PeakUtils Documentation

Release 1.3.0

Lucas Hermann Negri

Sep 06, 2018

Contents

1 Installation **3**

2 Contribute **5**

3 License **7**

4 Topics **9**

4.1 PeakUtils tutorial 9

4.2 Complete Reference 13

Python Module Index **17**

This package provides utilities related to the detection of peaks on 1D data. Includes functions to estimate baselines, finding the indexes of peaks in the data and performing Gaussian fitting or centroid computation to further increase the resolution of the peak detection.

The documentation is available at http://peakutils.readthedocs.io/en/latest .

CHAPTER 1

Installation

To install PeakUtils from the source package, run:

```
python setup.py install
```

PeakUtils targets Python 2.7+ and depends on numpy, scipy, and optionally on matplotlib.

CHAPTER 2

Contribute

- Source Code: https://bitbucket.org/lucashnegri/peakutils
- Issues: https://bitbucket.org/lucashnegri/peakutils/issues
- Direct contact: Lucas Hermann Negri - lucashnegri <at> gmail.com

CHAPTER 3

License

The project is licensed under the MIT license.

CHAPTER 4

Topics

4.1 PeakUtils tutorial

This tutorial shows the basic usage of PeakUtils to detect the peaks of 1D data.

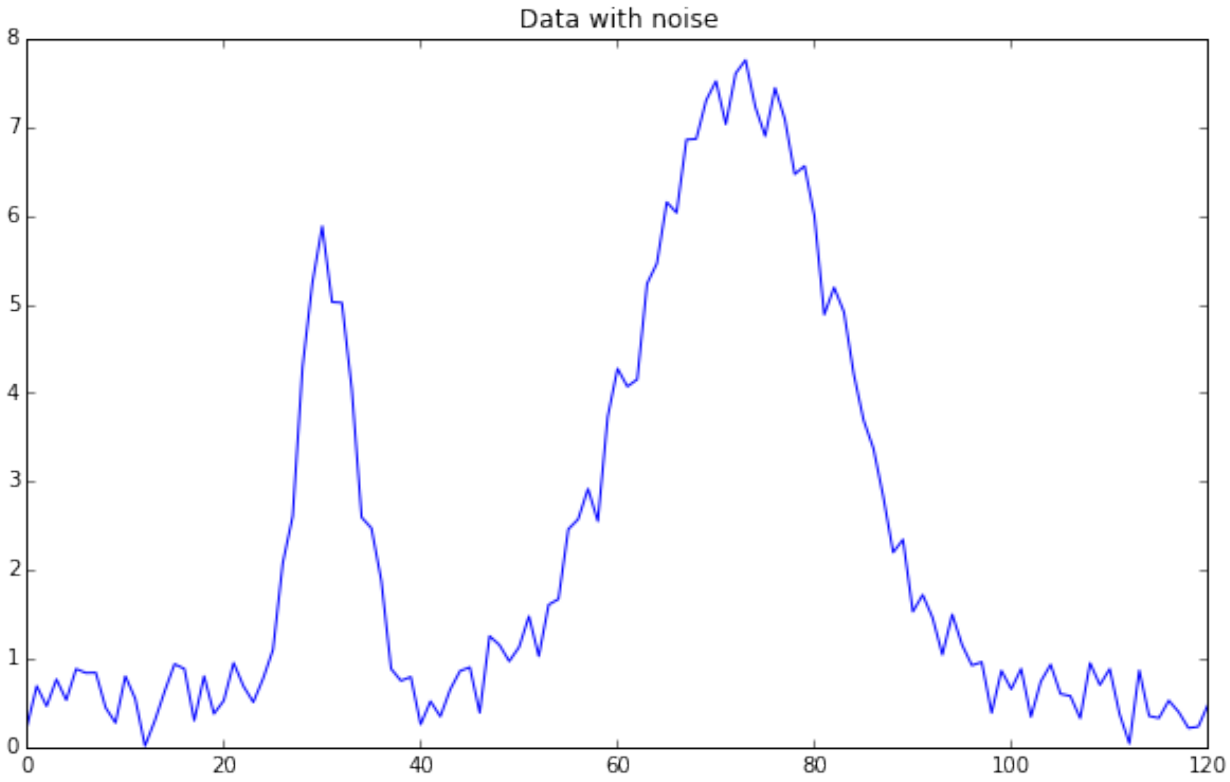
4.1.1 Importing the libraries

```
import numpy
import peakutils
from peakutils.plot import plot as pplot
from matplotlib import pyplot
%matplotlib inline
```

4.1.2 Preparing the data

Lets generate some noisy data from two Gaussians:

```
centers = (30.5, 72.3)
x = numpy.linspace(0, 120, 121)
y = (peakutils.gaussian(x, 5, centers[0], 3) +
    peakutils.gaussian(x, 7, centers[1], 10) +
    numpy.random.rand(x.size))
pyplot.figure(figsize=(10,6))
pyplot.plot(x, y)
pyplot.title("Data with noise")
```

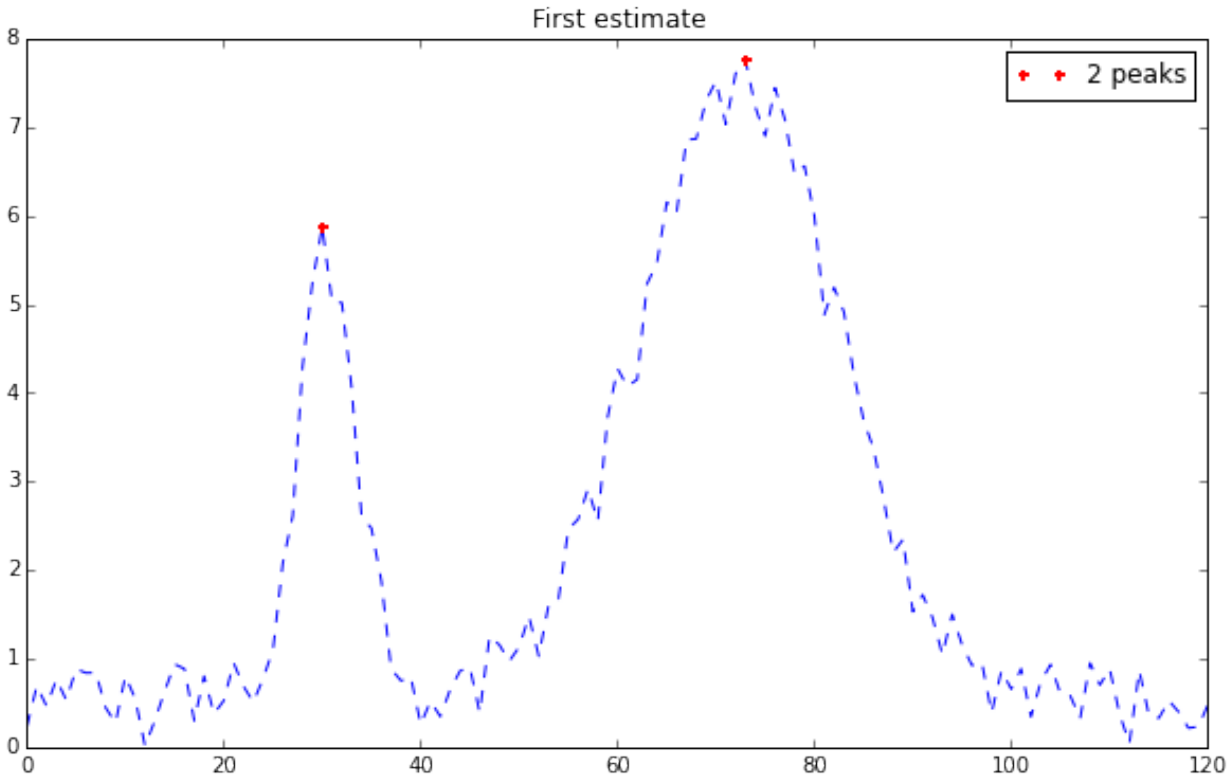


4.1.3 Getting a first estimate of the peaks

By using peakutils.indexes, we can get the indexes of the peaks from the data. Due to the noise, it will be just a rough approximation.

```
indexes = peakutils.indexes(y, thres=0.5, min_dist=30)
print(indexes)
print(x[indexes], y[indexes])
pyplot.figure(figsize=(10,6))
pplot(x, y, indexes)
pyplot.title('First estimate')
```

```
[31 74]
[ 31.  74.] [ 5.67608909  7.79403394]
```



4.1.4 Enhancing the resolution by interpolation

We can enhance the resolution by using interpolation. We will try to fit a Gaussian near each previously detected peak.

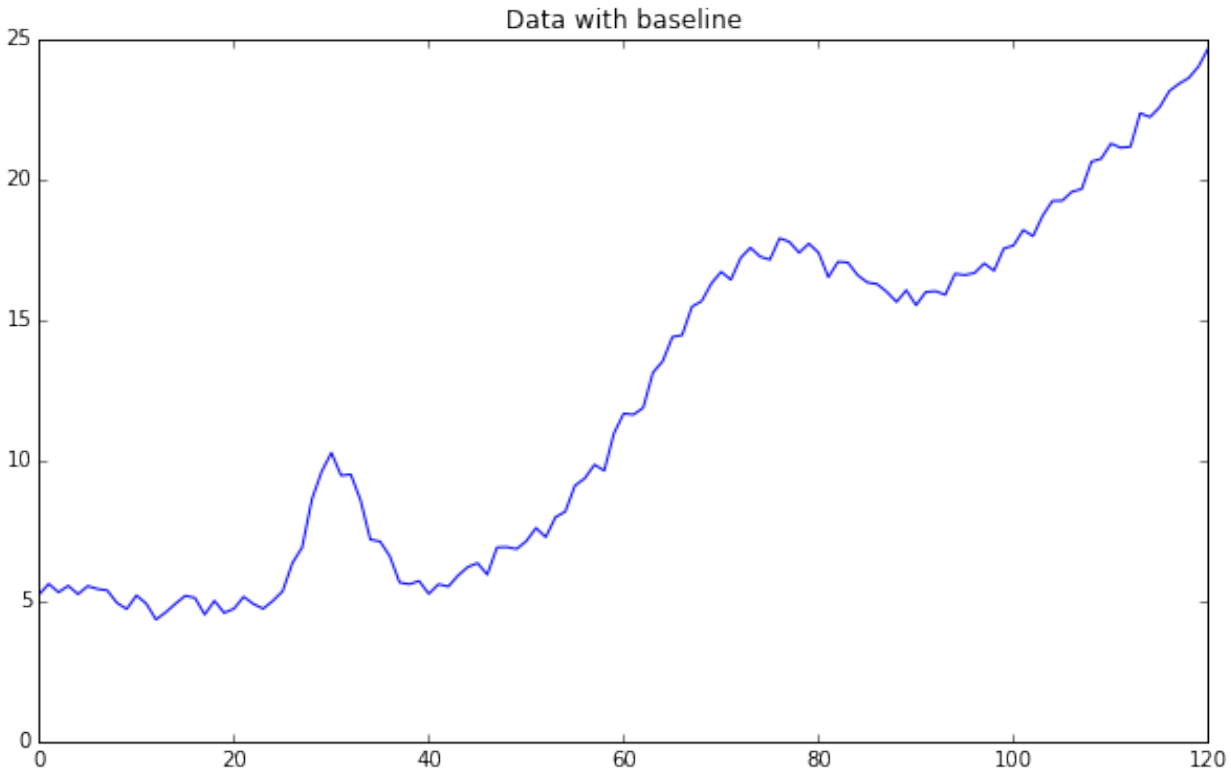
```
peaks_x = peakutils.interpolate(x, y, ind=indexes)
print(peaks_x)
```

```
[ 30.58270223  72.34348214]
```

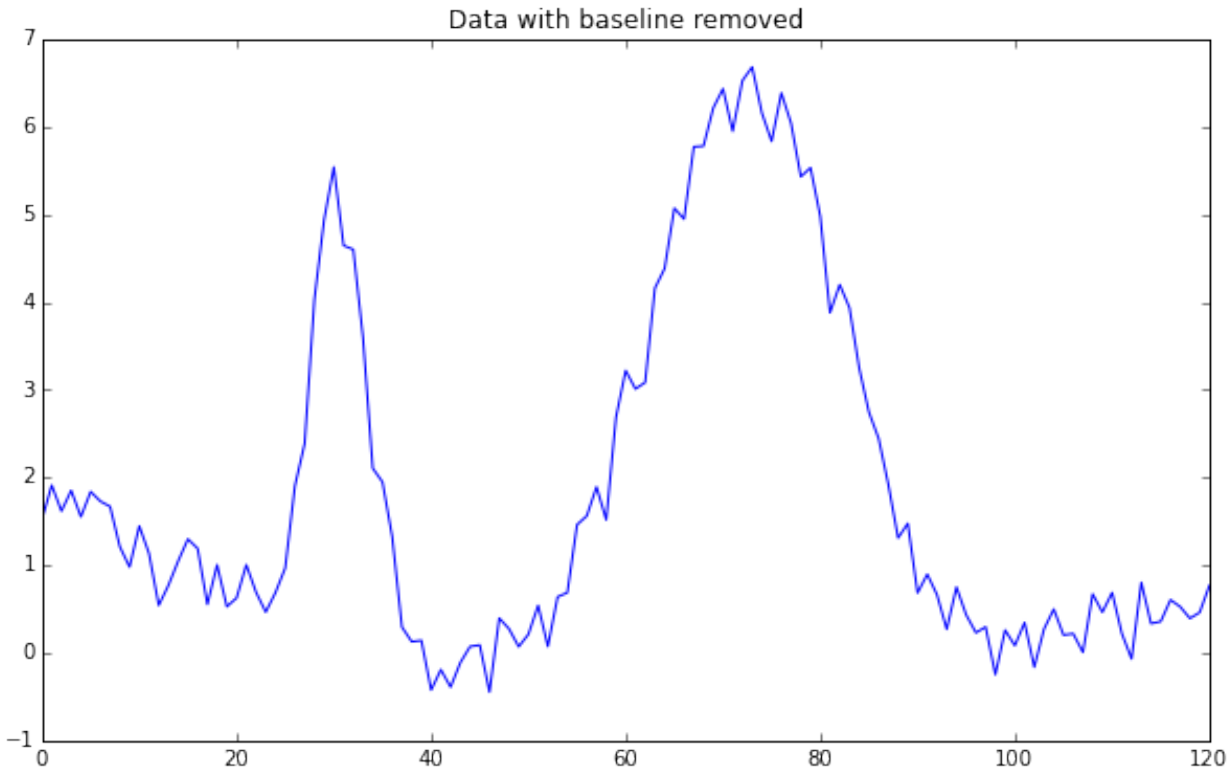
4.1.5 Estimating and removing the baseline

It is common for data to have an undesired baseline. *PeakUtils* implements a function for estimating the baseline by using an iterative polynomial regression algorithm.

```
y2 = y + numpy.polyval([0.002,-0.08,5], x)
pyplot.figure(figsize=(10,6))
pyplot.plot(x, y2)
pyplot.title("Data with baseline")
```




```
base = peakutils.baseline(y2, 2)
pyplot.figure(figsize=(10,6))
pyplot.plot(x, y2-base)
pyplot.title("Data with baseline removed")
```



4.1.6 Related functionality in SciPy

SciPy also implements functions that can be used for peak detection.

Some examples:

- scipy.signal.find_peaks_cwt
- scipy.signal.savgol_filter

4.2 Complete Reference

The contents of the submodules *baseline* and *peak* are imported to *peaktuils*, and there is no need to import them directly.

An exception is the *plot* submodule that must be explicitly imported due to its *matplotlib* dependency.

4.2.1 peakutils.baseline

Baseline estimation algorithms.

peakutils.baseline.**baseline**(*y*, *deg=None*, *max_it=None*, *tol=None*)

Computes the baseline of a given data.

Iteratively performs a polynomial fitting in the data to detect its baseline. At every iteration, the fitting weights on the regions with peaks are reduced to identify the baseline only.

Parameters

- **y** (*ndarray*) – Data to detect the baseline.
- **deg** (*int (default: 3)*) – Degree of the polynomial that will estimate the data baseline. A low degree may fail to detect all the baseline present, while a high degree may make the data too oscillatory, especially at the edges.
- **max_it** (*int (default: 100)*) – Maximum number of iterations to perform.
- **tol** (*float (default: 1e-3)*) – Tolerance to use when comparing the difference between the current fit coefficients and the ones from the last iteration. The iteration procedure will stop when the difference between them is lower than *tol*.

Returns Array with the baseline amplitude for every original point in *y*

Return type ndarray

peakutils.baseline.**envelope**(*y*, *deg=None*, *max_it=None*, *tol=None*)

Computes the upper envelope of a given data. It is implemented in terms of the *baseline* function.

Parameters

- **y** (*ndarray*) – Data to detect the baseline.
- **deg** (*int*) – Degree of the polynomial that will estimate the envelope.
- **max_it** (*int*) – Maximum number of iterations to perform.
- **tol** (*float*) – Tolerance to use when comparing the difference between the current fit coefficients and the ones from the last iteration.

Returns Array with the envelope amplitude for every original point in *y*

Return type ndarray

4.2.2 peakutils.peak

Peak detection algorithms.

peakutils.peak.**centroid**(*x*, *y*)

Computes the centroid for the specified data. Refer to centroid2 for a more complete, albeit slower version.

Parameters

- **x** (*ndarray*) – Data on the x axis.
- **y** (*ndarray*) – Data on the y axis.

Returns Centroid of the data.

Return type float

peakutils.peak.**centroid2**(*y*, *x=None*, *dx=1.0*)

Computes the centroid for the specified data. Not intended to be used

Parameters

- **y** (*array_like*) – Array whose centroid is to be calculated.
- **x** (*array_like, optional*) – The points at which y is sampled.

Returns Centroid and standard deviation of the data.

Return type (centroid, sd)

peakutils.peak.**gaussian**(*x*, *ampl*, *center*, *dev*)

Computes the Gaussian function.

Parameters

- **x** (*number*) – Point to evaluate the Gaussian for.
- **a** (*number*) – Amplitude.
- **b** (*number*) – Center.
- **c** (*number*) – Width.

Returns Value of the specified Gaussian at *x*

Return type float

peakutils.peak.**gaussian_fit**(*x*, *y*, *center_only=True*)

Performs a Gaussian fitting of the specified data.

Parameters

- **x** (*ndarray*) – Data on the x axis.
- **y** (*ndarray*) – Data on the y axis.
- **center_only** (*bool*) – If True, returns only the center of the Gaussian for *interpolate* compatibility

Returns If center_only is *False*, returns the parameters of the Gaussian that fits the specified data If center_only is *True*, returns the center position of the Gaussian

Return type ndarray or float

peakutils.peak.**indexes**(*y*, *thres=0.3*, *min_dist=1*, *thres_abs=False*)

Peak detection routine.

Finds the numeric index of the peaks in *y* by taking its first order difference. By using *thres* and *min_dist* parameters, it is possible to reduce the number of detected peaks. *y* must be signed.

Parameters

- **y** (*ndarray (signed)*) – 1D amplitude data to search for peaks.
- **thres** (*float between [0., 1.]*) – Normalized threshold. Only the peaks with amplitude higher than the threshold will be detected.
- **min_dist** (*int*) – Minimum distance between each detected peak. The peak with the highest amplitude is preferred to satisfy this constraint.
- **thres_abs** (*boolean*) – If True, the thres value will be interpreted as an absolute value, instead of a normalized threshold.

Returns Array containing the numeric indexes of the peaks that were detected

Return type ndarray

peakutils.peak.**interpolate**(*x*, *y*, *ind=None*, *width=10*, *func=<function gaussian_fit>*)

Tries to enhance the resolution of the peak detection by using Gaussian fitting, centroid computation or an arbitrary function on the neighborhood of each previously detected peak index.

RuntimeErrors raised in the fitting function will be converted to warnings, with the peak being mantained as the original one (in the ind array).

Parameters

- **x** (*ndarray*) – Data on the x dimension.
- **y** (*ndarray*) – Data on the y dimension.

- **ind** (*ndarray*) – Indexes of the previously detected peaks. If None, indexes() will be called with the default parameters.
- **width** (*int*) – Number of points (before and after) each peak index to pass to *func* in order to increase the resolution in *x*.
- **func** (*function(x,y)*) – Function that will be called to detect an unique peak in the x,y data.

Returns Array with the adjusted peak positions (in *x*)

Return type ndarray

4.2.3 peakutils.prepare

Data preparation / preprocessing algorithms.

peakutils.prepare.**scale**(*x*, *new_range=(0.0, 1.0)*, *eps=1e-09*)

Changes the scale of an array

Parameters

- **x** (*ndarray*) – 1D array to change the scale (remains unchanged)
- **new_range** (*tuple (float, float)*) – Desired range of the array
- **eps** (*float*) – Numerical precision, to detect degenerate cases (for example, when every value of *x* is equal)

Returns

- *ndarray* – Scaled array
- *tuple (float, float)* – Previous data range, allowing a rescale to the old range

4.2.4 peakutils.plot

Python Module Index

p

peakutils.baseline, 13
peakutils.peak, 14
peakutils.prepare, 16

Index

B

baseline() (in module peakutils.baseline), 13

C

centroid() (in module peakutils.peak), 14
centroid2() (in module peakutils.peak), 14

E

envelope() (in module peakutils.baseline), 14

G

gaussian() (in module peakutils.peak), 14
gaussian_fit() (in module peakutils.peak), 15

I

indexes() (in module peakutils.peak), 15
interpolate() (in module peakutils.peak), 15

P

peakutils.baseline (module), 13
peakutils.peak (module), 14
peakutils.prepare (module), 16

S

scale() (in module peakutils.prepare), 16