Bioinformatics, 31(5), 2015, 791–793
doi: 10.1093/bioinformatics/btu729
Advance Access Publication Date: 3 November 2014
Applications Note

OXFORD

Data and text mining

JAMSS: proteomics mass spectrometry simulation in Java

Rob Smith[1,*] and John T. Prince[2]

[1]Department of Computer Science, University of Montana, Missoula, MT 59812, USA and [2]Department of Chemistry, Brigham Young University, Provo, UT 84602, USA

*To whom correspondence should be addressed.

Associate Editor: Jonathan Wren

Received on January 22, 2014; revised on October 22, 2014; accepted on October 29, 2014

Abstract

Summary: Countless proteomics data processing algorithms have been proposed, yet few have been critically evaluated due to lack of labeled data (data with known identities and quantities). Although labeling techniques exist, they are limited in terms of confidence and accuracy. *In silico* simulators have recently been used to create complex data with known identities and quantities. We propose Java Mass Spectrometry Simulator (JAMSS): a fast, self-contained *in silico* simulator capable of generating simulated MS and LC-MS runs while providing meta information on the provenance of each generated signal. JAMSS improves upon previous *in silico* simulators in terms of its ease to install, minimal parameters, graphical user interface, multithreading capability, retention time shift model and reproducibility.

Availability and implementation: The simulator creates mzML 1.1.0. It is open source software licensed under the GPLv3. The software and source are available at https://github.com/optimusmoose/JAMSS.

Contact: 2robsmith@gmail.com

1 Introduction

Proteomics studies require the prediction of the quantity and identity of proteins in sample. The accuracy of the determination relies wholly on the accuracy of the data processing pipeline modules that systematically extract and process the components of the sample output file (Smith *et al.*, 2013a). Despite the criticality of data processing accuracy, few published algorithms have quantitative comparisons against other algorithms using labeled data—data where the correct protein quantity and identity are known (Smith *et al.*, 2013c).

Common strategies for labeling data are limited in terms of confidence and accuracy. For example, MS/MS identifications are biased towards the approximately 16% most intense signals, and have an approximately 50% false-positive rate (Michalski *et al.*, 2011), leading to evaluative results that are not representative of the dataset, particularly among the more biologically significant but less-abundant peptides. Hand-labeled datasets exist (Conley *et al.*, 2014; Smith *et al.*, 2014), but due to the complexity of labeling by hand usually consist of small segments of data within an intensity threshold. Using existing tools, hand labeling consists of many subjective decisions, and creating a set of replicates would take years.

Construction of *in silico* datasets is an attractive alternative to labeling, as these datasets automatically include labels. *In silico* simulation consists of emulating the physiochemical processes involved in the mass spectrometry analysis of a sample in order to produce an mzML (or equivalent) output file similar to what would be generated in a real run but without any material or instrument time cost. Although more research is needed before an exact replicate of a real sample run can be simulated, the overall characteristics of the output data in terms of density, noise, signal shape, etc., are similar enough to be valuable as labeled data for LC-MS data processing algorithmic evaluation.

LC-MS simulation is still in its infancy. LC-MSsim, an incorporated module of OpenMS, was the first simulator to produce full-featured MS simulated data (Schulz-Trieglaff *et al.*, 2008). It has since been replaced by MSSimulator, featuring more realistic isotope

trace variance (in both intensity and *m/z*) and MS/MS simulation (Bielow *et al.*, 2011). Most recently, Mspire-simulator, a standalone simulator in the Ruby programming language, provided automatic charge modeling, realistic hourglass-shaped isotope traces (increased variance at lower intensities), direct control over post-translational modifications (PTMs), and the ability to extract simulation parameters from existing mzML files using machine learning (Noyce *et al.*, 2013).

There remains much to be done in MS simulation. Existing simulators do not provide the molecular provenance of output signals necessary for generating ground truth data for quantitative MS data processing algorithm evaluation. They have involved installation processes, requiring installation of their parent libraries and a sometimes onerous degree of dependency management. They are also both command-line programs with very little documentation. Neither program is multithreaded. Both programs feature many parameters, some of which significantly alter the simulation outcome in unclear ways. Although Mspire-simulator produces run-to-run variation (unlike MSSimulator), it does not vary the RT of eluents across runs and cannot produce a clone of a previous run when fed the same input and parameters. Although Mspire-simulator's isotope trace generation features more realistic variance in *m/z* and intensity than MSSimulator, it is many times slower and has no bound on RAM requirements. Both programs seem limited in regards to PTMs: Mspire-simulator renders all PTMs as static (even variable ones) and MSSimulator seems to as well. MSSimulator produces the same isotope trace shape (scaled for intensity) for every peptide, meaning its utility for generating datasets for evaluating data processing algorithms is limited.

This article describes the Java Mass Spectrometry Simulator (JAMSS), a novel simulator designed to address each of the above-mentioned drawbacks of current simulation software.

2 Methods

JAMSS takes any protein.fasta file as input. Optionally, users can specify the quantity of each protein as a percentage of the total sample content (see program documentation in README file). The GUI provides several clear options to modify the run. For example, the user can specify how many cores to use if using a multicore machine. They can select one of 16 digestion enzymes. They can select how many scans per second, how many missed cleavages to allow, how many MS2s per scan to generate, how many noise points to include and at what intensity range and the pH of the sample. They can control the resolution of the simulation through a merge parameter. In addition, the GUI can be set for a one-dimensional (non-chromatographic) simulation, which is useful in modeling direct injection experiments. There are also settings for PTMs. The program includes options for carbamidomethylation, pyroglutamation, phosphorylation and methionine oxidation. Although these options do not include all possible PTMs, limiting them by explicit mention allows for treating variable PTMs as they should be treated: that is, the combinatoric possibilities of all selected PTMs are calculated, and the total quantity of each protein is split according to the percentage of the proteins each PTM combination will affect.

After reading the .fasta file, the simulator performs an *in silico* digestion of each protein. At this point, the work is divided among the number of user-selected available CPU cores. For each peptide, atom counts are calculated from which the isotopic envelopes and charge are calculated. If PTMs are selected, this process is executed for each PTM applicable to the peptide. From there, the amino acid

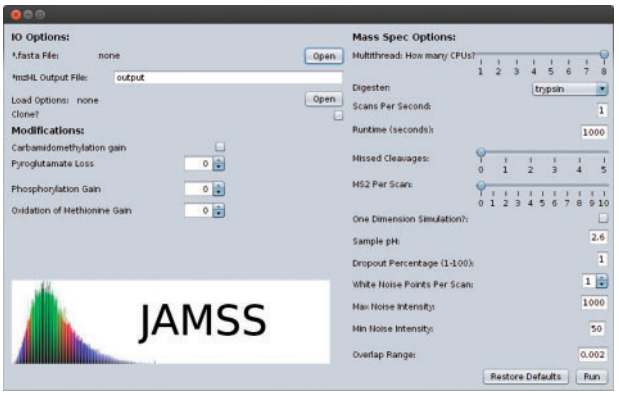

Fig. 1. JAMSS has a straightforward GUI interface to facilitate parameter selection for MS simulation

profile of the peptide (or PTM-modified peptide, if applicable) is fed into the same machine learning model used in Noyce *et al.* (2013) to predict the retention time of the peptide (see Noyce *et al.*, 2013 for details). The intensity of the peptide is determined by the user-provided intensity or, if none is provided, an inverse exponential sample. The shape and variance of each isotope trace in a molecular envelope's isotopic envelopes is modeled using the same mechanisms in Noyce *et al.*, (2013). Isotope trace shapes are determined via a modified Gaussian function with sufficient variation so that no two isotope traces are identical, providing variation for replicate runs. Further variation is achieved by modeling RT shifts as normally distributed events. From there, each centroid is subject to general noise in *m/z*, as well as intensity-specific noise in *m/z* (providing splayed isotope traces in the head/tail regions).

Memory usage is bounded by having the MS objects periodically writing their produced centroids onto disk. Frequency of writing out is determined automatically as a factor of the size of the JVM selected by the user and the CPUs in use by the user. For faster processing, more RAM can be selected by the user at runtime. After all centroids are created, the program finishes each RT scan by merging points within the resolution set by the user and generating an mzML output file, as well as .csv files providing all necessary meta-information to trace the provenance of each signal in the mzML file to the protein it originated from. This provenance meta-information is essential to creating ground truth data for MS data processing algorithm evaluation.

3 Results

JAMSS is a GUI-based MS simulator in Java (see Fig. 1). It creates fully annotated complex proteomic datasets in mzML with straightforward.csv provenance meta-information. It can be used to generate LC-MS and MS data, allowing for the evaluation of a wide range of data processing algorithms such as isotope trace extraction (both in chromatographic and non-chromatographic applications—Smith *et al.*, 2012), isotopic envelope extraction, molecular envelope extraction and reduction, and correspondence (Smith *et al.*, 2013b). It has a limited number of intuitive parameters, a self-contained one-click installation with no external libraries or dependencies, and supports multithreading. It creates isotope traces with realistic variance in both *m/z* and intensity and has a user-set memory upper bound. JAMSS handles variable PTMs and static PTMs. JAMSS features controlled randomized trace shape generators to create

run-to-run variation in replicates but uses controlled random seeding so it is possible to produce a clone of a previous run. It maintains relative protein abundance in isotope traces accounting for isotope trace variability and abundance distribution over PTMs. JAMSS also models RT shifts in order to provide more realistic replicates for generating datasets to test LC-MS correspondence algorithms.

References

Bielow,C. *et al.* (2011) MSSimulator: simulation of mass spectrometry data. *J. Proteome Res.,* **10**, 2922–2929.

Conley,C. *et al.* (2014) Massifquant: open-source Kalman filter based XC-MS feature detection. *Bioinformatics* **30**, 2636–2643.

Michalski,A. *et al.* (2011) More than 100,000 detectable peptide species elute in single shotgun proteomics runs but the majority is inaccessible to data-dependent LC-MS/MS. *J. Proteome Res.,* **10**, 1785–1793.

Noyce,A.B. *et al.* (2013) Mspire-Simulator: LC-MS shotgun proteomic simulator for creating realistic gold standard data. *J. Proteome Res.,* **12**, 5742–5749.

Schulz-Trieglaff,O. *et al.* (2008) LC-MSsim—a simulation software for liquid chromatography mass spectrometry data. *BMC Bioinformatics* **9**, 423.

Smith,R. *et al.* (2012) Statistical agglomeration: peak summarization for direct infusion lipidomics. *Bioinformatics* **29**, 2445–2451.

Smith,R. *et al.* (2013a) Controlling for confounding variables in ms-omics protocol: why modularity matters. *Brief. Bioinform.* **15**, 768–770.

Smith,R. *et al.* (2013b) LC-MS alignment in theory and practice: a comprehensive algorithmic review. *Brief. Bioinform.* **16**, 104–117.

Smith,R. *et al.* (2013c) Novel algorithms and the benefits of comparative validation. *Bioinformatics,* **29**, 1583–1585.

Smith,R. *et al.* (2014) Proteomics, lipidomics, metabolomics: a mass spectrometry tutorial from a computer scientist's point of view. *BMC Bioinformatics*, **15**(Suppl. 7), S9.