BIOINFORMATICS APPLICATIONS NOTE Vol. 24 no. 23 2008, pages 2796–2797
doi:10.1093/bioinformatics/btn513

Data and text mining

mspire: mass spectrometry proteomics in Ruby

John T. Prince[1,2] and Edward M. Marcotte[1,2,3,*]

[1]Institute for Cellular and Molecular Biology, [2]Center for Systems and Synthetic Biology and [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

ABSTRACT

Summary: Mass spectrometry-based proteomics stands to gain from additional analysis of its data, but its large, complex datasets make demands on speed and memory usage requiring special consideration from scripting languages. The software library 'mspire'—developed in the Ruby programming language—offers quick and memory-efficient readers for standard xml proteomics formats, converters for intermediate file types in typical proteomics spectral-identification work flows (including the Bioworks .srf format), and modules for the calculation of peptide false identification rates.

Availability: Freely available at http://mspire.rubyforge.org. Additional data models, usage information, and methods available at http://bioinformatics.icmb.utexas.edu/mspire

Contact: marcotte@icmb.utexas.edu

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 repositories, such as PeptideAtlas (Desiere *et al*., 2006), but its analysis remains no less daunting. Strongly typed languages (e.g. C/C++ and Java) are well suited for intensive computational tasks, but less so for exploring landscapes of computational possibilities. Scripting languages (e.g. Python, Perl and Ruby) are ideal for quick prototyping and the exploration of new ideas, but can be too slow or memory inefficient for large datasets. Thus, a need exists for scripting language tools capable of dealing with the size and complexity of MS proteomics data.

Ruby is a full-featured programming language created with inspiration from Perl, Python, Smalltalk and Lisp. It is object oriented and remarkably consistent in its design. Ruby's syntax encourages the use of blocks and closures which lend flexibility and conciseness to programming style. Also, while it is powerful, Ruby is relatively easy to learn, making it a natural first programming language for budding bioinformaticians. Ruby does not have the same degree of support for scientific computation as Python (e.g. NumPy and PyLab), but it is building significant momentum in this area (e.g. SciRuby at http://sciruby.codeforpeople.com). These features encouraged our use of Ruby in the creation of a high-level library supporting MS proteomics analysis.

*To whom correspondence should be addressed.

A few libraries/tools exist for working with MS proteomics data outside of Ruby. InSilicoSpectro, the only other scripting language library, is an open-source library written in Perl for 'implementing recurrent computations that are necessary for proteomics data analysis'. While there is some overlap with the work described here (e.g. *in silico* protein digestion), that library is currently geared towards the support of the Phenyx and Mascot search engines and low-level spectral computation (Colinge *et al*., 2006), while mspire is geared towards supporting Thermo's Bioworks 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.

2 FEATURES

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

2.1 Memory usage and speed

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.

REXML, Ruby's standard library XML parser, can be far too slow when reading large XML files generated in MS proteomics. mspire can use either XMLParser or LibXML (both of which have C/C++ bindings) for rapid parsing of large files.

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 ∼20 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

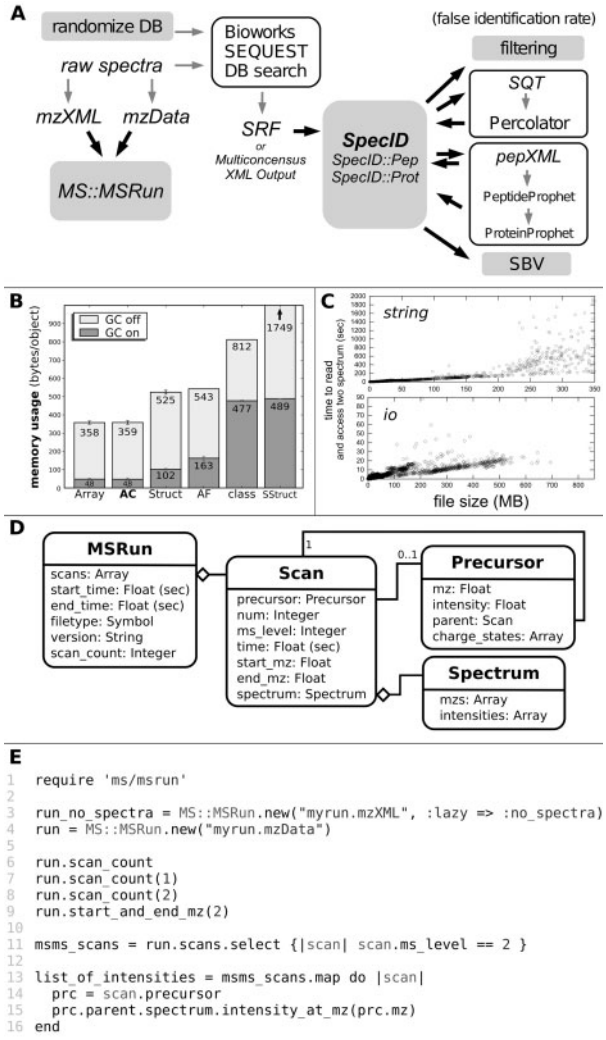

Fig. 1. (**A**) Overview of mspire functionality. Black arrows and gray boxes depict mspire functionality. From left to right, mspire creates randomized databases (DBs) for FIR determination. MS::MSRun is a unified model for working with LC-MS/MS datasets. The Bioworks search engine produces peptide 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

Coline,J. *et al*. (2006) InSilicoSpectro: an open-source proteomics library. *J. Proteome Res.*, **5**, 619–624.

Desiere,F. *et al*. (2006) The PeptideAtlas project. *Nucleic Acids Res.*, **34**, D655–D658.

Falkner,J.A. *et al*. (2007) ProteomeCommons.org IO framework: reading and writing multiple proteomics data formats. *Bioinformatics*, **23**, 262–263.

Kall,L. *et al*. (2007) Semi-supervised learning for peptide identification from shotgun proteomics datasets. *Nat. Methods*, **4**, 923–925.