................................................... 3
   2.2 Combiners .......................................................... 12
   2.3 File info ............................................................ 12
   2.4 Core functionality ................................................ 14

3 Changes
   3.1 Changes ............................................................ 17
      3.1.1 v0.1 ............................................................ 17

4 Contribute ............................................................. 19

5 Indices and tables .................................................... 21

Python Module Index .................................................. 23
Transformer Example

```python
import sox
# create transformer
tfm = sox.Transformer('path/to/input_audio.wav', 'path/to/output/audio.aiff')
# trim the audio between 5 and 10.5 seconds.
tfm.trim(5, 10.5)
# apply compression
tfm.compand()
# apply a fade in and fade out
tfm.fade(fade_in_len=1.0, fade_out_len=0.5)
# create the output file.
tfm.build()
# see the applied effects
tfm.effects_log
> ['trim', 'compand', 'fade']
```

Combiner Example

The previous example operates on monophonic signals, but resampy also supports stereo resampling, as demonstrated below.

```python
import sox
# create combiner
cbn = sox.Combiner(
    ['input1.wav', 'input2.wav', 'input3.wav'], output.wav, 'concatenate'
)
# pitch shift combined audio up 3 semitones
cbn.pitch(3.0)
# convert output to 8000 Hz stereo
cbn.convert(samplerate=8000, channels=2)
```
# create the output file

cbn.build()
Transformers

Python wrapper around the SoX library. This module requires that SoX is installed.

class sox.transform.Transformer(input_filepath, output_filepath)

Audio file transformer. Class which allows multiple effects to be chained to create an output file, saved to output_filepath.

Parameters

- **input_filepath**: str
  Path to input audio file.

- **output_filepath**: str
  Path to desired output file. If a file already exists at the given path, the file will be overwritten.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input_filepath</td>
<td>Path to input audio file.</td>
</tr>
<tr>
<td>output_filepath</td>
<td>Path where the output file will be written.</td>
</tr>
<tr>
<td>input_format</td>
<td>(list of str) Input file format arguments that will be passed to SoX.</td>
</tr>
<tr>
<td>output_format</td>
<td>(list of str) Output file format arguments that will be passed to SoX.</td>
</tr>
<tr>
<td>effects</td>
<td>(list of str) Effects arguments that will be passed to SoX.</td>
</tr>
<tr>
<td>effects_log</td>
<td>(list of str) Ordered sequence of effects applied.</td>
</tr>
<tr>
<td>globals</td>
<td>(list of str) Global arguments that will be passed to SoX.</td>
</tr>
</tbody>
</table>

Methods

**allpass** *(frequency, width_q=2.0)*

Apply a two-pole all-pass filter. An all-pass filter changes the audio’s frequency to phase relationship.
without changing its frequency to amplitude relationship. The filter is described in detail in at http://
musicdsp.org/files/Audio-EQ-Cookbook.txt

Parameters frequency : float

The filter’s center frequency in Hz.

width_q : float, default=2.0

The filter’s width as a Q-factor.

See also:

equalizer, highpass, lowpass, sinc

bandpass (frequency, width_q=2.0, constant_skirt=False)

Apply a two-pole Butterworth band-pass filter with the given central frequency, and (3dB-point) band-
width. The filter rolls off at 6dB per octave (20dB per decade) and is described in detail in http://musicdsp.
org/files/Audio-EQ-Cookbook.txt

Parameters frequency : float

The filter’s center frequency in Hz.

width_q : float, default=2.0

The filter’s width as a Q-factor.

custom_skirt : bool, default=False

If True, selects constant skirt gain (peak gain = width_q). If False, selects constant 0dB

See also:

bandreject, sinc

bandreject (frequency, width_q=2.0)

Apply a two-pole Butterworth band-reject filter with the given central frequency, and (3dB-point) band-
width. The filter rolls off at 6dB per octave (20dB per decade) and is described in detail in http://musicdsp.
org/files/Audio-EQ-Cookbook.txt

Parameters frequency : float

The filter’s center frequency in Hz.

width_q : float, default=2.0

The filter’s width as a Q-factor.

custom_skirt : bool, default=False

If True, selects constant skirt gain (peak gain = width_q). If False, selects constant 0dB

See also:

bandreject, sinc

bass (gain_db, frequency=100.0, slope=0.5)

Boost or cut the bass (lower) frequencies of the audio using a two-pole shelving filter with a response
similar to that of a standard hi-fi’s tone-controls. This is also known as shelving equalisation.

The filters are described in detail in http://musicdsp.org/files/Audio-EQ-Cookbook.txt

Parameters gain_db : float

The gain at 0 Hz. For a large cut use -20, for a large boost use 20.
**frequency**: float, default=100.0

The filter’s cutoff frequency in Hz.

**slope**: float, default=0.5

The steepness of the filter’s shelf transition. For a gentle slope use 0.3, and use 1.0 for a steep slope.

See also:

*treble, equalizer*

**build()**

Builds the output file by executing the current set of commands.

**compand** *(attack_time=0.3, decay_time=0.8, soft_knee_db=6.0, tf_points=[(-70, -70), (-60, -20), (0, 0)])*

Compand (compress or expand) the dynamic range of the audio.

**Parameters**

**attack_time**: float, default=0.3

The time in seconds over which the instantaneous level of the input signal is averaged to determine increases in volume.

**decay_time**: float, default=0.8

The time in seconds over which the instantaneous level of the input signal is averaged to determine decreases in volume.

**soft_knee_db**: float or None, default=6.0

The amount (in dB) for which the points at where adjacent line segments on the transfer function meet will be rounded. If None, no soft_knee is applied.

**tf_points**: list of tuples

Transfer function points as a list of tuples corresponding to points in (dB, dB) defining the compander’s transfer function.

See also:

*mcompand, contrast*

**convert**(samplerate=None, channels=None, bitdepth=None)**

Converts output audio to the specified format.

**Parameters**

**samplerate**: float, default=None

Desired samplerate. If None, defaults to the same as input.

**channels**: int, default=None

Desired channels. If None, defaults to the same as input.

**bitdepth**: int, default=None

Desired bitdepth. If None, defaults to the same as input.

See also:

*rate*

**equalizer**(frequency, width_q, gain_db)**

Apply a two-pole peaking equalisation (EQ) filter to boost or reduce around a given frequency. This effect can be applied multiple times to produce complex EQ curves.

**Parameters**

**frequency**: float

2.1. Transformers
The filter’s central frequency in Hz.

\textbf{width\_q} : float

The filter’s width as a Q-factor.

\textbf{gain\_db} : float

The filter’s gain in dB.

\textbf{See also:}

\texttt{bass,treble}

\textbf{fade} (\texttt{fade\_in\_len=0.0, fade\_out\_len=0.0, fade\_shape='q'})

Add a fade in and/or fade out to an audio file. Default fade shape is 1/4 sine wave.

\textbf{Parameters} \texttt{fade\_in\_len} : float, default=0.0

Length of fade-in (seconds). If \texttt{fade\_in\_len} = 0, no fade in is applied.

\texttt{fade\_out\_len} : float, default=0.0

Length of fade-out (seconds). If \texttt{fade\_out\_len} = 0, no fade in is applied.

\texttt{fade\_shape} : str, default=’q’

\textbf{Shape of fade. Must be one of}

- ‘q’ for quarter sine (default),
- ‘h’ for half sine,
- ‘t’ for linear,
- ‘l’ for logarithmic
- ‘p’ for inverted parabola.

\textbf{See also:}

\texttt{splice}

\textbf{gain} (\texttt{gain\_db=0.0, normalize=True, limiter=False, balance=None})

Apply amplification or attenuation to the audio signal.

\textbf{Parameters} \texttt{gain\_db} : float, default=0.0

Target gain in decibels (dB).

\texttt{normalize} : bool, default=True

If True, audio is normalized to \texttt{gain\_db} relative to full scale. If False, simply adjusts the audio power level by \texttt{gain\_db}.

\texttt{limiter} : bool, default=False

If True, a simple limiter is invoked to prevent clipping.

\texttt{balance} : str or None, default=None

\textbf{Balance gain across channels. Can be one of:}

- None applies no balancing (default)
- ‘e’ applies gain to all channels other than that with the highest peak level, such that all channels attain the same peak level
- ‘B’ applies gain to all channels other than that with the highest RMS level, such that all channels attain the same RMS level
• ‘b’ applies gain with clipping protection to all channels other than that with the highest RMS level, such that all channels attain the same RMS level.

If normalize=True, ‘B’ and ‘b’ are equivalent.

See also:

norm, loudness

highpass (frequency, width_q=0.707, n_poles=2)
Apply a high-pass filter with 3dB point frequency. The filter can be either single-pole or double-pole. The filters roll off at 6dB per pole per octave (20dB per pole per decade).

Parameters frequency : float
The filter’s cutoff frequency in Hz.

width_q : float, default=0.707
The filter’s width as a Q-factor. Applies only when n_poles=2. The default gives a Butterworth response.

n_poles : int, default=2
The number of poles in the filter. Must be either 1 or 2

See also:

lowpass, equalizer, sinc, allpass

loudness (gain_db=-10.0, reference_level=65.0)
Loudness control. Similar to the gain effect, but provides equalisation for the human auditory system. The gain is adjusted by gain_db and the signal equalised according to ISO 226 w.r.t. reference_level.

Parameters gain_db : float, default=-10.0
Output loudness (in dB)

reference_level : float, default=65.0
Reference level (in dB) according to which the signal is equalized. Must be between 50 and 75 (dB)

See also:

gain, loudness

lowpass (frequency, width_q=0.707, n_poles=2)
Apply a low-pass filter with 3dB point frequency. The filter can be either single-pole or double-pole. The filters roll off at 6dB per pole per octave (20dB per pole per decade).

Parameters frequency : float
The filter’s cutoff frequency in Hz.

width_q : float, default=0.707
The filter’s width as a Q-factor. Applies only when n_poles=2. The default gives a Butterworth response.

n_poles : int, default=2
The number of poles in the filter. Must be either 1 or 2
See also:

*highpass, equalizer, sinc, allpass*

```
**norm** *(db_level=-3.0)*
```
Normalize an audio file to a particular db level. This behaves identically to the gain effect with normalize=True.

**Parameters**

- **db_level**: float, default=-3.0
  
  Output volume (db)

See also:

*gain, loudness*

```
**overdrive** *(gain_db=20.0, colour=20.0)*
```
Apply non-linear distortion.

**Parameters**

- **gain_db**: float, default=20
  
  Controls the amount of distortion (dB).

- **colour**: float, default=20
  
  Controls the amount of even harmonic content in the output (dB).

```
**pad** *(start_duration=0.0, end_duration=0.0)*
```
Add silence to the beginning or end of a file. Calling this with the default arguments has no effect.

**Parameters**

- **start_duration**: float
  
  Number of seconds of silence to add to beginning.

- **end_duration**: float
  
  Number of seconds of silence to add to end.

See also:

*bend, speed, tempo*

```
**pitch** *(n_semitones, quick=False)*
```
Pitch shift the audio without changing the tempo.

This effect uses the WSOLA algorithm. The audio is chopped up into segments which are then shifted in the time domain and overlapped (cross-faded) at points where their waveforms are most similar as determined by measurement of least squares.

**Parameters**

- **n_semitones**: float
  
  The number of semitones to shift. Can be positive or negative.

- **quick**: bool, default=False
  
  If True, this effect will run faster but with lower sound quality.

See also:

*bend, speed, tempo*

```
**preview** *
```
Play a preview of the output with the current set of effects

```
**rate** *(samplerate, quality='h')*
```
Change the audio sampling rate (i.e. resample the audio) to any given **samplerate**. Better the resampling quality = slower runtime.
Parameters `samplerate`: float
Desired sample rate.

`quality`: str
Resampling quality. One of:
- `q`: Quick - very low quality,
- `l`: Low,
- `m`: Medium,
- `h`: High (default),
- `v`: Very high

`silence_threshold`: float
Silence threshold as percentage of maximum sample amplitude.

`min_silence_duration`: float
The minimum amount of time in seconds required for a region to be considered non-silent.

`buffer_around_silence`: bool
If True, leaves a buffer of `min_silence_duration` around removed silent regions.

See also:
`upsample`, `downsample`, `convert`

`reverb(reverberance=50, high_freq_damping=50, room_scale=100, stereo_depth=100, pre_delay=0, wet_gain=0, wet_only=False)`
Add reverberation to the audio using the ‘freeverb’ algorithm. A reverberation effect is sometimes desirable for concert halls that are too small or contain so many people that the hall’s natural reverberance is diminished. Applying a small amount of stereo reverb to a (dry) mono signal will usually make it sound more natural.

Parameters `reverberance`: float, default=50
Percentage of reverberance

`high_freq_damping`: float, default=50
Percentage of high-frequency damping.

`room_scale`: float, default=100
Scale of the room as a percentage.

`stereo_depth`: float, default=100
Stereo depth as a percentage.

`pre_delay`: float, default=0
Pre-delay in milliseconds.

`wet_gain`: float, default=0
Amount of wet gain in dB

`wet_only`: bool, default=False
If True, only outputs the wet signal.
See also:

`echo`

Reverse the audio completely

`reverse()`

Sets SoX’s global arguments. Overwrites any previously set global arguments. If this function is not explicitly called, globals are set to this function’s defaults.

Parameters

- `dither`: bool, default=False
  If True, dithering is applied for low files with low bit rates.

- `guard`: bool, default=False
  If True, invokes the gain effect to guard against clipping.

- `multithread`: bool, default=False
  If True, each channel is processed in parallel.

- `replay_gain`: bool, default=False
  If True, applies replay-gain adjustment to input-files.

- `verbosity`: int, default=2
  SoX’s verbosity level. One of:

  - 0: No messages are shown at all
  - 1: [Only error messages are shown. These are generated if SoX cannot complete the requested commands.]
  - 2: [Warning messages are also shown. These are generated if SoX can complete the requested commands, but not exactly according to the requested command parameters, or if clipping occurs.]
  - 3: [Descriptions of SoX’s processing phases are also shown.] Useful for seeing exactly how SoX is processing your audio.
  - 4, >4: Messages to help with debugging SoX are also shown.

`silence()`

Removes silent regions from an audio file.

Parameters

- `location`: int, default=0
  Where to remove silence. One of:

  - 0 to remove silence throughout the file (default),
  - 1 to remove silence from the beginning,
  - -1 to remove silence from the end,

- `silence_threshold`: float, default=0.1
  Silence threshold as percentage of maximum sample amplitude. Must be between 0 and 100.

- `min_silence_duration`: float, default=0.1
  The minimum amount of time in seconds required for a region to be considered non-silent.
buffer_around_silence : bool, default=False

If True, leaves a buffer of min_silence_duration around removed silent regions.

See also:

tempo (factor, audio_type=None, quick=False)

Time stretch audio without changing pitch.

This effect uses the WSOLA algorithm. The audio is chopped up into segments which are then shifted in the time domain and overlapped (cross-faded) at points where their waveforms are most similar as determined by measurement of least squares.

Parameters  

factor : float

The ratio of new tempo to the old tempo. For ex. 1.1 speeds up the tempo by 10%; 0.9 slows it down by 10%.

audio_type : str

Type of audio, which optimizes algorithm parameters. One of:

- m : Music,
- s : Speech,
- l : Linear (useful when factor is close to 1),

quick : bool, default=False

If True, this effect will run faster but with lower sound quality.

See also:

stretch, speed, pitch

treble (gain_db, frequency=3000.0, slope=0.5)

Boost or cut the treble (lower) frequencies of the audio using a two-pole shelving filter with a response similar to that of a standard hi-fi’s tone-controls. This is also known as shelving equalisation.

The filters are described in detail in http://musicdsp.org/files/Audio-EQ-Cookbook.txt

Parameters  

gain_db : float

The gain at the Nyquist frequency. For a large cut use -20, for a large boost use 20.

frequency : float, default=100.0

The filter’s cutoff frequency in Hz.

slope : float, default=0.5

The steepness of the filter’s shelf transition. For a gentle slope use 0.3, and use 1.0 for a steep slope.

See also:

bass, equalizer

trim (start_time, end_time)

Excerpt a clip from an audio file, given a start and end time.

Parameters  

start_time : float
Start time of the clip (seconds)

end_time : float
End time of the clip (seconds)

Combiners

Python wrapper around the SoX library. This module requires that SoX is installed.

class sox.combine.Combiner(\texttt{input\_filepath\_list, output\_filepath, combine\_type, input\_volumes=\texttt{None}})

Audio file combiner. Class which allows multiple files to be combined to create an output file, saved to output\_filepath.

Inherits all methods from the Transformer class, thus any effects can be applied after combining.

\textbf{Parameters}\n
- \texttt{input\_filepath\_list} : list of str
  List of paths to input audio files.

- \texttt{output\_filepath} : str
  Path to desired output file. If a file already exists at the given path, the file will be overwritten.

- \texttt{combine\_type} : str
  Input file combining method. One of the following values:
  \begin{itemize}
  \item \texttt{concatenate} [combine input files by concatenating in the] order given.
  \item \texttt{merge} [combine input files by stacking each input file into] a new channel of the output file.
  \item \texttt{mix} [combine input files by summing samples in corresponding] channels.
  \item \texttt{mix\_power} [combine input files with volume adjustments such] that the output volume is roughly equivalent to one of the input signals.
  \item \texttt{multiply} [combine input files by multiplying samples in] corresponding samples.
  \end{itemize}

- \texttt{input\_volumes} : list of float, default=\texttt{None}
  List of volumes to be applied upon combining input files. Volumes are applied to the input files in order. If None, input files will be combined at their original volumes.

\textbf{Methods}

- \texttt{build()} 
  Executes SoX.

\textbf{File info}

Audio file info computed by soxi.
sox.file_info.bitrate(input_filepath)
Number of bits per sample (0 if not applicable).

Parameters:
- input_filepath: str
  Path to audio file.

Returns:
- bitrate: int
  Number of bits per sample returns 0 if not applicable.

sox.file_info.channels(input_filepath)
Show number of channels.

Parameters:
- input_filepath: str
  Path to audio file.

Returns:
- channels: int
  Number of channels.

sox.file_info.comments(input_filepath)
Show file comments (annotations) if available.

Parameters:
- input_filepath: str
  Path to audio file.

Returns:
- comments: str
  File comments from header. If no comments are present, returns an empty string.

sox.file_info.duration(input_filepath)
Show duration in seconds (0 if unavailable).

Parameters:
- input_filepath: str
  Path to audio file.

Returns:
- duration: float
  Duration of audio file in seconds. If unavailable or empty, returns 0.

sox.file_info.encoding(input_filepath)
Show the name of the audio encoding.

Parameters:
- input_filepath: str
  Path to audio file.

Returns:
- encoding: str
  Audio encoding type.

sox.file_info.file_extension(filepath)
Get the extension of a filepath.

Parameters:
- filepath: str
  File path.

sox.file_info.file_type(input_filepath)
Show detected file-type.

Parameters:
- input_filepath: str
  Path to audio file.

Returns:
- file_type: str
  File format type (ex. ‘wav’).
sox.file_info.num_samples(input_filepath)
Show number of samples (0 if unavailable).

Parameters input_filepath : str
Path to audio file.

Returns n_samples : int
total number of samples in audio file. Returns 0 if empty or unavailable

sox.file_info.sample_rate(input_filepath)
Show sample-rate.

Parameters input_filepath : str
Path to audio file.

Returns samplerate : float
number of samples/second

sox.file_info.validate_input_file(input_filepath)
Input file validation function. Checks that file exists and can be processed by SoX.

Parameters input_filepath : str
The input filepath.

sox.file_info.validate_input_file_list(input_filepath_list)
Input file list validation function. Checks that object is a list and contains valid filepaths that can be processed by SoX.

Parameters input_filepath_list : list
A list of filepaths.

sox.file_info.validate_output_file(output_filepath)
Output file validation function. Checks that file can be written, and has a valid file extension. Throws a warning if the path already exists, as it will be overwritten on build.

Parameters output_filepath : str
The output filepath.

Core functionality

Base module for calling SoX

exception sox.core.SoxError(*args, **kwargs)
Exception to be raised when SoX exits with non-zero status.

exception sox.core.SoxiError(*args, **kwargs)
Exception to be raised when SoXi exits with non-zero status.

sox.core.all_equal(list_of_things)
Check if a list contains identical elements.

Parameters list_of_things : list
list of objects

sox.core.enquote_filepath(fpath)
Wrap a filepath in double-quotes to protect difficult characters.

sox.core.is_number(var)
Check if variable is a numeric value.

Parameters var : object
sox.core.play(args)
    Pass an argument list to play.
    **Parameters** args : iterable
    Argument list for play. The first item can, but does not need to, be ‘play’.

sox.core.sox(args)
    Pass an argument list to SoX.
    **Parameters** args : iterable
    Argument list for SoX. The first item can, but does not need to, be ‘sox’.

sox.core.soxi(filepath, argument)
    Base call to Soxi.
    **Parameters** filepath : str
    Path to audio file.

    argument : str
    Argument to pass to Soxi.

    **Returns** shell_output : str
    command line output of Soxi
CHAPTER 3

Changes

Changes

v0.1

- Initial release.
Contribute

- Issue Tracker
- Source Code
CHAPTER 5

Indices and tables

- genindex
- modindex
- search
S

sox, 3
sox.combine, 12
sox.core, 14
sox.file_info, 12
sox.transform, 3
Index

A
all_equal() (in module sox.core), 14
allpass() (sox.transform.Transformer method), 3

B
bandpass() (sox.transform.Transformer method), 4
bandreject() (sox.transform.Transformer method), 4
bass() (sox.transform.Transformer method), 4
bitrate() (in module sox.file_info), 12
build() (sox.combine.Combiner method), 12
build() (sox.transform.Transformer method), 5

C
channels() (in module sox.file_info), 13
Combiner (class in sox.combine), 12
comments() (in module sox.file_info), 13
compand() (sox.transform.Transformer method), 5
convert() (sox.transform.Transformer method), 5

duration() (in module sox.file_info), 13

E
encoding() (in module sox.file_info), 13
enquote_filepath() (in module sox.core), 14
equalizer() (sox.transform.Transformer method), 5

F
fade() (sox.transform.Transformer method), 6
file_extension() (in module sox.file_info), 13
file_type() (in module sox.file_info), 13

G
gain() (sox.transform.Transformer method), 6

H
highpass() (sox.transform.Transformer method), 7

I
is_number() (in module sox.core), 14

L
loudness() (sox.transform.Transformer method), 7
lowpass() (sox.transform.Transformer method), 7

N
norm() (sox.transform.Transformer method), 8
num_samples() (in module sox.file_info), 13

O
overdrive() (sox.transform.Transformer method), 8

P
pad() (sox.transform.Transformer method), 8
pitch() (sox.transform.Transformer method), 8
play() (in module sox.core), 14
preview() (sox.transform.Transformer method), 8

R
rate() (sox.transform.Transformer method), 8
reverb() (sox.transform.Transformer method), 9
reverse() (sox.transform.Transformer method), 10

S
sample_rate() (in module sox.file_info), 14
set_globals() (sox.transform.Transformer method), 10
silence() (sox.transform.Transformer method), 10
sox (module), 3
sox() (in module sox.core), 15
sox.combine (module), 12
sox.core (module), 14
sox.file_info (module), 12
sox.transform (module), 3
SoxError, 14
soxi() (in module sox.core), 15
SoxiError, 14
tempo() (sox.transform.Transformer method), 11
Transformer (class in sox.transform), 3
treble() (sox.transform.Transformer method), 11
trim() (sox.transform.Transformer method), 11

validate_input_file() (in module sox.file_info), 14
validate_input_file_list() (in module sox.file_info), 14
validate_output_file() (in module sox.file_info), 14