spectrum_utils: A Python package for mass spectrometry data processing and visualization

Wout Bittremieux[1,2,3,*]

[1]Skaggs School of Pharmacy and Pharmaceutical Sciences, University of California San Diego, La Jolla, CA 92093, USA; [2]Department of Mathematics and Computer Science, University of Antwerp, 2020 Antwerp, Belgium; [3]Biomedical Informatics Network Antwerpen (biomina), 2020 Antwerp, Belgium

*Corresponding author: wout.bittremieux@uantwerpen.be, +32 3 265 34 07.

Abstract

`spectrum_utils` is a Python package for mass spectrometry data processing and visualization. To deal with the large volumes of data that can be generated during mass spectrometry experiments, data processing is highly optimized for computational efficiency. Additionally, `spectrum_utils` can be used to produce publication-ready spectrum plots as well as interactive spectrum plots for inclusion on web pages.
`spectrum_utils` is available for Python 3.6+, includes extensive online documentation and examples, and can be easily installed via pip or conda. It is freely available as open source under the Apache 2.0 license at `https://github.com/bittremieux/spectrum_utils`.

1 Introduction

Mass spectrometry (MS) is a powerful, high-throughput analytical technique that can be used to identify and quantify proteins and other molecules in complex biological samples. Because during a typical MS experiment tens of thousands of mass spectra are generated, suitable bioinformatics tools are needed to analyze such large data volumes. MS data processing has traditionally been done using monolithic software solutions that aim to provide fully end-to-end solutions from the raw data to the final identification or quantification results. However, because there exist a large variety of experimental set-ups and configurations, such tools necessarily cannot cover all possible use cases.

Instead, customized data analysis workflows are often needed to fully interpret the results from an MS experiment. In recent years several software packages for the general-purpose analysis of MS data in popular scripting languages have been developed. These include, for example, MSnbase [1] for MS data processing, visualization and quantification in R; Pyteomics [2, 3] for a variety of proteomics data processing tasks in Python, pyOpenMS [4] to expose the rich functionality of OpenMS [5] from C++ to Python, pymzML [6, 7] to efficiently read and process spectra in the mzML format [8] using Python, etc.

Here we present the `spectrum_utils` package for MS data processing and visualization in Python. `spectrum_utils` allows the user to easily manipulate mass spectral data and quickly prototype ideas for computational MS projects. A key feature of `spectrum_utils` is its focus on computational efficiency to process large amounts of spectral data. `spectrum_utils` is freely available as open source under the Apache 2.0 license at `https://github.com/bittremieux/spectrum_utils`.

2 Methods

The functionality provided by `spectrum_utils` is built around the concept of tandem mass spectrometry (MS/MS) spectra as basic elements. An MS/MS spectrum is defined by its precursor m/z and precursor charge, and its m/z fragments. Additionally, a peptide sequence, potentially including modifications, can be provided. Various processing steps operating on MS/MS spectra are available (figure 1). Uninformative peaks, such as peaks outside a specified mass range, can be removed. Additionally, the precursor ion and its isotopic peaks can be removed, as well as low-intensity peaks (relative to the base peak intensity). Further peak filtering can be performed by only retaining the top most intense peaks. Next, peak intensities can be scaled to de-emphasize overly dominant peaks. Possible transformations are root scaling (default square root), log scaling (default $\log_2$), or rank-based scaling. Finally, fragment peaks can be annotated based on the provided peptide sequence, including post-translational modifications (PTMs). Fragment masses of unmodified peptides are calculated using Pyteomics [2, 3], and `spectrum_utils` provides a user-friendly interface to globally modify the mass of an amino acid to specify static modifications and to set dynamic modifications on amino acids at specific positions for individual peptides.

An important emphasis is placed on the computational efficiency of the spectrum processing steps. Operations on the peaks of MS/MS spectra are implemented using NumPy [9], a popular Python li-

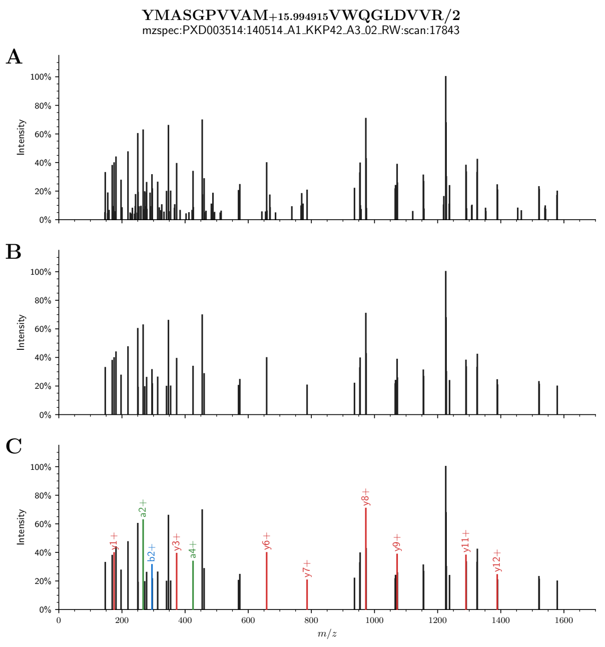

Figure 1: MS/MS spectrum processing. (a) The raw spectrum is processed by (b) removing low-intensity noise peaks. (c) Peak fragments are annotated based on the peptide sequences, including PTMs.

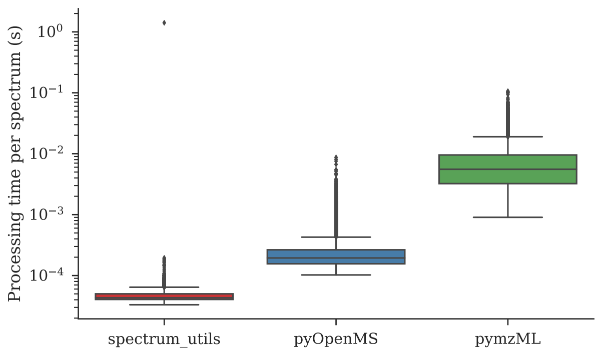

Figure 2: Spectrum processing runtime comparison. The runtime for processing the spectra generated for the iPRG2012 challenge [11] is reported for similar processing steps for `spectrum_utils`, pyOpenMS, and pymzML. Spectrum processing consisted of fixing the m/z range, removing precursor ion peaks (not supported by pymzML) and low-intensity noise peaks, and scaling the peak intensities by their square root. Note that the significant outlier for `spectrum_utils` is caused by Numba's JIT compilation of the first method calls, allowing subsequent calls to be made very efficiently.

brary for efficient numerical computation. Additionally, Numba [10], a Python just-in-time (JIT) compiler, is used to further speed up many processing steps by compiling Python and NumPy code into efficient machine instructions. Using these highly efficient techniques for numerical computation, `spectrum_utils` is able to speed up spectrum processing by up to orders of magnitude compared to similar alternatives (figure 2).

The MS/MS spectra can be easily visualized using `spectrum_utils`'s plotting functionality (figure 1). Spectra can be plotted directly or their peaks can be annotated with the appropriate fragments if the peptide sequence is known. Additionally, a mirror plot can be used to clearly show the similarity between two different spectra, for example during spectral library searching. The default plotting functionality uses matplotlib [12] to generate high-quality, publication-ready spectrum plots. Additionally, interactive plotting functionality is available using Altair [13], which is based on the Vega and Vega-Lite grammar of interactive graphics [14]. Interactive plotting is a drop-in replacement for the standard plotting functionality, allowing the user to trivially produce interactive spectrum plots to visualize MS/MS spectra on web pages or in Jupyter notebooks [15].

2.1 Code availability

`spectrum_utils` is available for Python 3.6+ and can easily be installed via pip or conda. `spectrum_utils` depends on Numpy [9] and Numba [10] for efficient numerical computation, Pyteomics [2, 3] for fragment ion mass calculations, matplotlib [12] for static plotting, and Altair [13] and Pandas [16] for interactive plotting.

All code and detailed documentation on how to use `spectrum_utils` is freely available as open source under the Apache 2.0 license at `https://github.com/bittremieux/spectrum_utils`.

3 Conclusion

Here we have presented the `spectrum_utils` package for MS data processing and visualization in Python. Its clearly defined functionality allows `spectrum_utils` to fill an important gap in the Python MS processing ecosystem. For example, `spectrum_utils` does not provide functionality to read spectral data files because there already exist several excellent tools to do this. Instead, `spectrum_utils` takes the MS data provided by these other tools as input to subsequently be processed. `spectrum_utils` has a well-defined, Pythonic application programming interface, allowing developers to easily harness its powerful functionality in a low number of lines of code. Additionally, it is trivial to switch between the static plotting functionality to produce high-quality spectrum plots to include in scientific manuscripts and the interactive plotting functionality. This interactive plotting functionality can be very powerful during an explorative analysis of MS data or to include dynamic spectrum plots in web resources.

`spectrum_utils` has already been used in a wide variety of computational MS projects, showcasing its versatile functionality. Among others, it has been used to process spectra and visualize spectrum identifications in the ANN-SoLo open modification spectral library search engine [17, 18], to perform a specialized analysis of an unknown protein sample [19], and to generate MS/MS spectra visualizations that have been included in scientific manuscripts and presentations. These examples illustrate how `spectrum_utils` facilitates several MS data processing tasks and allows developers to quickly prototype ideas for computational MS projects.

Acknowledgement

W.B. is a postdoctoral researcher of the Research Foundation – Flanders (FWO).

References

[1] Gatto, L. and Lilley, K. S. “MSnbase-an R/Bioconductor Package for Isobaric Tagged Mass Spectrometry Data Visualization, Processing and Quantitation.” In: *Bioinformatics* 28.2 (Jan. 15, 2012), pp. 288–289. DOI: 10.1093/bioinformatics/btr645.

[2] Goloborodko, A. A. et al. “Pyteomics-a Python Framework for Exploratory Data Analysis and Rapid Software Prototyping in Proteomics.” In: *Journal of The American Society for Mass Spectrometry* 24.2 (Feb. 1, 2013), pp. 301–304. DOI: 10.1007/s13361-012-0516-6.

[3] Levitsky, L. I. et al. “Pyteomics 4.0: Five Years of Development of a Python Proteomics Framework.” In: *Journal of Proteome Research* 18.2 (Feb. 1, 2019), pp. 709–714. DOI: 10.1021/acs.jproteome.8b00717.

[4] Röst, H. L. et al. “OpenMS: A Flexible Open-Source Software Platform for Mass Spectrometry Data Analysis.” In: *Nature Methods* 13.9 (Aug. 30, 2016), pp. 741–748. DOI: 10.1038/nmeth.3959.

[5] Röst, H. L. et al. “pyOpenMS: A Python-Based Interface to the OpenMS Mass-Spectrometry Algorithm Library.” In: *PROTEOMICS* 14.1 (Jan. 2014), pp. 74–77. DOI: 10.1002/pmic.201300246.

[6] Bald, T. et al. “pymzML–Python Module for High-Throughput Bioinformatics on Mass Spectrometry Data.” In: *Bioinformatics* 28.7 (Apr. 1, 2012), pp. 1052–1053. DOI: 10.1093/bioinformatics/bts066.

[7] Kösters, M. et al. “pymzML v2.0: Introducing a Highly Compressed and Seekable Gzip Format.” In: *Bioinformatics* 34.14 (July 15, 2018). Ed. by Wren, J., pp. 2513–2514. DOI: 10.1093/bioinformatics/bty046.

[8] Martens, L. et al. “mzML—a Community Standard for Mass Spectrometry Data.” In: *Molecular & Cellular Proteomics* 10.1 (Jan. 1, 2011), R110.000133–R110.000133. DOI: 10.1074/mcp.R110.000133.

[9] Van der Walt, S., Colbert, S. C., and Varoquaux, G. “The NumPy Array: A Structure for Efficient Numerical Computation.” In: *Computing in Science & Engineering* 13.2 (Mar. 2011), pp. 22–30. DOI: 10.1109/MCSE.2011.37.

[10] Lam, S. K., Pitrou, A., and Seibert, S. “Numba: A LLVM-Based Python JIT Compiler.” In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC - LLVM ’15*. Austin, TX, USA: ACM Press, Nov. 15, 2015, pp. 1–6. DOI: 10.1145/2833157.2833162.

[11] Chalkley, R. J. et al. “Proteome Informatics Research Group (iPRG)_2012: A Study on Detecting Modified Peptides in a Complex Mixture.” In: *Molecular & Cellular Proteomics* 13.1 (Jan. 1, 2014), pp. 360–371. DOI: 10.1074/mcp.M113.032813.

[12] Hunter, J. D. “Matplotlib: A 2D Graphics Environment.” In: *Computing in Science & Engineering* 9.3 (June 18, 2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[13] *Altair: Declarative visualization in Python*. https://altair-viz.github.io/. (accessed: 2019-08-02).

[14] Satyanarayan, A. et al. “Vega-Lite: A Grammar of Interactive Graphics.” In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (Aug. 10, 2016), pp. 341–350. DOI: 10.1109/TVCG.2016.2599030.

[15] Thomas, K. et al. “Jupyter Notebooks – A Publishing Format for Reproducible Computational Workflows.” In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. IOS Press, 2016, pp. 87–90.

[16] McKinney, W. “Data Structures for Statistical Computing in Python.” In: *Proceedings of the 9th Python in Science Conference*. Ed. by van der Walt, S. and Millman, J. Austin, Texas, USA, 2010, pp. 51–56.

[17] Bittremieux, W. et al. “Fast Open Modification Spectral Library Searching through Approximate Nearest Neighbor Indexing.” In: *Journal of Proteome Research* 17.10 (Oct. 5, 2018), pp. 3463–3474. DOI: 10.1021/acs.jproteome.8b00359.

[18] Bittremieux, W., Laukens, K., and Noble, W. S. “Extremely Fast and Accurate Open Modification Spectral Library Searching of High-Resolution Mass Spectra Using Feature Hashing and Graphics Processing Units.” In: *bioRxiv* (May 5, 2019). DOI: 10.1101/627497.

[19] Pino, L., Lin, A., and Bittremieux, W. "2018 YPIC Challenge: A Case Study in Characterizing an Unknown Protein Sample." In: *PeerJ Preprints* (June 14, 2019). DOI: 10.7287/peerj.preprints.27802v1.