ProteoWizard: Open Source Software for Rapid Proteomics Tools Development

Darren Kessner, Parag Mallick

Spielberg Family Center for Applied Proteomics, Cedars-Sinai Medical Center, Los Angeles, CA

Abstract

The ProteoWizard software project provides a modular and extensible set of open-source, cross-platform tools and libraries. The tools perform proteomics data analyses; the libraries enable rapid tool creation by providing a robust, pluggable development framework that simplifies and unifies data file access, and performs standard chemistry and LCMS dataset computations. Although many common computations are widely used throughout the proteomics community, there is no single standard, software development framework. Here we propose a software library to begin filling this need. The software is available now, under the Apache open source license.

Features

- supports the new HUPO-PSI mzML standard mass spectrometry data format
- uses modern C++ techniques and design principles
- cross-platform with native compilers (gcc, MSVC)
- has modular design, for testability and extensibility
- facilitates rapid development of data analysis tools
- open source license suitable for both academic and commercial projects (Apache v2)

Introduction

Proteomics laboratories often struggle with poor interoperability between data analysis software and data formats. Historically, vendor-specific, proprietary, closed formats made the development of cross-vendor data analysis tools challenging, and several open data formats were developed as a result. Open source software projects, such as the Trans-Proteomic Pipeline (TPP) and The OpenMS Proteomic Pipeline (TOPP), read these open data formats, and rely on conversion tools to translate proprietary data files. However, there has been no standard, open-source software library that provides both full access to and generation of data in these open formats.

The HUPO Proteomic Standards Initiative has taken the first important step in solving this problem, with the development of the new mzML data format standard, which will be released in June 2008.

ProteoWizard provides not only an implementation of the mzML standard, but a cross-platform data file abstraction layer. This allows the data analysis programmer to focus on the actual data analysis, without having to deal with specific details about the source file format or the operating system.

Researchers also face the task of evaluating different software tools, each with different interfaces. The ProteoWizard analysis framework provides a unified method for implementing and comparing analysis algorithms.

Architecture

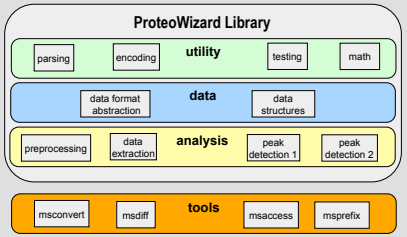

Figure 1. High level architecture of ProteoWizard

ProteoWizard is built from many independent libraries, grouped together in dependency levels. Each library is independently testable, and depends only on libraries in lower levels of the hierarchy. The utility layer contains independent classes that perform computations applicable in a wide variety of situations. The data layer abstracts the source data file, hiding any format-specific details. The analysis layer contains all scientific computation, in reusable modules. The tools layer code is responsible only for regulating interaction between the user and the analysis modules.

Data Layer Design

The underlying data model is a one-to-one translation from mzML data elements to C++ data structures. The data layer has the following features:

- a virtual interface for accessing spectrum lists, to allow lazy evaluation when accessing the spectra contained in a data file
- a plug-in Reader interface to allow reading of vendor proprietary data formats
- built-in diff calculation, for comparison of two data files, useful for validation after data format conversion or preprocessing
- iostream serialization to/from mzML and mzXML

```
<fileDescription>
  <fileContent>
    <cvParam cvLabel="MS" accession="MS:1000580" name="MSn spectrum" value=""/>
  </fileContent>
  <sourceFileList count="1">
    <sourceFile id="rawfile" name="data01.RAW" location="c:/data/raw">
      <cvParam cvLabel="MS" accession="MS:1000563" name="Xcalibur RAW file" value=""/>
      <cvParam cvLabel="MS" accession="MS:1000569" name="SHA-1 "value="6bde97 [...]"/>
    </sourceFile>
  </sourceFileList>
</fileDescription>
```

Figure 2a. mzML fragment

```
struct SourceFile : public ParamContainer
{
    string id;
    string name;
    string location;
};
```

```
struct FileDescription
{
    FileContent fileContent;
    vector<SourceFilePtr> sourceFilePtrs;
    vector<Contact> contacts;
};
```

Figure 2b. Corresponding ProteoWizard data structures

Analysis Layer Design

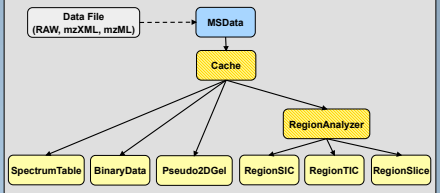

Figure 3. Analysis module design for the *msaccess* tool

The analysis layer contains classes that provide a simplified interface to the underlying data structures. The analysis modules provide reusable plug-in functionality that may be incorporated into data analysis tools. The design above shows the use of the Cache and RegionAnalyzer modules for efficient data access and processing by other modules in the *msaccess* tool, a tool that allows users to easily extract segments of LCMS data.

```
class HelloAnalyzer : public MSDataAnalyzer
{
    public:

    virtual void open(const DataInfo& dataInfo)
    {
        cout << "sourceFilename: " << dataInfo.sourceFilename << endl;
        cout << "outputDirectory: " << dataInfo.outputDirectory << endl;
    }

    virtual UpdateRequest updateRequested(const DataInfo& dataInfo,
        const SpectrumIdentity& spectrumIdentity) const
    {
        return UpdateRequest_NoBinary;
    }

    virtual void update(const DataInfo& dataInfo,
                        const Spectrum& spectrum)
    {
        cout << "spectrum: "
             << spectrum.index << " "
             << spectrum.id << " "
             << "ms"
             << spectrum.cvParam(MS_ms_level).value << " "
             << spectrum.spectrumDescription.
                 scan.cvParam(MS_filter_string).value
             << endl;
    }

    virtual void close(const DataInfo& dataInfo) {}
};
```

Figure 4a. HelloAnalyzer implementation

```
int main(int argc, const char* argv[])
{
    try
    {
        MSDataAnalyzerApplication app(argc, argv);

        if (app.filenames.empty())
        {
            cout << "Usage: hello_analyzer [options] [filenames]\n";
            cout << "Options:\n" << app.usageOptions << endl;
            return 1;
        }

        HelloAnalyzer analyzer;
        app.run(analyzer, &cerr);

        return 0;
    }
    catch (exception& e)
    {
        cerr << e.what() << endl;
    }
    catch (...)
    {
        cerr << "Caught unknown exception.\n";
    }

    return 1;
}
```

Figure 4b. hello_analyzer program

Tools

msconvert: data format conversion from vendor proprietary formats to mzML and mzXML

msdiff: comparison of two data files, for validation of conversion and preprocessing

msaccess: command line access to mass spec data files, including spectrum binary data and metadata, selected ion chromatograms, and pseudo-2D gel image creation.

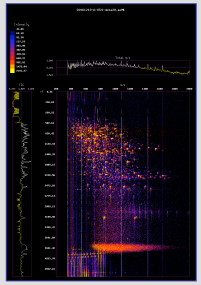

Figure 5a. Pseudo 2D gel image generated by *msaccess*

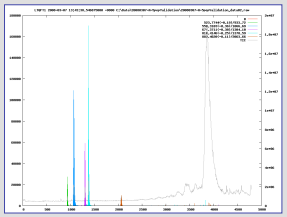

Figure 5b. gnuplot image generated by script from data extracted by *msaccess*

Current Development

- The Trans-Proteomic Pipeline tools will be using ProteoWizard to support mzML in the next major release.
- We are currently in ongoing collaboration with other proteomics software groups to support all vendor formats (Institute for Systems Biology, Vanderbilt University Mass Spectrometry Research Center).
- The mzML standard will be officially released at ASMS 2008.

More Information

Darren Kessner
darren.kessner@cshs.org

ProteoWizard
http://proteowizard.sourceforge.net

Spielberg Family Center for Applied Proteomics
http://sfcap.cshs.org

HUPO Proteomic Standards Initiative
http://www.psidev.info