DIMEpy Documentation

Release 1.0.0

Keiron O'Shea

Aug 09, 2019

Contents:

Python package for the high-throughput nontargeted metabolite fingerprinting of nominal mass direct injection mass spectrometry directly from mzML files.

This work is very much inspired by the methods detailed in High-throughput, nontargeted metabolite fingerprinting using nominal mass flow injection electrospray mass spectrometry (Beckmann, et al, 2008).

CHAPTER 1

Features

- Loading mass spectrometry files from mzML.
  - Support for polarity switching.
  - MAD-estimated infusion profiling.
- Assay-wide outlier spectrum detection.
- Spurious peak elimination.
- Spectrum export for direct dissemination using Metaboanalyst.
- Spectral binning.
- Value imputation.
- Spectral normalisation.
  - including TIC, median, mean...
- Spectral transformation.
  - including log10, cube, nlog, log2, glog, sqrt, ihs...
- Export to array for statistical analysis in Metaboanalyst.

CHAPTER 2

Contributors

- **Lead Developer:** Keiron O'Shea (keo7@aber.ac.uk)
- **Developer:** Rob Bolton (rab26@aber.ac.uk)
- **Project Supervisor:** Chuan Lu (cul@aber.ac.uk)
- **Project Supervisor:** Luis AJ Mur (lum@aber.ac.uk)
- **Methods Expert:** Manfred Beckmann (meb@aber.ac.uk)

CHAPTER 3

License

DIMEpy is licensed under the GNU General Public License v3.0.

3.1 Installation

DIMEpy requires Python 3+ and is unfortunately not compatible with Python 2. If you are still using Python 2, a clever workaround is to install Python 3 and use that instead.

You can install it through `pypi` using `pip`:

```
pip install dimepy
```

If you want the 'bleeding edge' version this, you can also install directly from this repository using `git` - but beware of dragons:

```
pip install git+https://www.github.com/AberystwythSystemsBiology/DIMEpy
```

3.2 Getting Started

To use the package, type the following into your Python console:

```
>>> import dimepy
```

At the moment, this pipeline only supports mzML files. You can easily convert proprietary formats to mzML using ProteoWizard.

3.2.1 Loading a single file

If you are only going to load in a single file for fingerprint matrix estimation, then just create a new spectrum object. If the sample belongs to a characteristic, it is recommend that you also pass it through when instantiating a new

`Spectrum` object.

```
>>> filepath = "/file/to/file.mzML"
>>> spec = dimepy.Spectrum(filepath, identifier="example", stratification="class_one")
/file/to/file.mzML
```

By default the Spectrum object doesn't set a snr estimator. It is **strongly recommended** that you set a signal to noise estimation method when instantiating the Spectrum object.

If your experimental protocol makes use of mixed-polarity scanning, then please ensure that you limit the scan ranges to best match what polarity you're interested in analysing:

```
>>> spec.limit_polarity("negative")
```

If you are using FIE-MS it is strongly recommended that you use just the infusion profile to generate your mass spectrum. For example, if your scan profiles look like this:

```
  |          _
T |         / \
I |        /   \_
C |_______/      \____________________
  0      0.5     1      1.5      2 [min]
```

Then it is fair to assume that the infusion occured during the scans ranging from 30 seconds to 1 minute. The `limit_infusion()` method does this by estimating the median absolute deviation (MAD) of total ion count (TIC) before limiting the profile to the range between the time range in which whatever multiple of MAD has been estimated:

```
>>> spec.limit_infusion(2) # 2 times the MAD.
```

Now, we are free to load in the scans to generate a base mass_spectrum:

```
>>> spec.load_scans()
```

You should now be able to access the generated mass spectrum using the `masses` and `intensities` attributes:

```
>>> spec.masses
array([ ... ])
>>> spec.intensities
array([ ... ])
```

3.2.2 Working with multiple files

A more realistic pipeline would be to use multiple mass-spectrum files. This is where things really start to get interesting. The `SpectrumList` object facilitates this through the use of the `append` method:

```
>>> speclist = dimepy.SpectrumList()
>>> speclist.append(spec)
```

You can make use of an iterator to recursively generate `Spectrum` objects, or do it manually if you want.

If you're only using this pipeline to extract mass spectrum for Metabolanalyst, then you can now simply call the `_to_csv` method:

```
>>> speclist.to_csv("/path/to/output.csv", output_type="metaboanalyst")
```

That being said, this pipeline contains many of the preprocessing methods found in Metaboanalyst - so it may be easier for you to just use ours.

As a diagnostic measure, the TIC can provide an estimation of factos that may adversely affect the overal intensity count of a run. As a rule, it is common to remove spectrum in which the TIC deviates 2/3 times from the median-absolute deviation. We can do this by calling the `detect_outliers` method:

```
>>> speclist.detect_outliers(thresh = 2, verbose=True)
Detected Outliers: outlier_one;outlier_two
```

A common first step in the analysis of mass-spectrometry data is to bin the data to a given mass-to-ion value. To do this for all `Spectrum` held within our `SpectrumList` object, simply apply the `bin` method:

```
>>> speclist.bin(0.25) # binning our data to a bin width of 0.25 m/z
```

In FIE-MS null values should concern no more than 3% of the total number of identified bins. However, imputation is required to streamline the analysis process (as most multivariate techniques are unable to accomodate missing data points). To perform value imputation, just use `value_imputate`:

```
>>> speclist.value_imputate()
```

Now transforming and normalisating the the spectrum objects in an samples independent fashion can be done using the following:

```
>>> speclist.transform()
>>> speclist.normalise()
```

Once completed, you are now free to export the data to a data matrix:

```
>>> speclist.to_csv("/path/to/proc_metabo.csv", output_type="matrix")
```

This should give you something akin to:

Sample ID	M0	M1	M2	M3	...
Sample 1	213	634	3213	546	...
Sample 2	132	34	713	6546	...
Sample 3	1337	42	69	420	...

3.3 Modules

3.3.1 dimepy.Spectrum

Initialise Spectrum object for a given mzML file.

class dimepy.**Spectrum**(*filepath: str*, *identifier: str*, *injection_order: int = None*, *stratification: str = None*, *snr_estimator: str = False*, *peak_type: str = 'raw'*, *MS1_precision: float = 5e-06*, *MSn_precision: float = 2e-05*)

Initialise Spectrum object for a given mzML file.

__init__(*filepath: str*, *identifier: str*, *injection_order: int = None*, *stratification: str = None*, *snr_estimator: str = False*, *peak_type: str = 'raw'*, *MS1_precision: float = 5e-06*, *MSn_precision: float = 2e-05*)

Initialise a Spectrum object for a given mzML file.

Arguments: filepath (str): Path to the mzML file to parse.

identifier (str): Unique identifier for the Spectrum object.

injection_order (int): The injection number of the Spectrum object.

stratification (str): Class label of the Spectrum object.

snr_estimator (str): Signal to noise method used to filter.

Currently supported signal-to-noise estimation methods are:

- 'median' (default)
- 'mean'
- 'mad'

peak_type (raw): What peak type to load in.

Currently supported peak types are:

- raw (default)
- centroided
- reprofiled

MS1_precision (float): Measured precision for the MS level 1.

MSn_precision (float): Measured precision for the MS level n.

bin (*bin_width: float = 0.01*, *statistic: str = 'mean'*)

" Method to conduct mass binning to nominal mass and mass spectrum generation across a Spectrum.

Arguments: bin_width (float): The mass-to-ion bin-widths to use for binning.

statistic (str): The statistic to use to calculate bin values.

Supported statistic types are:

- **'mean' (default): compute the mean of intensities for points within each bin.** Empty bins will be represented by NaN.
- **'std': compute the standard deviation within each bin. This is** implicitly calculated with ddof=0.
- **'median': compute the median of values for points within each bin.** Empty bins will be represented by NaN.
- **'count': compute the count of points within each bin.** This is identical to an unweighted histogram. values array is not referenced.
- **'sum': compute the sum of values for points within each bin.** This is identical to a weighted histogram.
- **'min': compute the minimum of values for points within each bin.** Empty bins will be represented by NaN.
- **'max': compute the maximum of values for point within each bin.** Empty bins will be represented by NaN.

limit_infusion (*threshold: int = 3*) → None

This method is a slight extension of Manfred Beckmann's (meb@aber.ac.uk) LCT/Q-ToF scan retrieval method in FIEMSpro in which we use the median absolute deviation of all TICs within a Spectrum to determine when the infusion has taken place.

Consider the following Infusion Profile:

```
      _
    /  \
  /     \_
```

(continues on next page)

(continued from previous page)

```
____/        \___________________
0     0.5     1     1.5     2 [min]
    |--------| Apex
```

We are only interested in the scans in which the infusion takes place (20 - 50 seconds). Applying this method changes the to_use values to only be True where the TIC is >= TIC * mad_multiplier.

Arguments:

mad_multiplier (int): The multiplier for the median absolute deviation method to take the infusion profile from.

limit_polarity (*polarity: str*) → None
Limit the Scans found within the mzML file to whatever polarity is given. This should only be called where fast-polarity switching is used.

Arguments: polarity (str): polarity type of the scans required

Supported polarity types are:

- 'positive'
- 'negative'

load_scans () → None
This method loads the scans in accordance to whatever Scans are set to True in the to_use list.

Note: If you want to actually make use of masses and intensities (you probably do), then ensure that you call this method.

---

remove_spurious_peaks (*bin_width: float = 0.01*, *threshold: float = 0.25*, *scan_grouping: float = 50.0*)
Method that's highly influenced by Jasen Finch's (jsf9@aber.ac.uk) binneR, in which spurios peaks can be removed. At the time of writing, this method has serious performance issues and needs to be rectified. but should still work as intended (provided that you don't mind how long it takes to complete)

Arguments: bin_width (float): The mass-to-ion bin-widths to use for binning.

threshold (float): Percentage of scans in which a peak must be in in order for it to be considered.

scan_grouping (float): Mass-to-ion scan groups, this splits the scans into groups to ease the processing somewhat. It is strongly recommended that you keep this at it's default value of of 50.0

Note: load_scans() must first be run in order for this to work.

---

reset () → None
A method to reset the Spectrum object in its entirety.

3.3.2 dimepy.SpectrumList

class dimepy.**SpectrumList**

__init__ ()
Initialize self. See help(type(self)) for accurate signature.

append(*spectrum: dimepy.spectrum.Spectrum*)
Method to append a Spectrum to the SpectrumList.

Arguments: spectrum (Spectrum): A Spectrum object.

bin(*bin_width: float = 0.5*, *statistic: str = 'mean'*)
Method to conduct mass binning to nominal mass and mass spectrum generation across a SpectrumList.

Arguments: bin_width (float): The mass-to-ion bin-widths to use for binning.

statistic (str): The statistic to use to calculate bin values.

Supported statistic types are:

- **'mean' (default): compute the mean of intensities for points within each bin.** Empty bins will be represented by NaN.
- **'std': compute the standard deviation within each bin. This is** implicitly calculated with ddof=0.
- **'median': compute the median of values for points within each bin.** Empty bins will be represented by NaN.
- **'count': compute the count of points within each bin.** This is identical to an unweighted histogram. values array is not referenced.
- **'sum': compute the sum of values for points within each bin.** This is identical to a weighted histogram.
- **'min': compute the minimum of values for points within each bin.** Empty bins will be represented by NaN.
- **'max': compute the maximum of values for point within each bin.** Empty bins will be represented by NaN.

detect_outliers(*threshold: float = 1*, *verbose: bool = False*)
Method to locate and remove outlier spectrum using the median-absolute deviation of the TICS within the SpectrumList.

Note: This method is still being actively developed, so is likely to change.

Arguments: threshold (int): Threshold for MAD outlier detection.

verbose (bool): Whether to print out the identifiers of the removed Spectrum.

normalise(*method: str = 'tic'*) → None
Method to conduct sample independent intensity normalisation.

Arguments: method (str): The normalisation method to use.

Currently supported normalisation methods are:

- **'tic' (default): Normalise to the total ion current** of the Spectrum:
- 'median': Normalise to the meidan of the Spectrum.
- 'mean': Normalise to the mean of the Spectrum.

to_csv(*fp: str*, *sep: str = ', '*, *output_type: str = 'base'*)
Method to export the spectrum list.

Arguments: fp (str): Filepath to export the file to.

sep (str): Separator to use for file export

output_type (str): What form of output to export:

Supported output types are:

- **'base': masses and intensities of each spectrum in a column each** in a single CSV file.
- **'matrix': The way in which I personally analyse the data.** This will not work unless the data has been binned.
- 'metaboanalyst': A zipfile ready for uploading to metaboanalyst.

transform (*method: str = 'log10'*) → None
Method to conduct sample independent intensity transformation.

Arguments: method (str): The transformation method to use.

Currently supported transformation methods are: 'log10' (default) 'cube' 'nlog' 'log2' 'glog' 'sqrt' 'ihs'

value_imputate (*method: str = 'min', threshold: float = 0.5*) → None
A method to deploy value imputation to the Spectrum List.

Note: As most metabolite selection methods fail to deal with missing values, it is strongly recommended to run this method once binning has been performed over the SpectrumList

---

Arguments: method (str): Method to use for value imputation.

Currently supported value imputation methods are:

- **'basic' (default)** [Replace thresholded null values] with half the minimum intensity value per Spec
- **'mean': Replace thresholded null values with the** mean intensity value per Spec.
- **'min': Replace thresholded null values with the** minimum intensity value per Spec.
- **'median': Replace thresholded null values with the** minimum intensity value per Spec.

threshold (float): Number of samples an intensity needs to be present in to be taken forward for imputation.

3.3.3 dimepy.Scan

class dimepy.**Scan** (*pymzml_spectrum, snr_estimator: str = False, peak_type: str = 'raw'*)

__init__ (*pymzml_spectrum, snr_estimator: str = False, peak_type: str = 'raw'*)
Initialise a Scan object for a given pymzML Spectrum.

Arguments:

pymzml_spectrum (pymzml.Spectrum): Spectrum object.

snr_estimator (str): Signal to noise method used to filter.

peak_type (str): Peaks to take forward.

bin(*bin_width: float = 0.01*, *statistic: str = 'mean'*) → None

Method to conduct mass binning to nominal mass and mass spectrum generation.

Arguments: bin_width (float): The mass-to-ion bin-widths to use for binning.

statistic (str): The statistic to use to calculate bin values.

Supported statistic types are:

- **'mean' (default): compute the mean of intensities for points within each bin.** Empty bins will be represented by NaN.
- **'std': compute the standard deviation within each bin. This is** implicitly calculated with ddof=0.
- **'median': compute the median of values for points within each bin.** Empty bins will be represented by NaN.
- **'count': compute the count of points within each bin.** This is identical to an unweighted histogram. values array is not referenced.
- **'sum': compute the sum of values for points within each bin.** This is identical to a weighted histogram.
- **'min': compute the minimum of values for points within each bin.** Empty bins will be represented by NaN.
- **'max': compute the maximum of values for point within each bin.** Empty bins will be represented by NaN.

3.4 Example Scripts

Here's a quick run through of key functionality in DIMEpy.l

```
import dimepy
import os

data_dir = "/path/to/mzMLs"

sl = dimepy.SpectrumList()

for fn in os.listdir(data_dir):
     # Appending the file name to the data directory
     fp = os.path.join(data_dir,fn)

     # We only have two classes, denoted by the first letter of the file
     strat = fn[0]

     # Note how I'm applying mean-based snr estimation
     spec = dimepy.Spectrum(fp, fn, stratification=strat, snr_estimator="mean")

     # As these are polarity-switching, I'm limiting to positive
     spec.limit_polarity("positive")
```

(continues on next page)

(continued from previous page)

```
    # A threshold of 3 seemed fine from earlier.
    spec.limit_infusion(3)

    # Make sure I load the scans
    spec.load_scans()

    # Appending the Spectrum object to the SL
    sl.append(spec)

sl.detect_outliers(3, verbose=True)

sl.bin(0.25)
sl.value_imputate()
sl.transform()
sl.normalise()

sl.to_csv("/path/to/output.csv", output_type="matrix")
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index

Symbols

A

B

D

L

N

R

S

T

V