A cross-platform toolkit for mass spectrometry and proteomics

To the Editor:

Mass spectrometry–based proteomics has become an important component of biological research. Numerous proteomics methods have been developed to identify and quantify the proteins in biological and clinical samples[1], identify pathways affected by endogenous and exogenous perturbations[2] and characterize protein complexes[3]. Despite successes, the interpretation of vast proteomics data sets remains a challenge. There have been several calls for improvements and standardization of proteomics data analysis frameworks, as well as for an application-programming interface for proteomics data access[4,5]. In response, we present here the ProteoWizard Toolkit, a robust set of open-source, software libraries and applications designed to facilitate proteomics research. The libraries implement the first-ever, noncommercial, unified data access interface for proteomics, bridging field-standard open formats and all common vendor formats. In addition, diverse software classes enable rapid development of vendor-agnostic proteomics software. Additionally, ProteoWizard projects and applications, building upon the core libraries, are becoming standard tools for enabling significant proteomics inquiries.

Historically, the development of proteomics software tools has been hindered by three factors: first, developers must develop readers and writers for the numerous file formats used for holding mass spectrometry data and analysis results, which range from vendor-specific mass spectrometry data formats to software application–specific formats; second, developers must implement numerous common, but critical algorithms (e.g., protein digestion, mass computation, peak integration, charge-state detection and isotope deconvolution), which is both time-consuming and error-prone; and third, comparison and validation of analysis algorithms is complicated by the vast diversity of possible workflows. Together, these three impediments create a significant bottleneck in the development of new proteomics software applications. Beyond slowing the pace of proteomics software development, these impediments have also hampered the field of proteomics by interfering in the meaningful comparison, sharing and exchange of data analyses obtained on different platforms or by different laboratories.

Efforts to mitigate these issues led initially to the development of several 'open' interchange formats[6,7] and a series of software tools that extracted data from vendor formats into open formats. The majority of mass spectrometry vendors also now provide approaches to export their data to open formats. Although this is an important step forward, both the academic and commercial tools suffer from a few limitations. For example, despite extensive conversion tools, a robust code-base that allowed developers to easily extract data from data files for use in their own applications did not exist. Efforts by our group and by the OpenMS team attempted to address this issue[8,9]. In addition, early converters depended upon instrument control software libraries; consequently, users without instruments could neither access nor convert vendor data files. Furthermore, each vendor format had its own converter (e.g., MassWolf for Waters Files and ReAdW for Thermo Fisher files), thus complicating software maintenance. Lastly, despite the amazing success of these open formats and the proliferation of tools that use them, the converter-centric, common-format approach did not address the issue of direct access to primary raw data. Most native vendor formats encode valuable, but vendor-specific, metadata including details of instrument settings and instrument readouts.

Direct access to raw, primary data can critically affect the comparability of experimental platforms because common computational processing steps associated with export, such as centroiding, may affect benchmarking results. The comparison challenge is even more important for data analysis approaches; a bioinformatics approach could easily appear inferior because of unintended (possibly error-filled) upstream data processing steps. Lastly, cross-platform comparison of workflows (both computational and experimental) is hampered when tools are developed to read files from a particular vendor but cannot be applied to data from other instrument types. As the field of proteomics attempts to become more robust, the need for integrated pipelines for processing and analyzing complex proteomics data sets in a platform-agnostic manner has become critical.

With version 3.0 of the ProteoWizard Toolkit[8], we attempt to mitigate these challenges through open-source, permissively licensed, cross-platform software. The Toolkit has two components: first, a suite of libraries that facilitate the development and comparison of tools for proteomics data analysis; and second, a set of tools, developed using these libraries, that performs a wide array of common proteomics analyses. The Toolkit has been developed under modern design principles in the C++ language and supports a variety of platforms with native compilers (GCC on Linux, MSVC on Windows and XCode on OSX). The toolkit was released under the Apache 2.0 license[10] to ensure that it can be used in both academic and commercial projects. New to ProteoWizard 3.0 and unlike previous efforts, vendor reader libraries are now directly distributed with the Toolkit independently of instrument control libraries (a further description of new features can be found in **Supplementary Text 1**). Furthermore, ProteoWizard employs a single converter and access interface for all formats; this singular point of maintenance allows a more stable and optimized set of tools. Additional robustness comes from ProteoWizard's use of a continuous integration and testing environment. Although common in commercial projects, this scale of quality assurance is uncommon in traditional academic projects.

ProteoWizard is built upon a modular framework of many independent libraries grouped in dependency levels (**Fig. 1a**). Each library only depends on libraries in lower levels of the hierarchy. The data layer provides a unified access interface to mass spectrometry data, independent of the format-specific details associated with a given source file. The underlying data model of the data layer directly translates Human Proteome Organization Proteomics Standard Initiative (HUPO-PSI) data elements to C++ data structures. In **Supplementary Text 2**, we show this mapping for a piece of the msData module that implements mzML[11]; equivalent mappings exist for mzIdentML[12] and TraML[7].

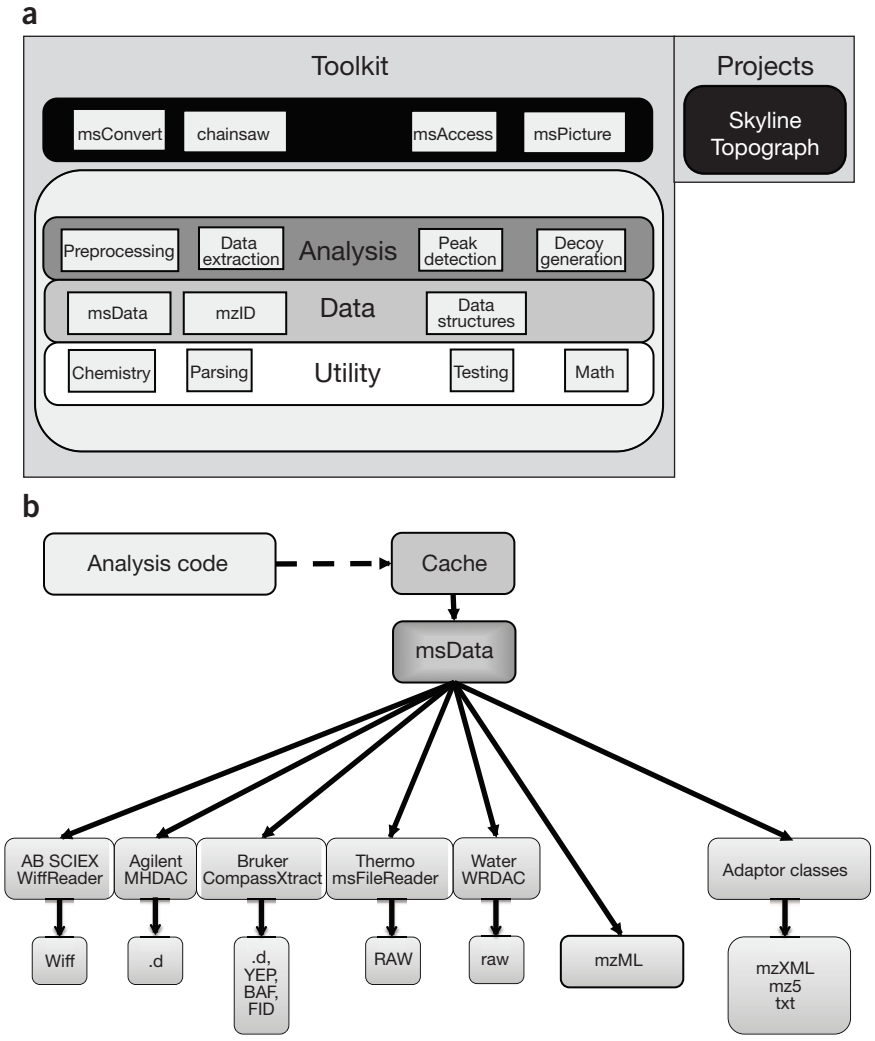

Figure 1 Design of ProteoWizard. (**a**) ProteoWizard uses modern design principles to implement a modular framework of many independent libraries grouped in dependency levels with strict interfaces. This allows extensive development at each level while enforcing stability. (**b**) The data layer presents a unified access interface to mass spectrometry data. The modular framework allows additional readers for diverse file types to be easily added by means of plug-in adaptor classes. Developers only need interact with the primary interface to access data, agnostic to the details of an input source file.

Field-standard open formats (e.g., mzML, mzXML, MGF, pepXML and mzIdentML) and vendor proprietary formats are handled with a plug-in reader interface (**Fig. 1b**). In partnership with proteomics standards bodies and instrument and software vendors, we have developed a series of adapters that translate between input files and the core msData data structures to support a wide range of formats (**Supplementary Tables 1** and **2**). These adapters bridge vendor-provided libraries that read proprietary formats and the fully open ProteoWizard data layer. Through a series of generous licenses, the ProteoWizard Software Foundation (Los Angeles) has permission to distribute vendor-provided libraries from AB SCIEX, Agilent, Bruker, Thermo Fisher Scientific and Waters with the ProteoWizard Toolkit. Consequently, bioinformatics developers are not required to have direct access to an instrument to develop software that can analyze data generated by it.

Furthermore, any application built upon the ProteoWizard framework is largely format agnostic for the dominant formats in the field. By writing their software using ProteoWizard's msData application programming interface (API), developers can focus on algorithmic challenges, rather than on the complex details of the wide array of formats prevalent throughout the field of proteomics. Furthermore, the use of the ProteoWizard API has the potential to improve the robustness and reliability of other proteomics software efforts. As vendors frequently change their file formats to accommodate new instruments and public standards evolve rapidly, software tools can rapidly become unusable unless substantial resources are devoted to continually update data-reader code. The robust upkeep of ProteoWizard, in concert with its widespread use, will effectively reduce the investment that the public has to make in maintaining the longevity of open-source software.

Supplementary Example 1 illustrates how the mass spectral data from a mass spectrometer data file can be browsed and printed. Also highlighted in **Supplementary Example 1** are the benefits of ProteoWizard's Common Language Infrastructure bindings, which allow the library to be accessed from diverse languages including C#, IronPython and Visual Basic. **Supplementary Example 2** illustrates how peptide and protein identification data can be browsed and printed. In **Supplementary Example 3**, we illustrate how the mzR library enables ProteoWizard-based data access within the R statistical analysis toolkit. Notably, mass spectrometry data can be used for a variety of applications other than proteomics investigation. The data layer does not impose any restrictions that inhibit its use for any mass-spectrometry-based problem. ProteoWizard is already used in metabolomics applications[13] and should find utility in analysis of glycomics data.

Below the data layer is the 'utility layer' (**Fig. 1a**). The utility layer contains applications that perform computations, such as binary to text encoding, XML parsing and mathematical calculations that are common in data analysis. A list of available utility classes is provided in **Supplementary Table 3**. Although the majority of computations available in these classes are straightforward, their implementation can be time consuming. Using ProteoWizard, developers are able to focus on developing novel algorithms, rather than on redundant implementation of requisite parsing and data handling code, thus accelerating the development timeline.

The 'analysis layer' further builds upon the data layer and provides common proteomics-centric analysis modules. A major bottleneck in proteomics software development can arise from the time required to implement the vast array of standard operations routinely required of a proteomics algorithm, such as computing the mass of a peptide (**Supplementary**

Example 4) or performing an *in silico* digest of a protein read from a FASTA file (**Supplementary Example 5**). There are also independent modules for handling chemical formulas, peptide calculations and isotope envelopes. All these computations are contained in reusable, platform-independent modules in the analysis layer. A list of available analysis classes is provided in **Supplementary Table 3**.

Additional analysis modules are currently in development with an emphasis on establishing standard interfaces for common proteomics computations, such as peak picking, isotope deconvolution and precursor estimation[14]. Our goal is to work collaboratively to create a modular analysis infrastructure in which experts will be able to contribute a module that can then be plugged into various software tools. This will allow, for example, an expert in signal processing to contribute a peak picker without having to handle details of file formats, operating systems or command-line configurations. The ProteoWizard Toolkit also includes a number of small, useful applications, listed in **Supplementary Table 4**, that are built upon the libraries. These applications support data conversion (msConvert, msConvertGUI and idConvert), data visualization (msPicture and seeMS), data access (msAccess, msCat, idCat and msPicture), data analysis (peekaboo and msPrefix[14]) and basic proteomics utilities (chainsaw).

Beyond the ProteoWizard Toolkit, the ProteoWizard Software Foundation has built several projects on top of the ProteoWizard Toolkit that provide useful end-user applications. The most widely known example, Skyline[15], is becoming the standard tool for targeted proteomics investigation. A second project, Topograph, is focused on measuring protein turnover in metabolic labeling time-course experiments. Other projects are underway. To be included in ProteoWizard, projects must demonstrate broad applicability within the field and active ownership within the contributing organization. They must also adopt nonrestrictive licensing[1] and continue to develop new features in open-source formats. Project contributors must provide thorough automated testing and participate in the ProteoWizard build and continuous integration processes.

The ProteoWizard Toolkit and Projects attempt to provide useful analytic tools to the proteomics community while simplifying the process of software development and bioinformatics for mass spectrometry and proteomics. Our hope is that a standardized toolkit will enable rigorous development and assessment of diverse computational approaches to rapidly accelerate proteomics research.

Note: Supplementary information is available at http://www.nature.com/doifinder/10.1038/nbt.2377.

ACKNOWLEDGMENTS

This work is supported by the Wunderkinder Foundation, Redstone Family Foundation and Canary Foundation, and the National Cancer Institute and US National Institutes of Health grants and contracts P41 RR011823, CCNE-TR 5U54CA119367, CCNE-T 1U54CA151459, PSOC-MCSTART 5U54CA143907, R01CA126218 and U24CA126479. R.L.M. and C.P. have been supported by National Science Foundation MRI grant No. 0923536. L.G. has been supported by the European Union 7th Framework Program PRIME-XS project, grant agreement number 262067. The authors also thank the ProteoWizard developer community and TPP developer community for their contributions to the project and manuscript.

COMPETING FINANCIAL INTERESTS

The authors declare no competing financial interests.

Matthew C Chambers[1,18], Brendan Maclean[2,18], Robert Burke[3,18], Dario Amodei[4], Daniel L Ruderman[3], Steffen Neumann[5], Laurent Gatto[6], Bernd Fischer[7], Brian Pratt[8], Jarrett Egertson[2], Katherine Hoff[3], Darren Kessner[3], Natalie Tasman[8], Nicholas Shulman[2], Barbara Frewen[2], Tahmina A Baker[2], Mi-Youn Brusniak[9], Christopher Paulse[9], David Creasy[10], Lisa Flashner[3], Kian Kani[3], Chris Moulding[11], Sean L Seymour[12], Lydia M Nuwaysir[12], Brent Lefebvre[12], Frank Kuhlmann[13], Joe Roark[13], Paape Rainer[14], Suckau Detlev[14], Tina Hemenway[15], Andreas Huhmer[15], James Langridge[16], Brian Connolly[17], Trey Chadick[17], Krisztina Holly[11], Josh Eckels[17], Eric W Deutsch[9], Robert L Moritz[9], Jonathan E Katz[3], David B Agus[3], Michael MacCoss[2], David L Tabb[1] & Parag Mallick[3,4]

[1]Department of Biomedical Informatics, Vanderbilt University, Nashville, Tennessee, USA. [2]Department of Genome Sciences, University of Washington, Seattle, Washington, USA. [3]Center for Applied Molecular Medicine, University of Southern California, Los Angeles, California, USA. [4]Canary Center for Cancer Early Detection, Stanford University, Stanford, California, USA. [5]Department of Stress & Developmental Biology, Leibniz Institute for Plant Biochemistry, Halle (Saale), Germany. [6]Proteomics Services, Cambridge Centre for Proteomics, Cambridge, England. [7]Genome Biology, EMBL Heidelberg, Germany. [8]Insilicos, Seattle, Washington, USA. [9]Institute for Systems Biology, Seattle, Washington, USA. [10]Matrix Science, Boston, Massachusetts, USA. [11]USC Stevens Institute for Innovation, University of Southern California, Los Angeles, California, USA. [12]AB SCIEX, Foster City, California, USA. [13]Agilent Technologies, Santa Clara, California, USA. [14]Bruker Daltonik GmbH, Bremen, Germany. [15]Thermo Fisher Scientific, San Jose, California, USA. [16]Waters Corporation, Manchester, UK. [17]LabKey Software, Seattle, Washington, USA. [18]These authors contributed equally to this work.
email: paragm@stanford.edu

1. Schwanhausser, B. *et al. Nature* **473**, 337–342 (2011).
2. Raj, L. *et al. Nature* **475**, 231–234 (2011).
3. Bouwmeester, T. *et al. Nat. Cell Biol.* **6**, 97–105 (2004).
4. Patterson, S.D. *Nat. Biotechnol.* **21**, 221–222 (2003).
5. Askenazi, M., Parikh, J.R. & Marto, J.A. *Nat. Methods* **6**, 240–241 (2009).
6. Pedrioli, P.G. *et al. Nat. Biotechnol.* **22**, 1459–1466 (2004).
7. Orchard, S. *et al. Proteomics* **10**, 1895–1898 (2010).
8. Kessner, D., Chambers, M., Burke, R., Agus, D. & Mallick, P. *Bioinformatics* **24**, 2534–2536 (2008).
9. Sturm, M. *et al. BMC Bioinformatics* **9**, 163 (2008).
10. Apache.Software.Foundation 2004. http://www.apache.org/licenses/LICENSE-2.0.html.
11. Martens, L. *et al. Mol. Cell. Proteomics* **10**, R110 000133 (2011).
12. Eisenacher, M. *Methods Mol. Biol.* **696**, 161–177 (2011).
13. Benton, H.P., Wong, D.M., Trauger, S.A. & Siuzdak, G. *Anal. Chem.* **80**, 6382–6389 (2008).
14. Luethy, R. *et al. J. Proteome Res.* **7**, 4031–4039 (2008).
15. MacLean, B. *et al. Bioinformatics* **26**, 966–968 (2010).

Assessing unintended hybridization-induced biological effects of oligonucleotides

To the Editor:

Oligonucleotide drugs come in various forms, lengths and modifications, archetypes being single-stranded antisense oligonucleotides (ASOs)[1] or double-stranded small interfering RNA (siRNAs)[2]. Similar to all drugs, oligonucleotide therapeutics carry the risk of causing unintended toxicities or side effects. The mechanisms of toxicity for oligonucleotides can be subdivided into hybridization-independent and hybridization-dependent effects. Hybridization-independent toxicities are due to interactions between