Journal of proteome research

pubs.acs.org/jpr

Technical Note

Comprehensive Protein Inference Analysis with PyProteinInference Elucidates Biological Understanding of Tandem Mass Spectrometry Data

Published as part of Journal of Proteome Research special issue "Software Tools and Resources 2025".

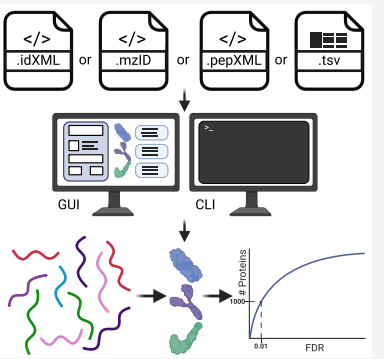
Trent B. Hinkle and Corey E. Bakalarski*

Cite This: *J. Proteome Res.* 2025, 24, 2135−2140

Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: Selection and application of protein inference algorithms can have a significant impact on the data output from tandem mass spectrometry (MS/MS) experiments. However, this critical step is often taken for granted, with many studies simply utilizing the inference method embedded within the end-to-end software pipeline employed for analysis without consideration of the particular algorithm's suitability for the experiment at hand or its effects on the resulting data. Although many individual inference algorithms have been demonstrated, few unified tools are available that allow the researcher to quickly apply a variety of different inference algorithms to meet the needs of their analysis, are agnostic of other tools in the analysis pipeline, and are easy to use for the bench biologist. PyProteinInference provides a comprehensive suite of tools that enable researchers to apply different inference algorithms and compute protein-level set-based false discovery rates (FDR) from MS/MS data through a unified interface. Here, we describe the software and its application to a traditional protein inference benchmarking data set and to a K562 whole-cell lysate to demonstrate its utility in facilitating conclusions about underlying biological mechanisms in proteomic data.



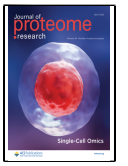
KEYWORDS: *Protein Inference, Proteomics, Mass Spectrometry, Python*

INTRODUCTION

Advances in tandem mass spectrometry (MS/MS) for proteome-level research have driven many novel insights in basic biology and disease.[1] In a typical bottom-up mass-spectrometry-based proteomics experiment, the protein component of interest is first proteolytically digested into constituent peptides for easier analysis by mass spectrometry. This step produces peptides that are more amenable than intact proteins to high-throughput analysis; however, it comes at a cost, severing the relationship between the peptide and its original protein. As this linkage is no longer maintained, it must instead be inferred computationally after the collected mass spectra are matched to their generative peptides using popular algorithms such as MSFragger,[2] DIA-NN[3] or Comet.[4] This process of protein inference attempts to reassemble peptides into a list of proteins thought to be present in a sample. While peptide-spectral matching approaches are well established and rely on data provided by ion masses generated during peptide fragmentation, the digestion of proteins into peptides during sample processing leaves behind scant evidence that can be used to determine the original relationships necessary for data interpretation. Thus, despite being a well-described problem for over 20 years,[5,6] protein inference is still a challenging area of active research.

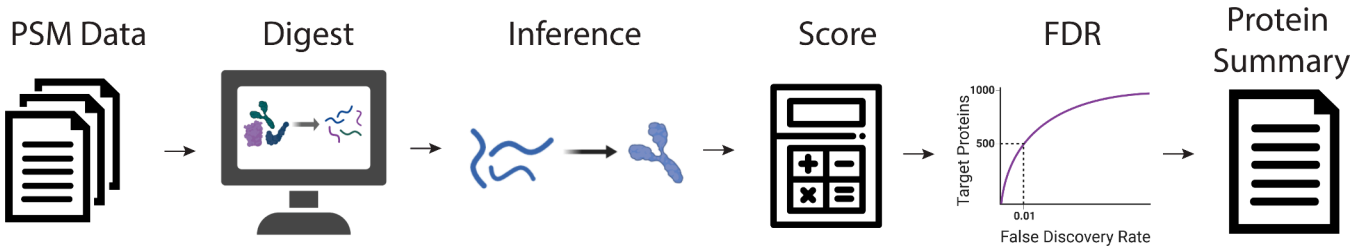
Multiple algorithms to perform protein inference have been developed in response to this challenge, including Protein Prophet,[7] Fido,[8] PIA,[9] and Percolator Protein Inference.[10] These tools typically provide mutually exclusive, widely varying approaches to inference and scoring which are also data set-dependent. Different combinations of factors, including sequence overlap or others which may not be predictable *a priori*, greatly alter the sensitivity of the results by influencing how peptides are mapped and which proteins are ultimately reported. In addition to multiple inference assignment methods, differences in protein scoring approaches can also lead to differences in the results.[10] While multiple comparisons between methods have been reported,[11−13] there is no clear consensus on a singular approach with the best performance, as ultimately the choice of algorithm is not simply to maximize the number of proteins identified or to produce a more conservative result but dependent upon the aims of the

Received: August 26, 2024
Revised: January 22, 2025
Accepted: February 13, 2025
Published: February 28, 2025



https://doi.org/10.1021/acs.jproteome.4c00734
J. Proteome Res. 2025, 24, 2135−2140

Scheme 1. PyProteinInference Workflow



hypothesis being studied. For example, an approach selected for a global profiling experiment may not be appropriate for a focused protein−protein interaction study.[14] The ability to quickly apply and compare multiple protein inference approaches can be critical to understand mechanistic relationships, especially in the context of experiments where sequence similarity of potential biological drivers could obscure results and a thorough consideration of the peptide evidence present for each protein is necessary.

For those not well-versed in the nuanced difference between approaches, this can lead to the use of a "one size fits all" approach of applying default methods included alongside search algorithms, without consideration for their effects on the resulting data and its suitability to answer the biological questions raised. Beyond scientific considerations, the facile usage and comparison of different methods can be difficult due to varying requirements for installation, input, configuration, and output of multiple algorithms, making comparisons between different approaches time-consuming and beyond the reach of many biologists.

To address this need, we implemented pyProteinInference, a standalone, lightweight software tool to easily apply multiple protein inference algorithms and associated protein level set-based false discovery rate determinations dissociated from any particular proteomic pipeline or analysis suite yet widely compatible with various proteomic data formats. The software has been implemented as a cross-platform package written in Python that provides a streamlined, easy-to-use graphical user interface to apply multiple protein inference algorithms and can be run on any modern computing system, from individual workstations to high-performance cloud computing environments. The application does not require the installation of any other software or platforms, is permissively licensed for commercial and noncommercial use, accepts common input formats, and presents a consistent output, with the aim of assisting the biologist in matching the appropriate inference approach with their data set. The application can also be run as a standalone command line tool for easy integration into larger modular analysis pipelines, using common tools like Cromwell[15] or NextFlow.[16] Here, we introduce the open-source software and present it in the context of both a benchmarking data set and whole cell lysate analysis, which is representative of many proteome-wide analysis experiments, utilizing both to highlight the application of the algorithm and to illustrate the differences in resulting protein sets generated from the different algorithms and summarization methods, and considerations for their use.

EXPERIMENTAL PROCEDURES

Implementation of pyProteinInference

PyProteinInference utilizes a standardized workflow (Scheme 1) to quickly apply multiple protein inference methods, contained in an easily installed Python package or binary executable from GitHub at https://github.com/thinkle12/pyproteininference, where the application source code, user, and developer documentation are also available. The code is free for commercial and noncommercial usage under the permissive Apache 2.0 license and has no external dependencies other than the internal use of several Python libraries within the code. The library supports most common input formats including Percolator-generated tab-separated values output[17] and support for the multiple XML-based formats,[18] including idXML, mzIdentML (mzID), and pepXML, as well as accepting generic tabular input. The graphical user interface provides a form-based configuration of multiple options and allows selection of the inference approach, scoring approach, input results and optional fasta database files, output file, and execution of the program. Options are documented within the program and within the user documentation on GitHub. Configuration can also be provided via a YAML file, and the algorithm can be called from the command line, ideal for use as part of larger, automated workflows.

PyProteinInference can perform four different protein inference methods including parsimony,[19] inclusion, exclusion,[13] and peptide-centric.[20] For each method, input is first ingested through an XML-based file (including pepXML, mzIdentML, and idXML) or one or more tab-delimited files; these tab-delimited files can be in Percolator result format (including support for combined or separate target-decoy searches). PyProteinInference also has internal support for other custom tab-delimited formats. An optional fasta file containing the sequence database used for the search can be supplied and can be used to provide alternative protein mappings if they were not supplied by the search engine or PSM filtering algorithm (or in addition to the mappings provided in the results). If multiple PSMs are provided for the same spectrum, the top-ranked hit is chosen based on the user-selected score (defaulting to posterior error probability). PSMs can then be filtered by any of the posterior error probability, q-value, minimum peptide length, or any user-supplied score. Inference algorithms were reimplemented *de novo* from their original descriptions[19,20] within Python. PyProteinInference recalculates protein-level false discovery rates based upon a variety of protein set scoring transformations of individual peptide scores (e.g., posterior error probability, q-value, or any custom score): multiplicative log, best peptide per protein, top two combined (multiplicative log of the top two PSM scores per protein), geometric mean, iterative down weighted log, down weighted multiplicative log, and simple additive. Ranked

https://doi.org/10.1021/acs.jproteome.4c00734
J. Proteome Res. 2025, 24, 2135−2140

scores are used to derive protein-level set-based false discovery rates through an iterative approach. Protein sets are reported according to the method used: inclusion sets include all possible mappings, exclusion sets consist of the single unique mapping, parsimony reports a group lead, and peptide-centric reports back groups as defined by its approach. More detailed information including all parameter options and more in-depth inference algorithm descriptions can be found in the package documentation.

PyProteinInference can be installed as a package for software developer use using standard Python methods (pip via pyPI) or via a standalone binary hosted on GitHub. It runs on any modern Windows, Linux or Mac system with 8 GB of memory; memory requirements scale with the size of the protein database (if supplied) and the number of PSMs and have been tested with over 200,000 filtered PSMs and over 50 instrument analyses. Source code and package documentation is available at: https://github.com/thinkle12/pyproteininference.

Validation of pyProteinInference Algorithms Using Protein Epitope Signature Tags (PrESTs)

To validate the performance of the algorithmic implementations within pyProteinInference, the benchmarking PrEST data set reported by The et al.[13] was downloaded from PRIDE (PXD008425; https://proteomecentral.proteomexchange.org/cgi/GetDataset?ID=PXD008425). Triplicate raw data files for each mixture (A, B, A+B) were converted to mzML and PSMs were assigned using Comet (v2023.1)[4] using a concatenated target-decoy (generated from reversed target sequences) database consisting of the union of the protein fragments from mixture A, mixture B, along with 1000 representative entrapment sequences as in the original publication. The Comet search used a peptide mass tolerance of 20 ppm, a fragment bin tolerance of 1.0005 Da, a fragment bin offset of 0.4 Da, and tryptic enzyme specificity with up to 2 missed cleavages. For modifications, a static cysteine carbamidomethyl (+57.02) modification and variable methionine oxidation (+15.99) modification were used. PSM data was then filtered at the peptide level only using Percolator (v3.1.0) via the OpenMS PercolatorAdaptor[18] with target-decoy competition and q-value scoring, test and training peptide-level FDRs of 0.01, and up to 10 iterations, which was then used as input to pyProteinInference (v1.1.1). Results were generated utilizing four inference options (inclusion, exclusion, parsimony, and peptide-centric) with posterior error probability as the PSM score, with best-peptide-per-protein as the protein score, and all PSMs from the Percolator results (Table S1). PrEST numbers were generated for the "A" PrEST set as well as the "A+B" PrEST set, as previously described. Entrapment FDR statistics at a 5% EFDR and comparative inference plots were generated using a python script (https://github.com/thinkle12/pyProteinInferenceReAnalysis/blob/main/prest/proteoform-standard/generate_prest_report.py) slightly modified from the original at https://github.com/statisticalbiotechnology/proteoform-standard to compare PrEST protein sets generated with pyProteinInference algorithms to protein sets previously reported using the best-peptide-per-group scoring method.

Validation of pyProteinInference Algorithms Using K562 Whole Cell Lysate

For algorithm validation of pyProteinInference, 0.5 μg of K562 whole cell lysate (Promega, Madison WI; Catalog #V6951 was loaded into a 25 cm × 75 μm ID, 1.6 μm C18 IonOpticks Aurora Series column (IonOpticks, AUR2-25075C18A) on a Thermo UltiMate 3000 high-performance liquid chromatography (HPLC) system (Thermo Fisher Scientific) at a flow rate of 400 nl min^{-1}. Peptides were separated with a 45 min gradient of 2% to 35% buffer B (98% ACN, 2% water, and 0.1% FA) at a flow rate of 300 nl min^{-1}. The gradient was then raised to 75% buffer B for 5 min and to 90% buffer B for 4 min at the same flow rate before final equilibration with 98% buffer A (98% water, 2% ACN and 0.1% FA) and 2% buffer B for 10 min at a flow rate of 400 nl min^{-1}. Peptide mass spectra were acquired using an Orbitrap Fusion Lumos (Thermo Fisher Scientific) with an MS^1 Orbitrap resolution of 240,000 and MS/MS fragmentation of the precursor ions by collision-induced dissociation (CID), followed by spectra acquisition at an MS^2 Orbitrap resolution of 15,000. PSMs were assigned with Comet (v2023.1)[4] as described above, however, with the use of the concatenated target-decoy (generated from reversed target sequences) database of human proteins from the UniProt *Homo sapiens* reference proteome (UP5640) consisting of SwissProt, TrEMBL, and matching varsplic isoform sequences along with common contaminants (release 2024_05; consisting of 20,434 SwissProt, 22,088 isoform, 62,441 TrEMBL and 75 contaminant entries for 105,038 target sequences in total). The Comet search used a peptide mass tolerance of 20 ppm, a fragment bin tolerance of 1.0005 Da, a fragment bin offset of 0.4 Da, and tryptic enzyme specificity with up to 2 missed cleavages. For modifications, a static cysteine carbamidomethyl (+57.02) modification and variable methionine oxidation (+15.99) modification were used. PSM data was then filtered at the peptide level only using Percolator (v3.1.0) via the OpenMS PercolatorAdaptor[18] with target-decoy competition and q-value scoring, test and training peptide-level FDRs of 0.01, and up to 10 iterations, which was then used as input to pyProteinInference (v1.1.1).

PyProteinInference results were generated utilizing three inference options for which comparator software was available (inclusion, exclusion, and parsimony) with the posterior error probability as the PSM score and all PSMs from the Percolator results (Table S2). For method validation, inclusion results were obtained from PIA's[9] (v1.4.5) "report all" method along with parsimony results from its Occam's Razor algorithm[9] (accessed from within KNIME v4.5.3), while exclusion results were generated using Percolator Protein Inference (v3.07.1)[10] (Table S3).

The raw data file, fasta database, search result files and pyProteinInference inputs and outputs for the K562 data set described in the text above can be found in the UCSD MassIVE repository at https://doi.org/doi:10.25345/C5KW57N8X with accession number MSV000089698, along with PrEST search results generated for this analysis.

RESULTS AND DISCUSSION

Protein Inference Method Comparison to Existing Benchmarks

Each protein inference approach coalesces peptide-level data into a report of proteins or protein groups (in addition to peptide mappings) via different algorithms, influencing which proteins are output in final reports and providing differing "best guesses" that approximate the true protein component of the original sample (Figure S1). Exclusion provides the most conservative estimate, excluding all nonglobally unique peptides from further analysis and ensuring that each protein

https://doi.org/10.1021/acs.jproteome.4c00734
J. Proteome Res. 2025, 24, 2135−2140

has at least one distinguishable peptide to support its presence. Parsimony maps all of the peptides to a minimal set of proteins able to fully explain the results, without discarding any peptide data; peptide-centric assigns peptides to all possible proteins and then reduces redundancy by coalescing peptides into groups of proteins that map to the exact same set of peptides. These two methods differ in that parsimony minimizes the number of proteins reported, while the peptide-centric approach creates groups of peptides which match to one or more of the same proteins but cannot distinguish between them. The most permissive method, inclusion, assigns peptides to all possible proteins without regard to redundancy. In addition to multiple inference assignment methods, differences in protein scoring approaches can also lead to differences in the results.[10] Thus, pyProteinInference implements an exhaustive set of seven protein scoring algorithms including common approaches such as multiplicative log, best peptide per protein, iterative down weighted log, down weighted multiplicative log, top two combined, geometric mean, and simple additive. While these methods are not a comprehensive catalog of all of the inference methods reported in the literature, they were chosen due to their diversity of representation.

To validate the performance of the inference algorithms implemented within pyProteinInference and to highlight the differences in outcomes between the different protein inference methods, we compared the number of identified proteins at specified protein false discovery rates (FDR) between each protein inference method integrated into pyProteinInference using two different methodologies: a comparison to published benchmarks of clean-room implementations of the methods using recombinant protein fragments (PrESTs)[13] and to real-world implementations of algorithms within popular stand-alone tools in the context of a whole-cell K562 lysate mimicking a common proteome-wide analysis experiment.

To test the performance and validity of the protein inference methods integrated into pyProteinInference, we compared the number of significant proteins identified at a 5% entrapment false discovery rate from pyProteinInference for all four methods (parsimony, inclusion, exclusion, and peptide-centric using a best-peptide-per-protein score) to the PrEST results (sets "A" and "A+B") generated in Table 1 of The et al.[13] (Table 1, Figure S2). PyProteinInference shows strong concordance with the previously published benchmarks, differing by less than 1.68% in all methods except for parsimony for the A set, where the difference was approximately 4.19%. Some variance may be expected due to reanalysis of the raw benchmark data with different versions of search and filtering algorithms than in the original study.

Table 1. Results of PrEST Benchmark Analysis Showing the Number of PrESTs Identified at a 5% Entrapment FDR from pyProteinInference Results and the Results Generated by The *et al.*[13]

Inference Principle	*The et al. 2019 − A*	*The et al. 2019 − A+B*	*pyProteinInference − A*	*pyProteinInference − A + B*
Anticipated # PrESTs	191 × 1.05 = 201	382 × 1.05 = 401	191 × 1.05 = 201	382 × 1.05 = 401
Inclusion	0	395	0	395
Exclusion	185	355	186	361
Parsimony	174	365	167	368
Peptide Centric	NA	NA	190	337

To compare the performance of pyProteinInference as a unified alternative for popular implementations of the available algorithms, we analyzed a whole-cell lysate K562 data set with pyProteinInference alongside results from PIA[9] and Percolator Protein Inference.[10] Examining the individual methods, we see that inclusion identifies the most proteins in the K562 data set (19,026) followed by peptide-centric (14,864), parsimony (4,142), and finally exclusion (1,056), which identifies the fewest proteins at a 1% FDR. Each method identifies measurably different numbers of proteins at an identical FDR. As expected, we observed the most restrictive method to be exclusion followed by parsimony, peptide-centric, and finally inclusion, being the least restrictive method. While the different methods may have considerable similarity in the number of proteins reported overall, they are not identical. Full overlap of proteins between the different inference methods can be found in the UpSet plot in Figure 1. These differences are not unexpected given the different ways in which each method maps peptides to proteins and leads to different significant proteins (at a 1% FDR) being identified depending on the protein inference method used, especially in the case of highly similar proteins (such as isoforms or computationally annotated sequences from TrEMBL). These differences highlight the necessity of choosing an appropriate inference method tailored for the system under study, which should not depend on maximizing the number of proteins identified but rather on validating the biological hypothesis being tested. More stringent methods such as exclusion and parsimony can produce higher overall specificity necessary for validating specific mechanistic models with small, nonredundant databases, while the more permissive results generated within the peptide-centric approach may be more appropriate for screening-based approaches; inclusion methods, while not generally suitable as a representation of the likely set of proteins identified, may still be useful to understand the mapping of individual PSMs to different proteoforms, critical for the study of specific protein interactions. Given the differences in the number of proteins identified between different inference methods and the measurable set differences between protein inference methods, the flexibility of employing alternative protein inference approaches beyond those bundled with other toolsets is readily apparent.

To compare the implementations of inference approaches in pyProteinInference compared to corresponding published tools, FDR correlation plots of the K562 data set were generated. FDR correlation plots (Figure S3A−C) show our implementations to be in agreement with published algorithms. Spearman correlations for FDR values up to 0.20 (20%) across proteins were computed between data generated from the pyProteinInference inference methods and the data generated from the comparator published protein inference tools and were approximately 0.99 for each comparison. We also compared the overlap of proteins identified in our results against those obtained from previously published tools at a 1% protein FDR (Figure S3D−F), a commonly applied protein FDR in many proteomic experiments. Percent overlap between pyProteinInference and the comparator algorithms exceeded 99% across all comparisons at a 1% FDR. All algorithms in pyProteinInference performed virtually identically to implementations currently available; the minor differences observed between pyProteinInference and other implementations may

https://doi.org/10.1021/acs.jproteome.4c00734
J. Proteome Res. 2025, 24, 2135−2140

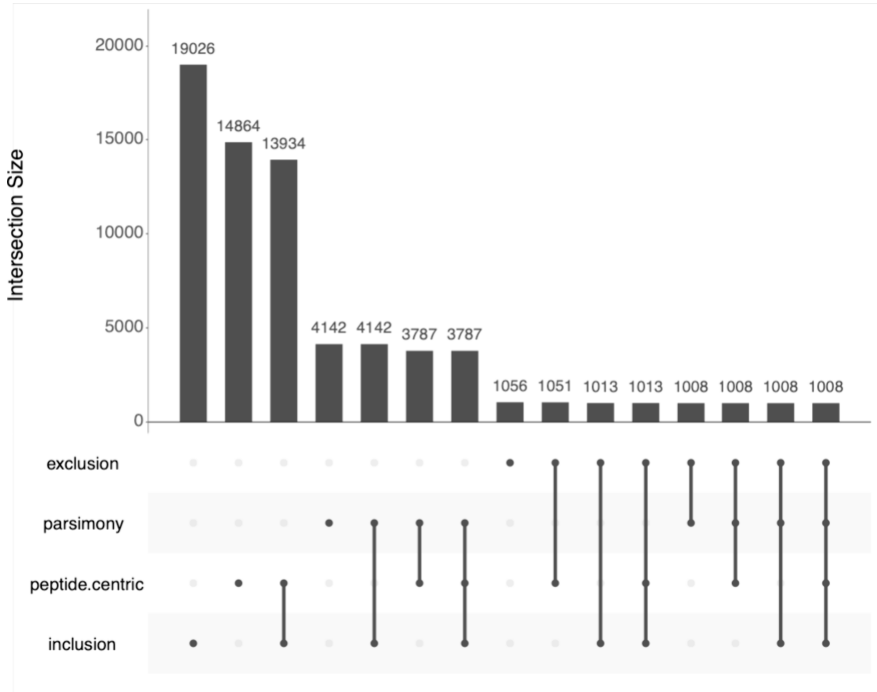


Figure 1. Upset plot showing the overlap of identified proteins at a 1% protein false discovery rate (FDR) from the K562 whole cell lysate data set. The X axis indicates the individual method or overlapping methods examined, while the Y axis indicates the intersection size between methods on the X axis. Peptide-centric protein groups are expanded out to individual proteins to facilitate comparisons to other methods that only output protein group leads; individual methods show total counts for comparison purposes (not only counts unique to the method).

be attributed to differences in filtering (e.g., based on minimum score or peptide length) in addition to the stochastic nature of some aspects of the algorithms themselves (i.e., choosing randomly between two assignments of equal weight).

Beyond the implementation of the most popular protein inference algorithms within a single tool, the overall aim of pyProteinInference is to make these methods easily accessible to bench biologists, computational scientists, and scientific software developers alike. The tool can be downloaded and run as a standalone executable program with a graphical user interface for ease of use and can also be run from the command line or via Docker container for easy incorporation into large-scale analysis pipelines. Finally, as a Python package, pyProteinInference can be used as a library to enable inference approaches as part of other scientific software.

CONCLUSIONS

PyProteinInference is a comprehensive software package that implements multiple inference methods for tandem MS/MS data in a single interface, for biologists and computational analysts alike. By providing a single unified set of tools for inference analysis and FDR summarization methods, pyProteinInference provides an easy-to-use standalone software set that can be used in conjunction with a variety of search methods, including both data-dependent and data-independent approaches. By following the Unix Philosophy of minimalist, modular development,[21] it requires no additional dependencies or larger frameworks to be installed, making it a portable and flexible approach that can be run on a single desktop or as part of automated analysis pipelines. This makes it stand apart from other software tools which may be tied to specific data formats, including vendor provided software such as Proteome Discoverer (ThermoFisher Scientific, Waltham, Massachusetts), difficult for the average user to install and configure, or incorporate inference into larger application suites for reanalysis of data or integration of multiple search results (e.g., PeptideShaker,[22] PIA[9]). By providing a powerful, integrated tool with a low barrier to entry, pyProteinInference helps researchers maximize understanding of their proteomics data sets and their application to interrogate both biology and disease.

ASSOCIATED CONTENT

Data Availability Statement

PyProteinInference can be installed using pip. Source code is available at: https://github.com/thinkle12/pyproteininference and package documentation is available at: https://thinkle12.github.io/pyproteininference.

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jproteome.4c00734.

Protein Inference Methods Diagram (Figure S1), PrEST Entrapment FDR Plots (Figure S2), and PyProteinInference Comparison to Other Tools (Figure S3) (PDF)
PrEST PyProteinInference Data (Table S1) (XLSX)
K562 PyProteinInference Data (Table S2) (XLSX)
K562 External Protein Inference Data (Table S3) (XLSX)

AUTHOR INFORMATION

Corresponding Author

Corey E. Bakalarski − *Department of Proteomic & Genomic Technologies and Department of Computational Catalysts, Genentech Inc., South San Francisco, California 94080, United States;* orcid.org/0000-0003-1085-7828; Email: bakalarski.corey@gene.com

https://doi.org/10.1021/acs.jproteome.4c00734
J. Proteome Res. 2025, 24, 2135−2140

Author

Trent B. Hinkle − *Department of Proteomic & Genomic Technologies, Genentech Inc., South San Francisco, California 94080, United States*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jproteome.4c00734

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

We thank Tommy Cheung for generation of the K562 dataset; figures created with BioRender.com.

■ REFERENCES

(1) Yates, J. R., III Recent Technical Advances in Proteomics. *F1000Research* **2019**, *8*, F1000 Faculty Rev-351. 351.
(2) 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.
(3) Demichev, V.; Messner, C. B.; Vernardis, S. I.; Lilley, K. S.; Ralser, M. DIA-NN: Neural Networks and Interference Correction Enable Deep Proteome Coverage in High Throughput. *Nat. Methods* **2020**, *17* (1), 41−44.
(4) Eng, J. K.; Jahan, T. A.; Hoopmann, M. R. Comet: An Open-source MS/MS Sequence Database Search Tool. *PROTEOMICS* **2013**, *13* (1), 22−24.
(5) Nesvizhskii, A. I.; Aebersold, R. Interpretation of Shotgun Proteomic Data: The Protein Inference Problem. *Mol. Cell Proteomics* **2005**, *4* (10), 1419−1440.
(6) Rappsilber, J.; Mann, M. What Does It Mean to Identify a Protein in Proteomics? *Trends Biochem. Sci.* **2002**, *27* (2), 74−78.
(7) Nesvizhskii, A. I.; Keller, A.; Kolker, E.; Aebersold, R. A Statistical Model for Identifying Proteins by Tandem Mass Spectrometry. *Anal. Chem.* **2003**, *75* (17), 4646−4658.
(8) Serang, O.; MacCoss, M. J.; Noble, W. S. Efficient Marginalization to Compute Protein Posterior Probabilities from Shotgun Mass Spectrometry Data. *J. Proteome Res.* **2010**, *9* (10), 5346−5357.
(9) Uszkoreit, J.; Maerkens, A.; Perez-Riverol, Y.; Meyer, H. E.; Marcus, K.; Stephan, C.; Kohlbacher, O.; Eisenacher, M. PIA: An Intuitive Protein Inference Engine with a Web-Based User Interface. *J. Proteome Res.* **2015**, *14* (7), 2988−2997.
(10) The, M.; MacCoss, M. J.; Noble, W. S.; Käll, L. Fast and Accurate Protein False Discovery Rates on Large-Scale Proteomics Data Sets with Percolator 3.0. *J. Am. Soc. Mass Spectrom.* **2016**, *27* (11), 1719−1727.
(11) Audain, E.; Uszkoreit, J.; Sachsenberg, T.; Pfeuffer, J.; Liang, X.; Hermjakob, H.; Sanchez, A.; Eisenacher, M.; Reinert, K.; Tabb, D. L.; Kohlbacher, O.; Perez-Riverol, Y. In-Depth Analysis of Protein Inference Algorithms Using Multiple Search Engines and Well-Defined Metrics. *J. Proteom.* **2017**, *150*, 170−182.
(12) Claassen, M.; Reiter, L.; Hengartner, M. O.; Buhmann, J. M.; Aebersold, R. Generic Comparison of Protein Inference Engines. *Mol. Cell Proteomics* **2012**, *11* (4), O110.007088.
(13) The, M.; Edfors, F.; Perez-Riverol, Y.; Payne, S. H.; Hoopmann, M. R.; Palmblad, M.; Forsström, B.; Käll, L. A Protein Standard That Emulates Homology for the Characterization of Protein Inference Algorithms. *J. Proteome Res.* **2018**, *17* (5), 1879−1886.
(14) Venkatanarayan, A.; Liang, J.; Yen, I.; Shanahan, F.; Haley, B.; Phu, L.; Verschueren, E.; Hinkle, T. B.; Kan, D.; Segal, E.; Long, J. E.; Lima, T.; Liau, N. P. D.; Sudhamsu, J.; Li, J.; Klijn, C.; Piskol, R.; Junttila, M. R.; Shaw, A. S.; Merchant, M.; Chang, M. T.; Kirkpatrick, D. S.; Malek, S. CRAF Dimerization with ARAF Regulates KRAS-Driven Tumor Growth. *Cell Reports* **2022**, *38* (6), 110351.
(15) Voss, K.; Auwera, G. V. D.; Gentry, J. Full-Stack Genomics Pipelining with GATK4 + WDL + Cromwell [Version 1; Not Peer Reviewed]. *ISCB Comm J.* **2017**, *6*, 1381.
(16) 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.
(17) Käll, 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), 923−925.
(18) 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.; Röst, H.; Kohlbacher, O.; Sachsenberg, T. OpenMS 3 Enables Reproducible Analysis of Large-Scale Mass Spectrometry Data. *Nat. Methods* **2024**, *21* (3), 365−367.
(19) Zhang, B.; Chambers, M. C.; Tabb, D. L. Proteomic Parsimony through Bipartite Graph Analysis Improves Accuracy and Transparency. *J. Proteome Res.* **2007**, *6* (9), 3549−3557.
(20) Meyer-Arendt, K.; Old, W. M.; Houel, S.; Renganathan, K.; Eichelberger, B.; Resing, K. A.; Ahn, N. G. IsoformResolver: A Peptide-Centric Algorithm for Protein Inference. *J. Proteome Res.* **2011**, *10* (7), 3060−3075.
(21) Pike, R.; Kernighan, B. W. The UNIX System: Program Design in the UNIX Environment. *T Bell Lab. Technol. J.* **1984**, *63* (8), 1595−1605.
(22) 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), 22−24.