Data and text mining

mspire: mass spectrometry proteomics in Ruby

John T. Prince\textsuperscript{1,2} and Edward M. Marcotte\textsuperscript{1,2,3,}\textsuperscript{*}

\textsuperscript{1}Institute for Cellular and Molecular Biology, \textsuperscript{2}Center for Systems and Synthetic Biology and \textsuperscript{3}Department of Chemistry and Biochemistry, University of Texas, Austin, TX 78712, USA

Received on May 15, 2008; revised on September 26, 2008; accepted on October 1, 2008

Advance Access publication October 16, 2008

Associate Editor: John Quackenbush

1 INTRODUCTION

The analysis of mass spectrometry (MS) proteomics data is challenging on many fronts. Datasets are complex, with information spanning multi-level hierarchies, and they are also very large—files are often of near gigabyte size. Access to MS proteomics data is increasing with the advent of standardized formats, such as mzXML and mzData formats, converters for intermediate file types in typical proteomics software (SEQUEST) and downstream analysis, such as false identification rate (FIR) determination. The ProteomeCommons.org IO framework also has the ability to read/write and convert common data formats (Falkner et al., 2007), but this library is written in Java and does not provide any higher level language tools.

mspire is a software package for working with MS proteomics data as outlined in Figure 1A.

2 FEATURES

mspire relies on several memory-saving techniques that are critical for working with large data files. Large quantities of objects are implemented as Arrayclass (http://arrayclass.rubyforge.org) objects, providing highly efficient memory usage (Fig. 1B), while preserving accessor behavior common to typical Ruby objects.

By default, spectra from MS file formats (mzXML and mzData) are decoded into memory-efficient strings and are only completely cast when spectral information is accessed. An option is also available for storing only byte indices of spectral information that can be used for fast, random access of spectra or for reading files of essentially unlimited size.

Performance reading and then accessing two spectra across thousands of mzXML files from the PeptideAtlas is shown in Figure 1C. Late evaluation of a spectrum allows files to be read at \( \sim 20 \text{MB/s} \) with no file-size limit.

2.2 Reading MS proteomics data formats

mspire parses mzXML and mzData formats into a unified object model to simplify working with liquid chromatography (LC) MS...
Fig. 1. (A) Overview of mspire functionality. Black arrows and gray boxes depict mspire functionality. From left to right, mspire creates randomized databases (DBs) for FIR determination. MS::MSRun is a unified model for working with LC-MS/MS datasets. The Bioworks search engine produces peptide spectral matches (PSMs) in a .srf binary file or XML format. mspire extracts PSMs and presents them via a simple interface, SpecID, while preserving access to the underlying data structures. FIRs can be determined with various downstream software tools and reread into SpecID objects. SBV, sample bias validation. (B) mspire uses Arrayclass objects for efficient memory usage. GC, garbage collection; AC, Array-class; AF, Arrayfields; class, a traditional ruby object; SStruct, SuperStruct. (C) Lazy evaluation of spectra allows very large files to be read quickly. Shown are the times to read all 7830 well-formed mzXML files from PeptideAtlas and access two spectra for 'io' and 'string' lazy evaluation methods. A total of 181 files >350 MB in size were not read with the 'string' option. (D) Object model for capturing MS runs. (E) 3: an MSRun object can be instantiated with several lazy evaluation schemes. 4: typical instantiation. 6–8: total number of scans, the number of MS scans, and the number of MS/MS scans. 9: retrieves the start and end m/z values for all MS/MS scans. 11: a Ruby block that selects only MS/MS scans. 13–16: the scans are mapped to intensities; the block (designated between the 'do' and 'end') receives the scan object and returns the value of the last line, which is collected as an array (list_of_intensities). 14–15: chained method calls (equivalent to calling prc.intensity).

and MS/MS runs. Figure 1D shows the basic class hierarchy and Figure 1E demonstrates a simple ‘use case’.

2.3 Bioworks SEQUEST results files (.srf)
Bioworks previously produced separate text files for each spectrum, but now outputs a single SEQUEST results file (.srf) for each set of searches. This increases the speed of a search, decreases disk space usage and is much easier to work with in file system operations. Unfortunately, because the output is binary, accessing its contents can be difficult and downstream analysis tools (outside of Bioworks) do not currently support this format.

We created a reader for .srf files using the Ruby ‘unpack’ function. It extracts both spectral information and SEQUEST results. The reader is fast and also works across platforms because it does not rely on any vendor software libraries.

2.4 Reading/writing spectral identification formats
Even when derived from the same upstream data source, formats for working with spectra identifications can vary widely. We designed readers and writers for common downstream spectral-identification software formats for SEQUEST-based data: pepXML files which are used in the trans-proteomic pipeline (Protein Prophet) and also the .sqt format, which can be used with DTASelect and Percolator (Kall et al., 2007).

Readers are tailored to their respective format so that users can not only extract format-specific information easily but also implement a common interface so that users can easily extract information shared across these formats.

2.5 Determining FIRs
Bioworks software support for determining FIRs is currently non-existent, and so downstream tools are necessary. mspire supports peptide FIR determination from target-decoy database searches (both the creation of decoy databases and the summary of search results), PeptideProphet and Percolator. Known biases in sample content can also be used to establish an FIR.

ACKNOWLEDGEMENTS
Simon Chiang offered helpful discussion on the implementation of lazy evaluation of spectrum.

Funding: National Science Foundation; the National Institutes of Health; the Welch Foundation (F1515); Packard Fellowship (to E.M.M.). NIH grant numbers (GM067779,GM076536).

Conflict of Interest: none declared.

REFERENCES

