nature protocols

https://doi.org/10.1038/s41596-024-01000-3

Protocol

Check for updates

An MSstats workflow for detecting differentially abundant proteins in large-scale data-independent acquisition mass spectrometry experiments with FragPipe processing

Devon Kohler [1,2], Mateusz Staniak [3], Fengchao Yu [4], Alexey I. Nesvizhskii [4,5] & Olga Vitek [1,2]

Abstract

Technological advances in mass spectrometry and proteomics have made it possible to perform larger-scale and more-complex experiments. The volume and complexity of the resulting data create major challenges for downstream analysis. In particular, next-generation data-independent acquisition (DIA) experiments enable wider proteome coverage than more traditional targeted approaches but require computational workflows that can manage much larger datasets and identify peptide sequences from complex and overlapping spectral features. Data-processing tools such as FragPipe, DIA-NN and Spectronaut have undergone substantial improvements to process spectral features in a reasonable time. Statistical analysis tools are needed to draw meaningful comparisons between experimental samples, but these tools were also originally designed with smaller datasets in mind. This protocol describes an updated version of MSstats that has been adapted to be compatible with large-scale DIA experiments. A very large DIA experiment, processed with FragPipe, is used as an example to demonstrate different MSstats workflows. The choice of workflow depends on the user's computational resources. For datasets that are too large to fit into a standard computer's memory, we demonstrate the use of MSstatsBig, a companion R package to MSstats. The protocol also highlights key decisions that have a major effect on both the results and the processing time of the analysis. The MSstats processing can be expected to take 1–3 h depending on the usage of MSstatsBig. The protocol can be run in the point-and-click graphical user interface MSstatsShiny or implemented with minimal coding expertise in R.

Key points

- Technological advances in bottom-up mass spectrometry-based proteomics have resulted in a substantial increase in the volume and complexity of the resulting data, and for comparative studies, large numbers of samples are required to get statistically meaningful results.
- MSstats can be used to perform statistical analysis of the data after the peptides and proteins have been identified and quantified. MSstatsBig is a variant specifically designed to manage very large datasets.

Key references

Kohler, D. et al. *J. Proteome Res.* **22**, 1466–1482 (2023)

Kohler, D. et al. *J. Proteome Res.* **22**, 551–556 (2023)

Kong, A. et al. *Nat. Methods* **14**, 513–520 (2017)

Yu, F. et al. *Nat. Commun.* **14**, 4154 (2023)

Clark, D. J. et al. *Cell* **179**, 964–983 (2019)

[1]Khoury College of Computer Sciences, Northeastern University, Boston, MA, USA. [2]Barnett Institute for Chemical and Biological Analysis, Northeastern University, Boston, MA, USA. [3]University of Wrocław, Wrocław, Poland. [4]Department of Pathology, University of Michigan, Ann Arbor, MI, USA. [5]Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI, USA. e-mail: o.vitek@northeastern.edu

Introduction

Bottom-up mass spectrometry (MS)-based proteomics is the tool of choice for researchers interested in detecting differential changes in protein abundance between conditions[1–3]. It is used to answer a wide range of interesting biological questions including questions that involve determining changes in the proteome with time or differences between the proteomes of different experimental samples. A common example is measuring overall protein versus post-translational modifications (PTMs).

The number of possible research goals is one level of complexity; the choice of method is another. Methods can differ in terms of labeling type (i.e., label-free (LF) versus tandem mass tag (TMT) labeling) or the method used to acquire the data (i.e., data-dependent acquisition (DDA), selected reaction monitoring (SRM) or data-independent acquisition (DIA))[4–6].

More recently, MS-based proteomic experiments have been increasing in size and scale. This is in part due to modern workflows, which are robust and have high sample throughput, increasing the number of samples that can be measured at a reasonable monetary and time cost while also increasing proteome coverage[7,8]. In addition, researchers now appreciate the fact that most biological questions can be answered only with an appropriately large number of biological replicates, making higher throughput workflows a requirement for many studies[9].

Software for spectral feature analysis

Analyses of bottom-up MS-based proteomic experiments require software, such as FragPipe, DIA-NN, Spectronaut, Skyline and MaxQuant (among others), that determines the peptide sequence identity of the spectral features and quantifies their relative abundance[10–14]. Feature analysis can be particularly challenging in DIA acquisitions in which the spectra peaks from different peptides and fragment ions are convoluted[15,16]. Recent advances in machine learning approaches, such as those in DIA-NN, have allowed researches to deconvolute these peaks and consequently enabled deeper proteome coverage, making DIA a very useful tool for researchers interested in exploratory studies[11]. With these new DIA approaches, researchers can identify and quantify many more proteins and features per protein (e.g., peptide ions for DDA, peptide transitions for SRM and peptide ions and fragment ions for DIA) than were possible with older acquisition methods. FragPipe, used in this paper, expands the fast database searching in MSFragger and, in combination with DIA-NN's quantification module, enables accurate and efficient processing of DIA data[17,18]. Together with the increasingly large sample sizes mentioned previously, these new DIA methods have resulted in increasingly large datasets, after identification and quantification.

The importance of statistical methods

Statistical analysis is performed on the resulting datasets to detect differentially abundant proteins or proteoforms. It is therefore important that advances in statistical methods and their associated software match the pace of any advances in technology, experimental methodology or computational methods for data analysis.

Even with the advent of novel machine learning approaches for DIA experiments, general analytical and statistical challenges for detecting differentially abundant proteins remain. These challenges stem from both biological and technical aspects of the experiment[19,20]. From the biological perspective, statistical methods must handle different experimental designs, including group comparison and repeated measure designs that have different structures of variation[21]. Technological challenges include appropriately aggregating feature intensities into protein-level abundances and accounting for missing values, outliers and batch effects over subsets of runs.

As the sample throughput and ability to identify and quantify additional features increases, the technological challenges are further exacerbated, and new challenges are introduced. This includes the presence of an excessive number of features per protein, many of which are noisy (e.g., with high variance, many outliers and many missing values)[22]. Therefore, large-scale experiments require statistical methods that are not only accurate but are also computationally

efficient and numerically stable. Beyond this, large-scale experiments make accurate statistical modeling unreachable for experimentalists with limited computational infrastructure and coding ability. Therefore, there is a need for tighter integration of statistical methods with data-processing tools, development and computational optimization of statistical models and graphical user interfaces (GUI), which remove the need for users to write code. Simple methods, such as *t* test and ANOVA, do not address these challenges, and more advanced statistical methods are required[23].

Software for statistical analysis

Various software tools for statistical analyses that detect differentially abundant proteins have been developed to address these challenges. These include MSstats[24,25], msqrob2[26–28], DEqMS[29] and prolfqua[30], among others. These tools have recently been thoroughly compared and benchmarked by Bai et al., who compared functionalities in each package such as normalization and imputation and assessed the tools' performance on three benchmark datasets with known ground truth[31]. MSstats, in particular, has been shown to be broadly applicable and perform well compared to other methods for detecting differentially abundant proteins[24,31].

Advantages and limitations of MSstats

As with any tool, MSstats has both advantages and limitations associated with its methods. In terms of advantages, MSstats is immediately applicable (i.e., without any adjustment) to a wide range of experimental designs, including different acquisition methods, such as DDA, DIA and SRM/parallel reaction monitoring (PRM), as well as group comparison and repeated measure designs, such as time series and paired designs. Beyond this, the newly released MSstatsShiny GUI opens the methods to a broader user base, including those without the ability to write code. This contributes to a broad adoption of MSstats for practical applications.

Many statistical analysis tools implement a wider range of analytical methods than MSstats. For example, msqrob2 recommends in their vignette using the QFeatures[32] package, which in turn provides users with 14 different algorithms for missing value imputation. Although this variety of options enables flexibility, it can become a double-edged sword for users without a strong statistical background, leading to arbitrary analysis choices, overfitting and inflation of false-positive results[33]. By contrast, MSstats was designed with an eye toward users without a background in statistics and includes a limited number of well-documented functionalities that cover most use cases, as opposed to many different choices.

Because MSstats was created before the advances in DIA[25], the package was designed with smaller proteomic experiments in mind. However, as experiments have become increasingly large both in number of samples and number of proteins, the original implementation was limited in analysis speed and computational resource usage. Recent improvements in MSstats and the new MSstatsBig package attempt to resolve these speed and resource limitations.

Overview of the procedure

Here, we present two MSstats workflows with a particular focus on large-scale DIA experiments processed with FragPipe. The workflows address different users' needs and include using the GUI MSstatsShiny, as well as the code version of MSstats[34]. We show how a user can leverage new developments in the MSstatsBig R package, which allow users to analyze large modern DIA experiments that are too large to fit into a standard computer's memory. We highlight important choices in the analysis of large DIA experiments, such as feature selection and missing value imputation, as they relate to processing time and computational resources and to the accuracy and reproducibility of the statistical conclusions drawn from the analysis. We anticipate that users will have different needs, and the provided workflows cover various circumstances.

Figure 1 summarizes the procedures described in this protocol. FragPipe is the first step for all procedures. After FragPipe, the user has two options: Procedure 1 or Procedure 2.

- Procedure 1: analysis using MSstatsShiny performed entirely in the GUI. The main limitation of Procedure 1 is that it does not include any functionality for filtering down very large datasets and does not include the flexibility to perform analyses that are outside of MSstats

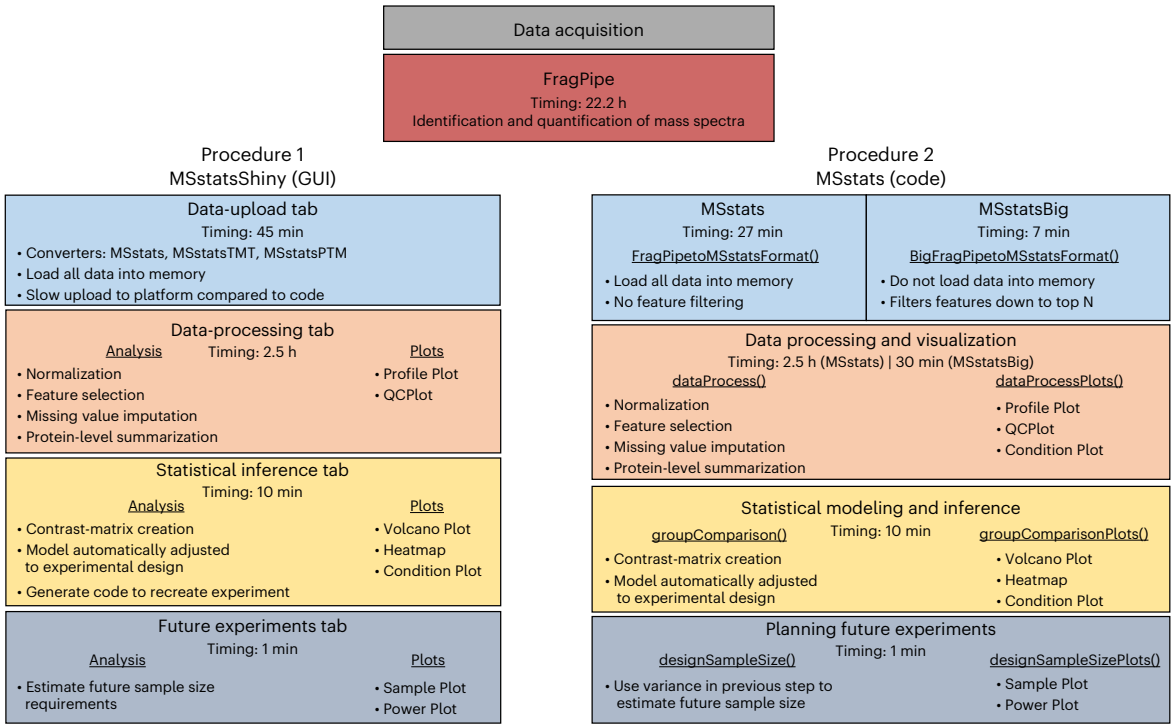

Fig. 1 | Overview of the protocol workflow. The pipeline starts with data acquisition and identification and quantification with FragPipe. Then, there are two options: MSstatsShiny (GUI) or MSstats (code). The MSstatsShiny GUI includes four tabs: Data Upload, Data Processing, Statistical Inference and Future Experiments. The MSstats code includes two options for converters: MSstats converter or MSstatsBig converter. From there, the workflow uses the same functions: `dataProcess()`, `groupComparison()` and `designSampleSize()`.

capabilities. Users who follow this procedure must have enough computational resources available to process the data (≥256 GB of random access memory (RAM)).

- Procedure 2: analysis performed entirely in R code. The user needs to convert the FragPipe data into a format suitable for MSstats. Here, there are two options for converters: the base MSstats converter and the MSstatsBig converter. As with Procedure 1, users who use the classic MSstats code must have sufficient computational resources. MSstatsBig code can be processed with less computational resources (only requiring 16 GB of RAM).

FragPipe

FragPipe is a GUI that includes various analytic tools for the analysis of MS-based proteomics. In brief, FragPipe uses MSFragger[10] to perform database searching, MSBooster[35] to calculate deep-learning-based scores, Percolator to rescore peptide-spectrum matches (PSMs), ProteinProphet to perform protein inference, Philosopher[36] to calculate false discovery rate (FDR), EasyPQP to build a spectral library by using Philosopher's output and DIA-NN to perform quantification by using the spectral library. Direct identification of peptides from DIA data in FragPipe has been previously possible by using DIA-Umpire followed by a conventional MSFragger search. With MSFragger's recent enhancement, the MSFragger-DIA[18] module now provides an alternative, faster and more-sensitive workflow for direct DIA analysis. Regardless of the choice (DIA-Umpire or MSFragger-DIA based), FragPipe processes the DIA data, constructing a spectral library from the searched and FDR-filtered peptides. The DIA-NN quant module (available as part of FragPipe) then uses this library to extract peptide signals and perform quantification. After DIA-NN concludes, FragPipe converts its output to the msstats.csv format for subsequent analysis. During the format conversion, FragPipe propagates the experiment group information (metadata) to the msstats.csv file.

MSstats

MSstats is a suite of R packages and an individual package with the same name, designed for statistical analysis of MS-based proteomic experiments. The individual MSstats R package is specifically built for the analysis of LF techniques that can be acquired with DDA, DIA, SRM or PRM[24]. The package is at the core of all other MSstats packages, including MSstatsPTM[37], MSstatsTMT[38], MSstatsLiP[39] and MSstatsShiny[34].

MSstats' workflow is broadly separated into four main components:

1. Tool-specific conversion
2. Data pre-processing and summarization
3. Group comparison and statistical modeling
4. Sample size calculation for future experiments

The tool-specific converters take the output of various spectral processing tools, such as FragPipe, DIA-NN and Skyline, and perform core filtering and conversion into the format required by MSstats. A full list of the available converters can be seen in Table 1.

The data pre-processing and summarization step performs crucial data processing, such as normalization and missing value imputation, as well as summarizing features to a single value per protein per run. This step also includes feature selection, such as using all features or only selecting the 'Top-N' features.

Table 1 | MSstats converter availability

Level of comparison	Labeling type	Dedicated converter
Protein	LF	FragPipe[a]
		DIA-NN
		Skyline
		Spectronaut[a]
		MaxQuant
		Proteome Discoverer
		DIA-Umpire
		OpenMS
		OpenSWATH
		Progenesis
		MetaMorpheus
	TMT	FragPipe
		MaxQuant
		OpenMS
		Proteome Discoverer
		SpectroMine
PTMs	LF	FragPipe
		MaxQuant
		Proteome Discoverer
		Progenesis
		Skyline
		Peak Studio
		Spectronaut
	TMT	FragPipe
		MaxQuant
		Proteome Discoverer
LiP	LF	Spectronaut
		Skyline

[a]Converter is available in both MSstats and MsstatsBig.

Group comparison takes the summarized values and fits a specific statistical model depending on the experimental design, returning the statistical results including fold changes and *P* values.

Finally, sample size calculation looks at the average variance seen in the previous step and calculates how many samples would be needed for a future experiment with the same variance.

MSstats was originally designed for DDA, SRM and PRM experiments, which were small enough to fit in an average computer's memory and could be processed in a reasonable amount of time. With the advent of modern DIA technology, the classic MSstats workflow runs into new computational challenges. First, the PSM files output by spectral processing tools can be too large to even load into memory at all, meaning there is no way to even start the MSstats pipeline. Second, with the large number of features in DIA experiments, the data pre-processing steps, such as feature summarization and missing value imputation, take a very long time to run and use up even more of a computer's memory. Within the classic MSstats framework, the only solution to these challenges is to access more computational resources, in particular memory.

MSstatsShiny

The MSstats workflow was originally designed for users to code in R directly on the command line. The R analysis provides flexibility to the user, allowing the user to go outside the built-in MSstats functionalities and implement complementary analyses. However, although flexible, the R version of MSstats requires at least a basic understanding of writing R code and is not accessible to all users. MSstatsShiny was created to open the MSstats functionalities to a wider user base.

MSstatsShiny is a GUI that includes the core functionality of MSstats including converters, data-processing options, plotting functions and statistical inference. In addition, MSstatsShiny includes the functionality to track a user's analysis and produce an R script that exactly reproduces the analysis that was run. This script supports reproducibility, allowing future researchers to know exactly what version of the software was used and which parameters were chosen at each step. Beyond this, the script can be extended to include analyses that are outside of the MSstats ecosystem. For example, the script could be easily extended to perform principal component analysis, which is not currently available within MSstats itself.

MSstatsBig

Although the MSstats R package and the MSstatsShiny GUI are broadly applicable to various experimental designs, they were not originally intended for large data files that come with modern DIA workflows (in which PSM files exceed 10 GB). These files generally include a large number of MS runs and many proteins and features per protein compared to the average DDA, SRM or PRM experiment. In base MSstats, simply loading these datasets into memory can cause R to crash. MSstatsBig avoids loading the data into memory and executes pre-processing steps in a way that reduces the size of the file without affecting the statistical analysis. These methods are packed in novel converters for FragPipe and Spectronaut that are designed to replace the standard MSstats converters and leverage the R package arrow to process and filter the dataset without loading it into memory[40]. After pre-processing, the dataset can then be input into the remaining steps of the MSstats workflow.

One main aspect of MSstatsBig with DIA datasets is to filter down the number of features that are used to infer the underlying protein abundance in a sample. Generally, feature selection is performed at the data pre-processing and summarization step so that normalization can be performed with all data. However, in DIA datasets, the number of features can become redundantly large and create computational challenges with no added benefit. MSstatsBig pre-filters features down to a default of 20. An in-depth analysis on the effect of MSstatsBig's feature selection and reasoning for why a default value of 20 was chosen can be found in Supplementary Method 1 and Supplementary Figs. 2–8.

MSstatsBig also includes an option to use the Apache Spark database to process an entire dataset without pre-filtering features[41]. However, unlike the arrow backend, this requires

the configuration of several additional software tools. Thus, we restrict our attention to the arrow-based approach.

An in-depth benchmarking comparison of MSstats and MSstatsBig, including processing time and computational memory usage, can be found in Supplementary Method 2.

MSstatsBig is open source, with code available on Github, and installable via Bioconductor. The package is currently available only as R code and is not integrated into the MSstatsShiny GUI. Integration into MSstatsShiny is planned; however, this requires several technical hurdles to overcome, including ensuring that the GUI can correctly access files on a user's local drive without loading them into memory. However, MSstatsBig can be integrated into the R code generated by MSstatsShiny (by generating the code with a subset of the data and replacing the converter with the corresponding MSstatsBig converter).

Adapting the procedures to other applications

The procedures discussed here are extendable to workflows beyond LF DIA experiments, including workflows that are label based (acquired with TMTs), and those interested in biological questions beyond the overall proteome, such as those targeting peptides or PTMs or using limited proteolysis (LiP)[42].

For peptide-centric analyses, the procedures need only minimal adjustment. We highlight what steps to change to perform these analyses.

For other workflows, specifically those using TMT labeling, those targeting PTMs or LiP workflows, users must use the corresponding R packages MSstatsTMT[38], MSstatsPTM[37] or MSstatsLiP[39]. These packages follow the same general workflow discussed here but include different choices and assumptions that are beyond the scope of this protocol. For more information on these specific workflows, we point readers to the corresponding publications and the extensive vignettes included in the packages (available on Bioconductor and Github).

Example experimental data

In this protocol, we show the workflow by using a large DIA experiment by Clark et al.[43], which is described below. This is an example dataset, and any MS-based proteomics experiment acquired with DIA can be processed with this workflow.

Clark et al. investigated deregulation that drove clear cell renal cell carcinoma. The study included proteomic, phosphoproteomic, genomic, epigenomic and transcriptomic measurements of treatment-naive clear cell renal cell carcinoma and normal tissue samples. Details of the settings used in the MS runs can be found in the original publication. One hundred and ten patients were measured in a paired design, with normal adjacent tissue (NAT) and tumor (T) samples being taken from the same replicate. For 26 of the patients, no NAT was collected, and only tumor samples could be analyzed, resulting in an unbalanced design. Covariate information was collected for each patient, including features such as age and gender. In this analysis, age was included as a covariate and was discretized into two categories, young and old. This resulted in four total conditions, NAT and T samples measured at both young and old ages.

Both DDA/TMT and DIA acquisition were performed to measure the global protein. DDA/TMT was also used to measure the PTM levels. In this protocol, we use only the global protein levels measured with DIA.

There are 187 single-injection DIA samples plus 8 fractionated DDA runs from pooled samples. The quantification output was from the 187 DIA samples. FragPipe quantified 8,613 proteins, 145,038 precursors and 146,186,038 fragment ions.

This experiment was a clinical study with human samples, which generally exhibit higher variance than samples collected under control conditions such as model organisms or cell lines. As a result, clinical studies typically require more biological replicates than studies performed under controlled conditions. Therefore, it is important to take the extent of this variance into account when designing experiments.

Materials

Equipment

- Computer: Windows personal computer or Linux with quad-core or greater CPU and ≥256 GB of RAM for the base MSstats and MSstatsShiny workflows and only 16 GB of RAM for the MSstatsBig workflow
- Software: MSstats suite of R packages (MSstats, MSstatsShiny and MSstatsBig); R version 4.3 or higher; FragPipe (version 20.1-build4), MSFragger (version 3.8), Philosopher (version 5.0.0) and Python 3.9 (optional)

Equipment setup

The data

For the workflow outlined in this protocol, any DIA dataset acquired on any platform can be used as input. The dataset used in this protocol is freely available at https://pdc.cancer.gov/pdc/study/PDC000200 and on MassIVE (https://massive.ucsd.edu/ProteoSAFe/dataset.jsp?task=6847574c13964a1f9482ee4d71f33eb1). The processed data at each step and MSstats code used to analyze the data are available in the MassIVE.quant Reanalysis RMSV000000696.1.

R version 4.3 or higher

To install R, follow the instructions provided by the software developers at https://www.r-project.org/. Note that installing and using an earlier version of R will create issues in the MSstats installation.

MSstats suite of R packages

All MSstats R packages are available on Bioconductor. After installing R, users should follow the installation instructions on Bioconductor for MSstats (https://www.bioconductor.org/packages/release/bioc/html/MSstats.html), MSstatsShiny (https://www.bioconductor.org/packages/release/bioc/html/MSstatsShiny.html) and MSstatsBig (https://bioconductor.org/packages/release/bioc/html/MSstatsBig.html). Note that if you install MSstatsShiny, it will automatically install MSstats as a dependency. The installation of R and MSstats can be expected to take ~15 min.

FragPipe

The installation instructions for FragPipe can be found on the developer's website (https://fragpipe.nesvilab.org/). FragPipe requires an installation of Java 9+. If Java is not installed, FragPipe offers a specific download file that includes a Java runtime for Windows only. If you are not on Windows, you will have to install Java separately. To run FragPipe, unzip the downloaded file; in the '/bin' directory, you will find a shell script for Linux, a bat file for Windows and an exe file for Windows. FragPipe is not available for the Mac operating system. After downloading, FragPipe must be configured in the 'Config' tab. Instructions for how to perform the configuration can be found in the FragPipe tutorial (https://fragpipe.nesvilab.org/docs/tutorial_fragpipe.html#configure-fragpipe).

To use spectral library building (required for DIA analysis), as well as database splitting (optional) in FragPipe, Python version 3.9 must be installed. Database splitting reduces the size of the in-memory fragment ion index, which can be helpful when computational memory is limited or for complex searches.

The installation of FragPipe and all of the accompanying tools can be expected to take ~10 min.

Note that the procedure described in this protocol uses pre-released features in FragPipe that are available only in the development version at the time of writing (specifically version 20.1-build4). The source code for this pre-released version can be found on GitHub (https://github.com/Nesvilab/FragPipe/tree/94708da1c73b069f5fa8cbfc5bdaf6ff45a25476). All the pre-released features will be included in the future formal release.

Procedure 1: MSstatsShiny analysis pipeline

Section 1: processing raw data files with FragPipe

● **TIMING 22.2 h**

▲ **CRITICAL** FragPipe uses workflows to load the best default parameters for specific types of experiments. Users are recommended to always start with an existing workflow and then tweak the parameters as needed. Customized workflows can be saved and used in the future or shared with the FragPipe community.

1. Select the 'DIA_SpecLib_Quant' workflow in the workflow tab and click 'Load workflow'.
2. Add raw spectral files and set the datatype in the workflow tab. As you add files, the file name will appear in the table at the bottom of the screen.
   - Add the files by clicking the 'Add files' button and selecting the files or by clicking the 'Add folder recursively' button and selecting the folder that contains the files.
   - Set the data type by clicking the 'Set DIA' button.
3. Annotate the data in the table where your file names appeared to specify the experiment (called 'condition' in MSstats) and biological replicates. The annotation can be saved for future use by clicking the 'Save as manifest' button.
   ▲ **CRITICAL STEP** The information entered into this annotation table will be used by MSstats to determine the design of your experiment and ensure that the correct statistical model is used. Thus, it is very important that this step is done correctly, because incorrect annotation can lead to unreliable statistical results. See Box 1 for more information on how to set up the annotation information correctly for MSstats.
4. Navigate to the 'Quant (DIA)' tab and check the 'Generate MSstats input' box in the middle of the page. This will ensure that the 'msstats.csv' file is output.
5. Navigate to the 'Run' tab. Click the 'Browse' button and select the folder where you would like to save the FragPipe results. Click the 'Run' button to execute the FragPipe analysis.
6. Confirm that the 'msstats.csv' file was output in the folder that you selected in the previous step.
   ▲ **CRITICAL STEP** FragPipe must output this file before you can move onto the MSstats section of this protocol.

Section 2: data uploading

● **TIMING 45 min**

▲ **CRITICAL** The first step of the MSstatsShiny analysis pipeline is to convert the output of the spectral processing tool, in this case FragPipe, into MSstats format. There are various dedicated converters included in MSstats for different tools depending on the biological question of interest, the acquisition strategy and the labeling type. The full converter list can be seen in Table 1.

▲ **CRITICAL** On the MSstatsShiny home page, under 'Please select from the following options to get started', there is an orange button labeled 'Help!'. Clicking this button will take the user to the MSstats Google Group forum, which is an active community in which users can ask questions directly to the developers of MSstats.

7. Launch MSstatsShiny from the R console by executing `MSstatsShiny::launch_MSstatsShiny()`. The application will launch in a new window if executed from Rstudio or in your default web browser if executed in the R console.
8. Click on the orange 'Run MSstats Pipeline' button, which will take you to the 'Data Uploading' tab of the application.
9. In the gray box on the left-hand side of the page, select 'Protein' under 'Biological Question', 'Label-free' under 'Label Type' and 'FragPipe' under 'Type of File'. New options will then appear at the bottom of the table.
   ▲ **CRITICAL STEP** MSstatsShiny will use the options here to import the data and determine the type of experiment. It is very important that the correct options are selected, because incorrect options could result in an error.

BOX 1

Creating the annotation file

MSstats uses a user-provided annotation file to determine the experimental design and ensure that the correct statistical model is fit to the data. This is done by mapping the run name (generally in the form of a file name or other string) to the condition and biological replicate. The biological replicate in particular needs to be input with a unique identifier so that MSstats can determine whether the experiment is a group comparison or repeated measurement design (such as a time series or paired design). Below is a subset of the annotation file for the DIA experiment included in this protocol. The experiment is a paired design in which each biological replicate is measured twice (once in each condition). Note the repeated unique identifier for the BioReplicate in the paired design. In addition, the 'Run' column includes the full name of the run and must match exactly with the run name in the files output by the spectral processing tool.

Run	Condition	BioReplicate
CPTAC_CCRCC_W_JHU_20190112_LUMOS_C3L-00004_NAT	NAT_Old	1
CPTAC_CCRCC_W_JHU_20190112_LUMOS_C3L-00004_T	T_Old	1
CPTAC_CCRCC_W_JHU_20190112_LUMOS_C3L-00010_NAT	NAT_Young	2
CPTAC_CCRCC_W_JHU_20190112_LUMOS_C3L-00010_T	T_Young	2
CPTAC_CCRCC_W_JHU_20190112_LUMOS_C3L-00079_NAT	NAT_Young	3
CPTAC_CCRCC_W_JHU_20190112_LUMOS_C3L-00079_T	T_Young	3

Subset of annotation file.

▲ **CRITICAL STEP** This protocol focuses on a protein-level analysis; however, MSstatsShiny includes options here for peptide-level analysis. Simply select 'Peptide' instead of 'Protein' and the platform will analyze any experiment on the peptide level.

◆ **TROUBLESHOOTING**

10. In the gray box on the left-hand side of the page, under '4. Upload quantification dataset', click on 'Browse…' and select the 'msstats.csv' file output from FragPipe. Click on the 'Comma' selector in the 'Column separator in uploaded file' section.

    ▲ **CRITICAL STEP** When a dataset is uploaded to MSstatsShiny, a progress bar will appear below the 'Browse…' button. You must wait until the progress bar is complete before proceeding onto the next steps and clicking the 'Upload Data' button. The progress bar will be complete when the file name in the bar changes to 'Upload complete'. This may take some time, especially if the dataset is very large (e.g., multiple gigabytes).

    ◆ **TROUBLESHOOTING**

11. Leave '5. Upload annotation file' blank. The annotation information is already provided in the 'msstats.csv' file from FragPipe.

    ▲ **CRITICAL STEP** Only FragPipe and Skyline will automatically input the annotation information for MSstats into your input file. For all other tools, you will be required to upload an annotation file.

12. Check, but do not change, the default options under 'Select the options for pre-processing'. The 'Use unique peptides' box will be checked, and both 'Remove proteins with 1 peptide and charge' and 'Remove proteins with 1 feature' will be unchecked.

13. Click on the orange 'Upload Data' button at the bottom of the gray box. A spinning wheel will appear in the middle of the screen, and the rest of the screen will go gray. A summary of the uploaded data and experimental design will appear in the middle of the screen when complete.

    ▲ **CRITICAL STEP** The data upload step may take a long time if the input data are very large. It is important that the tables in the middle of the screen are displayed before proceeding to the next step. If the tables do not appear immediately, continue to wait until they do.

14. Click the orange 'Next step' button to proceed to the 'Data Processing' tab.

Section 3: data processing

● **TIMING 2.5 h**

▲ **CRITICAL** The next step of the MSstatsShiny pipeline is the data processing step. Here, key data transformations, such as normalization and missing value imputation, and summarization from the feature level to the protein level are performed. All data-processing options are included in the gray box of the left-hand side of the screen.

15. Under 'Log transformation', select either 'log2' or 'log10'. This selection will dictate whether the software transforms the raw intensity values to a $\log_2$ or $\log_{10}$ scale. This option is entirely dependent on the individual's ease of interpretation in using either scale and will not affect the downstream statistical results.
16. Under 'Normalization', ensure that 'equalize medians' is selected.
    ▲ **CRITICAL STEP** The normalization option chosen here will have a major effect on the results of the analysis. Ensure that you have a full understanding of the options and the assumptions in your experiment before making this selection. For a detailed explanation of normalization, see Box 2.
17. Under 'Feature subset', select the top N features and enter '20' into the box that appears.
    ▲ **CRITICAL STEP** In DIA experiments, using all the features available can cause your analysis to run very slowly. For more information about feature selection and the options available here, see Box 2.
18. Under 'Missing values (not random missing or censored)', keep the default selection for 'assume all NA as censored' and the default 'Max quantile for censored' at '.999'.
19. Under 'Imputation', uncheck the 'Model based imputation' box.
    ▲ **CRITICAL STEP** Missing value imputation is the most computationally expensive data-processing step in the MSstats workflow. For large DIA experiments, turning off missing value imputation is recommended to prevent the data-summarization step from taking a long time. For more details on missing value imputation, see Box 2.
20. Click the orange 'Run protein summarization' button at the bottom of the gray side panel. A processing bar will pop up in the middle of the screen showing the progress. Wait until the bar is full before proceeding.

Section 4: post-summarization analysis of experimental results and quality control

● **TIMING 5 min (user dependent)**

▲ **CRITICAL** After data pre-processing and summarization have been performed, MSstatsShiny includes multiple options to analyze the results and perform quality control. All results will be shown in the middle of the page, under tabs labeled 'Summarized Results', 'Summarization Plots' and 'Download Data'.

21. View the results of the summarization in the 'Summarized Results' tab. Here, the quantification results are shown on either the condition or biological replicate level. To show the table, click the 'Update Summarized Results' button. All results can be downloaded to a csv file by clicking the 'Download' button.
22. View plots of the summarized results in the 'Summarization Plots' tab. Two plotting options are available: quality control and profile plots.
    - Quality control plots can be shown for all proteins, if the user is interested in overall trends in the data, or on the individual protein level.
    - Profile plots are done for individual proteins and can show features only or features along with the summarized results. Examples of these plots are shown in Fig. 2.
      ▲ **CRITICAL STEP** Both quality control and profile plots are very useful for ensuring that the summarized results are reasonable. We recommend that users spend time reviewing these plots for a subsampling of different proteins. Profile plots can help to dive deeper into the data and provide empirical evidence for the results of statistical testing.
23. Download the summarized results in the 'Download Data' tab. Both the feature-level data and the protein-level (i.e., summarized) data can be downloaded. The feature-level data show additional information, such as the missing value imputation (if it was performed).

BOX 2

Data-processing options

1. Normalization
   Four options for normalization are included in MSstats: median, quantile, global standards and no normalization. There is no single best normalization for all experiments. Researchers must consider the assumptions underlying each normalization option and the appropriateness of the assumptions for their study. Below, we summarize the normalization options, their assumptions and the effect on downstream statistical analysis.

Name	Description	Assumption	Effect
Median	Equalize medians of all log feature intensities in each run	All steps of data collection and acquisition were randomized	The normalization estimates the artifact deviations in each run with a single quantity, reducing overfitting
		Most of the proteins in the experiment are the same and have the same concentration for all of the runs	The normalization reduces bias and variance of the estimated log fold change
		The experimental artifacts affect every peptide in a run by the same constant amount	
Quantile	Equalize the distributions of all log feature intensities in each run	All steps of data collection and acquisition were randomized	The normalization estimates the artifact deviations in each run with a complex non-linear function, potentially leading to overfitting
		Most of the proteins in the experiment are the same and have the same concentration for all of the runs	The normalization reduces bias and variance of the estimated log fold change but may over-correct
		The experimental artifacts affect every peptide non-linearly, as a function of its log intensity	
Global standards	Equalize median log-intensities of endogenous or spiked-in reference peptides or proteins. Apply adjustment to the remainder of log feature intensities	All steps of data collection and acquisition were randomized	The normalization estimates the artifact deviations in each run with a single quantity, which reduces overfitting
		The reference peptides or proteins are present in each run and have the same concentration for all of the runs	The normalization estimates the artifact deviations from a small number of peptides, which may increase overfitting
		All experimental artifacts occur only after standards were added	The normalization does not eliminate artifacts that occurred before adding spiked references
		The experimental artifacts affect every protein in a run by the same constant amount	The normalization reduces bias and variance of the estimated log fold change
None	Do not apply any normalization	All steps of data collection and acquisition were randomized	All patterns of variation of interest and of nuisance variation are preserved
		The experiment has no systematic artifacts or has been normalized in another custom manner	

   If the assumptions of the normalization are not verified, the normalization may, in fact, increase bias or variance of the estimated log fold change. For example, if the experiment is not randomized and the experimental artifacts are confounded with the conditions, the median and quantile normalizations will introduce bias.
2. Selecting informative features for protein summarization
   Feature selection is used to determine which protein features should be used to infer the overall protein abundance in a sample. The options here are:
   - Using all features
   - Using the top 'N' features
   - Removing uninformative features and outliers

   Using all features will simply leverage all available information to infer the underlying protein abundance. Top 'N' features selects a pre-specified number of features with the highest average intensity across all runs for protein-level inference. This option is useful if you believe that the features with lower average intensity are less reliable, or in cases in which some of the proteins have a very large number of features (such as in DIA experiments). For any individual protein, it is usually possible to determine changes in abundance by looking at the peaks with highest intensity; in these cases, using all features results in redundancy while greatly increasing the computational processing time. Finally, removing uninformative features and outliers attempts to select the 'best' features by removing features that have too many missing values, that are too noisy or have outliers[22].

(continued from previous page)

3. Missing value imputation
   Missing value imputation attempts to infer feature intensities in runs in which they were not measured. MSstats imputes these values by using an accelerated failure time model[45]. For a detailed mathematical description of the imputation methods as well as more practical visualizations, see Kohler et al.[24]

Name	Description	Assumption	Effect
Imputation	Infer missing feature intensities by using an accelerated failure time model. It will not impute for runs in which all features are missing	Features are missing for reasons of low abundance (e.g., features are missing not at random)	If the assumption is true, imputation will remove bias toward high intensities in the summarization step. Otherwise, bias will be introduced via inaccurate imputation
No imputation	Do not apply imputation	Assume no information about reasons for missingness or that features are missing at random	If the assumption is true, no new bias will be introduced. Otherwise, if features are missing for reasons of low abundance, summarized values will be biased toward high intensities

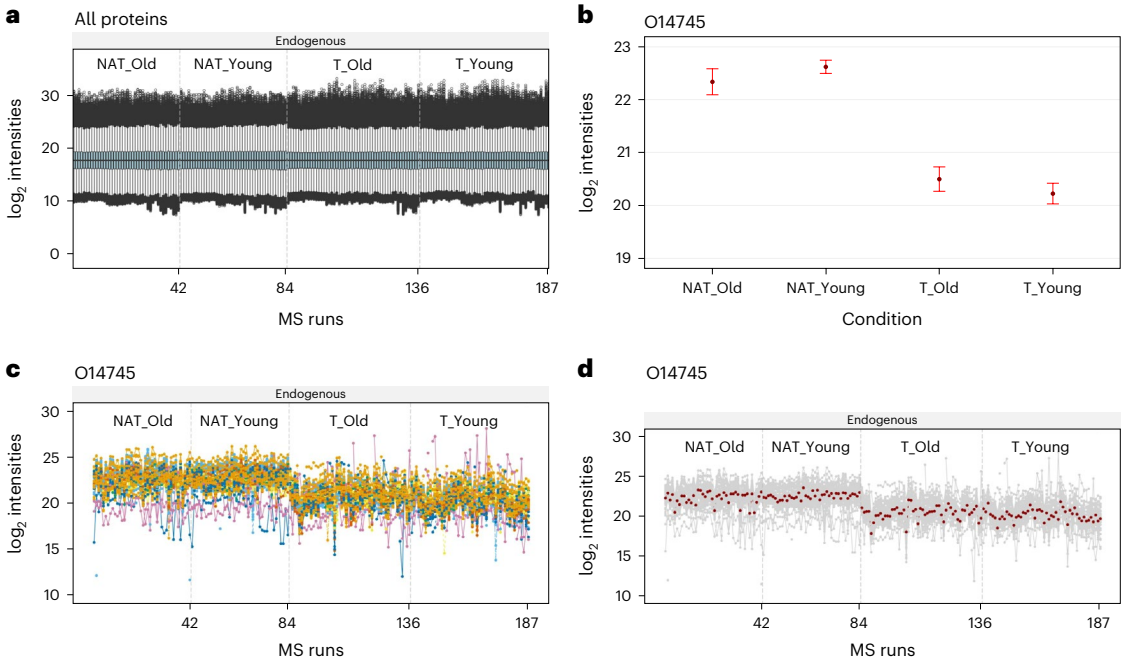

Fig. 2 | Overview of the plots available in MSstats after data pre-processing and summarization. Plot **a** shows the feature intensities for all proteins and features across the entire experiment, whereas **b**–**d** show results for a single protein. For **b**–**d**, protein O14745 (NHRF1) was chosen because it was shown to be differentially abundant after analysis. **a**, Quality control plot. This plot is used to visualize experiment-wide trends across all proteins and features before summarization and can be used to determine the effectiveness of normalization methods. A single boxplot of all feature intensities is created for each MS run. The runs are separated into sections determined by the four conditions in the experiment: NAT measured at young (NAT_Young) and old (NAT_Old) ages and T tissues measured at young (T_Young) and old (T_Old) ages. The *x*-axis shows the MS runs separated by each condition in the experiment. The *y*-axis shows the $\log_2$-transformed intensities of all features in each run. The blue area represents the inner quartile range (e.g., the inner second and third quartiles) of intensities, the lines around the blue areas represent the first and fourth quartiles, and the area around the lines are outlier points. The median of each boxplot is equalized by the selected 'equalize median' normalization option. **b**, Condition plot for protein O14745 (NHRF1). This plot shows the mean plot of log ratios across conditions. The *x*-axis represents the conditions in the experiment, and the *y*-axis represents the $\log_2$-transformed intensities. The error bars show the 95% confidence interval around the mean for each condition. **c**, Feature-level profile plot for protein O14745. Each color represents a single peptide, and fragments of the same peptide are the same color but differentiated by different line types (e.g., solid versus dashed lines). The *x*-axis shows the MS runs, and the *y*-axis shows the $\log_2$-transformed feature intensities. **d**, Summarized level profile plot for protein O14745. The features from **c** are shown in gray, and the summarized values are overlaid and shown in dark red.

Protein-level data will show summarized protein intensities and provide additional summary statistics, such as the percentage of missing features in the summarized run.

24. Once you are satisfied with the results of this step, click the orange 'Next Step' button.

BOX 3

Creating a contrast matrix

MSstats uses a user-defined contrast matrix to determine what experimental conditions to compare. These comparisons can be simple pairwise comparisons or more complex averaging of conditions. For example, the dataset in this protocol includes four conditions: NAT_Young, NAT_Old, T_Young and T_Old. The contrast for a simple comparison between NAT_Young and T_Young would be as shown below.

Label	NAT_Young	NAT_Old	T_Young	T_Old
T-NAT Young	−1	0	1	0

On the other hand, we could also compare the average of NAT versus the average of T over all ages:

Label	NAT_Young	NAT_Old	T_Young	T_Old
T-NAT	−1/2	−1/2	1/2	1/2

Adding in a comparison between NAT_Old and T_Old, the full contrast matrix used in this protocol is:

Label	NAT_Young	NAT_Old	T_Young	T_Old
T-NAT	−1/2	−1/2	1/2	1/2
T-NAT Young	−1	0	1	0
T-NAT Old	0	−1	0	−1

MSstats supports any number of additional covariates by using contrasts. To add a covariate, the variable must be added into the condition and then averaged out by using the contrast matrix, as above for the T-NAT comparison. The only caveat to this is that the covariates cannot be continuous variables. For instance, in this experiment, age was bucketed into groups: young and old. Adding in covariates reduces the standard error and will probably increase the number of differentially abundant proteins. However, covariates will also increase the degrees of freedom, which can cause the model to overfit.

Section 5: statistical inference

● **TIMING 10 min**

▲ **CRITICAL** The next step of the MSstatsShiny pipeline is the Statistical Inference step. Here, the summarized results from the previous step are used to perform group comparison and determine differentially abundant proteins between conditions. All modeling options are included in the gray box of the left-hand side of the screen.

25. Define your comparisons of interest via a contrast matrix under the 'Define comparisons – contrast matrix'. Check the 'Create custom non-pairwise comparisons' box.
    ▲ **CRITICAL STEP** The contrast matrix is how you define what conditions you want to compare in your experiment. These comparisons can be pairwise (comparing only two conditions) or more complex (such as an average of multiple conditions). For a more thorough review of contrast matrices and how to make more complex comparisons, see Box 3.
26. Enter the data that you would like to compare. In this case, enter three comparisons.
    - First, enter 'T-NAT Young' in the 'Comparison Name' box, enter '-1' into the 'NAT_Young' box and enter '1' into the 'T_Young' box and click the orange 'Add' button.
    - Next, enter 'T-NAT Old' in the 'Comparison Name' box, enter '-1' into the 'NAT_Old' box and enter '1' into the 'T_Old' box and click the orange 'Add' button.
    - Finally, enter 'T-NAT' in the 'Comparison Name' box, '-1/2' into the 'NAT_Old' box, '-1/2' into the 'NAT_Young' box, '1/2' into the 'T_Old' box and '1/2' into the 'T_Young' box. Click the orange 'Add' button.

    ◆ **TROUBLESHOOTING**
27. In the 'Group comparison' section, set the significance threshold (e.g., the alpha value) that you would like to use, by using the slider under the 'Significance level' header. The default is set at 0.05. Then, click the orange 'Start' button. A progress bar will appear; wait for the bar to complete. The modeling results will then appear in the middle of the page. The only proteins that will show up in the application are proteins that were marked as differentially abundant.
28. Download the modeling results by using the buttons under the results table. To download all the modeling results, including proteins that were not differentially abundant, click the 'Download all modeling results' button. To download the differentially abundant proteins only, click the 'Download significant proteins' button.

29. Download the code for your analysis by clicking the green 'Download analysis code' button at the top of the page. This button will appear only after the modeling step is complete. The application has automatically tracked all the analysis options chosen in the protocol up to this point and will insert them into this code. The code will recreate the analysis performed exactly.
▲ **CRITICAL STEP** The code version of your analysis is the best way to document what steps were taken so that it can be reliably reproduced. The downloaded code also includes notes on what version of MSstats was used so that the analysis can be repeated with the same results.

Section 6: analysis of statistical results

● **TIMING 5 min (user dependent)**

▲ **CRITICAL** After the group comparison has been performed, MSstatsShiny provides different plotting options to analyze the statistical inference results. All plotting options are available under 'Visualization - select plot type' in the gray box at the bottom left-hand side of the application. Example outputs for all plots can be seen in Fig. 3.

30. Analyze the results of your inference by using a volcano plot. Select 'Volcano Plot' and the comparison of interest under 'Select comparison to plot'. You can view the plot in the application by clicking the 'View plot in browser (only for one comparison/protein)' button. The plot can be saved to a PDF by clicking the 'Save plot results as pdf' button.
◆ **TROUBLESHOOTING**
31. Analyze the results with a heatmap. Select 'Heatmap' in the drop-down and specify the other plot options as desired. You can view the plot in the application by clicking the 'View plot in browser (only for one comparison/protein)' button. The plot can be saved to a PDF by clicking the 'Save plot results as pdf' button.
▲ **CRITICAL STEP** Heatmaps will show all the proteins that were modeled in your experiment. If you try to plot the heatmap in the browser, only one page of the heatmap will

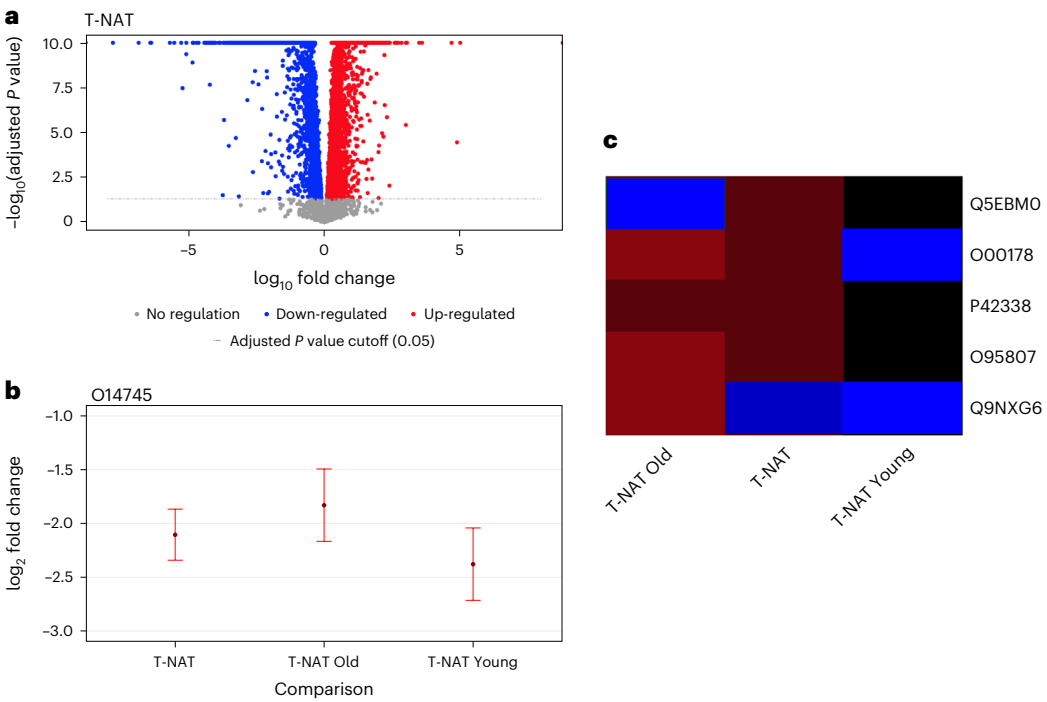

Fig. 3 | Overview of modeling plots available in MSstats. a, Volcano plot. Each dot is a protein in the comparison 'T-NAT'. The *x*-axis shows the $\log_{10}$ fold change, and the *y*-axis shows the $-\log_{10}$ adjusted *P* value. Up-regulated proteins are colored red, and down-regulated proteins are colored blue. Proteins without regulation are gray. **b**, Comparison plot. An individual plot is made for each protein. The *x*-axis shows the different comparisons, and the *y*-axis shows the $\log_2$ fold change. The dots are the reported fold changes, and the error bars are the 95% confidence intervals for the comparison. The confidence interval is calculated on the log scale and shows differences in the log fold change. **c**, Heatmap. The heatmap shows the adjusted *P* value across comparisons for each protein. Each horizontal area is a single protein. Up-regulated proteins are colored red, and down-regulated proteins are colored blue.

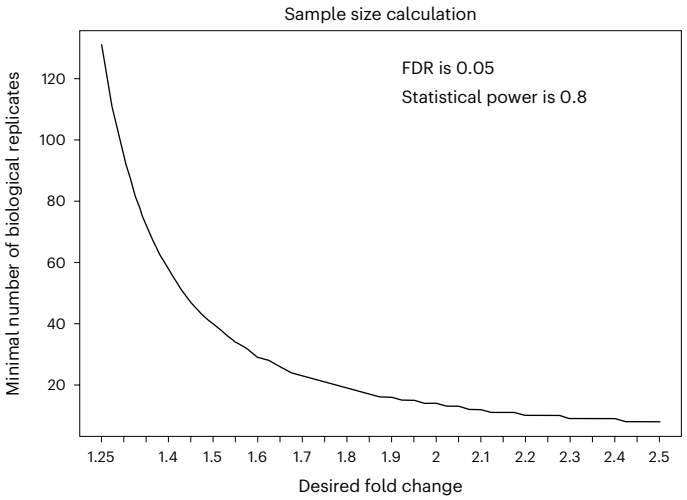

Fig. 4 | Example sample size calculation plot produced by MSstats. The FDR and statistical power were set at 0.05 and 0.8, respectively. The variance was estimated by using the mean of the variance seen in the model. The *x*-axis shows a range of desired fold changes. The *y*-axis shows the number of biological replicates. The line shows the number of biological replicates required to reach a statistical power of 0.8 given the desired fold change. This plot can be used in planning future experiments.

be shown. Generally, the heatmap will include multiple pages. To see all the pages, you must save the heatmap to a PDF.

◆ **TROUBLESHOOTING**

32. Analyze the results with a comparison plot. Select 'Comparison Plot' and choose the protein that you want to visualize. You can view the plot in the application by clicking the 'View plot in browser (only for one comparison/protein)' button. The plot can be saved to a PDF by clicking the 'Save plot results as pdf' button.
33. Click on the orange 'Next step' button to proceed to the final step of the application.

Section 7: planning of future experiments and sample size calculation

● **TIMING 1 min**

▲ **CRITICAL** MSstats can treat the experiment from the previous steps as a pilot study and use the average variance seen in the differential analysis to estimate how many samples would be needed in future experiments with the same variance.

34. Change the parameters in the 'Design future experiments' section to fit your requirements. You can choose to estimate the number of samples needed by choosing 'Sample size' or calculate the statistical power returned for a given number of samples by choosing 'Power'. To do this, change either the 'Number of samples' or 'Power' sliders depending on the previous choice. In this example, select 'Sample size', set the 'Power' slider to '.8' and change the 'Desired fold change' to be '1.25-2.5'.
35. The plot in the middle of the page will show how many replicates are required to detect a differentially abundant protein for a certain fold change at a power of 0.9. Download the plot by clicking the 'Download plot' button (example shown in Fig. 4).

Procedure 2: MSstats code analysis pipeline

Section 1: processing raw data files with FragPipe

● **TIMING 22.2 h**

1. Process the raw data as described in Steps 1–6 of Procedure 1.

Section 2: data import and converter

● **TIMING MSstats: 27 min; MSstatsBig: 7 min**

▲ **CRITICAL** The first step of the MSstats analysis pipeline is to load the data into R and convert the data into MSstats format. There are various dedicated converters included in MSstats for

different tools depending on the biological question of interest, the acquisition strategy and the labeling type. The full converter list can be seen in Table 1.

2. Load the R packages required for the analysis by executing the relevant commands in the R console.
   - For the MSstats workflow: library(MSstats)
   - For the MSstatsBig workflow, load both MSstats and MSstatsBig: library(MSstats) and library(MSstatsBig)
3. Run the FragPipe converter as described in option A if you are performing the MSstats workflow or follow the steps described in option B if you are using MSstatsBig.

   (A) **MSstats converter**

   ● **TIMING** 27 min

   (i) Load the data into R

   ● **TIMING** 3 min

   ```
   fragpipe_data = data.table::fread("msstats.csv")
   ```

   Note that other data-loading functions are possible here, such as read.csv from base R or read_csv from the readr package. We recommend fread from data.table because it is the fastest option[44].

   ▲ **CRITICAL STEP** To load the 'msstats.csv' file, you need to either set your working directory to the same folder that the file is in or enter the full file path into the `'fread'` function. For more information on working with directories in R, read https://www.r-bloggers.com/2021/05/working-with-files-and-folders-in-r-ultimate-guide/.

   (ii) Run the converter code:

   ```
   msstats_input = FragPipetoMSstatsFormat(fragpipe_data)
   ```

   (B) **MSstatsBig converter**

   ● **TIMING** 7 min

   (i) Run the converter code:

   ```
   msstats_input = BigFragPipetoMSstatsFormat(input_file="msstats.
   csv", output_file_name="msstats_converted.csv", backend="arrow",
   max_feature_count=20)
   ```

   (ii) MSstatsBig will both return the converted data in arrow format and save the output to the path provided in the 'output_file_name'. The user can then load this saved file or simply load the arrow output directly into memory. Run the following code to load the data into memory.

   ```
   msstats_input = dplyr::collect(msstats_input)
   ```

   ◆ **TROUBLESHOOTING**
4. For both Step 3A and 3B, the output will be a dataset in MSstats format that is used for protein-level analysis. This can be easily adjusted for peptide-level analysis by swapping out the protein column with the peptide column before inputting the data into the next step. To do this, simply execute the following line of code:

   ```
   msstats_input$PeptideSequence = msstats_input$ProteinName
   ```

Section 3: data processing and feature-level summarization

● **TIMING MSstats: 2.5 h; MSstatsBig: 30 min**

▲ **CRITICAL** The next step of the MSstats pipeline is the data processing step. Here, key data transformations, such as normalization and missing value imputation, and summarization from the feature level to the protein level are performed.

5. Execute the following code to perform data processing:

```
summarized_data = dataProcess(msstats_input, logTrans = 2, normalization =
"equalizeMedians", featureSubset = "topN", remove_uninformative_feature_
outlier = FALSE, n_top_feature = 20, MBimpute = FALSE)
```

▲ **CRITICAL STEP** The normalization option chosen here, set by using the 'normalization' parameter, will have a major effect on the results of the analysis. Ensure that you have a full understanding of the options and the assumptions in your experiment before selecting. For a detailed explanation of normalization, see Box 2.
▲ **CRITICAL STEP** In DIA experiments, using all the features available can cause your analysis to run very slowly. In addition, when there are a large number of features, it is not necessary to use all available features in your analysis, because they become redundant. To choose the features that the analysis will use, you must adjust the '`featureSubset`' parameter. Here, we use only the top 20 features for the analysis. For more information about feature selection and the options available here, please see Box 2 and Supplementary Method 1.
▲ **CRITICAL STEP** Missing value imputation is the most computationally expensive data-processing step in the MSstats workflow. For large DIA experiments, turning off missing value imputation is recommended to prevent the data-summarization step from taking a long time. Here, we set '`MBimpute`' to 'FALSE' to turn off missing value imputation. For more information about missing value imputation, see Box 2.

6. View the feature-level results (e.g., the normalized features) by looking at the 'FeatureLevelData' data table:

```
head(summarized_data$FeatureLevelData).
```

7. View the protein-level results (e.g., the summarized values with one value per protein per run) by looking at the 'ProteinLevelData' data table:

```
head(summarized_data$ProteinLevelData)
```

▲ **CRITICAL STEP** The data tables generated by performing Steps 6 and 7 can be saved to a flat file (e.g., a csv file). There are multiple ways to do this; however, one option would be to use the `write.csv` function.

Section 4: post-summarization analysis of experimental results and quality control

● **TIMING 5 min (user dependent)**

▲ **CRITICAL** MSstats provides various plotting and quantification functionalities for data exploration and quality control after summarization is performed. Plots include profile, quality control and condition plots and are performed by using the `dataProcessPlots` function. Quantification is done by using the `quantification` function. Examples and descriptions of all plots can be seen in Fig. 2.

8. Run quantification on the sample level and view the results:

```
quantification(summarized, type = "Sample")
```

9. Run quantification on the group level and view the results:

```
quantification(summarized, type = "Group")
```

10. Create a quality control plot for all proteins in the experiment to look for experiment-wide trends in the data:

```
dataProcessPlots(summarized_data, Type="QCPlot", which.
Protein="allonly", Address=FALSE)
```

▲ **CRITICAL STEP** The `dataProcessPlots` function includes various parameters to tweak the plot, including axis range adjustment, scales and text sizes. All available options and detailed descriptions are available in the function documentation (Use ?dataProcessPlots for quick access to the documentation in Rstudio).

▲ **CRITICAL STEP** The address parameter specifies where to save the resulting plot. Setting this parameter to 'FALSE' will show the plot in R directly. Otherwise, the function will save the plot as a PDF, and this parameter indicates the address to which the file should be saved.

11. Create a profile plot for a single protein to look at the results of the upstream data processing:

```
dataProcessPlots(summarized_data, Type="ProfilePlot", originalPlot=TRUE,
summaryPlot=TRUE, which.Protein="O14745", Address=FALSE)
```

▲ **CRITICAL STEP** Profile plots can plot features only or can include the summarized value per run. These plots are controlled by the originalPlot and summaryPlot parameters. originalPlot will plot a profile plot with the features only. summaryPlot will plot a profile plot with the features and summarized values. Setting both to 'TRUE' will create both plots.

12. Create a condition plot for a single protein to look at the results of the upstream data processing:

```
dataProcessPlots(summarized_data, Type="ConditionPlot", interval="CI",
which.Protein=" O14745", Address=FALSE)
```

▲ **CRITICAL STEP** The interval parameter determines what interval is shown around the quantified condition value. This parameter can either be set to 'CI' or 'SD'. 'CI' plots a confidence interval with a 0.95 level for the width of the error bars. 'SD' uses the standard deviation for the width of the error bars.

▲ **CRITICAL STEP** The which.Protein parameter controls which proteins will be plotted. By default, plots are created for all proteins in the dataset, which might be unnecessarily time consuming, especially for larger datasets. Proteins can be identified either by names or order numbers, which correspond to unique levels of the 'Protein' column in dataProcess output.

Section 5: group comparison and statistical results

● **TIMING 6 min**

▲ **CRITICAL** The next step of the MSstats pipeline is to perform statistical inference by using the `groupComparison` function. Here, the summarized results from the previous step are used to perform group comparison and determine differentially abundant proteins between conditions.

13. Specify which conditions to compare by using a contrast matrix:

```
comparison = matrix(c(-1/2, -1/2, 1/2, 1/2, -1, 0, 1, 0, 0, -1, 0,
1), nrow = 3, byrow=TRUE) row.names(comparison) <- c("T-NAT", "T-NAT
Old", "T-NAT Young") colnames(comparison) <- c("NAT_Old", "NAT_Young",
"T_Old", "T_Young")
```

▲ **CRITICAL STEP** The contrast matrix is how you define what conditions you want to compare in your experiment. These comparisons can be pairwise (comparing only two conditions) or more complex (such as an average of multiple conditions). For a more thorough review of contrast matrices, see Box 3.

14. Run the `groupComparison` function by using the contrast matrix and the summarized results:

```
model_results = groupComparison(comparison, summarized_data)
```

15. View the statistical testing results:

```
head(model_results$ComparisonResult)
```

▲ **CRITICAL STEP** The groupComparison returns a list with three objects named 'ComparisonResult', 'ModelQC' and 'FittedModel'.

- The ComparisonResult object is a data frame with results of statistical testing.
- ModelQC is a data frame with the data used to fit models for group comparison.
- FittedModel returns the detailed results of each individual protein model, beyond the results of statistical testing. Although this part of the output is required for sample size calculation, it is not needed in most cases, and including it may result in an unnecessarily large size of the resulting object.

Section 6: analysis of statistical results

● **TIMING 5 min (user dependent)**

▲ **CRITICAL** After group comparison is performed, MSstats includes different plotting options to analyze the statistical inference results. The `groupComparisonPlots` function includes all functionality for plotting, including volcano plots, heatmaps and comparison plots. Example outputs for all plots can be seen in Fig. 3.

16. Analyze the results of your inference by using a volcano plot:

```
groupComparisonPlots(model_results$ComparisonResult,
Type="VolcanoPlot")
```

▲ **CRITICAL STEP** When using the `groupComparisonPlots` function, users must include only the 'ComparisonResult' object from the model results.

▲ **CRITICAL STEP** The `groupComparisonPlots` function includes various parameters to tweak the plot, including axis range, size of points and text sizes. All available options and detailed descriptions are available in the function documentation (use ?groupComparisonPlots for quick access to the documentation in Rstudio).

▲ **CRITICAL STEP** The address parameter specifies where to save the resulting plot. Setting this parameter to 'FALSE' will show the plot in R directly. Otherwise, the function will save the plot as a PDF, and this parameter indicates the address to which the file should be saved.

◆ **TROUBLESHOOTING**

17. Analyze the results with a heatmap:

```
groupComparisonPlots(model_results$ComparisonResult, Type="Heatmap")
```

◆ **TROUBLESHOOTING**

18. Analyze the results with a comparison plot:

```
groupComparisonPlots(model_results$ComparisonResult,
Type="ComparisonPlot", which.Protein="O14745")
```

Section 7: planning of future experiments and sample size calculation

● **TIMING 1 min**

▲ **CRITICAL** MSstats can treat the experiment from the previous steps as a pilot study and use the average variance seen in the differential analysis to estimate how many samples would be needed in future experiments with the same variance. The calculations are performed by using the `designSampleSize` function, and plotting of the calculations is done by using the `designSampleSizePlots` function.

19. Perform future sample size calculations by using the model results:

```
sample_calc = designSampleSize(model_results$FittedModel,
desiredFC=c(1.25,2.5), FDR = 0.05, numSample = TRUE, power = 0.8)
```

20. Plot the results of the sample size calculation (example shown in Fig. 4):

```
designSampleSizePlots(sample_calc)
```

Troubleshooting

Troubleshooting advice can be found in Table 2. For any troubleshooting problems beyond what is listed in this protocol, we direct readers to the MSstats Google Group (https://groups.google.com/g/msstats), which was referenced in Procedure 1, Step 7. This group is an actively maintained forum in which users can ask questions directly to the developers of MSstats. These questions can entail errors encountered in the protocol or more in-depth methodological and experimental design questions.

Table 2 | Troubleshooting table

Step	Problem	Possible reason	Solution
Procedure 1, Step 9	The GUI crashes (error: 'Error in setnames: Can't assign 4 names to a 0 column data.table')	Incorrect experimental design entered into the GUI	Ensure that the experimental design parameters accurately reflect the experiment
Procedure 1, Step 10 and Procedure 2, Step 3	The GUI crashes (error: 'Error in .checkAnnotation: ** Columns Run, Condition, BioReplicate missing in the annotation. Please check the annotation file.')	The 'msstats.csv' file reformatted to a non-csv format	Resave the file in csv format. This can be done directly in tools such as Excel
Procedure 1, Step 26	The GUI shows 'The contrast weights should sum up to 0'	The comparisons entered into '1. Define comparisons – contrast matrix' do not sum to 0	Recheck the values that were entered into this section and ensure that they sum to 0. Do not proceed to the next step until the GUI shows a contrast matrix
Procedure 1, Step 30	The GUI shows an error when trying to create multiple plots at once	Trying to show multiple plots at the same time within the GUI	Save the plots to a PDF rather than showing them within the GUI
Procedure 1, Step 31 and Procedure 2, Step 17	The code results in an error: 'Error in .plotHeatmap(input, logBase.pvalue, ylimUp, FCcutoff, sig, clustering, : At least two comparisons are needed for heatmaps'	Only one comparison was added to the contrast matrix	Add at least one more comparison to the contrast matrix. Heatmaps cannot be created with only one comparison
Procedure 2, Step 16	The code results in an error: 'Error in data.table::setnames(input, colname_log_fc, c("logFC")) : 'old' is length 0 but 'new' is length 1'	The entire output from the `groupComparison` function was used as the input to the `groupComparisonPlots` function	Only input the ComparisonResult object into the `groupComparisonPlots` function. For example, use 'model_results$ComparisonResult'

Timing

Procedure 1

Section 1, Steps 1–6, processing raw data files with FragPipe: 22.2 h
Section 2, Steps 7–14, data uploading: 45 min
Section 3, Steps 15–20, data processing: 2.5 h
Section 4, Steps 21–24, post-summarization analysis of experimental results and quality control: 5 min (user dependent)
Section 5, Steps 25–29, statistical inference: 10 min
Section 6, Steps 30–33, analysis of statistical results: 5 min (user dependent)
Section 7, Steps 34–35, planning of future experiments and sample size calculation: 1 min

Procedure 2

Section 1, Step 1, processing raw data files with FragPipe: 22.2 h
Section 2, Steps 2–4, data import and converter: 27 min (Step 3A, MSstats converter) or 7 min (Step 3B, MSstatsBig converter)
Section 3, Steps 5–7, data processing and feature-level summarization: 2.5 h (Step 3A, MSstats converter) or 30 min (Step 3B, MSstatsBig converter)
Section 4, Steps 8–12, post-summarization analysis of experimental results and quality control: 5 min (user dependent)
Section 5, Steps 13–15, group comparison and statistical results: 6 min
Section 6, Steps 16–18, analysis of statistical results: 5 min (user dependent)
Section 7, Steps 19–20, planning of future experiments and sample size calculation: 1 min

Table 3 | Expected results

MSstats component	Result	Example
Tool-specific conversion	Feature-level data in MSstats format	'msstats_format.Rdata' file available on MassIVE.quant RMSV000000696.1
Data pre-processing and summarization	Feature-level data with additional information including missing value imputation, censored values and normalization	FeatureLevelData table in the 'summarized_data.Rdata' file available on MassIVE.quant RMSV000000696.1
Data pre-processing and summarization	Protein-level data with one value per protein per run; includes summary statistics such as the number of features used and percentage of missing values; used as input into the statistical model	ProteinLevelData table in the 'summarized_data.Rdata' file available on MassIVE.quant RMSV000000696.1
Data pre-processing and summarization	Quality Control plot; visualizes the distribution of all features per run; used to see experiment-wide trends across runs and to visualize the effect of normalization	Fig. 2a
Data pre-processing and summarization	Condition plot; visualizes the distribution of replicates per condition for a single protein; shows the mean log intensity across conditions and error bars representing the 95% confidence interval	Fig. 2b
Data pre-processing and summarization	Profile plot; visualizes a single protein per run; shows all features for a single protein and how they change across runs and conditions; if plotting only features, each feature is a separate color or line type; can optionally overlay the summarized value on top of gray-colored features	Fig. 2c: features only; Fig. 2d: summarized values overlayed on top of features
Group comparison and statistical modeling	Model results, including information about the protein, comparison, log fold change, standard error, degrees of freedom, *P* value and adjusted *P* value; includes a note column to indicate where modeling could not be performed, such as in the case where one condition is entirely missing	'model_results.Rdata' file available on MassIVE.quant RMSV000000696.1
Group comparison and statistical modeling	Volcano plot; visualizes the statistical results for all proteins in a single comparison; shows fold change and adjusted *P* value, as well as if the proteins are up- or down-regulated; used to visualize comparison level trends in modeling results	Fig. 3a
Group comparison and statistical modeling	Comparison plot; visualizes a single protein across all comparisons; for each comparison, the center is the reported log fold change, and the error bars represent the 95% confidence interval; if the 95% confidence interval crosses 0, this would indicate that the adjusted *P* value is greater than 0.05	Fig. 3b
Group comparison and statistical modeling	Heatmap; visualizes the statistical results for multiple proteins across all comparisons; used to visualize how the statistical results change across comparisons	Fig. 3c
Sample size calculation for future experiments	Sample size calculation results; can be made for a given power or a given number of samples; uses the mean variance seen in the modeling step as input	'sample_size_calc.Rdata' file available on MassIVE.quant RMSV000000696.1
Sample size calculation for future experiments	Plot showing the results of the sample size calculation; shows the number of replicates required to reach a statistical power of 0.8 given a desired fold change; note that even at a larger fold change of 1.5, 40 biological replicates are required in each group	Fig. 4

Anticipated results

Both procedures provide data tables and plots of the results at the end of each main component. In Table 3, we report the expected results and point to where examples of each output can be found. MassIVE.quant RMSV000000696.1 can be accessed from https://massive.ucsd.edu/ProteoSAFe/reanalysis.jsp?task=b88e6fe3f3564773be62eefde7122127.

Data availability

The dataset used in this protocol is freely available at https://pdc.cancer.gov/pdc/study/PDC000200 and MassIVE (https://massive.ucsd.edu/ProteoSAFe/dataset.jsp?task=6847574c13964a1f9482ee4d71f33eb1. The quantification results from FragPipe and the MSstats processed data at each step are available in the MassIVE.quant Reanalysis RMSV000000696.1. Source data are provided with this paper.

Code availability

All analysis scripts to recreate Procedure 2 can be found in the same MassIVE.quant Reanalysis RMSV000000696.1.

Received: 28 September 2023; Accepted: 11 March 2024;
Published online: 20 May 2024

References

1. Shuken, S. R. An introduction to mass spectrometry-based proteomics. *J. Proteom. Res.* **22**, 2151–2171 (2023).
2. Aebersold, R. & Mann, M. Mass spectrometry-based proteomics. *Nature* **422**, 198–207 (2003).
3. Ong, S.-E. & Mann, M. Mass spectrometry–based proteomics turns quantitative. *Nat. Chem. Biol.* **1**, 252–262 (2005).
4. Borràs, E. & Sabidó, E. What is targeted proteomics? A concise revision of targeted acquisition and targeted data analysis in mass spectrometry. *Proteomics* **17**, 1700180 (2017).
5. Mann, M. & Jensen, O. N. Proteomic analysis of post-translational modifications. *Nat. Biotechnol.* **21**, 255–261 (2003).
6. Li, Z. et al. Systematic comparison of label-free, metabolic labeling, and isobaric chemical labeling for quantitative proteomics on LTQ Orbitrap Velos. *J. Proteome Res.* **11**, 1582–1590 (2012).
7. Poulos, R. C. et al. Strategies to enable large-scale proteomics for reproducible research. *Nat. Commun.* **11**, 3793 (2020).
8. Cai, X. et al. PulseDIA: data-independent acquisition mass spectrometry using multi-injection pulsed gas-phase fractionation. *J. Proteome Res.* **20**, 279–288 (2021).
9. Krzywinski, M. & Altman, N. Power and sample size. *Nat. Methods* **10**, 1139–1140 (2013).
10. 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* **14**, 513–520 (2017).
11. 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* **17**, 41–44 (2020).
12. Bernhardt, O. M. et al. Spectronaut: A Fast and Efficient Algorithm for MRM-Like Processing of Data Independent Acquisition (SWATH-MS) Data. Presented at Proceedings of the 60th ASMS Conference on Mass Spectrometry and Allied Topics, Vancouver, BC, Canada, (unpublished), https://f1000research.com/posters/1096450 (2012).
13. MacLean, B. et al. Skyline: an open source document editor for creating and analyzing targeted proteomics experiments. *Bioinformatics* **26**, 966–968 (2010).
14. Tyanova, S., Temu, T. & Cox, J. The MaxQuant computational platform for mass spectrometry-based shotgun proteomics. *Nat. Protoc.* **11**, 2301–2319 (2016).
15. Röst, H. L. et al. OpenSWATH enables automated, targeted analysis of data-independent acquisition MS data. *Nat. Biotechnol.* **32**, 219–223 (2014).
16. Zhang, F., Ge, W., Ruan, G., Cai, X. & Guo, T. Data-independent acquisition mass spectrometry-based proteomics and software tools: a glimpse in 2020. *Proteomics* **20**, e1900276 (2020).
17. Demichev, V. et al. dia-PASEF data analysis using FragPipe and DIA-NN for deep proteomics of low sample amounts. *Nat. Commun.* **13**, 3944 (2022).
18. Yu, F. et al. Analysis of DIA proteomics data using MSFragger-DIA and FragPipe computational platform. *Nat. Commun.* **14**, 4154 (2023).
19. Käll, L. & Vitek, O. Computational mass spectrometry–based proteomics. *PLoS Comput. Biol.* **7**, e1002277 (2011).
20. Molloy, M. P., Brzezinski, E. E., Hang, J., McDowell, M. T. & VanBogelen, R. A. Overcoming technical variation and biological variation in quantitative proteomics. *Proteomics* **3**, 1912–1919 (2003).
21. Clough, T., Thaminy, S., Ragg, S., Ruedi, A. & Vitek, O. Statistical protein quantification and significance analysis in label-free LC-MS experiments with complex designs. *BMC Bioinforma.* **13**, S6 (2012).
22. Tsai, T.-H. et al. Selection of features with consistent profiles improves relative protein quantification in mass spectrometry experiments. *Mol. Cell. Proteom.* **19**, 944–959 (2020).
23. Girden, E. R. *ANOVA: Repeated Measures* (Sage Publications, 1992).
24. Kohler, D. et al. MSstats version 4.0: statistical analyses of quantitative mass spectrometry-based proteomic experiments with chromatography-based quantificationat scale. *J. Proteome Res.* **22**, 1466–1482 (2023).
25. Choi, M. et al. MSstats: an R package for statistical analysis of quantitative mass spectrometry-based proteomic experiments. *Bioinformatics* **30**, 2524–2526 (2014).
26. Goeminne, L. J. E., Sticker, A., Martens, L., Gevaert, K. & Clement, L. MSqRob takes the missing hurdle: uniting intensity- and count-based proteomics. *Anal. Chem.* **92**, 6278–6287 (2020).
27. Sticker, A., Goeminne, L., Martens, L. & Clement, L. Robust summarization and inference in proteomewide label-free quantification. *Mol. Cell. Proteom.* **19**, 1209–1219 (2020).
28. Goeminne, L. J. E., Gevaert, K. & Clement, L. Peptide-level robust ridge regression improves estimation, sensitivity, and specificity in data-dependent quantitative label-free shotgun proteomics. *Mol. Cell. Proteom.* **15**, 657–668 (2016).
29. Zhu, et al. DEqMS: a method for accurate variance estimation in differential protein expression analysis. *Mol. Cell. Proteom.* **19**, 1047–1057 (2020).
30. Wolski, W. E. et al. prolfqua: a comprehensive R-package for proteomics differential expression analysis. *J. Proteome Res.* **22**, 1092–1104 (2023).
31. Bai, M. et al. LFQ-based peptide and protein intensity differential expression analysis. *J. Proteome Res.* **22**, 2114–2123 (2023).
32. Gatto, L. & Vanderaa, C. *R Package Version 1.13.1*, https://github.com/RforMassSpectrometry/QFeatures (2023).
33. Simmons, J. P., Nelson, L. D. & Simonsohn, U. False-positive psychology: undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychol. Sci.* **22**, 1359–1366 (2011).
34. Kohler, D. et al. MSstatsShiny: a GUI for versatile, scalable, and reproducible statistical analyses of quantitative proteomic experiments. *J. Proteome Res.* **22**, 551–556 (2023).
35. Yang, K. L. et al. MSBooster: improving peptide identification rates using deep learning-based features. *Nat. Commun.* **14**, 4539 (2023).
36. Leprevost, F. D. V. et al. Philosopher: a versatile toolkit for shotgun proteomics data analysis. *Nat. Methods* **17**, 869–870 (2020).
37. Kohler, D. et al. MSstatsPTM: statistical relative quantification of posttranslational modifications in bottom-up mass spectrometry-based proteomics. *Mol. Cell. Proteom.* **22**, 100477 (2023).
38. Huang, T. et al. MSstatsTMT: statistical detection of differentially abundant proteins in experiments with isobaric labeling and multiple mixtures. *Mol. Cell. Proteom.* **19**, 1706–1723 (2020).
39. Malinovska, L. et al. Proteome-wide structural changes measured with limited proteolysis-mass spectrometry: an advanced protocol for high-throughput applications. *Nat. Protoc.* **18**, 659–682 (2022).
40. Richardson, N., et al. *Apache/Arrow*, https://github.com/apache/arrow/, https://arrow.apache.org/docs/r/ (2023).
41. Zaharia, M., Xin, R. S., Wendell, P., Das, T. & Armbrust, M. Apache Spark: a unified engine for big data processing. *Commun. ACM* **59**, 56–65 (2016).
42. Feng, et al. Global analysis of protein structural changes in complex proteomes. *Nat. Biotechnol.* **32**, 1036–1044 (2014).
43. Clark, D. J., Dhanasekaran, S. M., Petralia, F., Wang, P. & Zhang, H. Integrated proteogenomic characterization of clear cell renal cell carcinoma. *Cell* **179**, 964–983 (2019).
44. Dowle, M. & Srinivasan, A. *data.table*, https://r-datatable.com, https://Rdatatable.gitlab.io/data.table, https://github.com/Rdatatable/data.table (2023).
45. Venables, W. & Ripley, B. *Modern Applied Statistics with S* 359–364 (Springer, 2002).

Acknowledgements
We thank J. Carver for his help in setting up the MassIVE container that allowed us to share the datasets and analysis code for this paper. This work was supported by awards NSF-BIO/DBI-1759736 (to O.V.), NSF-BIO/DBI-1950412 (to O.V.) and NIH-NLM-1R01LM013115 (to O.V.), the Chan-Zuckerberg Foundation (to O.V.) and National Institutes of Health grants R01-GM-094231 and U24-CA271037 (to A.I.N.). M.S. was partially financially supported by the National Science Centre, Poland, grant Preludium 2020/37/N/ST6/04070.

Author contributions
D.K. analyzed the data in MSstats and wrote the relevant MSstats sections of the manuscript. D.K. and O.V. wrote the introduction for the manuscript. M.S. implemented the methods in MSstatsBig. F.Y. analyzed the data by using FragPipe and wrote the relevant FragPipe sections of the paper. F.Y. and A.I.N. determined the experimental dataset for the manuscript. A.I.N., O.V. and D.K. conceptually developed and scoped the manuscript. All authors provided feedback and edited the manuscript.

Competing interests
A.I.N. and F.Y. receive royalties from the University of Michigan for the sale of MSFragger and IonQuant software licenses to commercial entities. All license transactions are managed by the University of Michigan Innovation Partnerships office, and all proceeds are subject to the university technology transfer policy. The other authors declare no competing interests.

Additional information
Extended data is available for this paper at https://doi.org/10.1038/s41596-024-01000-3.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41596-024-01000-3.

Correspondence and requests for materials should be addressed to Olga Vitek.

Peer review information *Nature Protocols* thanks Chu Wang, Witold Wolski and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

Related links

Key references using this protocol

Kohler, D. et al. *J. Proteome Res.* **22**, 1466–1482 (2023)
Kohler, D. et al. *J. Proteome Res.* **22**, 551–556 (2023)
Kong, A. et al. *Nat. Methods* **14**, 513–520 (2017)
Yu, F. et al. *Nat. Commun.* **14**, 4154 (2023)
Clark, D. J. et al. *Cell* **179**, 964–983 (2019)