# DEIMoS GUI: An Open-Source User Interface for a High-Dimensional Mass Spectrometry Data Processing Tool

*Marjolein T. Oostrom[1], Sean M. Colby[2], and Thomas O. Metz[2,] **

[1]National Security Directorate, Pacific Northwest National Laboratory, Richland, Washington 99352, U.S.A.

[2]Earth and Biological Sciences Directorate, Pacific Northwest National Laboratory, Richland, Washington 99352, U.S.A.

KEYWORDS: Mass Spectrometry, DataShader, Holoviz, Param

ABSTRACT: We report here the creation of a graphical user interface (GUI) for the Data Extraction for Integrated Multidimensional Spectrometry (DEIMoS) tool. DEIMoS is a Python package to process data from high-dimensional mass spectrometry measurements. It is divided into several modules, each representing a data processing step, such as peak detection, alignment, and tandem mass spectra extraction and deconvolution. The inputs for and outputs from DEIMoS can include millions of N-dimensional data points, which can be challenging to visualize in a way that is interactive, informative, and responsive. Here, we used the HoloViz Python data stack, including DataShader and Param, to create an interactive visualization of mass spectrometry data. We believe the GUI will increase the accessibility of DEIMoS and the visualization methods could be useful for other open-source mass spectrometry tools.

## Introduction

Mass spectrometry (MS) data can be processed and visualized with both instrument vendor software and open-source software. Vendor software such as Agilent MassHunter (1), Bruker MetaboScape, and Waters Progenesis QI allow analysis and visualizations of the specific instrument data, but often cannot be generalized to data from other manufacturers. In addition, customizing vendor software to meet individual user needs can be challenging due to the unavailability of open-source code bases for the vendor software. In contrast, open-source programs allow additional customizations for MS data analysis and visualization.

There are other open-source software for mass spectrometry analysis and visualization such as MZMine (2), MS-DIAL (3), and OpenMS (4). All these software applications include analysis, graphical user interface (GUI), and visualization of mass spectrometry data. DEIMoS GUI is unique in that it was developed specially for the DEIMoS Python package. The DEIMoS, Data Extraction for Integrated Multidimensional Spectrometry, is a Python package created to process data from MS platforms in an instrument and dimension-agnostic manner previously described by Colby et al (5). DEIMoS functions allow any number of dimensions as input and makes minimal assumptions on the properties of these dimensions. As a result, DEIMoS can be used on data from any MS-based platform with any number or type of dimensions. DEIMoS is divided into several modules, each representing a data processing step. Currently, DEIMoS is run from Python scripts or the command line and contains functionality to generate visualizations with the Python package Matplotlib (6). The package thus includes functions to create built-in static plots, but currently lacks an interactive visualizer of the mass spectrometry data.

The GUI allows users to utilize the DEIMoS Python package without needing to run Python scripts. Instead, the GUI allows an intuitive use of DEIMoS with simple user-inputs, which increases the accessibility to the tool. In this version of the GUI, we implemented all the major functions within the package: 1) data smoothing, 2) peak detection, which finds the local maxima within the data through an efficient topological data analysis-based method 3) tandem mass spectra (MS/MS) extraction and deconvolution, which assigns fragments to respective MS parent ions 4) isotope detection, where the isotopes are matched based on m/z offset 5) calibration, where the calibration model is trained on a collision cross section calibration mix, and 6) data alignment, which aligns the dimensions from sample files with the dimensions of a reference file

using support vector regression to model the relationship between samples (5). All DEIMoS functions are run within the GUI backend.

Creating a useful, interactive visualization of high-dimensional MS data (e.g. from liquid chromatography-ion mobility spectrometry-tandem mass spectrometry (LC-IMS-MS/MS)) is challenging mainly due to the need to show an overview of the detailed contents of the large data files, where a single analysis file can be on the order of hundreds of megabytes to gigabytes in size. Despite the size of the data, our method for visualization is responsive to user inputs and allows for detailed examination and manipulation of the data. Interactive visualization is important for mass spectrometry analysis so that users can change function inputs based on the visualization of the inputs and outputs of the functions. Without the ability to view and zoom into the data, it would be more difficult for users to understand the effects of DEIMoS function input choices. When creating a GUI for the DEIMoS Python package, we enabled 1) informative aggregations 2) efficient interactivity, and 3) linked selections, where the selected region on one graph filters the underlying data and thereby effects the selected points in all graphs which are using the same underlying data. More generally, the GUI is a straight-forward method of utilizing the DEIMoS Python package for users who are unfamiliar with using Python packages directly.

First, the GUI visualizes dimensions of the first stage of mass spectrometry data (MS1), which can include millions of N-dimensional data points, with informative aggregation. One challenge with N-dimensional mass spectrometry data is there is no fixed dimension, so the aggregation method we utilized is able to handle the different ranges and spacings of different dimensions. It is also challenging to show the relationship between all the N number of dimensions when there are more than two dimensions, as 3D graphs are often difficult to visualize on a computer screen.

We dealt with the N-dimensional data by graphing the aggregates of each possible pair of the dimensions in our data (7) using the Python package DataShader. The package has been used before in publications and GUIs to showcase mass spectrometry data, including similar feature vs feature DataShader charts (8-10). In addition, Schessner et al conducted an overview of proteomic visualizations and mention DataShader as a useful tool to visualize and aggregate data (11). The application currently displays three dimensions, but modifying the code to use a different number of dimensions would be straight-forward.

Secondly, to increase the usability of the app, the GUI is interactive, with the user able to provide inputs, and efficient in processing input changes. There were many user inputs, including file path specification, manual input of the visualization bounds in each dimension, and DEIMoS function inputs.

Finally, the linked selection allows the users to interact with and view the 3D data by viewing different 2D projections of the same slice of the 3D data. One of the challenges of our GUI was to visualize a 3D dimensional space with multiple 2D graphs. The Holoviz "linked selection" function allows the plot data to be highlighted based on all dimensions in the underlying data , rather than only the two dimensions used in the x- and y-axes of the plot of the selected 3D data (12). For example, a box selecting a section of the *m/z* in one plot would filter the data on a linked plot with retention time and drift time, despite the linked plot not including a *m/z* dimension, as the underlying data is being filtered on *m/z* level.

We will show how DEIMoS was implemented within the GUI using the HoloViz stack, and how we handled the technical challenges of interactively visualizing mass spectrometry data.

## Implementation

**Table 1.** Packages used: Purpose of packages within the GUI

| Package | Purpose |
|---|---|
| DataShader (7) | Representing large datasets |
| Param (13) | Add user widgets |
| Panel (14) | Create dashboards |
| HoloViews (12) | Create interactive visualization |
| Dask (15) | Parallel computing |

In creating the GUI, we used the Holoviz stack (13), a set of compatible Python packages for interactive data visualizations. We utilized many of the useful functionalities from the libraries in the stack including data aggregation from DataShader (7), user widgets and Parameterized classes from Param (13), deployable dashboard creation from Panel (14), and interaction visualization from HoloViews (12) **(Table 1**). We also utilized Dask, a Python package for parallel computing to improve the speed of the app.

Visualizing the final and intermediate output of the DEIMoS Python package involved using DataShader. DataShader automatically addresses several problems inherent to plotting many points in a limited space (7). One important benefit of DataShader is its automatic re-aggregation (rasterization) when the x-axis or y-axis range changes (7). The user can zoom into new areas of the chart with the user-input widgets, and the aggregation and colormap level will automatically update, allowing the user to inspect the data on an overview level or a focused area. We disabled the re-aggregation when the user zoomed in using the toolbar to only trigger re-aggregation if the user chooses to refresh the plots with the user widgets in Figure 3.7. DataShader runs the

aggregation steps efficiently, which allows the user to display millions of datapoints even on a personal computer as shown in **Table 2.**

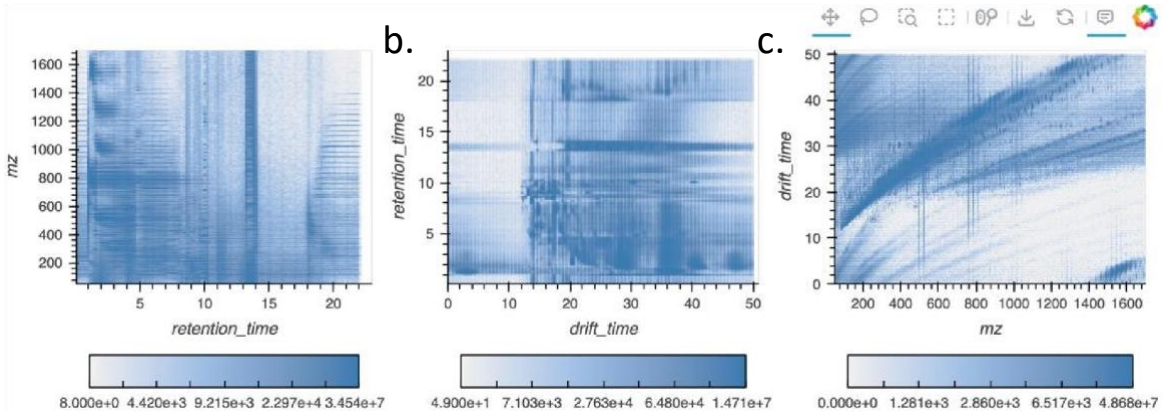
**Table 2.** Time to process the files visualized in this paper on a MacBook Pro 13.6.1with a 2.6 GHz 6-Core Intel Core i7 processor and 16 GB memory. The initial file is 1.12 GB in size

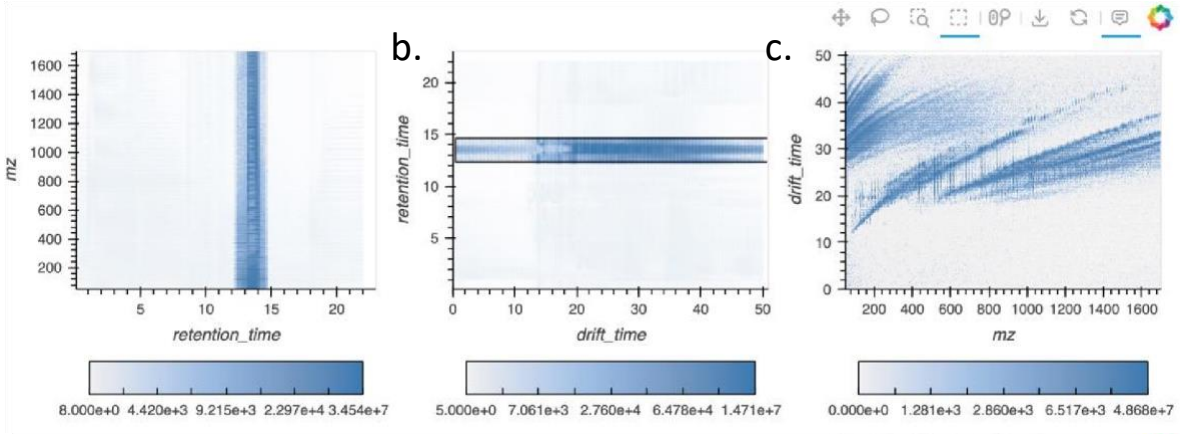
| **Process** | **Time (in seconds)** |
|---|---:|
| Load Initial Data | 19 |
| Smoothing | 64 |
| Peak Detection | 60 |
| Deconvolution | 47 |
| Calibration | 17 |
| Isotope Detection | 7 |
| Plot Alignment | 91 |

When using DataShader to plot MS data, DataShader first determines the ranges of the x- and yaxis dimensions and then divides the plot into grids and uses an operator function for the data within each grid space. For the MS data, the aggregator used was the sum of the intensity for all data points that fall into a grid. We used the Holoviz histogram equalization transformation. This ensures that an equal number of data points are assigned to each color in the color spectrum and ensures that a single high value will not render the other values indistinguishable from each other.

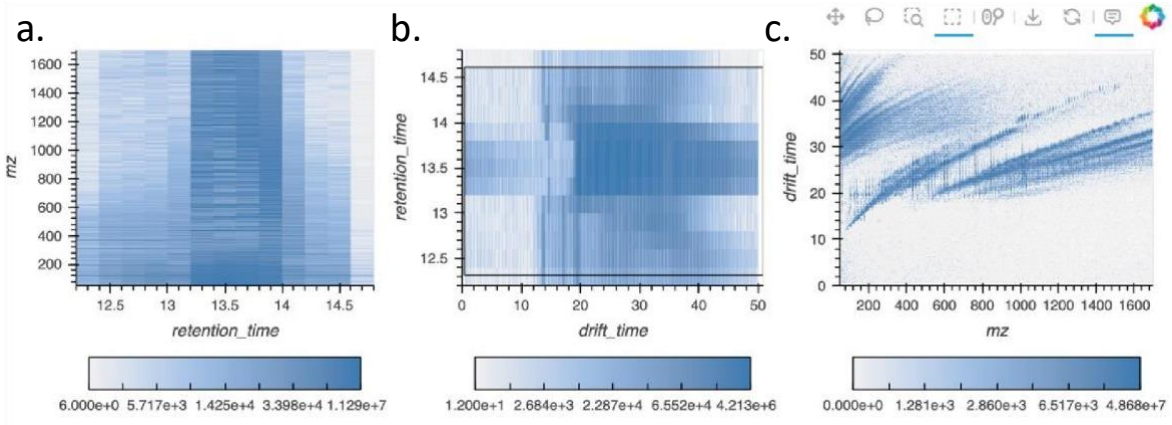
We used the Parameterized class to organize which functions would need to rerun after each user input. Within this class, it is possible to indicate which functions were downstream or upstream from parameter changes and only run functions downstream of the parameter changes (13). For example, changing the minimum size of the rasterized squares in DataShader causes downstream functions to run, such as the rasterizing function, but upstream functions, such as the original functions to load the input data, are not triggered to rerun. The only function that reran

entirely with a change in user input was the alignment function, which was necessary as the number of plots returned can vary depending on user input.


b.
c.
mz
retention_time
drift_time
8.000e+0 4.420e+3 9.215e+3 2.297e+4 3.454e+7
4.900e+1 7.103e+3 2.763e+4 6.480e+4 1.471e+7
0.000e+0 1.281e+3 2.860e+3 6.517e+3 4.868e+7


b.
c.
mz
retention_time
drift_time
8.000e+0 4.420e+3 9.215e+3 2.297e+4 3.454e+7
5.000e+0 7.061e+3 2.760e+4 6.478e+4 1.471e+7
0.000e+0 1.281e+3 2.860e+3 6.517e+3 4.868e+7

3. a.


b.
c.
mz
retention_time
drift_time
6.000e+0 5.717e+3 1.425e+4 3.398e+4 1.129e+7
1.200e+1 2.684e+3 2.287e+4 6.552e+4 4.213e+6
0.000e+0 1.281e+3 2.860e+3 6.517e+3 4.868e+7

1. a.

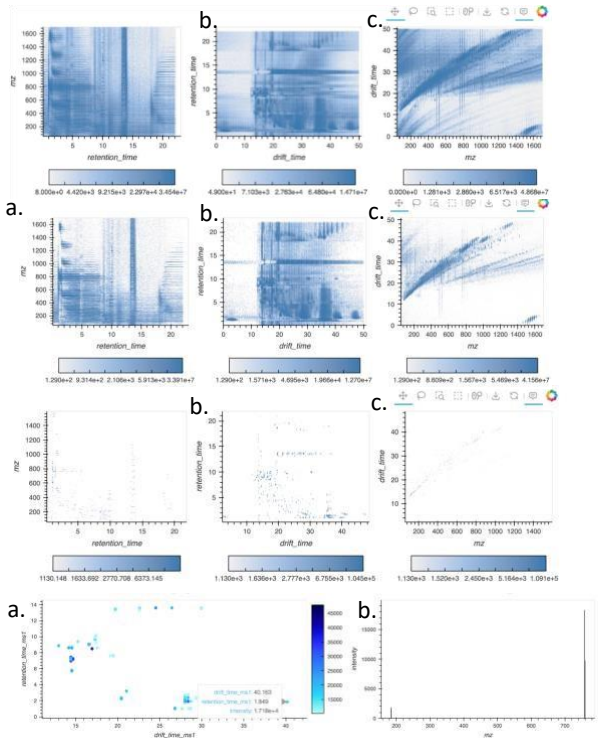
2. a.

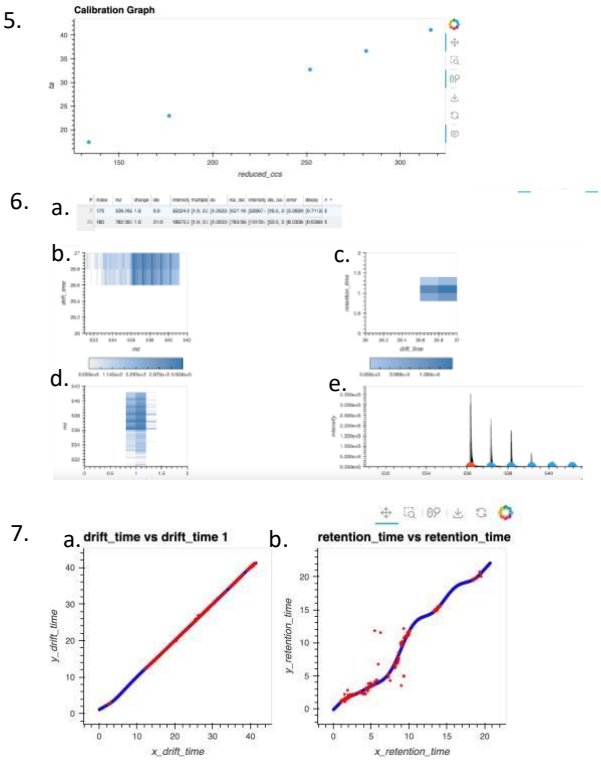
**Figure 1.** Datashader graphs: 1) initial data for all the dimensions, 2) the results of using "box select" on the middle drift time vs retention time plot, and 3) the result from filtering the axis ranges in the user inputs on the middle drift time vs retention time plot after using "box select".

The "box select" built-in HoloViews tool and the linked selection functions ensured that the plots are linked based on underlying data (12). The "zoom" tool will only filter the x and y range on the dimensions visible in the plot for other plots with shared axes, but the "box select" tool used with the Holoviz linked selection function allows the plot data to highlight selected data in linked plots based on the underlying data (**Figure 1**).

The primary data used in this demo were Agilent .d files converted to mzML with Proteowizard (16), which are then converted to HDF5 files. All the loaded files are converted to a Dask dataframe and persisted so that the data only needs to be loaded once. Dask (15) reduced

the graph-loading time by approximately 50%. The samples analyzed were LC-IMS-MS/MS metabolomics data from human plasma samples. The features in this data include *m/z* for MS1 and MS2, drift time, retention time, and abundance/intensity columns. Although DEIMoS is agnostic to the number and types of dimensions used, the GUI expects three dimensions to create the plots. However, the GUI does allow different column names to be assigned to these three dimensions.




1. a.

2.

3. a.

4.

**Figure 2. Representative visualizations of LC-IMS-MS/MS data**. **1-3)** The plots of the original MS1 data (1), smoothed data (2), and peak picked data (3), with, from left to right, retention time vs *m/z* (Subplot a), drift time vs retention time (Subplot b), and *m/z* vs drift time dimensions (subplots c), all displayed with DataShader **4)** The MS1 data with assigned MS2 deconvolution data is depicted on graphs with retention time vs *m/z*, drift time vs retention time, and *m/z* vs drift time (showing drift time vs retention time in Subplot a). The Subplot b plot displays the selected MS2 data associated with user-selected MS1 data, with the MS1 data with the highest intensity used if there are multiple MS1 data points within a small range. The range is set by user inputs **5)** The reduced collision cross section values vs arrival time values of the calibration model **6)** The matched isotopes (Subplot e) of the selected row in table (Subplot a) with graphs (Subplots b-e) of the slice of MS1 data within a range of the selected value **7)** The blue line represents the effect of the alignment function (support vector regression with a rbf kernel) on a straight line, and the red dots represent the data in files in folder (x-axis) to a reference file (y-axis) by drift time (Subplot a) and retention time (Subplot b).

The visualization from panels from the application are shown in **Figure 2**. This shows the original MS1 data (**Figure 2.1)**, the result of smoothed MS1 data **(Figure 2.2),** and the result of peak detection **(Figure 2.3).** The dimensions used are *m/z* vs retention time (**Subplot a**), retention time vs drift time (**Subplot b**), and drift time vs *m/z* (**Subplot c**). **Figure 2.4** shows the result of MS2 deconvolution, where MS2 fragments are assigned to MS1 data. The graph **(Subplot b)** shows the MS2 deconvolution data assigned to the MS1 with the highest intensity within a user-selected area of MS1 plots (**Subplot a**). **Figure 2.5** shows the calibration model's

reduced collision cross section values vs arrival time in a graph after being calibrated by a collision cross section calibration mix (CCS). **Figure 2.6** shows a graph of the isotopes (**Subplot e**) of the selected matched isotopes from table **(Subplot a)** after calculating the isotopes with the data. Finally, **Figure 2.7** shows the drift time **(Subplot a)** and retention time (**Subplot b**) of a MS1 data file in the x-axis and the drift time and retention time of a reference MS1 in the y-axis of the plots, with the blue line representing the effect of the alignment function.

The GUI saves the DEIMoS intermediate and final outputs by input name and parameter information and uses the saved files in new sessions if the arguments, including the input file, are the same. The user input panels for peak detection, smoothing, MS2 deconvolution, calibration, isotope detection, and alignment are shown in **Figure 3**. One of the user-inputs is the names of each of the dimensions used in the graph. In our examples, these dimensions are drift time, retention time, and *m/z*, but the user could use any dimension in their own data.

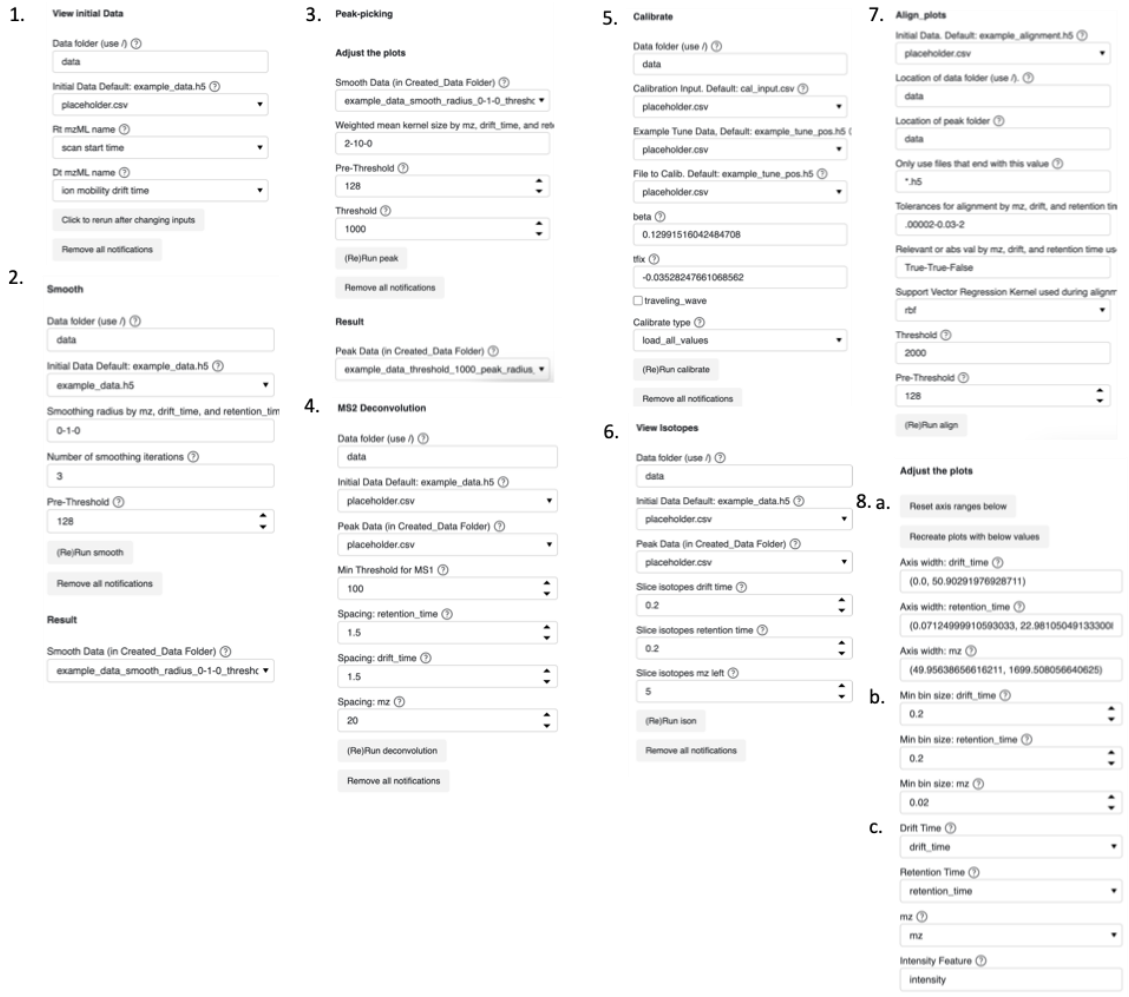



**Figure 3**. User-inputs: The user input for 1) the original data and the functions for 2) smoothing, 3) peak-picking, 4) MS2 deconvolution, 5) calibration, 6) isotopes, 7) alignment, and 8) plot adjustment with (a) the manual axis range adjustment (b) the minimum bin size for each feature and (c) the dimension names.

In addition to running this application locally, it's also possible to run the script on a research computer cluster. This was accomplished with secure shell (SSH) tunneling, where the Panel app was run on the research computing cluster but connected to a local port so it can be interacted with via the user's personal workstation. The advantage was that large files did not have to be downloaded to a local machine and we could use the central processing units (CPUs) and graphics processing units (GPUs) on the research computing cluster if necessary. Although we have not

deployed the application to a hosting platform, Panel is built on top of Bokeh, and the Bokeh server allows straightforward deployment to hosting services (17)

## Discussion

Using Holoviz stack of Python packages enabled us to create a GUI for the DEIMoS Python package that allowed 1) informative aggregations 2) efficient interactivity, and 3) linked selections. In general, the libraries were compatible with each other and were useful in creating an interactive GUI for mass spectrometry data. This is an useful open-code example of combining DataShader (7) and Param (13) Parameterized classes to create an interactive aggregation visualization of MS data. We believe the code would be useful for additional opensource MS data visualization and that the GUI will increase the accessibility of the DEIMoS Python package.

## Data and Software Availability

The software is available at https://github.com/pnnl/deimos_gui/.

The links to the example data shown are available here:

https://deimos.readthedocs.io/en/latest/getting_started/example_data.html

## Supporting Information

### Data Information (Supplemental)

To demonstrate the DEIMoS GUI, we used LC-IMS-MS/MS data from an Agilent 1260

Infinity II high flow liquid chromatography system (San Jose, CA). The details of the data are described in Colby et al. The data visualized in this paper can be found at https://deimos.readthedocs.io/en/latest/getting_started/example_data.html.


## Funding Sources

This research was supported by the National Institutes of Health, National Institute of Environmental Health Sciences grant U2CES030170 and is a contribution of the Pacific Northwest Advanced Compound Identification Core. A portion of the work was conducted under the Laboratory Directed Research and Development Program at Pacific Northwest National Laboratory (PNNL). PNNL is a multi-program national laboratory operated by Battelle for the DOE under Contract DE-AC05-76RLO 1830.


## Author Contributions

- Marjolein Oostrom: Software (Lead), Visualization (Lead), Writing – Original Draft Preparation (Lead) and Writing – Review & Editing (Equal)
- Sean Colby: Formal Analysis (Lead), Supervision (equal), Writing – Review & Editing (Equal), Conceptualization (Equal), Software (supporting), and Writing – Original Draft Preparation (supporting)
- Tom Metz: Funding Acquisition (Lead), Project Administration (Lead), Supervision (equal), Writing – Review & Editing (Equal), Conceptualization (Equal), and Writing – Original Draft Preparation (supporting)

## References


1. Naegele E. Agilent MassHunter–Fast, computer-aided analysis of LC/ESI-TOF data from complex natural product extracts. Agilent Application Note, publication number 5989-5928EN.
2. Schmid R, Heuckeroth S, Korf A, Smirnov A, Myers O, Dyrlund TS, et al. Integrative analysis of multimodal mass spectrometry data in MZmine 3. Nature biotechnology. 2023;41(4):447-9.
3. Tsugawa H, Cajka T, Kind T, Ma Y, Higgins B, Ikeda K, et al. MS-DIAL: data-independent MS/MS deconvolution for comprehensive metabolome analysis. Nature methods. 2015;12(6):523-6.
4. Röst HL, Sachsenberg T, Aiche S, Bielow C, Weisser H, Aicheler F, et al. OpenMS: a flexible open-source software platform for mass spectrometry data analysis. Nature methods. 2016;13(9):741-8.
5. Colby SM, Chang CH, Bade JL, Nunez JR, Blumer MR, Orton DJ, et al. DEIMoS: an opensource tool for processing high-dimensional mass spectrometry data. Analytical chemistry. 2022;94(16):6130-8.
6. Hunter JD. Matplotlib: A 2D graphics environment. Computing in science & engineering. 2007;9(03):90-5.
7. Holoviz. Datashader [updated 2023-02-02. Available from: https://datashader.org/.
8. Petras D, Phelan VV, Acharya D, Allen AE, Aron AT, Bandeira N, et al. GNPS Dashboard: collaborative analysis of mass spectrometry data in the web browser. Biorxiv. 2021.
9. Willems S, Voytik E, Skowronek P, Strauss MT, Mann M. AlphaTims: Indexing trapped ion mobility spectrometry–TOF data for fast and easy accession and visualization. Molecular & Cellular Proteomics. 2021;20.
10. Voytik E, Skowronek P, Zeng W-F, Tanzer MC, Brunner A-D, Thielert M, et al. AlphaViz: Visualization and validation of critical proteomics data directly at the raw data level. bioRxiv. 2022:2022.07. 12.499676.
11. Schessner JP, Voytik E, Bludau I. A practical guide to interpreting and generating bottom-up proteomics data visualizations. Proteomics. 2022;22(8):2100103.
12. HoloViz. Holoviews: Anaconda Inc.; [updated 2023-01-16. Available from: https://holoviews.org/
13. HoloViz. Param: Anaconda Inc.; [updated 2022-12-09. Available from: https://param.holoviz.org/
14. Holoviz. Panel [updated 2023-01-31. Available from: https://panel.holoviz.org/
15. Dask. Dask [Available from: https://www.dask.org/.
16. Chambers MC, Maclean B, Burke R, Amodei D, Ruderman DL, Neumann S, et al. A crossplatform toolkit for mass spectrometry and proteomics. Nature biotechnology. 2012;30(10):918-20.
17. Team BD. Bokeh: Python library for interactive visualization. 2018.

## For Table of Contents Only

Original data

Smoothed data

Peak Picked Data