PythoMS: A Python framework to simplify and assist in the processing and interpretation of mass spectrometric data

Lars P.E. Yunker, Darien Yeung, J. Scott McIndoe*.*

*Department of Chemistry, University of Victoria, P.O. Box 3065, Victoria, BC, V8W 3V6, Canada.† Email: larsy@uvic.ca, mcindoe@uvic.ca

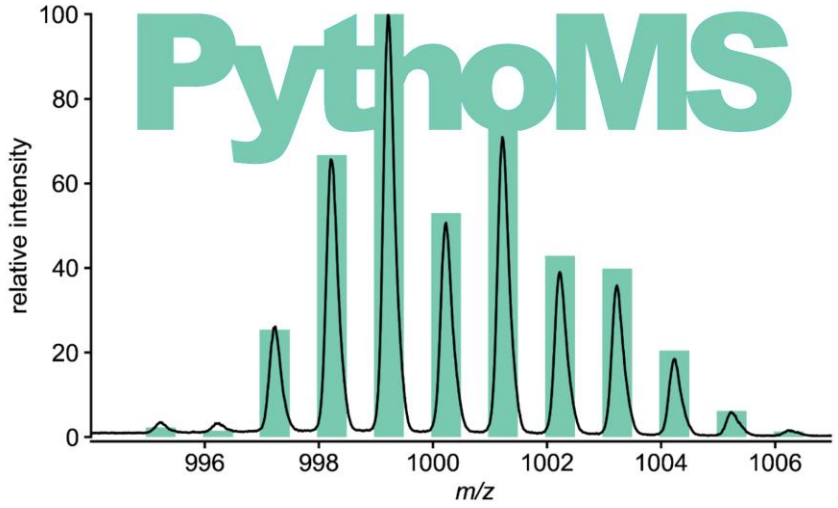
KEYWORDS. Mass Spectrometry, Python, Data Analysis, Video Visualization.

ABSTRACT

Mass spectrometric data are copious and generate a processing burden that is best dealt with programmatically. PythoMS is a collection of tools based on the Python programming language that assist researchers in creating figures and video output that is informative, clear and visually compelling. The PythoMS framework introduces a library of classes and a variety of scripts that quickly perform time-consuming tasks: making proprietary output readable; binning intensity vs time data to simulate longer scan times (and hence reduce noise); calculate theoretical isotope patterns and overlay them in histogram form on experimental data (an approach that works even for overlapping signals); render videos that enable zooming into the baseline of intensity vs. time

plots (useful to make sense of data collected over a large dynamic range) or that depict the evolution of different species in a time-lapse format; calculate aggregates; and provide a quick first-pass at identifying fragments in MS/MS spectra. PythoMS is a living project that will continue to evolve as additional scripts are developed and deployed.

GRAPHICAL ABSTRACT



INTRODUCTION

Mass spectrometers generate a substantial amount of complex data with every scan, which quickly amounts to an immense amount of data to process and interpret for any appreciable experiment length. Processing and interpreting these data is a time consuming and repetitive activity, particularly when trying to extract data series which were unforeseen by the programmers of the mass spectrometer's software package. As an example, the Waters QToF *micro* used in our laboratory stores many instrument variables (including the time, collision energy, total ion current, etc.) in addition to a paired list of *m/z* and intensity values across the entire scan range for every scan acquired in an experimental run. Frequently, we extract reconstructed single ion monitoring (RSIM) data from full scans; the *m/z* and intensity values are accessible to the user using the Waters MassLynx software, but extracting the data into spreadsheet format is time consuming (requiring the user to integrate each peak, switch windows, copy the data, switch to the spreadsheet program, paste the time and integration, then repeat for every other peak of interest). It is also difficult for the user to access the instrument variables through the provided software. The PythoMS framework was created using the Python programming language to address these drawbacks and limitations of the provided instrumental software. It is open source, registered on the Python Package Index (PyPI), and available on GitHub under the MIT license (https://github.com/larsyunker/PythoMS).

While there are more computationally efficient programming languages, the scripting language Python is attractive to researchers due to its ease of use and abundance of supporting libraries (allowing researchers to focus on the scripts themselves rather than building libraries). The great strength of Python is its ability to handle lists in an efficient manner, and since scientific research involves lists and arrays of data, it is well suited for scientific applications. The simple

set of objects and methods built into Python, combined with the ease of generating other objects and methods tailored to the user's needs, makes the language a powerful tool for extraction, manipulation, and storage of data. Additionally, the user-specified methods allow for complete control over the manipulation of those data, leading to reliable and reproducible batch application of the same method to multiple data sets.

The PythoMS framework has two main segments: a library of classes and a collection of scripts. The primary functionality of the framework is built into Python classes, providing fundamental tools required for the processing and interpretation of mass spectrometric data, and may be instantiated and implemented in user-created scripts as needed. Additionally, scripts have been written and generalized to enable straightforward execution of common mass spectrometric tasks like integration, spectrum summing, and isotope pattern overlays. There are existing examples of Python-based packages for both general chemistry and the analysis of mass spectrometric data (these are primarily focused on proteomics analysis), they did not meet our specific needs..[1-4] We sought to create a generalized framework which might be applied to any mass spectrometric application with ease. There are also web services available for mass spectrometrists: one such is ChemCalc,[5] which is a free platform allowing chemists to calculate isotope patterns, monoisotopic masses, find molecular formulae and more. This suite of programs is already heavily used in our laboratory, and PythoMS is a complement to these.

Framework

Mass Spectrometric Data Access Classes

In order to manipulate data in Python, one must first extract the data from the source data file, which can be non-trivial due to each manufacturer having their own proprietary data format (the encoding of which is not supported natively by Python). In an effort to address this, the mzML file format was defined by the Human Proteome Organization Proteomics Standards Initiative (HUPO-PSI) working group to be an open-source file format for mass spectrometry data, so that the files generated by any instrument could be read in a standardized fashion.[6-8] The mzML files themselves are based on the Extensible Markup Language (XML) file format, where data are grouped together with tags so that a particular piece of data may be found easily by that tag. Additionally, ProteoWizard provides an accessible tool to convert the mass spectrometric data formats of several manufacturers into mzML, allowing full access to the data generated by a wide variety of spectrometers.[9-10]

The mzML class serves as an abstraction layer between user and an mzML file, providing access to all functions, spectra, and attributes defined in the file. After instantiation (loading of the mzML file into the instance), the user may retrieve a single scan or chromatogram from the file or apply a method to every scan or chromatogram in the file (such as extract the time points or integrate a region of the spectrum). This functionality is enabled by a module created to access HUPO-PSI controlled variable (CV) definitions. Methods have been written to extract CV parameters or attributes from an XML branch, which gives the user simplified access to all variable names, values, and definitions associated with that branch. In the case of ProteoWizard, only accession IDs and CV names are included in the file, making it challenging for the user to inspect

or interpret the key. PythoMS addresses this by automatically retrieving the defined CV attributes from the HUPO-PSI file itself whenever an attribute is undefined in the mzML. Direct access to the CV parameter names or IDs of an mzML branch enables the user to create methods which will specifically extract or manipulate exactly which variables they choose, or to perform comparisons and conditionally evaluate spectra or chromatograms. Convenience wrappers have also been written which apply user-defined methods to spectra or chromatograms; combined with the flexibility and specificity of the CV accession keys, the mzML class provides a solid foundation for nearly any mass spectrometric data processing task. While there are other Python frameworks available for extracting data from mzML files (Pyteomics, pymzML, pyOpenMS, mMass),[1-3, 11-12] we have created an ecosystem which can be readily utilized by users for targeted data processing applications or developers in creating their own expanded framework. We foresee the built-in decorator functions of the mzML class being most useful to the end user, as they apply a user-defined method to each spectrum or chromatogram in an mzML file. For instance, any aspect of the scan/chromatogram can be retrieved and interpreted, which allows for Pythonic conditional statements or item modification.

Mass Spectrometric Data Classes

A substantial issue with values stored by mass spectrometers is that they will track *m/z* values to precisions well beyond the spectrometer itself. For instance, the Waters QToF *micro* used in our lab records *m/z* values to the 7th decimal place, while it is only routinely accurate to the 1st. From the perspective of a script attempting to sum mass spectra together, it becomes challenging to address effectively identical but formally unequal *m/z* values in different scans. As an example,

one scan may have intensity at *m/z* 123.4567890 and another intensity at *m/z* 123.4567891; these are formally unequal, but in actuality indistinguishable by the spectrometer. This provides a challenge when one wants to combine multiple scans from a mass spectrometer, where *m/z* values differ minutely from one scan to the next. To address these discrepancies, the Spectrum class was created to efficiently consolidate intensities of the same effective mass.

At its core, the Spectrum class is a container for two paired lists (one for *m/z*, one for intensity, with the first value of the *m/z* list corresponding to the first value of the intensity list, and so on), but has built-in methods for efficiently searching the *m/z* value list and adding new values while maintaining sorting. On instantiation, the user will specify a given number of decimal places that the instance is to track, and when new values are added, the *m/z* value rounded to the specified decimal place is compared to the existing *m/z* list, ignoring extraneous precision. This allows users to combine spectra that have slight mismatches for the *m/z* values, or to combine spectra of different numbers of (intensity, *m/z*) pairs (frequently, mass spectrum data files omit zero-intensity *m/z* values for data storage efficiency). Once complete, a single function of the class can be called to return a summed spectrum in the form of a pair of *m/z* and intensity lists. While this class was created with mass spectra in mind, it could be generally applied to the combination of any paired lists with similar but unequal *x* values or spectra of unequal length. The behaviour of the Spectrum class may be tweaked to the user's needs to optimize run-time for different spectral types. The authors have found that the most significant performance-affecting attribute is whether the spectrum is "filled" (whether on instantiation a *m/z* value is generated for every value in the specified range incremented at the specified decimal place). If the spectrum is likely to have an intensity value for the majority of *m/z* possibilities, it was found to be more efficient to pre-

populate, but when values are likely to be clustered it was more efficient to generate *m/z* values as necessary.

The creation of this class allows PythoMS to have a spectrum binning script, which can very efficiently sum all spectra in a mzML file. It also allowed for the creation of the Molecule class, as the algorithms used to calculate an isotope pattern require rapid generation and manipulation of list (a specific method was built into the Spectrum class to add an element's isotope distribution efficiently to an existing Spectrum object).

Isotope patterns are a fundamental component of interpreting mass spectra, and we desired an efficient method of accurately calculating isotope patterns, with the resulting pattern being a Python paired list. Despite the ubiquitous necessity of accurate isotope patterns, few publications discuss algorithms or methods of constructing those patterns. While the patterns of small molecules are not particularly arduous for modern microprocessors, patterns for large molecules or those containing more polyisotopic elements quickly can become very time consuming to generate. We created the IPMolecule class to efficiently generate an isotope pattern which is as accurate as possible (in that the monoisotopic mass and complete pattern should match experimental spectra as closely as possible). This class relies heavily on the efficiencies of the Spectrum class and has several generation algorithms available.

The IPMolecule class is based on a generic Molecule class which interprets a molecular formula and calculates basic molecular properties such as molecular weight. There are a wide variety of molecular weight calculators available, but the nature of IPMolecule required an object-oriented structure to build upon, so we created a class to do this. For the convenience of the user, the Molecule class allows for specification of defined formula abbreviations (e.g. "L" for "PPh_3"),

nested brackets, molecular charge, and single isotopes in the provided molecular formula. The IPMolecule generates three isotope patterns for each molecule, a "raw" pattern which preserves all mass defects, a "bar" isotope pattern which consolidates defects of similar mass into a combined intensity, and a "Gaussian" isotope pattern which simulates the observed pattern on a mass spectrometer of a given resolution. These patterns may be individually retrieved from the IPMolecule object (e.g. for isotope pattern overlay figures) or may be plotted directly with built-in methods of the object itself. There are two isotopic exact mass dictionaries available which may be used for calculating isotope patterns: one with values from the CRC Handbook of Chemistry and Physics,[13] and the other with values from National Institute of Standards and Technology (NIST) website and formatted by Pyteomics.[1, 14]

Scripts

The classes detailed above provide an agile framework to develop code for mass spectrometric applications, and we have utilized that framework in the development of several scripts which may be found in the PythoMS repository. These scripts are intended to be the point of interaction between the framework and the user, and we have structured them to be as user-friendly and intuitive as possible even for novice python users. The framework was constructed in such a fashion that new scripts can be written quickly to accomplish a new task; as an example, a script was generated in less than 5 minutes which calculated the masses of a series of products expected from a Suzuki polycondensation reaction using the IPMolecule class (this script was later generalized to be an aggregate calculator script). With some knowledge of Python and the help of

the documentation of this framework, any MS user can construct scripts to ease otherwise time consuming or repetitive tasks.

PyRSIR

Single ion monitoring is the process of tracking the abundance of a single ion over a time course, and is the data in which our lab is interested in for the online study of reactions.[15-18] To do this, we obtain full-spectrum scans on our instrument, then reconstruct a single ion monitoring trace by integrating the intensity of a particular ion over the entire time course of the experiment. The Python Reconstructed Single Ion Recording (PyRSIR) script automates this data interpretation and was the intended application of the PythoMS framework. The goal of this script is to automate how we process mass spectrometric data, tracking the integration of all the intensity associated with a given ion across all time points in a mass spectrometric experiment. This required not only interfacing with the scans, but also having access to low-level scan attributes to extract only the intended data, which is achieved using the mzML class.

The script requires that the user provide a dictionary of names, start and end *m/z* for each peak they wish to integrate. This dictionary may either be defined as such, or generated in an Excel workbook which they may then provide to the script (the integration parameters are automatically extracted from the workbook in this case). The user may instead specify a molecular formula for an ion rather than bounds, and the script will automatically determine the bounds to integrate by simulating the isotope pattern with IPMolecule and generating a confidence interval for the bounds based on the pattern and an automatically calculated resolution of the instrument. This of course requires that the instrument in use is accurately calibrated, with no substantial calibration drift. Additionally, they must also provide the name of a mass spectrometric data file (either in mzML

or Waters' .RAW format; support can be added as needed for any format interpretable by ProteoWizard). The script takes the supplied parameters and uses the mzML class to integrate between the provided bounds for every scan in the MS file. The returned data is then normalized normalized to the total ion current (a commonly applied transformation to MS data), and both the raw and normalized data is output to the Excel workbook.

The intensity of MS data can varies from scan to scan, which makes the data noisy, visually unappealing and difficult to interpret in cases where there are multiple traces present. To address this, PyRSIR has a built-in binning algorithm, where the user can specify the number of adjacent scans to bin together. For example, if the user specifies 3 scans to bin, the intensity of every three scans is combined for each tracked ion and is stored as a single data point. This has the effect of visually reducing the noise of the spectrum, without eliminating any data in a smoothing process (Figure 1), essentially simulating a longer scan time in the spectrometer. These binned data are also output to the Excel workbook, and the user may select which of the raw, normalized, or binned data they wish.

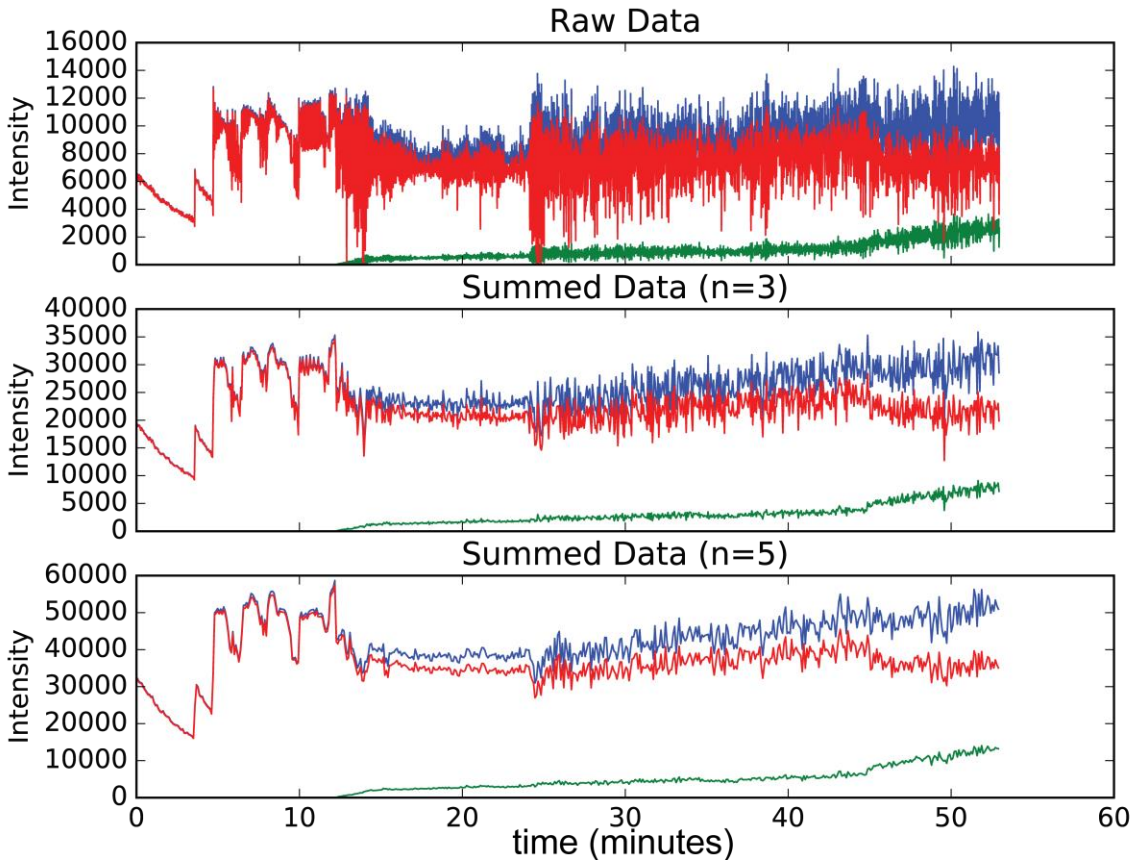


Figure 1. The PyRSIR output illustrating the effect that several levels of binning has on scan-to-scan noise. These data were chosen as an example of particularly severe scan-to-scan noise.

The script additionally sums spectra of all scans, as well as the isotope patterns contained within the supplied bounds of each ion, and outputs these to the Excel file. The former is retained for the user's convenience, as we frequently show a region of the summed spectra for illustrative purposes (the summed spectra functionality also has a minimal impact on performance). The latter allows the user to verify visually that the bounds they specified are appropriate (this is not always apparent when selecting the bounds).

This script automates the time consuming and repetitive actions involved in processing mass spectrometric data to extract reconstructed single ion monitoring traces, and it has

substantially improved the efficiency of our data processing. Historically, a researcher in our group might acquire reaction data for a day, then spend the next one or two days processing that data. This script fully automates the data extraction and processing, completing the same tasks in minutes. The substantial efficiency increase allows the student to spend more time considering the implications of the data, provides immediate feedback on the quality of their bound selection (the extracted isotope patterns provide excellent visual queues for whether there are multiple patterns overlapping in a bounded region), and enables rapid re-processing (e.g. if those bounds were incorrectly selected).

Isotope pattern overlay

When assigning a molecular formula to an observed ion in a mass spectrum, there are typically two matches made: the *m/z* value compared to the exact mass, and the comparison of observed and predicted isotope patterns. If there is a good visual match between predicted and observed patterns, the assignment as that formula is supported. These comparisons are generally qualitative in nature, but this is usually sufficient to decide whether a match is good (particularly for polyisotopic species). The isotope pattern overlay script was written to generate figures which allow the user to perform this comparison between experimental and predicted isotope patterns.

The script takes one or more molecular formulae as input and predicts the isotope patterns using the IPMolecule class. It then loads a provided experimental spectrum and overlays the predicted patterns over top of an experimental spectrum (usually in an Excel workbook), then saves the figure to file (Figure 2). There are a wide variety of parameters which may be tweaked by the user to control the behaviour of the script, all of which are detailed in the script's

documentation, and a set of commonly used defaults have been specified by the author to suit most applications (publication, inset, detailed analysis, etc.). A "detailed" preset for the script performs comparison calculations between the experimental exact mass and the predicted exact mass, including that information on the output figure (Figure 3). The script automatically scales the height of the predicted isotope pattern to match the maximum intensity of the spectrum within the bounds of that isotope pattern. Multiple isotope patterns can be predicted within a given figure in this way, allowing for analysis of adjacent or even overlapping isotope patterns (Figure 4). As well, the script automatically determines an appropriate *m/z* window to render for the output figure, which rarely requires end-user adjustment (typically the most time-consuming aspect of rendering an isotope pattern overlay image as it is an iterative process).

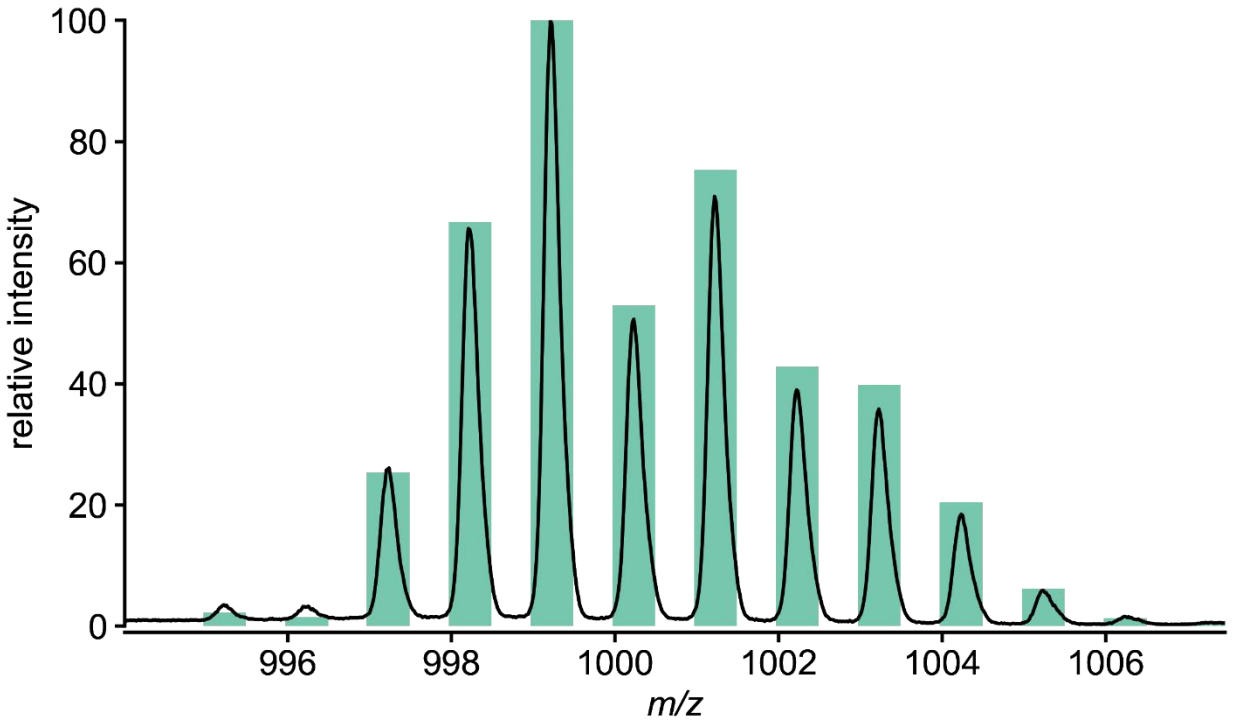


Figure 2. An example output of the isotope pattern overlay script showing the experimental (line) and predicted (bars) isotope pattern of $C_{61}H_{52}OP_3Pd$. This figure was generated using the publication preset of the isotope pattern overlay script.

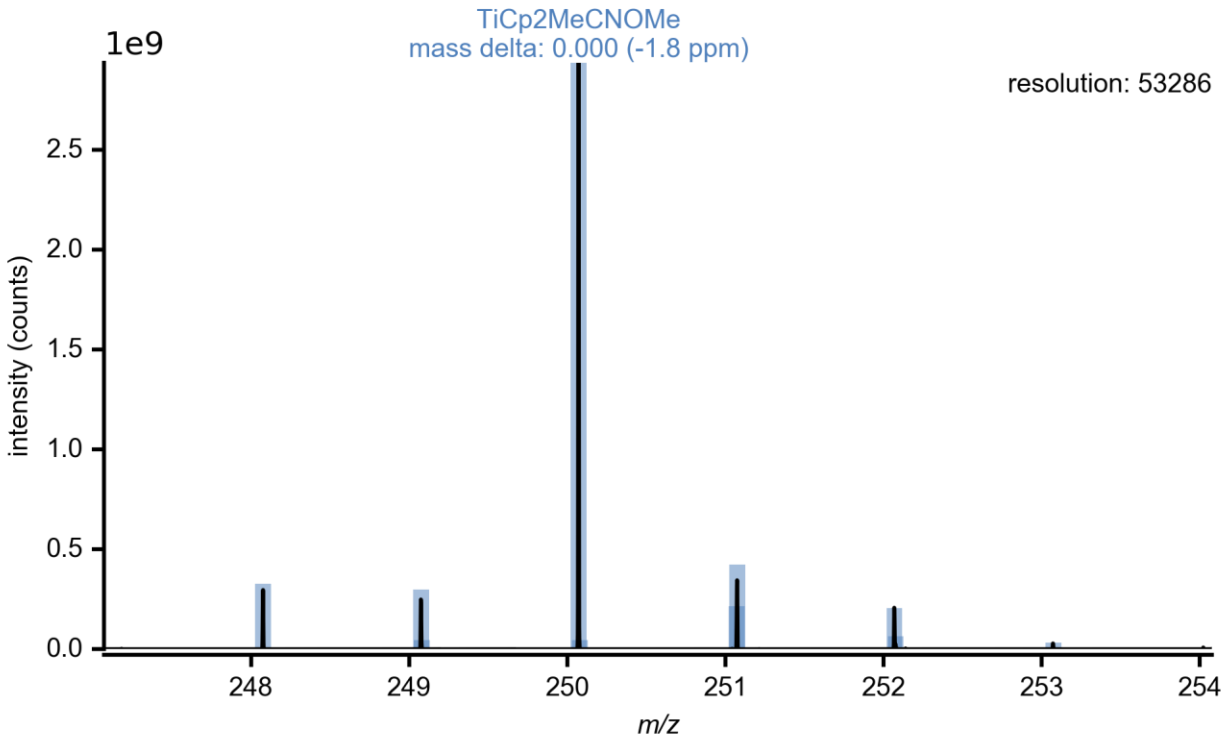


Figure 3. An example output of the "detailed" preset of the isotope pattern overlay script showing the experimental orbitrap spectrum (line) and predicted (bars) isotope pattern of a titanium species. The preset includes an automatic printout of the mass delta (in both *m/z* and ppm) between the experimental and predicted exact mass.

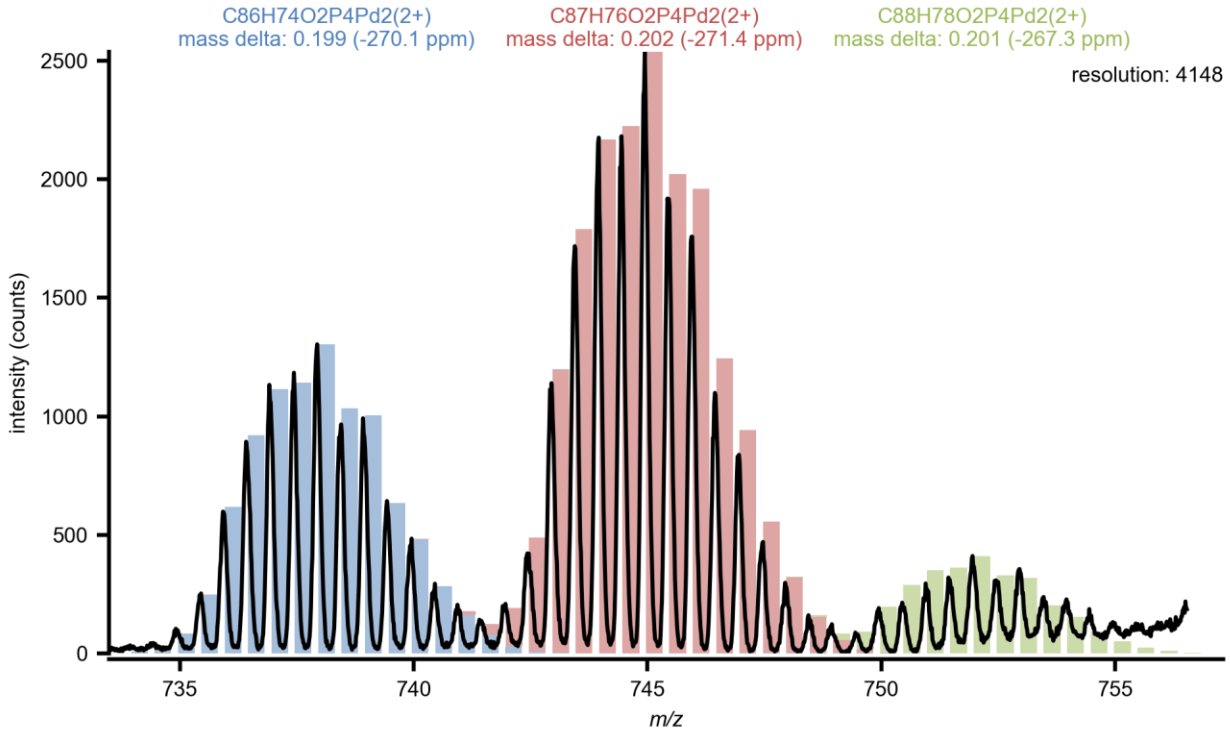


Figure 4. An example output of the isotope pattern overlay script predicting and overlaying three isotope patterns on a complicated palladium dimer series. The detailed preset of the script was used here, which includes a label, mass delta between actual and predicted exact mass, as well as the resolution of the spectrum (the font size was intentionally reduced to avoid label overlap).

Video frame renderers

Figures generated for publication are by necessity designed in such a manner to be easily readable as a static image. However, when discussing data in an oral presentation, animation of data can be far more engaging for the audience. Two animation scripts were written to render frames for use in video format. The first of these, video frame renderer, takes a similar input to PyRSIR and extracts data from a specified mzML file. The script then plots a series of images showing the mass spectrum at a given time point on one half of the image, and the normalized abundance traces on the other half.[19-20] When the entire series is rendered and encoded in video

format, it very effectively illustrates how we can observe the consumption and production of different ions in real time; it is essentially a time-lapse rendition of the evolving reaction.

Another frame rendering script (*y*-axis zoom figure), was created to illustrate the massive observable dynamic range of a mass spectrometer. Using a similar user input to that of the video frame renderer, it plots an abundance trace over time figure (like those seen in Figure 6). It then "zooms" into the baseline, expanding the *y* axis so that traces that were previously in the baseline can be viewed. It does this vertical expansion as many times as the user specifies, and a rendered video was included in the supporting information of a recent publication from our group.[15]

Spectrum binner

Combined spectra are frequently used when examining isotope patterns or generating figures. While the Waters MassLynx software has a tool which allows this, it can only track *m/z* values to 4 decimal places and can take some time to complete a combination of a long or complex acquisition (we also frequently encounter program crashes when asking it to do so). The spectrum binner script was created to address this and utilizes the Spectrum and mzML classes to combine all spectra in a provided MS file, tracking the user-specified number of decimal places. In comparison to the algorithm implemented by MassLynx™, the Spectrum outperforms the manufacturer with respect to computational resources and time (as an example, binning of a 3156-scan experiment took MassLynx 4 minutes and 4 seconds and PythoMS only 1 minute and 32 seconds on the same computer). The combined spectrum is saved to a Microsoft Excel workbook in both counts and normalized intensity for the user's convenience (the data are also returned if the function is called within another Python script).

Aggregate calculator

The calibration of an Electrospray Ionization Mass Spectrometer (ESI-MS) involves acquiring the spectrum of a solution known to aggregate (commonly NaI in MeOH, generating ions of the form $[Na_{x+1}I_x]^+$ in positive ion mode and $[Na_xI_{x+1}]^-$ in negative ion mode). The resulting aggregate series is compared to the exact masses, and a calibration polynomial is calculated by the software to move the observed masses to match the exact masses. The aggregate calculator script calculates the exact mass of a series of aggregates using the IPMolecule class, printing the exact masses of those aggregates in the console. The aggregate calculator may also be used to generate an accurate calibration file for use in calibrating mass spectrometers (this requires the user to predefine the syntax for the calibration files). We have also found this script useful in interpreting plots, as it can quickly calculate an aggregate series which the user can then compare to their spectrum to see if an aggregate assignment is reasonable.

MSMS interpreter assistant

When interpreting a tandem mass spectrometry experiment, typically the first action performed is to find the difference between the observed peaks and each other. This is yet another tedious and repetitive task, and the MSMS interpreter assistant was written to address this problem. The script sums an mzML file into a single spectrum, detects peaks in that spectrum, and calculates all the mass differences between those peaks. The resulting spectrum and output table are both printed to console and written to an Excel workbook. The user can then look at an appropriately formatted table of differences, making it substantially more straightforward to identify significant differences (e.g. a loss of 127 is likely I). The script also reads from a dictionary of common losses

predefined by the user (e.g. 77 is phenyl, 262 is triphenylphosphine, 79 is bromine, etc.) and guesses at what the observed difference might represent. While the guesses are by no means certain assignments, it is the experience of the author that the script has correctly assigned many differences, allowing the user to focus on assigning the more challenging differences.

CONCLUSIONS

Making sense of mass spectrometric data is assisted greatly by the development of programming tools, and PythoMS provides chemists with a starting collection of implements that we hope will not only prove useful but will inspire others to write and share their own. Learning to write Python code has become an essential part of training in our laboratory, and we anticipate continuing to build on the PythoMS tools in the near future.

AUTHOR INFORMATION

Present Addresses

† Lars P.E. Yunker. Department of Chemistry, University of British Columbia, 2036 Main Mall, Vancouver, BC V6T 1Z1, Canada. larsy@chem.ubc.ca

Author Contributions

The manuscript was written through contributions of all authors. All authors have given approval to the final version of the manuscript.

Funding Sources

JSM thanks NSERC (Discovery and Discovery Accelerator Supplement) for operational funding and CFI, BCKDF and the University of Victoria for infrastructural support.

Abbreviations

RSIM reconstructed single ion monitoring; HUPO PSI Human Proteome Organization Proteomics Standards Initiative; XML Extensible Markup Language; CV controlled vocabulary; CSV comma separated values; NIST National Institute of Standards and Technology.

REFERENCES

1. Goloborodko, A. A.; Levitsky, L. I.; Ivanov, M. V.; Gorshkov, M. V., Pyteomics—a Python Framework for Exploratory Data Analysis and Rapid Software Prototyping in Proteomics. *Journal of The American Society for Mass Spectrometry* **2013,** *24* (2), 301-304.
2. Kremer, L. P. M.; Leufken, J.; Oyunchimeg, P.; Schulze, S.; Fufezan, C., Ursgal, Universal Python Module Combining Common Bottom-Up Proteomics Tools for Large-Scale Analysis. *Journal of Proteome Research* **2016,** *15* (3), 788-794.
3. Strohalm, M.; Kavan, D.; Novák, P.; Volný, M.; Havlíček, V., mMass 3: A Cross-Platform Software Environment for Precise Analysis of Mass Spectrometric Data. *Analytical Chemistry* **2010,** *82* (11), 4648-4651.
4. Mentel, L. M. mendeleev - A Python resource for properties of chemical elements, ions and isotopes. https://bitbucket.org/lukaszmentel/mendeleev.
5. Patiny, L.; Borel, A., ChemCalc: A Building Block for Tomorrow's Chemical Infrastructure. *Journal of Chemical Information and Modeling* **2013,** *53* (5), 1223-1228.
6. Deutsch, E., mzML: A single, unifying data format for mass spectrometer output. *PROTEOMICS* **2008,** *8* (14), 2776-2777.
7. Deutsch, E. W., Mass Spectrometer Output File Format mzML. In *Proteome Bioinformatics*, Hubbard, S. J.; Jones, A. R., Eds. Humana Press: Totowa, NJ, 2010; pp 319-331.
8. Martens, L.; Chambers, M.; Sturm, M.; Kessner, D.; Levander, F.; Shofstahl, J.; Tang, W. H.; Römpp, A.; Neumann, S.; Pizarro, A. D., mzML—a community standard for mass spectrometry data. *Molecular & Cellular Proteomics* **2011,** *10* (1), R110. 000133.
9. Chambers, M. C.; Maclean, B.; Burke, R.; Amodei, D.; Ruderman, D. L.; Neumann, S.; Gatto, L.; Fischer, B.; Pratt, B.; Egertson, J.; Hoff, K.; Kessner, D.; Tasman, N.; Shulman, N.; Frewen, B.; Baker, T. A.; Brusniak, M.-Y.; Paulse, C.; Creasy, D.; Flashner, L.; Kani, K.; Moulding, C.; Seymour, S. L.; Nuwaysir, L. M.; Lefebvre, B.; Kuhlmann, F.; Roark, J.; Rainer,

P.; Detlev, S.; Hemenway, T.; Huhmer, A.; Langridge, J.; Connolly, B.; Chadick, T.; Holly, K.; Eckels, J.; Deutsch, E. W.; Moritz, R. L.; Katz, J. E.; Agus, D. B.; MacCoss, M.; Tabb, D. L.; Mallick, P., A cross-platform toolkit for mass spectrometry and proteomics. *Nat Biotech* **2012,** *30* (10), 918-920.
10. Kessner, D.; Chambers, M.; Burke, R.; Agus, D.; Mallick, P., ProteoWizard: open source software for rapid proteomics tools development. *Bioinformatics* **2008,** *24* (21), 2534-2536.
11. Bald, T.; Barth, J.; Niehues, A.; Specht, M.; Hippler, M.; Fufezan, C., pymzML—Python module for high-throughput bioinformatics on mass spectrometry data. *Bioinformatics* **2012,** *28* (7), 1052-1053.
12. Röst, H. L.; Schmitt, U.; Aebersold, R.; Malmström, L., pyOpenMS: A Python-based interface to the OpenMS mass-spectrometry algorithm library. *PROTEOMICS* **2014,** *14* (1), 74-77.
13. Lide, D. R., *CRC Handbook of Chemistry and Physics*. 93 ed.; CRC press: 2012.
14. Commerce, U. S. D. o. National Institute of Standards and Technology. https://www.nist.gov/ (accessed Feb 19).
15. Theron, R.; Wu, Y.; Yunker, L. P. E.; Hesketh, A. V.; Pernik, I.; Weller, A. S.; McIndoe, J. S., Simultaneous Orthogonal Methods for the Real-Time Analysis of Catalytic Reactions. *ACS Catalysis* **2016,** *6* (10), 6911-6917.
16. Vikse, K. L.; Henderson, M. A.; Oliver, A. G.; McIndoe, J. S., Direct observation of key intermediates by negative-ion electrospray ionisation mass spectrometry in palladium-catalysed cross-coupling. *Chemical Communications* **2010,** *46*, 7412-7414.
17. Vikse, K. L.; Woods, M. P.; McIndoe, J. S., Pressurized Sample Infusion for the Continuous Analysis of Air- And Moisture-Sensitive Reactions Using Electrospray Ionization Mass Spectrometry. *Organometallics* **2010,** *29* (23), 6615-6618.
18. Yunker, L. P. E.; Stoddard, R. L.; McIndoe, J. S., Practical approaches to the ESI-MS analysis of catalytic reactions. *Journal of Mass Spectrometry* **2014,** *49* (1), 1-8.
19. Yunker, L. P. E.; Ahmadi, Z.; Logan, J. R.; Wu, W.; Li, T.; Martindale, A.; Oliver, A. G.; McIndoe, J. S., Real-time mass spectrometric investigations into the mechanism of the Suzuki-Miyaura reaction. *Organometallics* **2018,** *In the press*.
20. Belli, R. G.; Wu, Y.; Ji, H.; Joshi, A.; Yunker, L. P. E.; Rosenberg, L.; McIndoe, J. S., Competitive Ligand Exchange and Dissociation in Ru Indenyl Complex-es. *Inorganic Chemistry* **2018,** (Submitted).