

Peptacular: A Python package for amino acid sequence analysis with ProForma 2.1

Patrick T. Garrett¹ and John R. Yates III¹

¹ The Scripps Research Institute, United States

DOI: [10.xxxxxx/draft](https://doi.org/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Published: unpublished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 identifies and characterizes proteins by measuring molecules and their fragments. Interpreting these measurements requires software that accounts for chemical modifications, charge states, and isotope composition ([Angel et al., 2012](#)). **Peptacular** is a Python library for representing and analyzing peptide and protein sequences using ProForma notation. It supports sequence editing, mass and mass-to-charge ratio (m/z) determination, elemental composition, isotope envelopes, enzymatic digestion, theoretical fragmentation, and physicochemical properties. Its chemical analysis APIs operate on individual peptide chains, while separate interfaces represent chimeric assignments and structured notation. Table 1 distinguishes notation support from limits on these operations.

Statement of Need

ProForma standardizes how peptide and protein sequences describe modifications and ambiguity ([LeDuc et al., 2022](#)). Version 2.1 extends this notation with additional chemical and structural annotations ([HUPO Proteomics Standards Initiative, 2026](#)). Proteomics developers still need to carry those annotations through sequence editing, digestion, and chemical analysis without silently discarding information. Peptacular targets researchers building analysis scripts, theoretical peptide libraries, and proteomics software that require consistent behavior across these operations.

The library provides scalar and batch interfaces for serialized sequences and parsed annotations. Batch results can be assigned to tabular data, including pandas DataFrames ([The pandas development team, 2025](#)). Streaming input and optional error collection support workflows in which some annotations are unresolved or unsuitable for a requested operation.

State of the Field

Existing packages address overlapping needs. **Pyteomics** ([Goloborodko et al., 2013](#); [Levitsky et al., 2019](#)) supports ProForma parsing, mass and composition calculations, and fragment generation alongside its historical modX format ([Pyteomics developers, n.d.](#)). **Biopython** ([Cock et al., 2009](#)) provides general sequence analysis, including protein property calculations through its ProtParam module ([Biopython contributors, n.d.](#)). The **mzcore** and related RustyMS libraries provide ProForma support, chemical representations, and theoretical fragmentation, with Python bindings for selected components ([Schulte et al., n.d., 2025](#)). **pyOpenMS** ([Röst et al., 2014](#)) exposes a broader mass spectrometry toolkit. Current OpenMS documentation also describes ProForma parsing and conversion ([OpenMS developers, n.d.](#)).

Peptacular's contribution is a common, editable ProForma annotation model shared by sequence transformations and chemical analysis in Python. This design allows researchers to inspect and

modify annotations directly while using consistent charge, modification, and ambiguity handling across operations. A dedicated package keeps this interface focused on sequence analysis. Optional adapters connect it to Pyteomics, psm_utils (Gabriels et al., 2023), and AlphaBase for their supported representations, with checks for conversion losses. These integrations complement the specialized capabilities of the surrounding ecosystem.

Software Design

Peptacular offers functional and object-oriented APIs. ProFormaAnnotation objects support inspection, serialization, and chained edits. Many editing methods modify the object by default and accept inplace=False to return a copy. Functional operations accept strings or annotations and support sequence batches.

Execution can be sequential, threaded, or process-based. Automatic functional calls use sequential processing below 1,000 inputs, avoiding worker startup costs for small batches. Larger batches use process-based execution on conventional GIL-enabled Python builds and threads on free-threaded builds. Explicit settings override this selection. Functional calls create temporary pools, while iter_batch() reuses an executor across chunks within a call and returns ordered results with optional diagnostics.

Lazy modification parsing and bounded caches reduce repeated work. Direct residue mass lookups and a scalar path for ordinary precursor mass and m/z avoid unnecessary fragment objects and annotation copies. A composition-based path handles isotope labels and elemental adjustments. Both paths account for intrinsic modification charge and electron mass. BRAIN recurrences generate aggregated isotope envelopes with a probability-weighted center mass per nominal isotope peak (Claesen et al., 2012; Dittwald & Valkenburg, 2014). Fine structure is not resolved. A separate averagine (Senko et al., 1995) API estimates envelopes from mass alone.

Shared reference data are supplied by Tacular (P. Garrett, 2026), including Unimod (Creasy & Cottrell, 2004), PSI-MOD (HUPO Proteomics Standards Initiative, n.d.-a), RESID (Protein Information Resource, n.d.), XLMOD (HUPO Proteomics Standards Initiative, n.d.-b), and GNOme (GlyGen, n.d.). Embedded data avoid runtime ontology downloads. Chemical analyses require a resolvable mass or elemental composition, depending on the operation. Unresolved annotations can still be represented, while diagnostics distinguish parsing, validation, and analysis failures.

Versioned JSON serialization preserves annotation structure and validates input against a closed set of supported types. Streaming FASTA input supports plain and gzip files. An optional local Model Context Protocol interface exposes the same sequence operations to agent clients. The core package requires Python 3.12 or later and Tacular, with additional dependencies installed only for optional integrations. Type annotations support static analysis. Continuous integration runs tests, linting, type checks, and package builds across Python 3.12-3.14 and Linux, macOS, and Windows configurations.

Research impact statement

Peptacular has been used outside its development group. Malsagova and colleagues used it to match theoretical b- and y-ions, including water and ammonia losses, when presenting peptide identifications from plasma proteomics in a study of exercise intensity (Malsagova et al., 2026, fig. 4). HUPO-PSI also lists Peptacular among Python implementations of ProForma (HUPO Proteomics Standards Initiative, 2026). Within the authors' software ecosystem, Spxtacular uses Peptacular's theoretical fragments in spectrum matching and scoring (P. T. Garrett & Yates, 2026). Peptacular has additionally been used by its developers to prepare figures for a proteomics textbook chapter (P. T. Garrett et al., 2025).

Example Usage

Object-oriented API

```
import peptacular as pt

# Parse a sequence into a ProFormaAnnotation
peptide: pt.ProFormaAnnotation = pt.parse("PEM[Oxidation]TIDE")

# Calculate mass and m/z
mass: float = peptide.mass() # 849.343
mz: float = peptide.mz(charge=2) # 425.679

# Chained edits modify the annotation
print(peptide.set_charge(2).set_peptide_name("Peptacular").serialize())
# (>Peptacular)PEM[Oxidation]TIDE/2
```

Functional API

```
import peptacular as pt

peptides = ['[Acetyl]-PEPTIDES', '<13C>ARE', 'SICK/2']

# Calculate mass and m/z for all peptides
masses: list[float] = pt.mass(peptides) # [928.4026, 388.2384, 451.2454]
mzs: list[float] = pt.mz(peptides, charge=2) # [465.2086, 195.1265, 225.6227]
```

Tabular workflow

```
import peptacular as pt
import pandas as pd

df = pd.DataFrame(
    {
        "seq": ["PEM[Oxidation]TIDE", "ACDEFGHIK", "AS[Phospho]TPEK"],
    }
)

df["mass"] = pt.mass(df["seq"].tolist())
```

Notation support and operation limits

Table 1: Representative ProForma 2.1 notation support and operation limits

S	Feature	Example	§ [Level]
Y	Amino acids (+UO)	AAHCFKUOT	6.1 [1]
Y	Unimod names	PEM[Oxidation]AT	6.2.1 [1]
Y	PSI-MOD names	PEM[monohydroxylated residue]AT	6.2.1 [1]
Y	Unimod numbers	PEM[UNIMOD:35]AT	6.2.2 [1]
Y	PSI-MOD numbers	PEM[MOD:00425]AT	6.2.2 [1]
Y	Delta masses	PEM[+15.995]AT	6.2.3 [1]
Y	N-terminal modifications	[Carbamyl]-QPEPTIDE	6.3 [1]
Y	C-terminal modifications	PEPTIDEG-[Methyl]	6.3 [1]
Y	Labile modifications	{Glycan:Hex}EM[U:Oxidation]EV	6.4 [1]

S	Feature	Example	§ [Level]
Y	Multiple modifications	MPGNW[Oxidation][Carboxymethyl]PESQE	6.5 [1]
Y	Information tags	ELV[INFO:AnyString]IS	6.6 [1]
Y	Ambiguous amino acids	BZJX	7.1 [2]
Y	Prefixed delta masses	PEM[U:+15.995]AT	7.2 [2]
Y	Mass gap	PEX[+147.035]AT	7.3 [2]
Y	Formulas	PEM[Formula:0]AT, PEM[Formula:[1701]]AT	7.4 [2]
Y	Mass with interpretation	PEM[+15.995 Oxidation]AT	7.5 [2]
Y	Unknown mod position	[Oxidation]?PEMAT	7.6.1 [2]
Y	Set of positions	PEP[Oxidation#1]M[#1]AT	7.6.2 [2]
Y	Range of positions	PRT(ESFRMS)[+19.0523]ISK	7.6.3 [2]
Y	Position scores	PEP[Oxidation#1(0.95)]M[#1(0.05)]AT	7.6.4 [2]
Y	Range position scores	(PEP)[Oxidation#1(0.95)]M[#1(0.05)]AT	7.6.5 [2]
Y	Amino acid ambiguity	(?VCH)AT	7.7 [2]
Y	Modification prefixes	PEPM[U:Oxidation]AS[M:0-phospho-L-serine]	7.8 [2]
Y	Labile locations	{Phospho#g1}EMEVS[#g1]	7.9 [2]
Y	RESID modifications	EM[R:L-methionine sulfone]EM[RESID:AA0251]	8.1 [T]
Y	Names	(>Heavy chain)EVQLVESG	8.2 [T]
Y	XL-MOD modifications	EVTK[X:DSS]LEK[XLMOD:02001]SEFD	9.1 [X]
N	Cross-linkers (intrachain)	EVTK[X:DSS#XL1]LEK[#XL1]SEFD	9.2.1 [X]
N	Cross-linkers (interchain)	EVTK[X:DSS#XL1]L//EK[#XL1]SEFD	9.2.2 [X]
N	Branches	ED[MOD:00093#BRANCH]//D[#BRANCH]ATR	9.3 [X]
Y	GNO modifications	NEEYN[GNO:G59626AS]K	10.1 [G]
Y	Glycan compositions	NEEYN[Glycan:Hex5HexNAc4NeuAc1]K	10.2 [G]
Y	Mixed glycan components	N[Glycan:Hex{H2O}{+204.068}]K	10.2 [G]
Y	Charged formulas	SEQUEN[Formula:Zn1:z+2]CE	11.1 [3]
P	Controlling placement	PTI(MERMERME)[+32 Position:E]PTIDE	11.2 [3]
Y	Global isotope	<13C>PEPTIDE	11.3.1 [3]
Y	Fixed modifications	<[Oxidation]@M>ATPEMILTCMGLK	11.3.2 [3]
Y	Chimeric spectra	NEEYN+SEQUEN	11.4 [3]
Y	Charges	SEQUEN/2, SEQUEN/[Na:z+1,H:z+1]	11.5 [3]
Y	Ion notation	SEQUEN-[b-type-ion]	11.6 [3]

Table 1 summarizes notation handling against the finalized ProForma 2.1 specification (HUPO Proteomics Standards Initiative, 2026). In column S, Y indicates supported notation, P indicates a preserved annotation whose placement constraints are not applied, and N indicates linked-chain notation that can be represented but is not supported by the analysis operations. Levels 1, 2, and 3 follow the specification, with top-down (T), cross-linking (X), and glycan (G) extensions. Notation support does not imply that every operation is available for every represented sequence.

Mass-ambiguous residues B and Z cannot produce a unique mass. Delta-mass tags and bare-mass glycan components lack elemental compositions. Unknown localization and ambiguous intervals restrict fragmentation. Labile modifications contribute to precursor mass but are omitted from fragment ions. Chimeric assignments are handled component-wise through `parse_chimeric()`. The structured model and JSON format can represent linked notation, but the analysis APIs do not operate on cross-linked or branched structures. The documentation provides operation-specific limits and examples.

AI usage disclosure

Opus 5 and Fable 5.1 through Claude Code, and Sol and Astra through OpenAI Codex, assisted with code generation, refactoring, tests, debugging, documentation, and manuscript revision. Verification included automated tests, static analysis, execution of manuscript examples, and checks against primary references. The human authors reviewed, edited, and validated all AI-assisted work and made the core design decisions.

Availability

Peptacular is distributed through PyPI (<https://pypi.org/project/peptacular/>) and available as open-source software on GitHub (<https://github.com/tacular-omics/peptacular>). Documentation is accessible at <https://peptacular.readthedocs.io>. The software is released under the MIT license.

Acknowledgements

This work was supported by the National Institutes of Health under grants R01 AG077046 (Analysis of protein interactions in neurodegenerative disease), R01 MH132570 (Brain-wide mapping of neuronal inhibition by novel inverse activity markers), R01 MH100175 (Proteogenetics of Autism Spectrum Disorders), R01 HL165168 (The CFTR Interactome), and U01 AG088679 (Understanding Gene-Environment Interactions in Brain Aging and Alzheimer's Disease (AD) and AD-Related Dementias (ADRD)).

The funders provided financial support only.

Competing interests

The authors report no competing interests.

References

- 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), 3912–3928. <https://doi.org/10.1039/c2cs15331a>
- Biopython contributors. (n.d.). *Bio.SeqUtils.ProtParam* module. Retrieved September 22, 2026, from <https://biopython.org/docs/latest/api/Bio.SeqUtils.ProtParam.html>
- Claesen, J., Dittwald, P., Burzykowski, T., & Valkenborg, D. (2012). An Efficient Method to Calculate the Aggregated Isotopic Distribution and Exact Center-Masses. *Journal of the American Society for Mass Spectrometry*, 23(4), 753–763. <https://doi.org/10.1007/s13361-011-0326-2>
- 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>
- Dittwald, P., & Valkenborg, D. (2014). BRAIN 2.0: Time and Memory Complexity Improvements in the Algorithm for Calculating the Isotope Distribution. *Journal of the*

- 145 *American Society for Mass Spectrometry*, 25(4), 588–594. [https://doi.org/10.1007/s13361-](https://doi.org/10.1007/s13361-013-0796-5)
146 [013-0796-5](https://doi.org/10.1007/s13361-013-0796-5)
- 147 Gabriels, R., Declercq, A., Bouwmeester, R., Degroeve, S., & Martens, L. (2023). psm_utils: A
148 High-Level Python API for Parsing and Handling Peptide-Spectrum Matches and Proteomics
149 Search Results. *Journal of Proteome Research*, 22(2), 557–560. [https://doi.org/10.1021/](https://doi.org/10.1021/acs.jproteome.2c00609)
150 [acs.jproteome.2c00609](https://doi.org/10.1021/acs.jproteome.2c00609)
- 151 Garrett, P. (2026). *tacular-omics/tacular: v1.0.1 - Zenodo + Docs* (Version v1.0.1). Zenodo.
152 <https://doi.org/10.5281/zenodo.18475557>
- 153 Garrett, P. T., Turner, N. P., Nakorchevsky, A., Pankow, S., & Yates, J. R. (2025). Mass
154 spectrometry-based proteomics. In *Reference module in life sciences*. Elsevier. <https://doi.org/10.1016/b978-0-323-99507-8.00013-5>
155
- 156 Garrett, P. T., & Yates, J. R., III. (2026). *Spxtacular*. [https://github.com/tacular-omics/](https://github.com/tacular-omics/spxtacular)
157 [spxtacular](https://github.com/tacular-omics/spxtacular)
- 158 GlyGen. (n.d.). *GNOME: Glycan Naming and Subsumption Ontology*. Retrieved September
159 13, 2026, from <https://github.com/glygen-glycan-data/GNOME>
- 160 Goloborodko, A. A., Levitsky, L. I., Ivanov, M. V., & Gorshkov, M. V. (2013). Pyteomics:
161 A Python framework for exploratory data analysis and rapid software prototyping in
162 proteomics. *Journal of the American Society for Mass Spectrometry*, 24(2), 301–304.
163 <https://doi.org/10.1007/s13361-012-0516-6>
- 164 HUPO Proteomics Standards Initiative. (n.d.-a). *PSI-MOD ontology for modified and*
165 *unmodified amino acid residues*. Retrieved September 13, 2026, from [https://github.com/](https://github.com/HUPO-PSI/psi-mod-CV)
166 [HUPO-PSI/psi-mod-CV](https://github.com/HUPO-PSI/psi-mod-CV)
- 167 HUPO Proteomics Standards Initiative. (n.d.-b). *XLMOD ontology for chemical cross-linkers*.
168 Retrieved September 13, 2026, from <https://github.com/HUPO-PSI/xlmod-CV>
- 169 HUPO Proteomics Standards Initiative. (2026). *ProForma 2.1: Proteoform and Peptidoform*
170 *Notation*. <https://www.psidev.info/proforma>
- 171 LeDuc, R. D., Deutsch, E. W., Binz, P.-A., Fellers, R. T., Cesnik, A. J., Klein, J. A., Van
172 Den Bossche, T., Gabriels, R., Yalavarthi, A., Perez-Riverol, Y., Carver, J., Bittremieux,
173 W., Kawano, S., Pullman, B., Bandeira, N., Kelleher, N. L., Thomas, P. M., & Vizcaíno,
174 J. A. (2022). Proteomics Standards Initiative's ProForma 2.0: Unifying the Encoding
175 of Proteoforms and Peptidoforms. *Journal of Proteome Research*, 21(4), 1189–1195.
176 <https://doi.org/10.1021/acs.jproteome.1c00771>
- 177 Levitsky, L. I., Klein, J. A., Ivanov, M. V., & Gorshkov, M. V. (2019). Pyteomics 4.0: Five
178 Years of Development of a Python Proteomics Framework. *Journal of Proteome Research*,
179 18(2), 709–714. <https://doi.org/10.1021/acs.jproteome.8b00717>
- 180 Malsagova, K. A., Butkova, T. V., Nikolsky, K. S., Petrovskiy, D. V., Kopylov, A. T., Nakhod,
181 V. I., Kulikova, L. I., Rudnev, V. R., Yurku, K. A., Balakin, E. I., Bukreeva, A. S., Izotov,
182 A. A., Pustovoyt, V. I., & Kaysheva, A. L. (2026). Changes in the proteomic profile
183 of athletes' plasma associated with exercise intensity. *Scientific Reports*, 16(1), 14205.
184 <https://doi.org/10.1038/s41598-026-44729-5>
- 185 OpenMS developers. (n.d.). *OpenMS ProForma class reference*. Retrieved September 13,
186 2026, from https://openms.de/current_doxygen/html/classOpenMS_1_1ProForma.html
- 187 Protein Information Resource. (n.d.). *RESID Database*. Retrieved September 13, 2026, from
188 <https://proteininformationresource.org/resid/>
- 189 Pyteomics developers. (n.d.). *Pyteomics: Peptide sequence formats*. Retrieved September 13,
190 2026, from <https://pyteomics.readthedocs.io/en/latest/sequences.html>
- 191 Röst, H. L., Schmitt, U., Aebersold, R., & Malmström, L. (2014). pyOpenMS: A Python-based

- 192 interface to the OpenMS mass-spectrometry algorithm library. *PROTEOMICS*, 14(1),
193 74–77. <https://doi.org/10.1002/pmic.201300246>
- 194 Schulte, D., Gabriels, R., & Heerdink, A. (n.d.). *mzcore*. Retrieved September 13, 2026, from
195 <https://github.com/rusteomics/mzcore>
- 196 Schulte, D., Leuvenink, R. W., Jager, S., Heck, A. J. R., & Snijder, J. (2025). A Universal
197 Spectrum Annotator for Complex Peptidoforms in Mass Spectrometry-Based Proteomics.
198 *Analytical Chemistry*, 97(42), 23120–23130. [https://doi.org/10.1021/acs.analchem.](https://doi.org/10.1021/acs.analchem.5c02832)
199 [5c02832](https://doi.org/10.1021/acs.analchem.5c02832)
- 200 Senko, M. W., Beu, S. C., & McLafferty, F. W. (1995). Determination of Monoisotopic Masses
201 and Ion Populations for Large Biomolecules from Resolved Isotopic Distributions. *Journal*
202 *of the American Society for Mass Spectrometry*, 6(4), 229–233. [https://doi.org/10.1016/](https://doi.org/10.1016/1044-0305(95)00017-8)
203 [1044-0305\(95\)00017-8](https://doi.org/10.1016/1044-0305(95)00017-8)
- 204 The pandas development team. (2025). *pandas-dev/pandas: Pandas* (Version 3.0.0rc1).
205 Zenodo. <https://doi.org/10.5281/zenodo.17992932>

DRAFT