The Journal of Open Source Software

Peptacular: A Python Library for ProForma 2.0-Compliant Peptide Analysis

Patrick Tyler Garrett [1] **and John R. Yates III** [1¶]

1 The Scripps Research Institute, United States ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

Mass spectrometry-based proteomics relies on computational methods to identify and characterize amino acid sequences (Angel et al., 2012). These sequences, which range from short peptides to complete proteins, exhibiting considerable chemical complexity. This complexity includes post-translational modifications, variable charge states, charge adducts, neutral losses, and isotopic patterns (Smith & Kelleher, 2013). ProForma notation provides a standardized, human-readable representation of this complexity, facilitating consistent communication of peptide and protein identifications across the proteomics community (LeDuc et al., 2018).

Peptacular is a pure Python library designed for working with **ProForma** notation in high-throughput proteomics applications. The library provides functionality for sequence manipulation, molecular mass and composition calculation, physicochemical property prediction, fragment ion generation, in silico enzymatic digestion, isotopic distribution computation, and format conversion between ProForma and vendor-specific notations.

Statement of Need

The proteomics field has historically lacked standardization for peptide and protein sequences. Individual software tools have implemented proprietary notations, creating barriers to data integration and reanalysis. While the Proteomics Standards Initiative developed ProForma notation to address this issue, adoption remains limited due to its relative novelty and the scarcity of computational tools supporting the standard.

Modern proteomics experiments routinely identify hundreds of thousands of peptide-spectrum matches. Results are typically exported as tabular files (CSV, TSV, Parquet) compatible with Python data analysis libraries such as **pandas** (Team, 2025) and **polars** (Vink et al., 2025). However, existing proteomics libraries lack efficient mechanisms for performing sequence-based calculations at this scale or integrating seamlessly with dataframe-centric workflows.

Peptacular addresses these limitations by providing two complementary APIs: an object-oriented interface, and a functional interface. The functional API accepts serialized ProForma sequences, applies transformations using parallel processing, and returns serialized results. This design enables direct integration with dataframes without sacrificing computational performance. The object-oriented interface allows for direct manipulation of the annotation objetcs without the need for serialiazation or parsing, improving performance for more complex operations.

Features and Implementation

Peptacular's employs lazy evaluation to minimize memory overhead: modifications remain in serialized form until explicitly required for calculations. Since Python strings are immutable

DRAFT

and proteomics datasets typically contain repeated modifications, Peptacular utilizes caching wherever possible, yielding performance and memory improvements.

The object-oriented API uses a factory pattern to construct Annotation objects, enabling sequence manipulation including modification placement, isotopic labeling, and custom fragment ion definitions. The functional API provides stateless functions that operate directly on serialized sequences, facilitating parallel execution across CPU cores.

Modification reference data, including masses, compositions, and identifiers from Unimod (Creasy & Cottrell, 2004), and other databases, are embedded within the package as Python modules rather than external files. This design eliminates file I/O overhead for parsing the supported ontologies, improving performance. Additionally, Peptacular uses conditional initialization: modification-related data structures are only instantiated when the corresponding modification type is present in the input sequences, reducing startup overhead and memory consumption.

The library supports bidirectional conversion between ProForma notation and common vendor-specific formats, facilitating integration with existing proteomics workflows while promoting migration toward standardized representations.

Related Software

Several existing tools address aspects of peptide manipulation and property calculation, including pyteomics (Goloborodko et al., 2013) and biopython (Cock et al., 2009). However, these tools were developed prior to the establishment of ProForma 2.0 notation and consequently lack native support for many modern ProForma-specific features. Furthermore, these packages include extensive dependencies and functionality that extends well beyond the scope of peptide notation standardization. Peptacular addresses this gap by providing a focused implementation designed to serve as computational infrastructure connecting different software tools and packages within the proteomics ecosystem.

Proforma Compliance

5.3.1 Base-ProForma Compliance

- ☒ **Amino acids (+UO)** - `AAHCFKUOT` (§6.1)
- ☒ **Unimod names** - `PEM[Oxidation]AT` (§6.2.1)
- ☒ **PSI-MOD names** - `PEM[monohydroxylated residue]AT` (§6.2.1)
- ☒ **Unimod numbers** - `PEM[UNIMOD:35]AT` (§6.2.2)
- ☒ **PSI-MOD numbers** - `PEM[MOD:00425]AT` (§6.2.2)
- ☒ **Delta masses** - `PEM[+15.995]AT` (§6.2.3)
- ☒ **N-terminal modifications** - `[Carbamyl]-QPEPTIDE` (§6.3)
- ☒ **C-terminal modifications** - `PEPTIDEG-[Methyl]` (§6.3)
- ☒ **Labile modifications** - `{Glycan:Hex}EM[U:Oxidation]EV` (§6.4)
- ☒ **Multiple modifications** - `MPGNW[Oxidation][Carboxymethyl]PESQE` (§6.5)
- ☒ **Information tag** - `ELV[INFO:AnyString]IS` (§6.6)

5.3.2 Level 2-ProForma Compliance

- ☒ **Ambiguous amino acids** - `BZJX` (§7.1)
- ☒ **Prefixed delta masses** - `PEM[U:+15.995]AT` (§7.2)
- ☒ **Mass gap** - `PEX[+147.035]AT` (§7.3)
- ☒ **Formulas** - `PEM[Formula:O]AT`, `PEM[Formula:[1701]]AT` (§7.4)
- ☒ **Mass with interpretation** - `PEM[+15.995|Oxidation]AT` (§7.5)
- ☒ **Unknown mod position** - `[Oxidation]?PEMAT` (§7.6.1)

DRAFT

- ☒ **Set of positions** - `PEP[Oxidation#1]M[#1]AT` (§7.6.2)
- ☒ **Range of positions** - `PRT(ESFRMS)[+19.0523]ISK` (§7.6.3)
- ☒ **Position scores** - `PEP[Oxidation#1(0.95)]M[#1(0.05)]AT` (§7.6.4)
- ☒ **Range position scores** - `(PEP)[Oxidation#1(0.95)]M[#1(0.05)]AT` (§7.6.5)
- ☒ **Amino acid ambiguity** - `(?VCH)AT` (§7.7)
- ☒ **Modification prefixes** - `PEPM[U:Oxidation]AS[M:O-phospho-L-serine]` (§7.8)

5.3.3 Level 2-ProForma + Top-Down

- ☒ **RESID modifications** - `EM[R:L-methionine sulfone]EM[RESID:AA0581]` (§8.1)
- ☒ **Names** - `(>Heavy chain)EVQLVESG` (§8.2)

5.3.4 Level 2-ProForma + Cross-Linking

- ☒ **XL-MOD modifications** - `EVTK[X:Aryl azide]LEK[XLMOD:00114]SEFD` (§9.1)
- ☐ **Cross-linkers (intrachain)** - `EVTK[X:Aryl azide#XL1]LEK[#XL1]SEFD` (§9.2.1)
- ☐ **Cross-linkers (interchain)** - `EVTK[X:Aryl azide#XL1]L//EK[#XL1]SEFD` (§9.2.2)
- ☐ **Branches** - `ED[MOD:00093#BRANCH]//D[#BRANCH]ATR` (§9.3)

5.3.5 Level 2-ProForma + Glycans

- ☒ **GNO modifications** - `NEEYN[GNO:G59626AS]K` (§10.1)
- ☒ **Glycan compositions** - `NEEYN[Glycan:Hex5HexNAc4NeuAc1]K` (§10.2)

5.3.6 Level 2-ProForma + Advanced Complexity

- ☒ **Charged formulas** - `SEQUEN[Formula:Zn1:z+2]CE` (§11.1)
- ☒ **Controlling placement** - `PTI(MERMERME)[+32|Position:E]PTIDE` (§11.2)
- ☒ **Global isotope** - `<13C>CARBON` (§11.3.1)
- ☒ **Fixed modifications** - `<[Oxidation]@M>ATPEMILTCMGCLK` (§11.3.2)
- ☒ **Chimeric spectra** - `NEEYN+SEQUEN` (§11.4)
- ☒ **Charges** - `SEQUEN/2`, `SEQUEN/[Na:z+1,H:z+1]` (§11.5)
- ☒ **Ion notation** - `SEQUEN-[b-type-ion]` (§11.6)

AI usage disclosure

Various AI models were used to aid in the creation of this softwares codebase, tests, and documentation, mainly Claude Sonnet 4.5, Gemini 3 Pro, and Github's copilot autocomlete extension. The AI models were used via the copilot extension in VS code. No AI models were used in the create of this manuscript.

Angel, T. E., Aryal, U. K., Hengel, S. M., Baker, E. S., Kelly, R. T., Robinson, E. W., & Smith, R. D. (2012). Mass spectrometry-based proteomics: existing capabilities and future directions. *Chemical Society Reviews*, *41*(10), 3912. https://doi.org/10.1039/c2cs15331a

Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., & De Hoon, M. J. L. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, *25*(11), 1422–1423. https://doi.org/10.1093/bioinformatics/btp163

Creasy, D. M., & Cottrell, J. S. (2004). Unimod: Protein modifications for mass spectrometry. *PROTEOMICS*, *4*(6), 1534–1536. https://doi.org/10.1002/pmic.200300744

Goloborodko, A. A., Levitsky, L. I., Ivanov, M. V., & Gorshkov, M. V. (2013). Pyteomics—A Python framework for exploratory data analysis and rapid software prototyping in proteomics. *Journal of the American Society for Mass Spectrometry*, *24*(2), 301–304. https://doi.org/10.1007/s13361-012-0516-6

DRAFT

LeDuc, R. D., Schwämmle, V., Shortreed, M. R., Cesnik, A. J., Solntsev, S. K., Shaw, J. B., Martin, M. J., Vizcaino, J. A., Alpi, E., Danis, P., Kelleher, N. L., Smith, L. M., Ge, Y., Agar, J. N., Chamot-Rooke, J., Loo, J. A., Pasa-Tolic, L., & Tsybin, Y. O. (2018). ProForma: a standard proteoform notation. *Journal of Proteome Research*, *17*(3), 1321–1325. https://doi.org/10.1021/acs.jproteome.7b00851

Smith, L. M., & Kelleher, N. L. (2013). Proteoform: a single term describing protein complexity. *Nature Methods*, *10*(3), 186–187. https://doi.org/10.1038/nmeth.2369

Team, P. D. (2025). pandas-dev/pandas: Pandas. *Zenodo (CERN European Organization for Nuclear Research)*. https://doi.org/10.5281/zenodo.17992932

Vink, R., De Gooijer, S., Beedie, A., Burghoorn, G., Nameexhaustion, Peters, O., Gorelli, M. E., Reswqa, Van Zundert, J., Marshall, Hulselmans, G., Grinstead, C., Denecker, K., Manley, L., chielP, Turner-Trauring, I., Valtar, K., Mitchell, L., Sprenkels, A., … Magarick, J. (2025). pola-rs/polars: Python Polars 1.36.1. *Zenodo (CERN European Organization for Nuclear Research)*. https://doi.org/10.5281/zenodo.17873635

DRAFT