Title

BASTet: Shareable and Reproducible Analysis and Visualization of Mass Spectrometry Imaging Data via OpenMSI

Journal

IEEE Transactions on Visualization and Computer Graphics, 24(1)

ISSN

1077-2626

Authors

Rübel, Oliver

Bowen, Benjamin P

Publication Date

2018

DOI

10.1109/tvcg.2017.2744479

BASTet: Shareable and reproducible analysis and visualization of mass spectrometry imaging data via OpenMSI

Oliver Rübel and Benjamin P. Bowen

Abstract—Mass spectrometry imaging (MSI) is a transformative imaging method that supports the untargeted, quantitative measurement of the chemical composition and spatial heterogeneity of complex samples with broad applications in life sciences, bioenergy, and health. While MSI data can be routinely collected, its broad application is currently limited by the lack of easily accessible analysis methods that can process data of the size, volume, diversity, and complexity generated by MSI experiments. The development and application of cutting-edge analytical methods is a core driver in MSI research for new scientific discoveries, medical diagnostics, and commercial-innovation. However, the lack of means to share, apply, and reproduce analyses hinders the broad application, validation, and use of novel MSI analysis methods. To address this central challenge, we introduce the Berkeley Analysis and Storage Toolkit (BASTet), a novel framework for shareable and reproducible data analysis that supports standardized data and analysis interfaces, integrated data storage, data provenance, workflow management, and a broad set of integrated tools. Based on BASTet, we describe the extension of the OpenMSI mass spectrometry imaging science gateway to enable web-based sharing, reuse, analysis, and visualization of data analyses and derived data products. We demonstrate the application of BASTet and OpenMSI in practice to identify and compare characteristic substructures in the mouse brain based on their chemical composition measured via MSI.

Index Terms—Mass spectrometry imaging, Data provenance, Visualization, Data management, Analysis Workflows, Data sharing.

1 Introduction

Mass Spectrometry Imaging (MSI): MSI [11, 36] is a transformative imaging method that enables the simultaneous, label-free, high-resolution measurement of the spatial distribution of thousands of molecules (e.g. lipids, proteins, natural products, etc.) for quantitative analysis of the chemical composition of complex, biological samples. MSI supports the detailed investigation of metabolic processes at scales ranging from subcellular to centimeter resolution.

Fig. 1 illustrates a common approach towards acquisition of MSI data by raster scanning a laser or ion beam across a sample. At each scan location molecules are desorbed from the sample surface, often with the assistance of a matrix coating or specially prepared surface to promote the formation of gas phase ions. The generated ions are then collected and analyzed via modern mass spectrometry instruments with very high, single electron-mass-level accuracy. The result is an image of typically 100^2 to 1000^2 discrete locations, each containing one or multiple mass spectra with typically 10^4 to 10^6 bins in m/z. Today, common MSI datasets are on the order of five to tens of GB, and can range in some cases to hundreds of GB to TBs for very large images [23]. While the data can be routinely collected, the broad application of MSI is currently limited by the lack of easily accessible analysis methods to explore and process data of the size, volume, diversity, and complexity generated by MSI experiments.

OpenMSI: The goal of the OpenMSI project is to overcome these challenges by making the most high-performance, advanced data management, model building, analysis and visualization resources for mass spectrometry imaging accessible to scientists via the web. OpenMSI started as a grassroots effort in 2012 and has since grown to become a public resource for storage, sharing, visualization, analysis, and management of MSI data (see `http://openmsi.nersc.gov`).

Rübel and Bowen et al. [45, 7] first introduced OpenMSI and described a novel, optimized HDF5-based data format for storage of raw MSI data. Using this novel format enables OpenMSI to accelerate image access operations by up to more than 2000× compared to traditional MSI binary formats, enabling spectrum and image retrieval from raw MSI datasets in less than $0.3s$ across the Internet even for large MSI data sets. The authors also introduced a simple web API to enable easy and fast retrieval of data subsets via the web. Based on this REST API, Rübel et al. also briefly introduced an early prototype of a web-based image and spectrum viewer for exploration of raw MSI data (an example of the current version is shown later in Fig. 9). Fischer et al. [23] then described the use of the OpenMSI web API to analyze raw MSI data in the browser via programmable Jupyter notebooks. These prior publications have focused mainly on the programmatic interaction with raw MSI data via a REST API.

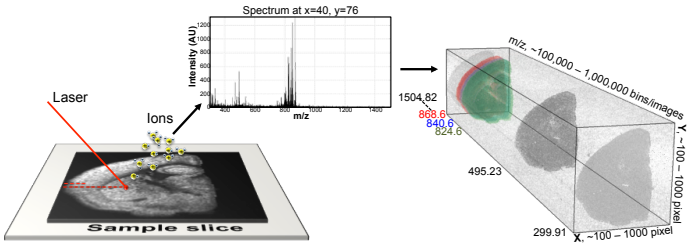

Fig. 1. Mass spectrometry images are commonly acquired by raster-scanning a laser across the surface of a sample slice followed by processing of the generated ions via mass spectrometry.

Challenge: The development and application of cutting-edge analysis methods is a core driver in MSI research for new scientific discoveries, commercial-innovations, and development of novel diagnostics. However, advancement of the state-of-the-art in MSI through the broad application, validation, and use of novel MSI analysis methods is critically hindered by the lack of means to share, apply, and reproduce analyses. To facilitate the in-depth study of MSI datasets and to enable the MSI research community to build an ecosystem of advanced analysis methods and protocols, it is critical that we can make advanced analyses and their results easily accessible to domain scientist for validation, reuse, and interpretation.

Contributions: In this manuscript we introduce for the first time the **B**erkeley **A**nalysis and **S**torage **T**oolkit (BASTet) (Sec. 3). BASTet serves as the analysis backend for OpenMSI and enables shareable and reproducible analysis of MSI data by providing critical support for:

- common interfaces to standardize the definition and use of data analyses (Sec. 3.1),
- data provenance for reproducible analyses (Sec. 3.2),
- standardized storage to facilitate sharing and reuse of analysis re-

- *Oliver Rübel is with the Computational Research Division, Lawrence Berkeley National Laboratory (LBNL) E-mail: oruebel@lbl.gov.*
- *Benjamin P. Bowen is with the Environmental Genomics & Systems Biology Division, LBNL, E-mail: bpbowen@lbl.gov.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

sults (Sec. 3.3),
- workflow management to facilitate the combination of analyses to solve complex problems (Sec. 3.4), and
- a broad set of tools and integrated analytics to facilitate the use and development of analysis methods and workflows (Sec. 3.5).

Based on BASTet, we describe the extension of the OpenMSI science gateway to support web-based visualization, management, and analysis of MSI data and derived analysis data products (Sec. 4). Specifically, we present:

- the extension of OpenMSI's web API to enable interaction with arbitrary, derived analysis data (Sec. 4.1), and
- we describe for the first time the visualization, analysis and data management capabilities of the OpenMSI web service and science gateway (Sec. 4.2).

Finally, we demonstrate the application of BASTet and OpenMSI in practice to study characteristic structures in the left coronal hemisphere of a mouse brain and to identify ions important to the chemical makeup of these structures (Sec. 5.1) and discuss the broader impact to MSI (Sec. 5.2).

2 Related Work

Supporting the evolution and life-cycle of scientific analysis software and the derived data products they generate has been the focus of several different, complementary technologies, mainly **i)** analysis frameworks, **ii)** provenance and workflow management systems, and **iii)** standardized data formats.

Analysis Frameworks: At a high level, some data analysis communities have formed around widely adopted, extensible programming environments—e.g., R, Matlab or Python—that provide easy access to a broad range of analysis packages. To create a stable foundation of algorithms, many communities have developed standard libraries and toolkits, e.g., VTK [50, 38] for visualization, ITK [30] for segmentation and registration, or ImageJ [48] for image processing, among others. Standard libraries play a critical role in defining and collecting the state-of-the-art in the respective domains. Higher-level tools often build on these libraries—e.g., VisIt [20] or ParaView [1] (VTK) or Fiji (ImageJ) [47]—to create user-oriented applications.

Provenance and Workflow Management: Provenance and management of complex, interdependent analytics has traditionally been a research focus of workflow and provenance management systems, e.g., VisTrails [4], Pegasus [16], Fireworks [28], among many others. In the context of visualization, VisTrails has notably been integrated with advanced visualization systems, e.g., ParaView [9] or UV-CDAT [14].

Standardized Data Formats: To facilitate persistent storage and exchange of analysis data, the scientific community utilizes a broad range of data formats, each of which typically focuses on different levels of the data organization and storage problem. In practice, low-level text and binary formats, e.g., CSV, BOV, JSON, or XML, are still commonly used for data analysis. While JSON, XML, and other standardized, text-based formats are portable, they are impractical for storage and exchange of large scientific data due to large overheads and cost for storage, transfer, and I/O. Self-describing formats, such as HDF5 [53] and NetCDF [44], have gained wide popularity to store large-scale scientific data. A diverse range of application sciences have adopted self-describing formats to define application-specific data standards. Examples include, among many others, the NeXus [31] format for neutron, x-ray, and muon data, the OpenMSI format for mass spectrometry imaging data [45], or the CXIDB format [35] for coherent x-ray imaging. To bridge the gap between general, self-describing formats and the need for standardized tools for data exchange, processing, and interpretation, formats like VizSchema [51] and XDMF [12], have proposed to augment HDF5 via lightweight, low-level schema (e.g., via XML) to further specify the data organization. Database technologies, e.g., SciDB [8], MongoDB [37], PostgresSQL [42], and many others, are also used to facilitate scientific data analytics and standardized data access.

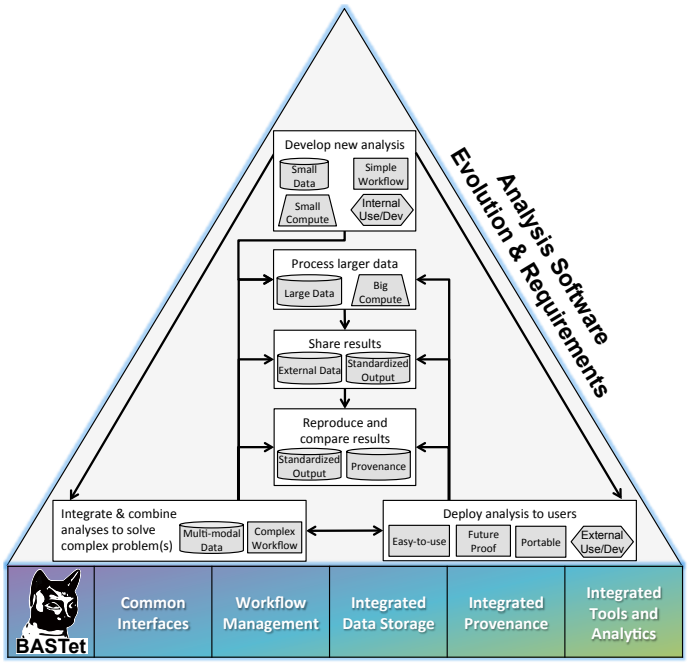

Fig. 2. Overview of the typical evolution of scientific data analyses for research and development (gray pyramid). The white boxes show common stages of development and use of analysis software. As we transition between the different phases (arrows) new requirements arise (gray inset shapes), making transitions challenging and costly. BASTet supports this process through common interfaces for analyses and data and a broad set of integrated features and tools for data storage, provenance, workflow management, and analysis (colored boxes).

3 BASTet: Integrated Data Analysis, Provenance, Workflow Management, and Storage

Scientific analysis software for research and development evolves, and in this process undergoes many transitions, e.g., from a research prototype, to production, to the integration with other components for complex analytics, as well as adoption to larger data or new applications, and sharing, publication, reproduction, and reuse of results (see Fig. 2). As our analysis softwares transition between these various different phases, new—often disruptive—requirements arise, making transitions challenging and costly and often hindering the adoption of new analyses in practice. This is particularly critical in growing and transformative science applications, e.g., mass spectrometry imaging, where the state-of-the-art in data analysis is evolving rapidly and new analysis strategies are constantly being proposed and developed.

The goal of BASTet is to help bridge the gaps between these critical facets of data analysis and to provide users with an environment that makes it easy to develop and deploy analyses via OpenMSI by providing standardized analysis interfaces (Sec. 3.1), automatic provenance (Sec. 3.2), standardized storage (Sec. 3.3), integrated workflow support (Sec. 3.4) and tools (Sec. 3.5).

While the design of BASTet has been motivated by the needs of OpenMSI, its core design and functionality are much more broadly applicable to other applications as well. BASTet is implemented in Python using `NumPy` for data processing, `h5py` for HDF5-based data storage, and `mpi4py` for distributed parallel data processing. We have publicly released BASTet in conjunction with this manuscript via `https://openmsi.nersc.gov/openmsi/client/bastet.html`

3.1 Common Interfaces: Standardizing the definition and use of analyses

Standardization of interfaces is central to enable the structured use of software and data. However, in many applied science domains like MSI, the majority of users and developers of new, dedicated analytics

are domain experts with varying degrees of expertise and experience in computing. To achieve adoption and enable domain experts to create new analyses that are accessible to the science community, it is critical that our solutions are **i)** easy-to-use, providing a low barrier of entry and enabling a continuous learning process, and **ii)** that we provide developers early on with desirable features that provide incentives to use the system from the beginning of the development process while providing access to advanced features as analyses mature.

BASTet provides developers with two easy ways to integrate new analyses (Suppl. 2). First, the simplest and most basic solution is to wrap analysis functions using either the `@bastet_analysis` Python decorator or by explicitly wrapping a function via `wrapped_funct = analysis_generic.from_function(my_function)`. The latter approach has the advantage that it allows us to easily generate multiple instances of a function that we can use and track independently. The ability to simply wrap functions eases early development and facilitates tracking of results from one-off analyses.

Second, to easily share and fully integrate an analysis with BASTet, we then create a new class that inherits from BASTet's base analysis interface class. Creating a new derived analysis class requires only minimal effort. In the constructor we need to **i)** define the inputs of our analysis using a similar syntax to Python's `argpars` module that many developers are already familiar with and **ii)** specify the names for our outputs. Finally, we implement our analysis as part of the `execute_analysis()` function (see Suppl. 2.3).

Once we have completed the basic integration using any of the above-described approaches, we immediately have access to a broad set of desirable features that already early on simplify the development of new analyses and that ease transitions as our software evolves throughout its life-cycle (see also Fig. 2):

- **Reproduce and compare results:** BASTet automatically tracks the provenance of analyses, so that we can directly trace their history and restore, reproduce, and profile analyses. (Sec. 3.2).
- **Share results:** BASTet implements a standard API and format for storing analysis results in OpenMSI HDF5 files (Sec. 3.3). Once we have integrated our analysis, we can, hence, immediately save and share our results in a common format. The integration with the file format also enables us to immediately share and access analysis results via OpenMSI's web services (see Sec. 4.1).
- **Integrate and combine analyses to solve complex problems:** BASTet supports delayed execution and direct integration of analyses with other analysis modules to define complex workflows and shareable workflow scripts. (Sec. 3.4).
- **Deploy and apply:** BASTet provides standardized interfaces and tools for executing analyses and workflows. This allows us to execute analyses using common tools and interfaces, facilitating deployment to users and integration with other systems (Sec. 3.5).
- **Process larger data:** BASTet supports storage of large collections of analysis data via HDF5 and facilitates the parallelization of analyses across spectra, images, and other data subsets via helper classes for parallel scheduling of analysis tasks via MPI (Sec. 3.5)
- **Develop new analysis:** Standardized tools for execution, storage, provenance, workflows, and support for memory and runtime profiling simplify development and validation of new analyses.

3.2 Provenance: Making analyses reproducible

Reproducibility of analysis results is critical in many applications of MSI, e.g., science or health, and is essential for data reuse and sharing. Data provenance provides a historical record of data and its origins by documenting the inputs, entities, systems, and processes that influence the data of interest. Provenance data provides critical evidence to study data dependencies, detect and recover errors, and facilitate interpretation and auditing of analyses. Provenance information can also be a resource for "reflection-in-action,"—i.e., to support planning and reframing of objectives—, as well as after the fact to support the interpretation of claims, audit, accountability, or training. To enable users to easily reproduce analyses, BASTet automatically collects a broad range of provenance data.

For all analyses, BASTet automatically records all inputs and outputs as well as the name of the analysis class. For analyses that have been wrapped or created dynamically, the analysis function or class itself is recorded as well. We here use `cloudpickle` [41] to serialize such dynamically created analyses, as it allows us to serialize a much broader set of objects, including functions defined dynamically in the Python interpreter, lamdas and nested functions, and a range of other objects, that are unsupported by the standard Python `pickle` module.

Beyond this basic provenance data, BASTet also automatically detects and creates dependencies. Dependencies are descriptions of links to other data an analysis depends on. Dependencies may point to data objects in OpenMSI HDF5 files as well as to outputs of other BASTet analyses in memory that may or may not have been computed yet. For each dependency we store: **i)** the name of the target analysis parameter, **ii)** the source of the dependency described via a combination of the name of the dataset, the Python object that contains the dataset (e.g, an instance of another analysis or file object manager), and an optional selection describing the relevant subset of the data. In addition, we also store **iii)** an optional help string, **iv)** the type of the dependency, and **v)** the link name to be used for storage. From a user's perspective the creation of dependencies is handled transparently and automatically as part of the assignment of parameter settings. As such, a user can use an analysis implemented in BASTet as usual, without the need to learn complex new interfaces. Parameters may be set prior to the execution of analyses using a Python dictionary-like interface or as part of the call to execute the analysis directly.

When an analysis is executed, we also automatically collect basic runtime data, such as, **i)** start, stop, and execution times, **ii)** architecture, system, and software metadata, e.g., OS and library versions, processor metadata etc., and **iii)** runtime metadata, e.g., basic memory usage data. In addition, BASTet can optionally collect detailed execution profiling data using Python's `Profile` module and/or memory profiling data via `memory_profiler` [40]. For MPI-based, parallel analyses, BASTet transparently supports resolution of dependencies and collection of runtime data from all compute cores.

When saving an analysis to file (described next in Sec. 3.3) all provenance data is saved as well. With the provenance of the analysis captured, we can then restore and/or recreate an analysis from file via a single function call. This makes it trivial for users to load prior analyses to: **i)** visualize and reuse results (see Suppl. 1.4), **ii)** expand a prior analysis workflow (see Suppl. 1.2), **iii)** modify an analysis by re-executing it with updated parameters (see Suppl 1.3), **iv)** recreate an analysis to validate and confirm results (see Suppl 1.3), or to **v)** compare results from different analyses.

3.3 Storage: Making analyses shareable

State-of-the-art: In MSI research today, sharing and reuse of analysis results is fundamentally hindered by a lack of standardized methods for storage of analysis results. Currently, a broad range of proprietary, commercial data standards by instrument manufacturers, e.g, Brucker, Thermo, or Waters, and few open standards, e.g., imzML [49] and mzML [21], are being used to store MSI data. While the XML-based mzML format is a widely used standard in mass spectrometry, it is inefficient for storage of imaging data due to the large overheads for storing large data volumes in ASCII form. The imzML format extends mzML by storing metadata in an mzML-like XML file while the actual MSI data is stored separately as a raw, flat binary file. A central advantage of imzML, mzML and the OpenMSI data format are their transparency, portability, low cost, and compatibility with standard laboratory automation. However, none of the existing formats support standardized storage of arbitrary analysis data.

Among the existing MSI formats, the OpenMSI data format [45] is unique in that it is extensible and based on an open, self-describing data standard (i.e., HDF5) that is portable across computing platforms and natively supported by languages, e.g., Matlab, R, C/C++, or Fortran. Our format also supports storage of large collections of data, lossless data compression, and optimization of common MSI data access operations via data replication to enable fast access to both spectra and images. In the format, data is organized hierarchically into groups

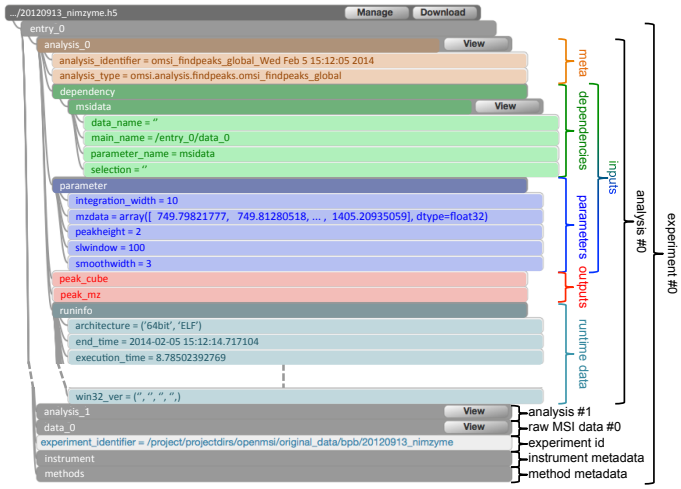

Fig. 3. Example showing the basic data layout for storage of analysis results. HDF5 groups are shown in dark-gray and datasets in light gray (attributes not shown).

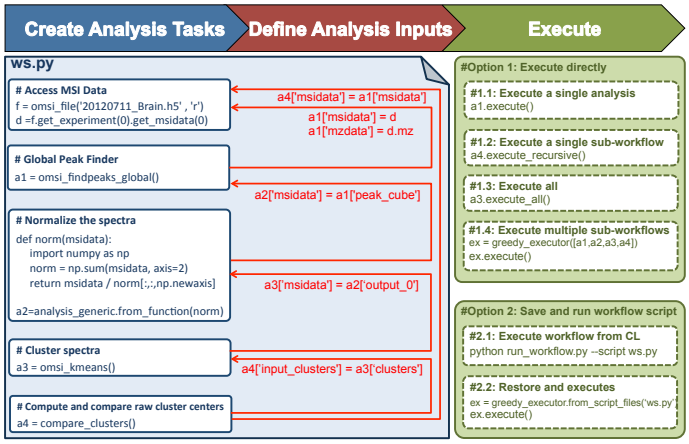

Fig. 4. Illustration showing the main steps and code for defining and executing an example analysis workflow for clustering MSI data. The workflow is defined via a standard Python script with instructions for creating analysis tasks (blue) and defining their inputs (red). We can then execute the workflow, or components of it, directly or save the script to file (here ws.py) to enable reuse and scheduled execution (green).

(see Fig. 3). A HDF5 group can be thought of as a folder within a file that may contain an arbitrary number of other groups or datasets (i.e., n-dimensional arrays). Each file may contain an arbitrary number of experiments (`/entry_#`) consisting of an arbitrary number of raw MSI measurements, experiment-level metadata about the instrument and methods, and a user-defined identifier. We here describe the extension of this format to support storage of arbitrary analysis data.

Storing arbitrary analysis data: The modular design of the OpenMSI format allowed us to easily extend it via a new module for storing analysis data. All data for a given analysis is stored in a separate group within an experiment (`/entry_#/analysis_#`, Fig. 3). Each analysis stores: **i)** a user-defined identifier, **ii)** all outputs of the analysis, **iii)** all runtime provenance data, **iv)** all user-defined parameters, **v)** all dependencies, and **vi)** the analysis type and, in the case of dynamically defined analyses, also the Python pickle of the analysis class itself. Each analysis dependency is stored in a separate managed subgroup according to its `link_name` and containing all data describing the dependency. Dependencies may point to data stored in the same or in external HDF5 files. We chose to manage dependency information directly in our own file format—rather than using HDF5 links—in order to make the dependencies explicit and enable flexible extension of dependencies with additional metadata in the future.

Data Storage API: BASTet implements a dedicated API for the OpenMSI file format. The API is object-oriented and utilizes the concept of managed objects. A managed object is a group in the HDF5 file that has a corresponding API class responsible for its management, creation, and interaction. Using the analysis managed object class, creation and interaction with analysis data is simple. To store an analysis we simply hand our analysis object—i.e., the instance of our BASTet analysis or wrapped function—to the `create_analysis` function. The function in turn creates all required objects in the HDF5 file, saves the analysis data, and automatically resolves all dependencies. For dependencies that point to analyses that exist only in memory (i.e., analyses that have not been saved previously), we automatically save the dependent analyses as well. For dependencies that point to analyses that have already been saved, we resolve the link, avoiding redundant storage of analyses. A user may optionally suppress the storage of dependent analyses as well as force storage of all dependencies.

With all dependency data stored in file, we can reconstruct for each analysis all its inputs, outputs, runtime metadata, and direct dependencies on prior analyses or raw MSI datasets. Using this information, BASTet provides convenient functions to directly: **i)** reconstruct the complete provenance graph for a given analysis, describing all analyses and MSI datasets (nodes) and their inter-dependencies (edges), and to **ii)** restore and recreate an analysis from file.

3.4 Workflow Management: Combining analyses to solve complex problems

To enable users to solve complex analysis problems, BASTet supports the integration of analyses to advanced analysis workflows. As illustrated in Fig. 4, BASTet allows us to separate the specification of workflows into three logical steps: **i)** create analysis tasks (blue), **ii)** define analysis inputs (red) and **iii)** execute analyses (green). The creation of analysis tasks typically consists simply of the instantiation of the required analysis objects. We then typically define the inputs of analyses prior to their execution using a Python dictionary-like syntax. As described earlier in Sec. 3.2, as part of this process, we automatically discover and transparently define dependencies between the analyses, while the dependencies may point to outputs of analyses that may not have been computed yet. This strategy allows us to elegantly separate the execution of workflows from their specification and to encapsulate the creation and definition of workflows in simple scripts. Finally, we can execute our analyses (Fig. 4, green) by either **i)** executing analysis tasks individually (Fig. 4, #1.1), **ii)** recursively executing sub-workflows of inter-dependent analyses (Fig. 4, #1.2 and #1.4), or by **iii)** executing the workflow as a whole (Fig. 4, #1.3).

To simplify the use of workflows and ease extension of BASTet's workflow capabilities in the future, we define extensible base classes for both workflow drivers and executors. Workflow drivers are responsible for the setup of workflows, i.e., the creation of analysis tasks and definition of their inputs. Workflow executors then are responsible for controlling the execution of workflows. To ease sharing, setup, and running of workflows, BASTet then provides an easy-to-use command line driver tool, which enables users to run workflows directly from workflow scripts in a standardized fashion (see Fig. 4, #2.1 and Suppl. 1.1). In the workflow script, developers can flexibly define which analysis inputs are fixed and which ones a user should be allowed to customize (see Suppl. 1.1). The command line tool then automatically collects all customizable analysis inputs and exposes them to the users via corresponding command line options. The combination of workflow scripts and standard workflow drivers enables us to easily define complex, shareable, and customizable analysis workflows.

Interactive Visualization and Analytics: In addition to automatic analyses and workflows, analyses may also define interactive visualizations to enable users to manually determine select analysis outputs (Suppl. 3). To support this use model, the developer creates the interactive view in their analysis function and simply returns a dependency object (i.e., the current value) for any outputs that the user will define later interactively. This strategy enables independent components of

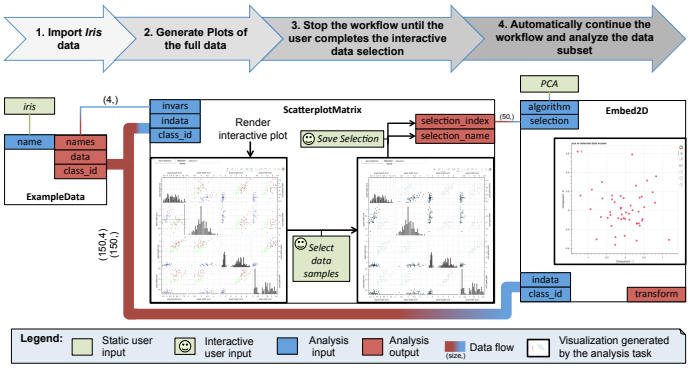

Fig. 5. Example interactive analysis workflow.

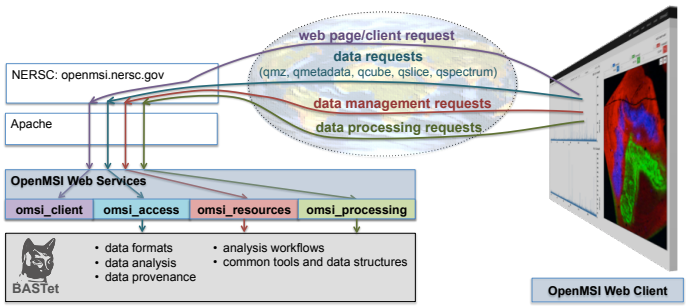

Fig. 6. Schematic of the base architecture of OpenMSI.

a workflow to proceed while the user interacts with the visualization. Once the user has defined the remaining outputs, the analysis then calls its `output_ready` function to notify BASTet that the outputs are available. In case that an interactive analysis blocks the completion of a workflow, BASTet automatically stops the workflow and restarts it once the blocking analysis is ready. As an example, Fig. 5 shows an analysis of Fischer's Iris dataset [24] using BASTet, consisting of a scatterplot matrix analysis to allow the user to explore and interactively select a data subset of interest which is then further analyzed via principal component analysis (PCA). Suppl. 3 describes an extended version of this example in detail. In addition, it is also possible to expand workflows by creating new analysis tasks in response to interactive user inputs (Suppl. 4). The primary focus of BASTet here is to enable sharing and reuse of data products generated by the interactive analyses (e.g., data and parameter selections).

Comparison: Existing, advanced workflow systems, such as Fireworks [28], VisTrails [4], or Pegasus [16], typically follow a process-centric theme with the goal to enable management, optimization, provenance and sharing of workflows. As such, these systems focus on standardization and reuse of workflow specifications but typically do not provide mechanisms for standardized storage of derived data products generated by analysis tasks. In contrast, a main goal of BASTet and OpenMSI is to enable collaborative sharing and reuse of analysis data products. BASTet, therefore, follows an analysis-centric theme where each analysis maintains its own provenance information to enable users to easily reuse analyses in a self-contained fashion and extend existing workflows *post-hoc* through the reuse of derived data products created by select analysis tasks from prior workflows, while still maintaining the ability to retrace and reproduce complete workflows. In contrast to common workflow systems, standardized storage of derived data products (Sec. 3.3) is, therefore, central to BASTet.

The goal of BASTet is not to replace advanced workflow systems, but it is synergistic to them. Through its standardized data storage mechanisms, BASTet fills a critical gap to enable users to share their derived data products. BASTet provides MSI users a lightweight and easy-to-use entry-point to define and manage their workflows without the often steep learning curves and technical requirements of advanced workflow systems, while providing avenues for users to graduate to such systems in the future. Advanced workflow systems can be integrated with BASTet as workflow drivers and executors and analyses integrated with, e.g., VisTrails, could be easily wrapped to integrate them with BASTet. Conversely, analyses defined in BASTet could be easily wrapped to integrate them with VisTrails as well.

3.5 Integrated Tools and Data Analytics

BASTet's standardized command line tools for execution of individual analyses (Suppl. 1.2, paragraph 1), and workflows (Suppl. 1.1, paragraph 3 and Fig. 4, box #2.1), greatly simplify the analysis development process and reduce cost by eliminating the need for custom tools to run and combine different analyses. The ability to run analyses in a standardized fashion as well as to create custom workflows via command line scripts also simplifies deployment to and application and validation by domain scientists (see Suppl. 1.2, 1.3).

To further facilitate development of new analyses, BASTet provides a broad range of integrated features, e.g., for logging, data selection, third-party MSI file formats, or data conversion. For example, in practice, many MSI analyses can be naturally parallelized across spectra or image slices. To facilitate the implementation of scalable, parallel analytics, we provide easy-to-use helper classes for parallel decomposition and execution of analysis tasks via MPI. BASTet also provides a growing set of integrated analytics, e.g., for clustering, matrix factorization, peak detection, or TIC normalization.

4 Web-based Visualization, Data Management, and Analysis

So far we have focused on methods to define, execute, store, combine, and track analyses via BASTet. Next, we describe the extension of the OpenMSI web service (Fig. 6) to enable users to easily access (Sec. 4.1), explore (Sec. 4.2.1), and visualize (Sec. 4.2.2) analysis results. Being able to make analysis data quickly and easily accessible to application experts is critical for evaluation, interpretation, and application of analysis methods for scientific knowledge discovery.

The OpenMSI science gateway is hosted at the National Energy Research Scientific Computing Center (NERSC). The science gateway is implemented in Python using DJANGO [17] and using NEWT [10] to interact with NERSC high-performance compute systems and resources. Visualizations are implemented via client-side rendering using D3 [6]. As illustrated in Fig. 6, the implementation consists of a series of DJANGO apps: i) *omsi_client* manages and serves web pages, ii) *omsi_access* handles all requests for data via OpenMSI's web API (Sec. 4.1), iii) *omsi_resources* implements all data management-related functionality, e.g., SQL database models and DJANGO views for managing file permissions, and finally iv) *omsi_processing* is responsible for creation, submission, and management of all compute jobs, e.g., for data import and analysis. All data storage, processing, and analysis functionality are then implemented by BASTet.

4.1 Web API: Standardized web-based data access

OpenMSI's REST API for interacting with MSI data, as introduced by Rübel et al. [45], consists of five simple functions: **i)** *qmetadata* for retrieval of metadata about files and their content, **ii)** *qcube* to access arbitrary data subsets from HDF5 datasets, **iii)** *qslices* to retrieve image slices from MSI data, **iv)** *qspectrum* to retrieve spectra from MSI data, and **v)** *qmz* for collecting information about static axes of data objects. We here discuss the extension of this base API to enable users to easily interact with arbitrary analysis data.

4.1.1 General Data Access Functions

The *qmetadata* and *qcube* functions enable general-purpose access to the HDF5 files. Once a user has stored an analysis using BASTet, we can immediately access all derived analysis data via these REST functions. Simply by wrapping an analysis, we can, hence, immediately interact with derived analysis data remotely. Similar to the strategies Fischer et al. [23] described for remote analysis of raw MSI data using

Jupyter notebooks, we can now implement the same kind of remote analytics also for arbitrary, derived analysis data.

To enable users to easily locate analysis data, we extended *qmetadata* and *qcube* via the following input parameters: **i)** *anaIndex* to locate an analysis based on its index, **ii)** *anaIdentifier* to locate an analysis based on the user-defined identifier, and **ii)** *anaDataName* to allow selection of a specific dataset from an analysis. In addition, we also expanded *qmetadata* to enable users to retrieve the provenance graph for analyses and other data objects. We encode the provenance graph in JSON via node and edge lists. Each node is defined via a dictionary with the objects metadata and each link is defined by the index of its source, target, and the link type. An illustration of an example provenance graph is shown later in Fig. 10.

4.1.2 Specialized Subset Data Access Functions

The *qslice*, *qspectrum* and *qmz* functions are designed to facilitate specific, common data access operations from spectral imaging data, specifically, the retrieval of image slices, spectra, and descriptions of static data axes, respectively.

In contrast to raw MSI data, there are often many useful ways to define image slices and spectra for a given analysis and the appropriate behaviors are often highly specific. We, therefore, define the behaviors of *qslice*, *qspectrum* and *qmz* for a given analysis type via the corresponding derived BASTet analysis class. For each analysis we may define an arbitrary number of view options to construct images and spectra, respectively. A developer can customize the available view options and define new ones by overwriting the corresponding functions in the derived analysis class. BASTet defines default implementations for all three patterns, which automatically resolve all analysis dependencies and adds their views to the list of view options. This approach enables users to easily navigate all dependencies of an analysis simply by selecting the corresponding view option via the new *viewerOption* parameter of the *qslice*, *qspectrum* and *qmz* patterns. Similar to *qmetadata* and *qcube*, we also added the *anaIndex*, *anaIdentifier*, and *anaDataName* parameters. To support sparse representations of spectra, we then extended the *qspectrum* pattern to allow the return of custom m/z axes on a per-spectrum basis in addition to the specification of global axes via *qmz*.

Being able to define the behavior for specialized data access operations directly in the corresponding analysis class has several critical advantages. First, developers can create new analyses for OpenMSI in a self-contained, plug-in-like fashion, with all analysis-specific behaviors being defined by a single class. Second, resolution and visualization of dependencies is handled automatically. Third, users can retrieve images and spectra for analyses without having to know specific implementation details of the analysis. For example, the required dataset(s) and transformations are typically automatically determined by the specific implementation of *qslice*, *qspectrum* and *qmz* based on the given *viewerOption*. Fourth, developers can make their analyses with minimal effort accessible to users for testing, evaluation, and ultimately production use. Users in turn can access and visualize analyses results in a standardized fashion (Sec. 4.2), significantly lowering the barrier for use and adoption.

4.1.3 Programmable Data Processing Pipelines

When requesting data subsets via the REST API, data is transferred via the internet in JSON, PNG, or binary HDF5 format. To achieve high performance for interactive applications, it is critical to reduce already early on the amount of data that needs to be transferred. For example, visualization of the spatial distribution of ions is often based on normalized projections from tens to hundreds of image slices. By performing such data reductions prior to the data leaving the NERSC HPC network infrastructure, we can greatly reduce the need for large external data transfers.

We previously described the use of a small, fixed set of simple data reduction operations—specifically, maximum, minimum, average, standard deviation, and variance—as part of data requests [45]. However, the specific data reduction operations needed are in practice often complex and highly application specific. We, therefore, extended the *qcube*, *qslice*, and *qspectrum* data retrieval functions to support custom, programmable data reduction pipelines defined via JSON. To balance web-server load and security needs with flexibility, we define a broad list of allowed operations and focus on data-parallel, loop-free processing pipelines as part of web requests. In the JSON specification, users can combine data transformation and reduction operations to define advanced processing pipelines (see Suppl. 6 for further details). We use this functionality in the web-based viewer (Sec. 4.2.2), to reduce data transfer cost by computing normalized ion images as part of *qslice* requests directly while allowing the user to easily customize normalization parameters via the UI.

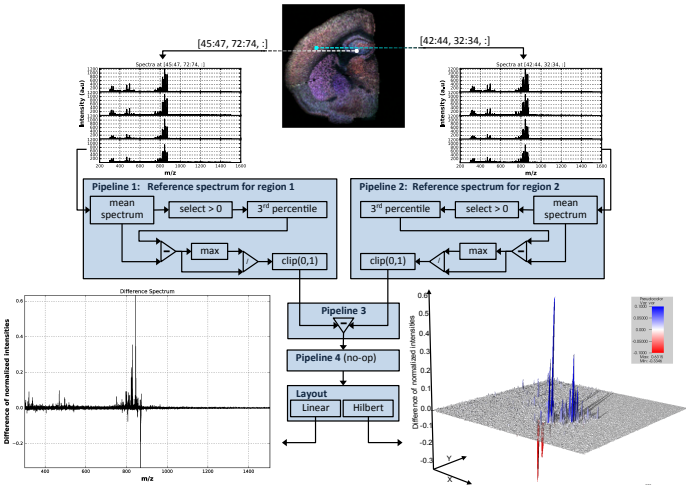

Fig. 7. Illustration showing a single *qspectrum* web request for retrieval of the difference spectrum of the normalized, background-subtracted, average spectrum from two distinct 2×2 regions in space.

To facilitate common comparative analyses of spectra, we further extended *qspectrum* to enable retrieval of two sets of spectra, each of which may be processed via separate user-defined processing pipelines. The two spectra are then combined via a third pipeline and the resulting spectrum can optionally be further processed via a fourth pipeline. Finally, a user may choose between a standard 1D linear and a 2D hilbert curve [2] layout for the spectrum data.

Fig. 7 illustrates a common use-case for such advanced spectrum requests. We here first compute the normalized, background-subtracted, average spectrum for two distinct 2×2 regions in the left coronal hemisphere of a mouse brain. Finally, we subtract the two average spectra and select the spectrum layout. The resulting difference spectrum aids in the identification of differences in the chemical composition of the two tissues (see Suppl. 6.2).

4.2 Web interfaces: Visualization and Management of MSI Data and Analyses

The typical workflow for a user interacting with the OpenMSI science gateway consists of the following steps (see Suppl. 7 for renderings of the main pages). First, the user uploads their raw MSI data to NERSC using the embedded Globus web service [26] via the data upload page or other alternative methods (e.g., ftp). Using the data import page, the user then initiates the file conversion to HDF5 and further data processing steps (e.g., NMF or peak detection). Data processing is performed via scheduled jobs using the Cori and Edison high-performance compute systems at NERSC. A user can view and manage their compute jobs directly via OpenMSI's job management page. Once the data has been imported, users can **i)** browse and explore files via the integrated file browser (Sec. 4.2.1 and Fig. 8) **ii)** visualize the data via the integrated web-based viewer (Sec. 4.2.2 and Fig. 9) **iii)** share the data with others (Fig. 8, b) and, as mentioned in Sec. 4.1, **iv)** access their data programmatically via the REST API for remote analytics.

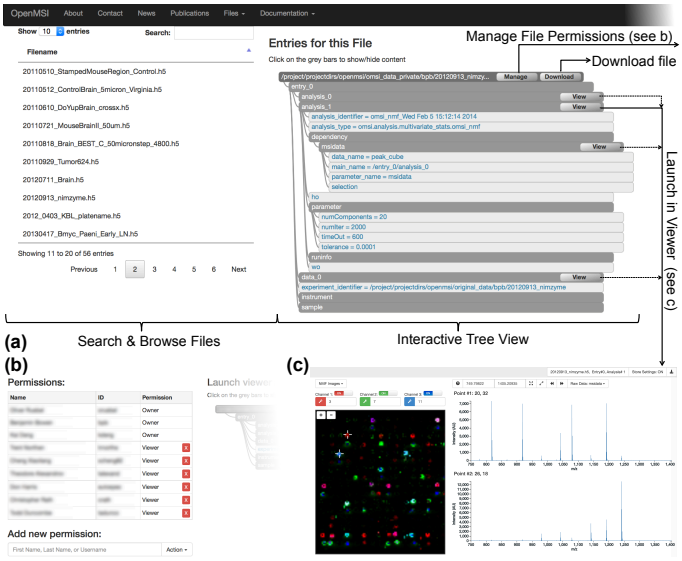

Fig. 8. (a) Online file browser with a searchable list of files (left) and an interactive tree-view showing the content of the current file (right). From here, users can download the file, manage access permissions for sharing with other users (see b), and launch the web-based viewer (see c) for the available MSI and analyses datasets.

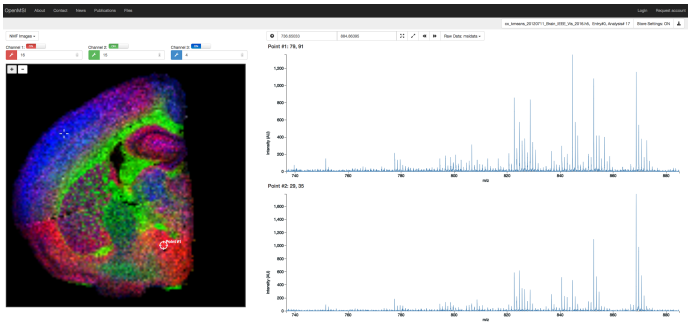

Fig. 9. Screenshot of the web-based data viewer. The image viewer (left) shows a three-channel image from a non-negative matrix factorization (NMF). Each channel represents a select NMF component image and is mapped to the red, green, or blue component of the RGB image, respectively. The user selected in the viewer two locations in distinct regions of the sample, marked by two white crosshair symbols. The spectrum viewer (right) shows the corresponding spectra from the raw MSI dataset for which the NMF was computed.

4.2.1 File Browser

To enable users to easily browse and explore their data files online, we developed a web-based file browser (Fig. 8). The file browser allows users to browse and search files via a list view (Fig. 8a, left) and to explore the content of HDF5 files via an interactive tree view (Fig. 8a, right). Small metadata, e.g., information about dependencies, analysis parameters, types, etc., is displayed directly as part of the tree view. From the tree view, users can download the HDF5 file and manage file access and sharing permissions (Fig. 8b). For any object that can be viewed via the web-based viewer (e.g., analyses, raw MSI, or dependencies), the user can directly launch the viewer from here (Fig. 8c).

4.2.2 Viewer

A central goal of the web-based viewer (Fig. 9, 8c) is to make MSI data easily accessible to novice users and users with no programming experience. The viewer consists of two intuitive views. The image slice viewer (left) allows users to explore the spatial composition of samples and to interactively select two locations of interest. The spectrum viewer (right) then displays the spectra associated with the selected image locations. The interactive zoom and pan locations are synchronized between the two spectrum plots, allowing users to easily explore and compare the makeup of the different loci. We here describe the extension of the viewer to support visualization of arbitrary derived analysis data and their dependencies.

The web-based viewer is based on the *qslice*, *qspectrum*, and *qmz* URL patterns and, hence, does not require any specific knowledge about different analyses or dataset names. The viewer and the user remain in this way isolated from any analysis and application specific implementation details and can flexibly explore images and spectra for raw data and all derived analyses and their dependencies.

We have extended the viewer to allow users to independently select the analysis *viewerOption* (described earlier in Sec. 4.1.2) for the image slice and spectrum viewer via easy-to-use drop-down menus. This enables the user to easily create mixed visualizations where the image and spectrum viewer show related data from different stages of an analysis workflow (e.g., in Fig. 9 raw mass spectra selected based on NMF images). Using the provenance graph to automatically construct the list of dependent view options makes it easy for users to locate and visualize dependent data and avoids critical errors. To allow users to easily explore a file without having to switch between the file browser and viewer page, the file browser (Fig. 8) can also be rendered in an overlay to the viewer. Suppl. 7.5 provides an overview of the graphical user interface of the web-based viewer.

The web-based viewer allows us to make results of highly complex analyses and workflows easily accessible to target users, further lowering the barriers for use, adoption, and deployment. Users can download URLs that capture the data and visualizations parameters, so that they can easily record and share interactive visualizations with collaborators and the science community at large.

4.2.3 Comparison to Existing MSI Analysis Tools

To date, MSI analysis tools—e.g., TissueView, BioMap, flexImaging, ImageQuest, MITICS among others—have been largely desktop-oriented. To the best of our knowledge, OmniSpect [39] is the only other publicly available web-based system for MSI analysis. OmniSpect is based on MATLAB and allows users to execute a static analysis workflow remotely and to afterwards download output files or view static plots. In contrast to these existing tools, OpenMSI in combination with BASTet is, to the best of our knowledge, the only MSI analysis system that supports: **i)** private and public sharing of large-scale MSI data and derived analysis data via the web, **ii)** remote analytics of raw and derived data products via a REST API, **iii)** interactive visualization and exploration of large MSI data and derived analyses via the web, **iv)** automatic, built-in provenance of analyses and complex workflows, **v)** easy reuse and reproduction of analyses, and **vi)** that enables users to utilize state-of-the-art high-performance computing systems for large-scale parallel analytics.

5 Evaluation

Next, we demonstrate the application of our system to study the metabolic makeup of a mouse brain sample (Sec. 5.1) and then discuss the broader impact on MSI (Sec. 5.2).

5.1 Application Case Study

Across applications of MSI, some of the most common, central questions are: **i)** which characteristic structures does a sample comprise, e.g., different tissues, **ii)** how do they differ, and **iii)** which ions are most important to the chemical composition of these structures. Here we demonstrate the application of our system to study these questions for an MSI dataset of the left coronal hemisphere of a mouse brain.

Analysis Workflow: Fig. 10 illustrates the analysis workflow. Using peak finding, we first identify the most intense ions and integrate the peaks. Using this approach, the dataset is reduced from a series of 80,399 images, each spanning a narrow range in m/z, to a set of 697 ion images each representing a singe main peak. Next, we normalize the spectra by dividing by the per-spectrum total ion count; also refered to as TIC normalization. Due to desorption and ionization

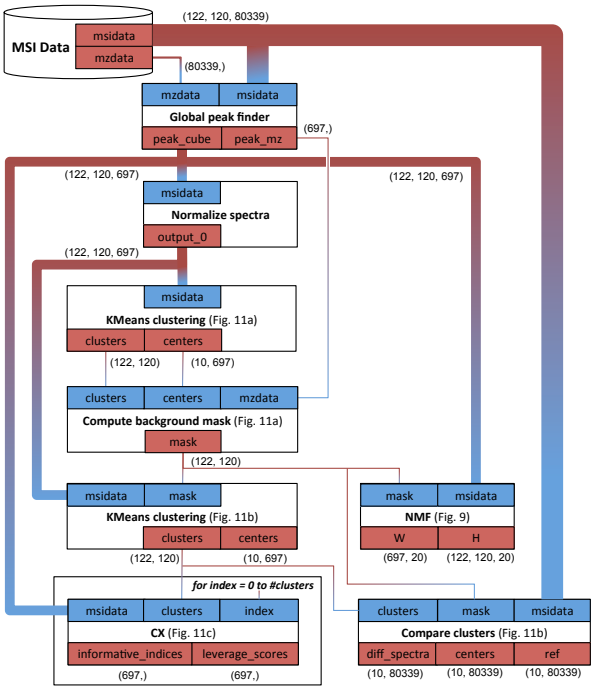

Fig. 10. Cluster analysis workflow showing **i)** the analysis tasks (white boxes), **ii)** dependent analysis task inputs (blue boxes), **iii)** analysis task outputs (red boxes), and **iv)** data flow connection lines with data flowing from red to blue and labels and line width indicating data size.

from heterogeneous surfaces, TIC normalization is commonly applied to help even out the ion-intensities across an image.

Clustering is widely used in MSI to find ions with similar spatial patterns and pixels with similar spectra. To be able to focus our analysis on the main sample, we next apply kmeans clustering and automatically classify the clusters into two groups (background vs. sample) based on whether a cluster is dominated by ions with an $m/z < 500Da$. Fig. 11a summarizes the results of the mask computation.

Using the resulting mask, we next classify the spectra of the sample into 10 distinct clusters using kmeans clustering. Finally we compute the average raw spectrum for each of the clusters and compare it with the remaining spectra. Fig. 11b shows the spectral centers computed from the full MSI data and the spatial map of the selected pixels for each cluster. We can see that the clusters are highly localized, selecting distinct tissues within the left coronal hemisphere of the mouse brain.

We next perform CX [55] matrix factorization independently for each of the clusters to identify which ions are most important to their chemical composition. The resulting leverage scores provide a ranking of all ions based on their contribution to the submatrix of spectra of the given region. Fig. 11c shows for each of the 10 clusters (rows) the top 10 important ions (columns) identified via CX matrix factorization, of which 20 are unique. Tab. 1 summarizes these ions and the clusters in which each was ranked within the top 10. 7 ions were ranked among the top ten in all or most clusters (Tab. 1, left column). Another 7 ions where ranked among the top ten in 3 to 5 clusters (Tab. 1, middle column). Finally, 6 ions were ranked among the top 10 in only 1 or 2 clusters (Tab. 1, right column). The identity and localization of these ions is comprehensively covered in a previous work by Lee et al.[18].

Implementation: We implemented the above workflow using BASTet via a combination of reusable workflow scripts and a series of Jupyter notebooks. Fig. 4 shows a code example of a similar but simplified workflow. The complete Jupyter notebooks are available in Suppl. 1. In this process, BASTet's workflow, provenance, and storage capabilities have shown to be complementary to the provenance capabilities and interactivity that Jupyter notebooks provide. Using BASTet allowed us to efficiently specify and validate the analysis

m/z	in top 10		m/z	in top 10		m/z	in top 10
852.7	all		806.71	[1,3,6,9,10]		800.66	[2,6]
844.7	all		389.1	[1,5,7,8]		840.66	[4,5]
868.7	all		409.11	[1,3,7,8]		403.26	[5,6]
828.69	all but [4]		469.28	[2,4,9,10]		332.41	[7,8]
824.68	all but [1,7]		804.69	[1,2,3,6]		845.69	[9]
822.7	all but [7,8]		798.74	[3,7,8]		853.7	[4]
497.32	all but [3,5]		869.69	[4,5,10]			

Table 1. Top ions identified via per-cluster CX matrix factorization and the clusters (row in Fig. 11b, c) in which they were ranked in the top 10.

workflow in an interactive and self-contained fashion while avoiding the need for user interaction as part of the workflow execution itself. The time-consuming execution of the analyses can in this way be easily delayed to the end of the notebook or even be offloaded to the command line for scheduled, offline execution (Suppl. 1.1). Being able to easily save and restore analyses to/from file then enabled us to elegantly separate the execution of the workflow from the visualization of analysis results (see Suppl. 1.4). BASTet's provenance and storage capabilities fill in this way a critical gap in the provenance capabilities of Jupyter notebooks, as it allows us to record the results and provenance of large-scale analyses not captured in the output of Jupyter code cells.

Impact: In practice, MSI data analyses are often based on lab-specific combinations of vendor-specific and custom tools and output formats, a process that hinders provenance, sharing, and reuse of analysis results. Beyond the advantages already seen during the implementation above, BASTet greatly simplified the development and subsequent integration and deployment of the individual analyses. BASTet also allowed us to easily share the workflow, analysis results, and visualizations with target users for validation, customization, and execution throughout the development process while avoiding the need for costly reexecution of analyses. Another main advantage of our approach has been the ability to reuse and extend workflows. E.g., several users requested the ability to use non-negative matrix factorization (NMF) to study the diversity of samples and to compute characteristic spectra. BASTet's provenance capabilities and command line analysis tools enabled us to easily expand the workflow (Fig. 10) after-the-fact by adding NMF based on the global peak finding results (Suppl. 1.2). Using the OpenMSI science gateway then greatly simplified the sharing of data and enabled users to easily access and explore analysis results interactively via the website and web API. For example, using the online viewer (see Fig. 9) we can study the spatial diversity of the brain sample via NMF images (left) while assessing metabolic differences between tissues using raw MSI spectra (right).

5.2 Broad Impact to MSI

One important measure for success is broader impact. Users have applied the visualization, analysis, and data management capabilities we have described here to study a broad range of biological phenomena. For example, Dalisay et al. [15] have successfully used OpenMSI for plant-biology studies, specifically to investigate the formation of dirigent protein-mediated lignan and cyanogenic glucoside in flax seeds. Silva and Northen [52] and Louie et al. [34] have used OpenMSI to study microbial chemical interactions and to deconstruct how cells interact to transform their small molecule environment. Raad et al. [43] are applying OpenMSI to perform large-scale assays of spotted samples (e.g, Fig. 8c) for bio-energy applications and have developed OMAAT, an advanced tool for analysis of arrayed experiments based on OpenMSI. Beyond these select published applications, users are also applying OpenMSI to study the heterogeneity and metabolism of tumors for applications in cancer research, and many others.

Users of OpenMSI have made a number of MSI datasets available to the public from a broad range of applications, e.g., images of a mouse brain [18], rat lung [22], mouse liver [32], cultures of microbial interactions [34] poplar leaf [13], potato, and flax pod [15]. The public sharing of data is an important resource for the scientific community at large. For example, other scientists have used this public data to study a broad range of topics, e.g., the application of **i)** multi-

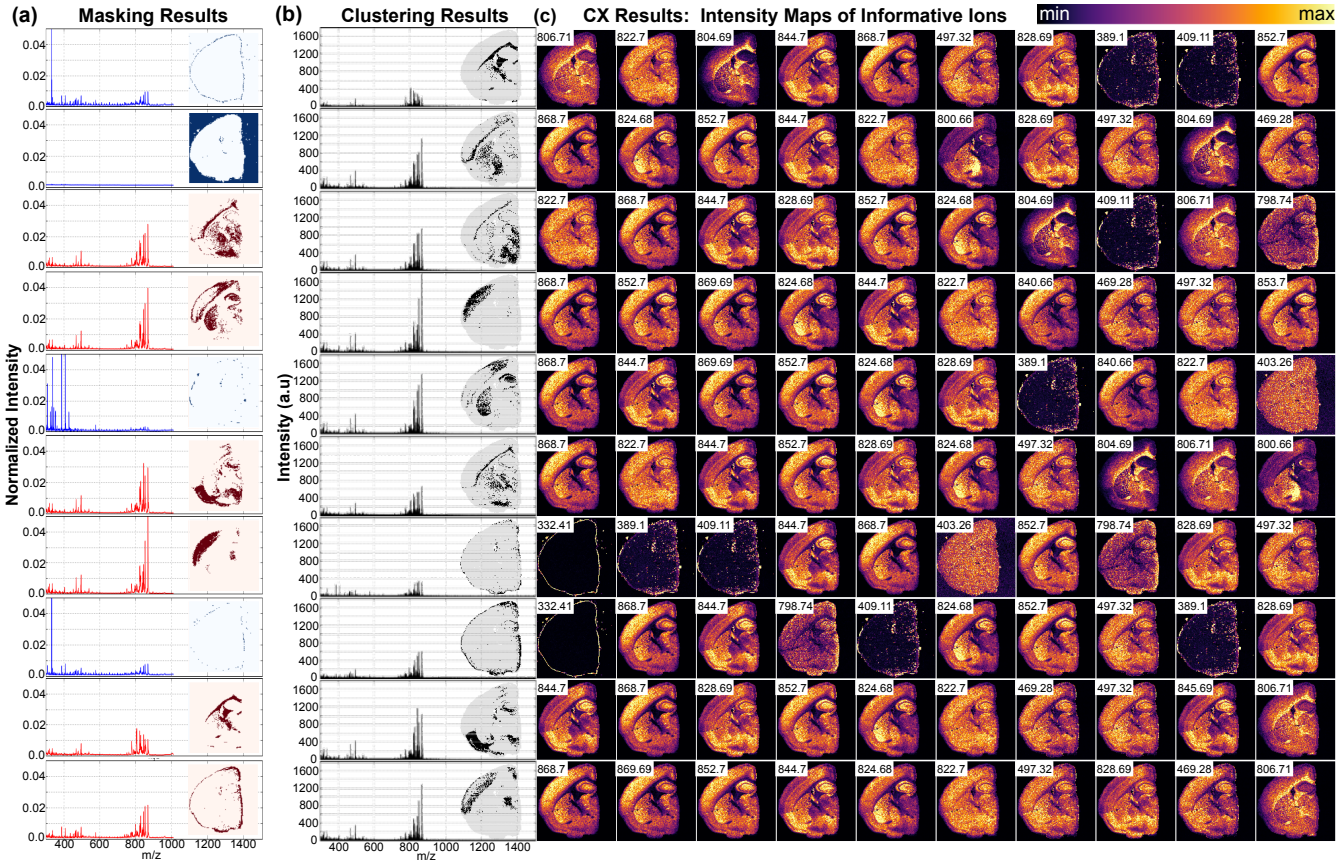

Fig. 11. Overview of results of the analysis workflow outlined in Fig. 10. **(a)** Visualization of the initial clustering results with curve plots of the mean spectra and spatial maps of the selected locations for each cluster. In contrast to the sample (red), the background clusters (blue) predominantly show high peaks for ions at $m/z < 500Da$. **(b)** Visualization of clustering results after masking with curve plots showing for each cluster the mean spectrum computed from the full MSI data and a map showing the locations selected by the cluster (black) and sample mask (gray). **(c)** Intensity maps of the ten most informative ions identified for each cluster via CX matrix factorization and ranked by their leverage scores.

variate curve resolution to MSI [29], **ii)** automatic data reorganization to accelerate data analysis [19], or **iii)** the application of CUR matrix decomposition to MSI to identify important ions and positions [55].

Finally, OpenMSI has been selected as one of the 100 most technologically significant new products of the year in Software/Services by the prestigious R&D100 awards in 2015 [27] and ImaBiotech, a CRO company developing and offering MSI services, applications, and software, licensed OpenMSI intellectual property in 2016 [25].

6 Conclusion

We have introduced BASTet, a novel framework for shareable and reproducible data analysis that supports standardized data and analysis interfaces, integrated data storage, data provenance, workflow management, and a broad set of integrated tools. Based on BASTet, we have described the extension of the OpenMSI mass spectrometry imaging science gateway to enable web-based sharing, reuse, analysis, and visualization of data analyses and derived data products.

BASTet and OpenMSI together provide critical means to share, apply, and reproduce MSI analyses. This research provides an important path for the MSI community towards enabling the broad application, validation, and use of novel MSI analysis methods and for building an ecosystem of standard and validated analysis methods, protocols, and workflows. Making sharebale and reproducible analysis methods and data easily accessible holds promise to enable the broad application of MSI across disciplines and ultimately to move MSI from the research bench to routine application in hospitals, drug development, and other commercial applications.

To enable sharing of visualization methods, our system currently supports the design of custom remote visualization and analysis tools based on OpenMSI's web API (e.g., Fischer et al. [23] or OMAAT [43]) as well as integration of interactive analyses as part of BASTet (Sec. 3.4). To further expand support for sharing of custom visualizations, we plan in the future to: **i)** expand the integration of the Jupyter web service [54] with OpenMSI and **ii)** extend BASTet's analysis API to support specification and sharing of declarative, analysis-specific visualizations [33], e.g., defined via Vega [46] or Bokeh [5], in combination with corresponding extensions of OpenMSI's web API and interface to facilitate access and display. We also plan to further expand the workflow management capabilities of BASTet by integrating complimentary workflow and provenance tools, e.g., VisTrails [4] or Tigres [3]. Finally, we plan to further generalize our system and demonstrate its application to other application domains, e.g., atomic force microscopy. For example, Suppl. 5 demonstrates the application of BASTet and OpenMSI to neuroscience electrophysiology data.

Acknowledgments

This work was supported by and used resources of Berkeley Lab's Laboratory Directed Research and Development (LDRD) support; the National Energy Research Scientific Computing Center (NERSC) supported by the Office of Science of the U.S. Department of Energy; and the Low-Dose Radiation Research of the Office of Science, Office of Biological and Environmental Research, of the U. S. Department of Energy under Contract No. DE-AC02-05CH11231. We thank S. Cholia and the Scientific Data Services team at NERSC for their ongoing efforts and support to help deliver scientific data and high-performance computing to science communities. We thank the Northen Lab (LBNL) and all OpenMSI users for their help, support, feedback, and contributions of MSI data.

REFERENCES

[1] J. Ahrens, B. Geveci, and C. Law. ParaView: An End-User Tool for Large-Data Visualization. In C. D. Hansen and C. R. Johnson, editors, *Visualization handbook*, pages 717–731. Elsevier, 2005.

[2] S. Anders. Visualization of genomic data with the Hilbert curve. *Bioinformatics*, 25(10):1231–1235, 2009.

[3] J. R. Balderrama, M. Simonin, L. Ramakrishnan, V. Hendrix, C. Morin, D. Agarwal, and C. Tedeschi. Combining Workflow Templates with a Shared Space-Based Execution Model. In *9th Workshop on Workflows in Support of Large-Scale Science (WORKS)*, pages 50–58, Nov 2014.

[4] L. Bavoil, S. P. Callahan, P. J. Crossno, J. Freire, C. E. Scheidegger, C. T. Silva, and H. T. Vo. VisTrails: Enabling interactive multiple-view visualizations. In *IEEE Visualization 2005*, pages 135–142. IEEE, 2005.

[5] Bokeh Development Team. *Bokeh: Python library for interactive visualization*, 2014.

[6] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[7] B. P. Bowen and O. Rübel. System and method of managing large data files, June 12 2014. US Patent App. 14/091,986.

[8] P. G. Brown. Overview of SciDB: large scale array storage, processing and analysis. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 963–968. ACM, 2010.

[9] S. P. Callahan, J. Freire, C. E. Scheidegger, C. T. Silva, and H. T. Vo. Towards provenance-enabling ParaView. In *Provenance and Annotation of Data and Processes*, pages 120–127. Springer, 2008.

[10] S. Cholia, D. Skinner, and J. Boverhof. NEWT: A RESTful service for building High Performance Computing web applications. In *Gateway Computing Environments Workshop (GCE)*, pages 1–11. IEEE, 2010.

[11] K. Chughtai and R. M. A. Heeren. Mass spectrometric imaging for biomedical tissue analysis. *Chemical Reviews*, 110(5):3237–3277, 2010.

[12] J. Clarke and E. Mark. Enhancements to the eXtensible Data Model and Format (XDMF). In *DoD High Performance Computing Modernization Program Users Group Conference, 2007*, pages 322–327, June 2007.

[13] M. A. Costa, J. V. Marques, D. S. Dalisay, B. Herman, D. L. Bedgar, L. B. Davin, and N. G. Lewis. Transgenic hybrid poplar for sustainable and scalable production of the commodity/specialty chemical, 2-phenylethanol. *PloS one*, 8(12):e83169, 2013.

[14] D. Williams et al. Ultrascale Visualization of Climate Data. *IEEE Computer*, 46(9):68–76, Sept. 2013.

[15] D. S. Dalisay, K. W. Kim, C. Lee, H. Yang, O. Rübel, B. P. Bowen, L. B. Davin, and N. G. Lewis. Dirigent Protein-Mediated Lignan and Cyanogenic Glucoside Formation in Flax Seed: Integrated Omics and MALDI Mass Spectrometry Imaging. *Journal of Natural Products*, 78(6):1231–1242, 2015.

[16] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, et al. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.

[17] Django Software Foundation. DJANGO. [ONLINE] `https://www.djangoproject.com/`.

[18] V. P. Do Yup Lee, B. Bowen, K. Louie, C. Canaria, C. T. McMurray, and T. Northen. Resolving brain regions using nanostructure initiator mass spectrometry imaging. *Integrative biology: quantitative biosciences from nano to macro*, 4(6):693, 2012.

[19] B. Dong, S. Byna, and K. Wu. Expediting scientific data analysis with reorganization of data. In *IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–8. IEEE, 2013.

[20] H. C. et al. VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *Proceedings of SciDAC 2011*, 2011.

[21] L. M. et al. mzMLa community standard for mass spectrometry data. *Molecular & Cellular Proteomics*, 10(1):R110–000133, 2011.

[22] T. E. Fehniger, Á. Végvári, M. Rezeli, K. Prikk, P. Ross, M. Dahlbck, G. Edula, R. Sepper, and G. Marko-Varga. Direct demonstration of tissue uptake of an inhaled drug: proof-of-principle study using matrix-assisted laser desorption ionization mass spectrometry imaging. *Analytical chemistry*, 83(21):8329–8336, 2011.

[23] C. R. Fischer, O. Rübel, and B. P. Bowen. An accessible, scalable ecosystem for enabling and sharing diverse mass spectrometry imaging analyses. *Archives of Biochemistry and Biophysics*, 589:18 – 26, January 2016. Applications of Metabolomics.

[24] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[25] E. Fontaine. ImaBiotech signs license agreement Lawrence Berkeley National Laboratory. *ImaBiotech*, Apr 2016. [ONLINE] `https://www.imabiotech.com/IMG/pdf/-55.pdf`.

[26] I. Foster. Globus Online: Accelerating and Democratizing Science through Cloud-Based Services. *IEEE Internet Computing*, 15(3):70–73, May 2011.

[27] L. Hock. 2015 R&D 100 Award Winners. *R&D Magazine*, Nov. 2015. [ONLINE] `http://tinyurl.com/zcmuk7w`.

[28] A. Jain, S. P. Ong, W. Chen, B. Medasani, X. Qu, M. Kocher, M. Brafman, G. Petretto, G.-M. Rignanese, G. Hautier, D. Gunter, and K. A. Persson. FireWorks: a dynamic workflow system designed for high-throughput applications. *Concurrency and Computation: Practice and Experience*, pages n/a–n/a, 2015.

[29] J. Jaumot and R. Tauler. Potential use of multivariate curve resolution for the analysis of mass spectrometry images. *Analyst*, 140:837–846, 2015.

[30] H. J. Johnson, M. M. McCormick, and L. Ibanez. *The ITK Software Guide Book 1: Introduction and Development Guidelines - Volume 1*. Kitware, Inc., USA, 2015.

[31] P. Klosowski, M. Koennecke, J. Tischler, and R. Osborn. NeXus: A common format for the exchange of neutron and synchroton data. *Physica B: Condensed Matter*, 241:151–153, 1997.

[32] M. Lagarrigue, R. Lavigne, E. Tabet, V. Genet, J.-P. Thom, K. Rondel, B. Guvel, L. Multigner, M. Samson, and C. Pineau. Localization and in Situ Absolute Quantification of Chlordecone in the Mouse Liver by MALDI Imaging. *Analytical Chemistry*, 86(12):5775–5783, 2014.

[33] D. R. Lipşa, R. S. Laramee, S. J. Cox, J. C. Roberts, R. Walker, M. A. Borkin, and H. Pfister. Visualization for the physical sciences. In *Computer graphics forum*, volume 31, pages 2317–2347. Wiley Online Library, 2012.

[34] K. B. Louie, B. P. Bowen, X. Cheng, J. E. Berleman, R. Chakraborty, A. Deutschbauer, A. Arkin, and T. R. Northen. "Replica-Extraction-Transfer" Nanostructure-Initiator Mass Spectrometry Imaging of Acoustically Printed Bacteria. *Analytical Chemistry*, 85(22):10856–10862, 2013.

[35] F. R. Maia. The coherent X-ray imaging data bank. *Nature methods*, 9(9):854–855, 2012.

[36] L. A. McDonnell and R. M. Heeren. Imaging mass spectrometry. *Mass Spectrometry Reviews*, 26(4):606–643, 2007.

[37] MongoDB Inc. MongoDB. [ONLINE] `www.mongodb.org`.

[38] K. Moreland, C. Sewell, W. Usher, L. t. Lo, J. Meredith, D. Pugmire, J. Kress, H. Schroots, K. L. Ma, H. Childs, M. Larsen, C. M. Chen, R. Maynard, and B. Geveci. Vtk-m: Accelerating the visualization toolkit for massively threaded architectures. *IEEE Computer Graphics and Applications*, 36(3):48–58, May 2016.

[39] R. M. Parry, A. S. Galhena, C. M. Gamage, R. V. Bennett, M. D. Wang, and F. M. Fernández. OmniSpect: An Open MATLAB-Based Tool for Visualization and Analysis of Matrix-Assisted Laser Desorption/Ionization and Desorption Electrospray Ionization Mass Spectrometry Images.

[40] F. Pedregosa and P. Gervais. memory_profiler. [ONLINE] `https://pypi.python.org/pypi/memory_profiler`.

[41] PiCloud, Inc. Cloud Pickle. [ONLINE] `https://pypi.python.org/pypi/cloud`.

[42] PostgreSQL Global Development Group. PostgresSQL. [ONLINE] `http://www.postgresql.org/`.

[43] M. Raad, T. de Rond, O. Rübel, J. Keasling, T. Northen, and B. Bowen. OpenMSI Arrayed Analysis Toolkit: Analyzing spatially defined samples using mass spectrometry imaging. *Analytical Chemistry*. (submitted). [ONLINE] `https://github.com/biorack/omaat`,.

[44] R. Rew and G. Davis. NetCDF: an interface for scientific data access. *Computer Graphics and Applications, IEEE*, 10(4):76–82, July 1990.

[45] O. Rübel, A. Greiner, S. Cholia, K. Louie, E. W. Bethel, T. R. Northen, and B. P. Bowen. OpenMSI: A High-Performance Web-Based Platform for Mass Spectrometry Imaging. *Analytical Chemistry*, 85(21):10354–10361, 2013.

[46] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer. Reactive Vega: A Streaming Dataflow Architecture for Declarative Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):659–668, Jan. 2016.

[47] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, et al. Fiji: an open-source platform for biological-image analysis. *Nature methods*, 9(7):676–682, 2012.

[48] C. A. Schneider, W. S. Rasband, K. W. Eliceiri, et al. NIH Image to ImageJ: 25 years of image analysis. *Nature methods*, 9(7):671–675, 2012.

[49] T. Schramm, A. Hester, I. Klinkert, J.-P. Both, R. M. Heeren, A. Brunelle, O. Laprvote, N. Desbenoit, M.-F. Robbe, M. Stoeckli, B. Spengler, and A. Rmpp. imzML A common data format for the flexible exchange and processing of mass spectrometry imaging data. *Journal of Proteomics*, 75(16):5106 – 5110, 2012.

[50] W. Schroeder, K. Martin, and B. Lorensen. *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Kitware, 4th edition, Dec 2006.

[51] S. Shasharina, J. R. Cary, S. Veitzer, P. Hamill, S. Kruger, M. Durant, and D. A. Alexander. VizSchema–Visualization Interface for Scientific Data. In *IADIS International Conference, Computer Graphics, Visualization, Computer Vision and Image Processing*, page 49, 2009.

[52] L. P. Silva and T. R. Northen. Exometabolomics and MSI: deconstructing how cells interact to transform their small molecule environment. *Current Opinion in Biotechnology*, 34:209 – 216, 2015. Systems biology Nanobiotechnology.

[53] The HDF Group. Hierarchical Data Format, version 5, 1997-2015. [ONLINE] `http://www.hdfgroup.org/HDF5/`.

[54] R. Thomas, S. Canon, S. Cholia, L. Gerhardt, and E. Racah. Toward Interactive Supercomputing at NERSC with Jupyter. In *Cray User Group (CUG) Conference Proceedings*. Cray User Group (CUG), May 2017.

[55] J. Yang, O. Rübel, Prabhat, M. W. Mahoney, and B. P. Bowen. Identifying Important Ions and Positions in Mass Spectrometry Imaging Data Using CUR Matrix Decompositions. *Analytical Chemistry*, 87(9):4658–4666, 2015.