OpenMS: A flexible open-source software platform for mass spectrometry data analysis

Hannes L. Röst[*, 1, 2] Timo Sachsenberg[*, 3, 4] Stephan Aiche[*†, 5] Chris Bielow[*, 6, 7] Hendrik Weisser[*, 8] Fabian Aicheler[3, 4] Sandro Andreotti[5] Hans-Christian Ehrlich[†, 5] Petra Gutenbrunner[8] Erhan Kenar[3, 4, 9] Xiao Liang[10] Sven Nahnsen[9] Lars Nilse[11] Julianus Pfeuffer[3, 4] George Rosenberger[1] Marc Rurik[3, 4] Uwe Schmitt[13] Johannes Veit[3, 4] Mathias Walzer[3, 4] David Wojnar[9] Witold E. Wolski[1, 14] Oliver Schilling[11, 15] Jyoti S. Choudhary[8] Lars Malmström[1, 16] Ruedi Aebersold[1, 17] Knut Reinert[5, 18] and Oliver Kohlbacher[3, 4, 9, 19, 20]

1 Department of Biology, Institute of Molecular Systems Biology, ETH Zurich, Zurich, Switzerland

2 Department of Genetics, Stanford University, Stanford, CA, USA

3 Dept. of Computer Science, University of Tübingen, Tübingen, Germany

4 Center for Bioinformatics, University of Tübingen, Tübingen, Germany

5 Department of Mathematics and Computer Science, Freie Universität Berlin, Berlin, Germany

6 Berlin Institute for Medical Systems Biology,
Max-Delbrück-Center for Molecular Medicine, Berlin, Germany

7 Berlin Institute of Health, Berlin, Germany

8 Wellcome Trust Sanger Institute, Hinxton, UK

9 Quantitative Biology Center, University of Tübingen, Tübingen, Germany

10 International Max Planck Research School for Computational Biology and Scientific Computing (IMPRS-CBSC)

11 Institute of Molecular Medicine and Cell Research, University of Freiburg, Freiburg, Germany

13 ID Scientific IT Services, ETH Zurich, Zurich, Switzerland

14 Functional Genomics Center Zurich, ETH Zurich, Zurich, Switzerland

15 BIOSS Centre for Biological Signalling Studies, University of Freiburg, Freiburg, Germany

16 S3IT, University of Zurich, Zurich, Switzerland

17 Faculty of Science, University of Zurich, CH-8057 Zurich, Switzerland

18 Max Planck Institute for Molecular Genetics, Berlin, Germany

19 Faculty of Medicine, University of Tübingen, Tübingen, Germany

20 Max Planck Institute for Developmental Biology, Tübingen, Germany

∗ These authors contributed equally

† Present Address: SAP SE, Konrad-Zuse-Ring 10, 14469 Potsdam, Germany

Editorial summary:
OpenMS is a flexible, user-friendly, open-source software platform for the biological analysis of mass spectrometry proteomics and metabolomics data. The modular platform allows developers to seamlessly generate custom data analysis workflows and directly make such ready-made workflows available to biologist end-users.

High-resolution mass spectrometry (MS) has become an important tool in the life sciences, contributing to the diagnosis and understanding of human diseases, elucidating structural information and characterizing cellular signaling networks. However, the rapid growth in the volume and complexity of mass spectrometry data makes transparent, accurate and reproducible analysis difficult. We present OpenMS 2.0 (http://www.openms.org), a robust open-source, cross-platform software specifically designed for the flexible and reproducible analysis of high-throughput MS data. The extensible OpenMS software implements common mass spectrometric data processing tasks through a well-defined API in C++ and Python using standardized open data formats. OpenMS additionally provides a set of 185 tools and ready-made workflow to carry out common mass spectrometric data processing tasks, which enable users to perform complex quantitative mass spectrometric analyses with ease.

In the field of high-throughput mass spectrometry, transparent and reproducible data analysis has traditionally been challenging due to rapidly evolving technology, a highly heterogeneous software landscape and multi-faceted analysis workflows that have to be tailored to a specific set of samples or experimental conditions. Mass spectrometry is the major technique applied in several fields, including metabolomics, proteomics, interactomics and lipidomics, each of which requires substantially different approaches to data acquisition and analysis. Furthermore, multiple separation methods (gas chromatography, liquid chromatography), different fragmentation methods (CID, ECD, ETD, HCD, etc.) and acquisition strategies (data-dependent, data-independent or targeted) are used in a bewildering range of combinations. For quantification, different label-free, isobaric or isotopic labelling strategies are available (e.g., ICAT, SILAC, iTRAQ, TMT for proteomics). Finally, the data analysis step may include a database search (as in proteomics and metabolomics), spectral library search, or targeted analysis. This flexibility usually calls for complex, multi-step analysis workflows with inter-dependent steps that need to be tailored to the experiment at hand. For example, in a discovery proteomics pipeline, the workflow steps include file conversion, centroiding, database searching, feature detection, retention time alignment, cross-run feature linking, protein inference, and statistical error control at various steps of the process [1].

This diversity in data analysis strategies leads to multiple challenges for the computational analysis of MS data. First, multiple different data formats are used in the field to describe the raw data and the output of individual steps of the analysis (identification, quantification, RT alignment, *etc.*). Second, a multitude of software packages have been developed that may perform one or several steps of a complex analysis workflow, and often rely on custom code for input, output and data pre-processing steps (such as centroiding, smoothing or isotopic deconvolution). Such re-implementation of common algorithms is both time-consuming and error-prone, and it can lead to substantial differences in the results between different software tools. Third, the individual tools are generally difficult to integrate into larger workflow schemes on a single computing platform.

Past efforts to mitigate these issues have led to the development of standardized data exchange formats [2, 3, 4, 5], which have recently been adopted by several software projects [6, 7, 8, 9, 10]. These standard formats enable the integration of tools from different sources, simplify the analysis of MS data from multiple vendors, and render published results more readily accessible. They permit access to publicly deposited data without the need for proprietary software for conversion and inspection of the data, thus forming a cornerstone of open science. Yet, proteomics studies and practitioners in the field are still plagued by the difficulty of reproducing results from other labs and integrating individual tools into a workflow – a task that often needs to be started from scratch for a new experiment. Maintaining and documenting computational workflows which often requires a high level of computational knowledge. However, for transparent and reproducible data analysis, the full analysis workflow including all parameters used by the tools needs to be extensively documented and easy to set up in a new lab [11]. Such a complete workflow then becomes the equivalent of a standard operating procedure (SOP), a "computational SOP" capturing each aspect of the data analysis in a digital and reproducible form. This can reduce the potential for errors in data analysis and make complex analyses more accessible to new users [12].

The OpenMS project addresses these challenges by providing a highly flexible and professional software environment equally suited for end users and for software developers. To this end, it provides different levels of access to the software: (i) pre-assembled workflows for standard tasks provide best-practice "recipes" implementing widely used types of analyses from within a powerful graphical analysis workbench or from commercial workflow tools (e.g., Proteome Discoverer, Compound Discoverer or BaseSpace); (ii) individual tools, over 180 altogether, can be used alone or together with a number of other compatible open-source tools; and (iii) a powerful algorithmic core library and a Python scripting interface provide an interface for software developers to implement novel algorithms. Since the publication of the initial C++ software library in 2008, the OpenMS project has added algorithms for high-resolution shotgun proteomics, targeted proteomics, isotopic labeling, cross-linking, metaproteomics, proteogenomics, and metabolomics. With the new release of OpenMS 2.0 (http://www.openms.org/download/openms-binaries/), we have extended its functionality and provide a large number of pre-built user-oriented tools and workflows for reproducible MS data analysis (see **Supplementary Note 1**). We have also fully integrated OpenMS with graphical workflow systems (such as KNIME [13], Galaxy [14], Proteome Discoverer, or Compound Discoverer [15]) making it it more easily accessible to experimental life scientists and more powerful in the hands of bioinformaticians. For a guide to getting started, see **Box 1**.

OpenMS for bioinformaticians

OpenMS is written in C++ and offers native compiler support on Windows (MSVS), Linux (Clang and GCC) and OS X (Clang) as well as an interface to the Python scripting language for direct access to core data structures and algorithms. OpenMS uses modern software engineering concepts with an emphasis on modularity, reusability, and extensive testing using continuous integration (**Supplementary Note 1**). The permissive BSD license encourages usage in commercial and academic environments, facilitating collaborations and joint research.

Architecture

The multi-level architecture of the OpenMS framework (**Fig. 3)** enables interaction and extension on different levels: The user-generated workflows consists of individual executable tools (TOPP tools), which themselves use the algorithms from the core C++ library. This design allows the generic algorithms from the library to be easily combined into powerful, high-performance analysis tools and complete analysis workflows.

Each layer only depends on the functionality of the previous layer and provides increased utility through the combination of powerful individual components.

The lowest layer, the data and signal processing layer, is built using a set of open-source libraries for numerical computing, XML parsing, sequence analysis [16] and statistics. It provides a unified access point to mass spectrometric data formats and common pre-processing algorithms (**Fig. 3**). OpenMS provides implementations for over 30 file formats, including the current and upcoming open standards defined by the Proteomics Standards Initiative of the Human Proteome Organization (HUPO-PSI) such as mzML, TraML, mzIdentML, mzQuantML, mzTab and qcML [2, 3, 4, 5, 17, 18] (see **Supplementary Note 1**). In addition, OpenMS includes standard algorithms related to biomolecular properties (such as mass calculation, isotopic composition, protein digestion *etc*.) and signal processing (such as smoothing, baseline-correction, de-isotoping and peak-picking). This allows developers to use a robust library as a foundation for new algorithms without having to spend time on re-implementing basic functionality. Even more importantly, the availability of validated software libraries reduces errors since the code is maintained at a single location and is tested by multiple developers who are likely to notice and correct errors. For users, this means more reliable code that behaves identically and predictably across multiple software tools and computing platforms.OpenMS also provides implementations to address the most common quantitative proteomics and metabolomics tasks – including quantification, identification, and visualization (see **Fig. 3**), as well as algorithms for isotopic deconvolution, chromatographic peak picking, retention time alignment, or feature linking across multiple runs. Developers can directly benchmark new algorithms against these reference implementations, enabling meaningful comparisons to prior algorithms. Comparisons are also supported through the free availability of several “gold standard” datasets used for the OpenMS algorithm development. Using the underlying data access and signal processing routines, new algorithms can be readily added to existing pipelines by enhancing a given tool or simply exchanging an existing step with a new tool. For life science labs using OpenMS, this means fast access to improved algorithms with minimal disruption of existing workflows.

The core OpenMS library has been used to implement a wide range of user-oriented tools and workflows (**Fig. 3**). These so-called TOPP tools are individual executables implemented using the `TOPPBase` base class and generally provide access to a specific functionality (SWATH analysis, RNA cross-linking analysis, SILAC quantification, *etc.*; see **Supplementary Note 1**). TOPP tools either provide functionality by using the set of algorithms available in OpenMS or they facilitate the usage of external tools (using so-called adaptor tools). Most TOPP tools are also automatically integrated into the viewer TOPPView, from which non-developers can apply it directly to their data from a graphical user interface. Furthermore, the individual TOPP tools are automatically available as nodes in workflow managers such as KNIME or Galaxy, allowing users to integrate them into their workflows and thus profit from the development of new algorithmic approaches.

Scripting interface

For rapid software prototyping OpenMS provides direct access to all data structures and algorithms in the Python programming language [19]. The pyOpenMS interface offers even users with basic programming skills an extraordinary amount of flexibility. They can directly interact with the OpenMS core library and implement algorithms in Python using the same (and efficient) data structures as in C++. This is especially attractive for researchers that are not familiar with low-level languages but still want to have access to algorithms provided in OpenMS. Python is also ideally suited for rapid software prototyping, allowing developers to quickly test new algorithmic ideas and add them to the C++ core once refined. The OpenMS architecture makes it easy to port algorithms back to C++. Finally, the Python framework has also been used to build stand-alone applications that use OpenMS for heavy data processing (see for example eMZed or TAPIR [20, 21]).

By writing code directly in Python, researchers can benefit from a substantial amount of available bioinformatics packages and algorithms outside mass spectrometry. This code is generally accessible through individual packages, which can be imported into existing code. Functions range from numerics and signal processing to sequencing-based analysis and structural analysis. By adding the full OpenMS functionality as a package to Python, the community gets the ability to mix and match these different software packages. We recently used pyOpenMS together with PyMOL [22] to extract identified peptides from an MS experiment and plot these peptides on the 3D structure of the respective protein using less than 50 lines of Python code [19].

Community-driven development

Software engineering, testing and distribution of executables to users are nontrivial tasks that can best be managed by a team of developers. OpenMS development is performed as a team effort with individual maintenance tasks distributed across all developers. Active online discussions within the international team, enforced code reviews, high test coverage and continuous integration ensure high code quality and promote collaborations. Nightly builds for multiple platforms and frequent releases allow new code to be distributed quickly to users. The tight integration of OpenMS executables with workflow engines means that users can receive new tools directly as nodes inside their favorite environment. Developers profits from the centralized infrastructure that generates these nodes across multiple platforms for testing and development.

The OpenMS project is one of multiple software solutions addressing the computational challenges of mass spectrometry-based proteomics, each contributing in different areas such as protein inference, peptide database search or file conversion [6, 7, 23, 24]. While some proprietary software packages offer solutions for specialized use cases, their closed nature makes their adaption to new requirements and integration into more complex workflows difficult. In contrast, OpenMS integrates multiple open-source projects. Users as well as developers are invited to contribute in multiple ways (see **Box 2**).

Code reuse and extensibility

The option to reuse code and rely on well-tested and robust libraries is a major advantage inherent to software frameworks such as OpenMS. Writing high-quality code that is robust, correct, and efficient is time-consuming and often requires years of testing with real-world data to identify possible edge-cases. OpenMS has a long track record in the field and a large community of experienced developers working on the core library. In OpenMS, the code is organized into individual classes and modules that each implement a specific concept, file format or algorithm, and each module can be used independently and behaves according to the principle of least surprise. This allows developers to incorporate OpenMS modules in their C++ and Python code instead of writing their own, thus saving time and avoiding possible errors. The extensive testing performed in OpenMS ensures that the code will run on all supported platforms and behave identically even in future releases.

OpenMS for experimentalists

OpenMS tools and workflows

OpenMS offers a selection of over 180 different tools for performing tasks of varying complexity (from file conversion and data extraction to quantification, RT alignment, database search *etc.*; see **Supplementary Note 2**). Each of these tools offers a standardized interface and common file formats to read input and write output. Tools can be used individually from the command line, or invoked from within an analysis workflow consisting of a defined sequence of pre-configured tools (**Fig. 1**). By combining different tools in various combinations and with different parameters, OpenMS can cover a wide range of possible experimental applications (see the tutorial in **Supplementary Protocol 1**). Ideally, these workflows start directly with the conversion of the initial raw data and create a complete analysis of the data set ready for deposition in a public repository (as required by most publishers).

While pre-configured workflows based on OpenMS are available for many standard analyses (see **Supplementary Note 2**) the simple creation of new workflows is a strength of OpenMS. Constructing these workflows does not require any programming expertise and can be done through graphical user interfaces in integrated workflow environments (e.g., KNIME, Galaxy and others [25, 14, 26, 13, 27]). These workflow managers allow even inexperienced users to modify and extend existing data analysis workflows within minutes. User-created workflows can then be shared via the OpenMS website [26] or via public workflow repositories (e.g., http://www.myexperiment.org), allowing other labs to reproduce workflows simply by downloading a single file and opening it in a workflow editor. The workflow file specifies all the input files, the tools to be executed and parameters for each tool, thus rendering every detail of the analysis fully transparent and reproducible. Note that these workflows can also be executed on larger data sets without the graphical user interface (see the "OpenMS for Bioinformaticians" section below). The integration of external software such as ProteoWizard [28] (for file conversion), OMSSA [29], X! Tandem [23], MSGF+ [30] and Percolator [31] (for database search) or Fido [32] (for protein inference) allows researchers to perform all processing in a single workflow environment.

Visualization

The OpenMS viewer application TOPPView provides advanced visualization of MS data. It is capable of displaying one-dimensional representations of the data (spectra, chromatograms), but can also render the data in two- and three-dimensional representations (intensities in *m/z* and RT dimension). A layer-based system known from many graphics programs permits the superposition of different data sets as layers, for example for the comparison of multiple data sets or for the comparison of raw data and computed quantification results (see **Fig. 1**). Users can interact with the data in multiple ways, zooming in to explore their raw data down to the spectrum level or extracting ion chromatograms (see the tutorial in **Supplementary Protocol 2**). Tight integration with the numerous OpenMS executables enables users to directly apply OpenMS algorithms to their data and observe their effect. The tools are applied in the background so that the user can continue working. This direct feedback on the visible subset of the data allows users to fine-tune parameters and understand the effect of the data processing intuitively. Selected OpenMS tools (e.g., label-free quantification for proteomics, RNA-protein cross-linking, non-targeted metabolite quantification) have also been integrated into commercial workflow engines and analysis suites (Proteome Discoverer and Compound Discoverer [15]). This integration brings the OpenMS tools into a platform many users are already familiar with. It also enables a seamless integration with vendor-built visualization of the MS raw data.

Reproducible research The complexity of large-scale omics analyses means that reproduction of the computational approach is only possible if a complete and exhaustive documentation of all tools and their settings is provided. OpenMS is designed to support reproducible research by allowing users to store workflows in easily transferable workflow files which automatically contain complete records of all tools, their version numbers and parameters used. This means that users can obtain the exact workflow applied in a published manuscript, analyze it in detail and run it on their own data. The comprehensive integration of OpenMS with external tools performing data conversion, database searching, error rate estimation and protein inference ensures that the parameters of all steps are documented and there are no "missing pieces" in the description of the analysis. The results can then be visualized in TOPPView. Once a new analysis pipeline has been established in a lab, researchers can upload the workflow file to the OpenMS website or to another repository, thus documenting their work and enabling other researchers to reproduce their analysis (see **Fig. 1**).

Case studies Some recent examples where researchers have used OpenMS to build complex analysis tools include improvements to a label-free quantification pipeline[1], the development of a metabolomics workflow [33], the integration of RNA cross-linking workflows [34] and the addition of a workflow for targeted data analysis using SWATH-MS [35]. In addition, OpenMS has been extended to include a probabilistic scoring engine [36], SILAC analysis[37], isobaric quantification, random-access XML file parsing [38] and an MS simulation framework [39]. These examples illustrate the flexibility and versatility of the core framework.

The following three case studies highlight the extensibility of the OpenMS core library, the power of using the sensitive OpenMS feature detection algorithms for quantification and the application of OpenMS workflows to large public datasets (see also **Supplementary Note 3**).

SWATH data analysis The degree of variation in human plasma protein abundance and the underlying causes of such variation are largely unknown. Large-scale proteomics measurements of human blood plasma provide in-depth information about inter- and intra-person variability of protein abundances and can be used to investigate longitudinal trends during aging. In a recent study, Liu et al. [40] used SWATH-MS to acquire over 200 blood plasma samples collected from monozygotic and dizygotic twins in a longitudinal study (**Fig. 2**). For maximal sensitivity, the authors chose a targeted analysis strategy for SWATH-MS data [41] which was developed previously using the OpenMS software library. In particular, the OpenSWATH analysis module [35] was developed using existing algorithms and data structures for file parsing, RT alignment and signal processing. The developers could thus rely on the well-tested OpenMS framework for core tasks and focus their efforts on the development of novel algorithms. Additionally, the extensibility of OpenMS meant that the new algorithms specific to targeted data analysis (such as targeted extraction, chromatographic peak detection and scoring) could be seamlessly integrated into the existing C++ library. The high sensitivity of OpenSWATH allowed Liu et al. to quantify over 300 plasma proteins and allowed them to decompose the observed variance for each protein into heritable, environmental (common and individual) and longitudinal components (**Fig. 2**). Interestingly, the authors found that the heritable component of plasma protein variation ranged from over 60 % to almost zero. Furthermore, the integration of the proteomics data with individual genomes led to the identification of 13 cis-pQTL, two of which were associated with human disease.

Degradomics Human proteases are a large class of diverse proteins implicated in a range of diseases, but their study is analytically challenging. In a recent study, Lai et al. [42] used quantitative proteomics and stable isotope labeling to identify proteolytically generated protein neo-termini from formalin-fixed, paraffin-embedded (FFPE) specimens (which represent the vast majority of samples stored in clinical archives) using biochemical enrichment methods. OpenMS enabled the parallel and unbiased analysis of multiple, prevalent N-terminal modifications in different, isotopically labeled states. OpenMS also enabled accurate relative quantification of stable-isotope-labeled peptides. This included addressing so-called “knock-out” situations, that is, when a peptide occurs in only one isotope state and is missing in the corresponding second isotope channel. Typically, this situation is prone to distorted quantification due to chromatographic background signals [43]; a problem that can be resolved by accurate peptide feature detection in OpenMS. The workflow characterized thousands of protein N-termini in FFPE specimens and identified 23 novel potential substrates for the lysosomal protease cathepsin L [42].

Proteogenomics Wright et al. [44] used publicly available mass spectrometric data to refine the annotation of protein-coding regions in the human genome using proteomics data. The purpose of the study was to identify novel protein-coding genes, improve evidence for existing gene models and develop a stringent workflow for data analysis. Wright et al. used OpenMS to to implement a complete proteogenomic analysis workflow, which encompasses database searches using multiple search engines, rescoring and combining of search results, filtering according to various criteria on PSM, peptide and protein levels, and data export – all within OpenMS (Weisser et al. *in preparation*). Processing of over 55 million tandem mass spectra from different publicly available datasets using OpenMS led them to discover over 40 new protein-coding gene annotations in the GENCODE human reference genome [45]. In addition to the identification of novel genes, the information gained in a proteogenomic analysis pertaining to known proteins can also be utilized, for example, to elucidate tissue-specific protein expression levels [46].

Conclusion and future developments

The OpenMS project is a community-driven project backed by several major labs. Over the last decade, OpenMS has built a large community of users and developers alike. The high code quality and long-term maintenance of the project render it a good solution for sustainable software development. Since its initial publication, training sessions at leading conferences and over eight specialized user meetings have been held to educate users and disseminate knowledge of established workflows to practitioners in the field. These meetings provide opportunities for users and developers to meet and exchange ideas while describing the newest algorithms and advances incorporated into the OpenMS project every year. The OpenMS project not only sees high activity from its direct contributors, but several other open-source tools such as MSStats [47], aLFQ [48] and Skyline [49] have started to integrate their tools with OpenMS.

While OpenMS provides expert users and developers with great flexibility, the complexity of OpenMS can present a challenge to novice users. The OpenMS team is addressing this issue by extensive training activities and great efforts to simplify interfaces. OpenMS will also have to adapt to the changing landscape of computation, which is increasingly becoming heterogeneous and distributed. Workflow-driven solutions for cloud computing, including Docker-based virtualization are thus at the focus of current developments. At the same time, these efforts should not complicate analyses for end-users. The OpenMS team is thus also increasing the number of pre-built workflows with detailed explanation for a variety of use-cases and instrument types. By building a community of users sharing workflows and parameter settings using workflow files, best practices can be passed from lab to lab and stored alongside the raw data to make data analysis reproducible. The OpenMS team aims to implement further MS-based workflows (such as protein-protein cross-linking, lipidomics, small molecules) in order to meet the demands of its growing user base. While the current integration with workflow managers, vendor software and scripting languages already provides non-expert users with several options to access the OpenMS algorithms, we also aim to establish further collaborations with instrument vendors and developers of other open-source software, supporting emerging workflows in computational mass spectrometry and provide additional user-interfaces. Finally, the OpenMS community will continue to promote open-source software and open standards in the field of proteomics.

Availability OpenMS is open-source software and distributed under a BSD three-clause license. The most convenient way to explore OpenMS is through the KNIME analytics platform. Instructions for download and examples can be found on the OpenMS website under (http://www.openms.org/getting-started). Binary installers for all major platforms are available from the project website (http://www.openms.org/downloads). The project source code is hosted in its entirety on GitHub (https://github.com/OpenMS/OpenMS).

Competing financial interests

C.B. is a part-time employee of CodeMS which operates in the field covered in the article.

Acknowledgements We gratefully acknowledge the contributions of all OpenMS developers as well as of our users who helped to improve the software. H.L.R. was funded by ETH (ETH-30 11-2) and SNSF (P2EZP3_162268), G.R. was funded by the Swiss Federal Commission for Technology and Innovation CTI (13539.1 PFFLI-LS), R.A. was funded by ERC Proteomics v3.0 (233226), the PhosphonetX project of SystemsX.ch and the Swiss National Science Foundation. H.W. P.G. and J.S.C. were funded by Wellcome Trust grant (WT098051) to the Sanger Institute. X.L. acknowledges funding from German Academic Exchange Service (DAAD, grant no.57076385). K.R. and C.B. acknowledge funding from the European Union (FP7, Predict-IV, GA 202222). O.S. and L.N. acknowledge funding from the Deutsche Forschungsgemeinschaft (SCHI 871/5, SCHI 871/6, SCHI 871/8, SCHI 871/9, GR 1748/6, INST 39/900-1, and SFB850-Project B8) and the European Research Council (ERC-2011- StG 282111-ProteaSys). S.N., D.W., and O.K. acknowledge fundingfrom DFG (QBiC). M.W. and O.K. acknowledge funding from DFG (SFB685). M.R., F.A., S.A., K.R. and O.K. received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under EC-GA No. 263215 "MARINA". M.W., S.N., and O.K. acknowledge funding from the European Union (PRIME-XS). E.K. and O.K. acknowledge funding from BMBF (grant 01GI1104A). E.K., J.P., T.S., and O.K. acknowledge funding from BMBF (grant 01ZX1301F). F.A. and O.K. acknowledge funding from BMBF (grant 031A430C). J.V., L.N., and O.K. acknowledge funding from BMBF (grant 0315395B). O.K., K.R., S.A., J.P, and T.S. acknowledge funding from BMBF (grants 031A367 and 031A535A). We are deeply grateful for the support of the KNIME team, the Proteome Discoverer team, and the Compound Discoverer team.

References

[1] H. Weisser, S. Nahnsen, J. Grossmann, L. Nilse, A. Quandt, H. Brauer, M. Sturm, E. Kenar, O. Kohlbacher, R. Aebersold, and L. Malmström, "An automated pipeline for high-throughput label-free quantitative proteomics," *Journal of Proteome Research*, vol. 12, no. 4, pp. 1628–1644, 2013.

[2] L. Martens, M. Chambers, M. Sturm, D. Kessner, F. Levander, J. Shofstahl, W. H. Tang, A. Römpp, S. Neumann, A. D. Pizarro, L. Montecchi-Palazzi, N. Tasman, M. Coleman, F. Reisinger, P. Souda, H. Hermjakob, P.-A. A. Binz, and E. W. Deutsch, "mzML–a community standard for mass spectrometry data.," *Molecular & cellular proteomics : MCP*, vol. 10, no. 1, 2011.

[3] M. Walzer, D. Qi, G. Mayer, J. Uszkoreit, M. Eisenacher, T. Sachsenberg, F. F. Gonzalez-Galarza, J. Fan, C. Bessant, E. W. Deutsch, *et al.*, "The mzquantml data standard for mass spectrometry–based quantitative studies in proteomics," *Molecular & Cellular Proteomics*, vol. 12, no. 8, pp. 2332–2340, 2013.

[4] J. Griss, A. R. Jones, T. Sachsenberg, M. Walzer, L. Gatto, J. Hartler, G. G. Thallinger, R. M. Salek, C. Steinbeck, N. Neuhauser, *et al.*, "The mztab data exchange format: communicating mass-spectrometry-based proteomics and metabolomics experimental results to a wider audience," *Molecular*

& Cellular Proteomics, vol. 13, no. 10, pp. 2765–2775, 2014.

[5] A. R. Jones, M. Eisenacher, G. Mayer, O. Kohlbacher, J. Siepen, S. J. Hubbard, J. N. Selley, B. C. Searle, J. Shofstahl, S. L. Seymour, R. Julian, P. A. Binz, E. W. Deutsch, H. Hermjakob, F. Reisinger, J. Griss, J. A. Vizcaíno, M. Chambers, A. Pizarro, and D. Creasy, "The mzIdentML data standard for mass spectrometry-based proteomics results.," *Molecular & Cellular Proteomics*, vol. 11, p. M111.014381, 2012.

[6] E. W. Deutsch, L. Mendoza, D. Shteynberg, T. Farrah, H. Lam, N. Tasman, Z. Sun, E. Nilsson, B. Pratt, B. Prazen, J. K. Eng, D. B. Martin, A. I. Nesvizhskii, and R. Aebersold, "A guided tour of the trans-proteomic pipeline," *Proteomics*, vol. 10, no. 6, pp. 1150–1159, 2010.

[7] M. C. Chambers, B. Maclean, R. Burke, D. Amodei, D. L. Ruderman, S. Neumann, L. Gatto, B. Fischer, B. Pratt, J. Egertson, K. Hoff, D. Kessner, N. Tasman, N. Shulman, B. Frewen, T. A. Baker, M.-Y. Brusniak, C. Paulse, D. Creasy, L. Flashner, K. Kani, C. Moulding, S. L. Seymour, L. M. Nuwaysir, B. Lefebvre, F. Kuhlmann, J. Roark, P. Rainer, S. Detlev, T. Hemenway, A. Huhmer, J. Langridge, B. Connolly, T. Chadick, K. Holly, J. Eckels, E. W. Deutsch, R. L. Moritz, J. E. Katz, D. B. Agus, M. MacCoss, D. L. Tabb, and P. Mallick, "A cross-platform toolkit for mass spectrometry and proteomics," *Nature Biotechnology*, vol. 30, no. 10, pp. 918–920, 2012.

[8] M. Sturm, A. Bertsch, C. Gröpl, A. Hildebrandt, R. Hussong, E. Lange, N. Pfeifer, O. Schulz-Trieglaff, A. Zerck, K. Reinert, and O. Kohlbacher, "OpenMS - an open-source software framework for mass spectrometry.," *BMC Bioinformatics*, vol. 9, no. 1, p. 163, 2008.

[9] M. Vaudel, J. M. Burkhart, R. P. Zahedi, E. Oveland, F. S. Berven, A. Sickmann, L. Martens, and H. Barsnes, "Peptideshaker enables reanalysis of ms-derived proteomics data sets," *Nature biotechnology*, vol. 33, no. 1, pp. 22–24, 2015.

[10] R. Wang, A. Fabregat, D. Ríos, D. Ovelleiro, J. M. Foster, R. G. Côté, J. Griss, A. Csordas, Y. Perez-Riverol, F. Reisinger, *et al.*, "Pride inspector: a tool to visualize and validate ms proteomics data," *Nature biotechnology*, vol. 30, no. 2, pp. 135–137, 2012.

[11] Nature, "Devil in the details," *Nature*, vol. 470, pp. 305–306, 2011.

[12] Nature, "Code share," *Nature*, vol. 514, p. 336, 2014.

[13] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel, *KNIME: The Konstanz information miner*. Springer, 2008.

[14] J. Goecks, A. Nekrutenko, J. Taylor, *et al.*, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," *Genome Biol*, vol. 11, no. 8, p. R86, 2010.

[15] J. Veit, T. Sachsenberg, A. Chernev, F. Aicheler, H. Urlaub, and O. Kohlbacher, "LFQProfiler and RNPxl – Open-source tools for label-free quantification and protein-RNA cross-linking integrated into Proteome Discoverer," *in review*, 2016.

[16] A. Döring, D. Weese, T. Rausch, and K. Reinert, "Seqan an efficient, generic c++ library for sequence analysis," *BMC bioinformatics*, vol. 9, no. 1, p. 11, 2008.

[17] M. Walzer, L. E. Pernas, S. Nasso, W. Bittremieux, S. Nahnsen, P. Kelchtermans, P. Pichler, H. W. P. van den Toorn, A. Staes, J. Vandenbussche, M. Mazanek, T. Taus, R. A. Scheltema, C. D. Kelstrup, L. Gatto, B. van Breukelen, S. Aiche, D. Valkenborg, K. Laukens, K. S. Lilley, J. V. Olsen, A. J. R. Heck, K. Mechtler, R. Aebersold, K. Gevaert, J. A. Vizcaíno, H. Hermjakob, O. Kohlbacher, and L. Martens, "qcML: an exchange format for quality control metrics from mass spectrometry experiments," *Molecular & cellular proteomics: MCP*, vol. 13, no. 8, pp. 1905–1913, 2014.

[18] E. W. Deutsch, M. Chambers, S. Neumann, F. Levander, P.-A. Binz, J. Shofstahl, D. S. Campbell, L. Mendoza, D. Ovelleiro, K. Helsens, L. Martens, R. Aebersold, R. L. Moritz, and M.-Y. Brusniak, "TraML–a standard format for exchange of selected reaction monitoring transition lists," *Molecular & Cellular Proteomics*, vol. 11, no. 4, 2012.

[19] H. L. Röst, U. Schmitt, R. Aebersold, and L. Malmström, "pyOpenMS: A python-based interface to the OpenMS mass-spectrometry algorithm library," *Proteomics*, vol. 14, no. 1, pp. 74–77, 2014.

[20] P. Kiefer, U. Schmitt, and J. A. Vorholt, “eMZed: an open source framework in python for rapid and interactive development of LC/MS data analysis workflows,” *Bioinformatics*, vol. 29, no. 7, pp. 963–964, 2013.
[21] H. L. Röst, G. Rosenberger, R. Aebersold, and L. Malmström, “Efficient visualization of high-throughput targeted proteomics experiments: Tapir,” *Bioinformatics*, p. btv152, 2015.
[22] W. L. Delano, “The PyMOL Molecular Graphics System,” 2002.[23] R. Craig and R. C. Beavis, “TANDEM: matching proteins with tandem mass spectra.,” *Bioinformatics (Oxford, England)*, vol. 20, no. 9, pp. 1466–1467, 2004.
[24] S. Tyanova, T. Temu, P. Sinitcyn, A. Carlson, M.Y. Hein, T. Geiger, M. Mann, J. **Cox, “**The Perseus computational platform for comprehensive analysis of (prote)omics data,” *Nat Methods*, 2016
[25] J. Junker, C. Bielow, A. Bertsch, M. Sturm, K. Reinert, and O. Kohlbacher, “TOPPAS: a graphical workflow editor for the analysis of high-throughput proteomics data,” *Journal of Proteome Research*, vol. 11, no. 7, pp. 3914–3920, 2012.
[26] S. Aiche, T. Sachsenberg, E. Kenar, M. Walzer, B. Wiswedel, T. Kristl, M. Boyles, A. Duschl, C. G. Huber, M. R. Berthold, K. Reinert, and O. Kohlbacher, “Workflows for automated downstream data analysis and visualization in large-scale computational mass spectrometry,” *Proteomics*, 2015.
[27] P. Kunszt, L. Blum, B. Hullar, E. Schmid, A. Srebniak, W. Wolski, B. Rinn, F.-J. Elmer, C. Ramakrishnan, A. Quandt, *et al.*, “iportal: the swiss grid proteomics portal: Requirements and new features based on experience and usability considerations,” *Concurrency and Computation: Practice and Experience*, vol. 27, no. 2, pp. 433–445, 2015.
[28] D. Kessner, M. Chambers, R. Burke, D. Agus, and P. Mallick, “ProteoWizard: open source software for rapid proteomics tools development.,” *Bioinformatics*, vol. 24, no. 21, pp. 2534–2536, 2008.
[29] L. Y. Geer, S. P. Markey, J. A. Kowalak, L. Wagner, M. Xu, D. M. Maynard, X. Yang, W. Shi, and S. H. Bryant, “Open mass spectrometry search algorithm.,” *Journal of Proteome Research*, vol. 3, no. 5, pp. 958–964, 2004.
[30] S. Kim, N. Mischerikow, N. Bandeira, J. D. Navarro, L. Wich, S. Mohammed, A. J. R. Heck, and P. A. Pevzner, “The generating function of CID, ETD, and CID/ETD pairs of tandem mass spectra: applications to database search,” *Molecular & cellular proteomics: MCP*, vol. 9, no. 12, pp. 2840–2852, 2010.
[31] L. Käll, J. D. Canterbury, J. Weston, W. S. Noble, and M. J. MacCoss, “Semi-supervised learning for peptide identification from shotgun proteomics datasets,” *Nature Methods*, vol. 4, no. 11, pp. 923–925, 2007.
[32] O. Serang, M. J. MacCoss, and W. S. Noble, “Efficient marginalization to compute protein posterior probabilities from shotgun mass spectrometry data,” *Journal of Proteome Research*, vol. 9, no. 10, pp. 5346–5357, 2010.
[33] E. Kenar, H. Franken, S. Forcisi, K. Wörmann, H.-U. Häring, R. Lehmann, P. Schmitt-Kopplin, A. Zell, and O. Kohlbacher, “Automated label-free quantification of metabolites from liquid chromatography-mass spectrometry data,” *Molecular & cellular proteomics: MCP*, vol. 13, no. 1, pp. 348–359, 2014.
[34] K. Kramer, T. Sachsenberg, B. M. Beckmann, S. Qamar, K.-L. Boon, M. W. Hentze, O. Kohlbacher, and H. Urlaub, “Photo-cross-linking and high-resolution mass spectrometry for assignment of RNA-binding sites in RNA-binding proteins,” *Nature Methods*, vol. 11, no. 10, pp. 1064–1070, 2014.
[35] H. L. Röst, G. Rosenberger, P. Navarro, L. Gillet, S. M. Miladinović, O. T. Schubert, W. Wolski, B. C. Collins, J. Malmström, L. Malmström, and R. Aebersold, “OpenSWATH enables automated, targeted analysis of data-independent acquisition MS data.,” *Nature Biotechnology*, vol. 32, no. 3, pp. 219–223, 2014.
[36] S. Nahnsen, A. Bertsch, J. Rahnenführer, A. Nordheim, and O. Kohlbacher, “Probabilistic consensus scoring improves tandem mass spectrometry peptide identification,” *Journal of Proteome*

Research, vol. 10, no. 8, pp. 3332–3343, 2011.

[37] L. Nilse, F. C. Sigloch, M. L. Biniossek, and O. Schilling, “Towards improved peptide feature detection in quantitative proteomics using stable isotope labelling,” *Proteomics Clinical Applications*, vol. 9, no. 7-8, pp. 706–714, 2015.

[38] H. L. Röst, U. Schmitt, R. Aebersold, and L. Malmström, “Fast and efficient xml data access for next-generation mass spectrometry,” *PloS one*, vol. 10, no. 4, p. e0125108, 2015.

[39] C. Bielow, S. Aiche, S. Andreotti, and K. Reinert, “MSSimulator: Simulation of mass spectrometry data,” *Journal of Proteome Research*, vol. 10, no. 7, pp. 2922–2929, 2011.

[40] Y. Liu, A. Buil, B. C. Collins, L. C. Gillet, L. C. Blum, L.-Y. Cheng, O. Vitek, J. Mouritsen, G. Lachance, T. D. Spector, *et al.*, “Quantitative variability of 342 plasma proteins in a human twin population,” *Molecular systems biology*, vol. 11, no. 2, p. 786, 2015.

[41] L. C. Gillet, P. Navarro, S. Tate, H. Röst, N. Selevsek, L. Reiter, R. Bonner, and R. Aebersold, “Targeted data extraction of the MS/MS spectra generated by data-independent acquisition: a new concept for consistent and accurate proteome analysis,” *Molecular & cellular proteomics: MCP*, vol. 11, no. 6, p. O111.016717, 2012.

[42] Z. W. Lai, J. Weisser, L. Nilse, F. Costa, E. Keller, M. Tholen, J. N. Kizhakkedathu, M. Biniossek, P. Bronsert, and O. Schilling, “Formalin-fixed, paraffin-embedded tissues (ffpe) as a robust source for the profiling of native and protease-generated protein amino termini,” *Molecular & Cellular Proteomics*, vol. 15, no. 6, pp. 2203–2213, 2016.

[43] S. Tholen, M. L. Biniossek, A.-L. Geßler, S. Müller, J. Weißer, J. N. Kizhakkedathu, T. Reinheckel, and O. Schilling, “Contribution of cathepsin l to secretome composition and cleavage pattern of mouse embryonic fibroblasts,” *Biological chemistry*, vol. 392, no. 11, pp. 961–971, 2011.

[44] J. C. Wright, J. Mudge, H. Weisser, M. P. Barzine, J. M. Gonzalez, A. Brazma, J. S. Choudhary, and J. Harrow, “Improving GENCODE reference gene annotation using a high-stringency proteogenomics workflow,” *Nature Communications*, vol. 7, pp. 11778+, June 2016.

[45] J. Harrow, F. Denoeud, A. Frankish, A. Reymond, C.-K. Chen, J. Chrast, J. Lagarde, J. G. Gilbert, R. Storey, D. Swarbreck, *et al.*, “Gencode: producing a reference annotation for encode,” *Genome Biol*, vol. 7, no. Suppl 1, p. S4, 2006.

[46] R. Petryszak, M. Keays, Y. A. Tang, N. A. Fonseca, E. Barrera, T. Burdett, A. Füllgrabe, A. M.-P. Fuentes, S. Jupp, S. Koskinen, *et al.*, “Expression atlas update—an integrated database of gene and protein expression in humans, animals and plants,” *Nucleic acids research*, vol. 44, no. D1, pp. D746–D752, 2016.

[47] M. Choi, C.-Y. Chang, T. Clough, D. Broudy, T. Killeen, B. MacLean, and O. Vitek, “MSstats: an R package for statistical analysis of quantitative mass spectrometry-based proteomic experiments,” *Bioinformatics*, p. btu305, 2014.

[48] G. Rosenberger, C. Ludwig, H. L. Röst, R. Aebersold, and L. Malmström, “aLFQ: an R-package for estimating absolute protein quantities from label-free LC-MS/MS proteomics data,” *Bioinformatics*, vol. 30, no. 17, pp. 2511–2513, 2014.

[49] B. MacLean, D. M. Tomazela, N. Shulman, M. Chambers, G. L. Finney, B. Frewen, R. Kern, D. L. Tabb, D. C. Liebler, and M. J. MacCoss, “Skyline: an open source document editor for creating and analyzing targeted proteomics experiments,” *Bioinformatics*, vol. 26, no. 7, pp. 966–968, 2010.

Highlighted references:

- Sturm et al. 2008: Contains the first description of OpenMS as a C++ software library
- Weisser et al. 2013: Extended description of the OpenMS label-free workflow and compares the results to other software.
- Röst et al. 2014: First publication of an automated workflow for targeted analysis of SWATH-MS data,

implemented in OpenMS.

- Kenar et al. 2014: First application of OpenMS to metabolomics.
- Kramer et al. 2014: Application of OpenMS to investigate RNA-protein cross linking.
- Aiche et al. 2015: Highlights the importance of workflows in the world of MS and discusses open-source software solutions for workflow management.

Figure Legends for main text

Figure 1: **Automated and interactive data analysis using OpenMS.** OpenMS supports both the fully automated processing and analysis of mass spectrometric data and their interactive exploration. Automated processing workflows including raw data conversion, quantification, identification, and downstream statistical analysis can be integrated into the analytics workbench KNIME. From within this interface, the raw and processed data can be inspected using TOPPView, the OpenMS visualization tool. After manual or fully automated processing, the results can be exported in open standard formats for data deposition. If the processing was fully done through the workflow system, then depositing the workflow (including all its parameters) together with the raw data and results data yields a reproducible documentation of the complete data experiment and analysis.

Figure 2: **The data analysis workflow for SWATH-MS analysis of human blood plasma study.** Over 240 blood plasma samples were acquired using SWATH-MS and then analyzed with the OpenSWATH module in OpenMS for the targeted analysis of the SWATH maps. A machine learning approach implemented in Python was used for error rate control of the final quantitative data matrix. This allowed Liu et al. [32] to decompose the variance of over 300 blood plasma proteins into heritable, environmental and longitudinal components.

Figure 3: **The structure of the OpenMS framework.** The OpenMS software framework consists of three main layers, (i) the object-oriented OpenMS core library (containing over 1,300 classes) built on modern C++ infrastructure, (ii) the pyOpenMS interactive Python interface, providing easy integration of the OpenMS library with other scientific Python libraries as well as the TOPP tools, a set of pre-built tools covering most core tasks in computational mass spectrometry, and (iii) the workflow layer, allowing users to apply flexible workflows using different workflow engines (e.g., TOPPAS, KNIME, Galaxy or vendor-specific workflow engines). Each level of increasing abstraction provides better usability, but limits extensibility, as the Python and workflow levels only have access to the exposed Python API or the available set of executables, respectively. Increasing abstraction, however, makes it easier to design and execute complex analyses, even across multiple omics types. The different layers cater to the needs of different communities from bioinformaticians developing new tools to life scientists interested in data analysis only.

Boxes

Box 1: Getting started with OpenMS

- Visit the project webpage and follow the instructions at http://www.openms.org/downloads/ to obtain the OpenMS binaries. Optionally, you can install a workflow manager like KNIME (https://www.knime.org/) or Galaxy (https://galaxyproject.org/) which will allow you to easily execute workflows using OpenMS.
- Read the “getting started” guide at http://www.openms.org/getting-started/ and familiarize yourself with the tutorials http://www.openms.org/tutorials.
- Investigate your data interactively with TOPPView or build your first workflow with KNIME following the examples in the tutorials.
- Get help on the mailing list if you get stuck (see http://www.openms.org/support/ for all the ways to get support).

Box 2: How to contribute to OpenMS If you want to contribute to OpenMS, you can do this on many levels.

- To help other researchers with their questions, you can subscribe to the mailing list and share your expertise in mass spectrometry and OpenMS. You can share analysis workflows for specific analytical questions on https://www.openms.org/ or through http://www.myexperiment.org/ which will enable other researchers to use them directly in their own data analysis.
- Proposed enhancements or encountered bugs can be reported on the OpenMS GitHub page (https://github.com/OpenMS/OpenMS/issues) where different approaches to address the issue can be discussed. This is especially useful for new developers that may not be familiar with all code inside OpenMS – often they will find that the proposed feature is already available in OpenMS or can be achieved with minor modifications of existing code.
- In order to contribute code to OpenMS, you can obtain the source code using git from https://github.com/OpenMS/OpenMS. You should familiarize yourself with the data structures and the existing algorithms inside OpenMS, which are described in the developer tutorial. The tutorials, the OpenMS mailing list and the developer retreats are great ways to get started when developing new functionality within the OpenMS framework. While experienced developers can directly start writing C++ tools, users more comfortable with high-level languages can use Python and pyOpenMS to develop their first algorithms.
- To add new functionality to OpenMS, a developer has to submit the code through a “pull request” on GitHub. The developer uploads the code to GitHub and presents the proposed enhancement or fix to the other developers with a short rationale. The core developers will then review the code and either accept the code as is or ask for improvements to make the code compatible with OpenMS. At this point, please make sure that the new code is properly tested and all previous tests still pass.

- Once new functionality is accepted, it will automatically be included in the nightly builds and in the next full release. Releases are scheduled every 6-12 months, allowing for rapid development.
- Each class inside OpenMS and each tool has a designated maintainer who must keep the code up to date and fix errors in their code. If you have questions regarding a certain module, contacting the responsible maintainer is good way to understand the code and improving it.

Figure 1

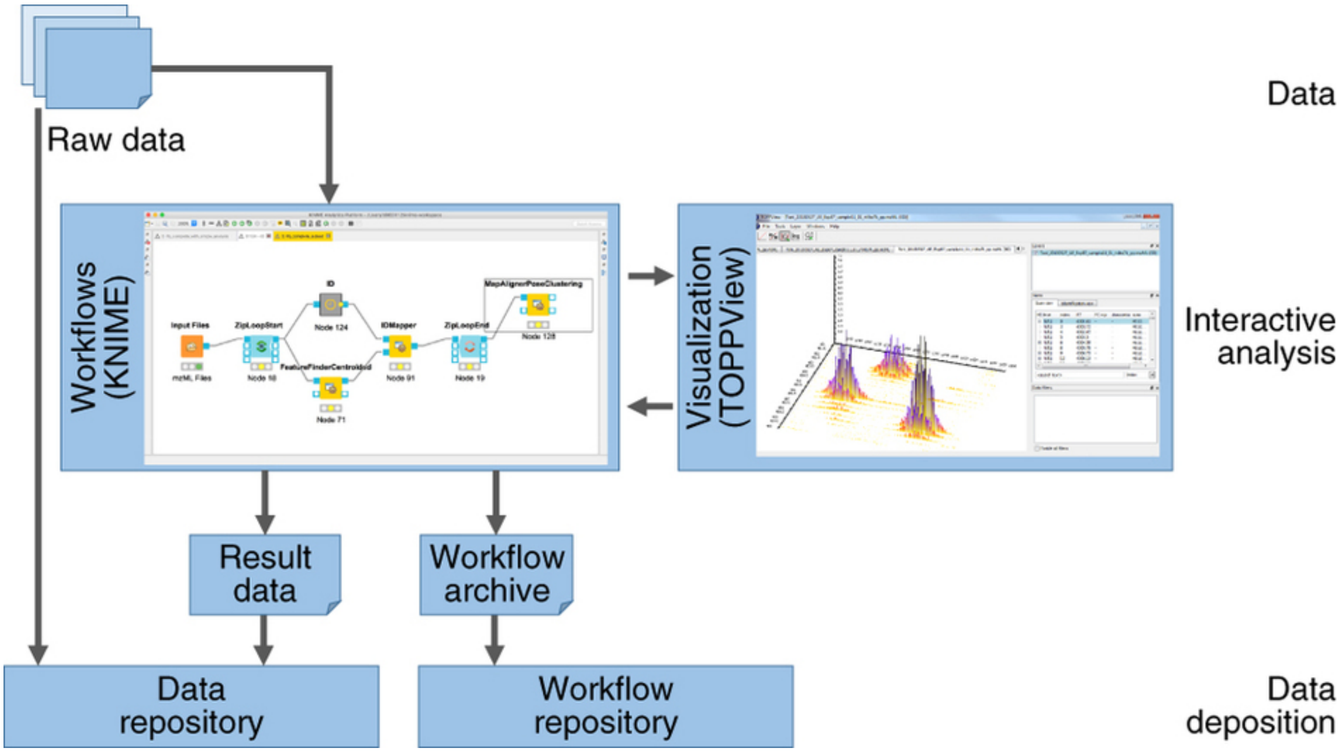

Raw data
Data
Workflows (KNIME)
Input Files
mzML Files
ZipLoopStart
Node 18
ID
Node 124
FeatureFinderCentroided
Node 71
IDMapper
Node 91
ZipLoopEnd
Node 19
MapAlignerPoseClustering
Node 128
Visualization (TOPPView)
Interactive analysis
Result data
Workflow archive
Data repository
Workflow repository
Data deposition

Figure 2

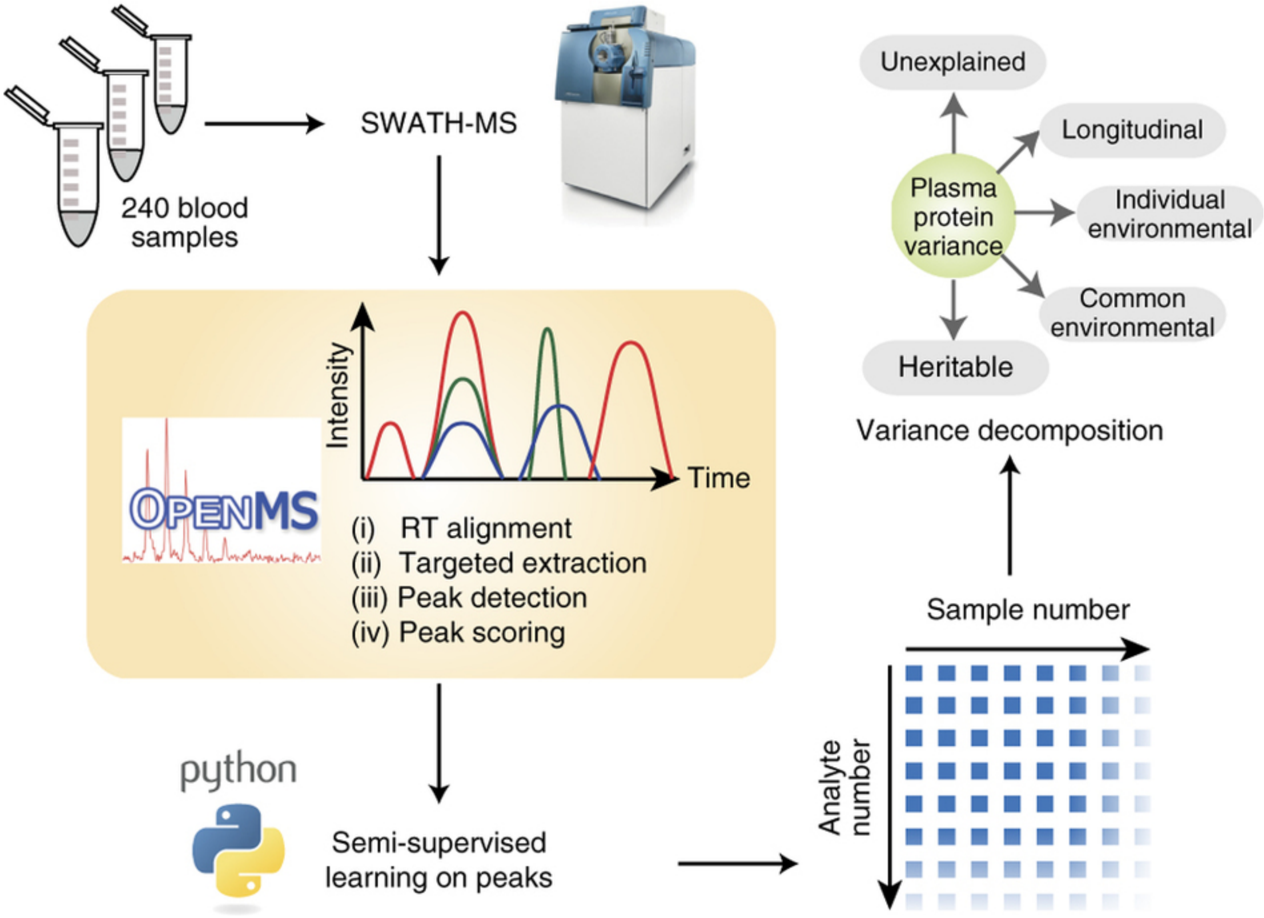
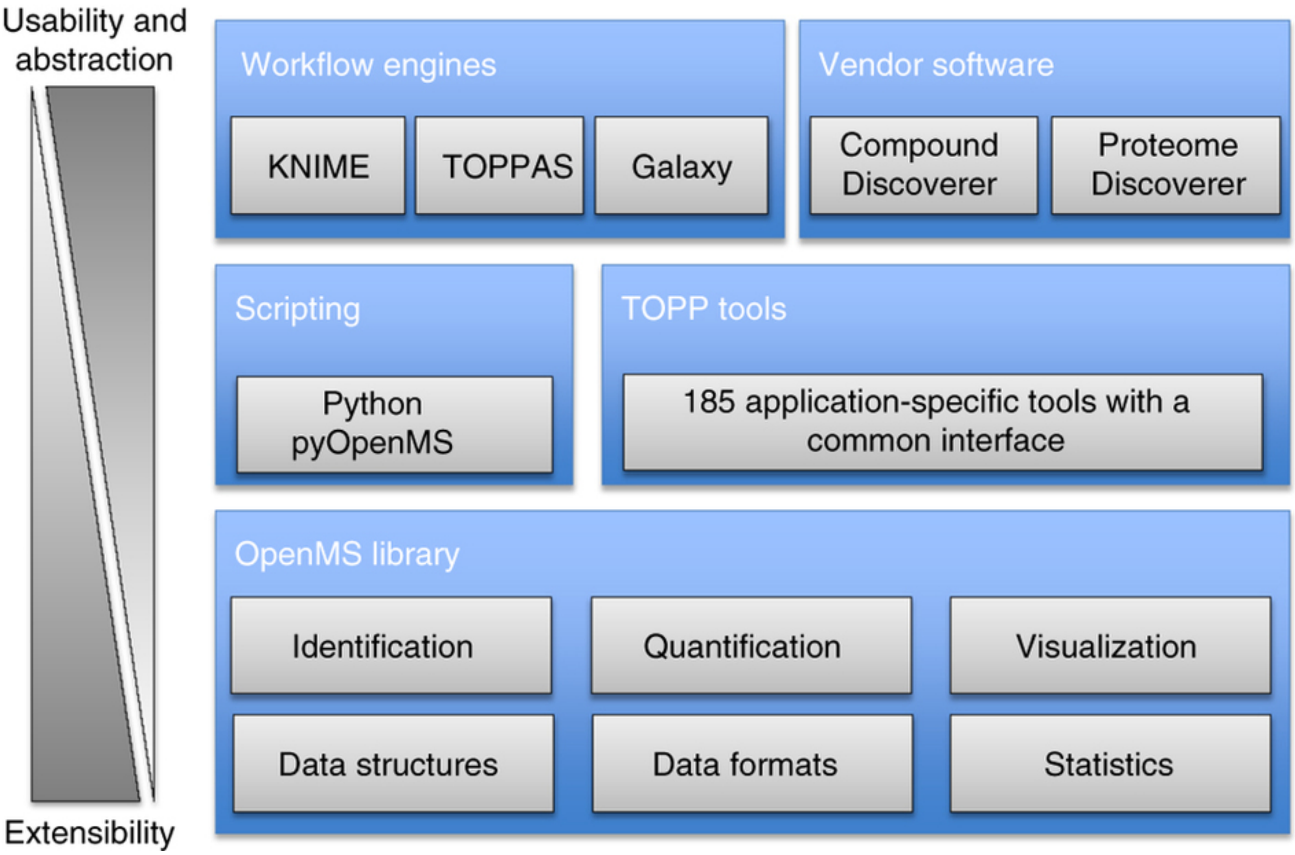

240 blood samples
SWATH-MS
Intensity
Time
OpenMS
(i) RT alignment
(ii) Targeted extraction
(iii) Peak detection
(iv) Peak scoring
python
Semi-supervised learning on peaks
Sample number
Analyte number
Variance decomposition
Unexplained
Longitudinal
Plasma protein variance
Individual environmental
Common environmental
Heritable

Figure 3


Usability and abstraction
Extensibility
Workflow engines
KNIME
TOPPAS
Galaxy
Vendor software
Compound Discoverer
Proteome Discoverer
Scripting
Python pyOpenMS
TOPP tools
185 application-specific tools with a common interface
OpenMS library
Identification
Quantification
Visualization
Data structures
Data formats
Statistics