audiolab Documentation

Release 0.9.0dev

David Cournapeau

December 04, 2008

CONTENTS

CHAPTER

ONE

Introduction

For people doing audio processing, it is useful to be able to import data from audio files, and export them back, as well as listening to the results of some processing; matlab have functions such as wavread, wavwrite, soundsc, etc... for that purposes. The goal of audiolab is to give those capabilities to the scipy environment by wrapping the excellent library libsndfile from Erik Castro de Lopo. Audiolab supports all format supported by libsndfile, including wav, aiff, ircam files, and flac (an open source lossless compressed format); see here for a complete list.

> **Note**: starting at version 0.10, read/write data convention will change from one column per channel to one row per channel, to follow numpy conventions.
>
> **Note**: The library is still in beta stage: reading and writing data is possible, but only in frames, not per item. Also, the ability to play data on the system's soundcard is not there yet, except on Linux.
>
> **Note**: The online version of this document is not always up to date. The pdf included in the package is the reference, and always in sync with the package. If something does not work, please refer first to the pdf included in the package.

Tables of contents

CHAPTER

TWO

Download and installation

2.1 Supported platforms

Audiolab has been run succesfully on the following platforms:

- linux ubuntu (32 and 64 bits)
- windows XP (32 bits)
- Mac OS X (10.5)

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

2.2 Download

audiolab is part of scikits: its source can be downloaded directly from the scikits svn repository:

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

2.3 Requirements

audiolab requires the following softwares:

- a python interpreter.
- libsndfile (including the header sndfile.h, which means linux users should download the libsndfile-dev package).
- numpy (any version >= 1.0 should work).
- ctypes (version >= 1.0.1; ctypes is included in python 2.5)
- setuptools

On Ubuntu, you can install the dependencies as follow:

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

2.4 Optional

Audiolab can optionally install audio backends. For now, only alsa is supported, for linux audio support. You need alsa headers for this to work; on Ubuntu, you can install them with the following command:

```
sudo apt-get install libasound2-dev
```

2.5 Installation

For unix users, if libsndfile 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 libsndfile and its header (there are site.cfg examples which should give you an idea how to use them on your platform).

For windows users: the library distributed by Erik Castro de Lopo cannot be used directly; you need to follow the instructions given in libsndfile distribution in the file README-precompiled-dll.txt. See also site.cfg.win32.

2.6 License

audiolab is released under the LGPL, which forces you to release back the modifications you may make in the version of audiolab you are distributing, but you can still use it in closed softwares, as long as you don't use a modified version of it.

CHAPTER

THREE

Quick view

The following code shows you how to open a file for read, reading the first 1000 frames, and closing it:

```
import scikits.audiolab as  audiolab

a       = audiolab.sndfile('test.wav', 'read')
data    = a.read_frames(1000)
a.close()
```

CHAPTER

FOUR

Usage

4.1 Opening a file and getting its parameters

Once imported, audiolab gives you access the sndfile class, which is the class of audiolab use to open audio files. You create a sndfile instance when you want to open a file for reading or writing (the file test.flac is included in the audiolab package, in the test_data directory):

```
import scikits.audiolab as audiolab

filename    = 'test.wav'
a           = audiolab.sndfile(filename, 'read')

print a
```

Prints you the informations related to the file, like its sampling rate, the number of frames, etc... You can of course get each parameter individually by using the corresponding sndfile.get* accessors.

4.2 Importing audio data

Now that we've opened a file, we would like to read its audio content, right ? For now, you can only import the data as floating point data, float (32 bits) or double (64 bits). The function sndfile.read_frames read n frames, where a frame contains a sample of each channel (one in mono, 2 in stereo, etc...):

```
import numpy as N

import scikits.audiolab as audiolab

filename    = 'test.wav'
a           = audiolab.sndfile(filename, 'read')

tmp         = a.read_frames(1e4)
float_tmp   = a.read_frames(1e4, dtype = N.float32)

import pylab as P
P.plot(tmp[:])
```

The above code import 10000 frames, and plot the first channel using matplotlib (see below). A frame holds one sample from each channel: 1000 frames of a stereo file is 2000 samples. Each channel is one column of the numpy array. The read functions follow numpy conventions, that is by default, the data are read as double, but you can give a dtype argument to the function.

4.3 The format class

When opening a file for writing, you need to give various parameters related to the format such as the file format, the encoding. The format class is used to create valid formats from those parameters By default, the format class creates a format object with file type wav, and 16 bits pcm encoding:

```
from scikits.audiolab import formatinfo as format

f = format('aiff', 'ulaw')
print f

f = format('ircam', 'float32')
print f
```

prints back "Major Format: AIFF (Apple/SGI), Encoding Format: U-Law" and "Major Format: SF (Berkeley/IRCAM/CARL), Encoding Format: 32 bit float".

To get a list of all possible file format and encoding, the function supported_* are available:

```
from scikits.audiolab import supported_format, supported_encoding, \
            supported_endianness

print supported_format()
print supported_encoding()
print supported_endianness()
```

Note that not all combination of encoding, endianness and format are possible. If you try to create a format with incompatible values, you will get an exception while creating an instance of format.

4.4 Writing data to a file

Opening a file for writing is a bit more complicated than reading; you need to say which format you are requesting, the number of channels and the sampling rate (in Hz) you are requesting; all thoses information are mandatory ! The class format is used to build a format understable by libsndfile from 'user-friendly' values. Let's see how it works.

```
from tempfile import mkstemp
from os import remove

import numpy as N
from scikits.audiolab import formatinfo as format
import scikits.audiolab as audiolab

# Create a temp file in the system temporary dir, and always remove
# it at the end
cd, filename    = mkstemp('tmptest.wav')
try:
    fmt         = format('wav', 'pcm24')
    nchannels   = 2
```

```
    fs          = 44100

    afile =  audiolab.sndfile(filename, 'write', fmt, nchannels, fs)

    # Create a stereo white noise, with Gaussian distribution
    tmp = 0.1 * N.random.randn(1000, nchannels)

    # Write the first 500 frames of the signal
    # Note that the write_frames method uses tmp's numpy dtype to determine how
    # to write to the file; sndfile also converts the data on the fly if necessary
    afile.write_frames(tmp, 500)

    afile.close()

    # Let's check that the written file has the expected meta data
    afile = audiolab.sndfile(filename, 'read')
    assert(afile.get_samplerate() == fs)
    assert(afile.get_channels() == nchannels)
    assert(afile.get_nframes() == 500)
finally:
    remove(filename)
```

4.5 Matlab-like API

audiolab also have a matlab-like API for audio IO. Its usage is as similar as it can get using python:

```
from tempfile import mkstemp
from os.path import join, dirname
from os import remove

from scikits.audiolab import wavread, wavwrite

(tmp, fs, enc)  = wavread('test.wav')
if tmp.ndim < 2:
    nc  = 1
else:
    nc  = tmp.shape[1]

print "The file has %d frames, %d channel(s)" % (tmp.shape[0], nc)
print "FS is %f, encoding is %s" % (fs, enc)

fd, cfilename   = mkstemp('pysndfiletest.wav')
try:
    wavwrite(tmp, cfilename, fs = 16000, enc = 'pcm24')
finally:
    remove(cfilename)
```

4.6 Sound output

New feature in 0.9: only ALSA (Linux sound API) has been implemented so far.

```
from scikits.audiolab import play, wavread

data, fs, enc = wavread('test.wav')
# There is a discrepency between wavread (one column per channel) and play
# convention (one row per channel). Audiolab will be fully converted to 'numpy
# conventions' (last axis per default) after version 0.9
play(data.T, rate=fs)
```

CHAPTER

FIVE

Known bugs:

- the function supported_* are broken (they never worked correctly). This will be fixed for audiolab 0.10
- there seems to be a problem when using libsndfile fseek facilities with flac files (which are necessary for the functions flacread/flacwrite). The problem seems to be with libFLAC; for this reason, seek in flac files is not enabled by default for now. See FLAC_SUPPORT.txt for more informations.

CHAPTER

SIX

TODO

audiolab is still in early stages. Before a release, I would like to implement the follwings:

- support (at least some) meta-data embedded in some audio files format.
- support the libsndfile's error system
- player on all major plateforms (at least linux/windows/max OS X)