Open-source and FAIR Research Software for Proteomics

Yasset Perez-Riverol [1], Wout Bittremieux [2], William S. Noble [3], Lennart Martens [4,5], Aivett Bilbao [6], Michael R. Lazear [7], Bjorn Grüning [8], Daniel S. Katz [9], Michael J. MacCoss [10], Chengxin Dai [11], Jimmy K. Eng [12], Robbin Bouwmeester [4,5], Michael R. Shortreed [13], Enrique Audain [14], Timo Sachsenberg [15], Jeroen Van Goey [16], Georg Wallmann [17], Bo Wen [3], Lukas Käll [18,*], William E. Fondrie [19]

[1] European Molecular Biology Laboratory, European Bioinformatics Institute, Wellcome Genome Campus, Cambridge, UK.

[2] Department of Computer Science, University of Antwerp, 2020 Antwerpen, Belgium.

[3] Department of Genome Sciences, University of Washington, Seattle, WA, USA.

[4] VIB-UGent Center for Medical Biotechnology, VIB, Ghent 9052, Belgium.

[5] Department of Biomolecular Medicine, Ghent University, Ghent 9052, Belgium.

[6] Environmental Molecular Sciences Laboratory, Pacific Northwest National Laboratory, Richland, Washington 99352, USA. US Department of Energy Agile BioFoundry, Emeryville, CA, 94608, USA.

[7] Belharra Therapeutics, 3985 Sorrento Valley Boulevard Suite C, San Diego, California 92121, USA.

[8] Bioinformatics Group, Department of Computer Science, Albert-Ludwigs University Freiburg, Freiburg, Germany.

[9] National Center for Supercomputing Applications & Siebel School of Computing and Data Science & School of Information Sciences, University of Illinois Urbana-Champaign, Urbana, Illinois, USA.

[10] Department of Genome Sciences, University of Washington, 3720 15th St. NE, Seattle, Washington 98195, USA.

[11] State Key Laboratory of Proteomics, Beijing Proteome Research Center, National Center for Protein Sciences (Beijing), Beijing Institute of Life Omics, Beijing, China.

[12] Proteomics Resource, University of Washington, Seattle, Washington 98195, USA.

[13] Department of Chemistry, University of Wisconsin-Madison, Madison, WI, USA.

[14] Institute of Medical Genetics, University Medicine Oldenburg, Carl von Ossietzky University, Oldenburg, Germany.

[15] Department of Computer Science, Applied Bioinformatics, University of Tübingen, Tübingen, Germany.

[16] InstaDeep.

[17] Proteomics and Signal Transduction, Max Planck Institute of Biochemistry, Martinsried, Germany
[18] Science for Life Laboratory, School of Engineering Sciences in Chemistry, Biotechnology and Health, KTH Royal Institute of Technology, Stockholm, Sweden.
[19] Talus Bioscience, Seattle, WA, USA.

Corresponding author: Lukas Käll (lukas.kall@scilifelab.se)

Abstract

Scientific discovery relies on innovative software as much as experimental methods, especially in proteomics, where computational tools are essential for mass spectrometer setup, data analysis, and interpretation. Since the introduction of SEQUEST, proteomics software has grown into a complex ecosystem of algorithms, predictive models, and workflows, but the field faces challenges, including the increasing complexity of mass spectrometry data, limited reproducibility due to proprietary software, and difficulties integrating with other omics disciplines. Closed-source, platform-specific tools exacerbate these issues by restricting innovation, creating inefficiencies, and imposing hidden costs on the community. Open-source software (OSS), aligned with the FAIR Principles (Findable, Accessible, Interoperable, Reusable), offers a solution by promoting transparency, reproducibility, and community-driven development, which fosters collaboration and continuous improvement. In this manuscript, we explore the role of OSS in computational proteomics, its alignment with FAIR principles, and its potential to address challenges related to licensing, distribution, and standardization. Drawing on lessons from other omics fields, we present a vision for a future where OSS and FAIR principles underpin a transparent, accessible, and innovative proteomics community.

Introduction

Scientific discovery today is as much a product of innovative software as it is of groundbreaking experiments, and the right tools often mean the difference between success and stagnation. Indeed, the majority of scientists recognize scientific software as indispensable for their work and often impossible to conduct research without it (1, 2). This reliance on software is equally crucial in proteomics, where researchers depend on a range of tools and algorithms for every step, from mass spectrometer configuration and data acquisition to the subsequent stages of processing, analysis, and interpretation (3, 4).

Since the original publication of the first mass spectrum database search tool, SEQUEST (5), proteomics software has evolved into a sophisticated ecosystem encompassing multiple

stages of data processing, advanced predictive models, and robust computational frameworks. The publication of SEQUEST exemplifies a subsequent recurring pattern in MS-based proteomics, in which the development of software drives the adoption of novel experimental methodology. Numerous examples of open-source software tools have been developed and used by the proteomics community. These include tools like Percolator (6), which is used to improve peptide and protein identification using machine learning, MS2PIP (7, 8) and Prosit (9), which apply gradient boosting and deep learning, respectively, to predict fragment ion intensities, aiding in more accurate spectral matching, which can in turn be used by Percolator-based rescoring approaches like MS2Rescore (10), and Proteowizard (11), which provides shared libraries and tools for data access. Platforms like GalaxyP (12) and quantms (13) facilitate accessible, reproducible analyses through high-performance computing (HPC) and distributed workflows, supporting researchers in handling large datasets and complex analyses. Together, these advances underscore how proteomics software has transformed into a multidisciplinary field, involving a complex ecosystem of algorithms, models, and software tools that build upon sophisticated computational and algorithmic expertise.

Computational proteomics faces several key challenges common to other omics fields:

- The increasing complexity and size of data acquired by mass spectrometers, the complex sequence of steps, including spectral processing, statistical analysis, and biological interpretation, along with the need to manage algorithmic details and parameter settings, all contribute to making software development in proteomics a complex and demanding endeavour.
- Although most software tools are described in publications, the absence of open-source code, comprehensive documentation, and version control often impedes the reproducibility, reuse and interpretation of the results generated by these tools. This lack of transparency prevents researchers from extending existing algorithms and adapting software to keep pace with rapidly advancing instrumentation and acquisition methods. Transparency is especially important during the relatively frequent shifts in data processing paradigms, like the rapid adoption of data-independent acquisition over data-dependent acquisition.
- Ensuring reproducibility is a challenge due to the limited access to detailed algorithmic information, which hinders validation and extension of methods.
- Custom licenses and restrictions on software distribution can further complicate the situation, making it difficult to share, modify, or redistribute software, and hindering the development of a collaborative and open-source ecosystem. Proteomics software is often distributed under restrictive licenses and tailored to specific platforms (e.g.,

operating systems, and computer architectures), limiting its use across diverse environments and services. Not surprisingly, such restrictive licenses hinder the field's adaptability and impact the integration of proteomics with other omics disciplines.

- Complex workflows and high-throughput data analysis are growing in proteomics (12-14). This complexity requires managing dependencies, configurations, and environments consistently across diverse systems and architectures without human intervention. The community has tackled this challenge with continuous integration and deployment (CI/CD) pipelines as, for example, described in (15), but these pipelines rely upon the permission to freely redistribute software along the entire dependency chain. If one piece in this supply chain is not redistributable, then these exceptions must be handled, and automation is harder or impossible. A substantial additional burden is therefore created downstream of any non-OSS software package, which is a hidden cost on its own, and one that affects the entire community. In sum, restricted distribution terms create an additional burden for the entire community downstream, which on its own is a hidden cost.
- The lack of standardization in proteomics software development, including inconsistent documentation, variable code quality, and limited community engagement, can hinder the adoption and use of software tools, leading to inefficiencies and redundancies in software development.
- Closed-source, platform-specific software has caused lock-in effects, restricting users to specific tools and hindering innovation. Until recently, major instrument vendors lacked open-source, cross-platform libraries for data access, limiting data reuse and algorithm development (16). Thermo Fisher's RawFileReader library marks important progress in this respect, enabling tools like ThermoRawFileParser (17) and PRIDE Archive USI (18, 19). This idea has been recently extended for Bruker timsTOF data with the timsrust library (https://github.com/MannLabs/timsrust/), which is open-source and already used by tools like the Sage search engine (20).

A solution to these challenges is offered by open-source software (OSS) which is also aligned with the FAIR principles (Findable, Accessible, Interoperable, Reusable) that have initially been established for scientific data (21). The FAIR principles were expanded in 2022 to research software (FAIR4RS) to address the growing recognition of research software as a foundational research asset (22). Following FAIR4RS principles empowers proteomics with OSS tools that are not only accessible, but also foster community-driven development, rigorous validation, and transparent sharing of methodologies (22, 23). Although OSS is not an explicit requirement for implementing FAIR principles, it facilitates the realization of these principles by making software more accessible, transparent, and reusable. OSS has

demonstrated clear benefits in increasing the accessibility, usability, and visibility of scientific software (24). In particular, OSS makes reproducibility, traceability, and auditability possible. With code freely available for inspection, modification, and distribution, OSS encourages collaboration and creates avenues for continuous improvement—factors that are critical in fields as data-intensive as proteomics.

In this manuscript, we aim to explore the role of OSS in computational proteomics and its implications for the development of FAIR research software. We will discuss the benefits and challenges of OSS in proteomics, the role of OSS in the development of FAIR research software, and the importance of distribution, licensing, and citation of software in computational proteomics. We will also explore how other omics fields deal with OSS and FAIR software and how these experiences can inform the development of proteomics software. Our goal is to present a vision for a future where OSS and FAIR software are encouraged and supported in the proteomics community.

2. What does it mean for software to be "open source"?

2.1. Attributes of an open-source project

Open-source software (OSS) is defined by its publicly accessible source code, allowing anyone to view, modify, and distribute it under an Open-Source Initiative-approved license. Merely making source code available is not enough; licenses that restrict use or modification to specific fields (e.g., non-commercial use) do not qualify as open source. Unlike closed-source, OSS guarantees transparency, fostering trust, collaboration, and scientific progress. To address misconceptions in proteomics, we aim to clarify for the community (users, developers, and reviewers) the essential criteria for OSS (https://opensource.org/osd):

- **Source Code Availability**: The source code—the instructions that define how software functions—is publicly accessible, allowing anyone to view, download, and examine the code's details (https://opensource.org/osd).
- **OSI-Approved License**: The software must use an Open Source Initiative-approved license, which specifies rights to freely use, modify, and distribute the software, regardless of its application or environment (https://opensource.org/licenses).
- **Freedom to Modify and Distribute**: Open-source software, in contrast to source-available software, allows users not only to access the code but also to modify it and share these modifications, encouraging innovation and collaboration.
- **Transparency and Community Trust**: With open source, the code is transparent by design, allowing the community to understand, verify, and contribute to the project. This transparency fosters trust and credibility, which is essential in scientific fields.

- **Collaborative Development**: Open-source projects can be maintained by communities or dedicated teams, and they welcome contributions, such as bug reports, enhancements, and new features, from a diverse group of users and developers.
- **Long-term Sustainability**: Because the code is publicly accessible, open-source projects are less dependent on single organizations or developers for their maintenance and long-term survival, promoting continuity and stability even if the original contributors leave.
- **No Restriction to Specific User Groups**: Unlike “free-for-academic-use” licenses, which restrict usage to academic settings, open-source licenses do not impose limitations on the types of users or institutions that can access or use the software.
- **Not Necessarily Free of Cost**: Open-source software is “free” in terms of freedom, not necessarily in terms of price. Users might pay for support, hosting, or additional services, but they retain freedom in how they use and modify the software.

2.2. Misconceptions about open source

In proteomics, and bioinformatics in general, multiple misconceptions exist about open/closed source software:

- Cost-free software is not always open-source: Many programs are freely available for non-commercial or academic use but do not meet open-source criteria. Similarly, "free and open-source software" (FOSS) refers to the freedom to run, modify, and share the software, not necessarily its financial cost. FOSS may involve expenses for services like support or hosting, but it ensures that users retain the freedom to use, adapt, and distribute the software as they wish (https://www.gnu.org/philosophy/free-sw.html).
- So-called "academic licenses" only refer to free-for-academic-use: The source code is not necessarily open, shareable, or modifiable. Even the term "academic" is not well-defined as it can refer to a wide range of institutions and organizations. To simplify this complexity, we can define OSS as any software that uses a license approved by the Open Source Initiative (OSI, https://opensource.org/licenses).
- Accessible source code does not mean open-source: Open-source software does not only mean that the source code is available but that it is allowed to be freely modified and shared regardless of whether the users work in academic or commercial settings.
- Open-source software does not imply a lack of professional quality: Many open-source projects are maintained by dedicated teams with robust testing and good programming practices. In genomics, projects like samtools (https://github.com/samtools/samtools) (25), an MIT-licensed (https://opensource.org/license/mit-0) project with over 80 contributors and 50,000+ citations, and the Genome Analysis Toolkit (GATK,

https://github.com/broadinstitute/gatk) (26), now open-source with over 100 maintainers and 26,000+ citations, exemplify this standard. In proteomics, Percolator (https://github.com/percolator/percolator) (6) has over 2000 citations and 20 contributors, serving as a core tool for projects like MS2Rescore (10), OpenMS (27), MSBooster (28), DeepRescore (29), Crux (30), and even commercial tools like Mascot and Proteome Discoverer, and many others (13, 31-33). Other successful open-source projects in proteomics, such as OpenMS (27), Skyline (34), Comet (35), PeptideShaker (36), ThermoRawFileParser (17), and ProteoWizard (11), demonstrate the benefits of transparency and collaboration. Despite these successes, academic open-source proteomics software is still perceived as lower quality. In 2018, Rob Smith highlighted the community's concerns about academic proteomics and metabolomics software, including poor documentation, lack of transparency, and limited support (37) . However, it should be noted that much of this feedback was directed at academic and free-for-academic-use software rather than exclusively open-source software.

2.3. Detrimental practices in using public repositories

In addition to the described misconceptions and complexity, many journals and some funding agencies mandate code availability as part of publishing, which has prompted multiple bad practices from software developers and bioinformaticians aiming to fulfil these requirements. Notable examples include:

- **Open-source Facade**: Researchers may upload closed-source software to platforms like GitHub, giving an impression of openness with features such as issue tracking, while the actual source code remains inaccessible. Although often well-intentioned, this practice can mislead scientists and, in our view, should be discouraged. In these instances, a clear statement in the repository should indicate to the users that the software is not open source.
- **Alterations Post-Publication**: Software is deposited in GitHub as open source during the submission of the manuscript, but after publication, software licenses in GitHub repositories are changed, or repositories are deleted or made private, all of which complicates efforts to ensure long-term accessibility.
- **License Misuse or Ambiguity**: Some repositories may use inappropriate or ambiguous licenses, causing confusion about the terms of use, distribution, and modification (more details discussed in the section Licenses in proteomics software).
- **Obscure Dependencies**: Software repositories may have dependencies that are not clearly documented, which may require closed-source or proprietary software. This can create barriers for other researchers attempting to run or build on software, as they may not have access to necessary components or may need to purchase expensive

licenses. Clear documentation of all dependencies along with their licensing terms is essential to ensure transparency and reproducibility.

3. Licenses in proteomics software

We want to emphasize a fundamental aspect and challenge in proteomics software development: the choice of the licenses. Licenses serve as the foundation for defining key aspects of software, including commercialization, code reuse, distribution, and proper citation. It is therefore crucial to provide a license, and vital to choose a relevant one. As the gold standard for proteomics software development, we recommend using a standard OSS license like Apache 2.0, MIT, BSD, LGPL, and GPL; the full list of applicable licenses can be found at (https://opensource.org/licenses). These licenses are all well known, are in use across many fields, and are well understood by the community. Additionally, they are compatible with the FAIR principles and the OSI guidelines (22). These established licenses moreover all have a clear definition of what is allowed and what is not, and how the software can be distributed, reused, and cited.

Many proteomics code repositories do not have a software license specified (Figure 1). It is important to note that without a specified license the software is not open source. With an unspecified software license, the software and contributions are exclusively owned by the authors, and no one can use, copy, or distribute the contributions. The fact that so many proteomics tools have unspecified licenses underscores a misunderstanding of software licensing in the proteomics community.

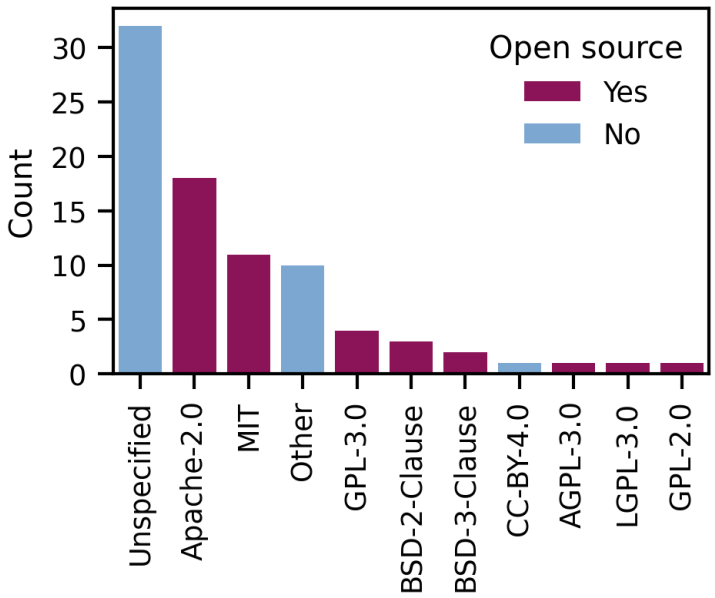

Figure 1: Software licenses in use in proteomics. Scientific papers published in the *Journal of Proteome Research* that include a GitHub URL in their abstract were automatically retrieved from PubMed and

information on the software license of the corresponding GitHub repository was retrieved through the GitHub API. The code to generate these data is available at https://gist.github.com/bittremieux/70905e5d9dcc829ae49aab49e85954af.

In addition, as the field is evolving and software becomes more complex and has multiple components, different components could have different licenses. Consequently, dependencies between these components should be clearly stated. We recommend clearly stating the dependencies that a piece of software might have and the licenses of each of them. Full disclosure of such dependencies is necessary to ensure that the user is aware of this, such that the community, developers, and journal reviewers are able to understand this challenge.

4. Why open-source software is essential for scientific research

4.1. Transparency promotes scientific rigor

The scientific community increasingly recognizes that algorithms, while not software or tools themselves but rather the underlying steps and methodology, are becoming significant research outputs in their own right. Algorithms are no longer seen merely as tools but are valued as core research outputs, reflecting the critical steps and methodologies at the heart of scientific innovation. For example, the peptide spectrum scoring function HyperScore was originally implemented in the open-source search engine X!Tandem (38), later adopted by search engines including MSFragger (39), EncyclopeDIA (40), PepQuery (41) and Sage (20). This shift highlights the importance of not only software as a means of implementation but also the reproducibility and reliability of the underlying computational methods that drive new discoveries. Both algorithms and their software implementations are now held to rigorous validation and reproducibility standards, similar to those for traditional experimental data and methodology.

Transparent computational methods open doors to innovation, enabling researchers to test hypotheses, refine methodologies, and build upon one another's work with confidence. For instance, providing open-source implementations allows the scientific community to verify methods, adapt them to new challenges, and explore alternative approaches. Consider a proteomics experiment: without details on sample preparation or instrument settings or the raw data, the final results lack reproducibility. Similarly, open-source code ensures that computational methods can be accurately understood, replicated, and extended across labs worldwide. This transparency is particularly relevant for core proteomics workflows—as demonstrated by AlphaDIA (42)—where understanding the underlying algorithms of protein search engines directly impacts data interpretation and research outcomes.

When algorithms and models are shared as open-source software, they inherently uphold the FAIR principles applied to scientific data. This level of openness strengthens scientific rigor, enabling others to examine the code, replicate findings, and contribute improvements. A transparent approach to computational research, through openly available code, fosters a collaborative environment where the community can validate results and improve tools, ultimately building trust in computational methodologies and accelerating innovation.

Moreover, open-source implementations guard against unintended variation in outcomes caused by minor differences in coding practices, dependencies, or hardware environments. Even small programming choices can lead to significant changes in results. Open-source code mitigates these risks by making the entire process visible, allowing other scientists to understand the nuances and make informed adjustments. Transparency is key in computational research, not just for ensuring rigor but for building a reliable foundation that drives the entire field forward.

Finally, open-source code allows researchers to apply and compare different implementations, revealing assumptions and enhancing understanding. For instance, discrepancies between implementations of common tools, such as variations in BLOSUM matrices for sequence alignment (43), demonstrate how essential code transparency is for ensuring scientific consistency. Open-source practices thus empower researchers to expand on established methods with confidence, propelling science toward more robust, reproducible, and innovative outcomes.

4.2. Shared knowledge pushes the field forward

Open-source software fosters a collaborative ecosystem where researchers across institutions can freely contribute, refine, and extend tools, accelerating scientific progress. Unlike proprietary software that confines advances to specific labs or companies, OSS allows researchers to rapidly build on each other's work without duplicating efforts, promoting efficient resource use and transforming individual achievements into collective gains. This is particularly vital in proteomics, where bioinformatics is integral to every workflow, and progress depends on the synergy between wet-lab experimentation and computational innovation. Extending and building on top of existing algorithms is crucial for scientific progress.

Proteomics has already greatly benefited from this open-source approach. Projects like ProteoWizard (11, 44), with tools such as Skyline (34), msConvert (45), and SearchGUI (46), exemplify OSS's impact. Skyline, for instance, supports over twenty external plugins available in its Tool Store, allowing users to perform specialized tasks far more efficiently than if they had

to build solutions from scratch. Similarly, msConvert provides a standardized interface for mass spectrometry data, sparing developers the need to manage proprietary formats. SearchGUI (46), finally, provides a unified graphical user interface to twelve different search engines, in addition to ThermoRawFileParser and the abovementioned msConvert. Together, such well-supported OSS projects create a foundational infrastructure that accelerates proteomics advancements.

The field of genomics offers a compelling example of how open-source initiatives can drive transformative progress. OSS such as reference-based aligners, e.g., BWA (47), variant-calling algorithms, e.g., GATK-HaplotypeCaller (26), and large-scale cloud-based genomic data analysis tools, e.g., Hail (https://hail.is), have revolutionized genomics research. Furthermore, these tools have been seamlessly integrated into broader computational frameworks like the nf-core/sarek (48), demonstrating how community-driven collaboration and standardization can amplify the impact of individual tools. This collaborative model underscores the potential for proteomics and other fields to follow a similar path, leveraging OSS to achieve greater integration, scalability, and innovation.

However, sustaining successful OSS projects in proteomics requires ongoing community engagement, which has often proven challenging. Despite their long history, projects like ProteoWizard and Skyline see few external contributions. Many researchers opt to develop independent tools rather than contribute enhancements within Skyline, missing opportunities for broader collaboration. Skyline's external tools framework, which lowers technical barriers to contributions, has helped, but much of the development remains within the original labs. Community contributions in proteomics face barriers associated with multiple challenges. Developing software for proteomics demands specialized technical skills that many labs lack, especially when resources are focused on biological research rather than software engineering. The need for continuous updates to accommodate evolving data formats and instruments also requires substantial resources. Additionally, academic incentives often prioritize novel software creation over contributions to existing projects, further deterring collaborative development.

To create a more robust and impactful OSS ecosystem in proteomics, stronger incentives for community involvement and frameworks that support sustained collaboration are essential. With enhanced incentives, collaborative frameworks, and dedicated resources, the proteomics community can achieve a more sustainable, widely supported, and effective ecosystem of open-source tools. Apart from the engagement needed from the community to foster the development of open-source software, proteomics could create and sustain some of the core

functionalities of the field in small libraries and tools that could be used by the entire community: for example tools like MS2Rescore (10) for rescoring peptide identifications, pyOpenMS (49) for Python-based proteomics functions, or spectrum_utils (50) for spectral data manipulation.

4.3. The community can contribute to development

One of the greatest strengths of open-source software is that "given enough eyeballs, all bugs are shallow" (51). Many of the critical pieces of software that underpin the modern technology stack are open source: Linux powers operating systems across the globe, Chromium serves as the foundation for multiple web browsers, PostgreSQL is a backbone of data storage, and Python and PyTorch have revolutionized machine learning and data science. Bringing this open-source ethos to proteomics holds the potential to accelerate advances in the field, creating tools that are not only robust but also accessible to a global community of researchers.

Bugs and mistakes are inevitable in complex software, but collaborative scrutiny allows them to be addressed more efficiently. In proteomics, as in other scientific fields, the diverse expertise of the community enhances both the quality and the utility of open-source tools. Users who encounter issues or limitations often provide feedback, suggest solutions, or even contribute code to address the challenges, fostering continuous improvement. In our own work, users have uncovered bugs that we subsequently corrected or asked questions about the underlying code which led to new features, fewer bugs, and more efficient algorithms. This feedback loop is unique to OSS, where the contributions from the community enhance the quality and precision of the software over time. Compared to proprietary software, OSS can often move faster and defray development costs by enabling users to build and contribute the features they need, rather than hoping that the maintainers of the software are willing or able to add the features themselves. This dynamic frees developers from the burden of predicting and implementing every possible use case and shifts some of the innovation to the broader community. For users, OSS reduces reliance on software maintainers, allowing research to advance even in the absence of formal support.

Without such transparency, computational research risks becoming a "black box" that stifles innovation rather than promoting it, hindering the growth of scientific knowledge. OSS can foster a culture of shared accountability, where code is not just released but continuously scrutinized and refined, driving the field forward in a collective effort toward scientific rigor. We

have indeed observed this in some of our projects: at the time of writing, quantms (13) and mokapot (52) now have 12 and 13 contributors, respectively.

5. Open source and ML/AI models in proteomics

Machine learning and deep learning are increasingly used in proteomics, with examples like the MS2 prediction models Prosit, pDeep (53), AlphaPeptDeep (54) and MS2PIP (7, 8), retention time prediction models DeepLC (55) and AutoRT (56), and the *de novo* peptide sequencing models Casanovo (57) and InstaNovo (58). Many deep learning-based proteomics tools enhance reproducibility by clearly reporting source code, training parameters, and other details.

While closed-source tools have contributed to research, their models may carry biases that are difficult to detect and diagnose, and their potential utility can be hard to assess when code and models are not accessible. A more contentious issue arises when closed-source or commercial models are trained on publicly shared community datasets, often under open-source licenses.

Open-source software has proven its value by removing barriers to learning, sharing, and improving systems. For AI in proteomics, society needs similar freedoms: autonomy, transparency, ease of reuse, and collaborative improvement. The Open-Source Initiative's Open-Source AI Definition (OSAID) outlines these freedoms:

- Use the system for any purpose.
- Study how the system works and inspect its components.
- Modify the system, including changing its output.
- Share the system, with or without modifications, for any purpose.

AI and machine learning are more than software: they encompass data, configurations, documentation, and artefacts like model weights and biases. "Open source" should apply to the entire system, including models, parameters, and structural elements. However, it is unclear what mechanisms or licenses ensure that these models, particularly their parameters, are freely available for use, research, modification, and sharing. We recommend clear assertions accompanying parameter distribution to ensure that they remain freely accessible.

6. Increasing emphasis on open science and open source by funding agencies

As open science gains prominence, major funding agencies worldwide are implementing mandates to ensure that software developed with public funds is made openly accessible and reusable. Horizon Europe, the European Commission's flagship research program, has set

stringent requirements for open science, strongly recommending that research outputs, including software, are shared under open or free licenses aligned with FAIR principles (https://commission.europa.eu/about-european-commission/departments-and-executive-agencies/digital-services/open-source-software-strategy_en). Additionally, all Horizon Europe funded research is required to establish a data management plan (DMP), which is a structured document that outlines plans for open software and code sharing where possible, including tools needed for interoperability.

In the United States, agencies like the National Institutes of Health (NIH) and the National Science Foundation (NSF) strongly encourage, and in some cases require, software and code sharing through public repositories, aiming to maximize reproducibility and scientific transparency (https://datascience.nih.gov/tools-and-analytics/best-practices-for-sharing-research-software-faq). Similarly, the Wellcome Trust in the United Kingdom also recommends all research outputs, such as software integral to funded research, be available to ensure other research can verify it, build on it and use it to advance knowledge and make health improvements (https://wellcome.org/grant-funding/guidance/policies-grant-conditions/data-software-materials-management-and-sharing-policy). However, the same recommendations recognised that in some circumstances, controls and limits on sharing are necessary – for example, to protect the confidentiality and privacy of research participants, or to enable IP to be protected.

Many other funding agencies all over the world have similar open-source guidelines. This trend underscores a commitment from funders to foster collaborative scientific ecosystems, democratizing access to essential research tools and enhancing reproducibility across disciplines.

7. Challenges of Maintaining Open-Source Scientific Software

Open-source software in computational proteomics offers significant benefits but also poses challenges, particularly around sustainability. These challenges often deter long-term commitment, with some researchers transitioning to closed-source software after facing sustainability issues. Below, we outline key barriers to maintaining OSS and propose strategies—both practical and aspirational—to help advance OSS in the field.

7.1. Financial Sustainability

Maintaining an OSS project requires ongoing funding for updates, bug fixes, testing, and user support. However, funding agencies like the NIH often prioritize novelty over software maintenance, leaving many projects to become "abandonware" once the initial grant(s) end.

Problem: Without consistent funding, OSS projects in proteomics lose momentum after the initial development phase.

Potential Solutions:

- **Dedicated Maintenance Grants**: Funding agencies should offer grant mechanisms for software maintenance, such as the Chan Zuckerberg Initiative's "Essential Open-Source Software for Science" grants. For example, the NIH previously supported software maintenance through an R01 mechanism, and today has a program to support sustainable OSS projects (https://grants.nih.gov/grants/guide/rfa-files/RFA-OD-24-010.html) directly.
- **Commercialization Models**: OSS projects could explore commercialization, potentially leading to academic spin-offs or new revenue streams (read the section about commercialization strategies).

7.2. Misaligned Incentive Structures in Academia

The academic incentive structure prioritizes publications and novelty, encouraging researchers to develop new software instead of maintaining existing tools. Contributions to OSS, especially those owned by others, are undervalued and rarely recognized in tenure or promotion evaluations.

Problem: The "publish-or-perish" culture discourages OSS maintenance, as it doesn't align with traditional academic metrics.

Potential Solutions:

- **Recognition for OSS Contributions**: Institutions and funding agencies should acknowledge OSS maintenance as valuable scholarly work, similar to publications, and include it in grant and tenure evaluations, as is, for instance, the case in the European Commission's ERC programme CV template.
- **Community-driven Publications**: Journals should accept papers on software updates, offering academic recognition for maintenance work, as seen in the Journal of Proteome Research's Software Tools and Resources issue.

7.3. The Challenge of Consistent Maintainers

In academic settings, many OSS projects are led by students, postdocs, or temporary researchers who eventually leave for other opportunities, often in unrelated fields. This results in a lack of long-term maintainers, leading to project stagnation or abandonment.

Problem: The reliance on transient academic positions means OSS projects are vulnerable to disruptions as contributors move on.

Potential Solutions:

- **Governance Models**: Establishing community-driven governance structures, such as steering committees or core maintainer teams, can provide continuity even as individual contributors leave. Notably, this kind of governance is likely only feasible for larger, well-established open-source projects.
- **Transition Plans**: Projects should develop clear transition plans, ensuring that new maintainers can seamlessly take over. This could involve thorough documentation, onboarding guidelines, and mentoring new contributors.

Addressing these challenges requires a multi-pronged approach, combining changes in funding structures, academic incentives, and community engagement. The scientific community, funding agencies, companies, and academic institutions must collaborate to ensure that OSS can continue to thrive. By addressing these challenges head-on, we can build a more sustainable and collaborative ecosystem for open-source scientific software, ultimately driving innovation and reproducibility in proteomics research.

8. How to start a gold-standard OSS project in proteomics

Box 1: How to get started with OSS. The following steps provide a guideline that can foster a successful open-source project that grows in adoption, value, and contributions over time.

1- **Define clear goals and scope**: Start by defining the specific problem or gap your software aims to solve. Ensure it addresses an unmet need or provides a significant improvement over existing solutions. Before starting an independent OSS project, consider contributing to an existing OSS project by evaluating if your use case could take advantage of existing frameworks. For example, adding a new feature within Skyline or OpenMS would not require using your resources for implementing a raw data reading component and a user interface.

2- **Choose an open-source license**: Choose an OSI approved license that aligns with the project's intended use and desired level of openness. For projects that may later require commercialization or enterprise use, dual licensing (e.g., open source with an option for commercial licensing) can be considered to support sustainability.

3- **Plan for sustainability**: Research potential funding sources, such as grants, academic support, or partnerships. Decide if the project will rely on donations, grants, or if it might later incorporate paid services. If applicable, consider models like SaaS, support-based revenue, or feature-based licensing that could sustain the project without sacrificing its open-source nature.

4- **Set up a well-structured repository:** Use a version-control platform like GitHub or GitLab for easy access, community contributions, and versioning. Use clear folder structures, name conventions, and modular code design to enhance usability and maintainability. Provide a clear guide on how others can contribute to the project, including coding standards, pull request policies, and a Code of Conduct to foster a positive collaborative environment.

5- **Incorporate early user feedback**: Develop a prototype and engage a select group of users as beta testers than can try the software and provide feedback to ensure its usefulness and effectiveness.

6- **Implement rigorous testing and quality control**: Use CI/CD practices like GitHub Actions to automate testing and improve code quality. Create robust tests to ensure functionality and compatibility, and regularly review code with input from experienced contributors or collaborators.

7- Develop thorough documentation:
   a. **User documentation**: Provide tutorials, installation guides, and usage examples that lower barriers to entry for new users.
   b. **Developer documentation**: Include technical details that make it easier for new developers to understand the codebase, contribute, and debug.
   c. **Version control and changelog**: Maintain a detailed changelog for tracking updates, and consider using semantic versioning for releases to help users track changes and updates.

8- **Build a community**: Create forums, mailing lists, or a Slack channel to facilitate communication and support for users and contributors. Promote the project within academic and industry circles, social media, or conferences. Encourage diverse participation, whether from seasoned developers, scientists, or students, by being open to questions, feedback, and contributions of varying levels.

9- **Ensure long-term maintenance and evolution**: Provide a roadmap to outline planned features and long-term goals, keeping contributors aligned and users confident. Build an engaged community by recognizing contributors, hosting events, and welcoming new ideas. Adopt a governance model, such as a core maintainer group, to ensure the project's mission endures despite contributor changes.

10- **Monitor and measure success**: Track metrics like repository stars, downloads, citations, or code contributions to gauge adoption and impact. Regularly collect user feedback and address concerns or feature requests to ensure the project stays relevant and useful to its audience.

11- **Stable DOIs:** To prevent issues with license or code changes after publication, OSS projects should use archival platforms like Zenodo, Figshare, or Software Heritage, which offer DOIs for long-term citation and access. These platforms integrate with GitHub for automated, enduring accessibility.

9. Creation of Sustainable, Open-Source Software in an Academic Setting

A primary purpose of the academic laboratory is the training of graduate and post-doctoral students. These positions are by their nature, of limited duration. The creation and development of software tools can be an ideal mechanism for creating a deep understanding of the concepts and best practices of proteomics. However, tools created during training can languish following the graduation and departure of the student unless there is a considered and established plan for sustainability in place. We have established and maintained a procedure for sustainable software using the following established practices.

Box 2: One working approach to sustainable software development in an academic setting.

1- All students create code in the same language.
2- The language used by the lab should operate across major platforms (Windows, Linux, and MacOS).
3- All new code must make maximal use of existing code for efficiency.
4- Any new tools that are created by students or staff are incorporated into that code base, if possible, rather than downstream applications, so that they can be made use of in many projects.
5- All adaptations of existing code or newly created code must be covered by unit tests, that become a permanent part of the code base.
6- All new code must be reviewed and approved by an additional member of the team through code reviews.
7- All code must pass nightly build tests before public release.
8- New applications should be extensions of existing applications whenever possible.

The rationale for these rules is as follows. Choosing a single language for the laboratory means that all students will be well-versed and deeply knowledgeable in that language. This enables an easy understanding of existing code and the ability to understand the code written by other contributors. The re-use of an established codebase eventually results in robust, reliable, and bug-free operation. Moreover, all contributors become extremely conversant with the individual capabilities and their straightforward and facile integration into new tools. Student contributions are guided by the consensus of the group, being incorporated into our codebase where they make the most sense and with an eye toward their future use. The requirement for unit test coverage means that new code functions as expected and maintains the functionality of existing functions. The requirement for three reviews means that all code created in the lab is well understood by many other lab members. Therefore, when a student leaves the lab, there are many individuals still around who understand all that student's code and can maintain it moving forward. The requirement that students extend projects with new functionality rather than create stand-alone software provides an avenue to re-use established code with proven reliability, limiting potential bugs only to the new portions of code. The effect of some code changes cannot be predicted. Therefore, the use of nightly build tests, where many code operations are evaluated with large datasets enable the team to find unexpected

changes to the results or operation time. A key ancillary benefit of maximizing code re-use and minimizing new monolithic applications is the great reduction in the amount of code that needs maintenance over the long term. Code maintenance can require a significant investment of capital and human resources. Therefore, for the academic lab, a concerted effort to reduce the need for both of those precious resources is vital.

10. Strategies to commercialize OSS

Open-source software (OSS) is not free from costs; maintaining, running, and developing it requires resources. To ensure long-term sustainability, several commercialization strategies have been developed, balancing openness with financial viability, in a manner suiting the needs of the owner. Here, we consider "commercialization" as any means to monetize OSS, whether it remains in an academic setting, is adopted by a company, or spun out into a startup. We argue that healthy OSS projects must be financially supported by methods such as charitable means, grants, or commercialization, in order for the development of the project to be sustainable. We discuss a few commercialization models that have become popular with OSS, which try to strike a balance between supporting openness and supporting future development. It is worth noting that these strategies are not necessarily mutually exclusive.

- **Dual licensing**: A popular commercialization option for OSS has been to offer the software under both a strong copyleft license (like GPL or AGPL) and a more permissive commercial license. The code itself is typically the same for both license types. The difference lies in how the code can be used, modified, and redistributed depending on the license under which it is acquired. Projects using this strategy are often available under a strong copyleft license (GPL, AGPL, etc.) with no financial cost. However, the copyleft nature of these licenses requires any derivative works to be published under a compatible open-source license, which is often undesirable for corporate users. Thus, projects also offer more permissive commercial licenses to paying customers, allowing them to use the OSS project within proprietary code. Although this approach may seem prone to abuse (e.g., improper use of GPL code), our experience has been that companies tend to be risk-averse and prefer purchasing proper licenses to avoid violating a copyleft license. A successful example of this strategy from outside of proteomics has been RStudio by Posit. RStudio is currently available under an open-source AGPLv3 license, or under a commercial license when AGPLv3 is incompatible. Notably, developers should make sure to include a "contributor license agreement" as part of their requirements for new contributors to ensure their contributions can be distributed under both licenses.

- **Support or services**: Some OSS projects commercialize by offering support services or new feature development at a cost. Often users, particularly from corporate entities, are willing to pay for specialized training and ongoing support for their use of OSS projects. In special instances, it may even be the case that outside entities are able to pay for the prioritization of specific features. For example, the major mass spectrometry instrument vendors have been providing financial support to both the Skyline and Proteowizard projects to ensure features, support, and documentation are provided for their customers. This road must be trod carefully though; while there is a benefit to allowing sponsored features, and they do benefit everyone once implemented; such a model risks losing control over the direction of an OSS project. Features added to Skyline from a vendor are made available to all vendors if they have compatible instrumentation. Red Hat is the most prominent example of a company using this strategy to commercialize their enterprise Linux offering.
- **Software as a service (SaaS)**: The SaaS commercialization model has become increasingly popular in recent times. When using a SaaS model, the OSS project remains open source, but commercialization occurs by building a platform around it. The platform then allows users to more easily use the OSS project. This model often includes a managed hardware or cloud infrastructure component, where users pay to interact with a web application to use the OSS tool, reducing the barrier to entry. In the bioinformatics space, NextFlow (59) is an open-source bioinformatics workflow engine that has been commercialized by Seqera Labs using the SaaS model. Their current Seqera Platform product provides an interface to launch, observe, and explore workflow executions with NextFlow, in addition to other features.
- **Open-core**: The open-core commercialization model provides access to new features only to paying customers. Rather than essential functionality, this refers to optional features such as a nicer user interface or early access to new features. Some variants of this model use a time delay for new features, where paying users have access to new features sooner than those using the fully OSS version. Practically, the implementation of this strategy often involves the creation of a private, upstream fork of the OSS code repository. New features are then added to the private fork and synced to the OSS version at a later date. Such a strategy can also be used by academic labs looking to protect new features while preparing for publication and until a manuscript is accepted. Although we advocate for developing those features in the open, we recognize that there are instances where this is not practical. For example, when a junior researcher is publishing a novel algorithm, they may want to avoid the risk of having their work pre-empted by others. Similarly, collaborators may request that the software be kept private to prevent other researchers from using it and publishing their

findings first. While we believe that these situations are rare in the proteomics community, they could lead to the original researchers losing recognition and credit for their work. The open-core model is quite common, and in proteomics, it is used for ScaffoldDIA from Proteome Software: the open-source core of ScaffoldDIA is EncyclopeDIA (40).

10. Concluding remarks

As proteomics increasingly depends on computational tools, adopting open-source and FAIR principles is crucial for ensuring transparency, reproducibility, and accessibility. We urge researchers, funding agencies, institutions, and companies to prioritize open-source practices, particularly for publicly funded work, to foster a truly collaborative scientific ecosystem. By collectively advancing open-source software, the scientific community can build an inclusive, rigorous foundation that fosters innovation and extends the benefits of research to scientists and the public alike.

Moving forward, we as a community should explore mechanisms to make OSS sustainable, for example, by creating a foundation for proteomics software to support the maintenance of OSS in our field. Emphasizing scalable, user-friendly software with complex features hidden behind intuitive interfaces will help ensure widespread adoption and success (60). This approach can also counteract negative perceptions of the quality of academic or OSS in mass spectrometry (37). Additionally, we expect that AI-assisted software development will enhance the quality of proteomics OSS by automating error detection, optimizing code performance, and enhancing feature integration—ultimately boosting reliability and user satisfaction. Regardless, let us unite in our commitment to open science and pursue a shared, sustainable future in our exploration of the proteome.

Acknowledgements

Y.P.-R. is funded by Wellcome grants (numbers 208391/Z/17/Z, 223745/Z/21/Z) and EMBL core funding. T.S. acknowledges funding by the Federal Ministry of Education and Research in the frame of de.NBI/ELIXIR-DE (W-de.NBI-022) and is supported by the Ministry of Science, Research and Arts Baden-Württemberg. MJM acknowledges financial support from National Institutes of Health grants R24 GM141156, U01 DK137097, and U19 AG065156. R.B. and L.M. acknowledge funding from the Research Foundation Flanders (FWO) (12A6L24N, G010023N, and G028821N). L.M. acknowledges funding from the Ghent University Concerted Research Action (BOF21/GOA/033) and the European Union's Horizon Europe Programme (101080544, 101103253, 101195186, and 10119173). L.K. acknowledges funding from the Swedish Research Council (VR 2024-05887)

Conflicts of interest

W.E.F. is an employee of Talus Bioscience Inc., a drug-discovery biotechnology company that develops and contributes to OSS and does not currently sell software. Additionally, Talus Bioscience has a collaborative research agreement with Bruker. T.S. is an officer in OpenMS Inc., a non-profit foundation that manages the international coordination of OpenMS development. MRL is an employee of Belharra Therapeutics, Inc., and an officer of Chaparral Labs, Inc., a company offering SaaS solutions for proteomics, in addition to commercial support for OSS software. JVG is an employee of InstaDeep Ltd. The MacCoss Lab at the University of Washington receives funding from Agilent, Bruker, Sciex, Shimadzu, Thermo Fisher Scientific, and Waters to support the development of Skyline, an open-source software tool for quantitative proteomics. MJM is a paid consultant for Thermo Fisher Scientific. The CompOmics group at Ghent University and VIB (RB and LM) receives funding from Bruker.

References

1. Hettrick, S.; Antonioletti, M.; Carr, L.; Chue Hong, N.; Crouch, S.; De Roure, D. C.; Emsley, I.; Goble, C.; Hay, A.; Inupakutika, D., UK research software survey 2014. **2014**.
2. Jimenez, R. C.; Kuzak, M.; Alhamdoosh, M.; Barker, M.; Batut, B.; Borg, M.; Capella-Gutierrez, S.; Chue Hong, N.; Cook, M.; Corpas, M.; Flannery, M.; Garcia, L.; Gelpi, J. L.; Gladman, S.; Goble, C.; Gonzalez Ferreiro, M.; Gonzalez-Beltran, A.; Griffin, P. C.; Gruning, B.; Hagberg, J.; Holub, P.; Hooft, R.; Ison, J.; Katz, D. S.; Leskosek, B.; Lopez Gomez, F.; Oliveira, L. J.; Mellor, D.; Mosbergen, R.; Mulder, N.; Perez-Riverol, Y.; Pergl, R.; Pichler, H.; Pope, B.; Sanz, F.; Schneider, M. V.; Stodden, V.; Suchecki, R.; Svobodova Varekova, R.; Talvik, H. A.; Todorov, I.; Treloar, A.; Tyagi, S.; van Gompel, M.; Vaughan, D.; Via, A.; Wang, X.; Watson-Haigh, N. S.; Crouch, S., Four simple recommendations to encourage best practices in research software. *F1000Res* **2017,** 6.
3. Perez-Riverol, Y.; Wang, R.; Hermjakob, H.; Muller, M.; Vesada, V.; Vizcaino, J. A., Open source libraries and frameworks for mass spectrometry based proteomics: a developer's perspective. *Biochim Biophys Acta* **2014,** 1844, (1 Pt A), 63-76.
4. Halder, A.; Verma, A.; Biswas, D.; Srivastava, S., Recent advances in mass-spectrometry based proteomics software, tools and databases. *Drug Discov Today Technol* **2021,** 39, 69-79.
5. Eng, J. K.; McCormack, A. L.; Yates, J. R., An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *J Am Soc Mass Spectrom* **1994,** 5, (11), 976-89.

6. Kall, L.; Canterbury, J. D.; Weston, J.; Noble, W. S.; MacCoss, M. J., Semi-supervised learning for peptide identification from shotgun proteomics datasets. *Nat Methods* **2007,** 4, (11), 923-5.

7. Degroeve, S.; Martens, L., MS2PIP: a tool for MS/MS peak intensity prediction. *Bioinformatics* **2013,** 29, (24), 3199-203.

8. Declercq, A.; Bouwmeester, R.; Chiva, C.; Sabido, E.; Hirschler, A.; Carapito, C.; Martens, L.; Degroeve, S.; Gabriels, R., Updated MS(2)PIP web server supports cutting-edge proteomics applications. *Nucleic Acids Res* **2023,** 51, (W1), W338-W342.

9. Gessulat, S.; Schmidt, T.; Zolg, D. P.; Samaras, P.; Schnatbaum, K.; Zerweck, J.; Knaute, T.; Rechenberger, J.; Delanghe, B.; Huhmer, A.; Reimer, U.; Ehrlich, H. C.; Aiche, S.; Kuster, B.; Wilhelm, M., Prosit: proteome-wide prediction of peptide tandem mass spectra by deep learning. *Nat Methods* **2019,** 16, (6), 509-518.

10. Declercq, A.; Bouwmeester, R.; Hirschler, A.; Carapito, C.; Degroeve, S.; Martens, L.; Gabriels, R., MS(2)Rescore: Data-Driven Rescoring Dramatically Boosts Immunopeptide Identification Rates. *Mol Cell Proteomics* **2022,** 21, (8), 100266.

11. Chambers, M. C.; Maclean, B.; Burke, R.; Amodei, D.; Ruderman, D. L.; Neumann, S.; Gatto, L.; Fischer, B.; Pratt, B.; Egertson, J.; Hoff, K.; Kessner, D.; Tasman, N.; Shulman, N.; Frewen, B.; Baker, T. A.; Brusniak, M. Y.; Paulse, C.; Creasy, D.; Flashner, L.; Kani, K.; Moulding, C.; Seymour, S. L.; Nuwaysir, L. M.; Lefebvre, B.; Kuhlmann, F.; Roark, J.; Rainer, P.; Detlev, S.; Hemenway, T.; Huhmer, A.; Langridge, J.; Connolly, B.; Chadick, T.; Holly, K.; Eckels, J.; Deutsch, E. W.; Moritz, R. L.; Katz, J. E.; Agus, D. B.; MacCoss, M.; Tabb, D. L.; Mallick, P., A cross-platform toolkit for mass spectrometry and proteomics. *Nat Biotechnol* **2012,** 30, (10), 918-20.

12. Chambers, M. C.; Jagtap, P. D.; Johnson, J. E.; McGowan, T.; Kumar, P.; Onsongo, G.; Guerrero, C. R.; Barsnes, H.; Vaudel, M.; Martens, L.; Gruning, B.; Cooke, I. R.; Heydarian, M.; Reddy, K. L.; Griffin, T. J., An Accessible Proteogenomics Informatics Resource for Cancer Researchers. *Cancer Res* **2017,** 77, (21), e43-e46.

13. Dai, C.; Pfeuffer, J.; Wang, H.; Zheng, P.; Käll, L.; Sachsenberg, T.; Demichev, V.; Bai, M.; Kohlbacher, O.; Perez-Riverol, Y., quantms: a cloud-based pipeline for quantitative proteomics enables the reanalysis of public proteomics data. *Nature Methods* **2024**.

14. Deutsch, E. W.; Mendoza, L.; Shteynberg, D. D.; Hoopmann, M. R.; Sun, Z.; Eng, J. K.; Moritz, R. L., Trans-Proteomic Pipeline: Robust Mass Spectrometry-Based Proteomics Data Analysis Suite. *J Proteome Res* **2023,** 22, (2), 615-624.

15. Bray, S.; Chilton, J.; Bernt, M.; Soranzo, N.; van den Beek, M.; Batut, B.; Rasche, H.; Cech, M.; Cock, P. J. A.; Gruning, B.; Nekrutenko, A., The Planemo toolkit for developing, deploying, and executing scientific data analyses in Galaxy and beyond. *Genome Res* **2023,** 33, (2), 261-268.

16. Martens, L.; Nesvizhskii, A. I.; Hermjakob, H.; Adamski, M.; Omenn, G. S.; Vandekerckhove, J.; Gevaert, K., Do we want our data raw? Including binary mass spectrometry data in public proteomics data repositories. *Proteomics* **2005,** 5, (13), 3501-5.

17. Hulstaert, N.; Shofstahl, J.; Sachsenberg, T.; Walzer, M.; Barsnes, H.; Martens, L.; Perez-Riverol, Y., ThermoRawFileParser: Modular, Scalable, and Cross-Platform RAW File Conversion. *J Proteome Res* **2020,** 19, (1), 537-542.

18. Perez-Riverol, Y.; Bandla, C.; Kundu, D. J.; Kamatchinathan, S.; Bai, J.; Hewapathirana, S.; John, N. S.; Prakash, A.; Walzer, M.; Wang, S.; Vizcaino, J. A., The PRIDE database at 20 years: 2025 update. *Nucleic Acids Res* **2024**.

19. Deutsch, E. W.; Perez-Riverol, Y.; Carver, J.; Kawano, S.; Mendoza, L.; Van Den Bossche, T.; Gabriels, R.; Binz, P. A.; Pullman, B.; Sun, Z.; Shofstahl, J.; Bittremieux, W.; Mak, T. D.; Klein, J.; Zhu, Y.; Lam, H.; Vizcaino, J. A.; Bandeira, N., Universal Spectrum Identifier for mass spectra. *Nat Methods* **2021,** 18, (7), 768-770.

20. Lazear, M. R., Sage: An Open-Source Tool for Fast Proteomics Searching and Quantification at Scale. *J Proteome Res* **2023,** 22, (11), 3652-3659.

21. Wilkinson, M. D.; Dumontier, M.; Aalbersberg, I. J.; Appleton, G.; Axton, M.; Baak, A.; Blomberg, N.; Boiten, J. W.; da Silva Santos, L. B.; Bourne, P. E.; Bouwman, J.; Brookes, A. J.; Clark, T.; Crosas, M.; Dillo, I.; Dumon, O.; Edmunds, S.; Evelo, C. T.; Finkers, R.; Gonzalez-Beltran, A.; Gray, A. J.; Groth, P.; Goble, C.; Grethe, J. S.; Heringa, J.; t Hoen, P. A.; Hooft, R.; Kuhn, T.; Kok, R.; Kok, J.; Lusher, S. J.; Martone, M. E.; Mons, A.; Packer, A. L.; Persson, B.; Rocca-Serra, P.; Roos, M.; van Schaik, R.; Sansone, S. A.; Schultes, E.; Sengstag, T.; Slater, T.; Strawn, G.; Swertz, M. A.; Thompson, M.; van der Lei, J.; van Mulligen, E.; Velterop, J.; Waagmeester, A.; Wittenburg, P.; Wolstencroft, K.; Zhao, J.; Mons, B., The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data* **2016,** 3, 160018.

22. Barker, M.; Chue Hong, N. P.; Katz, D. S.; Lamprecht, A. L.; Martinez-Ortiz, C.; Psomopoulos, F.; Harrow, J.; Castro, L. J.; Gruenpeter, M.; Martinez, P. A.; Honeyman, T., Introducing the FAIR Principles for research software. *Sci Data* **2022,** 9, (1), 622.

23. de Visser, C.; Johansson, L. F.; Kulkarni, P.; Mei, H.; Neerincx, P.; Joeri van der Velde, K.; Horvatovich, P.; van Gool, A. J.; Swertz, M. A.; Hoen, P. A. C.; Niehues, A., Ten quick tips for building FAIR workflows. *PLoS Comput Biol* **2023,** 19, (9), e1011369.

24. In *Open Science by Design: Realizing a Vision for 21st Century Research*, Washington (DC), 2018.

25. Li, H.; Handsaker, B.; Wysoker, A.; Fennell, T.; Ruan, J.; Homer, N.; Marth, G.; Abecasis, G.; Durbin, R.; Genome Project Data Processing, S., The Sequence Alignment/Map format and SAMtools. *Bioinformatics* **2009,** 25, (16), 2078-9.

26. McKenna, A.; Hanna, M.; Banks, E.; Sivachenko, A.; Cibulskis, K.; Kernytsky, A.; Garimella, K.; Altshuler, D.; Gabriel, S.; Daly, M.; DePristo, M. A., The Genome Analysis

Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res* **2010,** 20, (9), 1297-303.

27. Pfeuffer, J.; Bielow, C.; Wein, S.; Jeong, K.; Netz, E.; Walter, A.; Alka, O.; Nilse, L.; Colaianni, P. D.; McCloskey, D.; Kim, J.; Rosenberger, G.; Bichmann, L.; Walzer, M.; Veit, J.; Boudaud, B.; Bernt, M.; Patikas, N.; Pilz, M.; Startek, M. P.; Kutuzova, S.; Heumos, L.; Charkow, J.; Sing, J. C.; Feroz, A.; Siraj, A.; Weisser, H.; Dijkstra, T. M. H.; Perez-Riverol, Y.; Rost, H.; Kohlbacher, O.; Sachsenberg, T., OpenMS 3 enables reproducible analysis of large-scale mass spectrometry data. *Nat Methods* **2024,** 21, (3), 365-367.

28. Yang, K. L.; Yu, F.; Teo, G. C.; Li, K.; Demichev, V.; Ralser, M.; Nesvizhskii, A. I., MSBooster: improving peptide identification rates using deep learning-based features. *Nat Commun* **2023,** 14, (1), 4539.

29. Li, K.; Jain, A.; Malovannaya, A.; Wen, B.; Zhang, B., DeepRescore: Leveraging Deep Learning to Improve Peptide Identification in Immunopeptidomics. *Proteomics* **2020,** 20, (21-22), e1900334.

30. Kertesz-Farkas, A.; Nii Adoquaye Acquaye, F. L.; Bhimani, K.; Eng, J. K.; Fondrie, W. E.; Grant, C.; Hoopmann, M. R.; Lin, A.; Lu, Y. Y.; Moritz, R. L.; MacCoss, M. J.; Noble, W. S., The Crux Toolkit for Analysis of Bottom-Up Tandem Mass Spectrometry Proteomics Data. *J Proteome Res* **2023,** 22, (2), 561-569.

31. Granholm, V.; Kim, S.; Navarro, J. C.; Sjolund, E.; Smith, R. D.; Kall, L., Fast and accurate database searches with MS-GF+Percolator. *J Proteome Res* **2014,** 13, (2), 890-7.

32. Brosch, M.; Yu, L.; Hubbard, T.; Choudhary, J., Accurate and sensitive peptide identification with Mascot Percolator. *J Proteome Res* **2009,** 8, (6), 3176-81.

33. Wen, B.; Li, G.; Wright, J. C.; Du, C.; Feng, Q.; Xu, X.; Choudhary, J. S.; Wang, J., The OMSSAPercolator: an automated tool to validate OMSSA results. *Proteomics* **2014,** 14, (9), 1011-4.

34. MacLean, B.; Tomazela, D. M.; Shulman, N.; Chambers, M.; Finney, G. L.; Frewen, B.; Kern, R.; Tabb, D. L.; Liebler, D. C.; MacCoss, M. J., Skyline: an open source document editor for creating and analyzing targeted proteomics experiments. *Bioinformatics* **2010,** 26, (7), 966-8.

35. Eng, J. K.; Jahan, T. A.; Hoopmann, M. R., Comet: an open-source MS/MS sequence database search tool. *Proteomics* **2013,** 13, (1), 22-4.

36. Vaudel, M.; Burkhart, J. M.; Zahedi, R. P.; Oveland, E.; Berven, F. S.; Sickmann, A.; Martens, L.; Barsnes, H., PeptideShaker enables reanalysis of MS-derived proteomics data sets. *Nat Biotechnol* **2015,** 33, (1), 22-4.

37. Smith, R., Conversations with 100 Scientists in the Field Reveal a Bifurcated Perception of the State of Mass Spectrometry Software. *J Proteome Res* **2018,** 17, (4), 1335-1339.

38. Craig, R.; Beavis, R. C., TANDEM: matching proteins with tandem mass spectra. *Bioinformatics* **2004,** 20, (9), 1466-7.

39. Kong, A. T.; Leprevost, F. V.; Avtonomov, D. M.; Mellacheruvu, D.; Nesvizhskii, A. I., MSFragger: ultrafast and comprehensive peptide identification in mass spectrometry-based proteomics. *Nat Methods* **2017,** 14, (5), 513-520.

40. Searle, B. C.; Pino, L. K.; Egertson, J. D.; Ting, Y. S.; Lawrence, R. T.; MacLean, B. X.; Villen, J.; MacCoss, M. J., Chromatogram libraries improve peptide detection and quantification by data independent acquisition mass spectrometry. *Nat Commun* **2018,** 9, (1), 5128.

41. Wen, B.; Wang, X.; Zhang, B., PepQuery enables fast, accurate, and convenient proteomic validation of novel genomic alterations. *Genome Res* **2019,** 29, (3), 485-493.

42. Wallmann, G.; Skowronek, P.; Brennsteiner, V.; Lebedev, M.; Thielert, M.; Steigerwald, S.; Kotb, M.; Heymann, T.; Zhou, X.-X.; Schwoerer, M., AlphaDIA enables End-to-End Transfer Learning for Feature-Free Proteomics. *bioRxiv* **2024**, 2024.05. 28.596182.

43. Styczynski, M. P.; Jensen, K. L.; Rigoutsos, I.; Stephanopoulos, G., BLOSUM62 miscalculations improve search performance. *Nat Biotechnol* **2008,** 26, (3), 274-5.

44. Kessner, D.; Chambers, M.; Burke, R.; Agus, D.; Mallick, P., ProteoWizard: open source software for rapid proteomics tools development. *Bioinformatics* **2008,** 24, (21), 2534-6.

45. Adusumilli, R.; Mallick, P., Data Conversion with ProteoWizard msConvert. *Methods Mol Biol* **2017,** 1550, 339-368.

46. Vaudel, M.; Barsnes, H.; Berven, F. S.; Sickmann, A.; Martens, L., SearchGUI: An open-source graphical user interface for simultaneous OMSSA and X!Tandem searches. *Proteomics* **2011,** 11, (5), 996-9.

47. Li, H.; Durbin, R., Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* **2009,** 25, (14), 1754-60.

48. Hanssen, F.; Garcia, M. U.; Folkersen, L.; Pedersen, A. S.; Lescai, F.; Jodoin, S.; Miller, E.; Seybold, M.; Wacker, O.; Smith, N.; Gabernet, G.; Nahnsen, S., Scalable and efficient DNA sequencing analysis on different compute infrastructures aiding variant discovery. *NAR Genom Bioinform* **2024,** 6, (2), lqae031.

49. Rost, H. L.; Schmitt, U.; Aebersold, R.; Malmstrom, L., pyOpenMS: a Python-based interface to the OpenMS mass-spectrometry algorithm library. *Proteomics* **2014,** 14, (1), 74-7.

50. Bittremieux, W., spectrum_utils: A Python Package for Mass Spectrometry Data Processing and Visualization. *Anal Chem* **2020,** 92, (1), 659-661.

51. Raymond, E. S. Release Early, Release Often. http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ar01s04.html

52. Fondrie, W. E.; Noble, W. S., mokapot: Fast and Flexible Semisupervised Learning for Peptide Detection. *J Proteome Res* **2021,** 20, (4), 1966-1971.

53. Zhou, X. X.; Zeng, W. F.; Chi, H.; Luo, C.; Liu, C.; Zhan, J.; He, S. M.; Zhang, Z., pDeep: Predicting MS/MS Spectra of Peptides with Deep Learning. *Anal Chem* **2017,** 89, (23), 12690-12697.

54. Zeng, W. F.; Zhou, X. X.; Willems, S.; Ammar, C.; Wahle, M.; Bludau, I.; Voytik, E.; Strauss, M. T.; Mann, M., AlphaPeptDeep: a modular deep learning framework to predict peptide properties for proteomics. *Nat Commun* **2022,** 13, (1), 7238.

55. Bouwmeester, R.; Gabriels, R.; Hulstaert, N.; Martens, L.; Degroeve, S., DeepLC can predict retention times for peptides that carry as-yet unseen modifications. *Nat Methods* **2021,** 18, (11), 1363-1369.

56. Wen, B.; Li, K.; Zhang, Y.; Zhang, B., Cancer neoantigen prioritization through sensitive and reliable proteogenomics analysis. *Nat Commun* **2020,** 11, (1), 1759.

57. Yilmaz, M.; Fondrie, W. E.; Bittremieux, W.; Melendez, C. F.; Nelson, R.; Ananth, V.; Oh, S.; Noble, W. S., Sequence-to-sequence translation from mass spectra to peptides with a transformer model. *Nat Commun* **2024,** 15, (1), 6427.

58. Eloff, K.; Kalogeropoulos, K.; Morell, O.; Mabona, A.; Jespersen, J. B.; Williams, W.; van Beljouw, S. P.; Skwark, M.; Laustsen, A. H.; Brouns, S. J., De novo peptide sequencing with InstaNovo: Accurate, database-free peptide identification for large scale proteomics experiments. *bioRxiv* **2023**, 2023.08. 30.555055.

59. Di Tommaso, P.; Chatzou, M.; Floden, E. W.; Barja, P. P.; Palumbo, E.; Notredame, C., Nextflow enables reproducible computational workflows. *Nat Biotechnol* **2017,** 35, (4), 316-319.

60. Ross, D. H.; Bhotika, H.; Zheng, X.; Smith, R. D.; Burnum-Johnson, K. E.; Bilbao, A., Computational tools and algorithms for ion mobility spectrometry-mass spectrometry. *Proteomics* **2024,** 24, (12-13), e2200436.