ProtyQuant: Comparing label-free shotgun proteomics datasets using accumulated peptide probabilities

Robert Winkler*

Center for Research and Advanced Studies (CINVESTAV) Irapuato, Department of Biochemistry and Biotechnology, Km. 9.6 Libramiento Norte Carr. Irapuato-León, 36824 Irapuato Gto., Mexico

E-mail: robert.winkler@cinvestav.mx

Phone: +52 (462) 623 9635

Abstract

Comparing multiple label-free shotgun proteomics datasets requires various data processing and formatting steps, including peptide-spectrum matching, protein inference, and quantification. Finally, the compilation of results files into a format that allows for downstream analyses. ProtyQuant performs protein inference and quantification calculations, and combines the results of individual datasets into plain text tables. These are lightweight, human-readable, and easy to import into databases or statistical software. ProtyQuant reads validated pepXML from proteomic workflows such as the Trans-Proteomic Pipeline (TPP), which makes it compatible with many commercial and free search engines. For protein inference and quantification, a modified version of the PIPQ program (He et al. 2016) was integrated. In contrast to simple spectral-counting, PIPQ sums up peptide probabilities. For assigning peptides to proteins, three algorithms are available: Multiple Counting, Equal Division, and Linear Programming. The accumulated peptide probabilities (app) are used for both tasks, protein probability estimation, and quantification. ProtyQuant was tested using a reference dataset for label-free shotgun proteomics, obtained from different concentrations of 48 human UPS proteins spiked into yeast lysate. Compared to ProteinProphet, ProtyQuant detected up to 126 (15%) more proteins in the mixture, applying an equal false positive rate (FPR). Using the app values for label-free quantification showed suitable sensitivity and linearity. Strikingly, the app values represent a realistic measure of 'Protein Presence,' an integral concept of protein probability and quantity. ProtyQuant provides a graphical user interface (GUI) and scripts for console-based processing. It is available (GNU GLP v3) for Windows, Linux, and Docker from `https://bitbucket.org/lababi/protyquant/`.

Keywords

shotgun proteomics, protein inference, label-free quantification

Introduction

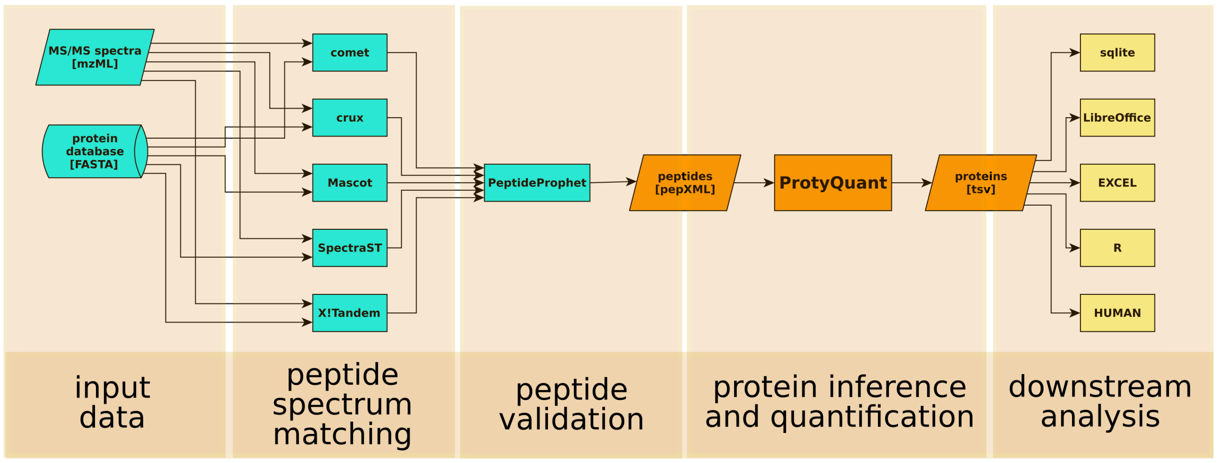


Figure 1: General shotgun proteomics flowchart with integration of ProtyQuant. First, MS/MS spectra are matched against a database of protein sequences. Following, the peptide-spectrum matches (PSM) are validated. ProtyQuant processes the PSM to calculate the accumulated peptide probabilities (app) of each protein and its probability (Pr). The results are exported into tabulator separated value (TSV) files, facilitating direct inspection or further analysis in downstream software.

Shotgun proteomics is a standard research method in biology and medicine. Several informatics platforms, such as crux[1,2], OpenMS/KNIME[3] and the Trans-Proteomic Pipeline (TPP)[4–6], perform label-free quantification and export the results in text files. However, the data processing for comparative proteomics is still fiddly for non-experts. Thus, the focus of this study was to simplify the data processing for end-users, who are interested in comparing label-free shotgun proteomics datasets, either manually or with external software.

Transforming raw mass spectrometry data into quantitative protein hits is a multistep procedure, as shown in **Figure 1**.

A critical step after searching peptide-spectrum matches (PSM) is the protein inference, i.e., the assembly of valid proteins from the peptide hits. The cleavage of proteins leads to degenerated peptides, leading to ambiguity in finding peptide → protein relationships (see **Fig. 2**). Numerous algorithms have been reported to solve the 'protein inference problem'[7,8]; however, there is no consistently best-performing PSM/protein inference combination found yet[9]. For testing and benchmarking of ProtyQuant, Comet was used as PSM search engine, PeptideProphet[10] for peptide hit validation, and the ProteinProphet[11] for protein inference, since those programs are part of the well established TPP. The correctness of protein hits was evaluated by using a target-decoy approach[12], and false-positive rate calculations[13].

Quantification of proteins is usually done after protein inference, and several labeling and label-free strategies have been reported[vaudel_chapter_2020]. For label-free quantification, spectral-counting is a simple and computationally efficient method, which has been implemented into different proteomics pipelines[14–17].

A new concept of integrating the protein inference and quantification was introduced by Huang et al.[18] and He et al.[19]. It is based on the understanding that the true presence of proteins is a special case of protein quantification[18,19]. For quantification, the authors refined the conventional spectral-counting strategy. Instead of counting identified PSM with an equal weight of '1', their score or probability values are summed up:

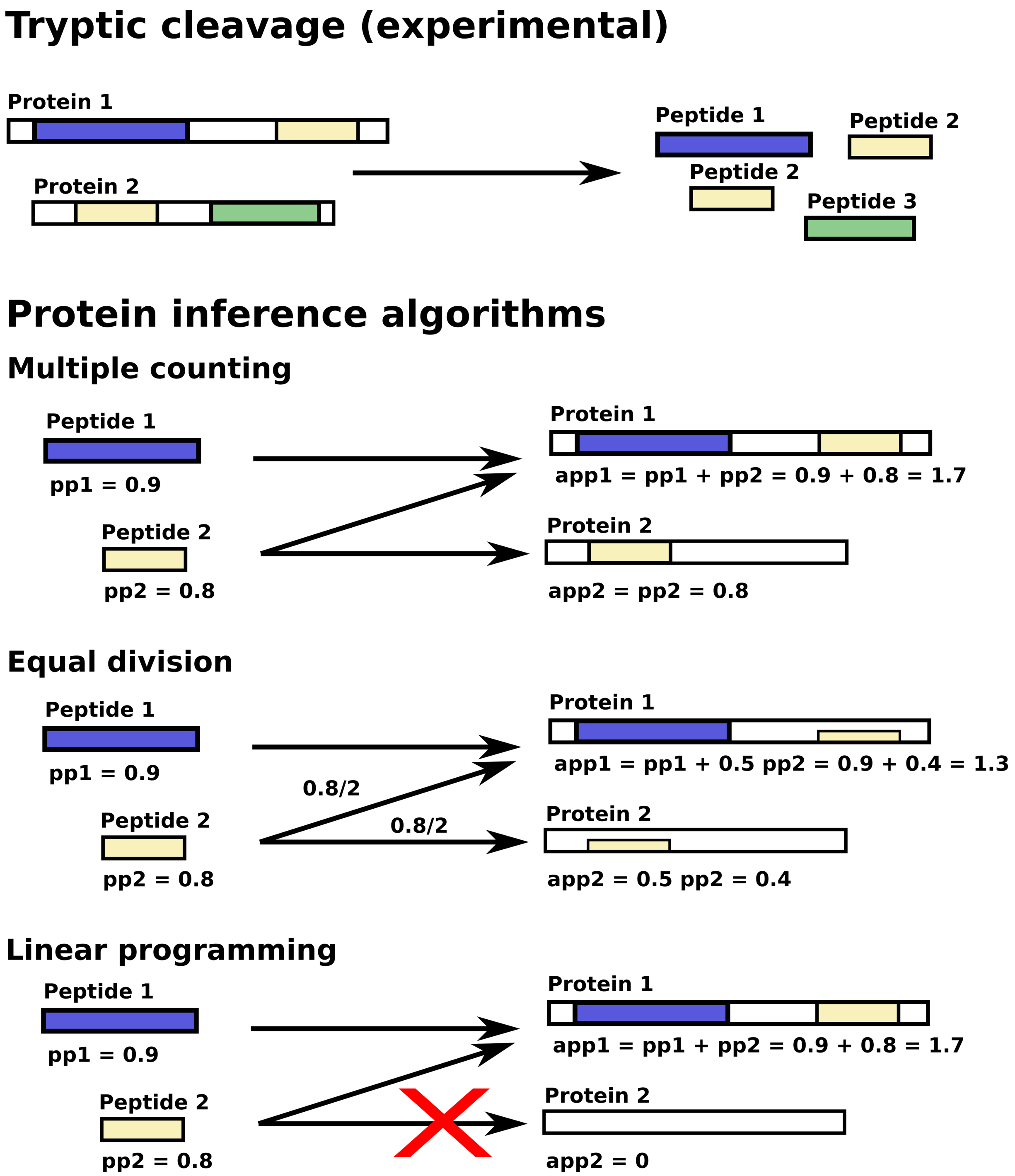


Figure 2: Creation of degenerated peptides by tryptic cleavage and protein inference algorithms implemented in ProtyQuant.

$$b_j = \sum_{(x_i, y_j) \in E_1} a_i \tag{1}$$

where b_j is the summed peptide probability of peptide y_j, a_i the probability of spectrum x_i matching y_i, and $(x_i, y_j) \in E_1$ the PSM.

This refined method improves the quantification and discrimination of proteins. The peptides are assigned to candidate proteins using the inference algorithms presented in **Figure 2**. The Multiple Counting algorithm adds the peptides to all possible proteins (**Equ. 2**), the Equal Division algorithm splits the peptide quantity proportionally (**Equ.3**), and the Linear Programming algorithm[20] optimizes the peptide-protein distribution (**Equ. 4**):

$$app_k = \sum_{(y_j, z_k) \in E_2} b_j \tag{2}$$

$$app_k = \sum_{(y_j, z_k) \in E_2} \frac{b_j}{q_j} \tag{3}$$

$$app_k = \sum_{\{j | (y_j, z_k) \in E_2\}} d_{jk} \tag{4}$$

where $(y_j, z_k) \in E_2$ means that peptide y_j is a part of the sequence of protein z_k, q_j the number of proteins with the same peptide y_j, and d_{jk} is the abundance contribution of each protein z_k to the peptide y_i. Details on the development of the formula can be found in the papers of Huang et al.[18] and He et al.[19]. In contrast to the other algorithms, the Linear Programming algorithm is capable of eliminating low-scoring protein candidates.

As a result, for each possible protein k, an accumulated peptide probability (app_k) is obtained. These quantitative protein scores can be transformed into protein probabilities (Pr) using a method reported by Gao and Tan 2006;[21]. The algorithms for app-counting and protein inferences were implemented in a C++ program, PIPQ, which was published under the terms of GNU GPL v3 (`http://code.google.com/p/protein-inference/`)[18,19]. The original program was modified for quantifying proteins in ProtyQuant, based on their calculated app values.

The objective of this study was the development of a software tool that simplifies the comparison of multiple shotgun proteomics datasets. The program should provide the following functions:

- Creation of sample protein reports and a sample comparison table.
- Use of plain text table format (readable for humans and downstream software) for results.
- Annotation of results with information for interpretation (protein names and Uniprot link).
- Compatibility with community file standards such as pepXML and FASTA.
- Implementation of PIPQ for protein inference and quantification.
- Computationally efficient and cross-platform compatible.
- Graphical User Interface (GUI) and usable for non-experts.

This paper describes the program ProtyQuant and demonstrates its functionality by analyzing a reference dataset for label-free shotgun proteomics.

Methods

ProtyQuant software architecture

ProtyQuant is a Python (`https://www.python.org/`) program with a Graphical User Interface (GUI). **Figure 3** shows the program flowchart. ProtyQuant processes all pepXML files in a given directory and creates a sample comparison table. For each sample, a report with protein probability and estimated quantity is created. All result files are formatted as plain text tables. Optionally, the protein identifications are labeled with protein information and Uniprot links, using a FASTA database.

Protein inference and quantification are made by the external program PIPQ, which was originally written by Huang et al.[18] (`https://code.google.com/archive/p/protein-inference/`)[18,19]. The C++ program was modified to report both, the protein probability (Pr), and its accumulated peptide probability (app), which was used as a quantitative measure. The formulas which were used

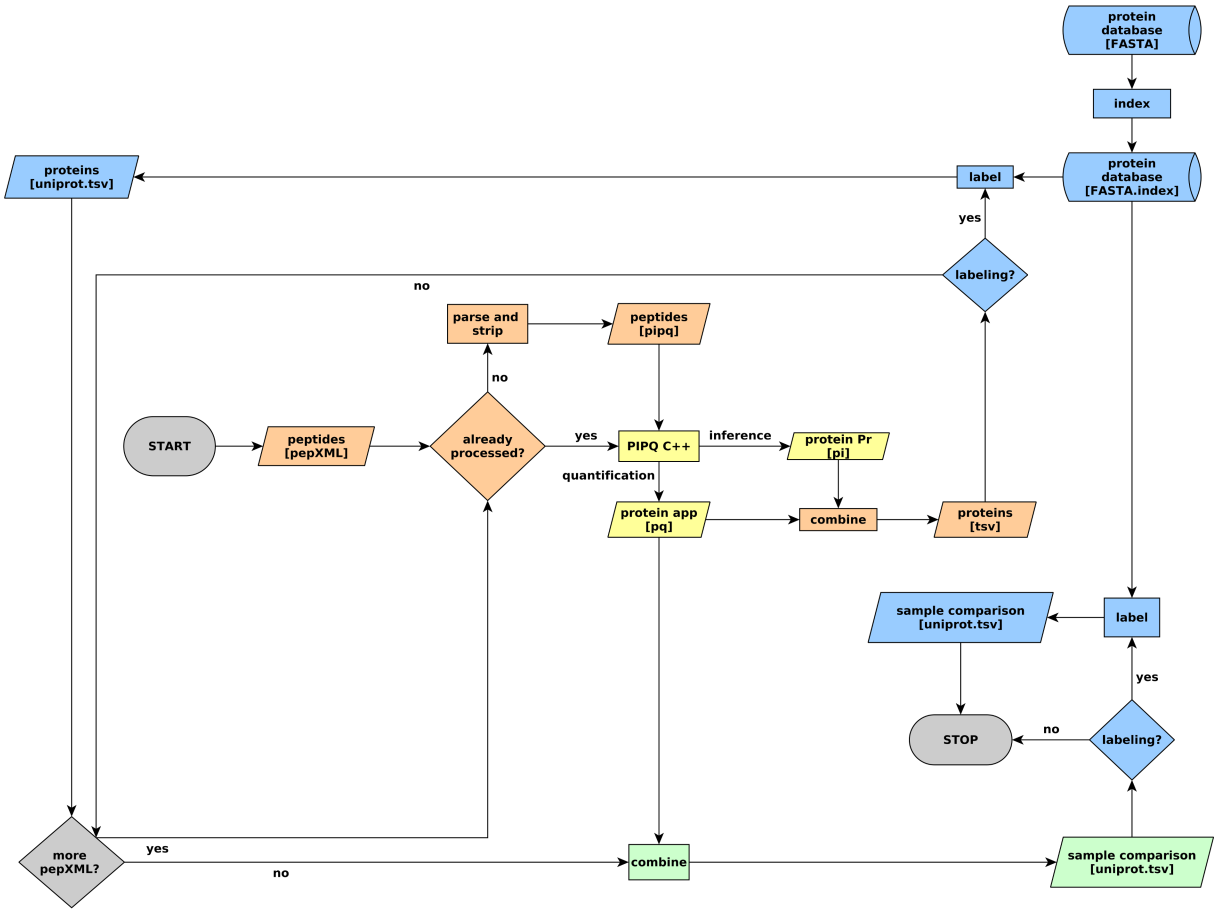
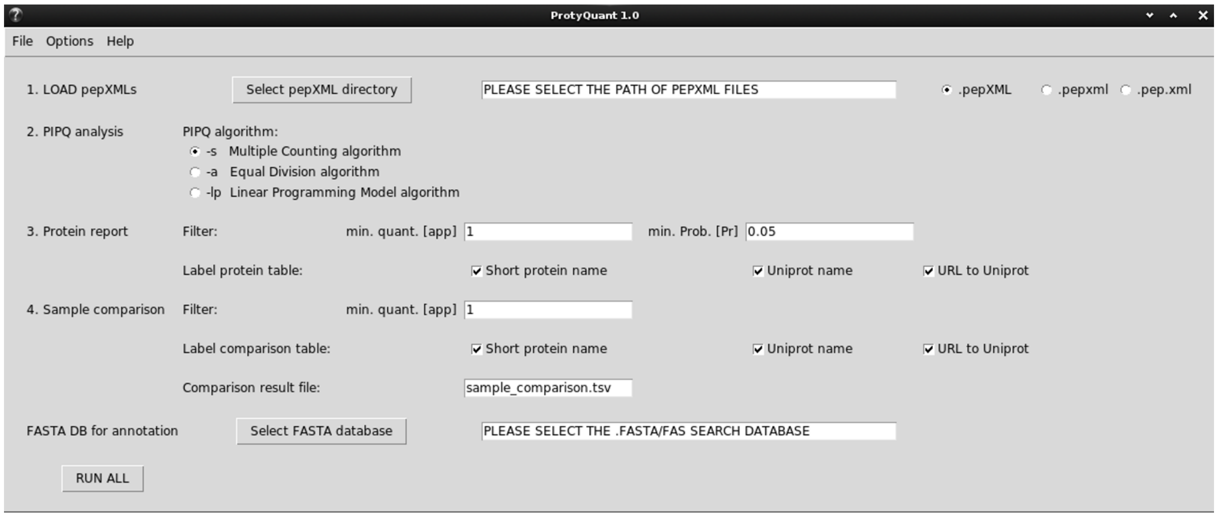


Figure 3: Data processing flowchart of ProtyQuant. Peptide XML and protein text file processing are colored in orange, PIPQ protein inference and quantification in yellow, sample comparison in green, and protein labeling in blue. The main program control is drawn in gray.

for calculations are reported in the paper of He et al.[19]. For solving the linear equations GNU Linear Programming Kit library, GLPK (`https://www.gnu.org/software/glpk/`), was used. PIPQ was compiled for 64-bit versions of Microsoft Windows and Linux, with statical linking of libraries to prevent compatibility issues. The Microsoft Visual C++ and the GNU C++ compiler were used for creating the binaries. The code repository (`https://bitbucket.org/lababi/protyquant`) contains makefiles for both platforms.


ProtyQuant 1.0

File Options Help

1. LOAD pepXMLs — Select pepXML directory — PLEASE SELECT THE PATH OF PEPXML FILES — [✓ .pepXML] .pepxml .pep.xml

2. PIPQ analysis — PIPQ algorithm:
- [✓ -s Multiple Counting algorithm]
- -a Equal Division algorithm
- -lp Linear Programming Model algorithm

3. Protein report — Filter: min. quant. [app] 1 — min. Prob. [Pr] 0.05

Label protein table: [✓ Short protein name] [✓ Uniprot name] [✓ URL to Uniprot]

4. Sample comparison — Filter: min. quant. [app] 1

Label comparison table: [✓ Short protein name] [✓ Uniprot name] [✓ URL to Uniprot]

Comparison result file: sample_comparison.tsv

FASTA DB for annotation — Select FASTA database — PLEASE SELECT THE .FASTA/FAS SEARCH DATABASE

RUN ALL

Figure 4: Graphical User Interface of ProtyQuant.

The ProtyQuant GUI and command-line scripts were implemented in Python 3 `https://www.python.org`, using the Tkinter GUI and the Python Data Analysis (Pandas) Library (`https://pandas.pydata.org/`) modules. **Figure 4** shows the ProtyQuant GUI running on Linux.

Running ProtyQuant requires two files only: The Python program `protyquant.py` and the PIPQ binary. The 'python-gui' directory of the code repository contains PIPQ for Microsoft Windows and Linux, `PIPQ.exe` and `PIPQ.x64`, respectively. Running ProtyQuant by `python3 protyquant.py` selects the correct program for the operating system.

The GUI was converted into a 64-bit Microsoft Windows executable (.exe) using the Python package cx_Freeze (`https://anthony-tuininga.github.io/cx_Freeze/`). A Microsoft Windows installer, which also installs the PIPQ.exe program, was created with Inno Script Studio 2.2.2.32 (Kymoto Solutions, `https://www.kymoto.org/`).

Further, a Docker (`https://docker.com`) version of the ProtyQuant GUI was created. The image is based on Ubuntu 20.04 LTS and available from `https://hub.docker.com/repository/docker/robertmp64/protyquant`.

Source code and binaries for Microsoft Windows (including an installer) and Linux are freely available under the terms of the GNU General Public License v3 (`https://www.gnu.org/licenses/gpl-3.0.en.html`), from `https://bitbucket.org/lababi/protyquant`.

Raw data and protein database

For testing ProtyQuant, a reference dataset for label-free quantitative proteomics was used, which was published by Ramus et al. 2016[22,23]. The raw Orbitrap Velos data were downloaded from the ProteomeXchange repository (`http://www.proteomexchange.org`)[24], identifier PXD001819. The samples of this dataset were generated by spiking different concentrations of the Universal Proteomics Standard (UPS1, `https://www.sigmaaldrich.com/life-science/proteomics/mass-spectrometry/ups1-and-ups2-proteomic.html`) into baker's yeast (*Saccharomyces cerevisiae*) lysate. The 48 human proteins included in the UPS1 are listed in **Supplemental Table 1**. The raw data were converted to .mzML profile data and .mgf centroid data using `msconvert` of the ProteoWizard project (http://proteowizard.sourceforge.net)[25].

The target database for peptide matching was composed of the UniProt (`https://www.uniprot.org`) *Saccharomyces cerevisiae* entries and the Sigma UPS sequences (`https://www.sigmaaldrich.com/content/dam/sigma-aldrich/life-science/proteomics-and-protein/ups1-ups2-sequences.fasta`). The final FASTA database contained 6,769 entries.

Proteomics data processing

Computer hardware and operating system

All analyses were performed on a standard Lenovo Y720 laptop with 16 Gb RAM and Intel(R) Core(TM) i7-7700HQ CPU (2.80GHz). The operating system was the 64-bit Linux platform Peppermint 10 (`https://peppermintos.com/`), which is based on Ubuntu 18.04 LTS. The Docker Engine was version 19.03.8 of the Community Edition.

Trans-Proteomic Pipeline (TPP) with Docker

The Docker image of the Trans-Proteomic Pipeline (TPP) version 5.2.0 ('Flammagenitus') was used, with the following sequence:

1. **Start the TPP docker image** and **mount the mgf data directory**:

```
docker run -it --privileged=true -v /home/rob/dataspace/nextcloud/DATA/
UPS48_yeast_centroided/mgf:/data spctools/tpp bash
```

2. **Create and edit a comet parameters file**:

```
comet -p
```

The file was saved as `comet.params` and modified for high-resolution MS (`peptide_mass_tolerance = 20.00`, in ppm) and low-resolution MS/MS (`fragment_bin_tol = 1.0005`) data. `decoy_search = 1` (`decoy_prefix = DECOY_`) was set for a concatenated target-decoy search[12]. Iodoacetamide alkylation of cysteine residues was defined as a fixed modification, and methionine oxidation, and deamidation of asparagine and glutamine as variable modifications.

3. **Run the comet search**:

```
time comet *.mgf
```

4. **Run PeptideProphet** with the standard mixture model:

```
for i in *.pep.xml; do PeptideProphetParser $i; done
```

5. **Run ProteinProphet**:

```
for i in *.pep.xml; do ProteinProphet $i $i.prot.xml NOGROUPS; done
```

6. **Leave TPP Docker session**:

```
exit
```

The `prot.xml` result files were **converted to plain text tables** by an own script:

1. **Download script**:

```
wget https://bitbucket.org/lababi/protyquant/raw/
06deaeb70a09b8121ce0adc1d7d6da389afe7175/python-scripts/protxml_to_tsv.py
```

2. **Convert all prot.xml files to tsv**:

```
bash-script: for i in *.prot.xml; do python3 protxml_to_tsv.py $i; done
```

ProtyQuant with Docker

The Docker GUI version of ProtyQuant was used, with all Pr and app threshold values set to 0.

```
docker run -e DISPLAY=$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix:rw
-v /home/rob/dataspace:/data robertmp64/protyquant
```

Opening the graphical display with the root user was enabled by `xhost +`.

Calculation of False Positive Rates

The false positive rate (FPR) was taken as evaluation criteria for the algorithms to prevent possible drawbacks in estimating the false discovery rate (FDR)[13]. The FPR was calculated by dividing the number of decoy hits by the number of total hits above the defined threshold[12]:

$$FPR = \frac{FP}{TP + FP} = \frac{N_{decoy}}{N_{targets} + N_{decoy}} = \frac{N_{decoy}}{N_{hits \geq cut-off}} \tag{5}$$

with FP = false positives = decoy hits and TP = true positives = target hits.

Data analysis and plotting

For analyzing result files of ProtyQuant, standard GNU (<`https://www.gnu.org`>) command-line programs (grep, wc), the text editors Geany (`https://www.geany.org/`) and Vim (<`https://www.vim.org`>) and LibreOffice (`https://www.libreoffice.org/`) were used. Graphs were plotted with Gnuplot (`http://www.gnuplot.info/`).

Receiver Operating Characteristics (ROC) and the Area Under the Curve (AUC) were calculated using the Python scikit-learn library (`https://scikit-learn.org`). The used scripts are available from the `python-scripts` directory of the code repository (`https://bitbucket.org/lababi/protyquant`).

Flowcharts were drawn with yEd (<`https://www.yworks.com/products/yed`>), vector graphics were created with Inkscape (<`https://inkscape.org`>), and the GNU Image Manipulation Program (GIMP, <`https://www.gimp.org`>) was used for editing pixel graphics.

Results and discussion

ProtyQuant

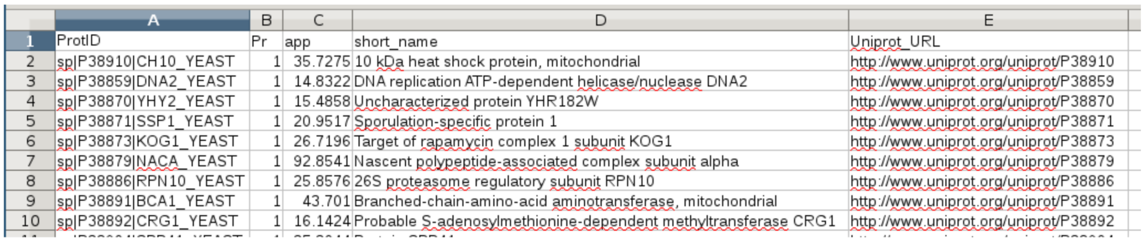
Installation on different operating systems

ProtyQuant was successfully installed and tested on standard computers with Microsoft Windows 10 and Linux (Fedora 29-31 and Ubuntu 18.04 LTS) operating systems. Installation instructions are given on the project code repository (`https://bitbucket.org/lababi/protyquant`). The results presented below were generated using the Docker version with GUI, running on a Ubuntu 18.04 LTS Linux.

Input and output file formats

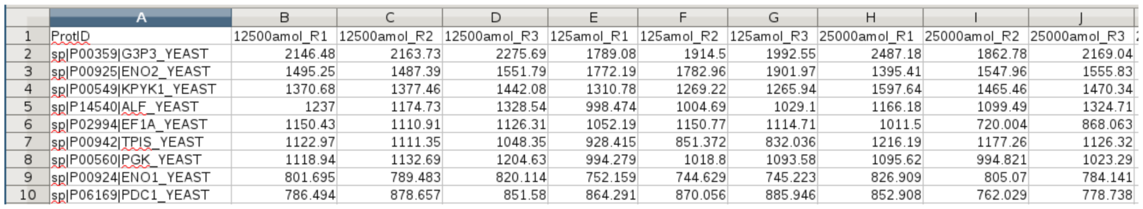
ProtyQuant expects pepXML files containing PeptideProphet[10] probabilities. Such files are generated validating peptide-spectrum matches (PSM) in the Trans-Proteomic Pipeline (TPP)[4–6]. The

A



	A	B	C	D	E
1	ProtID	Pr	app	short_name	Uniprot_URL
2	sp|P38910|CH10_YEAST	1	35.7275	10 kDa heat shock protein, mitochondrial	http://www.uniprot.org/uniprot/P38910
3	sp|P38859|DNA2_YEAST	1	14.8322	DNA replication ATP-dependent helicase/nuclease DNA2	http://www.uniprot.org/uniprot/P38859
4	sp|P38870|YHY2_YEAST	1	15.4858	Uncharacterized protein YHR182W	http://www.uniprot.org/uniprot/P38870
5	sp|P38871|SSP1_YEAST	1	20.9517	Sporulation-specific protein 1	http://www.uniprot.org/uniprot/P38871
6	sp|P38873|KOG1_YEAST	1	26.7196	Target of rapamycin complex 1 subunit KOG1	http://www.uniprot.org/uniprot/P38873
7	sp|P38879|NACA_YEAST	1	92.8541	Nascent polypeptide-associated complex subunit alpha	http://www.uniprot.org/uniprot/P38879
8	sp|P38886|RPN10_YEAST	1	25.8576	26S proteasome regulatory subunit RPN10	http://www.uniprot.org/uniprot/P38886
9	sp|P38891|BCA1_YEAST	1	43.701	Branched-chain-amino-acid aminotransferase, mitochondrial	http://www.uniprot.org/uniprot/P38891
10	sp|P38892|CRG1_YEAST	1	16.1424	Probable S-adenosylmethionine-dependent methyltransferase CRG1	http://www.uniprot.org/uniprot/P38892

B



	A	B	C	D	E	F	G	H	I	J
1	ProtID	12500amol_R1	12500amol_R2	12500amol_R3	125amol_R1	125amol_R2	125amol_R3	25000amol_R1	25000amol_R2	25000amol_R3
2	sp|P00359|G3P3_YEAST	2146.48	2163.73	2275.69	1789.08	1914.5	1992.55	2487.18	1862.78	2169.04
3	sp|P00925|ENO2_YEAST	1495.25	1487.39	1551.79	1772.19	1782.96	1901.97	1395.41	1547.96	1555.83
4	sp|P00549|KPYK1_YEAST	1370.68	1377.46	1442.08	1310.78	1269.22	1265.94	1597.64	1465.46	1470.34
5	sp|P14540|ALF_YEAST	1237	1174.73	1328.54	998.474	1004.69	1029.1	1166.18	1099.49	1324.71
6	sp|P02994|EF1A_YEAST	1150.43	1110.91	1126.31	1052.19	1150.77	1114.71	1011.5	720.004	868.063
7	sp|P00942|TPIS_YEAST	1122.97	1111.35	1048.35	928.415	851.372	832.036	1216.19	1177.26	1126.32
8	sp|P00560|PGK_YEAST	1118.94	1132.69	1204.63	994.279	1018.8	1093.58	1095.62	994.821	1023.29
9	sp|P00924|ENO1_YEAST	801.695	789.483	820.114	752.159	744.629	745.223	826.909	805.07	784.141
10	sp|P06169|PDC1_YEAST	786.494	878.657	851.58	864.291	870.056	885.946	852.908	762.029	778.738

Figure 5: ProtyQuant results are written into tab-separated values files (TSV) which facilitates their direct inspection and further processing with other programs, such as statistics and plotting software, databases, and spreadsheets. The files in A) and B) were opened with LibreOffice Calc. A) Protein report with short protein description and links to the UniProt database. B) Sample comparison table (The sample names have been shortened manually).

TPP supports many popular free and commercial search engines[26], such as Comet[27?], Mascot[28], MS-GF+[29], MyriMatch[30], and X!Tandem[31]. Therefore, PSM results of different search engines can be processed with ProtyQuant after TPP validation. The data processing workflow in this study was using Comet, which is a default TPP search engine. All 27 PeptideProphet pepXML files of the test datasets were processed successfully by ProtyQuant.

ProtyQuant saves its results to tab-separated values (TSV) files. The plain text files with the tabular structure are lightweight and human-readable. TSV files can be directly imported into most statistics programs and databases. Therefore, ProtyQuant integrates smoothly into existing software environments[32]. The information of the FASTA protein database file was correctly processed and added to the protein hits (see **Fig. 5A**). Quantification results of the 27 individual analyses were adequately combined in a sample comparison table (see **Fig. 5B**).

Computational performance

Table 1: File formats and approximate average size during the processing of the UPS1-yeast lysate shotgun proteomics reference data.

File	**Format**	**Size**
MS/MS raw profile data	binary	1.6 Gb
MS/MS mzML profile data	XML	7.4 Gb
MS/MS mgf centroided data	plain text	664 Mb
TPP pepXML files	XML	80 Mb
tsv reports	plain text	<500 kb

The speed of a program depends hugely on the computer system performance and chosen parameters. However, the following measurements give a general idea about the computational performance of the ProtyQuant workflow on a standard laptop. The average time for the comet peptide searches was 1.2 min. The first step of ProtyQuant, parsing and stripping the pepXML files to pipq, took 45 s in average. Protein inference with the multiple counting or equal distribution algorithm took about 1 s, and 4 s with the linear programming algorithm. Less than one minute was necessary for the complete processing of TPP pepXML data with ProtyQuant, which is sufficient for productive proteomics environments.

Another critical aspect of proteomics workflows is the size of files. Large files occupy lots of hard drive space and negatively impact the downstream data processing. **Table 1** summarizes the file format and size for each processing step. Converting the MS/MS raw data from the proprietary binary format to the community XML format mzML[33] increased the file size to almost five times, from ~1.6 to ~7.4 Gb for each sample. Such massive data files are difficult to handle, especially for projects with a high number of samples. In the future, binary community formats such as mzMLb[34] are expected to reduce such bloating of data. Converting the mzML profile data to the Mascot Generic Format (MGF) centroid format[35] reduced the file size to less than 10%. The validated TPP pepXML files which are used for ProtyQuant were about 80 Mb in size. The protein reports generated by ProtyQuant in TSV format were less than 500 kb in size, which means a significant decrease in file sizes during the proteomics data processing.

Altogether, ProtyQuant demonstrated its functionality for processing shotgun proteomics data on different platforms, using TPP pepXML files and a UniProt-derived FASTA database.

Protein inference algorithms

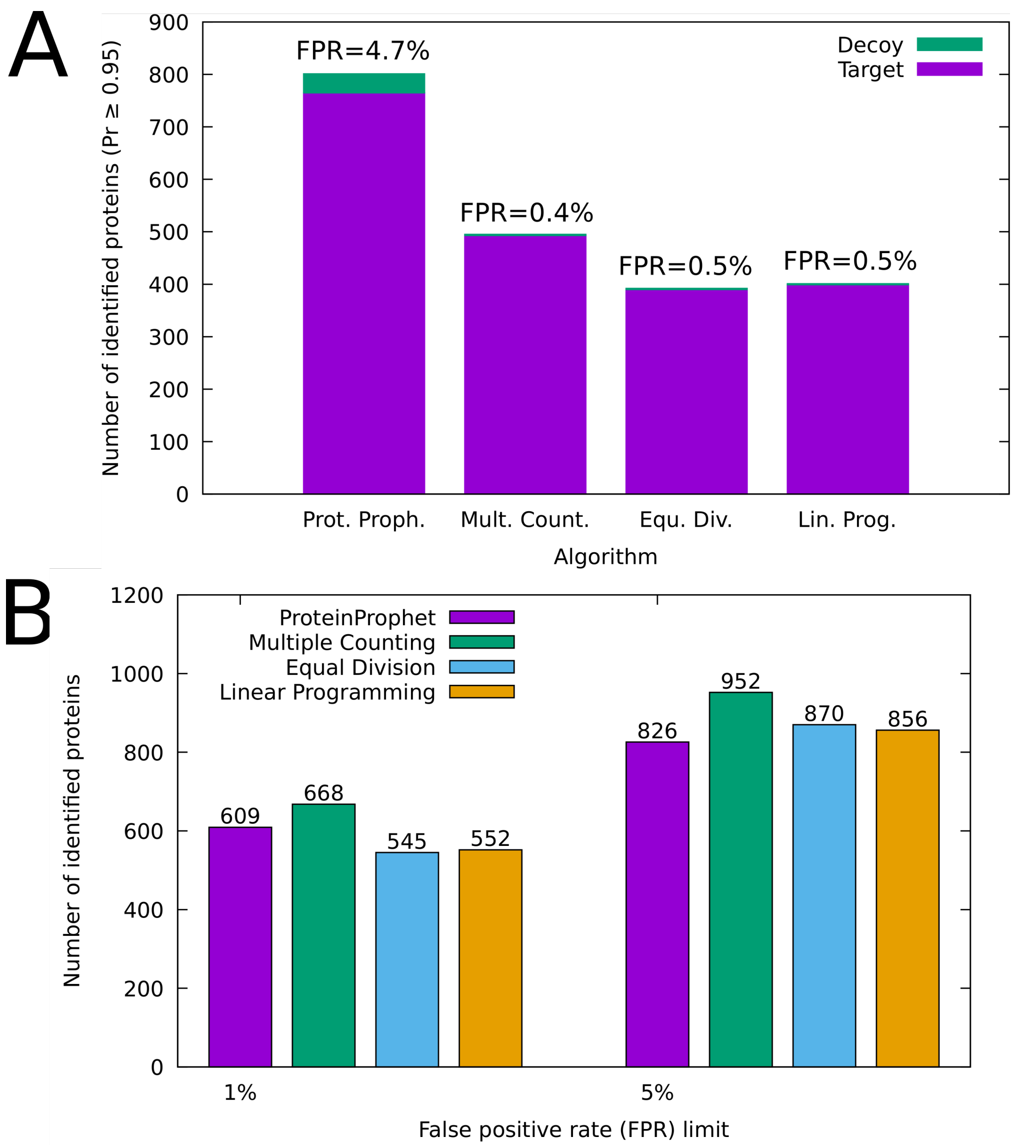


Figure 6: A) Average number of identified proteins with the different protein inference algorithms, using a using a probability (Pr) threshold Pr $\geq$ 0.95. B) Median of identified proteins with a defined false positive rate (FPR).

Table 2: Identified target and decoy hits of the different algorithms using a probability (Pr) threshold of Pr $\geq$ 0.95.

Algorithm	Target hits (min-max, avg.)	Decoy hits (min-max, avg.)	Average FPR [%]
ProteinProphet	694-878, 765	26-50, 36	4.7
Multiple Counting	403-535, 493	0-4, 2	0.4
Equal Division	286-500, 390	0-4, 2	0.5
Linear Programming	315-441, 399	0-4, 2	0.5

For evaluating the PIPQ protein inference algorithms of ProtyQuant, they were benchmarked against the ProteinProphet. Using a probability (Pr) threshold of Pr $\geq$ 0.95. As shown in **Table 2** and **Figure 6A**, the ProteinProphet found the highest number of proteins (765), followed by the Multiple Counting (493), the Linear Programming (399) and the Equal Division (390) algorithm. However, the proportion of false positive hits was about 10 times lower in the ProtyQuant results. Whereas the ProteinProphet results in average had a false positive rate (FDR) of 4.7%, the PIPQ algorithms had a FPR 0.4-0.5%. In average, only two decoy hits passed the Pr filter of ProtyQuant, compared to 36 for ProteinProphet.

For a better comparison of the precision of the algorithms, the number of true positive (TP) hits at different FPR was determined. **Figure 6B** shows the median target identifications at 1% and 5% FPR. The Multiple Counting algorithm performed best at both FPR cut-offs, identifying 59 (9.7%) and 126 (15.3%), respectively, proteins more than the ProteinProphet. The Equal Division and Linear Programming algorithms performed slightly below the ProteinProphet for the 1% FPR limit and somewhat better for the 5% FPR limit.

Besides, the Area Under the Curve (AUC) was used to assess the classifier performance[36]. The ProteinProphet showed AUC values between 0.74 and 0.79, the Multiple Counting algorithm 0.76 - 0.82, the Equal Division algorithm 0.73 - 0.80 and the Linear Programming algorithm 0.71 - 0.78, respectively. **Figure 7** shows the Receiver Operating Characteristic (ROC) curves for sample 2500/R2, with typical results. Overall, the Multiple Counting algorithm was found to be the best

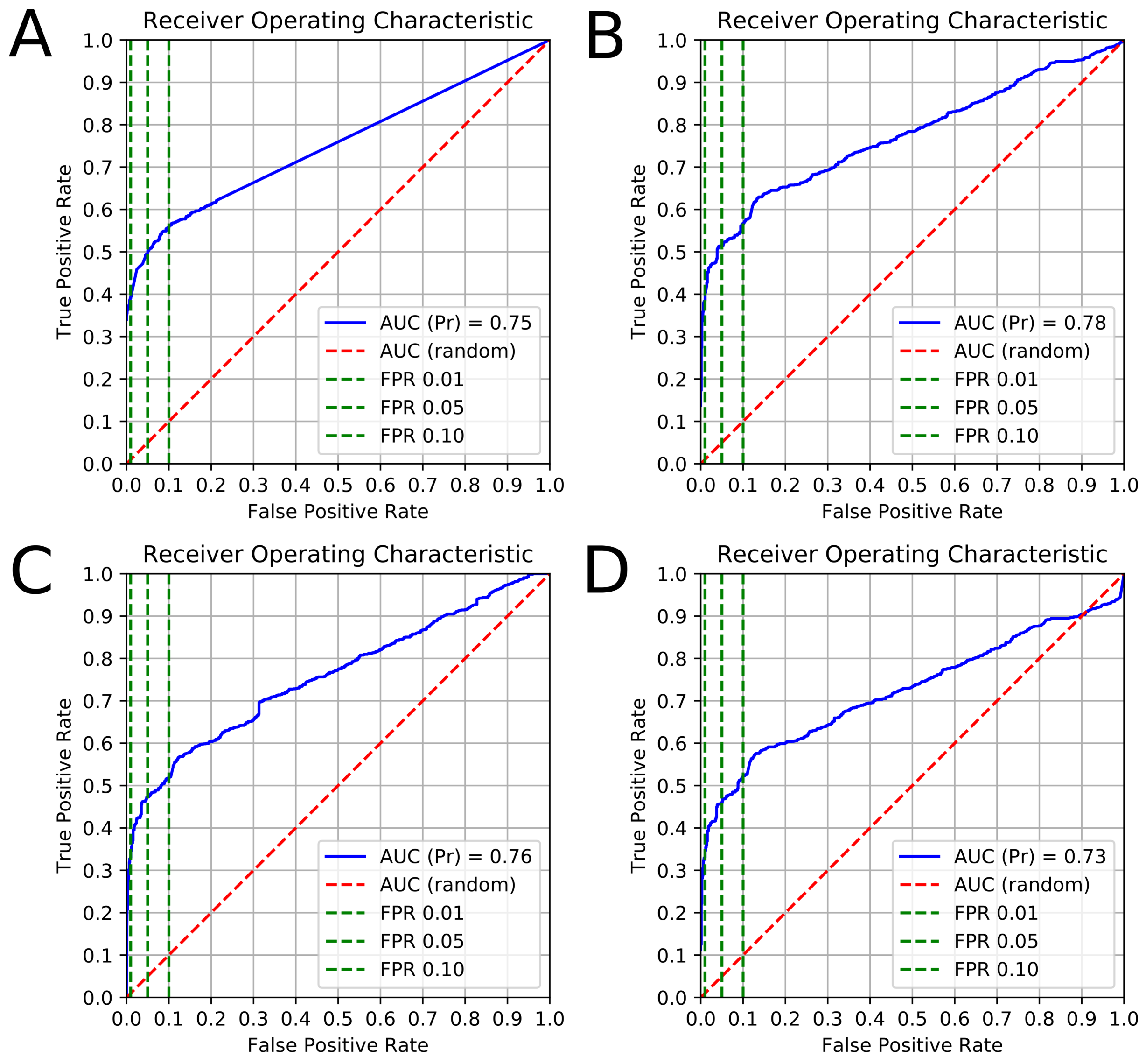


Figure 7: Receiver Operating Characteristics (ROC) of the different protein inference algorithms for sample 2500/R2. without probability cut-off. A) ProteinProphet, B) Multiple Counting, C) Equal Division, D) Linear Programming.

classifier in this comparison, since it demonstrated the highest AUC measures and consistency.

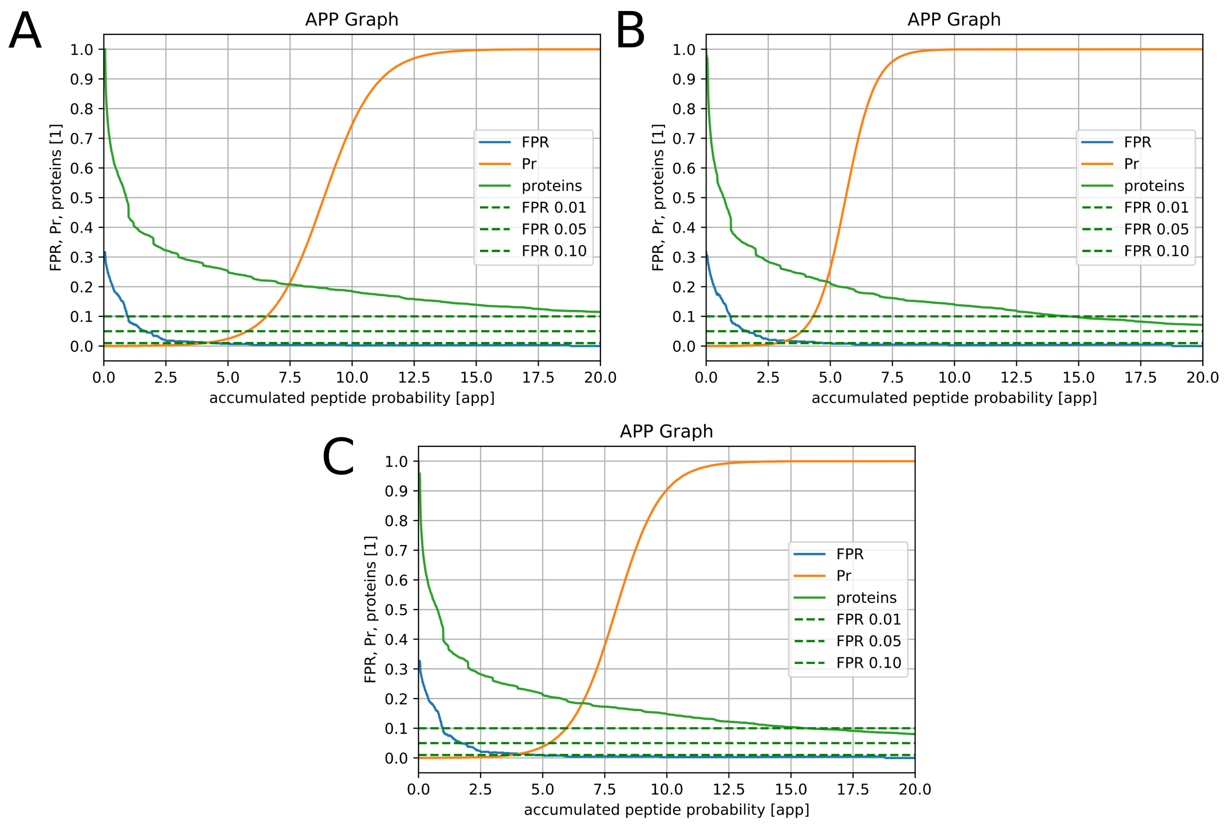


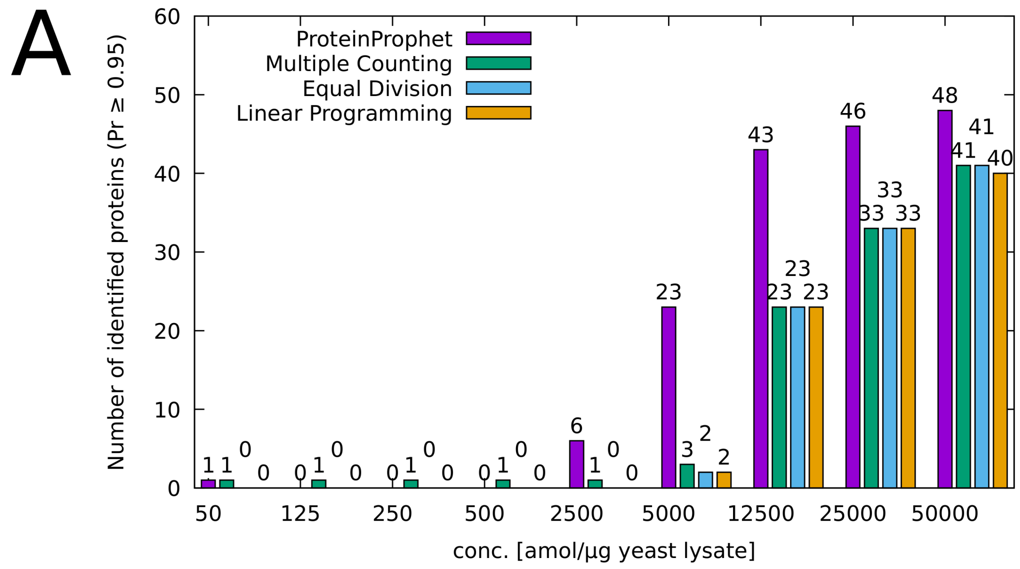
Figure 8: False positive rate (FPR), protein probability (Pr), and proportion of identified proteins as function of the accumulated peptide probability (app). A) Multiple Counting, B) Equal Division, C) Linear Programming.

In **Figure 8**, FPR, Pr, and identified proteins were plotted as a function of the accumulated peptide probability (app). The Multiple Counting algorithm summed more proteins for a given app, followed by the Linear Programming and Equal Division algorithms. The Pr curves were shifted towards higher app in the opposite direction. These curves indicate that the multiple counting of peptides improved the sensitivity; however, the possible rise of false-positive identifications in this strategy was compensated by more strict acceptance criteria.

A comparison with the ProteinProphet demonstrated the suitability of the PIPQ algorithms for protein inference. Although the performance of the algorithms could vary, depending on the analyzed datasets[18,19] and the upstream data processing, similar identification results, and in some cases, even better ones, than the ones obtained with established programs can be expected with

ProtyQuant.

Quantification



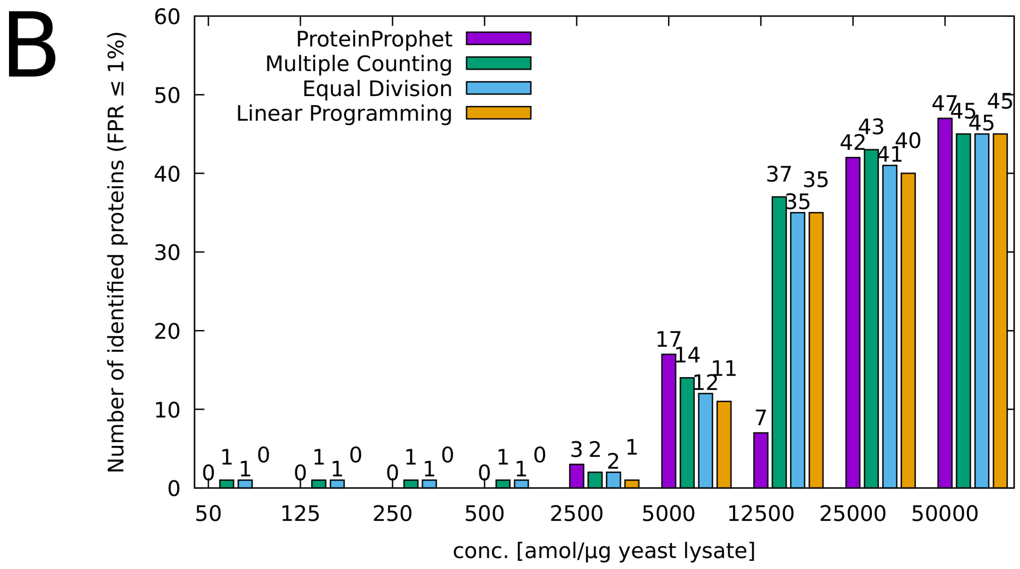


Figure 9: Spiked UPS proteins identified (median) with different algorithms and criteria. A) Probability (Pr) ≥ 0.95, B) false discovery rate (FPR) ≤ 1%.

For evaluating the performance of ProtyQuant for quantification, the identification and linearity of 48 spiked human USP proteins in the yeast lysate background were investigated. **Figure 9** shows the number of detected UPS proteins, using different threshold criteria. Using the probability (Pr) ≥

0.95 cut-off, the ProteinProphet showed by far the highest sensitivity of the four algorithms, which is congruent with the global protein analysis (see **Fig. 9A**). Again, using the false discovery rate (FPR) as criteria, improved the true positive rate (TPR) of the PIPQ algorithms. At an FPR $\leq 1\%$, the four algorithms delivered similar results.

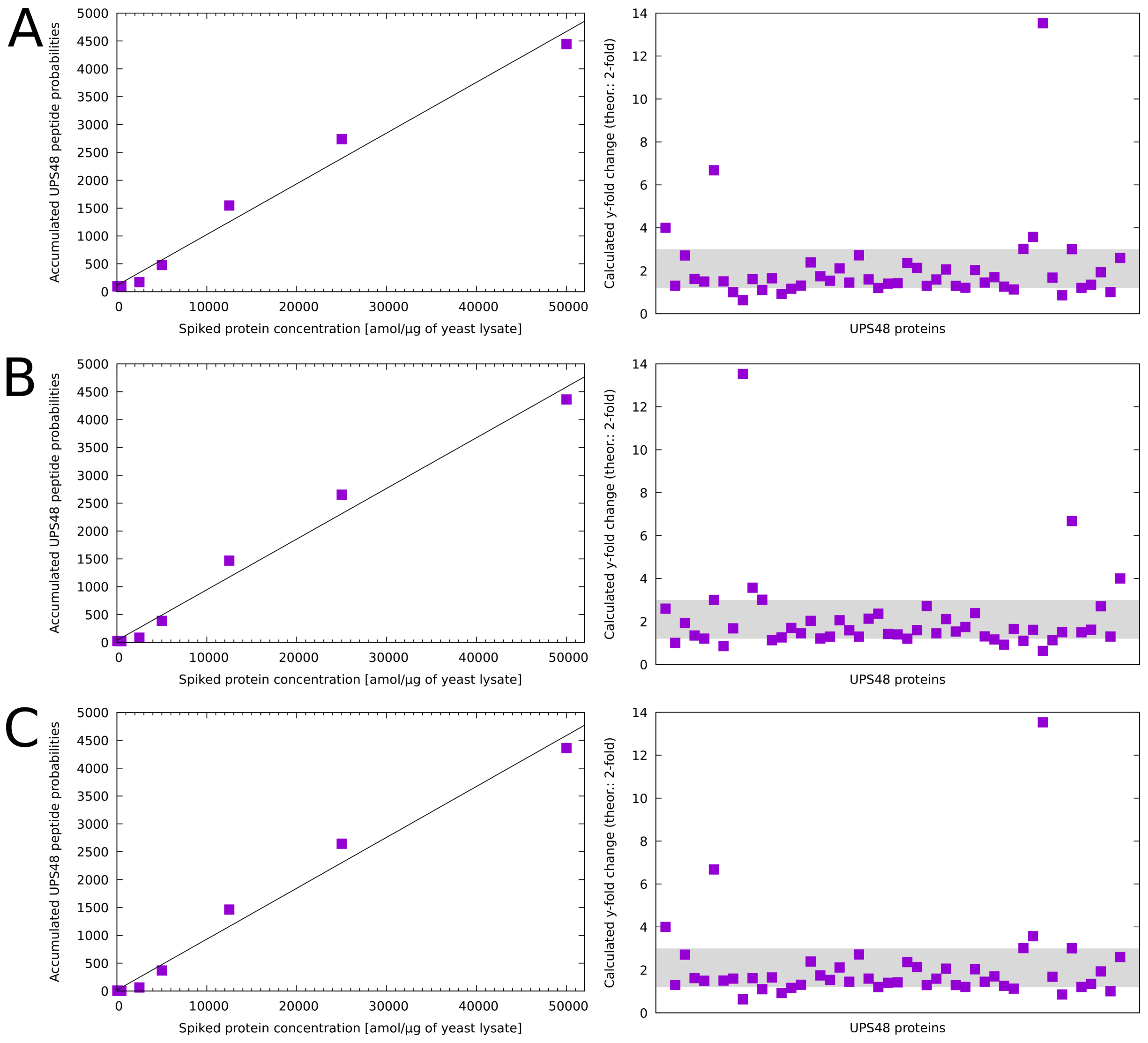


Figure 10: Linearity of the PIPQ algorithms. On the left side, the accumulated peptide probabilities (app) of all UPS proteins are plotted for each spiked protein concentration. On the right side, the determined fold-change of the individual UPS proteins is shown for the two highest spiked protein concentrations. A) Multiple Counting, B) Equal Division, C) Linear Programming.

Figure 10 shows the linearity of the accumulated peptide probabilities (app) of all spiked UPS proteins on the left side, and the calculated fold-change of the two highest concentrations, 50,000 vs. 25,000 amol/µg yeast lysate, on the right side. For all three PIPQ algorithms, satisfactory linearity was found for higher spiked protein concentrations. The average fold-change, which was determined using the Multiple Counting algorithm, based on the app of the two highest spiked protein concentrations, was 2.072, and therefore close to the theoretical factor of 2.0. For 35 of 48 (73%) proteins, a fold-change between 1.2 and 3.0 was found, 8 (17%) proteins were underestimated (< 1.2 fold change), 5 (10%) proteins were overestimated (> 3 fold change). Using the Equal Division, 33 of 48 (69%) proteins were found in the 1.2-3.0 fold range, 9 (19%) underestimated, and 6 (13%) overestimated, with an average of 2.075. With Linear Programming, 35 of 48 (73%) proteins felt into the 1.2-3.0 fold range, 7 (15%) proteins were underestimated, and 6 (13%) proteins overestimated, with an average calculated fold-change of 2.084. Thus, all three PIPQ algorithms demonstrated good quantification capability.

Interestingly, for all 48 proteins, an app was calculated for the two highest concentrations and with all three PIPQ algorithms, although they did not reach the Pr or FPR limit necessary for being considered as confidently identified proteins. The main limitation of the app protein inference strategy is the lack of sensitivity for low-abundance proteins. For studies including such proteins, the combination of multiple search engines[37], or the combination with complementary protein inference algorithms provide possible solutions.

Assessing 'Protein Presence'

As Huang et al. 2012;[18] and He et al. 2016;[19] already stressed out, protein inference and quantification are closely linked. Reporting the presence or, even more complicated, the absence of a protein in a given sample without a quantitative measure is not meaningful. The conventional approach, to first identify proteins, and then to quantify the proteins in a second step, leaves aside the real contribution of a protein to the composition of a mixture. Since validated peptides passing a pre-defined threshold are considered as 'true,' the underlying non-binary probability information is lost in the protein inference and spectral-counting based quantification.

In contrast, the algorithms used in this study take into account the probability of individual peptides for both protein inference and quantification. Therefore, the accumulated peptide probability (app) presents a measure for an integrated assessment of '*Protein Presence.'* Since the app values are quantitative and continuous, uncertain peptide hits have less impact on the final results. Using the app also facilitates the rational fine-tuning of score thresholds, because detection and quantification limits, as well as background levels, can be experimentally determined by sample loading, spiking, and carry-over experiments.

Conclusions

ProtyQuant performs the quantitative comparison of label-free proteomics datasets from validated pepXML files. The software has a human-friendly interface and integrates well with upstream proteomics workflows and downstream data analyses.

The PIPQ program, which performs protein inference and quantifications, sums up the peptide probabilities for each candidate protein. This strategy preserves information about peptide-spectrum match (PSM) hit qualities. For assembling possible proteins and estimating their probability (Pr), the three different PIPQ protein inference algorithms were tested. The probability (Pr) estimation of the PIPQ program is based on the accumulated peptide probabilities (app) and was found to be more conservative than the ProteinProphet algorithm. For equal false-positive rates (FPR), the Multiple Counting approach showed the highest sensitivity and identified the highest number of true-positive proteins.

Calculating the app represents a more precise spectral-counting method than summing up peptides with equal weight. The app values are continuous and were shown in this study to be proportional to the spiked protein concentration, which makes them suitable for label-free protein quantification. The risk of overestimating the protein quantity because of low-significance peptides is reduced when using app values instead of discrete spectral-counting.

This study contributes to the understanding that protein probability and quantity are intimately connected and should be seen as an entity. The app is easy to calculate and could serve as an integral measure of ‘Protein Presence.’

Acknowledgments

I thank Prof. Zengyou He for discussion about his algorithm and Dr. Ignacio Raúl Rosas Román for compiling the Microsoft Windows versions of the PIPQ C++ program. The project was supported by the CONACyT Fronteras project 2015-2/814 and the bilateral grant CONACyT-DFG 2016/277850.

References

(1) Park, C. Y.; Klammer, A. A.; Käll, L.; MacCoss, M. J.; Noble, W. S. Rapid and accurate peptide identification from tandem mass spectra. *J. Proteome Res.* **2008**, *7*, 3022–3027.

(2) McIlwain, S.; Tamura, K.; Kertesz-Farkas, A.; Grant, C. E.; Diament, B.; Frewen, B.; Howbert, J. J.; Hoopmann, M. R.; Käll, L.; Eng, J. K.; MacCoss, M. J.; Noble, W. S. Crux: rapid open source protein tandem mass spectrometry analysis. *J. Proteome Res.* **2014**, *13*, 4488–4491.

(3) Aiche, S.; Sachsenberg, T.; Kenar, E.; Walzer, M.; Wiswedel, B.; Kristl, T.; Boyles, M.; Duschl, A.; Huber, C. G.; Berthold, M. R.; Reinert, K.; Kohlbacher, O. Workflows for automated downstream data analysis and visualization in large-scale computational mass spectrometry. *PROTEOMICS* **2015**, *15*, 1443–1447.

(4) Keller, A.; Eng, J.; Zhang, N.; Li, X.-j.; Aebersold, R. A uniform proteomics MS/MS analysis platform utilizing open XML file formats. *Mol Syst Biol* **2005**, *1*, 2005.0017.

(5) Deutsch, E. W.; Mendoza, L.; Shteynberg, D.; Farrah, T.; Lam, H.; Tasman, N.; Sun, Z.; Nilsson, E.; Pratt, B.; Prazen, B.; Eng, J. K.; Martin, D. B.; Nesvizhskii, A. I.; Aebersold, R. A guided tour of the Trans-Proteomic Pipeline. *Proteomics* **2010**, *10*, 1150–1159.

(6) Deutsch, E. W.; Mendoza, L.; Shteynberg, D.; Slagel, J.; Sun, Z.; Moritz, R. L. Trans-Proteomic Pipeline, a standardized data processing pipeline for large-scale reproducible proteomics informatics. *Proteomics Clin Appl* **2015**, *9*, 745–754.

(7) Nesvizhskii, A. I.; Aebersold, R. Interpretation of Shotgun Proteomic Data: The Protein Inference Problem. *Molecular & Cellular Proteomics* **2005**, *4*, 1419–1440, Publisher: American Society for Biochemistry and Molecular Biology Section: Tutorial.

(8) Huang, T.; Wang, J.; Yu, W.; He, Z. Protein inference: a review. *Brief Bioinform* **2012**, *13*, 586–614, Publisher: Oxford Academic.

(9) 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. *Journal of Proteomics* **2017**, *150*, 170–182.

(10) Keller, A.; Nesvizhskii, A. I.; Kolker, E.; Aebersold, R. Empirical statistical model to estimate the accuracy of peptide identifications made by MS/MS and database search. *Anal. Chem.* **2002**, *74*, 5383–5392.

(11) Nesvizhskii, A. I.; Keller, A.; Kolker, E.; Aebersold, R. A statistical model for identifying proteins by tandem mass spectrometry. *Anal. Chem.* **2003**, *75*, 4646–4658.

(12) Elias, J. E.; Gygi, S. P. Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. *Nat. Methods* **2007**, *4*, 207–214.

(13) Gupta, N.; Bandeira, N.; Keich, U.; Pevzner, P. A. Target-Decoy Approach and False Discovery Rate: When Things May Go Wrong. *J Am Soc Mass Spectrom* **2011**, *22*, 1111–1120.

(14) Blein-Nicolas, M.; Zivy, M. Thousand and one ways to quantify and compare protein abundances in label-free bottom-up proteomics. *Biochim. Biophys. Acta* **2016**, *1864*, 883–895.

(15) McIlwain, S.; Mathews, M.; Bereman, M. S.; Rubel, E. W.; MacCoss, M. J.; Noble, W. S. Estimating relative abundances of proteins from shotgun proteomics data. *BMC Bioinformatics* **2012**, *13*, 308.

(16) Weisser, H.; Nahnsen, S.; Grossmann, J.; Nilse, L.; Quandt, A.; Brauer, H.; Sturm, M.; Kenar, E.; Kohlbacher, O.; Aebersold, R.; Malmström, L. An automated pipeline for high-throughput label-free quantitative proteomics. *J. Proteome Res.* **2013**, *12*, 1628–1644.

(17) Hoopmann, M. R.; Winget, J. M.; Mendoza, L.; Moritz, R. L. StPeter: Seamless Label-Free Quantification with the Trans-Proteomic Pipeline. 2018; `https://pubs.acs.org/doi/abs/10.1021/acs.jproteome.7b00786`.

(18) Huang, T.; Zhu, P.; He, Z. Protein Inference and Protein Quantification: Two Sides of the Same Coin. *arXiv:1210.2515 [cs, q-bio]* **2012**, arXiv: 1210.2515.

(19) He, Z.; Huang, T.; Liu, X.; Zhu, P.; Teng, B.; Deng, S. Protein inference: A protein quantification perspective. *Computational Biology and Chemistry* **2016**, *63*, 21–29.

(20) Huang, T.; He, Z. A linear programming model for protein inference problem in shotgun proteomics. *Bioinformatics* **2012**, *28*, 2956–2962, Publisher: Oxford Academic.

(21) Gao, J.; Tan, P. n. Converting Output Scores from Outlier Detection Algorithms into Probability Estimates. Sixth International Conference on Data Mining (ICDM'06). 2006; pp 212–221.

(22) Ramus, C. et al. Spiked proteomic standard dataset for testing label-free quantitative software and statistical methods. *Data in Brief* **2016**, *6*, 286–294.

(23) Ramus, C. et al. Benchmarking quantitative label-free LC–MS data processing workflows using a complex spiked proteomic standard dataset. *Journal of Proteomics* **2016**, *132*, 51–62.

(24) Vizcaíno, J. A. et al. ProteomeXchange provides globally coordinated proteomics data submission and dissemination. *Nat. Biotechnol.* **2014**, *32*, 223–226.

(25) Kessner, D.; Chambers, M.; Burke, R.; Agus, D.; Mallick, P. ProteoWizard: Open source software for rapid proteomics tools development. *Bioinformatics* **2008**, *24*, 2534–2536.

(26) Deutsch, E. W.; Mendoza, L.; Shteynberg, D. D.; Sun, Z.; Hoopmann, M. H.; Moritz, R. L. *Processing Metabolomics and Proteomics Data with Open Software*; 2020; pp 333–344.

(27) Eng, J. K.; Hoopmann, M. R.; Jahan, T. A.; Egertson, J. D.; Noble, W. S.; MacCoss, M. J. A Deeper Look into Comet—Implementation and Features. *J. Am. Soc. Mass Spectrom.* **2015**, *26*, 1865–1874.

(28) Perkins, D. N.; Pappin, D. J. C.; Creasy, D. M.; Cottrell, J. S. Probability-based protein identification by searching sequence databases using mass spectrometry data. *ELECTROPHORESIS* **1999**, *20*, 3551–3567, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291522-2683%2819991201%2920%3A18%3C3551%3A%3AAID-ELPS3551%3E3.0.CO%3B2-2.

(29) Kim, S.; Pevzner, P. A. MS-GF+ makes progress towards a universal database search tool for proteomics. *Nat Commun* **2014**, *5*, 5277.

(30) Tabb, D. L.; Fernando, C. G.; Chambers, M. C. MyriMatch: highly accurate tandem mass spectral peptide identification by multivariate hypergeometric analysis. *J Proteome Res* **2007**, *6*, 654–661.

(31) Craig, R.; Beavis, R. C. TANDEM: matching proteins with tandem mass spectra. *Bioinformatics* **2004**, *20*, 1466–1467.

(32) Winkler, R., Ed. *Processing Metabolomics and Proteomics Data with Open Software: A Practical Guide*, 1st ed.; New Developments in Mass Spectrometry 8; Royal Society of Chemistry: Cambridge, UK, 2020.

(33) Martens, L. et al. mzML - a community standard for mass spectrometry data. *Mol Cell Proteomics* **2011**, *10*, R110.000133.

(34) Bhamber, R. S.; Jankevics, A.; Deutsch, E. W.; Jones, A. R.; Dowsey, A. W. mzMLb: a future-proof raw mass spectrometry data format based on standards-compliant mzML and optimized for speed and storage requirements. *bioRxiv* **2020**, 2020.02.13.947218, Publisher: Cold Spring Harbor Laboratory Section: New Results.

(35) Deutsch, E. W. File Formats Commonly Used in Mass Spectrometry Proteomics. *Mol Cell Proteomics* **2012**, *11*, 1612–1621.

(36) Fawcett, T. An introduction to ROC analysis. *Pattern Recognition Letters* **2006**, *27*, 861–874.

(37) Shteynberg, D.; Nesvizhskii, A. I.; Moritz, R. L.; Deutsch, E. W. Combining Results of Multiple Search Engines in Proteomics. *Mol Cell Proteomics* **2013**, *12*, 2383–2393.