spectraplotpy Documentation

Release 0.0.1

SpectralGroup

August 22, 2014

Contents

CHAPTER 1

Intro

spectraplotpy intents to implement input, processing and output functionalities specific for spectral data on top of the scientific python stack (scipy, numpy and matplotlib).

Contents:

1.1 Quick Start

1.1.1 Getting started

spectraplotpy helps you with common task when analaysing spectral data, providing functionalities for reading and writing several data formats, process and plot several kinds of spectra.

In order to install the library, you download this repository and build the package with setup tools,

```
$ git clone https://github.com/odarbelaeze/spectraplotpy.git
$ cd spectraplotpy
$ python setup.py install
```

Loading a generic spectrum from an Aviv formated file in the python environment:

```
>>> from spectraplotpy import AvivImporter
>>> from spectraplotpy import Spectrum
>>> a = AvivImporter('spectral_data_file_path/filename')
>>> s = Spectrum(a.dataset)
```

Adding two spectral datasets:

```
>>> a1 = AvivImporter('spectral_data_file_path/filename2')
>>> s1 = Spectrum(a1.dataset)
>>> s2 = s.sub(s1)  # s2 = s - s1
```

Plotting the spectrum with the default plot settings,

```
import matplotlib.pyplot as plt
s.plot(plt)
plt.show()
```

Exporting the spectrum to a CSVFile:

```
from spectraplotpy import CSVExporter
csve = CSVExporter(s.dataset)
csve.save('myspectrum.csv')
```

Futher operations can be gotten from the detailed documentation.

1.1.2 Development setup (for developer)

The basic dependencies to develop the project are,

```
matplotlib
scipy
numpy
pytest # For testing
sphinx # For documentation
pylint # For pep-8 compilance
```

You can install de dependecies through pip,

```
$ pip install matplotlib scipy numpy pytest sphinx pylint
```

or just let the setup script to install them for you.

In order to develop using virtual env, within your virtual env just call

```
$ python setup.py develop
```

this will allow you to do *import spectraplotpy* anywhere in your filesystem.

1.1.3 Testing

Once you get everything set up, you can run the tests using,

```
$ python setup.py test
```

Before you do a pull request make sure your code agrees with pylint (as far as possible) and passes all tests.

In order to run the tests for the *Importer* classes you'll need to provide some sample data available trough the trello board.

1.2 Reference

spectraplotpy is made up of four decoupled building blocks, the first one is the *Dataset*, the *Importers*, the *Spectra* and the *Exporters*.

The *Importers* take data from several different formated files, and populate a *Dataset*, then the the *Dataset* can be passed around to the *Spectra* in order to do some processing and plotting or directly to the *Exporters*.

1.2.1 Dataset

The dataset module defines the datastructure shared among the **spectraplotpy** classes, you need to implement a similar interface in order to leverage the class.

The users are encouraged to directly access the data members of this class, and is up to them to keep the data consistently.

```
import spectrapotpy as spp
import numpy as np

ds = spp.Dataset()
ds.x = np.arrange(0, np.pi, 1000)
ds.y = np.sin(x)
```

1.2.2 Importers

The importers functionalities to parse input from several file types, it also allows you to easily create new importers for your own formats subclasing the *Importer* class and overriding some methods.

The library provides several importer for several formats, and the users are encouraged to create their own.

Examples

Importing from an Aviv file,

```
import spectraplotpy as spp
avii = AvivImporter('filename.cd')
# then you can access the dataset and pass it around
print avii.dataset.x, avii,dataset.y
```

Creating a custom importer,

```
import spectraplotpy as spp

class XRDPanalithicalImporter(spp.Importer):
    def parse_metadata(self, metadata_txt):
        # override this function to get the
        pass

    def parse_data(self, data_txt):
        # override this to parse your custom data
        pass
```

1.2.3 Spectra

The *Spectrum* class provides some functionalities that allow the users to operate their data in a very intuitive fashion.

1.2.4 Exporters

Much like *Importers*, exporters provide functionality to export to several file formats, there are text based exporters and plot based exporters.

CHAPTER 2

Indices and tables

- *genindex*
- *modindex*
- *search*