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: wbittremieux@health.ucsd.edu

Abstract

Given the wide diversity in applications of biological mass spectrometry, custom data analyses are often needed to fully interpret the results of an experiment. Such bioinformatics scripts necessarily include similar basic functionality to read mass spectral data from standard file formats, process it, and visualize it. Rather than having to reimplement this functionality, to facilitate this task, `spectrum_utils` is a Python package for mass spectrometry data processing and visualization. Its high-level functionality enables developers to quickly prototype ideas for computational mass spectrometry projects in only a few lines of code. Notably, the data processing functionality is highly optimized for computational efficiency to be able to deal with the large volumes of data that are generated during mass spectrometry experiments. The visualization functionality makes it possible to easily produce publication-quality figures 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 using 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 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 tools 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 of an MS experiment. In recent years several software packages for the general-purpose analysis of MS data in popular scripting languages have been developed. Notable examples include MSnbase [6] for data processing, visualization, and quantification in R; pymzML [1, 10] to efficiently read and process spectra in the mzML format [13] using Python; Pyteomics [7, 12] for a variety of proteomics data processing tasks in Python; and pyOpenMS [20] to expose the rich functionality of OpenMS [19, 22] from C++ to Python.

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 & results

The functionality provided by `spectrum_utils` is built around the concept of tandem mass spectrometry (MS/MS) spectra as basic elements. MS/MS spectra can be processed in various ways. Uninformative peaks, such as the precursor ion and its isotopic peaks, and low-intensity noise peaks, can be removed. 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, log scaling, or rank-based scaling. Finally, peaks can be annotated with their peptide fragments, potentially including post-translational modifications (PTMs) at specified amino acid positions, molecules encoded as SMILES strings, or custom annotations (figure 1).

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 [23], a popular Python library for efficient numerical computation. Addi-

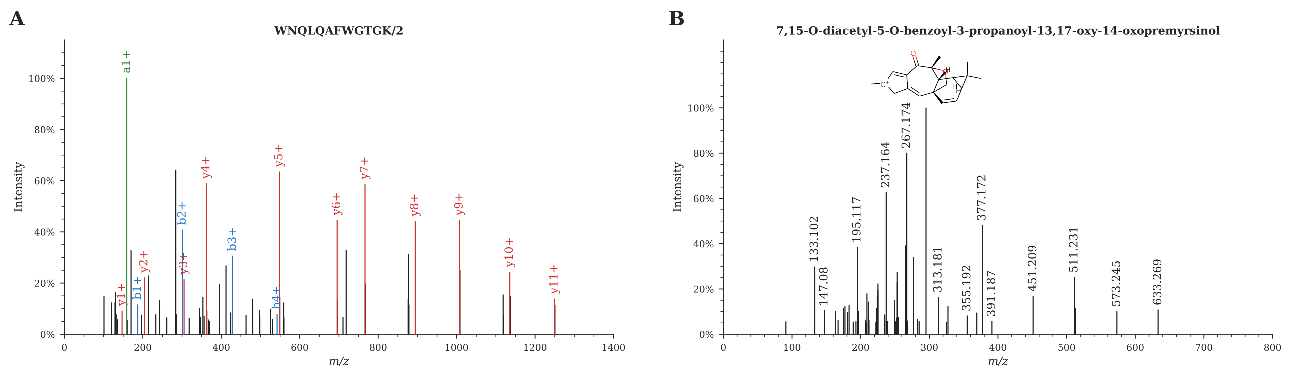

Figure 1: Visualization of (A) a proteomics MS/MS spectrum (mzspec:PXD004732:01650b_BC2-TUM_first_pool_53_01_01-3xHCD-1h-R2:scan:41840) [26] and (B) a metabolomics MS/MS spectrum (mzspec:GNPSLIBRARY:CCMSLIB00000840351) [16]. Fragment peaks can be easily annotated based on a peptide sequence, a SMILES string corresponding to a (sub)structure, or using custom annotations.

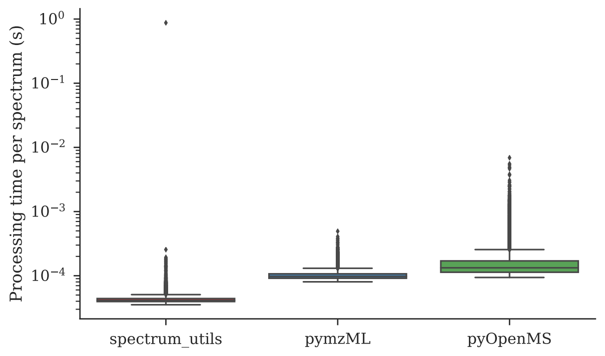

Figure 2: Spectrum processing runtime comparison. The runtime for processing the spectra generated for the iPRG2012 challenge [4] is reported for similar processing steps for `spectrum_utils` (version 0.3.0), pymzML (version 2.4.4) [1, 10], and pyOpenMS (version 2.4.0) [20]. The corresponding Python script is available in the supplementary information and at the online `spectrum_utils` documentation. Spectrum processing consisted of fixing the m/z range, removing precursor ion peaks 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 [11] of the first method call, allowing subsequent calls to be made very efficiently.

tionally, Numba [11], 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 techniques for optimized numerical computation, `spectrum_utils` is able to very efficiently process large amounts of MS data (figure 2).

MS/MS spectra can be easily visualized using `spectrum_utils`'s plotting functionality to help in producing publication-quality figures (figure 1). Additionally, a mirror plot can be used to clearly show the similarity between two different spectra, for example, to visualize identification results produced by spectral library searching. The default plotting functionality uses matplotlib [9] to generate high-quality spectrum figures. Alternatively, interactive plotting functionality is available using Altair [24], which is based on the Vega and Vega-Lite grammar of interactive graphics [21]. Interactive plotting is a drop-in replacement for the standard plotting functionality, allowing the user to trivially produce interactive spectrum plots for data exploration or visualization on web pages.

2.1 Use cases

We will briefly highlight two use cases to demonstrate the functionality of `spectrum_utils`. First, `spectrum_utils` is used by the ANN-SoLo open modification spectral library search engine [2, 3] to preprocess MS/MS spectra prior to spectral library searching. `spectrum_utils` provides the full functionality needed to preprocess the spectral data, including rounding the peak m/z values to a common mass resolution, removing the precursor peak and low-intensity noise peaks, and scaling the peak intensities. ANN-SoLo was developed to efficiently perform open modification searches, which typically suffer from a very large search space. In part due to the efficient spectrum processing functionality provided by `spectrum_utils`, ANN-SoLo is able to identify hundreds to thousands of spectra per minute, enabling researchers to perform untargeted PTM profiling via open searches at an unprecedented scale.

Second, the Global Natural Products Social Molecular Networking (GNPS) public data repository and analysis platform uses `spectrum_utils` to produce high-quality spectrum vector graphics. All MS/MS spectra processed by GNPS can be visualized using their universal spectrum identifier [5], as well as the individual spectra in reference spectral libraries compiled using the MSMS-Chooser workflow [25]. Additionally, spectral library identification results obtained using GNPS are visualized using mirror plots.

2.2 Code availability

spectrum_utils is available for Python 3.6+ and can be easily installed via conda using the Bioconda channel [8]. spectrum_utils depends on Numpy [23] and Numba [11] for efficient numerical computation, Pyteomics [7, 12] for peptide fragment ion mass calculations, RDKit [18] for SMILES string handling, matplotlib [9] for static plotting, and Altair [24] and Pandas [15] 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 perform this task. Instead, spectrum_utils takes the MS data provided by such tools as input for subsequent processing and visualization. spectrum_utils has a well-defined, Pythonic application programming interface, allowing developers to easily harness its powerful functionality in a small number of lines of code. Additionally, it is trivial to switch between the static plotting functionality to produce high-quality spectrum figures 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.

Besides the two use cases highlighted above, spectrum_utils has already been used in several other computational MS projects, showcasing its versatile functionality. Among others, it has been used to perform custom analyses of MS data corresponding to an unknown protein sample [17], to prepare MS/MS spectra for processing using deep neural networks [14], and to visualize spectra for scientific publications and presentations. This illustrates how spectrum_utils facilitates several MS data processing and visualization tasks and allows developers to quickly prototype ideas and implement diverse computational MS projects.

Funding

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

References

[1] Bald, T., Barth, J., Niehues, A., Specht, M., 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.

[2] 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: *Journal of Proteome Research* 18.10 (Aug. 26, 2019), pp. 3792–3799. doi: 10.1021/acs.jproteome.9b00291.

[3] Bittremieux, W., Meysman, P., Noble, W. S., and Laukens, K. "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.

[4] Chalkley, R. J., Bandeira, N., Chambers, M. C., Clauser, K. R., 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.

[5] Deutsch, E. W., Orchard, S., Binz, P.-A., Bittremieux, W., et al. "Proteomics Standards Initiative: Fifteen Years of Progress and Future Work." In: *Journal of Proteome Research* 16.12 (Dec. 1, 2017), pp. 4288–4298. doi: 10.1021/acs.jproteome.7b00370.

[6] 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.

[7] Goloborodko, A. A., Levitsky, L. I., Ivanov, M. V., and Gorshkov, M. V. "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.

[8] Grüning, B., Dale, R., Sjödin, A., Chapman, B. A., et al. "Bioconda: Sustainable and Comprehensive Software Distribution for the Life Sciences." In: *Nature Methods* 15.7 (July 2, 2018), pp. 475–476. doi: 10.1038/s41592-018-0046-7.

[9] 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.

[10] Kösters, M., Leufken, J., Schulze, S., Sugimoto, K., 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.

[11] 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.

[12] Levitsky, L. I., Klein, J. A., Ivanov, M. V., and Gorshkov, M. “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.

[13] Martens, L., Chambers, M., Sturm, M., Kessner, D., 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.

[14] May, D. H., Bilmes, J., and Noble, W. S. “A Learned Embedding for Efficient Joint Analysis of Millions of Mass Spectra.” In: *bioRxiv* (Nov. 29, 2018). DOI: 10.1101/483263.

[15] 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.

[16] Nothias-Esposito, M., Nothias, L. F., Da Silva, R. R., Retailleau, P., et al. “Investigation of Premyrsinane and Myrsinane Esters in *Euphorbia Cupanii* and *Euphobia Pithyusa* with MS2LDA and Combinatorial Molecular Network Annotation Propagation.” In: *Journal of Natural Products* 82.6 (June 28, 2019), pp. 1459–1470. DOI: 10.1021/acs.jnatprod.8b00916.

[17] Pino, L., Lin, A., and Bittremieux, W. “2018 YPIC Challenge: A Case Study in Characterizing an Unknown Protein Sample.” In: *Journal of Proteome Research* (in press Sept. 26, 2019). DOI: 10.1021/acs.jproteome.9b00384.

[18] *RDKit: Open-source cheminformatics*. https://www.rdkit.org/. (accessed: 2019-08-16).

[19] Röst, H. L., Sachsenberg, T., Aiche, S., Bielow, C., 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.

[20] Röst, H. L., Schmitt, U., Aebersold, R., and Malmström, L. “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.

[21] Satyanarayan, A., Moritz, D., Wongsuphasawat, K., and Heer, J. “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.

[22] Sturm, M., Bertsch, A., Gröpl, C., Hildebrandt, A., et al. “OpenMS – An Open-Source Software Framework for Mass Spectrometry.” In: *BMC Bioinformatics* 9.1 (Mar. 26, 2008), p. 163. DOI: 10.1186/1471-2105-9-163.

[23] 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.

[24] VanderPlas, J., Granger, B., Heer, J., Moritz, D., et al. “Altair: Interactive Statistical Visualizations for Python.” In: *Journal of Open Source Software* 3.32 (Dec. 10, 2018), p. 1057. DOI: 10.21105/joss.01057.

[25] Vargas, F., Weldon, K. C., Sikora, N., Wang, M., et al. “Protocol for Community-Created Public MS/MS Reference Library within the GNPS Infrastructure.” In: *bioRxiv* (Oct. 15, 2019). DOI: 10.1101/804401.

[26] Zolg, D. P., Wilhelm, M., Schnatbaum, K., Zerweck, J., et al. “Building ProteomeTools Based on a Complete Synthetic Human Proteome.” In: *Nature Methods* (Jan. 30, 2017). DOI: 10.1038/nmeth.4153.