python-samplerate Documentation

Release 0.1.0+6.ged73d7a

Tino Wagner

Apr 10, 2019

Contents

This is a wrapper around Erik de Castro Lopo's libsamplerate (aka Secret Rabbit Code) for high-quality sample rate conversion.

It implements all three APIs available in libsamplerate:

- **Simple API**: for resampling a large chunk of data with a single library call
- **Full API**: for obtaining the resampled signal from successive chunks of data
- **Callback API**: like Full API, but input samples are provided by a callback function

Library calls to libsamplerate are performed using CFFI.

CHAPTER 1

Installation

```
$ pip install samplerate
```

Binaries of libsamplerate for macOS and Windows (32 and 64 bit) are included and used if not present on the system. On Linux systems, you should also install *libsamplerate0* (Debian derivatives), *libsamplerate* (Arch), or similar.

CHAPTER 2

Usage

```python
import numpy as np
import samplerate

# Synthesize data
fs = 1000.
t = np.arange(fs * 2) / fs
input_data = np.sin(2 * np.pi * 5 * t)

# Simple API
ratio = 1.5
converter = 'sinc_best'  # or 'sinc_fastest', ...
output_data_simple = samplerate.resample(input_data, ratio, converter)

# Full API
resampler = samplerate.Resampler(converter, channels=1)
output_data_full = resampler.process(input_data, ratio, end_of_input=True)

# The result is the same for both APIs.
assert np.allclose(output_data_simple, output_data_full)

# See `samplerate.CallbackResampler` for the Callback API, or
# `examples/play_modulation.py` for an example.
```

See `samplerate.resample`, `samplerate.Resampler`, and `samplerate.CallbackResampler` in the API documentation for details.

CHAPTER 3

See also

- scikits.samplerate implements only the Simple API and uses Cython for extern calls. The *resample* function of *scikits.samplerate* and this package share the same function signature for compatiblity.
- resampy: sample rate conversion in Python + Cython.

CHAPTER 4

License

This project is licensed under the MIT license.

As of version 0.1.9, libsamplerate is licensed under the 2-clause BSD license.

CHAPTER 5

API documentation

5.1 *samplerate* module documentation

5.1.1 `samplerate` – Python bindings for libsamplerate based on CFFI and NumPy

Simple API

samplerate.**resample**()
See *samplerate.converters.resample()*.

Full API

class samplerate.**Resampler**
See *samplerate.converters.Resampler()*.

Callback API

class samplerate.**CallbackResampler**
See *samplerate.converters.CallbackResampler()*.

Exceptions

class samplerate.**ResamplingError**
See *samplerate.converters.CallbackResampler()*.

5.1.2 `samplerate.converters` – Sample rate converters

Converter types

class samplerate.converters.**ConverterType**
Enum of samplerate converter types.

Pass any of the members, or their string or value representation, as `converter_type` in the resamplers.

linear = 4

sinc_best = 0

sinc_fastest = 2

sinc_medium = 1

zero_order_hold = 3

Sample rate converters

Simple

samplerate.converters.**resample**(*input_data*, *ratio*, *converter_type='sinc_best'*, *verbose=False*)
Resample the signal in *input_data* at once.

Parameters

- **input_data** (*ndarray*) – Input data. A single channel is provided as a 1D array of *num_frames* length. Input data with several channels is represented as a 2D array of shape (*num_frames*, *num_channels*). For use with *libsamplerate*, *input_data* is converted to 32-bit float and C (row-major) memory order.
- **ratio** (*float*) – Conversion ratio = output sample rate / input sample rate.
- **converter_type** (ConverterType, *str, or int*) – Sample rate converter.
- **verbose** (*bool*) – If *True*, print additional information about the conversion.

Returns **output_data** – Resampled input data.

Return type ndarray

Note: If samples are to be processed in chunks, *Resampler* and *CallbackResampler* will provide better results and allow for variable conversion ratios.

Full API

class samplerate.converters.**Resampler**(*converter_type='sinc_fastest'*, *channels=1*)
Resampler.

Parameters

- **converter_type** (ConverterType, *str, or int*) – Sample rate converter.
- **num_channels** (*int*) – Number of channels.

channels
Number of channels.

converter_type
Converter type.

process(*input_data*, *ratio*, *end_of_input=False*, *verbose=False*)
Resample the signal in *input_data*.

Parameters

- **input_data** (*ndarray*) – Input data. A single channel is provided as a 1D array of *num_frames* length. Input data with several channels is represented as a 2D array of shape (*num_frames*, *num_channels*). For use with *libsamplerate*, *input_data* is converted to 32-bit float and C (row-major) memory order.
- **ratio** (*float*) – Conversion ratio = output sample rate / input sample rate.
- **end_of_input** (*int*) – Set to *True* if no more data is available, or to *False* otherwise.
- **verbose** (*bool*) – If *True*, print additional information about the conversion.

Returns **output_data** – Resampled input data.

Return type ndarray

reset()
Reset internal state.

set_ratio(*new_ratio*)
Set a new conversion ratio immediately.

Callback API

class samplerate.converters.**CallbackResampler**(*callback*, *ratio*, *converter_type='sinc_fastest'*, *channels=1*)
CallbackResampler.

Parameters

- **callback** (*function*) – Function that returns new frames on each call, or *None* otherwise. A single channel is provided as a 1D array of *num_frames* length. Input data with several channels is represented as a 2D array of shape (*num_frames*, *num_channels*). For use with *libsamplerate*, *input_data* is converted to 32-bit float and C (row-major) memory order.
- **ratio** (*float*) – Conversion ratio = output sample rate / input sample rate.
- **converter_type** (ConverterType, *str, or int*) – Sample rate converter.
- **channels** (*int*) – Number of channels.

ratio
Conversion ratio = output sample rate / input sample rate.

read(*num_frames*)
Read a number of frames from the resampler.

Parameters **num_frames** (*int*) – Number of frames to read.

Returns **output_data** – Resampled frames as a (*num_output_frames*, *num_channels*) or (*num_output_frames*,) array. Note that this may return fewer frames than requested, for example when no more input is available.

Return type ndarray

reset()
Reset state.

set_starting_ratio(*ratio*)
Set the starting conversion ratio for the next *read* call.

5.1.3 samplerate.exceptions – Sample rate exceptions

class samplerate.exceptions.**ResamplingError**(*error*)
Runtime exception of libsamplerate

5.1.4 samplerate.lowlevel – Lowlevel wrappers around libsamplerate

Lowlevel wrappers around libsamplerate.

The docstrings of the *src_** functions are adapted from the libsamplerate header file.

samplerate.lowlevel.**src_callback_new**(*callback*, *converter_type*, *channels*)
Initialisation for the callback based API.

Parameters

- **callback** (*function*) – Called whenever new frames are to be read. Must return a NumPy array of shape (num_frames, channels).
- **converter_type** (*int*) – Converter to be used.
- **channels** (*int*) – Number of channels.

Returns

- *state* – An anonymous pointer to the internal state of the converter.
- *handle* – A CFFI handle to the callback data.
- **error** (*int*) – Error code.

samplerate.lowlevel.**src_callback_read**(*state*, *ratio*, *frames*, *data*)
Read up to *frames* worth of data using the callback API.

Returns **frames** – Number of frames read or -1 on error.

Return type int

samplerate.lowlevel.**src_delete**(*state*)
Release *state*.

Cleanup all internal allocations.

samplerate.lowlevel.**src_error**(*state*)
Return an error number.

samplerate.lowlevel.**src_get_description**(*converter_type*)
Return the description of the converter given by *converter_type*.

samplerate.lowlevel.**src_get_name**(*converter_type*)
Return the name of the converter given by *converter_type*.

samplerate.lowlevel.**src_get_version**()
Return the version string of libsamplerate.

samplerate.lowlevel.**src_is_valid_ratio**(*ratio*)
Return *True* if ratio is a valid conversion ratio, *False* otherwise.

samplerate.lowlevel.**src_new**(*converter_type*, *channels*)
Initialise a new sample rate converter.

Parameters

- **converter_type** (*int*) – Converter to be used.
- **channels** (*int*) – Number of channels.

Returns

- *state* – An anonymous pointer to the internal state of the converter.
- **error** (*int*) – Error code.

samplerate.lowlevel.**src_process**(*state*, *input_data*, *output_data*, *ratio*, *end_of_input=0*)
Standard processing function.

Returns non zero on error.

samplerate.lowlevel.**src_reset**(*state*)
Reset the internal SRC state.

Does not modify the quality settings. Does not free any memory allocations. Returns non zero on error.

samplerate.lowlevel.**src_set_ratio**(*state*, *new_ratio*)
Set a new SRC ratio.

This allows step responses in the conversion ratio. Returns non zero on error.

samplerate.lowlevel.**src_simple**(*input_data*, *output_data*, *ratio*, *converter_type*, *channels*)
Perform a single conversion from an input buffer to an output buffer.

Simple interface for performing a single conversion from input buffer to output buffer at a fixed conversion ratio. Simple interface does not require initialisation as it can only operate on a single buffer worth of audio.

samplerate.lowlevel.**src_strerror**(*error*)
Convert the error number into a string.

CHAPTER 6

Change Log

6.1 Change Log

6.1.1 0.1.0

- Initial release.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index