spectrum$_{o}verloadDocumentation$

Release 0.1.0

Jason Neal

Nov 03, 2017

Contents

Spectrum_overload is to manipulate astronomical spectra in a spectrum class with *overloaded operators*. This means that you can easily divide and subtract spectra from each other using the math operators +, −, ⋆, /, ⋆⋆ keeping the wavelength, flux and headers together.

CHAPTER 1

Installation

Spectrum_overload is currently available from `github`.

Navigate to where you want to put files. Then Download:

with `git`:

```
$ git clone https://github.com/jason-neal/spectrum_overload.git
```

To install, run:

```
$ cd spectrum_overload
$ python setup.py install

# Or:

$ cd spectrum_overload
$ pip install .

# Or, for an editable installation:

$ cd spectrum_overload
$ pip install -e .
```

The plan is to have it available via `pypi` someday in the future.:

```
# Bug me about this.
$ pip install spectrum-overload
```

That day is not today...

If you have any issues with installation of spectrum_overload please open an issue so it can be resolved.

1.1 Requirements

The main requirements are

- numpy
- astropy
- scipy
- pyastronomy

If you are needing to use this package you probably have these installed already...

Unfortunately pyastronomy cannot be added to the `requirements.txt` alongside `numpy` due to issue #22 with the setup dependency of `numpy` in `pyastronomy`. If you do not have it installed already then run:

```
$ pip install pyastronomy
```

Other requirements are needed for running the pytest test suite. If you want to try run the tests, run:

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

after normal installation.

These other dependencies that you may need are

- pytest-runner
- hypothesis
- pytest
- pytest-cov
- python-coveralls
- coverage

1.2 Editable Installation

If you want to modify spectrum_overload you can instead install it like:

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

CHAPTER 2

Quickstart Guide

First install spectrum_overload via the *installation guide*.

Import the Spectrum class into your code.

```
from spectrum_overload.Spectrum import Spectrum
```

Then to create a spectrum object, and raise to the power of 2 use:

```
s = Spectrum(flux=f, xaxis=x)

# power law scaling
s = s ** 2
```

where `f` and `x` are 1D lists or numpy arrays.

Note: Flux is the first kwarg. Be careful not to switch the order of flux and xaxis when not specifying the kwarg names.

By default the spectrum is "calibrated". If the xaxis is only the pixel values of your spectra you can pass:

```
s = Spectrum(flux=f, xaxis=pxls, calibrated=False)
```

or if no xaxis is given it will be set by

```
xaxis = np.arange(len(flux))
```

You can calibrate the spectra with a polynomial using the `calibrate_with` method which uses `np.polyval`:

```
s.calibrate_with([p0, p1, p2 ...])
```

Some methods are only available when you have a wavelength calibrated spectra, such as Doppler shifting with `Spectrum.doppler_shift()`.

2.1 Normalization

You can continuum normalize the spectrum using the `Spectrum.normalize()` method. It does not overwrite the spectrum but returns a new spectrum. Possible methods include `scalar`, `linear`, `quadratic`, `cubic`, `poly`, `exponential`. The `poly` method requires a `degree` to also be provided.

E.g.:

```
s = Spectrum(...)
s = s.normalize("linear")
# or
s = s.normalize("poly", degree=1)
```

The normalization happens by dividing th spectrum by the fitted continuum.

2.1.1 Continuum fitting

The fitting of the continuum is rather tricky due to the presence of absorption lines. The continuum is fitted by dividing the spectrum into N even bins. The highest M points out of each bin are chosen to represent the continuum for that bin. Their median/mean wavelength and flux position are used for each bin. A fit across the N bins is then performed using the specified method. The arguments `nbins` and `ntop` are used to specify the N and M values and can be passed to the `normalize` and `continuum` methods.

```
continuum = s.continuum(method"poly", degree=2, nbins=50, ntop=15)
```

You probably need to change these parameters to for the size/length of your spectrum.

Note: Its probably best to experiment with the best `nbins` and `ntop` parameters for your spectrum.

2.2 Overloaded Operators

The main purpose of this package is overloading the operators $+$, $-$, $\star$, $/$, $\star\star$ to work with spectrum objects. e.g:

```
# Given two spectra
s1 = Spectrum(...)
s2 = Spectrum(...)

# You can easily do the following operations.
add = s1 + s2
subtract = s1 - s2
multiply = s1 * s2
divide = s1 / s2
power = s1 ** a  # where a is a number
```

This is to make easier to do some basic manipulation of spectra, average spectra, take the difference, normalization, exponential scaling etc...

Note: Its probably best to interpolate the spectra to the same xaxis yourself before hand. If the spectra do not have the same wavelength axis then it is automatically spline interpolated to match the first spectrum or to another defined new xaxis.

CHAPTER 3

Available Classes

Currently there are two classes available.

- *Spectrum*
- *DifferentialSpectrum*

3.1 Spectrum

A class to contain a spectrum. The operators have been overloaded so that you can easily manipulate spectra.

3.2 Differential Spectrum

Compute the difference between two *Spectrum* objects.

This is in an introductory state and need more work.

Would be useful add an s-profile function from Ferluga et al. 1997. The subtraction of two gaussian lines with a RV offset.

CHAPTER 4

Other Projects

There are many other packages that deal with spectra, none that I know of overload the operators.

In alphabetical order:

- astropy/specutils
- cokelaer/spectrum
- crawfordsm/specreduce
- pyspeckit/spectrum
- PySpectrograph/Spectra

I am sure there are others.

CHAPTER 5

Contributions

Any contributions and/or suggestions to improve this module are very welcome.

Submit issues, suggestions or pull requests to jason-neal/spectrum_overload.

This is my first attempt at creating classes, and packaging a python project so any *helpful* feedback is appreciated.

CHAPTER 6

Badges

master **develop**

Note: To build the documentation locally go to the root *spectrum_overload* directory then run:

```
pip install -e .[docs]   # editable installation, install docs dependencies.
cd docs
make html
```

The documentation pages will be docs/.build/index.html.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search