samplerate Documentation

Release 0.3.0

David Cournapeau

March 26, 2009

CONTENTS

CHAPTER

ONE

INTRODUCTION

1.1 Purpose of samplerate

Samplerate is a python module to do high quality resampling of audio signals, using sinc interpolation. Samplerate gives you functionalities similar to resample in matlab (the actual resampling method may differ, though), and is intended to be used with numpy arrays.

1.2 Acknowledgments

Please note that samplerate is essentially a wrapper around the Sampling Rate Conversion library, aka Source Rabbit Code, the high quality sampling rate convertion library of Erik Castrop de Lopo. All the features of samplerate are his owns, all the bugs mine.

CHAPTER

TWO

DOWNLOAD AND INSTALLATION

2.1 Supported platforms

Samplerate has been run succesfully on the following platforms:

- linux ubuntu (32 and 64 bits) and RHEL 5 (32 and 64 bits)
- windows XP (32 bits)
- Mac OS X (10.5, intel)

I would be interested to hear anyone who succeesfully used it on other platforms.

2.2 Download

Releases are available on Pypi:

http://pypi.python.org/pypi/scikits.samplerate/

Samplerate is part of scikits, and its source can be downloaded directly from the scikits svn repository:

```
svn co http://svn.scipy.org/svn/scikits/trunk/samplerate
```

2.3 Install from binaries

2.3.1 Requirements

To install the binaries, samplerate requires the following softwares:

- a python interpreter.
- numpy (any version >= 1.2 should work).

2.3.2 Binaries

Binaries for Mac OS X and Windows are provided on Pypi - they are statically linked to SRC (so that you don't need to install your own version of SRC first). If you are not familiar with building from sources, you are strongly advised to use those.

2.4 Installation from sources

2.4.1 Requirements

samplerate requires the following softwares:

- a python interpreter.
- Source Code Rabbit (SRC)
- numpy (any version >= 1.2 should work).
- setuptools

On Ubuntu, you can install the dependencies as follow:

```
sudo apt-get install python-dev python-numpy python-setuptools libsamplerate0-dev
```

2.4.2 Build

For unix users, if SRC is installed in standart location (eg /usr/lib, /usr/local/lib), the installer should be able to find them automatically, and you only need to do a "python setup.py install". In other cases, you need to create a file site.cfg to set the location of SRC and its header (there are site.cfg examples which should give you an idea how to use them on your platform).

2.5 License

Samplerate is released under the GPL, which forces you to release back the modifications you may make in the version of samplerate you are distributing,

Audiolab is under the GPL because SRC itself is under the GPL, and as such, a BSD python wrapper is of little value.

CHAPTER

THREE

USAGE

3.1 A simple example

```
import numpy as np
import pylab as plt
from scikits.samplerate import resample

fs  = 44100.
fr  = 48000.
# Signal to resample
sins    = np.sin(2 * np.pi * 1000/fs * np.arange(0, fs * 2))
# Ideal resampled signal
idsin   = np.sin(2 * np.pi * 1000/fr * np.arange(0, fr * 2))

conv1   = resample(sins, fr/fs, 'linear')
conv3   = resample(sins, fr/fs, 'sinc_best')

err1    = conv1[fr:fr+2000] - idsin[fr:fr+2000]
err3    = conv3[fr:fr+2000] - idsin[fr:fr+2000]

plt.subplot(3, 1, 1)
plt.plot(idsin[fs:fs+2000])
plt.title('Resampler residual quality comparison')

plt.subplot(3, 1, 2)
plt.plot(err1)
plt.ylabel('Linear')

plt.subplot(3, 1, 3)
plt.plot(err3)
plt.ylabel('Sinc')

plt.savefig('example1.png', dpi = 100)
```

The figure plots the error signal for a 1000 Hz sinusoid converted from 44.1 Khz to 48 Khz (compare to an sinusoid directly sampled @ 48 khz).

3.2 Listing available convertors

The following example shows how to get a long description of each available convertor in SRC.

```
from scikits.samplerate import available_convertors, convertor_description

for type in available_convertors():
    print convertor_description(type)
```

CHAPTER

FOUR

FULL API

available_convertors()
Return the list of available convertor.

convertor_description(*type*)
Return a detailed description of the given convertor type.

Parameters **type** : str
resample type (see Notes).

Returns **descr** : str
String describing the given convertor.

resample(*ndarray input, r, type, verbose=False*)
Resample the input array using the given converter type, with a ratio r (ie the resulting array will have a length ~ r * input's length).

Parameters **input: array** :
input data

r: float :
ratio

type: str :
resample type (see Note)

Returns **output: array** :
output data, whose size is approximately r * input.size

Notes

If input has rank 1, than all data are used, and are assumed to be from a mono signal. If rank is 2, the number columns will be assumed to be the number of channels.

The list of convertor types is available from the function available_convertors.

Only sinc_*-based interpolation provide good quality; linear and zero_order_hold should be avoided as much as possible, and be used only when speed is critical.

CHAPTER

FIVE

TODO

For now, samplerate is very limited, and supports only constant ratio. But SRC itself is able to do ratio which change overtime. It is usable as a matlab resample, though, which is why I started this in the first place.

INDEX