Published in final edited form as:
J Proteome Res. 2009 June ; 8(6): 3212–3217. doi:10.1021/pr900169w.

A software platform for rapidly creating computational tools for mass spectrometry-based proteomics

Damon May, **Wendy Law**, **Matt Fitzgibbon**, **Qiaojun Fang**, and **Martin McIntosh***
Molecular Diagnostics Program, Fred Hutchinson Cancer Research Center

Abstract

We describe and demonstrate the proteomics computational toolkit provided in the open-source msInspect software distribution. The toolkit includes modules written in Java and in the R statistical programming language to aid the rapid development of proteomics software applications. It contains tools for processing and manipulating standard MS data files, including signal processing of LC-MS data and parsing of MS/MS search results, as well as for modeling proteomics data structures, creating charts, and other common tasks. We present this toolkit's capability to rapidly develop new computational tools by presenting an example application, Qurate, a graphical tool for manually curating isotopically labeled peptide quantitative events.

Keywords

Qurate; msInspect; LC-MS; MS/MS; quantitation; isotopic labeling

INTRODUCTION

Computational researchers in proteomics who wish to make their research methods available for wider use are burdened with several challenging aspects of software development that are often only tangential to their area of interest. These tasks include effectively parsing and validating user input, reading and managing standardized data files specific to proteomics, managing protein database files such as FASTA databases, calculating peptide masses, performing virtual digestion of proteins, and many others. Each of these activities must be addressed prior to the development of any new computational strategy, but together they present a significant barrier to the computational researcher who intends to develop new tools suitable for the general research community.

Several open-source software packages exist that can help ease this burden. The Trans-Proteomic Pipeline (1) (TPP), for instance, provides tools written in various languages, including tools for parsing the standard mzXML file format. ProteoWizard(2) provides tools for parsing the mzML file format, and it goes a step further to provide a set of C++ tools and libraries designed to help researchers write proteomics software. In a similar spirit, msInspect contains its own (previously undocumented) growing software toolkit which has been used to create several published applications, including the original msInspect(3) application, as well as the msInspect/AMT(4,5) suite of tools for Accurate Mass and Time analysis, and MRMer (6), a tool for Multiple Reaction Monitoring (MRM) data analysis. Although these specific applications developed on the msInspect toolkit have been documented previously, the underlying software structure has not.

*E-mail: mmcintos@fhcrc.org.

Like the TPP, the msInspect codebase contains software for parsing standardized files (much of which is based on TPP code), and like ProteoWizard it contains a general toolkit to help build software applications. However, msInspect also includes some unique contributions. The platform is written in Java, and so is platform independent, and it integrates with the R statistical language, fostering the development of complex data analysis procedures and graphical representations. It also provides a robust framework for creating applications that may be invoked from the command line or via a graphical interface, removing much of the software development burden from the researcher.

In this technical note we give an overview of the application development framework contained in the msInspect platform, and we demonstrate its capabilities by describing a useful new interactive application using the platform that was built in under 5 days of development time, including designing, testing and debugging. The application, Qurate, reads and writes standardized files resulting from isotopically labeled LC-MS/MS experiments and visually presents the data to a user for visual inspection and annotation of incorrect or poor quality quantitative events. Qurate then rewrites the standardized files using these curated results and allows for the recalculation of the peptide and protein ratios. The functionality contained in Qurate is not itself novel, as some or all of its features are available in some commercial applications. However, Qurate is a freely-available tool demonstrating that useful applications operating on standard data file formats can be built with the platform in a timely manner. The msInspect platform, and all platform applications such as msInspect/AMT, MRMer and Qurate, are available from our website in an unrestricted open-source format.

METHODS

msInspect Development Platform

The msInspect platform is largely written in Java, with several statistical tasks performed in the R statistical language. The source code for the current released version is publicly available, as is the Subversion source code repository containing the latest code. Because the source code with documentation, a Developer's Guide describing platform tools in detail, and a support forum are available online at proteomics.fhcrc.org, we provide only a brief overview of the platform here. The software is released under an Apache v2 license that allows it to be modified without restriction and incorporated into both academic and commercial software. Figure 1 summarizes the types of tools provided by the msInspect platform.

Software Tools—msInspect contains a number of tools that are not specific to proteomics data analysis but are intended to support rapid software development and the generation of general-use tools. These tools perform routine tasks that are required in many different types of software applications.

The *Command Module Framework* allows the developer to easily create a discrete set of functionality accessed as a single “command” within the msInspect platform, useful in developing batch-mode, high-throughput processing. A Command Module defines a set of arguments to be supplied by the user, which are automatically validated when the module is invoked. For example, a Command Module for filtering database search results in a PepXML file may have mandatory arguments defining the input and output file locations, and optional arguments for various types of filtering. Command Modules may be invoked via the command line or via a graphical window that is automatically generated by the msInspect platform. This structure also provides automatic documentation of the Command Module: the msInspect platform can generate a usage guide automatically based on the arguments that are defined, and it can tell the user what went wrong when an argument fails to validate (e.g., a nonexistent input file or a negative value for a mass tolerance). Arguments may be of many different types, including numbers, files, and restricted value sets such as true-false arguments. Many

Command Modules are provided with the platform, including several that are documented extensively to help new developers get up to speed quickly.

The platform's *Testing Framework* takes advantage of the information defined for each Command Module, and the JUnit testing structure in Java, to aid the development of quality assurance or regression tests. The testing framework helps in building a set of tests that may be run automatically whenever changes are made to application code.

The platform contains several *File Management Utilities*. XML parsing and rewriting is supported, and the msInspect platform contains utilities for efficiently reading and writing tab-delimited files, which are particularly useful for rapid scientific application development because they can be opened by spreadsheet programs such as Excel. The platform can also create and manage temporary files as needed.

The msInspect platform contains many *Graphical Utilities*. It uses the JFreeChart Java API for most chart generation; users can easily save chart data to tab- or comma-delimited files. Bar charts, line charts, histograms, box-and-whisker charts, and others can be generated automatically. 3D contour plots are supported using the R statistical language (see Figure 3).

For applications that require translation into multiple natural languages, the msInspect platform provides structure for using text bundles stored in non-code files that may be translated either in-house or by independent translation services.

Proteomics-Specific Tools—The msInspect platform also contains many tools that are more specific to proteomics application development.

Utilities for Reading and Writing Proteomics Data Files: Spectrum data from mzXML files may be read into memory, manipulated, and written back out in mzXML format, e.g., for correcting mass calibration problems or for cutting out a small piece of a large mzXML file to share with a colleague (Command Modules that perform both of these tasks are provided with the platform). PepXML files containing database search results may also be written as well as read, allowing for filtration of database search results, normalization of quantitative ratios, etc (also provided). ProtXML files containing protein-level results may be read and processed. Protein databases in the FASTA format may be read and manipulated, e.g., for easy creation of custom forward-reverse databases. The msInspect platform also reads and writes the APML format for LC-MS peptide features(7). Care is taken to keep the memory footprint small. For example, XML files with a highly repetitive structure (like PepXML and mzXML) are processed in a streaming fashion, rather than keeping an in-memory representation of the entire file.

Utility Classes for Common Tasks in Proteomics: In silico protein digestion, peptide neutral mass calculation, theoretical isotopic distribution (and calculation of deviation from theoretical distribution), resampling of spectra onto a standardized grid, smoothing, background noise calculation and subtraction, and many other utilities are included in the platform.

Statistical Utilities: Since proteomics applications often involve statistical analysis, the msInspect platform includes utilities for calculating various summary statistics in Java, and also for linear regression. It also provides a full-featured interface with the R statistical language. This makes it easy to pass Java variables into R, work with them in R scripts, generate charts in R and display them, and retrieve results back into Java.

Use of the Platform to Build a Tool for Visual Curation of Quantitative Proteomics Experiments

The msInspect software has been used to create several comprehensive applications(3-6), but we discuss here the development of a tool to support visual inspection and curation of quantitative proteomics experiments. The software, called Qurate, allows the user to examine labeled quantitation events, such as SILAC, ICAT or Acrylamide labeling(8), visually, using several different informational charts, and to filter bad quantitation events out of the dataset. The software reads and writes the PepXML format(9) using utilities provided by the msInspect platform, and it is compatible with the XPress(10) and Q3(8) quantitation algorithms. Qurate is distributed freely at proteomics.fhcrc.org, and a self-contained demonstration is provided as supplementary material.

The motivation for developing Qurate stems from the insidious effects that errors in quantitative signal processing can have on the false discovery rate of quantitative proteomics experiments. For example, in an isotopically labeled experiment a bad peptide identification will typically lead to an extremely high or low ratio, depending on whether the peptide in question incorrectly matches a heavy or light species (since its ‘mate’ having the opposite label will not be present). False extreme ratios can stem from signal processing errors in the precursor ion scans, e.g., when coeluting peptides interfere with the signal, when low-intensity ions are dominated by noise, or when quantitation is based on a very small number of scans with high individual variability. These situations can all produce quantitative ratios that are extremely high or extremely low, erroneously causing the peptide in question to appear differentially abundant. Although the signal-processing error rate can be controlled in aggregate, the sheer volume of the calculations in any MS/MS interrogation means that, when comparing two complex protein mixtures differentiated by only a small number of proteins, the majority of extreme quantitative ratios identified by automated procedures may stem from errors of these types.

Fortunately, many of the errors in signal processing can be easily identified by visual inspection. For example, incorrect identifications often appear as an identified C13 (or higher) peak of an ion, which is easily observable. The existence of coeluting peptides missed by signal processing, too, can be seen easily by human eye. Qurate is intended as a means to curate, not an entire MS/MS experiment, but rather a small subset of its quantitative events, such as those for the proteins that may be selected for further evaluation, reducing the time wasted downstream on confirming proteins whose apparent differential abundance results from artifacts of signal processing. Manual curation of these ratios, while somewhat time-consuming, may save significant time in the later analysis of biomarker candidate proteins, which is typically a high-cost process.

Qurate provides a graphical interface that allows the user to navigate MS/MS database search results at the protein level contained in a ProtXML file (which may contain results from many sample fractions, as well as replicate runs), identify proteins of interest (by sorting based on various criteria, including quantitative ratio), and display a list of quantitative events that are used in calculating their ratios (alternatively, the user may provide a PepXML file containing peptide-level search results as a starting point). The user may then select a set of interesting quantitative events from these proteins and generate a set of charts visualizing the LC-MS spectra surrounding each event. To save a great deal of user time and effort for abundant peptides, quantitative events for the same peptide may be combined automatically into a single set of charts and curated as a single event if they occur in the same LC-MS fraction, with the same charge, with the same modifications, and with similar ratios.

For each quantitative event, four charts are generated (Figures 2 and 3): a set of intensity charts for each individual scan used in the quantitative event (Figure 2); a three-dimensional contour

plot of the region (Figure 3A, 3B); a heatmap showing a two-dimensional representation of the whole spectral region covered by the quantitative event, with different colors for different intensities (Figure 3C); and a chart showing the sum of all intensities throughout the quantitated region (Figure 3D). All charts are annotated with the *m/z* values of the peaks of the light and heavy versions of the peptide. Summary information for the quantitative event is given, and the theoretical isotopic peak distribution (assuming the quantitative ratio assigned by the quantitation algorithm) is shown (Figure 2). The user may mark each quantitative event as Good, Bad, or Unknown, and may provide a comment on each event. The same ratings may also be applied to the peptide identification, since sometimes it is possible to determine by looking at the spectra that the ID is likely incorrect (e.g., the theoretical monoisotopic *m/z* value is a Dalton lower than the *m/z* value of the lowest actual peak). These annotations may be saved to a tab-delimited file and reopened later. When the user has evaluated all the interesting events, Qurate can filter the 'Bad' events out of the source PepXML file. ProteinProphet may then be rerun in order to recalculate protein-level quantitative ratios using the remaining peptide quantitation events.

Specific use of msInspect Platform to develop Curate

Command Module Framework: The Qurate application is invoked (either graphically or on the command line) via a Command Module, using the Command Module framework described above. In order to capture the necessary information to begin analyzing data files, Qurate defines three required and six optional user arguments (input and output files, mass tolerances, protein names, etc.). These arguments are defined and processed within the Qurate Command Module, but the bulk of the code for argument capture and validation exists within the framework itself, and the framework invokes the Qurate code only if all arguments validate successfully. The user may easily display a user manual for Qurate, which is generated automatically via the framework using the argument definitions in the Qurate Command Module.

Graphical Utilities: Qurate displays several charts to the user to aid in curation of each quantitative event: a heatmap, a 3D contour plot, and multiple line charts to help visualize the MS1 spectra, and a line chart describing the theoretical isotopic distribution of the quantified peptide. The Qurate application itself simply manages the lists and arrays of data underlying the charts, which are all generated using the charting API in the platform.

File Utilities: Qurate reads ProtXML and PepXML files in order to provide the user with information about quantitative events, reads mzXML files to build charts based on MS1 spectra, writes out altered versions of the input PepXML files with bad quantitative events and identifications removed, and reads and writes tab-delimited files describing quantitative events of interest to the user. All of these interactions with data files are performed using platform utilities; the Qurate application itself simply manipulates Java objects representing proteomics data.

Proteomics Data Modeling: Qurate uses platform Java classes to model and manipulate protein identifications, peptide identifications and quantitative events, amino acid modifications, and LC-MS machine runs and spectra.

Self-Self Comparison Dataset—To demonstrate and pilot Qurate we examined a data set comparing a complex sample to itself, so that no proteins other than contaminants or those resulting from variations in biochemical processing (e.g., isotopic labeling) should be differentially abundant between light- and heavy-labeled samples.

In brief, Cerebrospinal fluid (CSF) collected from healthy individuals was purchased from Analytical Biological Services, Inc. (Wilmington, DE), and processed using protocols described previously (11), including depletion of the 7 top most abundant proteins (i.e. albumin, IgG, IgA, transferring, haptoglobin, antitrypsin, and fibrinogen) using a Multiple Affinity Removal LC Column from Agilent (Santa Clara, CA). The sample was then divided in two, with one half labeled with ^{12}C-acrylamide (light label) and the other half labeled with ^{13}C-acrylamide (heavy label)(8). Light and heavy acrylamide-labeled samples were then recombined and intact proteins were then fractionated into 17 different fractions using reversed phase methods(11), digested, and then submitted to interrogation on an LTQ-Orbitrap.

The spectra were acquired in a data-dependent mode in *m*/*z* range of 400−1800, with selection of the five most abundant +2 or +3 ions of each MS spectrum for MS/MS analysis. Raw data files were converted to mzXML format using ReAdW 1.1 and Xcalibur 2.2. All mzXML files were searched using X! Tandem (2007.01.01) with an alternative scoring plugin(12) compatible with PeptideProphet. Searches were conducted against the human International Protein Index database (IPI Human v3.48) plus common contaminants. All searches used the following parameters: ±1.5 Da precursor mass error, tryptic cleavage with up to two missed cleavage sites, static modification of 71.0366 Da (light acrylamide) on cysteine, potential modifications of 74.0466 Da (^{13}C acrylamide) on cysteine and 15.9949 Da (oxidation) on methionine. Peptide assignments were evaluated using PeptideProphet(9). Quantitative ratios between light and heavy labels were determined using Q3(8), an algorithm specifically designed to accommodate the 3-Da mass difference for singly labeled peptides, and ratio information was added to the existing PepXML files. We used msInspect platform tools to remove all identified peptides with probability less than 0.95, and ran ProteinProphet(13) to determine the proteins present in the experiment. We combined all peptide-level quantitation information for each identified protein in order to determine abundance ratios. In this case ProteinProphet was used only as a protein-inference tool, without regard to the confidence values assigned to proteins, since only high-confidence peptide identifications were used for protein inference

RESULTS

Use of Platform Utilities

All told, the Qurate application contains approximately 6,000 lines of application-specific code; it took less than 40 personnel hours to write and to test. It depends on roughly 20,000 lines of code from the msInspect platform and, indirectly, on roughly 3,500 lines of code from the TPP (for parsing mzXML files), which together represent a much larger investment of development time (several developer-months).

Qurate Self-Self Comparison

Our self-self quantitative dataset contained a total of 227 high-confidence, quantified proteins (with peptide identifications having probability greater than 0.95). Of these, 12 had quantitative ratios greater than 1.3. To mimic the process of selecting differential candidates, we selected 50 quantitative events (or groups of equivalent events, as described above) from these 12 proteins with high light-to-heavy ratios; we also selected 50 quantitative events to serve as negative controls, from proteins with ratios close to parity (0.9−1.1). We randomized the order of these ratios and presented them for evaluation to a data analyst who was blinded to whether the proteins were up-regulated or close to parity, and to the fact that the data resulted from a self-self comparison.

Qurate was used to discover quantitative events that appeared, based on visual inspection, to be of low or questionable quality. A single data analyst was able to visually curate these 100

quantitative events in less than 1 hour and to identify 35 quantitative events as "bad" (7 of the 50 events from the proteins with ratios near parity, and 28 of the 50 events from proteins with extreme ratios). See Table 1 for a summary, and see supplementary material for figures that provide representative examples of each type of flagged event. Note that some events were identified as having multiple issues, so the total of the numbers in the table exceeds the number of bad events. 6 events from near-parity proteins were marked as "unknown", and the remaining 37 were marked as "good"; 18 events from extreme-ratio proteins were marked as "unknown", and the remaining 4 were marked as "good". The distinction between "bad" and "unknown" is, of course, subjective, and where to draw the line between removing and retaining an event may depend on the goals of the individual project.

Among the "bad" events, the most common issue identified (23 from extreme events and 4 from the parity events) includes all examples in which the heavy or light masses presented isotopic distributions significantly different from the expected distribution given peptide mass, and/or with very different peak distributions between light and heavy versions. The next most frequent issue (8 times among 50 extreme events, no events among the controls) stems from observing high-quality ions for one member of the pair, but with a completely absent labeled mate. This may happen, for example, with an incorrect identification, or in real experiments when a peptide is observed in one condition but not the other. This condition is not flagged as bad, but rather flagged as questionable and requiring further investigation to determine the nature of the issue. In this instance, because we are evaluating a 1:1 experiment, we can assume that these likely result from either contamination or a bad identification. The third most frequent event (6 of 50 extreme ratios and 2 of 50 ratios near parity) stems from the visible isotopic peaks beginning at the theoretical *m/z* value of the C13 peak of the identified peptide, which is an indication of a potential bad identification. Other categories which result in a flagged event include several events in which the quantitation algorithm used only a small number of scans for a peptide that eluted over more scans; in which a coeluting ion affected light or heavy ion abundance; or in which signal intensity is so low relative to the nearby noise level that quantitation is not confident.

After manual curation, a new PepXML file was generated automatically, without those quantitative events and peptide identifications marked as described. In practice one may wish to exclude all or only some of these categories. For example, in a real comparative proteomics experiment one may opt not to exclude group 2, as this may eliminate the highest quality differences, even if it will also admit several potential sequencing errors. ProteinProphet was run on this new PepXML file to generate new protein-level quantitative ratios from the peptide ratios that remained after curation.

DISCUSSION

The purpose of this manuscript is to introduce the structure of the toolkit underlying the msInspect platform and to demonstrate its capabilities by creating a tool with practical utility in proteomics data analysis. Technical details on the software structure appropriate for software engineers are provided in supplementary materials, and in the open-source software distribution and developer's guide available on our website, proteomics.fhcrc.org.

We demonstrate the savings in application development time and effort provided by the platform by examining the development effort required to build Qurate, an application built with the platform that facilitates visual curation of isotopically labeled quantitation events. Qurate has been used within our lab to assess results from highly complex experiments, including a set of 10 related experiments, each containing up to 96 individual LC-MS/MS runs. We do not claim that the functionality provided by Qurate is novel, as some or all the functionality provided by Qurate may be available elsewhere. Rather, it is a demonstration of

the platform's ability to help create useful applications quickly, without dependencies on proprietary software. The design and development of Qurate, including several revisions, consumed less than 5 days of total personnel time (developer and data analyst), thanks to the extensive use of software tools already available in the platform. Qurate is documented on our website, and a tutorial dataset is provided.

Qurate was designed to provide a means to visually curate quantitative MS/MS experiments, particularly to control false discovery rates in biomarker screening applications. In this workflow researchers typically perform comparative experiments and select the most extreme proteins for further characterization. Whenever one compares two complex samples in which most proteins do not change between conditions, one can expect that the extreme ratios will include many false positives due to errors in signal processing, including database matching and quantitation of ion signal intensities. Because many errors are easily identified by visual (human) inspection, Qurate was developed to allow the rapid visual curation of an MS/MS experiment so that the easily identifiable errors can be eliminated from consideration. In our example of curating a self-self comparison, a data analyst was able to curate 100 quantitative events in less than one hour, which is not burdensome compared to the duration and investment in upstream or downstream efforts in proteomics analysis.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

ACKNOWLEDGMENTS

Financial support provided by National Cancer Institute grants U01 CA111273 and U54 CA119367, the National Heart, Lung and Blood Institute grant BAA-NHLBI-WH-09-01, by Department of Defense grant W81XWH-06-1-0100, and by the Canary Foundation.

REFERENCES

1. Keller A, Eng JK, Zhang N, Li XJ, Aebersold R. Molecular Systems Biology EPub. 2005
2. Kessner D, Chambers M, Burke R, Agus D, Mallick P. Bioinformatics 2008;24:2534–2536. [PubMed: 18606607]
3. Bellew M, Coram M, Fitzgibbon M, Igra M, Randolph T, Wang P, May D, Eng J, Fang R, Lin C, Chen J, Goodlett D, Whiteaker J, Paulovich A, McIntosh M. Bioinformatics 2006;22:1902–9. [PubMed: 16766559]
4. May D, Fitzgibbon M, Liu Y, Holzman T, Eng J, Kemp CJ, Whiteaker J, Paulovich A, McIntosh M. J Proteome Res 2007;6:2685–2694. [PubMed: 17559252]
5. May D, Liu Y, Law W, Fitzgibbon M, Wang H, Hanash S, McIntosh M. J Proteome Res. 2008
6. Martin D, Holzman T, May D, Peterson A, Eastham A, Eng J, McIntosh M. Mol Cell Proteomics 2008;7:2270–8. [PubMed: 18641041]
7. Brusniak M, Bodenmiller B, Campbell D, Cooke K, Eddes JS, Garbutt A, Lau H, Letartel S, Mueller LN, Sharma V, Vitek O, Zhang N, Aebersold R, Watts J. BMC Bioinformatics 2008;9:542. [PubMed: 19087345]
8. Faca V, Coram M, Phanstiel D, Glukhova V, Zhang Q, Fitzgibbon M, McIntosh M, Hanash S. J Proteome Res 2006;5:2009–18. [PubMed: 16889424]
9. Keller A, Nesvizhskii AI, Kolker E, Aebersold R. Anal Chem 2002;74:5383–92. [PubMed: 12403597]
10. Han DK, Eng J, Zhou H, Aebersold R. Nat Biotechnol 2001;19:946–951. [PubMed: 11581660]
11. Fang R, Elias DA, Monroe ME, Shen Y, McIntosh M, Wang P, Goddard CD, Callister SJ, Moore RJ, Gorby YA, Adkins JN, Fredrickson JF, Lipton MS, Smith RD. Mol Cell Proteomics 2006;5:714–725. [PubMed: 16401633]
12. MacLean B, Eng J, Beavis RC, McIntosh M. Bioinformatics 2006;22:2830–2. [PubMed: 16877754]
13. Nesvizhskii AI, Keller A, Kolker E, Aebersold R. Anal Chem 2003;75:4646–58. [PubMed: 14632076]

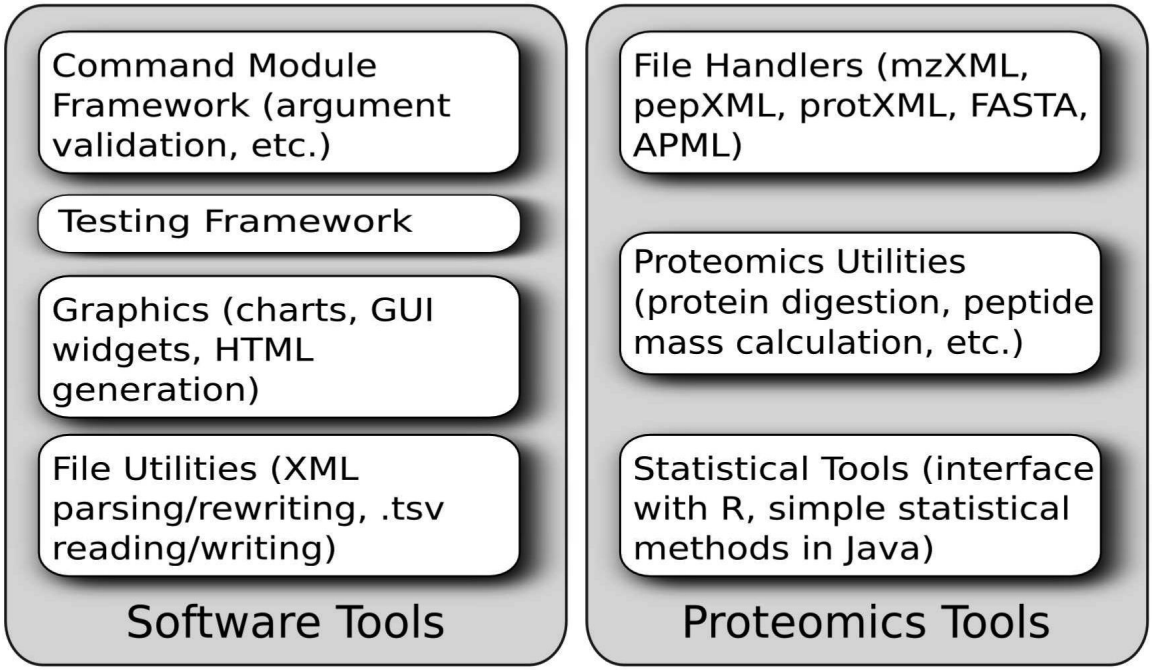

Figure 1.
A summary of the types of tools provided in the msInspect platform.

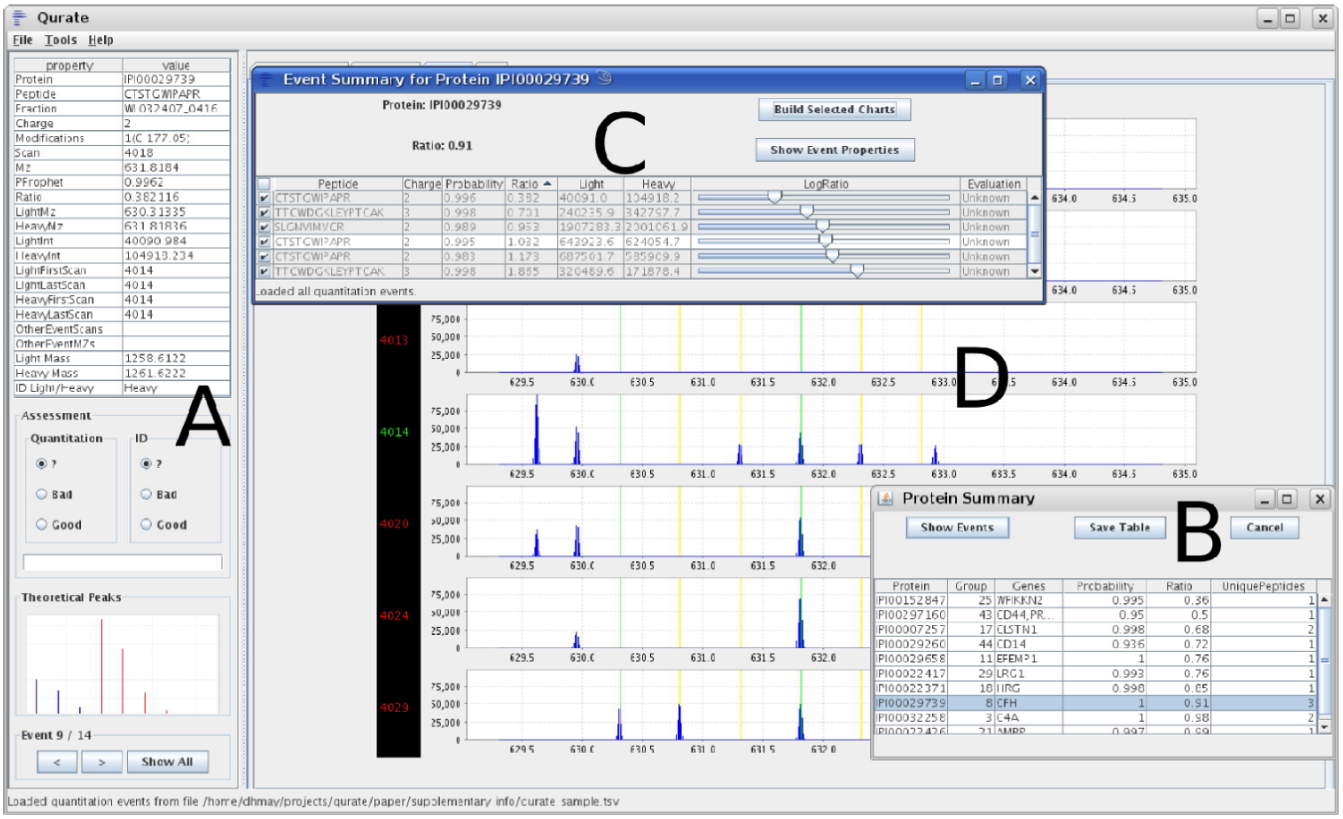

Figure 2.

The Qurate interface. A) Quantitative event information, with controls for curation and navigation, and the theoretical isotopic distribution. B) Summary information about all confidently identified, quantitated proteins. C) All quantitative events for a single protein; quantitative ratios are provided numerically and also represented, in log space, by a slider. D) A chart showing intensities near the quantitative event separately for each scan, with green and yellow lines indicating theoretical monoisotopic and other peaks; only the scan number in green is used by the quantitation algorithm for this event, meaning that the light peaks visible in the bottom scan were not used in ratio calculation.

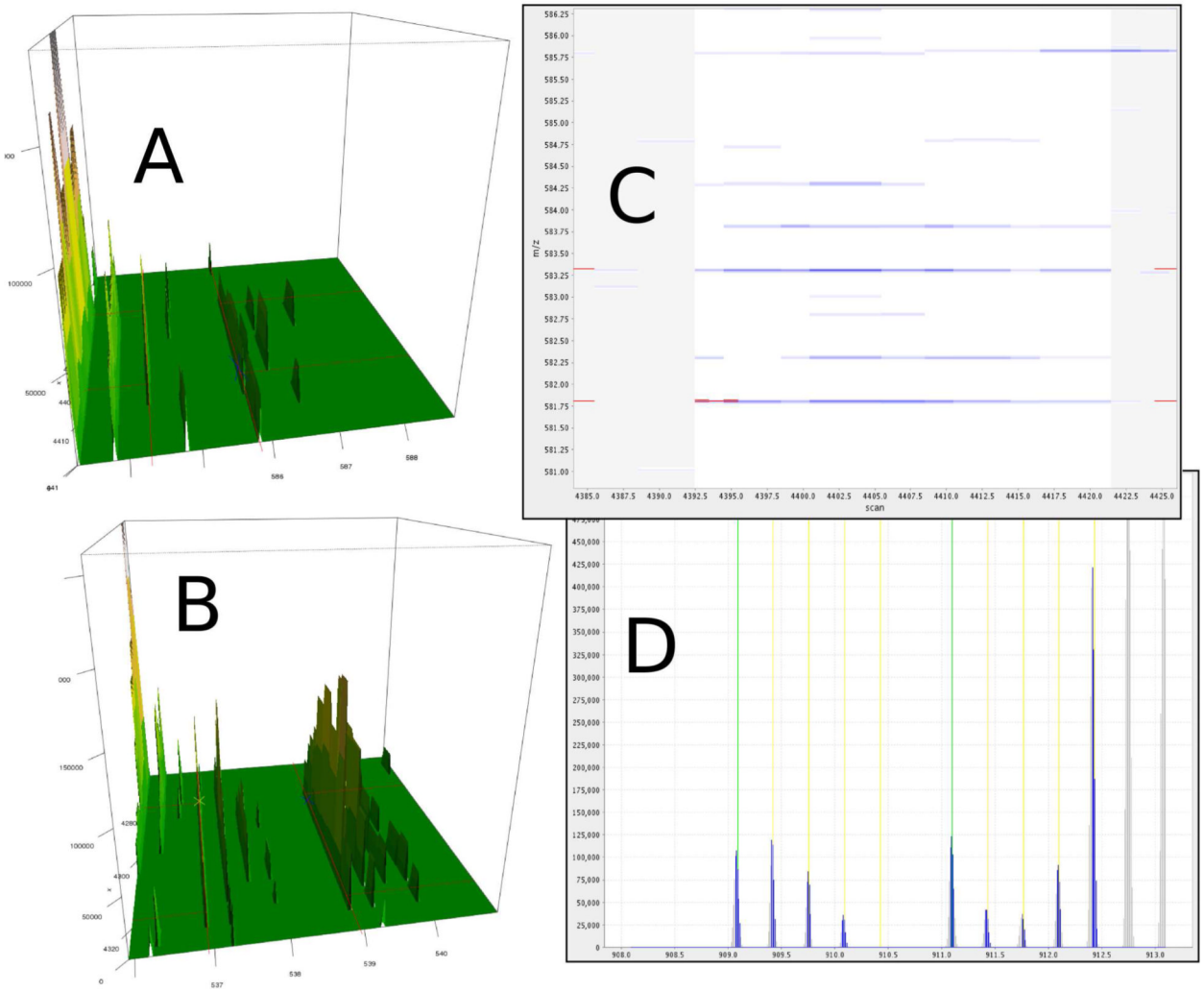

Figure 3.

Qurate charts for quantitative events. Different charts are best suited to evaluating different events. A) A 3D contour plot. Horizontal axis is *m/z*, and depth axis is retention time. Solid red lines indicate light and heavy monoisotope. Dashed lines indicate the first and last scans used by Q3, and X marks indicate peptide identification events. In this event, a coeluting peptide at lower *m/z* adds to light isotope intensity. B) In this event, a nearby peptide at lower *m/z*, eluting slightly earlier, appears to have minimal effect on quantitation. C) A heatmap showing a two-dimensional region of spectra around the quantitative event. Horizontal axis is retention time, vertical is *m/z*. Blue indicates high intensity. Gray indicates area not used for quantitation. Red tick marks indicate light and heavy monoisotopic *m/z*. Red mark indicates the position of the identified peptide. D) A plot of intensities summed over the entire range of quantitation. Horizontal axis is *m/z*, vertical is intensity. Green and yellow lines indicate monoisotopic and other peaks. Blue intensities are within the mass range around each peak used by Q3, gray are not. A coeluting peptide artificially inflates the fourth and fifth peaks of the heavy isotope.

Table 1

lists the various reasons that the events were marked as ‘Bad’ (some extreme-ratio events were marked as ‘Bad’ for multiple reasons). The Supplementary Materials contain example charts for each type of ‘Bad’ event.

Reason for Indicating ‘Bad’ Quantitative Event	Count (Extreme Ratios)	Count (Close Ratios)
1 Questionable isotopic distributions	28	4
2 Only light ion identified	8	
3 Identified C13 peak	6	2
4 Long-eluting ion quantitated based on too few scans	2	
5 Coeluting peptides affect quantitation	1	1
6 Too low intensity relative to noise.	1	