The Journal of Open Source Software

ViMMS 2.0: A framework to develop, test and optimise fragmentation strategies in LC-MS metabolomics

Joe Wandy*[1], Vinny Davies[2], Ross McBride[3], Stefan Weidt[1], Simon Rogers[3], and Rónán Daly[1]

1 Glasgow Polyomics, University of Glasgow, United Kingdom **2** School of Mathematics and Statistics, University of Glasgow, United Kingdom **3** School of Computing Science, University of Glasgow, United Kingdom

DOI: 10.21105/joss.03990

Software

- Review
- Repository
- Archive

Editor: Charlotte Soneson

Reviewers:

- @jspaezp
- @MKoesters

Submitted: 25 November 2021
Published: 13 December 2021

Summary

The choice of fragmentation strategies used during mass-spectrometry-based data acquisition directly affects the quality and coverage of subsequent structural identification – a crucial step in untargeted metabolomics data analysis. However, developing novel fragmentation strategies is challenging due to the high experimental cost of running an actual mass spectrometry instrument and the lack of a programmable simulation environment to support their development. ViMMS 2.0 is a software framework that can be used to develop new fragmentation strategies in metabolomics completely *in-silico* as well as on mass spectrometry instruments. The framework allows users to generate chemical objects (produced synthetically or extracted from existing mzML files) and simulate a tandem mass spectrometry process, where different fragmentation strategies can be rapidly implemented, tested and evaluated. In this paper, we present ViMMS 2.0, highlighting the software design choices of the framework and illustrate with an example how a new fragmentation strategy could be implemented in ViMMS 2.0.

Statement of need

Metabolomics is the study of small molecules that participate in important cellular processes of an organism. Being the closest to the phenotype, changes to metabolite levels are often expressed as a response to genetic or environmental changes (Guijas et al., 2018). Liquid chromatography (LC) coupled to mass spectrometry (MS) is commonly used to identify small molecules in untargeted experiments, where the identities of molecules of interests are not known in advance (Smith et al., 2014). In this setup, molecules elute through the LC column at different retention times (RTs) before being presented to the MS instrument. In tandem mass spectrometry, selected ions produced from survey scans (MS1; typically measurements of the intact ions) are isolated for fragmentation in a second MS instrument (MS2), resulting in a spectral fingerprint for each isolated ion that is often used for structural identification.

Typically the raw LC-MS/MS measurements are processed in a data pre-processing pipeline to produce a list of chromatographic MS1 peaks characterised by their m/z, RT and intensity values. During identification, molecular annotations are assigned to MS1 peaks through matching with internal standard compounds (having known m/z and RT values) or by searching spectral databases with the MS2 spectra associated with MS1 peaks. An important factor that determines how many molecules can be annotated with spectral databases using fragmentation data is the quality of the MS2 spectra acquired, and the coverage (for how many

*corresponding author

of the MS1 peaks were MS2 spectra collected). Good MS2 fragmentation strategies aim to produce spectra for as many unknown ions in the sample as possible, but also produce high quality spectra which can be reliably evaluated.

A common challenge faced by computational researchers with an interest in improving fragmentation strategies in tandem-mass-spectrometry-based metabolomics is the lack of access to and the high cost of running MS instruments. This issue is particularly relevant as developing and optimising novel fragmentation strategies tends to be conducted iteratively, requiring many measurements to be run to optimise the strategy until the desired performance level is reached. To lower this barrier, we present **Vi**rtual **M**etabolomics **M**ass **S**pectrometer (ViMMS) 2.0, a programmable and modular framework that simulates the chemical generation process and the execution of fragmentation strategies in LC-MS/MS-based metabolomics. A notable feature of the ViMMS framework is the ability to design strategies that can respond, in real time, to the data being produced.

Related works

Existing mass spectrometry simulators are ill-fitted to support the rapid development of fragmentation strategies that respond to incoming scans in real time. This is primarily due to the limited ways users can incorporate new strategies within existing simulator codebases. Currently available simulators such as as Mspire-Simulator (Noyce et al., 2013), JAMSS (Smith & Prince, 2015), OpenMS-Simulator (Wang et al., 2015), MSAcquisitionSimulator (Goldfarb et al., 2016) and SMITER (Kösters et al., 2021) exist as stand-alone programs or GUI applications, and are not easily scriptable or programmable.

Additionally the above-highlighted simulators have been developed to simulate the generation of proteomics data. In principle they could be extended to support metabolomics data, but this is not a trivial change. Our work in ViMMS 1.0 (Wandy et al., 2019) was the first simulator that allowed for a metabolomics-based simulation environment. However ViMMS 1.0 suffered from several weaknesses: its codebase was monolithic, and the tight coupling between modules made it difficult to instantiate input from different sources or to introduce different extensions to the base functionalities, including adding new fragmentation strategies. The focus of ViMMS 1.0 was on simulating a complete tandem mass spectrometry run in metabolomics, rather than enabling the development of new strategies.

The ViMMS 2.0 Framework

In ViMMS 2.0 we significantly improved the simulator framework architecture with the goal of supporting fragmentation strategy development and validation. An overview of the overall architecture is given in Figure 1. The simulator framework consists of two core functionalities: the generation of chemicals from multiple sources, and the execution of fragmentation strategies, implemented as controller classes. The improved modularity in ViMMS 2.0 allows many aspects of the framework to be swapped out with alternative implementations, including classes that generate various aspects of chemicals (e.g. measured m/z, RT, intensity) for simulation (Figure 1A), mass spectrometry simulation (Figure 1B), controllers that implement different fragmentation strategies (Figure 1C), as well as the environmental context to run them all (Figure 1D). Different evaluation methods are also provided to compare different controllers' performance.

The Journal of Open Source Software

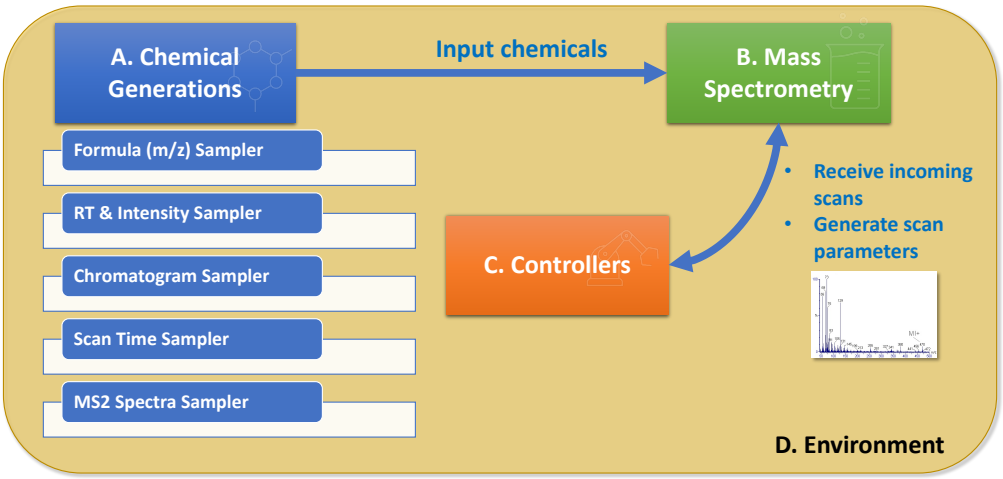

Figure 1: Overall ViMMS 2.0 System Architecture.

Generating Input Chemicals

Chemicals are input objects to the simulation process in ViMMS 2.0, and can be generated in different ways: either in a purely synthetic manner (e.g. sampling their properties from databases or fixed statistical distributions) or by construction from existing data (mzML files) via the extraction of RoI traces. The framework allows users to plug in modular classes that specify parameters of chemicals, such as the distribution of their m/z values, RT, intensities, chromatographic shapes and associated MS2 spectra (Figure 1A).

Chemicals can be generated in a single-sample or multi-sample settings. When generating multi-sample data, ViMMS allows users to specify how chemicals vary across samples. Given a list of base chemicals (chemicals that are shared) across samples, users can indicate in what proportion of extracts chemicals should appear (dropout rate) or how chemical intensities should vary across a case-control experiment.

Executing Fragmentation Strategies

Once chemical objects have been prepared (whether for single- or multi-sample settings), different fragmentation strategies can be run. Fragmentation strategies are implemented as controllers that extend from the base `Controller` class. Controllers are executed in the context of an environment bringing together input chemicals, mass spectrometer and controllers as a single context (Figure 1D). Note that the modularity of the mass spectrometry and environment classes means it is possible to swap purely simulated MS and environment implementation with alternatives that control an actual MS instrument while other aspects remain unchanged. In other work, we demonstrated the practicality of this idea by building alternative implementations of these classes that use the Thermo Fisher IAPI ("Thermo Fisher Scientific", n.d.) for bridging, making it possible for fragmentation strategies to be executed in simulation as well as unchanged on Thermo Tribrid Fusion instruments (Davies et al., 2021).

Implementing a New Controller

To illustrate how new strategies could be built on top of ViMMS 2.0, an example is given here of a Top-N controller that selects the N most intense precursor ions for fragmentation. This strategy is common in real-world data-dependent acquisition (DDA) experiments and is included to demonstrate how straightforward its implementation is within the framework.

In this implementation, the controller `SimpleTopNController` extends from a base `Controller` that provides base methods to handle various scan interactions with the mass spectrometer. This implementation overrides the `_process_scan` method, which determines how precursor ions in an incoming MS1 scan are prioritised for fragmentation. Information on what the mass spectrometer should do next is stored as a list of `ScanParameters`, which is returned from the controller to the mass spectrometer object by `_process_scan`.

Scan parameters control, for example, whether a controller flexibly targets a single precursor ion for fragmentation, in what is commonly known as data-dependent acquisition (DDA), or whether it instead covers a range of m/z values, potentially fragmenting many precursor ions (data-independent acquisition; DIA). Although not shown in this example, other methods in the parent `Controller` could also be overridden for different purposes, such as responding when an acquisition has been started or stopped.

```
import numpy as np
from vimms.Controller.base import Controller


class SimpleTopNController(Controller):

    def _process_scan(self, scan):
        new_tasks = []

        # If the controller has received an MS1 scan to process
        if self.scan_to_process is not None:

            # Extract m/z and intensity values in this MS1 scan
            mzs = self.scan_to_process.mzs
            intensities = self.scan_to_process.intensities

            # Select only the Top-N precursors, sorted by intensities descending
            idx = np.argsort(intensities)[::-1]
            idx = idx[0:self.N]

            # Loop over the Top-N precursors and target them for fragmentation
            for i in idx:
                mz, intensity = mzs[i], intensities[i]

                # Schedule a new MS2 scan targeting the selected precursor ion
                dda_scan_params = self.get_ms2_scan_params(mz, intensity, ...)
                new_tasks.append(dda_scan_params)
                self.current_task_id += 1

            # Schedule the next survey MS1 scan after doing N MS2 scans
            ms1_scan_params = self.get_ms1_scan_params()
            new_tasks.append(ms1_scan_params)
            self.current_task_id += 1

            # Set this MS1 scan as has been processed
            # and indicate what is the next MS1 scan id to process
            self.scan_to_process = None
            self.next_processed_scan_id = self.current_task_id

        # Return all the scheduled tasks to be executed by the mass spec
        return new_tasks
```

The simple Top-N scheme above could be enhanced to incorporate dynamic exclusion windows to prevent the same precursors from being fragmented repeatedly, or to incorporate different schemes prioritising which of the precursor ions to fragment. We have included a more complete Top-N strategy as the baseline controller in ViMMS 2.0 against which other strategies can be benchmarked. Two enhanced DDA controllers (named **SmartROI** and **WeightedDEW**, outlined by Davies et al. (2021)) are also provided that demonstrate how novel fragmentation strategies could be rapidly implemented and validated in ViMMS 2.0. SmartROI accomplishes this by tracking regions-of-interest in real-time and targeting those for fragmentation, while WeightedDEW implements a weighted dynamic exclusion scheme to decide prioritisation of precursor ions for fragmentation. Code is provided to compute various evaluation metrics, such as coverage and intensities of fragmented precursors. This allows users to benchmark different controller implementations comparatively.

Software requirements

ViMMS 2.0 is distributed as a Python package that can be easily installed using pip. We require Python 3.0 or higher to run the framework. It depends on common packages such as numpy, scipy and pandas. Automated unit tests are available in Python, as well as continuous integration that build and run those unit tests in our code repository. Our codebase is stored in Github and we welcome contributions from researchers with interest in developing novel fragmentation strategies in both data-dependent and data-independent acquisitions.

Conclusion

in this paper, we have introduced ViMMS 2.0, an extension of the simulator framework in ViMMS 1.0. ViMMS 2.0 is modular, extensible and can be used in Python to simulate fragmentation strategies for untargeted metabolomics studies. In other work (Davies et al., 2021), the utility of ViMMS 2.0 has been validated through additional bridging code that allows simulated controllers to run unchanged on both the simulator and actual mass spectrometers. It is our hope that the work outlined here will be used to advance the development of novel fragmentation strategies in untargeted metabolomics.

Acknowledgements

Joe Wandy, Vinny Davies, Stefan Weidt, Simon Rogers and Rónán Daly acknowledge EPSRC project EP/R018634/1 on 'Closed-loop data science for complex, computationally and data-intensive analytics.'

References

Davies, V., Wandy, J., Weidt, S., Van Der Hooft, J. J., Miller, A., Daly, R., & Rogers, S. (2021). Rapid development of improved data-dependent acquisition strategies. *Analytical Chemistry*, *93*(14), 5676–5683. https://doi.org/10.1021/acs.analchem.0c03895

Goldfarb, D., Wang, W., & Major, M. B. (2016). MSAcquisitionSimulator: Data-dependent acquisition simulator for LC-MS shotgun proteomics. *Bioinformatics*, *32*(8), 1269–1271. https://doi.org/10.1093/bioinformatics/btv745

Guijas, C., Montenegro-Burke, J. R., Warth, B., Spilker, M. E., & Siuzdak, G. (2018). Metabolomics activity screening for identifying metabolites that modulate phenotype. *Nature Biotechnology*, *36*(4), 316–320. https://doi.org/10.1038/nbt.4101

Kösters, M., Leufken, J., & Leidel, S. A. (2021). SMITER—a python library for the simulation of LC-MS/MS experiments. *Genes*, *12*(3), 396. https://doi.org/10.3390/genes12030396

Noyce, A. B., Smith, R., Dalgleish, J., Taylor, R. M., Erb, K., Okuda, N., & Prince, J. T. (2013). Mspire-simulator: LC-MS shotgun proteomic simulator for creating realistic gold standard data. *Journal of Proteome Research*, *12*(12), 5742–5749. https://doi.org/10.1021/pr400727e

Smith, R., Mathis, A. D., Ventura, D., & Prince, J. T. (2014). Proteomics, lipidomics, metabolomics: A mass spectrometry tutorial from a computer scientist's point of view. *BMC Bioinformatics*, *15*(7), 1–14. https://doi.org/10.1186/1471-2105-15-s7-s9

Smith, R., & Prince, J. T. (2015). JAMSS: Proteomics mass spectrometry simulation in java. *Bioinformatics*, *31*(5), 791–793. https://doi.org/10.1093/bioinformatics/btu729

"Thermo Fisher Scientific". (n.d.). *Thermo fisher application programming interface*. https://github.com/thermofisherlsms/iapi.

Wandy, J., Davies, V., J J van der Hooft, J., Weidt, S., Daly, R., & Rogers, S. (2019). In silico optimization of mass spectrometry fragmentation strategies in metabolomics. *Metabolites*, *9*(10), 219. https://doi.org/10.1101/744227

Wang, Y., Yang, F., Wu, P., Bu, D., & Sun, S. (2015). OpenMS-simulator: An open-source software for theoretical tandem mass spectrum prediction. *BMC Bioinformatics*, *16*(1), 1–6. https://doi.org/10.1186/s12859-015-0540-1