The Journal of Open Source Software

ProteoFlux: a modality-aware framework for downstream analysis of quantitative proteomics data

Dariush Mollet [1] **and Alexander Schmidt** [1]

1 Biozentrum, University of Basel, Basel, Switzerland

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 17 February 2026
Published: unpublished

Summary

ProteoFlux is an open-source Python framework for downstream analysis of quantitative proteomics data, operating on outputs generated by upstream peptide and protein identification and quantification tools. It provides a unified, modality-aware analysis workflow supporting protein-centric proteomics, peptide-centric analyses, and phosphoproteomics with optional protein-level covariate adjustment. ProteoFlux emphasizes explicit data semantics, transparent preprocessing, and reproducible statistical modeling, making it well suited for routine analysis and re-analysis in core facility and high-throughput settings. Results are stored in standardized, fully annotated AnnData objects that preserve raw data, intermediate representations, and statistical outputs. By decoupling analysis from visualization, ProteoFlux enables flexible downstream exploration and reporting, with ProteoViewer provided as an example consumer of the analysis-ready outputs.

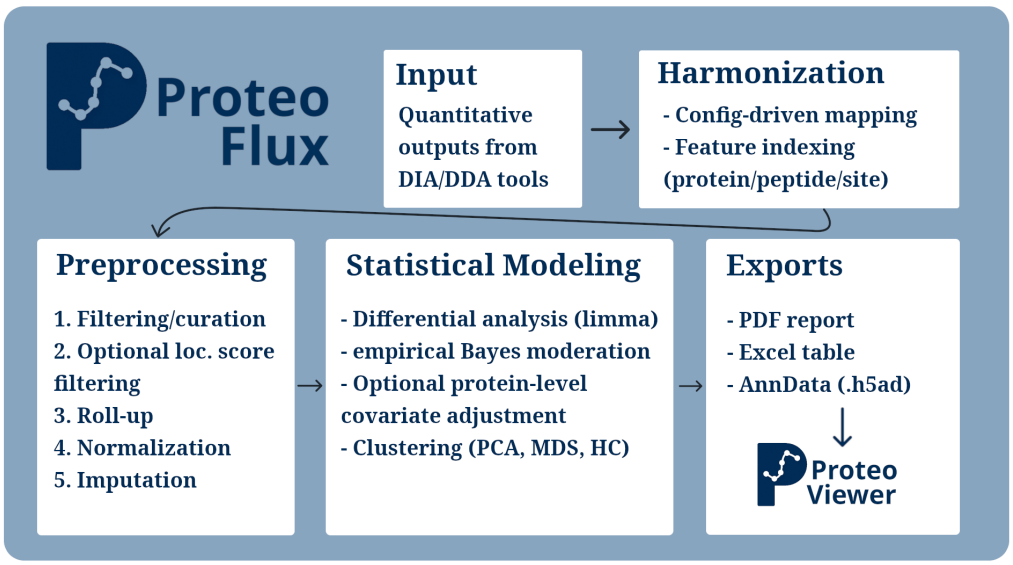

Figure 1: Figure 1: Overview of the ProteoFlux workflow. Quantitative outputs from upstream identification tools are harmonized via configuration into a canonical representation, followed by preprocessing, statistical modeling, and standardized exports for downstream visualization and reporting.

Statement of need

Modern quantitative proteomics experiments routinely generate measurements across multiple biological modalities, including proteins, peptides, and post-translational modifications. While

upstream identification and quantification tools have matured, downstream analysis remains fragmented and difficult to standardize across experiment types and data sources.

In practice, downstream proteomics analysis is often modality-specific and tightly coupled to individual tools or vendor ecosystems. Preprocessing, statistical modeling, and visualization are frequently intertwined, making analyses hard to inspect, modify, or reproduce consistently. Re-analysis of existing datasets, as well as analysis of new experiments following similar designs, often requires re-implementing custom scripts or adapting workflows originally developed for a different context. This variability complicates comparison across experiments and increases the risk of unintentional analytical bias.

These challenges are particularly acute for proteomics core facilities and advanced users, who routinely process large numbers of related experiments under comparable designs. Maintaining consistent preprocessing and statistical assumptions across analyses is critical to limit analytical variance and avoid selective parameter tuning. This requires software that makes preprocessing and modeling decisions explicit, produces stable and inspectable outputs, and clearly separates analysis from visualization and reporting.

ProteoFlux was developed to address these needs by providing a unified downstream framework with explicit configuration, deterministic input handling, early error detection, and transparent reporting at each stage of the pipeline.

State of the field

Several established tools exist for downstream proteomics analysis, including MSstats and related R-based workflows (Kohler et al. 2023; Michalik et al. 2025; Zhu et al. 2020). These tools provide robust statistical modeling frameworks and are widely adopted.

However, adapting such workflows across modalities or integrating advanced preprocessing and covariate-aware analyses often requires custom scripting. In many environments, preprocessing, statistical modeling, and visualization are implemented within partially overlapping toolchains, complicating reuse and consistent application across heterogeneous experiments.

ProteoFlux was designed to provide a single modality-aware downstream framework that standardizes harmonization, preprocessing, statistical modeling, and output generation across experiment types. Rather than replacing established statistical methodology, it integrates established approaches (e.g., limma-based modeling and directLFQ) within an explicitly configured Python pipeline and exports standardized analysis objects intended for downstream consumers, including ProteoViewer.

Software Design

Overall Design Philosophy

ProteoFlux is a downstream-only framework. It operates on quantitative outputs produced by upstream identification and quantification tools and does not perform peptide or protein identification or signal extraction. This scope avoids overlap with vendor- or search-engine–specific software and focuses on reproducible preprocessing and statistical modeling.

Experimental semantics such as conditions, replicates, feature identities, and quantitative signals are defined explicitly through configuration and interpreted deterministically. Inconsistent inputs trigger informative errors or explicit downgrade modes (e.g. pilot-study handling), rather than silent fallbacks. Harmonization, preprocessing, statistical modeling, and outputs are strictly separated.

Input Handling and Harmonization

ProteoFlux supports long- and wide-format quantitative tables from tools such as Spectronaut, FragPipe, and DIA-NN (Bruderer et al. 2015; Demichev et al. 2020; Yu et al. 2020). Input handling is driven by user-defined configuration files specifying column semantics, experimental layout, and analysis type.

Wide-format inputs are reshaped deterministically. When available, external annotation files define authoritative sample metadata; otherwise, identifiers are derived from configuration-defined patterns.

For peptide-centric and phosphoproteomics workflows, feature identities are constructed during harmonization. Modified peptide sequences are normalized by removing non-defining variable modifications. Phosphosite identifiers combine parent protein identifiers with absolute modification positions. Localization probabilities are retained for filtering.

Phosphoproteomics analysis is performed primarily at the phosphosite level. When the same site appears in multiple peptide contexts, measurements are collapsed using an explicit aggregation rule. For multi-site peptides, an "explode multisite" policy assigns quantitative values to each constituent site, acknowledging non-independence (Müller et al. 2025). Alternatively, a retain mode preserves multi-site peptides as single features.

Filtering, quantification, and normalization

Filtering is applied at the precursor level prior to aggregation. Quality criteria (e.g., PEP, q-values, precursor evidence counts) are applied conservatively when available. Contaminant filtering and optional low-intensity censoring are recorded explicitly. ProteoFlux does not filter features based on missingness across samples; all detected features are retained to maintain rectangular matrices and enable contrast-consistent tracking.

Precursor intensities are aggregated to peptides and proteins using explicit roll-up strategies. Protein quantification can be performed via summation or via directLFQ (Ammar et al. 2023), with user-defined inclusion criteria. All quantification parameters are recorded in metadata.

Normalization is applied after log transformation by default, using conservative median-based scaling (Callister et al. 2006; Cox et al. 2014; Karpievitch et al. 2012).

Imputation

Missing values are retained throughout preprocessing and handled explicitly at the imputation stage. ProteoFlux provides multiple imputation strategies (Shah et al. 2017; Wei et al. 2018). The default method, LC-ConMed, implements a conservative, condition-aware left-censoring strategy for log-scale proteomics data.

LC-ConMed performs imputation conditional on experimental grouping. If at least a user-defined minimum number of observations is present (default: one), missing values are sampled around the within-condition median using variance derived from pooled within-group variance across all conditions. Sampling width is constrained and imputed values are clipped to the empirical inter-quantile range of observed values for that condition, preserving condition structure while stabilizing variance for moderated modeling.

If insufficient observations are available within a condition, missing values are imputed near a global limit of detection estimated from the lower tail of the intensity distribution. These values are shifted below the detection limit, jittered with small variance, and constrained to remain below the global threshold, ensuring consistent treatment of left-censored features.

All intermediate matrices (non-imputed, log-transformed, normalized, imputed) are preserved as separate AnnData layers.

Statistical modeling

Differential analysis uses a limma-based linear modeling framework via the inmoose Python port (Ritchie et al. 2015; Colange et al. 2025), fitting expression ~ condition models without intercept. All pairwise contrasts are supported. Raw and empirical Bayes–moderated statistics are retained.

In phosphoproteomics, optional protein-level covariate adjustment is implemented via a two-stage residualization approach equivalent to ANCOVA without interaction (Wu et al. 2011). Phosphosite intensities are first regressed against matched protein-level abundances with a fixed slope of one, reflecting the assumption of a proportional (1:1) relationship between protein abundance and phosphosite signal; residuals are then analyzed using the same limma framework. Adjustment is suppressed when protein-level measurements are insufficient to support regression. For transparency, the decomposition raw log2FC ~ adjusted log2FC + covariate contribution is stored.

Pilot-study scenarios (e.g., single condition or replicate) are detected and formal testing disabled.

Fully imputed contrasts are detected on a per-feature, per-contrast basis using the pre-imputation intensity matrices. For such contrasts, statistical outputs are neutralized explicitly (zero fold change, unit p-value) prior to multiple-testing correction, ensuring consistent and interpretable downstream results.

Clustering and dimensionality reduction

ProteoFlux computes principal component analysis (PCA), multidimensional scaling (MDS), and hierarchical clustering on normalized expression matrices during analysis and stores embeddings and linkage structures directly in the AnnData object. Missingness-based clustering is computed separately to capture structure driven by data completeness.

For unsupervised analysis, matrices are fully imputed using a conservative, context-independent left-censoring strategy to rovide a contrast-independent representation suitable for quality control and exploratory visualization. This imputation layer is used exclusively for clustering and embedding and does not affect differential testing.

Outputs and provenance

The primary output is a compressed AnnData (.h5ad) object containing:

- Experimental design and metadata
- Raw, log-transformed, normalized, and imputed matrices
- Differential expression results (raw and empirical Bayes–moderated statistics)
- Residualized matrices and covariate components where applicable
- PCA and MDS embeddings
- Hierarchical clustering results
- Metadata describing preprocessing, modeling parameters, software versions, and analysis mode

Intermediate representations are preserved as separate layers to enable inspection without re-running the pipeline.

Additional exports include:

- A structured Excel results file integrating annotations, fold changes, statistical measures, missingness summaries, and intensities, with additional tabs for identification statistics and peptide tables.

- A static PDF report summarizing preprocessing decisions, filtering outcomes, normalization settings, and quality metrics, including hierarchical clustering and volcano plots. The report layout is configurable.

Visualization Decoupling

ProteoFlux writes all preprocessing results, statistical models, embeddings, and derived quantities into a single AnnData object, separating analysis from visualization.

ProteoViewer is a companion application that consumes ProteoFlux outputs without performing additional preprocessing or statistical computation. It can be deployed as a local desktop application or as a server-based web service.

ProteoViewer enables interactive exploration of:

- Quality control and preprocessing diagnostics

- Principal component analysis and multidimensional scaling embeddings

- Hierarchical clustering

- Differential expression results (including volcano plots and peptide trends)

- Feature-level trends across samples, conditions, and contrasts

Because clustering, embeddings, and statistical modeling are precomputed within ProteoFlux, visualization reflects exactly the analyzed data and introduces no additional analytical steps. Results can also be accessed directly in custom Python workflows without ProteoViewer.

Research impact statement

ProteoFlux is used for routine downstream analysis within a proteomics core facility at the University of Basel. The framework supports high-throughput LC–MS experiments spanning protein-centric, peptide-centric, and phosphoproteomics workflows.

Analyses are configured via reusable templates enabling semi-automated generation of experiment-specific configurations while maintaining consistent preprocessing and statistical assumptions.

ProteoFlux-generated outputs are distributed to internal and external collaborators in standardized formats, enabling reproducible re-analysis and downstream visualization without re-running the computational pipeline.

AI Usage Disclosure

Portions of this software and accompanying documentation were developed with assistance from generative AI tools (OpenAI ChatGPT, GPT-5 family models) for tasks including code refactoring suggestions, documentation drafting, CI scaffolding, and language editing of the manuscript.

All AI-assisted outputs were reviewed, validated, and edited by the human authors. The conceptual design, statistical methodology, software architecture, and final technical decisions

were made solely by the authors. The authors assume full responsibility for the correctness, originality, licensing, and integrity of all submitted materials.

Acknowledgements

We thank members of the Proteomics Core Facility for feedback and routine testing of the software.