Protein significance analysis of mass spectrometry-based proteomics experiments with R and MSstats (v4.6.3)

Meena Choi & Tsung-Heng Tsai

Jaunary, 2023

Contents

---

1. Statistical relative protein quantification: SRM, DDA and DIA experiments

`MSstats` is an open-source R-based package for statistical relative quantification of peptides and proteins in mass spectrometry-based proteomic experiments. This document describes `MSstats`, the most recent version of the package, and its use through the command line.

1.1 Applicability

`MSstats` version 4.0 and above is applicable to multiple types of sample preparation, including label-free workflows, workflows that use stable isotope labeled reference proteins and peptides, and workflows that use fractionation. It is applicable to targeted Selected Reaction Monitoring (SRM), Data-Dependent Acquisition (DDA or shotgun), and Data-Independent Acquisition (DIA or SWATH-MS). It is applicable to experiments that make arbitrary complex comparisons of experimental conditions or times.

`MSstats` is not applicable to experiments that compare multiple metabolically labeled endogenous samples within a same run, such as experiments with iTRAQ labeling or TMT labeling. These experiments are supported by `MSstatsTMT`, which is a sibling package. Please check `MSstatsTMT` in Bioconductor.

1.2 Statistical functionalities

`MSstats` version 4.0 and above performs three analysis steps. The first step, *data processing, visualization, and run-level summarization*, transforms, normalizes and summarizes the intensities of the peaks per MS run and per protein, and generates workflow-specific and customizable numeric summaries for data visualization and quality control.

The second step, *statistical modeling and inference*, automatically detects the experimental design (e.g. group comparison, paired design or time course, presence of labeled reference peptides or proteins) from the data. It then reflects the experimental design and the type of spectral acquisition strategy, and fits an appropriate linear mixed model by means of `lm` and `lmer` functionalities in R. The model is used to detect differentially abundant proteins or peptides, or to summarize the protein or peptide abundance in a single biological replicate or condition (that can be used, e.g. as input to clustering or classification).

The third step, *statistical experimental design*, views the dataset being analyzed as a pilot study of a future experiment, utilizes the variance components of the current dataset, and calculates the minimal number of replicates necessary in the future experiment to achieve a pre-specified statistical power.

1.3 Interoperability with existing computational tools

`MSstats` takes as input data in a tabular .csv format, which can be generated by any spectral processing tool such as Skyline (MacLean et al. 2010), MaxQuant (Cox and Mann 2008), Progenesis QI(Nonlinear dynamics/Waters), Proteome Discoverer (Thermo Scientific) MultiQuant(Applied Biosystems), OpenMS (Sturm et al. 2008), SuperHirn (Mueller et al. 2007), OpenSWATH (Röst et al. 2014), Spectronaut(Biognosys), or DIA-Umpire(Tsou et al. 2015). The functions to convert the required format from several processing tools are available from `MSstats` v4.6. Details are in the section below.

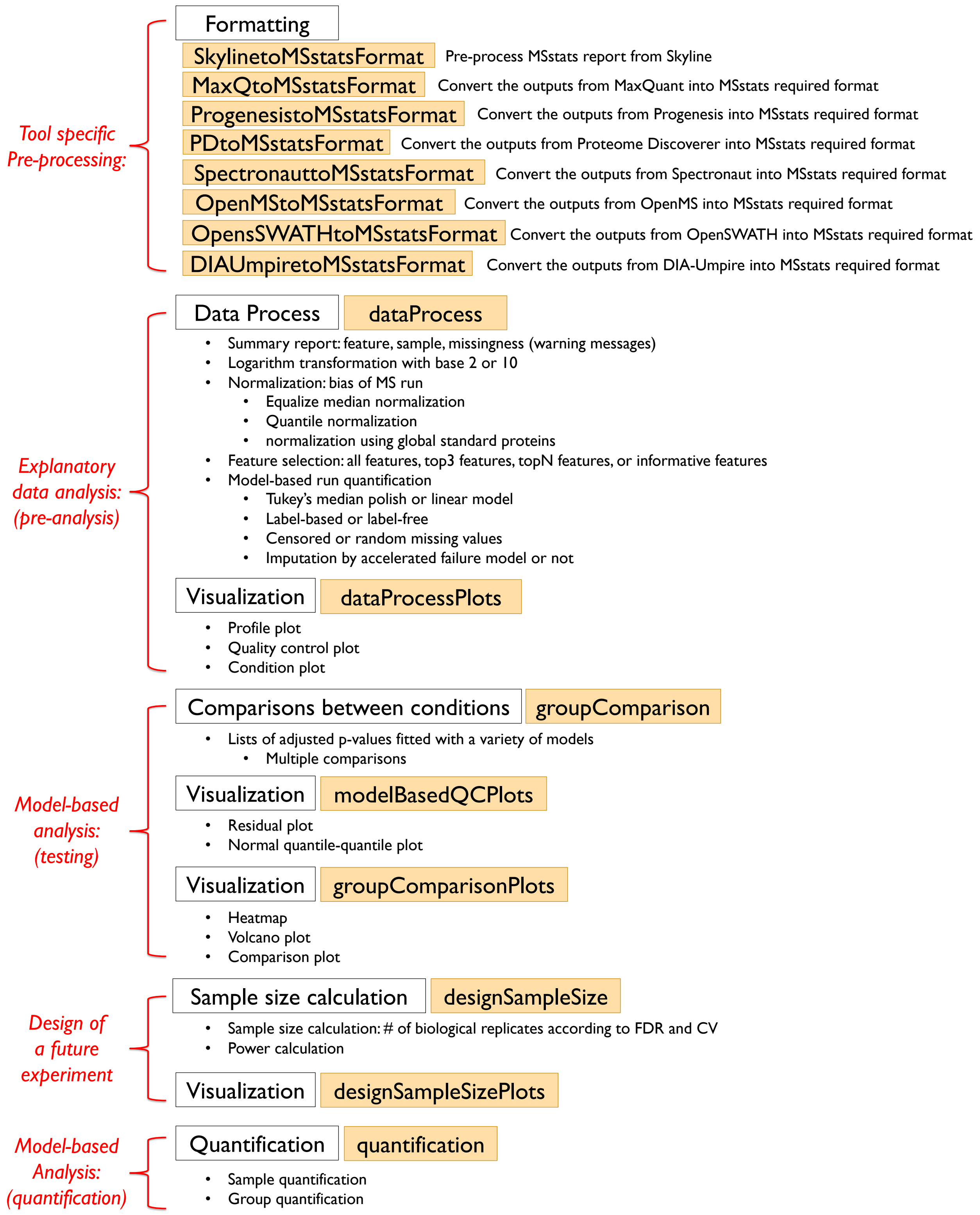
For statistics experts, `MSstats` 4.0 and above satisfies the interoperability requirements of Bioconductor. The command line-based workflow is partitioned into a series of independent steps, that facilitate the development and testing of alternative statistical approaches. It complies with the maintenance and documentation requirements of Bioconductor.

`MSstats` 4.0 and above is available as an external tool within Skyline. The external tool support within Skyline manages `MSstats` installation, point-and-click execution, parameter collection in Windows forms and output display. Skyline manages the annotations of the experimental design, and the processing of raw data. It outputs a custom report, that is fed as a single stream input into `MSstats`. This design buffers proteomics users from the details of the R implementation, while enabling rigorous statistical modeling. Also, `MSstat` can be combined with an OpenMS preprocessing pipeline (e.g. in KNIME). The OpenMS experimental design is used to present the data in an MSstats-conformant way for the analysis. Details are available in OpenMS tutorial.

1.4 Availability

`MSstats` is available under the Artistic-2.0 license at msstats.org. `MSstats` as an external tool for Skyline is available at http://proteome.gs.washington.edu/software/Skyline/tools.html. `MSstats` is now also available in Bioconductor. The most recent version of the package is available at msstats.org or MSstats GitHub. We suggest to use that if possible. The versioning of the main package is updated several times a year, to synchronise with the Bioconductor release.

1.5 Overview of the functionalities



Tool specific Pre-processing:

Formatting

- SkylinetoMSstatsFormat — Pre-process MSstats report from Skyline
- MaxQtoMSstatsFormat — Convert the outputs from MaxQuant into MSstats required format
- ProgenesistoMSstatsFormat — Convert the outputs from Progenesis into MSstats required format
- PDtoMSstatsFormat — Convert the outputs from Proteome Discoverer into MSstats required format
- SpectronauttoMSstatsFormat — Convert the outputs from Spectronaut into MSstats required format
- OpenMStoMSstatsFormat — Convert the outputs from OpenMS into MSstats required format
- OpensSWATHtoMSstatsFormat — Convert the outputs from OpenSWATH into MSstats required format
- DIAUmpiretoMSstatsFormat — Convert the outputs from DIA-Umpire into MSstats required format

Explanatory data analysis: (pre-analysis)

Data Process — dataProcess

- Summary report: feature, sample, missingness (warning messages)
- Logarithm transformation with base 2 or 10
- Normalization: bias of MS run
  - Equalize median normalization
  - Quantile normalization
  - normalization using global standard proteins
- Feature selection: all features, top3 features, topN features, or informative features
- Model-based run quantification
  - Tukey's median polish or linear model
  - Label-based or label-free
  - Censored or random missing values
  - Imputation by accelerated failure model or not

Visualization — dataProcessPlots

- Profile plot
- Quality control plot
- Condition plot

Model-based analysis: (testing)

Comparisons between conditions — groupComparison

- Lists of adjusted p-values fitted with a variety of models
  - Multiple comparisons

Visualization — modelBasedQCPlots

- Residual plot
- Normal quantile-quantile plot

Visualization — groupComparisonPlots

- Heatmap
- Volcano plot
- Comparison plot

Design of a future experiment

Sample size calculation — designSampleSize

- Sample size calculation: # of biological replicates according to FDR and CV
- Power calculation

Visualization — designSampleSizePlots

Model-based Analysis: (quantification)

Quantification — quantification

- Sample quantification
- Group quantification

1.6 Troubleshooting

To help troubleshoot potential problems with installation or functionalities of `MSstats`, a progress report is generated in a separate log file, *msstats.log.* The file includes information on the R session (R version, loaded software libraries), options selected by the user, checks of successful completion of intermediate analysis steps, and warning messages. If the analysis produces an error, the file contains suggestions for possible reasons for the errors. If a file with this name already exists in working directory, a suffix with a number will be appended to the file name. In this way a record of all the analyses is kept. Please see the file `KnownIssues-Skyline-MSstatsV3.6.pdf` on the "Installation" section of "MSstats" page in `msstats.org` for a list of known issues and possible solutions for installation problem of MSstats external tool in Skyline

---

2. Allowable data formats

2.1 SRM with stable isotope labeled reference peptides

2.1.1 10-column format

`MSstats` performs statistical analysis steps, that follow peak identification and quantitation. Therefore, input to MSstats is the output of other software tools (such as Skyline or MultiQuant) that read raw spectral files and identify and quantify spectral peaks. The preferred structure of data for use in `MSstats` is a .csv file in a "long" format with 10 columns representing the following variables: `ProteinName`, `PeptideSequence`, `PrecursorCharge`, `FragmentIon`, `ProductCharge`, `IsotopeLabelType`, `Condition`, `BioReplicate`, `Run`, `Intensity`. The variable names are fixed, but are case-insensitive.

(a) `ProteinName`: This column needs information about Protein id. Statistical analysis will be done separately for each unique label in this column. For peptide-level modeling and analysis, use peptide id in this column.

(b)-(e) `PeptideSequence, PrecursorCharge, FragmentIon, ProductCharge`: The combination of these 4 columns defines a *feature* of a protein (in SRM experiments, it is a transition that is identified and quantified across runs). If the information for one or several of these columns is not available, please do not discard these columns but use a single fixed value across the entire dataset. For example, if the original raw data does not contain the information of `ProductCharge`, assign the value 0 to the entries in the column `ProductCharge` for the entire dataset. If the peptide sequences should be distinguished based on post-translational modifications, this column can be renamed to `PeptideModifiedSequence`. For example, this allows us to use the `PeptideModifiedSequence` column from the Skyline report.

(f) `IsotopeLabelType`: This column indicates whether this measurement is based on the endogenous peptides (use "L") or labeled reference peptides (use "H").

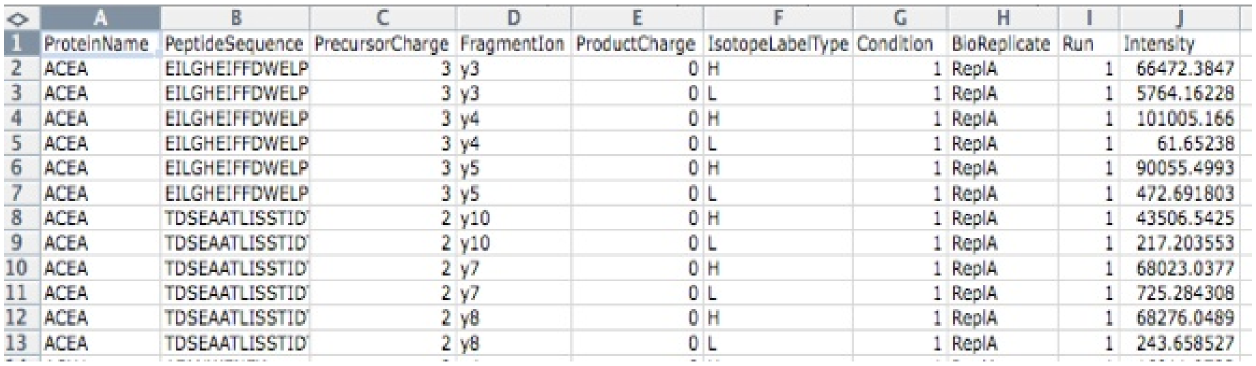
(g) `Condition`: For group comparison experiments, this column indicates groups of interest (such as "Disease" or "Control"). For time-course experiments, this column indicates time points (such as "T1", "T2", etc). If the experimental design contains both distinct groups of subjects and multiple time points per subject, this column should indicate a combination of these values (such as "Disease_T1", "Disease_T2", "Control_T1", "Control_T2", etc.).

(h) `BioReplicate`: This column should contain a unique identifier for each biological replicate in the experiment. For example, in a clinical proteomic investigation this should be a unique patient id. Patients from distinct groups should have distinct ids. `MSstats` does not require the presence of technical replicates in the experiment. If the technical replicates are present, all samples or runs from a same biological replicate should have a same id. `MSstats` automatically detects the presence of technical replicates and accounts for them in the model-based analysis.

(i) `Run`: This column contains the identifier of a mass spectrometry run. Each mass spectrometry run should have a unique identifier, regardless of the origin of the biological sample. In SRM experiments, if all the transitions of a biological or a technical replicate are split into multiple "methods" due to the technical limitations, each method should have a separate identifier. When processed by Skyline, distinct values of runs correspond to distinct input file names. It is possible to use the actual input file names as values in the column `Run`.

(j) `Intensity`: This column should contain the quantified signal of a feature in a run without any transformation (in particular, no logarithm transform). The signals can be quantified as the peak height or the peak of area under curve. Any other quantitative representation of abundance can also be used.

An example of an acceptable input dataset is shown below. This example dataset is from an SRM experiment with stable isotope labeled reference peptides. The dataset is stored in a .csv file in a "long" format. Each row corresponds to a single intensity. More details on assigning the values of `Condition, BioReplicate` and `Run`, depending on the structure of the experimental design, are given below.



◇	A	B	C	D	E	F	G	H	I	J
1	ProteinName	PeptideSequence	PrecursorCharge	FragmentIon	ProductCharge	IsotopeLabelType	Condition	BioReplicate	Run	Intensity
2	ACEA	EILGHEIFFDWELP	3	y3	0	H	1	ReplA	1	66472.3847
3	ACEA	EILGHEIFFDWELP	3	y3	0	L	1	ReplA	1	5764.16228
4	ACEA	EILGHEIFFDWELP	3	y4	0	H	1	ReplA	1	101005.166
5	ACEA	EILGHEIFFDWELP	3	y4	0	L	1	ReplA	1	61.65238
6	ACEA	EILGHEIFFDWELP	3	y5	0	H	1	ReplA	1	90055.4993
7	ACEA	EILGHEIFFDWELP	3	y5	0	L	1	ReplA	1	472.691803
8	ACEA	TDSEAATLISSTID	2	y10	0	H	1	ReplA	1	43506.5425
9	ACEA	TDSEAATLISSTID	2	y10	0	L	1	ReplA	1	217.203553
10	ACEA	TDSEAATLISSTID	2	y7	0	H	1	ReplA	1	68023.0377
11	ACEA	TDSEAATLISSTID	2	y7	0	L	1	ReplA	1	725.284308
12	ACEA	TDSEAATLISSTID	2	y8	0	H	1	ReplA	1	68276.0489
13	ACEA	TDSEAATLISSTID	2	y8	0	L	1	ReplA	1	243.658527

2.1.2 Assigning the values of `Condition, BioReplicate` and `Run`

The values of `Condition, BioReplicate, Run` depend on the design of the specific experiment.

1) Group comparison In a group comparison design, the conditions (e.g., disease states) are profiled across **non-overlapping sets of biological replicates** (i.e., subjects). In this example there are 2 conditions, Disease and Control (in general the number of conditions can vary). There are 3 subjects (i.e., biological replicates) per condition (in general an equal number of replicates per condition is not required). Each subject has 2 technical replicate runs (in general technical replicates are not required, and their number per sample may vary). Overall, in this example there are $2 \times 3 \times 2 = 12$ mass spectrometry runs.

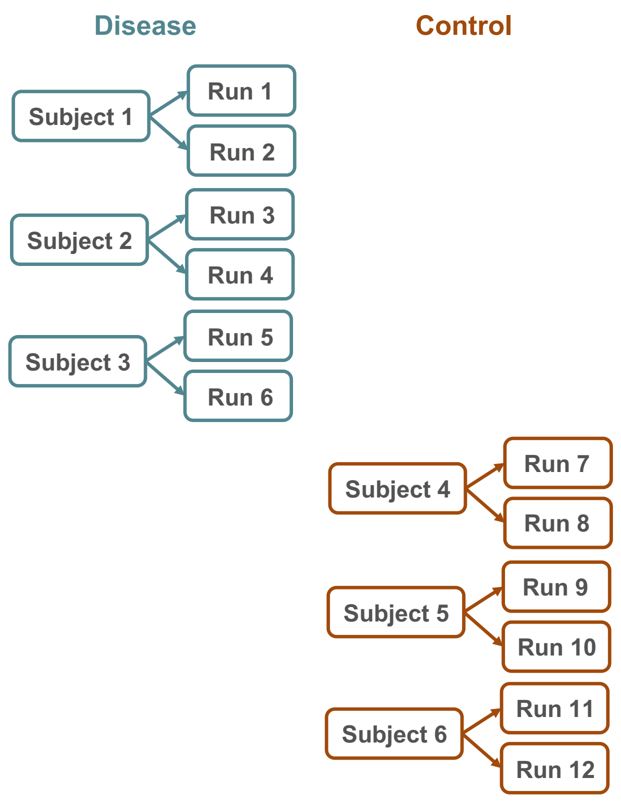


Table below shows the values of the columns `Condition, BioReplicate` and `Run` for this situation. It is important to note two things. First, the order of subjects and conditions in the experiment should be randomized, and run id does not need to represent the order of spectral acquisition. Second, the values of the columns are repeated for every quantified transition. For example, if in each run the experiment quantifies 50 endogenous transitions and 50 labeled reference counterparts, then the input file has $12 \times 50 \times 2 = 1200$ lines. When a feature intensity is missing in a run, the data structure should contain a separate row for each missing value. The rows should include all the information (from `ProteinName` to `Run`), and indicate missing intensities with `NA`.

`Condition`	`BioReplicate`	`Run`
Disease	Subject1	1
Disease	Subject1	2
Disease	Subject2	3
Disease	Subject2	4
Disease	Subject3	5
Disease	Subject3	6
Control	Subject4	7
Control	Subject4	8
Control	Subject5	9
Control	Subject5	10
Control	Subject6	11
Control	Subject6	12

2) Time course The important feature of a time course experimental design is that **a same subject (i.e., biological replicate) is repetitively measured across multiple time points**. In this example there are 2 time points, Time1 and Time2 (in general the number of times can vary). There are 4 subjects (i.e., biological replicates) measured across times (in general an equal number of times per replicate is not required). There are no technical replicates (in general the number of technical replicates per sample may vary). Overall, in this example there are $2 \times 4 \times 1 = 8$ mass spectrometry runs.

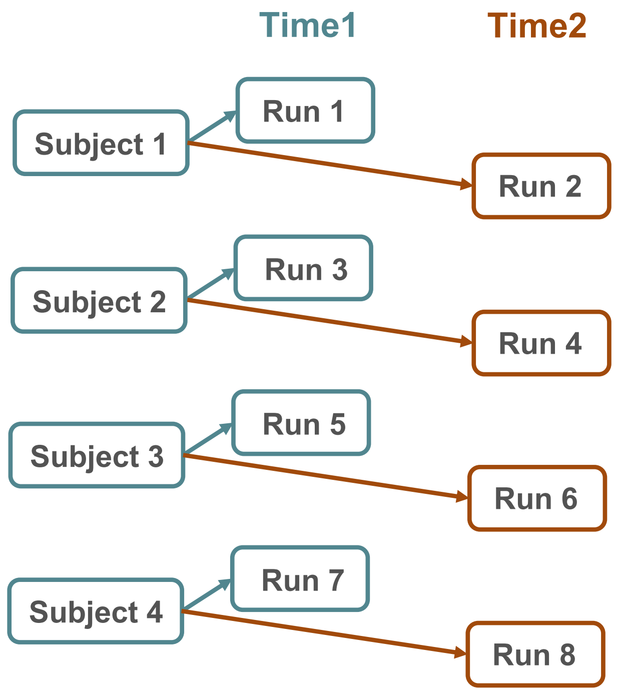


Table below shows the values of the columns `Condition, BioReplicate` and `Run` for this situation. Comments on the order of the runs, on the number of lines in the input data structure, and on the handling of missing peak intensities are as in the group comparison design.

`Condition`	`BioReplicate`	`Run`
Time1	Subject1	1
Time2	Subject1	2
Time1	Subject2	3
Time2	Subject2	4
Time1	Subject3	5
Time2	Subject3	6
Time1	Subject4	7
Time2	Subject4	8

3) Paired design Another frequently used experimental design is a *paired design*, where measurements from multiple conditions (such as healthy biopsy and disease biopsy) are taken from a same subject. The statistical model for this experimental design is the same as in the time course experiment, however the values in the columns of the input data may have a different appearence. In this example there are 2 subjects, PatientA and PatientB (in general the number of patients can vary). There are two conditions per subject, BiopsyHealthy and BiopsyTumor (in general the number of conditions per subject can exceed two). In this example there are 3 technical replicates of each type (in this example, the technical replicates are biopsies; in general these can also be replicate sample preparations or replicate mass spectrometry runs). Overall, in this example there are $2 \times 2 \times 3 = 12$ mass spectrometry runs.

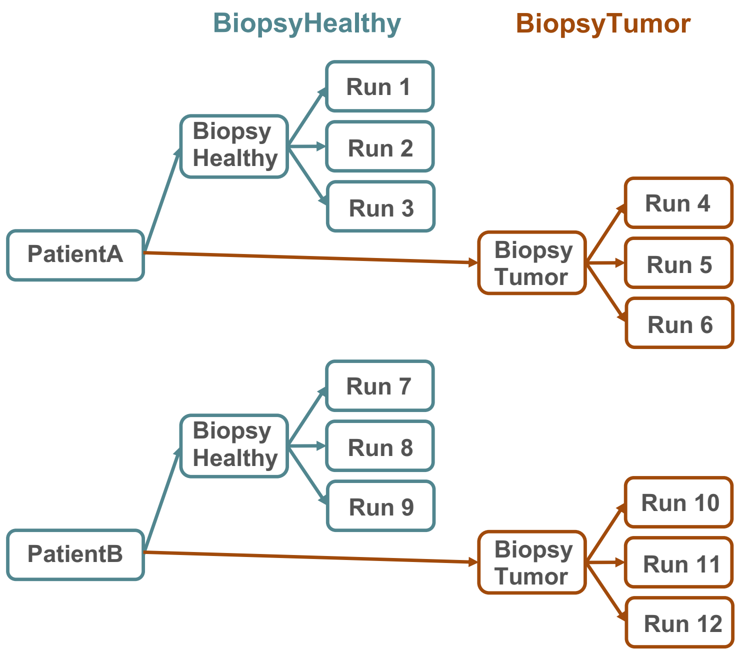
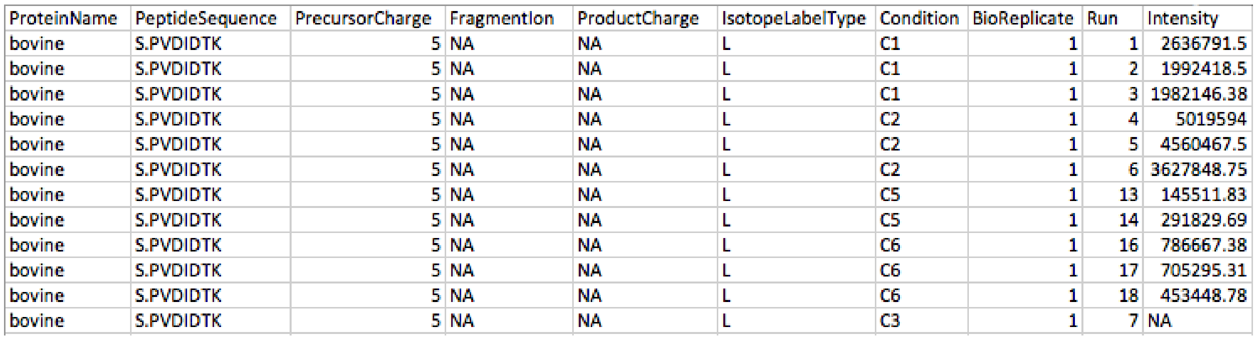


Table below shows the values of the columns `Condition, BioReplicate` and `Run` for this situation. Comments on the order of the runs, on the number of lines in the input data structure, and on the handling of missing peak intensities are as in the group comparison design.

`Condition`	`BioReplicate`	`Run`
BiopsyHealthy	PatientA	1
BiopsyHealthy	PatientA	2
BiopsyHealthy	PatientA	3
BiopsyTumor	PatientA	4
BiopsyTumor	PatientA	5
BiopsyTumor	PatientA	6
BiopsyHealthy	PatientB	7
BiopsyHealthy	PatientB	8
BiopsyHealthy	PatientB	9
BiopsyTumor	PatientB	10
BiopsyTumor	PatientB	11
BiopsyTumor	PatientB	12

2.2 Label-free DDA

For label-free DDA experiments the required input is the 10-column format, the same as described in section 2.1 for SRM experiments. In DDA experiments spectral features are defined as peptide ions, which are identified and quantified across runs. Since for label-free DDA experiments some of the columns `PeptideSequence`, `PrecursorCharge`, `FragmentIon`, and `ProductCharge` are not relevant, these columns will have a constant fixed value (such as `NA`) across the entire dataset. Furthermore, the column `IsotopeLabelType` will be set to "L" for the entire dataset.

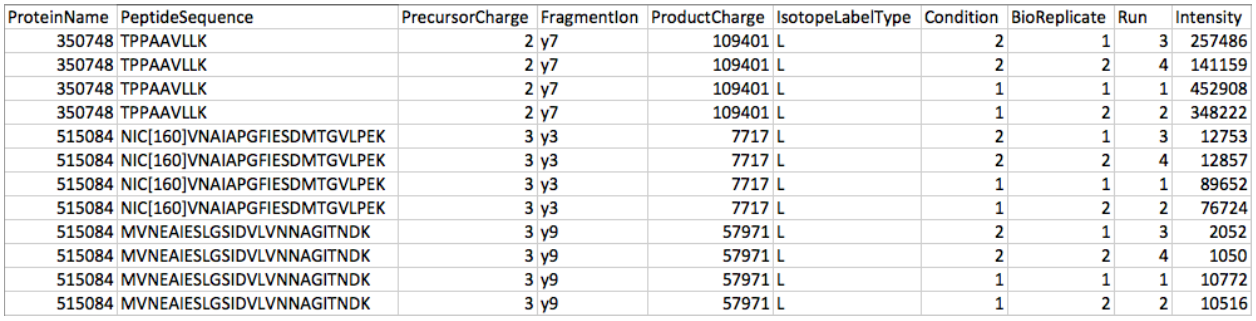


ProteinName	PeptideSequence	PrecursorCharge	FragmentIon	ProductCharge	IsotopeLabelType	Condition	BioReplicate	Run	Intensity
bovine	S.PVDIDTK	5	NA	NA	L	C1	1	1	2636791.5
bovine	S.PVDIDTK	5	NA	NA	L	C1	1	2	1992418.5
bovine	S.PVDIDTK	5	NA	NA	L	C1	1	3	1982146.38
bovine	S.PVDIDTK	5	NA	NA	L	C2	1	4	5019594
bovine	S.PVDIDTK	5	NA	NA	L	C2	1	5	4560467.5
bovine	S.PVDIDTK	5	NA	NA	L	C2	1	6	3627848.75
bovine	S.PVDIDTK	5	NA	NA	L	C5	1	13	145511.83
bovine	S.PVDIDTK	5	NA	NA	L	C5	1	14	291829.69
bovine	S.PVDIDTK	5	NA	NA	L	C6	1	16	786667.38
bovine	S.PVDIDTK	5	NA	NA	L	C6	1	17	705295.31
bovine	S.PVDIDTK	5	NA	NA	L	C6	1	18	453448.78
bovine	S.PVDIDTK	5	NA	NA	L	C3	1	7	NA

2.2 Label-free DIA

For label-free DIA experiments, the required input is the 10-column format, the same as described in section 2.1 for SRM experiments. The values of the required columns can be extracted from the output of signal processing software such as Skyline or OpenSWATH. By default, the combination of the values in the columns `PeptideSequence, PrecursorCharge, FragmentIon, ProductCharge` uniquely identifies each spectral feature (i.e., a fragment ion identified and quantified across multiple runs). If the signal processing software does not provide the information on some of these columns but provides a unique feature identifier, it is possible to use this unique identifier instead of one of these columns. Furthermore, the column `IsotopeLabelType` is set to "L" for the entire dataset.

An example dataset is shown below. In this example, feature id generated by OpenSWATH is used instead of `ProductCharge` to uniquely characterize each feature.



ProteinName	PeptideSequence	PrecursorCharge	FragmentIon	ProductCharge	IsotopeLabelType	Condition	BioReplicate	Run	Intensity
350748	TPPAAVLLK	2	y7	109401	L	2	1	3	257486
350748	TPPAAVLLK	2	y7	109401	L	2	2	4	141159
350748	TPPAAVLLK	2	y7	109401	L	1	1	1	452908
350748	TPPAAVLLK	2	y7	109401	L	1	2	2	348222
515084	NIC[160]VNAIAPGFIESDMTGVLPEK	3	y3	7717	L	2	1	3	12753
515084	NIC[160]VNAIAPGFIESDMTGVLPEK	3	y3	7717	L	2	2	4	12857
515084	NIC[160]VNAIAPGFIESDMTGVLPEK	3	y3	7717	L	1	1	1	89652
515084	NIC[160]VNAIAPGFIESDMTGVLPEK	3	y3	7717	L	1	2	2	76724
515084	MVNEAIESLGSIDVLVNNAGITNDK	3	y9	57971	L	2	1	3	2052
515084	MVNEAIESLGSIDVLVNNAGITNDK	3	y9	57971	L	2	2	4	1050
515084	MVNEAIESLGSIDVLVNNAGITNDK	3	y9	57971	L	1	1	1	10772
515084	MVNEAIESLGSIDVLVNNAGITNDK	3	y9	57971	L	1	2	2	10516

---

3. Prerequisites and setting for MSstats analysis

`MSstats` is an R-based package. It is assumed that you already have R installed. You can install `MSstats` from Bioconductor:

```r
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("MSstats")
```

Once you have the package installed, load `MSstats` into an R session and verify that you have the correct version. Note that in order to use `MSstats`, the package needs to be loaded every time you restart R.

```r
library('MSstats', warn.conflicts = F, quietly = T, verbose = F)
```

```
## Warning: package 'MSstats' was built under R version 4.2.2
```

```
?MSstats
```

```
## No documentation for 'MSstats' in specified packages and libraries:
## you could try '??MSstats'
```

```
sessionInfo()
```

```
## R version 4.2.1 (2022-06-23)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur ... 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] MSstats_4.6.3
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.10          log4r_0.4.3          nloptr_2.0.3
##  [4] pillar_1.8.1         compiler_4.2.1       bitops_1.0-7
##  [7] tools_4.2.1          boot_1.3-28.1        digest_0.6.31
## [10] lme4_1.1-31          checkmate_2.1.0      preprocessCore_1.60.2
## [13] evaluate_0.20        lifecycle_1.0.3      tibble_3.1.8
## [16] gtable_0.3.1         nlme_3.1-161         lattice_0.20-45
## [19] pkgconfig_2.0.3      rlang_1.0.6          Matrix_1.5-3
## [22] cli_3.6.0            rstudioapi_0.14      ggrepel_0.9.2
## [25] yaml_2.3.7           xfun_0.36            fastmap_1.1.0
## [28] dplyr_1.1.0          knitr_1.42           caTools_1.18.2
## [31] gtools_3.9.4         generics_0.1.3       vctrs_0.5.2
## [34] grid_4.2.1           tidyselect_1.2.0     glue_1.6.2
## [37] data.table_1.14.6    R6_2.5.1             marray_1.76.0
## [40] fansi_1.0.4          survival_3.5-0       rmarkdown_2.20
## [43] minqa_1.2.5          limma_3.54.1         ggplot2_3.4.0
## [46] magrittr_2.0.3       backports_1.4.1      gplots_3.1.3
## [49] scales_1.2.1         htmltools_0.5.4      splines_4.2.1
## [52] MASS_7.3-58.2        MSstatsConvert_1.8.2 colorspace_2.1-0
## [55] KernSmooth_2.23-20   utf8_1.2.2           munsell_0.5.0
```

Finally, set the working directory to where you saved files. Note that you may have a different path on your computer from the example.

```
setwd('MSstats_v4.6.3')
```

You can check your working directory by:

```
getwd()
```

```
## [1] "/Users/kohler.d/Downloads/User_guide"
```

---

4. DDA analysis with MSstats

4.1 General workflow for DDA

This section describes a typical workflow for DDA analysis with `MSstats`. Controlled mixture DDA data will be used for demonstration. This dataset is available as an example data (`DDARawData`) in MSstats. Also the csv file for the same dataset, RawData.DDA.csv, is available in MSstats material GitHub in the folder named 'example dataset/DDA_controlledMixture2009'. It is processed by Superhirn. (original reference link)

4.1.1 Preparing the data for MSstats input

The first step in using the `MSstats` is to format the data as described in Section 2. `DDARawData` is already formatted for `MSstats` input.

```
# Check the first 6 rows in DDARawData
head(DDARawData)
```

```
##   ProteinName PeptideSequence PrecursorCharge FragmentIon ProductCharge
## 1      bovine     S.PVDIDTK_5               5          NA            NA
## 2      bovine     S.PVDIDTK_5               5          NA            NA
## 3      bovine     S.PVDIDTK_5               5          NA            NA
## 4      bovine     S.PVDIDTK_5               5          NA            NA
## 5      bovine     S.PVDIDTK_5               5          NA            NA
## 6      bovine     S.PVDIDTK_5               5          NA            NA
##   IsotopeLabelType Condition BioReplicate Run Intensity
## 1                L        C1            1   1   2636792
## 2                L        C1            1   2   1992418
## 3                L        C1            1   3   1982146
## 4                L        C2            1   4   5019594
## 5                L        C2            1   5   4560468
## 6                L        C2            1   6   3627849
```

4.1.2 Processing the data

Normalizing and summarizing data with dataProcess After reading the datasets, `MSstats` performs 1) logarithm transformation of `Intensity` column, 2) normalization, 3) feature selection, (all features vs subset of features), 4) imputation for censored missing value, which are below the cutoff and undetectable, 5) run-level summarization.

To get started with this function, visit the help section of `dataProcess` first:

```
?dataProcess
```

NOTE At the logarithm transformation step, zero value in `Intensity` is problematic. When `Intensity=0`, `Inf` is the output from logarithm transformed intensities. Also, logarithm transformed intensites, when `Intensity < 1`, are negative values and it can make overestimated between log fold change. Therefore, logarithm transformed intensities for original intensity between 0 and 1 will be replaced with zero value after normalization.

Default normalization and summarization options `dataProcess` provides a variety of options in consideration of different experimental protocols. Default values for all options are our suggestion for general cases. However, the default options may not be appropriate for all possible scenarios. It is important to understand their underlying assumption to avoid misuse. Below is the additional explanation for main options.

- **logTrans** : logarithm transformation with base 2(default) of `Intensity` column.
- **Normalization** :

  - **'equalizeMedians'** : The default option for normalization is `equalizeMedians`, where all the intensities in a run are shifted by a constant, to equalize the median of intensities across runs for label-free experiment. This normalization method is appropriate when we can assume that the majority of proteins do not change across runs. *Be cautious when using the `equalizeMedians` option for a label-free DDA dataset with only a small number of proteins.* For label based experiment, `equalizeMedians` equalizes the median of reference intensities across runs and is generally proper even for a dataset with a small number of proteins.
  - **'globalStandards'** : Instead, if you have a spiked in standard, you may set this to `globalStandards` and define the standard with `nameStandards` option.
  - **'quantile'** : The distribution of all the intensities in each run will become the same across runs for label-free experiment. For label-based experiment, the distribution of all the reference intensities will be become the same across runs and all the endogenous intensities are shifted by a constant corresponding to reference intensities.
  - **FALSE** : No normalization is performed. If you had your own normalization before MSstats, you should use `Normalization=FALSE`.
  - NOTE : If there are multiple fractionations or injections for one sample, normalization is perform by each fractionation or different m/z range from multiple injections.

- **nameStandards** : Only for `Normalization='globalStandards'`, global standard peptide or Protein names, which you can assume that they have the same abundance across MS runs, should be assigned in the vector for this option.

- **featureSubset** :
  - **'all'** : Use all features in the dataset.
  - **'top3'** : Use top 3 features which have highest average of log2(intensity) across runs.
  - **'topN'** : Use top N features which have highest average of log2(intensity) across runs. It needs the input for `n_top_feature`option (ex. `n_top_feature=5` for top 5 features).

- **summaryMethod** : Method for run-level summarization.
  - **'TMP'** : Default. Tukey's median polish (`medpolish` function in stats). Robust parameter estimation method with median across rows and columns.
  - **'linear'** : Linear model (`lm` function). Average-based summarization.

- **MBimpute** : whether model-based imputation will be performed or not. Only for `summaryMethod='TMP'`.
  - **TRUE** : Default. Censored missing values will be imputed by Accelerated Failure Time model. Censored missing values will be determined by other options, `censoredInt` and `maxQuantileforCensored`
  - **FALSE** : No model-based imputation.

- **maxQuantileforCensored** : Maximum quantile for deciding censored missing value. Default is 0.999. If you don't want to apply the threshold of noise intensity in your data, you can use `maxQuantileforCensored=NULL`.

- **censoredInt** : The processing tools report missing values differently. This option is for distinguish which value should be considered as missing, and further whether it is censored or at random.
  - **'NA'** : Default. It assumes that all NAs in `Intensity` column are censored.
  - **'0'** : It assumes that all values between 0 and 1 in `Intensity` column are censored. If there are NAs in `Intensity` with this option, NAs will be considered as random missing.
  - **NULL** : It assumes that all missing values are randomly missing.

- **cutoffCensored** : cutoff value for AFT model. It is only with `censoredInt='NA'` or `censoredInt='0'`. If you have `censoredInt=NULL`, it assumes that there is no censored missing and any imputation will not be performed.
  - **'minFeature'** : cutoff for AFT model will be the minimum value for each feature across runs.
  - **'minRun'** : cutoff for AFT model will be the minimum value for each run across features.

- **'minFeatureNRun'** : cutoff for AFT model will be the smallest value between minimum value of corresponding feature and minimum value of corresponding run.

A typical label-free DDA dataset may have many missing values and noisy features with outliers. `MSstats` supports several ways to deal with this. The default option for summarization is `TMP` (robust parameter estimation method with median across rows and columns) after imputation by AFT (accelerated failure time model, `MBimpute=TRUE`) based on censored intensity for `NA` (`censoredInt="NA"`) with a cutoff as the minimum value for a feature (`cutoffCensored="minFeature"`).

This process handles missing values through imputation and reduces the influence of the outliers using the `TMP` estimation. Note, however, that those runs with no measurements at all will be removed and not be used for any calculation.

```
# default option
DDA2009.proposed <- dataProcess(raw = DDARawData,
                                normalization = 'equalizeMedians',
                                summaryMethod = 'TMP',
                                censoredInt = "NA",
                                MBimpute = TRUE,
                                maxQuantileforCensored=0.999)
```

```
## INFO  [2023-02-28 10:22:34] ** Features with one or two measurements across runs are removed.
## INFO  [2023-02-28 10:22:34] ** Fractionation handled.
## INFO  [2023-02-28 10:22:34] ** Updated quantification data to make balanced design. Missing values a
## INFO  [2023-02-28 10:22:35] ** Log2 intensities under cutoff = 13.456  were considered as censored m
## INFO  [2023-02-28 10:22:35] ** Log2 intensities = NA were considered as censored missing values.
## INFO  [2023-02-28 10:22:35] ** Use all features that the dataset originally has.
## INFO  [2023-02-28 10:22:35]
##  # proteins: 6
##  # peptides per protein: 11-32
##  # features per peptide: 1-1
## INFO  [2023-02-28 10:22:35]
##                     C1 C2 C3 C4 C5 C6
##              # runs  3  3  3  3  3  3
##     # bioreplicates  1  1  1  1  1  1
##  # tech. replicates  3  3  3  3  3  3
## INFO  [2023-02-28 10:22:35] Some features are completely missing in at least one condition:
##  D.GPLTGTYR_23_23_NA_NA,
##  F.HFHWGSSDDQGSEHTVDR_402_402_NA_NA,
##  G.PLTGTYR_8_8_NA_NA,
##  H.SFNVEYDDSQDK_465_465_NA_NA,
##  K.AVVQDPALKPL_156_156_NA_NA ...
## INFO  [2023-02-28 10:22:35]  == Start the summarization per subplot...
##   |                                                                              |
## INFO  [2023-02-28 10:22:35]  == Summarization is done.
```

Output of `dataProcess` Output of the `dataProcess` function contains the processed and run-level summarized data as well as relevant information for the summarization step.

```
# output of dataProcess includes several data types.
names(DDA2009.proposed)
```

```
## [1] "FeatureLevelData" "ProteinLevelData" "SummaryMethod"
```

```
# the data after reformatting and normalization
head(DDA2009.proposed$ProcessedData)
```

```
## NULL
```

`DDA2009.TMP$ProcessedData` has the data after normalization and deciding the data-specific threshold for censored missing value. There are several new columns in the datasets. Also dataset is reformated. `Intensity` column includes original intensities values in the input of `dataProcess`. `ABUNDANCE` column contains the *log2 transformed and normalized intensities and it will used for run-level summarization.* `censored` column has the decision about censored missing or not, based on `censoredInt` and `maxQuantileforCensored` options. `ABUNDANCE` with `TRUE` value in `censored` column will be considered as censored missing and imputed with `MBimpute=TRUE` option. Censored missing will be distinguished in Profile plot from `dataProcessPlots`.

```
# run-level summarized data
head(DDA2009.proposed$RunlevelData)
```

```
## NULL
```

`DDA2009.TMP$RunlevelData` includes run-level summarized data based on `DDA2009.TMP$ProcessedData`. `LogIntensities` is run-level summarized data and will be used for `groupComparison` function in next step. It will also used for summarized profile plot (`summaryPlot=TRUE` for `dataProcessPlots` function with `type='ProfilePlot'`). `NumMeasuredFeature` shows how many features were used for summarization of the corresponding run and protein. `MissingPercentage` means the percentage of random and censored missing in the corresponding run and protein out of the total number of feature in the corresponding protein. `more50missing` means whether `MissingPercentage` is greater than 50% or not. `NumImputedFeature` show how many features were imputed in the corresponding run and protein.

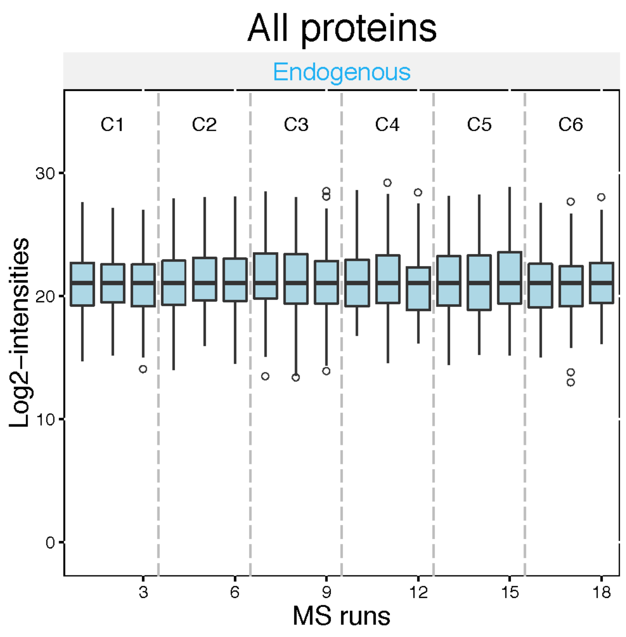
```
# here 'TMP' : It shows which summaryMethod is used for run-level summarization.
head(DDA2009.proposed$SummaryMethod)
```

```
## [1] "TMP"
```

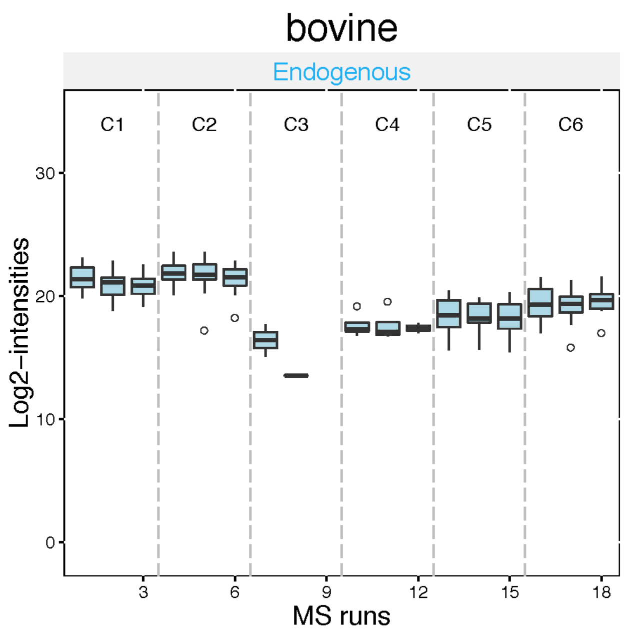
4.1.3 Visualization of processed data

Quality control and normalization effects QC plot visualizes potential systematic biases between mass spectrometry runs. Also it can be used to assess the effects of the normalization step. After constant normalization, the median intensities of reference transitions across all proteins should be equal between runs. After quantile normalization, the distribution of reference intensities across all proteins should be equal between runs. This step generates two types of QC plots: one for all the proteins combined, and the other separately for each protein (produced in a separate pdf file). These plots can be generated for either all proteins at once or each protein individually if we have a large dataset. The example below shows both options.

```
# use type="QCplot" with all proteins
# change the upper limit of y-axis=35
# set up the size of pdf
dataProcessPlots(data = DDA2009.proposed, type="QCplot", ylimUp=35,
                 width=5, height=5)
```



```
# use type="QCplot" for 1st protein only
# change the upper limit of y-axis=35
# set up the size of pdf
dataProcessPlots(data = DDA2009.proposed, type="QCplot", which.Protein=1,
                 ylimUp=35, width=5, height=5)
```

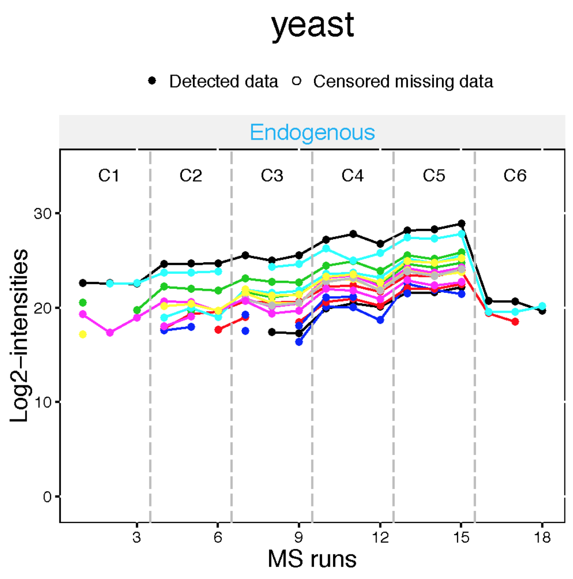


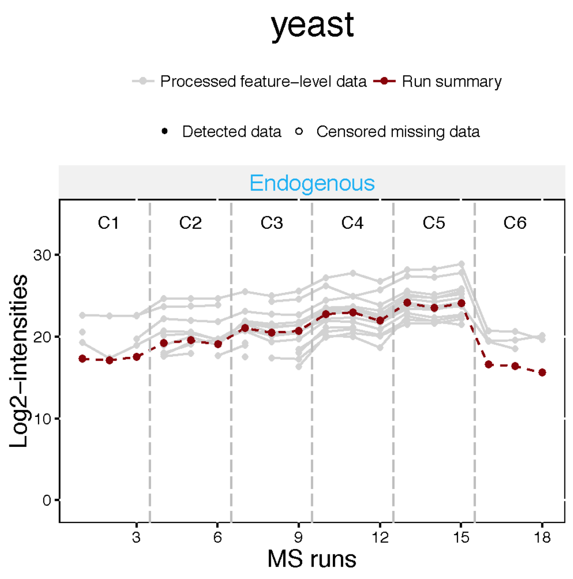
NOTE Don't worry about warning messages as below. It means `NA` values are not included in the plot, which is a proper way for this case.

```
Warning messages:
1: Removed 698 rows containing non-finite values
(stat_boxplot).
...
```

Profile plot Profile plot helps identify potential sources of variation (both variation of interest and nuisance variation) for each protein. Such plots should be done after the normalization. Profile plots with summarization present the effects of the summarization step by showing all individual measurements of a protein and their summarized intensity. With `type="profileplot"`, two pdfs will be generated. The first pdf includes plots (per protein) to show individual measurement for each peptide (peptide for DDA, transition for SRM or DIA) across runs, grouped per condition. Each peptide has a different color/type layout. Disconnected lines show that there are missing value (NA). To ignore these plots, please use the option `originalPlot=FALSE`. The second pdf, which is named with 'wSummarization' suffix, shows run-level summarized data per protein. The same peptides (or transition) in the first plot are presented in grey, with the summarized values (by TMP, in this example) overlaid in red. To ignore these plots with summarization, please use the option `summaryPlot=FALSE`.

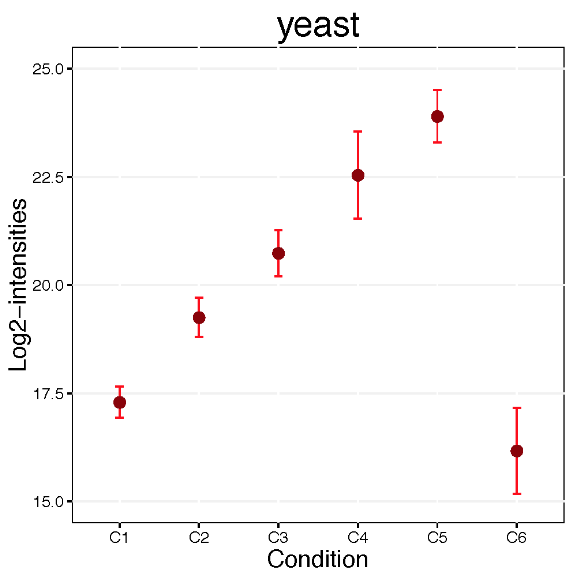
```
dataProcessPlots(data = DDA2009.proposed, type="Profileplot",  ylimUp=35,
                 featureName="NA", width=5, height=5, address="DDA2009_proposed_")
```





Condition plot Condition plot visualizes potential systematic diffrences in protein inensities between conditions. Dots indicate the mean of log2 intensities for each condition. With the option `interval='CI'`(default), error bars indicate the confidence interval with 0.95 significant level for each condition. With the option `interval='SD'`, error bars indicate the standard deviation among all feature intensities for each condition. **The intervals are for descriptive purposes only, as more refined model-based estimation is obtained as discussed below.** With the option `scale=TRUE`, the levels of conditions are scaled according to their labels. If `scale=FALSE` (default), the conditions on the x-axis are equally spaced.

```
dataProcessPlots(data = DDA2009.proposed, type="Conditionplot",
                 width=5, height=5, address="DDA2009_proposed_")
```



`dataProcessPlots` has a number of layout options, including size and description of axes labels, output file name etc for three types of plots above. The option `address` specifies the name of the folder storing pdf files with the plots. With the option `address=FALSE`, plots will be shown in the graphical window, but not saved in a file. If a file with this name already exists in working directory, a suffix with a number will be appended to the file name. In this way a record of all the analyses is kept.

For more details, visit the help file using the following code.

```
?dataProcessPlots
```

4.1.4 Different imputation options

Here is **the summary of combinations for imputation options** with `summaryMethod='TMP'`.

- `censoredInt=NULL` : It assumes that all intensities are missing at random and there is no action for missing values with `MBimpute=FALSE`. If you have `MBimpute=TRUE` with `censoredInt=NULL`, there will be error message to fix either `MBimpute` or `censoredInt` options.
- `censoredInt='NA'` or `'0'` & `MBimpute=TRUE` : AFT model-based imputation using `cutoffCensored` value in the AFT model.
- `censoredInt='NA'` or `'0'` & `MBimpute=FALSE` : censored intensities (here `NA`'s) will be replaced with the value specified in `cutoffCensored`.

NOTE1 The default option for `cutoffCensored` is `minFeature`, taking the minimum value for the corresponding feature. With this option, those runs with substantial missing measurements will be biased by the cutoff value. In such case, you may remove the runs that have more than 50% missing values from the analysis with the option `remove50missing=TRUE`.

NOTE2 In case that there are completely missing measurements in a run for a protein, any imputation will not be performed. In addition, the condition, which has no measurement at all in a protein, will be not imputed.

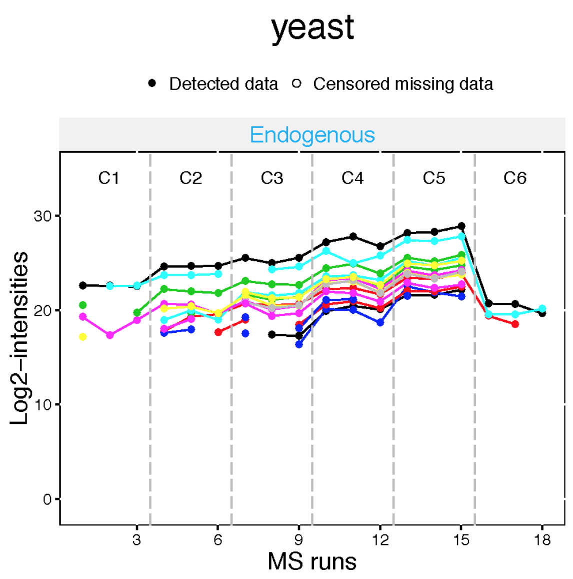
Here is the example of `dataProcess` option without imputation, assuming that all missing values are random.

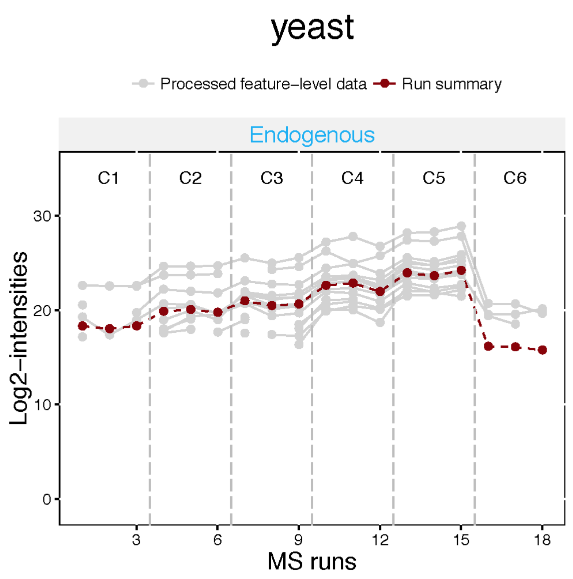
```
# No imputation
DDA2009.TMP <- dataProcess(raw = DDARawData,
                                normalization = 'equalizeMedians',
                                summaryMethod = 'TMP',
                                censoredInt = NULL, MBimpute=FALSE)
```

```
## INFO  [2023-02-28 10:22:39] ** Features with one or two measurements across runs are removed.
## INFO  [2023-02-28 10:22:39] ** Fractionation handled.
## INFO  [2023-02-28 10:22:39] ** Updated quantification data to make balanced design. Missing values a
## INFO  [2023-02-28 10:22:39] ** Use all features that the dataset originally has.
## INFO  [2023-02-28 10:22:39]
##  # proteins: 6
##  # peptides per protein: 11-32
##  # features per peptide: 1-1
## INFO  [2023-02-28 10:22:39]
##                     C1 C2 C3 C4 C5 C6
##              # runs  3  3  3  3  3  3
##       # biorepliates  1  1  1  1  1  1
##  # tech. replicates  3  3  3  3  3  3
## INFO  [2023-02-28 10:22:39] Some features are completely missing in at least one condition:
##  D.GPLTGTYR_23_23_NA_NA,
##  F.HFHWGSSDDQGSEHTVDR_402_402_NA_NA,
##  G.PLTGTYR_8_8_NA_NA,
##  H.SFNVEYDDSQDK_465_465_NA_NA,
##  K.AVVQDPALKPL_156_156_NA_NA ...
## INFO  [2023-02-28 10:22:39]  == Start the summarization per subplot...
##   |                                                                                  |
## INFO  [2023-02-28 10:22:39]  == Summarization is done.
```

These plots can be used compare and select among different options for imputation (e.g., TMP with or without considering missing values for summarization in dataProcess).

```
dataProcessPlots(data = DDA2009.TMP, type="Profileplot",  ylimUp=35,
                 featureName="NA", width=5, height=5, address="DDA2009_TMP_")
```





While original profile plots are the same, summarization plots reveal differences, especially for conditions ‘C1’ and ‘C2’ in ‘yeast’ protein, which have many missing values. Without imputation, summarized values in ‘C1’ group is higher than with imputation for missing values.

4.1.5 Different summarization options

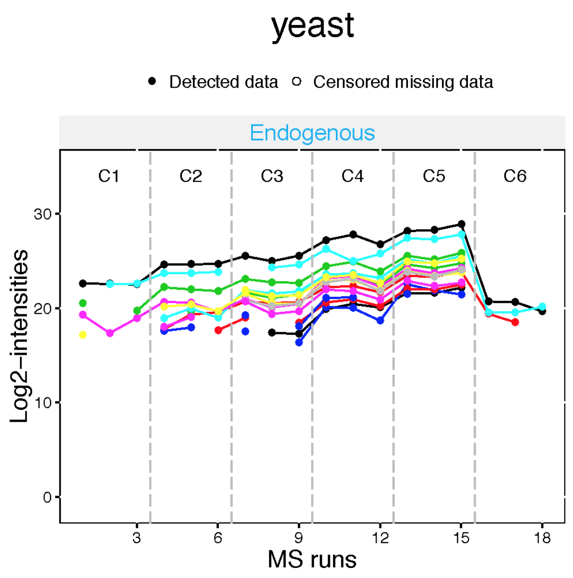
Besides summarizing observations with the median polish method, MSstats also offers a summarization option using linear model with option summaryMethod="linear" with censoredInt=NULL assumes that all

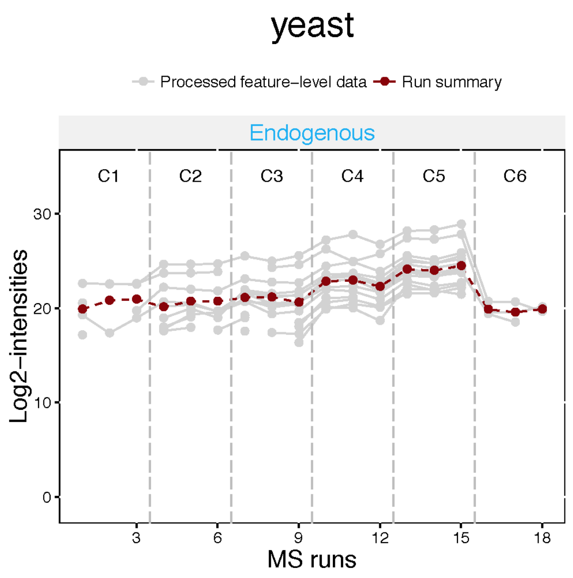
NA's are missing at random and uses lm for parameter estimation.

```
# linear model (lm) with run and feature
DDA2009.linear <- dataProcess(raw = DDARawData,
                              normalization = 'equalizeMedians',
                              summaryMethod = 'linear',
                              censoredInt = NULL,
                              MBimpute = FALSE)
```

Profile plots below can be used compare among different options for summarization (e.g., TMP with or without imputation vs linear for summarization in dataProcess).

```
dataProcessPlots(data = DDA2009.linear, type="Profileplot",  ylimUp=35,
                 featureName="NA", width=5, height=5, address="DDA2009_linear_")
```





While original profile plots are the same, summarization plots reveal differences, especially for conditions 'C1', 'C2', and 'C6' in 'yeast' protein, which have many missing values. Summarized values with linear model in these groups are much higher than those with TMP considering missing values or not.

4.1.6 Finding differentially abundant proteins across conditions

Comparing conditions with groupComparison With the normalized data and run-level summarized data obtained by applying one of the dataProcess summarization methods, it is of general interest to find proteins changing between groups of conditions. Within MSstats this can be done by using the groupComparison function, which takes the output of the dataProcess function as input.

```
?groupComparison
```

In addition to the processed data, the groupComparison function requires a contrast matrix to define the comparison to be made. The contrast matrix is created with each condition in column and each comparison in row. Note that the conditions are arranged **in alphabetical order**. The order of condition that MSstats recognizes can be shown by using levels:

```
levels(DDA2009.TMP$ProcessedData$GROUP_ORIGINAL)
```

```
## NULL
```

Entries in each row of the contrast matrix are filled in with 0, 1, or -1 to specify the comparison, where **0** is for conditions we would like to ignore, **1** is for conditions we would like to put in the numerator of the ratio or fold-change, and **-1** is for conditions we would like to put in the denumerator of the ratio or fold-change.

For example, if you want to compare C2-C1, which means log(C2)-log(C1) and the same as log(C2/C1), set '1' for C2 and '-1' for C1 in the row. Combining multiple groups for comparison is also possible. For example, if you want to compare between average of C2 and C3 and average of C1, (C3+C2)/2-C1 as formula, set '-1' for C1, '0.5' for C2 and '0.5' for C3, and '0' for rest of groups.

```
comparison1 <- matrix(c(-1,1,0,0,0,0),nrow=1)
comparison2 <- matrix(c(0,-1,1,0,0,0),nrow=1)
comparison3 <- matrix(c(0,0,-1,1,0,0),nrow=1)
comparison4 <- matrix(c(0,0,0,-1,1,0),nrow=1)
comparison5 <- matrix(c(0,0,0,0,-1,1),nrow=1)
comparison6 <- matrix(c(1,0,0,0,0,-1),nrow=1)

comparison<-rbind(comparison1,comparison2,comparison3,comparison4,comparison5,comparison6)
row.names(comparison) <- c("C2-C1","C3-C2","C4-C3","C5-C4","C6-C5","C1-C6")
colnames(comparison) <- c("C1","C2","C3","C4","C5","C6")
```

With the contrast matrix specified, group comparison can be performed as follows.

```
DDA2009.comparisons <- groupComparison(contrast.matrix = comparison, data = DDA2009.proposed)
```

```
## INFO  [2023-02-28 10:22:43]  == Start to test and get inference in whole plot ...
##   |                                                                              |
## INFO  [2023-02-28 10:22:43]  == Comparisons for all proteins are done.
```

Output of the `groupComparison` function contains three data frames:

```
# output from groupComparison function has three data frames
names(DDA2009.comparisons)
```

```
## [1] "ComparisonResult" "ModelQC"          "FittedModel"
```

Results of the statistical comparison are stored in the data frame named `ComparisonResult`:

```
# name of columns in result data.frame
head(DDA2009.comparisons$ComparisonResult)
```

```
##   Protein Label     log2FC        SE     Tvalue DF       pvalue    adj.pvalue
## 1  bovine C2-C1  0.6048799 0.4245943   1.424607 11 1.820186e-01 1.820186e-01
## 2  bovine C3-C2 -7.2166867 0.4747109 -15.202277 11 9.896993e-09 5.938196e-08
## 3  bovine C4-C3  1.3344126 0.4747109   2.811000 11 1.693817e-02 1.693817e-02
## 4  bovine C5-C4  2.0463466 0.4245943   4.819533 11 5.362805e-04 6.435367e-04
## 5  bovine C6-C5  1.5983090 0.4245943   3.764320 11 3.131096e-03 3.757316e-03
## 6  bovine C1-C6  1.6327386 0.4245943   3.845408 11 2.720989e-03 3.265186e-03
##   issue MissingPercentage ImputationPercentage
## 1    NA        0.03571429           0.03571429
## 2    NA        0.48809524           0.32142857
## 3    NA        0.84523810           0.67857143
## 4    NA        0.52380952           0.52380952
## 5    NA        0.16666667           0.16666667
## 6    NA        0.03571429           0.03571429
```

The result of the test for diffrential abundance is a table with columns `Protein`, `Label` (of the comparison), log2 fold change (`log2FC`), standard error of the log2 fold change (`SE`), test statistic of the Student test (`Tvalue`), degree of freedom of the Student test (`DF`), raw p-values (`pvalue`), p-values adjusted among all the proteins in the specific comparison using the approach by Benjamini and Hochberg (`adj.pvalue`). The cutoff of the adjusted p-value corresponds to the cutoff of the False Discovery Rate (Benjamini and Hochberg 1955). The positive values of log2FC for `Label=C2-C1` indicate evidence in favor of C2 > C1 (i.e. proteins upregulated in C2), while the negative values indicate evidence in favor of C2 < C1 (i.e. proteins downregulated in C2), as compared to C1. The same model can be used to perform several comparisons of conditions simultaneously in the same protein.

NOTE `issue` column shows if there is any issue for inference in corresponding protein and comparison, for example, `OneConditionMissing` or `CompleteMissing`. If one of condition for compariosn is completely missing, it would flag with `OneConditionMissiong` with `adj.pvalue=0` and `log2FC=Inf or -Inf` even though `pvalue=NA`. For example, if you want to compare 'Condition A - Condition B', but condition B has complete missing, `log2FC=Inf` and `adj.pvalue=0`. `SE`, `Tvalue`, and `pvalue` will be NA. if you want to compare 'Condition A - Condition B', but condition A has complete missing, then `log2FC=-Inf` and `adj.pvalue=0`. But, please be careful for using this `log2FC` and `adj.pvalue`.

Based on the comparison results and desired significance level, a short list of the differentially abundant proteins can be obtained for further investigation:

```
# get only significant proteins and comparisons among all comparisons
# To simultaneoulsy controll the overall FDR at the level, 0.05
SignificantProteins <- with(DDA2009.comparisons,
                            ComparisonResult[ComparisonResult$adj.pvalue < 0.05, ])
nrow(SignificantProteins)
```

```
## [1] 34
```

4.1.6 Verifying the assumption of the model

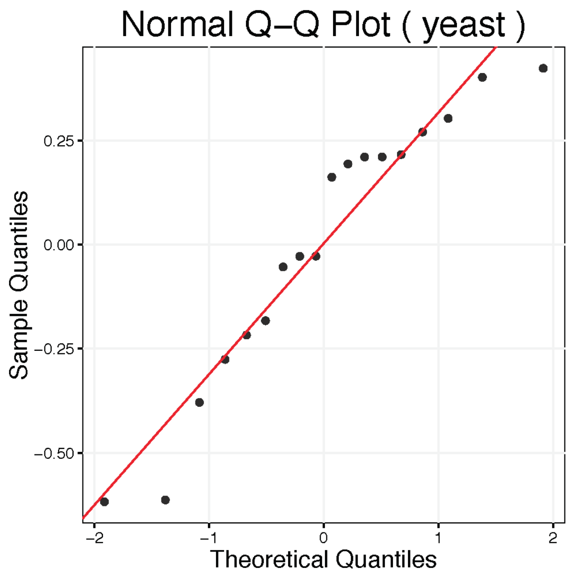
Results based on the statistical models are accurate as long as the assumptions of the models hold. Here we focus on the assumption of the Normal distribution of the measurement errors, and also on the assumption of constant variance of the measurement errors (if this option is specified in the model above). The assumptions can be checked by examining the residuals of the model fit (i.e., the deviations of the observed intensities of the transition from their model-based predictions).

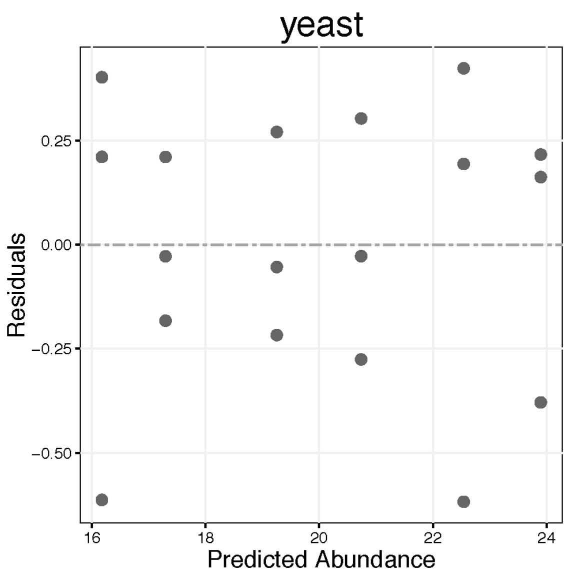
`modelBasedQCPlots` function generates residual plots and Normal quantile-quantile plots for each protein, taking as input the results of model fitting and testing in `groupComparison`. Normal quantile-quantile plot with the option `type='QQPlots'` illustrates that such deviations from constant variance can be mistaken for deviations from Normality. Only large deviations of transition intensities from the straight line are problematic.

Residual plot with the option `type='ResidualPlots'` shows variance of the residuals that is associated with the mean feature intensity. Any specific pattern, such as increasing or decreasing by predicted abundance, is problematic.

```
# normal quantile-quantile plots
modelBasedQCPlots(data=DDA2009.comparisons, type="QQPlots",
                  width=5, height=5, address="DDA2009_proposed_")

# residual plots
modelBasedQCPlots(data=DDA2009.comparisons, type="ResidualPlots",
                  width=5, height=5, address="DDA2009_proposed_")
```





For more details, visit the help file using the following code.

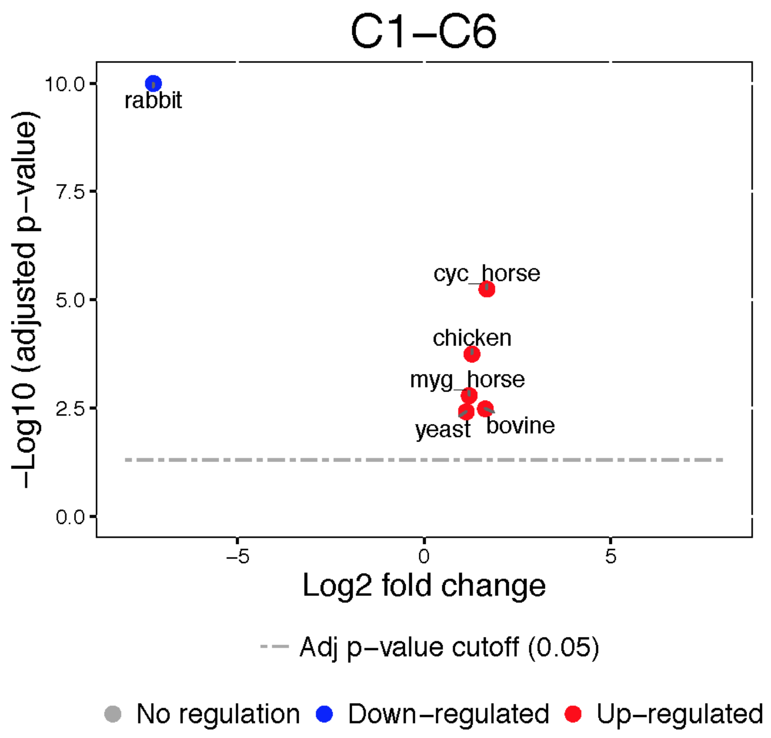
```
?modelBasedQCPlots
```

4.1.7 Visualization of differentially abundant proteins

Volcano plots Volcano plots visualize the outcome of one comparison between conditions for all the proteins, and combine the information on statistical and practical significance. The y-axis displays the FDR-adjusted p-values on the negative log10 scale, representing statistical significance. The horizontal dashed line shows the FDR cutoff. The points above the FDR cutoff line are statistically significant proteins that are differentially abundant across conditions. These points are colored in red and blue for upregulated and downregulated proteins, respectively. The x-axis is the model-based estimate of fold change on log scale (the base of logarithm transform is the same as specified in the `logTrans` option of the `dataProcess` function), and represents practical significance. It is possible to specify a practical significance cutoff based on the estimate of fold change in addition to the statistical significance cutoff. If the fold change cutoff is specified, the points above the horizontal cutoff line but within the vertical cutoff line will be considered as not differentially abundant (and will be colored in black). The practical significance cutoff should only be applied in addition to the statistical significance cutoff (i.e., the fold change alone does not present enough evidence for differential abundance).

```
groupComparisonPlots(data = DDA2009.comparisons$ComparisonResult, type = 'VolcanoPlot',
                     width=5, height=5, address="DDA2009_proposed_")
```

'VolcanoPlot.pdf' will be saved under the folder you assigned. It has the plots per comparison defined in `contrast.matrix`. Please check `?groupComparisonPlots` for detail, such as labelling protein names, size of dots, font sizes, etc. Below is one of volcano plots, for comparison 'C1-C6' including protein name labelling. Protein name will be shown for significant proteins, without overlapping protein names each other.



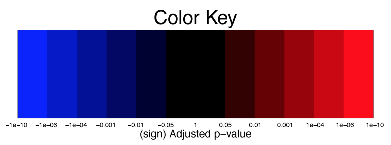
Heatmap Heatmaps illustrate the patterns of up- and down-regulation of proteins in several comparisons. Columns in the heatmaps are comparison of conditions assigned in `contrast.matrix`, and rows are proteins. The heatmaps display signed FDR-adjusted p-values of the tests, colored in red/blue for significantly up-/down-regulated proteins, while taking into account the specified FDR cutoff and the additional optional fold change cutoff. Brighter colors indicate stronger evidence in favor of differential abundance. Black color represents proteins that are not significantly differentially abundant.

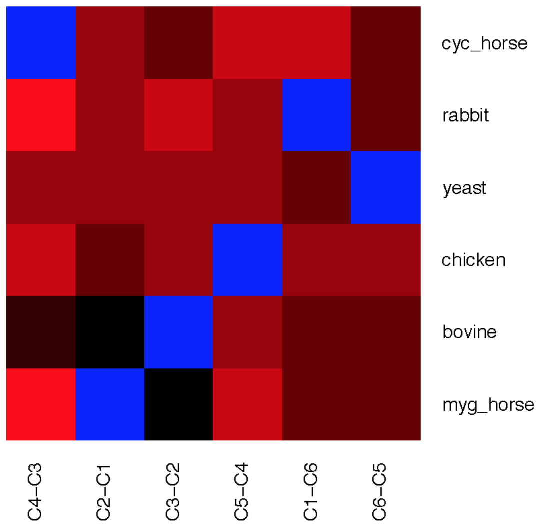
NOTE To draw heatmap, at least two comparisons are needed.

The rows and columns of the heatmaps can be ordered with the option clustering, which performs hierarchical clustering with the Ward method (minimum variance). The option `clustering='protein'` (default) clusters the rows (proteins) in the space of comparisons, based on the values of (sign of comparison) · (-log2(adjusted p-values)). The option `clustering='comparison'` clusters the columns in the space of proteins, based on the values of (sign of comparison) · (-log2(adjusted p-value)). The option `clustering='both` reorders both columns and rows.

```
groupComparisonPlots(data = DDA2009.comparisons$ComparisonResult, type = 'Heatmap')
```

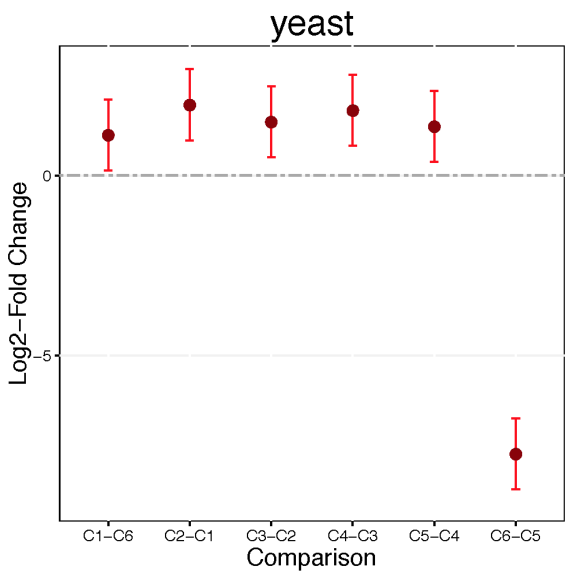
'Heatmap.pdf' will be saved under the folder you assigned. Below is one example, showing the results for several comparisons simultaneously.





Comparison plots Comparison plots illustrate model-based estimates of log-fold changes, and the associated uncertainty, in several comparisons of conditions for one protein. X-axis is the comparison of interest. Y-axis is the log fold change. The dots are the model-based estimates of log-fold change, and the error bars are the model-based 95% confidence intervals (the option sig can be used to change the significance level of significance). For simplicity, the confidence intervals are adjusted for multiple comparisons within protein only, using the Bonferroni approach. For proteins with N comparisons, the individual confidence intervals are at the level of 1-sig/N.

```
groupComparisonPlots(data=DDA2009.comparisons$ComparisonResult, type="ComparisonPlot",
                     width=5, height=5, address="DDA2009_proposed_")
```



For further details, such as labelling protein names, size of dots, font sizes, etc., visit the help file using the following code.

```
?groupComparisonPlots
```

4.1.8 Sample size calculation for a future experiment

This last analysis step views the dataset as a pilot study of a future experiment, utilizes its variance components, and calculates the minimal number of replicates required in a future experiment to achieve the desired statistical power. The calculation is performed by the function `designSampleSize`, which takes as input the fitted model in `groupComparison`. Sample size calculation assumes same experimental design (i.e. group comparison, time course or paired design) as in the current dataset, and uses the model fit to estimate the median variance components across all the proteins. Finally, sample size calculation assumes that a large proportion of proteins (specifically, 99%) will not change in abundance in the future experiment. This assumption also provides conservative results. Using the estimated variance components, the function relates the number of biological replicates per condition (`numSample`, rounded to 0 decimal), average statistical power across all the proteins (`power`), minimal fold change that we would like to detect (can be specified as a range, e.g. `desiredFC=c(1.1, 2)`), and the False Discovery Rate (`FDR`). The user should specify all these quantities but one, and the function will solve for the remainder. The quantity to solve for should be set to `= TRUE`.

```
# Minimal number of biological replicates per condition
result.sample <- designSampleSize(data=DDA2009.comparisons$FittedModel, numSample=TRUE,
                 desiredFC=c(1.25, 3), FDR=0.05, power=0.8)
result.sample
```

```
##    desiredFC numSample  FDR power    CV
## 1      1.250        35 0.05   0.8 0.004
## 2      1.275        30 0.05   0.8 0.005
## 3      1.300        25 0.05   0.8 0.006
## 4      1.325        22 0.05   0.8 0.007
## 5      1.350        19 0.05   0.8 0.007
## 6      1.375        17 0.05   0.8 0.008
## 7      1.400        16 0.05   0.8 0.009
## 8      1.425        14 0.05   0.8 0.010
## 9      1.450        13 0.05   0.8 0.010
## 10     1.475        12 0.05   0.8 0.011
## 11     1.500        11 0.05   0.8 0.012
## 12     1.525        10 0.05   0.8 0.013
## 13     1.550         9 0.05   0.8 0.014
## 14     1.575         9 0.05   0.8 0.014
## 15     1.600         8 0.05   0.8 0.015
## 16     1.625         7 0.05   0.8 0.017
## 17     1.650         7 0.05   0.8 0.017
## 18     1.675         7 0.05   0.8 0.016
## 19     1.700         6 0.05   0.8 0.019
## 20     1.725         6 0.05   0.8 0.018
## 21     1.750         6 0.05   0.8 0.018
## 22     1.775         5 0.05   0.8 0.022
## 23     1.800         5 0.05   0.8 0.021
## 24     1.825         5 0.05   0.8 0.021
## 25     1.850         5 0.05   0.8 0.021
## 26     1.875         4 0.05   0.8 0.026
## 27     1.900         4 0.05   0.8 0.025
## 28     1.925         4 0.05   0.8 0.025
## 29     1.950         4 0.05   0.8 0.025
## 30     1.975         4 0.05   0.8 0.024
## 31     2.000         4 0.05   0.8 0.024
## 32     2.025         4 0.05   0.8 0.024
## 33     2.050         3 0.05   0.8 0.031
## 34     2.075         3 0.05   0.8 0.031
## 35     2.100         3 0.05   0.8 0.030
## 36     2.125         3 0.05   0.8 0.030
## 37     2.150         3 0.05   0.8 0.030
## 38     2.175         3 0.05   0.8 0.029
## 39     2.200         3 0.05   0.8 0.029
## 40     2.225         3 0.05   0.8 0.029
## 41     2.250         3 0.05   0.8 0.028
## 42     2.275         3 0.05   0.8 0.028
## 43     2.300         3 0.05   0.8 0.028
## 44     2.325         2 0.05   0.8 0.041
## 45     2.350         2 0.05   0.8 0.041
## 46     2.375         2 0.05   0.8 0.040
## 47     2.400         2 0.05   0.8 0.040
## 48     2.425         2 0.05   0.8 0.039
## 49     2.450         2 0.05   0.8 0.039
## 50     2.475         2 0.05   0.8 0.039
## 51     2.500         2 0.05   0.8 0.038
## 52     2.525         2 0.05   0.8 0.038
## 53     2.550         2 0.05   0.8 0.038
## 54     2.575         2 0.05   0.8 0.037
## 55     2.600         2 0.05   0.8 0.037
## 56     2.625         2 0.05   0.8 0.036
## 57     2.650         2 0.05   0.8 0.036
## 58     2.675         2 0.05   0.8 0.036
## 59     2.700         2 0.05   0.8 0.035
## 60     2.725         2 0.05   0.8 0.035
## 61     2.750         2 0.05   0.8 0.035
## 62     2.775         2 0.05   0.8 0.034
## 63     2.800         2 0.05   0.8 0.034
## 64     2.825         2 0.05   0.8 0.034
## 65     2.850         2 0.05   0.8 0.034
## 66     2.875         2 0.05   0.8 0.033
## 67     2.900         2 0.05   0.8 0.033
## 68     2.925         2 0.05   0.8 0.033
## 69     2.950         1 0.05   0.8 0.065
## 70     2.975         1 0.05   0.8 0.064
## 71     3.000         1 0.05   0.8 0.064
```

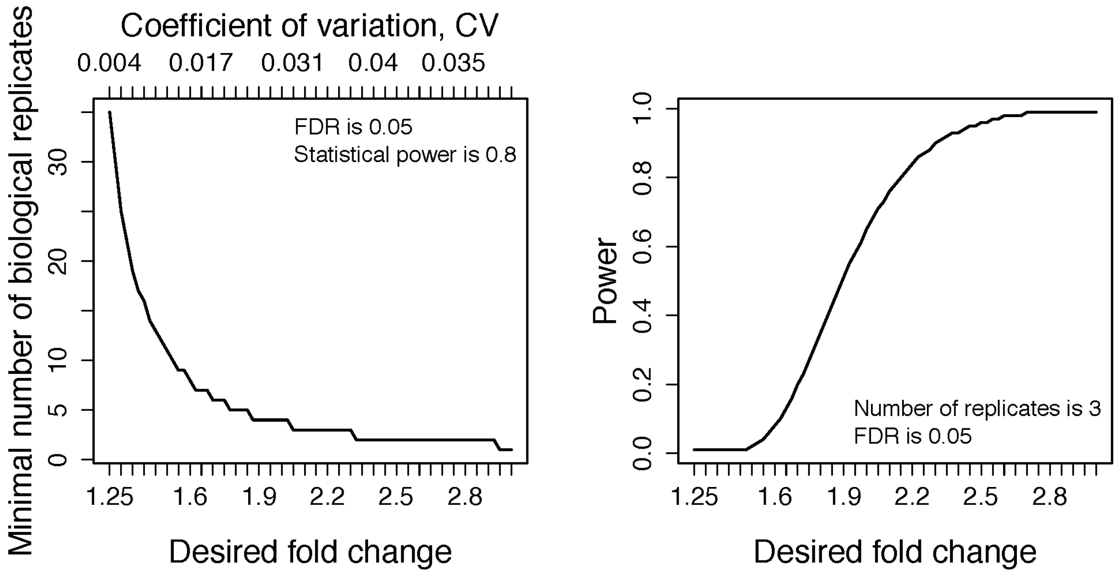
```
# Power calculation
result.power <- designSampleSize(data=DDA2009.comparisons$FittedModel, numSample=3,
                 desiredFC=c(1.25, 3), FDR=0.05, power=TRUE)
result.power
```

```
##    desiredFC numSample  FDR power    CV
## 1      1.250         3 0.05  0.01 0.051
## 2      1.275         3 0.05  0.01 0.050
## 3      1.300         3 0.05  0.01 0.049
## 4      1.325         3 0.05  0.01 0.048
## 5      1.350         3 0.05  0.01 0.047
## 6      1.375         3 0.05  0.01 0.046
## 7      1.400         3 0.05  0.01 0.046
## 8      1.425         3 0.05  0.01 0.045
## 9       1.450          3 0.05  0.01 0.044
## 10      1.475          3 0.05  0.01 0.043
## 11      1.500          3 0.05  0.02 0.043
## 12      1.525          3 0.05  0.03 0.042
## 13      1.550          3 0.05  0.04 0.041
## 14      1.575          3 0.05  0.06 0.041
## 15      1.600          3 0.05  0.08 0.040
## 16      1.625          3 0.05  0.10 0.039
## 17      1.650          3 0.05  0.13 0.039
## 18      1.675          3 0.05  0.16 0.038
## 19      1.700          3 0.05  0.20 0.038
## 20      1.725          3 0.05  0.23 0.037
## 21      1.750          3 0.05  0.27 0.036
## 22      1.775          3 0.05  0.31 0.036
## 23      1.800          3 0.05  0.35 0.035
## 24      1.825          3 0.05  0.39 0.035
## 25      1.850          3 0.05  0.43 0.034
## 26      1.875          3 0.05  0.47 0.034
## 27      1.900          3 0.05  0.51 0.034
## 28      1.925          3 0.05  0.55 0.033
## 29      1.950          3 0.05  0.58 0.033
## 30      1.975          3 0.05  0.61 0.032
## 31      2.000          3 0.05  0.65 0.032
## 32      2.025          3 0.05  0.68 0.032
## 33      2.050          3 0.05  0.71 0.031
## 34      2.075          3 0.05  0.73 0.031
## 35      2.100          3 0.05  0.76 0.030
## 36      2.125          3 0.05  0.78 0.030
## 37      2.150          3 0.05  0.80 0.030
## 38      2.175          3 0.05  0.82 0.029
## 39      2.200          3 0.05  0.84 0.029
## 40      2.225          3 0.05  0.86 0.029
## 41      2.250          3 0.05  0.87 0.028
## 42      2.275          3 0.05  0.88 0.028
## 43      2.300          3 0.05  0.90 0.028
## 44      2.325          3 0.05  0.91 0.027
## 45      2.350          3 0.05  0.92 0.027
## 46      2.375          3 0.05  0.93 0.027
## 47      2.400          3 0.05  0.93 0.027
## 48      2.425          3 0.05  0.94 0.026
## 49      2.450          3 0.05  0.95 0.026
## 50      2.475          3 0.05  0.95 0.026
## 51      2.500          3 0.05  0.96 0.026
## 52      2.525          3 0.05  0.96 0.025
## 53      2.550          3 0.05  0.97 0.025
## 54      2.575          3 0.05  0.97 0.025
## 55      2.600          3 0.05  0.98 0.025
## 56      2.625          3 0.05  0.98 0.024
## 57      2.650          3 0.05  0.98 0.024
## 58      2.675          3 0.05  0.98 0.024
## 59      2.700          3 0.05  0.99 0.024
## 60      2.725          3 0.05  0.99 0.023
## 61      2.750          3 0.05  0.99 0.023
## 62      2.775          3 0.05  0.99 0.023
## 63      2.800           3 0.05  0.99 0.023
## 64      2.825           3 0.05  0.99 0.023
## 65      2.850           3 0.05  0.99 0.022
## 66      2.875           3 0.05  0.99 0.022
## 67      2.900           3 0.05  0.99 0.022
## 68      2.925           3 0.05  0.99 0.022
## 69      2.950           3 0.05  0.99 0.022
## 70      2.975           3 0.05  0.99 0.021
## 71      3.000           3 0.05  0.99 0.021
```

For further details, visit the help file using the following code.

```
?designSampleSize
```

Visualization of sample size calculations The calculated relationship between the number of biological replicates per condition (`numSample`), average statistical power across all the proteins (`power`), minimal fold change that we would like to detect (`desiredFC`), and the False Discovery Rate (`FDR`) can be visualized using the function `designSampleSizePlots`. The function takes as input the output of `designSampleSize`.



For further details, visit the help file using the following code.

```
?designSampleSizePlots
```

4.1.9 Quantification of protein abundance in individual samples or conditions

Many downstream analysis steps (such as clustering or classification of individual samples in the space of their protein profiles) require summary values of protein abundance in each biological replicate or in each condition, on a relative scale that is comparable between runs.

`dataProcess` function performs model-based run-level summarization. `quantification` function enables subject-level summarization or group-level summarization with the run-level summarization from `dataProcess`.

The option, `type='sample'`(default), performs sample quantification, i.e. it outputs the estimates of relative protein abundance in each biological replicate. If there are technical replicates for biological replicates, sample quantification will be the median among technical replicates. If there is no technical replicate for biological replicate (sample), sample quantification will be the same as run-level summarization. In presence of completely missing values in biological replicate, the estimates will be zero.

The option `type='group'` performs group quantification, i.e. it outputs the estimates of relative protein abundance in each condition, summarized over the biological replicates (median among sample quantification). In presence of completely missing values in a condition, the estimates will be zero.

MSstats supports two output formats. The option `format='matrix'` (default) outputs an array where rows are `proteins`, and columns are `conditions` (for group quantification), or combinations of biological replicate and condition ids (for sample quantification). The option `format='long'` produces an array where each row corresponding to relative protein abundances, and columns are `Protein`, `Condition`, `LogIntensities` (and BioReplicate in the case of sample quantification).

```
subQuant <- quantification(DDA2009.proposed)
head(subQuant)
```

```
##      Protein     C1_1     C2_1     C3_1     C4_1     C5_1     C6_1
## 1:    bovine 20.85653 21.60443 14.32690 16.10441 17.63141 19.27802
## 2:   chicken 18.48792 19.43204 20.41274 22.42284 15.92462 17.09803
## 3: cyc_horse 20.25927 21.33967 22.22028 15.85252 17.62720 18.45536
## 4: myg_horse 22.66495 14.73701 14.99667 18.61740 20.26392 21.52022
## 5:    rabbit 14.89507 15.88492 17.43767 20.19014 21.27964 22.07550
## 6:     yeast 17.26792 19.19987 20.71073 22.73666 24.06156 16.38660
```

```
groupQuant <- quantification(DDA2009.proposed, type='group')
head(groupQuant)
```

```
##      Protein       C1       C2       C3       C4       C5       C6
## 1:    bovine 20.85653 21.60443 14.32690 16.10441 17.63141 19.27802
## 2:   chicken 18.48792 19.43204 20.41274 22.42284 15.92462 17.09803
## 3: cyc_horse 20.25927 21.33967 22.22028 15.85252 17.62720 18.45536
## 4: myg_horse 22.66495 14.73701 14.99667 18.61740 20.26392 21.52022
## 5:    rabbit 14.89507 15.88492 17.43767 20.19014 21.27964 22.07550
## 6:     yeast 17.26792 19.19987 20.71073 22.73666 24.06156 16.38660
```

For further details, visit the help file using the following code.

```
?quantification
```

4.2 Suggested workflow with Skyline output for DDA

This section describes steps and considerations to properly format data processed by Skyline, prior to the `MSstats` analysis. In the following example, the raw files for the benchmark dataset (Choi, M. and Eren-Dogu, Z. F. and Colangelo, C. and Cottrell, J. and Hoopmann, M. R. and Kapp, E. A. and Kim, S. and Lam, H. and Neubert, T. A. and Palmblad, M. and Phinney, B. S. and Weintraub, S. T. and MacLean, B. and Vitek, O. 2017) are used. Dataset was processed and quantified with Skyline (3.5.0.9319). Details for data processing are described in Choi, et al., 2017 and Panorama Web https://panoramaweb.org/iPRG-2015.url for iProphet cut-off 0.15. The datasets and details for data processing are available in MassIVE.quant, MSV000079843, Reanalysis : RMSV000000249.1

4.2.1 Load Skyline output

This required input data is generated automatically when using `MSstats` report format in Skyline. We first load and access the dataset processed by Skyline. The name of saved file from Skyline using `MSstats report format` is 'Choi2017_DDA_Skyline_input.csv' under the folder named dda_skyline.

```
# Read output from skyline
raw <- read.csv("dda_skyline/Choi2017_DDA_Skyline_input.csv")
```

We can read csv file. Here we will load R data file which is the exactly same data in Choi2017_DDA_Skyline_input.csv file.

```
# Load R data, which is convered from csv file, output from skyline
load("dda_skyline/iprg.skyline.rda")
raw <- iprg.skyline
head(raw)
```

```
##                   ProteinName PeptideSequence PeptideModifiedSequence
## 1 DECOY_sp|P0CF18|YM085_YEAST      KDMYGNPFQK         KDM[+16]YGNPFQK
## 2 DECOY_sp|P0CF18|YM085_YEAST      KDMYGNPFQK         KDM[+16]YGNPFQK
## 3 DECOY_sp|P0CF18|YM085_YEAST      KDMYGNPFQK         KDM[+16]YGNPFQK
## 4 DECOY_sp|P0CF18|YM085_YEAST      KDMYGNPFQK         KDM[+16]YGNPFQK
## 5 DECOY_sp|P0CF18|YM085_YEAST      KDMYGNPFQK         KDM[+16]YGNPFQK
## 6 DECOY_sp|P0CF18|YM085_YEAST      KDMYGNPFQK         KDM[+16]YGNPFQK
##   PrecursorCharge PrecursorMz FragmentIon ProductCharge ProductMz
## 1               3    415.1974   precursor             3  415.1974
## 2               3    415.1974   precursor             3  415.1974
## 3               3    415.1974   precursor             3  415.1974
## 4               3    415.1974   precursor             3  415.1974
## 5               3    415.1974   precursor             3  415.1974
## 6               3    415.1974   precursor             3  415.1974
##   IsotopeLabelType  Condition BioReplicate                     FileName
## 1            light Condition1            1 JD_06232014_sample1-A.raw
## 2            light Condition1            2 JD_06232014_sample1_B.raw
## 3            light Condition1            3 JD_06232014_sample1_C.raw
## 4            light Condition2            4 JD_06232014_sample2_A.raw
## 5            light Condition2            5 JD_06232014_sample2_B.raw
## 6            light Condition2            6 JD_06232014_sample2_C.raw
##             Area StandardType Truncated DetectionQValue
## 1  71765.046875           NA     False            #N/A
## 2 147327.265625           NA     False            #N/A
## 3     1373396.5           NA     False            #N/A
## 4 66387.4453125           NA     False            #N/A
## 5 107736.453125           NA     False            #N/A
## 6   380812.0625           NA     False            #N/A
```

Annotation information is required to fill in `Condition` and `BioReplicate` for corresponding `Run` information. Users have to prepare as csv or txt file like 'Choi2017_DDA_Skyline_annotation.csv', which includes `Run`, `Condition`, and `BioReplicate` information, and load it in R.

```
annot <- read.csv("dda_skyline/Choi2017_DDA_Skyline_annotation.csv", header=TRUE)
annot
```

```
##                          Run  Condition BioReplicate
## 1  JD_06232014_sample1-A.raw Condition1            1
## 2  JD_06232014_sample1_B.raw Condition1            1
## 3  JD_06232014_sample1_C.raw Condition1            1
## 4  JD_06232014_sample2_A.raw Condition2            2
## 5  JD_06232014_sample2_B.raw Condition2            2
## 6  JD_06232014_sample2_C.raw Condition2            2
## 7  JD_06232014_sample3_A.raw Condition3            3
## 8  JD_06232014_sample3_B.raw Condition3            3
## 9  JD_06232014_sample3_C.raw Condition3            3
## 10 JD_06232014_sample4-A.raw Condition4            4
## 11 JD_06232014_sample4_B.raw Condition4            4
## 12 JD_06232014_sample4_C.raw Condition4           4
```

4.2.2 Preprocessing with DDA experiment from Skyline output

The input data for `MSstats` is required to contain variables of `ProteinName`, `PeptideSequence`, `PrecursorCharge`, `FragmentIon`, `ProductCharge`, `IsotopeLabelType`, `Condition`, `BioReplicate`, `Run`, `Intensity`. These variable names should be fixed. `MSstats` input from Skyline adapts the column scheme of the dataset so that it fits `MSstats` input format. However there are several extra column names and also some of them need to be changed. `SkylinetoMSstatsFormat` function helps pre-processing for making right format of MSstats input from Skyline output. For example, it renames some column name, and replace truncated peak intensities with NA. Another important step is to handle isotopic peaks before using `dataProcess`. The output from Skyline for DDA experiment has several measurements of peak area from the monoisotopic, M+1 and M+2 peaks. To get a robust measure of peptide intensity, we can sum over isotopic peaks per peptide or use the highest peak. Here we take a summation per peptide ion.

Here is the summary of pre-processing steps in `SkylinetoMSstatsFormat` function.

[](img/(MSstats-converter-skyline.png){width=70%}

Options for `SkylinetoMSstatsFormat`

- **annotation** : name of 'annotation.txt' or 'annotation.csv' data which includes Condition, BioReplicate, and Run. If annotation is already complete in Skyline, use annotation=NULL (default). It will use the annotation information from input.
- **removeiRT** : TRUE (default) will remove the proteins or peptides which are labeld 'iRT' in 'StandardType' column. FALSE will keep them.
- **filter__with__Qvalue** : TRUE (default) will filter out the intensities that have greater than `qvalue_cutoff` in `DetectionQValue` column. Those intensities will be replaced with **zero** and will be considered as censored missing values for imputation purpose.
- **qvalue__cutoff** : Cutoff for `DetectionQValue`. Default is 0.01.
- **useUniquePeptide** : TRUE (default) removes peptides that are assigned for more than one proteins. We assume to use unique peptide for each protein.
- **fewMeasurement** : remove or keep the featurew with few measurements.
  - **'remove'** : (default) remove the features that have 1 or 2 measurements across runs.
  - **'keep'** : keep all the features. However, it could generate the error in the step for fitting the statistical model.
- **removeOxidationMpeptides** : TRUE will remove the peptides including 'oxidation (M)' in modification. FALSE is default.
- **removeProtein__with1Feature** : TRUE will remove the proteins which have only 1 peptide and charge. FALSE is default.

For further details, visit the help file using the following code.

```
?SkylinetoMSstatsFormat
```

Now, we use `SkylinetoMSstatsFormat` function for this example dataset. We chose to remove the proteins with only 1 peptide ion.

```
# reformating and pre-processing for Skyline output.
quant <- SkylinetoMSstatsFormat(raw,
                                annotation = annot,
                                removeProtein_with1Feature = TRUE)
```

```
## INFO  [2023-02-28 10:22:52] ** Raw data from Skyline imported successfully.
## Warning in eval(jsub, SDenv, parent.frame()): NAs introduced by coercion
```

```
## INFO  [2023-02-28 10:22:54] ** Raw data from Skyline cleaned successfully.
## INFO  [2023-02-28 10:22:54] ** Using provided annotation.
## INFO  [2023-02-28 10:22:54] ** Run labels were standardized to remove symbols such as '.' or '%'.
## INFO  [2023-02-28 10:22:54] ** The following options are used:
##   - Features will be defined by the columns: IsotopeLabelType, PeptideSequence, PrecursorCharge, Frag
##   - Shared peptides will be removed.
##   - Proteins with a single feature will be removed.
##   - Features with less than 3 measurements across runs will be removed.
## INFO  [2023-02-28 10:22:54] ** Rows with values of StandardType equal to iRT are removed
## INFO  [2023-02-28 10:22:55] ** Intensities with values of Truncated equal to TRUE are replaced with 
## INFO  [2023-02-28 10:22:55] ** Intensities with values not smaller than 0.01 in DetectionQValue are 
## INFO  [2023-02-28 10:22:55] ** Sequences containing DECOY, Decoys are removed.
## INFO  [2023-02-28 10:22:57] ** Three isotopic preaks per feature and run are summed
## INFO  [2023-02-28 10:22:58] ** Features with all missing measurements across runs are removed.
## INFO  [2023-02-28 10:22:58] ** Shared peptides are removed.
## INFO  [2023-02-28 10:22:59] ** Multiple measurements in a feature and a run are summarized by summar
## INFO  [2023-02-28 10:22:59] ** Features with one or two measurements across runs are removed.
## INFO  [2023-02-28 10:22:59] Proteins with a single feature are removed.
## INFO  [2023-02-28 10:23:00] ** Run annotation merged with quantification data.
## INFO  [2023-02-28 10:23:00] ** Features with one or two measurements across runs are removed.
## INFO  [2023-02-28 10:23:00] ** Fractionation handled.
## INFO  [2023-02-28 10:23:01] ** Updated quantification data to make balanced design. Missing values a
## INFO  [2023-02-28 10:23:01] ** Finished preprocessing. The dataset is ready to be processed by the d
```

This function shows the progress. The output of `SkylinetoMSstatsFormat`, called `quant`, is ready for next step.

```
head(quant)
```

```
##            ProteinName    PeptideSequence PrecursorCharge FragmentIon
## 1 sp|P38915|SPT8_YEAST AAAAGAGGAGDSGDAVTK               2          NA
## 2 sp|P38915|SPT8_YEAST AAAAGAGGAGDSGDAVTK               2          NA
## 3 sp|P38915|SPT8_YEAST AAAAGAGGAGDSGDAVTK               2          NA
## 4 sp|P38915|SPT8_YEAST AAAAGAGGAGDSGDAVTK               2          NA
## 5 sp|P38915|SPT8_YEAST AAAAGAGGAGDSGDAVTK               2          NA
## 6 sp|P38915|SPT8_YEAST AAAAGAGGAGDSGDAVTK               2          NA
##   ProductCharge IsotopeLabelType  Condition BioReplicate
## 1            NA            light Condition1            1
## 2            NA            light Condition1            1
## 3            NA            light Condition1            1
## 4            NA            light Condition2            2
## 5            NA            light Condition2            2
## 6            NA            light Condition2            2
##                       Run Fraction Intensity
## 1 JD_06232014_sample1-Araw        1  10839770
## 2 JD_06232014_sample1_Braw        1         0
## 3 JD_06232014_sample1_Craw        1  15918394
## 4 JD_06232014_sample2_Araw        1   7846124
## 5 JD_06232014_sample2_Braw        1  14196048
## 6 JD_06232014_sample2_Craw        1  12240984
```

4.2.3 Different options for Skyline in dataProcess

The difference between output from Skyline and other spectral processing tool is that Skyline distinguishes random missing (NA) by technical issues and low noisy intensity due to less than limit of etection. The output from Skyline can have both NA (expect small number of NAs or none of them) and very small intensity close to zero (less than 1 in intensity) and those should be treated different types of missing. In `dataProcess`, users need to use `censoredInt='0'` for Skyline output, which means to distinguish between `NA` as random missing and `0` as censored missing.

```
skyline.proposed <- dataProcess(quant,
                                normalization='equalizeMedian',
                                summaryMethod="TMP",
                                cutoffCensored="minFeature",
                                censoredInt="0", ## !! important
                                MBimpute=TRUE,
                                maxQuantileforCensored=0.999)
```

Further steps is the same as in general workflow (section 4.1).

4.3 Suggested workflow with MaxQuant output for DDA

The following R code chunks show steps to format a MaxQuant output for analysis by `MSstats`. In the following example, the raw files for the benchmark dataset (Choi, M. and Eren-Dogu, Z. F. and Colangelo, C. and Cottrell, J. and Hoopmann, M. R. and Kapp, E. A. and Kim, S. and Lam, H. and Neubert, T. A. and Palmblad, M. and Phinney, B. S. and Weintraub, S. T. and MacLean, B. and Vitek, O. 2017) are used. MS/MS spectra were searched using MaxQuant (v1.5.1.2) and Andromeda as search engine. The datasets and details for data processing are available in MassIVE.quant, MSV000079843, Reanalysis : RMSV000000249.2

4.3.1 Load MaxQuant outputs

Three files should be prepared before `MSstats`. Two files, 'proteinGroups.txt' and 'evidence.txt' are outputs from MaxQuant.

```
## First, get protein ID information
proteinGroups <- read.table("dda_maxquant/Choi2017_DDA_MaxQuant_proteinGroups.txt", sep = "\t", header =
```

```
## Read in MaxQuant file: evidence.txt
infile <- read.table("dda_maxquant/Choi2017_DDA_MaxQuant_evidence.txt", sep = "\t", header = TRUE)
```

One file is for annotation information, required to fill in `Condition` and `BioReplicate` for corresponding `Run` information. Users have to prepare as csv or txt file like 'Choi2017_DDA_MaxQuant_annotation.csv', which includes `Run`, `Condition`, and `BioReplicate` information, and load it in R.

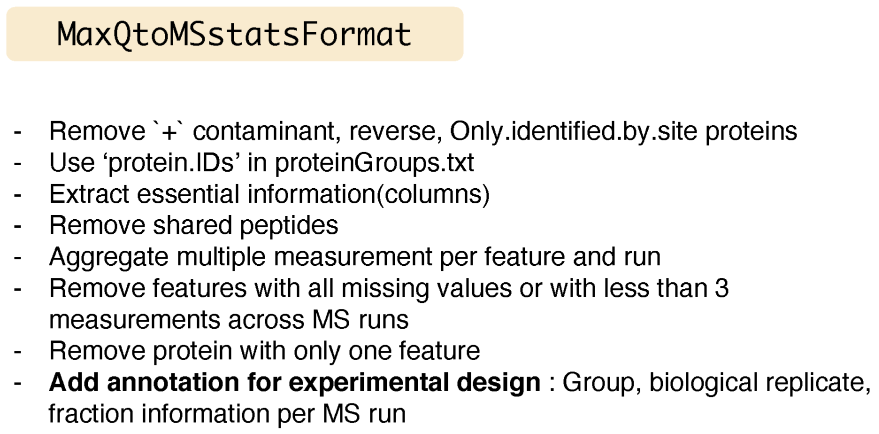
```
## Read in annotation including condition and biological replicates: annotation.csv
annot <- read.csv("dda_maxquant/Choi2017_DDA_MaxQuant_annotation.csv", header = TRUE)
annot
```

```
##                  Raw.file  Condition BioReplicate Experiment IsotopeLabelType
## 1  JD_06232014_sample1-A Condition1            1  sample1_A                L
## 2  JD_06232014_sample2_A Condition2            2  sample2_A                L
## 3  JD_06232014_sample4_B Condition4            4  sample4_B                L
## 4  JD_06232014_sample1_B Condition1            1  sample1_B                L
## 5  JD_06232014_sample1_C Condition1            1  sample1_C                L
## 6  JD_06232014_sample2_B Condition2            2  sample2_B                L
## 7  JD_06232014_sample2_C Condition2            2  sample2_C                L
## 8  JD_06232014_sample3_A Condition3            3  sample3_A                L
## 9  JD_06232014_sample3_B Condition3           3  sample3_B                L
## 10 JD_06232014_sample3_C Condition3           3  sample3_C                L
## 11 JD_06232014_sample4-A Condition4           4  sample4_A                L
## 12 JD_06232014_sample4_C Condition4           4  sample4_C                L
```

4.3.2 Preprocessing with DDA experiment from MaxQuant output

`MaxQtoMSstatsFormat` function helps pre-processing for making right format of MSstats input from MaxQuant output. Basically, this function gets peptide ion intensity from 'evidence.txt' file. In addition, there are several steps to filter out or to modify the data in order to get required information.

Here is the summary of pre-processing steps in `MaxQtoMSstatsFormat` function.



MaxQtoMSstatsFormat

- Remove `+` contaminant, reverse, Only.identified.by.site proteins
- Use 'protein.IDs' in proteinGroups.txt
- Extract essential information(columns)
- Remove shared peptides
- Aggregate multiple measurement per feature and run
- Remove features with all missing values or with less than 3 measurements across MS runs
- Remove protein with only one feature
- **Add annotation for experimental design** : Group, biological replicate, fraction information per MS run

Options for `MaxQtoMSstatsFormat`

- **evidence** : name of 'evidence.txt' data, which includes feature-level data
- **proteinGroups** : name of 'proteinGroups.txt' data. It needs to matching protein group ID. If proteinGroups=NULL, use 'Proteins' column in 'evidence.txt'.
- **annotation** :name of 'annotation.txt' or 'annotation.csv' data which includes Raw.file, Condition, BioReplicate, Run, and IsotopeLabelType information.
- **proteinID** : which column in evidence.txt will be used for ProteinName in MSstats.
  - **'Proteins'** : (default) `Proteins` column will be used.
  - **'Leading.razor.protein'** : `Leading.razor.protein` column will be used.
- **useUniquePeptide** : TRUE (default) removes peptides that are assigned for more than one proteins. We assume to use unique peptide for each protein.
- **summaryforMultipleRows** : max(default), sum, or mean. MSstats assumes that there is only one measurement (peak intensity) for one feature and one run. When there are multiple measurements for certain feature and certain run, MSstats need to know which measurements need to be used for further analysis. Users can use highest(max), sum or mean among multiple measurements for one feature and one run.
- **fewMeasurement** : remove or keep the featurew with few measurements.
  - **'remove'** : (default) remove the features that have 1 or 2 measurements across runs.
  - **'keep'** : keep all the features. However, it could generate the error in the step for fitting the statistical model.
- **removeMpeptides** : TRUE will remove the peptides including 'M' sequence. FALSE is default.

- **removeOxidationMpeptides** : TRUE will remove the peptides including 'oxidation (M)' in modification. FALSE is default.
- **removeProtein__with1Peptide** : TRUE will remove the proteins which have only 1 peptide and charge. FALSE is default.

For further details, visit the help file using the following code.

```
## check options for converting format
?MaxQtoMSstatsFormat
```

Now, we use `MaxQtoMSstatsFormat` function for this example dataset. We chose to remove the proteins with only 1 peptide ion.

```
quant <- MaxQtoMSstatsFormat(evidence=infile, annotation=annot, proteinGroups=proteinGroups,
                             removeProtein_with1Peptide=TRUE)
```

```
## INFO  [2023-02-28 10:23:21] ** Raw data from MaxQuant imported successfully.
## INFO  [2023-02-28 10:23:21] ** Rows with values of Potentialcontaminant equal to + are removed
## INFO  [2023-02-28 10:23:21] ** Rows with values of Reverse equal to + are removed
## INFO  [2023-02-28 10:23:22] ** Rows with values of Potentialcontaminant equal to + are removed
## INFO  [2023-02-28 10:23:22] ** Rows with values of Reverse equal to + are removed
## INFO  [2023-02-28 10:23:22] ** Rows with values of Onlyidentifiedbysite equal to + are removed
## INFO  [2023-02-28 10:23:22] ** + Contaminant, + Reverse, + Potential.contaminant, + Only.identified.
## INFO  [2023-02-28 10:23:23] ** Raw data from MaxQuant cleaned successfully.
## INFO  [2023-02-28 10:23:23] ** Using provided annotation.
## INFO  [2023-02-28 10:23:23] ** Run labels were standardized to remove symbols such as '.' or '%'.
## INFO  [2023-02-28 10:23:23] ** The following options are used:
##   - Features will be defined by the columns: PeptideSequence, PrecursorCharge
##   - Shared peptides will be removed.
##   - Proteins with a single feature will be removed.
##   - Features with less than 3 measurements across runs will be removed.
## INFO  [2023-02-28 10:23:23] ** Features with all missing measurements across runs are removed.
## INFO  [2023-02-28 10:23:23] ** Shared peptides are removed.
## INFO  [2023-02-28 10:23:23] ** Multiple measurements in a feature and a run are summarized by summar
## INFO  [2023-02-28 10:23:24] ** Features with one or two measurements across runs are removed.
## INFO  [2023-02-28 10:23:24] Proteins with a single feature are removed.
## INFO  [2023-02-28 10:23:24] ** Run annotation merged with quantification data.
## INFO  [2023-02-28 10:23:24] ** Features with one or two measurements across runs are removed.
## INFO  [2023-02-28 10:23:24] ** Fractionation handled.
## INFO  [2023-02-28 10:23:25] ** Updated quantification data to make balanced design. Missing values a
## INFO  [2023-02-28 10:23:25] ** Finished preprocessing. The dataset is ready to be processed by the d
```

This function shows the progress. The output of `MaxQtoMSstatsFormat`, called `quant`, is ready for next step.

```
## now 'quant' is ready for MSstats
head(quant)
```

```
##   ProteinName PeptideSequence PrecursorCharge FragmentIon ProductCharge
## 1      P38998    (ac)AAVTLHLR               2          NA            NA
## 2      P38998    (ac)AAVTLHLR               2          NA            NA
## 3      P38998    (ac)AAVTLHLR               2          NA            NA
## 4      P38998    (ac)AAVTLHLR               2          NA            NA
## 5      P38998    (ac)AAVTLHLR               2          NA            NA
## 6      P38998    (ac)AAVTLHLR               2          NA            NA
##   IsotopeLabelType  Condition BioReplicate                   Run Fraction
## 1                L Condition1            1 JD_06232014_sample1-A        1
## 2                L Condition1            1 JD_06232014_sample1_B        1
## 3                L Condition1            1 JD_06232014_sample1_C        1
## 4                L Condition2            2 JD_06232014_sample2_A        1
## 5                L Condition2            2 JD_06232014_sample2_B        1
## 6                L Condition2            2 JD_06232014_sample2_C        1
##   Intensity
## 1        NA
## 2   8792900
## 3  10744000
## 4   7647300
## 5  11407000
## 6  10736000
```

4.3.3 Different options for MaxQuant in dataProcess

MaxQuant has certain or fixed threshold for intensity value internally as an parameter. Intensities less than the threshold are reported as NA. All missing values are NA in output from MaxQuant. In `dataProcess`, users need to use `censoredInt='NA'`. Users can used the same choice for other options.

```
maxquant.proposed <- dataProcess(quant,
                                normalization='equalizeMedian',
                                summaryMethod="TMP",
                                cutoffCensored="minFeature",
                                censoredInt="NA", ## !! important for MaxQuant
                                MBimpute=TRUE,
                                maxQuantileforCensored=0.999)
```

Further steps is the same as in general workflow (section 4.1).

4.4 Suggested workflow with Progenesis output for DDA

This section describes steps and considerations to properly format data processed by Progenesis, prior to the `MSstats` analysis. In the following example, the raw files for the benchmark dataset (Choi, M. and Eren-Dogu, Z. F. and Colangelo, C. and Cottrell, J. and Hoopmann, M. R. and Kapp, E. A. and Kim, S. and Lam, H. and Neubert, T. A. and Palmblad, M. and Phinney, B. S. and Weintraub, S. T. and MacLean, B. and Vitek, O. 2017) are used. Peptide features were identified with the Progenesis algorithm (v4.0.6403), aligned across all files, and annotated with the peptide identification resulting from the database search result from Comet. The datasets and details for data processing are available in MassIVE.quant, MSV000079843, Reanalysis : RMSV000000249.3

4.4.1 Load Progenesis output

Here is the expected input for MSstats, which is output of Progenesis.

```
## First, read output of Progenesis
raw <- read.csv("dda_progenesis/Choi2017_DDA_Progenesis_input.csv")
head(raw)
```

```
##       X                  X.1    X.2               X.3               X.4
## 1
## 2     # Retention time (min) Charge               m/z     Measured mass
## 3    16     52.5563333333333      2 501.781277638303 1001.54800234285
## 4    32             38.15255      2 474.251481407549 946.488409881339
## 5 11167     36.2224333333333      2  474.25154745893 946.488541984099
## 6    41              45.5598      2 371.731536419815 741.448519905869
##                     X.5                X.6    X.7       X.8           X.9
## 1
## 2        Mass error (u)  Mass error (ppm)  Score  Sequence Modifications
## 3 -0.00255665715405939 -2.55269904358308      1 TANDVLTIR
## 4 -0.00219111866147159 -2.31499251990099      1 VTDGVMVAR
## 5  -0.0020590159006133 -2.17542139186367      1 VTDGVMVAR
## 6 -0.00120309413080122 -1.62262402086192 0.9996   AGLNIVR
##                    X.10
## 1
## 2             Accession
## 3 sp|P00549|KPYK1_YEAST
## 4 sp|P00549|KPYK1_YEAST
## 5 sp|P00549|KPYK1_YEAST
## 6 sp|P00549|KPYK1_YEAST
##                                                                                         X.11
## 1
## 2                                                                                Description
## 3 Pyruvate kinase 1 OS=Saccharomyces cerevisiae (strain ATCC 204508 \\ S288c) GN=CDC19 PE=1 SV=2
## 4 Pyruvate kinase 1 OS=Saccharomyces cerevisiae (strain ATCC 204508 \\ S288c) GN=CDC19 PE=1 SV=2
## 5 Pyruvate kinase 1 OS=Saccharomyces cerevisiae (strain ATCC 204508 \\ S288c) GN=CDC19 PE=1 SV=2
## 6 Pyruvate kinase 1 OS=Saccharomyces cerevisiae (strain ATCC 204508 \\ S288c) GN=CDC19 PE=1 SV=2
##                  X.12             X.13                  X.14
## 1
## 2 Use in quantitation  Max fold change Highest mean condition
## 3               False 1.23101575731737                      A
## 4               False 1.35108253622201                      B
## 5               False 1.25419527606242                      B
## 6               False 1.04868912680216                      A
##                   X.15               X.16             X.17
## 1
## 2 Lowest mean condition              Anova       Maximum CV
## 3                     C 0.0522715538027003 12.0175133289667
## 4                     A 0.0393818452091522 26.8776079679151
## 5                     A  0.253277920596793 27.2310093101224
## 6                     B  0.993981434364646 27.0631636013386
##    Normalized.abundance                  X.18                  X.19
## 1                     A
## 2 JD_06232014_sample1-A JD_06232014_sample2_A JD_06232014_sample3_A
## 3      234646642.659118      246323351.490501       306102714.66799
## 4      179120293.733639      104309665.701784      136741892.392964
## 5      2233197.90782367       1134566.5998162      1574437.81004362
## 6      123797188.716029       122761256.64621      116107425.243685
##                    X.20                  X.21                  X.22
## 1                                           B
## 2 JD_06232014_sample4-A JD_06232014_sample1_B JD_06232014_sample2_B
## 3      257629531.217182      235779468.539422      236753257.546934
## 4      105011188.469111      182469696.644175      183285243.781685
## 5      1701362.72342001      2336796.51015815      1788630.29256942
## 6       63610598.879437      108255803.660911      108785457.069653
##                    X.23                  X.24                  X.25
## 1                                                                 C
## 2 JD_06232014_sample3_B JD_06232014_sample4_B JD_06232014_sample1_C
## 3      186699807.218591      242959514.796972      223435557.783206
## 4      162853464.030243      180957229.609825      186073547.948691
## 5      1932732.32106691      2274168.76697097       2182403.4051037
## 6      92469286.5619254      96974519.2450418      115737794.475905
##                   X.26                  X.27                  X.28
## 1
## 2 JD_06232014_sample2_C JD_06232014_sample3_C JD_06232014_sample4_C
## 3      220456628.684641      197285091.954229      207473304.500931
## 4      163946644.909844      150247529.397207      176820166.306319
## 5       2040841.8048229      1497501.50492545      2156318.09540588
## 6      100428622.768589      109442962.863466      83998024.0208531
##           Raw.abundance                  X.29                  X.30
## 1                     A
## 2 JD_06232014_sample1-A JD_06232014_sample2_A JD_06232014_sample3_A
## 3      244531299.931508      221199440.186087      277078923.760572
## 4      186665863.932395      93670533.1411598      123776414.130536
## 5      2327272.96336292      1018845.73758267       1425154.0840074
## 6      129012233.635811      110240142.001841      105098448.610706
##                   X.31                  X.32                  X.33
## 1                                           B
## 2 JD_06232014_sample4-A JD_06232014_sample1_B JD_06232014_sample2_B
## 3      213112377.670857       265826760.55748      265610928.042007
## 4      86865756.2312952      205723291.596601       205625739.64841
## 5      1407375.36397932      2634593.46237989      2006645.04833363
## 6      52618990.9526949      122051719.672593       122045231.85501
##                   X.34                  X.35                  X.36
## 1                                                                 C
## 2 JD_06232014_sample3_B JD_06232014_sample4_B JD_06232014_sample1_C
## 3      219812880.452275      242959514.796972      210648874.118526
## 4      191737150.420336      180957229.609825      175425002.929311
## 5      2275521.67817463      2274168.76697097      2057509.66730008
## 6      108869636.960823      96974519.2450418      109114396.746851
##                   X.37                  X.38                  X.39
## 1
## 2 JD_06232014_sample2_C JD_06232014_sample3_C JD_06232014_sample4_C
## 3      207260386.701525      217212520.118758      194785326.277837
## 4      154133015.755399      165423774.187439      166006773.109084
## 5      1918679.71576989       1648761.5589421      2024448.99975338
## 6      94417098.3431607      120497608.498225      78861145.7987456
##         Spectral.counts                  X.40                  X.41
## 1                     A
## 2 JD_06232014_sample1-A JD_06232014_sample2_A JD_06232014_sample3_A
## 3                     2                     1                     1
## 4                     1                     1                     1
## 5                     1                     0                     1
## 6                     2                     2                     2
##                   X.42                  X.43                  X.44
## 1                                           B
## 2 JD_06232014_sample4-A JD_06232014_sample1_B JD_06232014_sample2_B
## 3                     3                     2                     3
## 4                     1                     1                     1
## 5                     1                     0                     0
## 6                     2                     2                     2
##                   X.45                  X.46                  X.47
## 1                                                                 C
## 2 JD_06232014_sample3_B JD_06232014_sample4_B JD_06232014_sample1_C
## 3                     2                     2                     2
## 4                       1                       1                       1
## 5                       0                       1                       1
## 6                       2                       2                       2
##                      X.48                    X.49                    X.50
## 1
## 2 JD_06232014_sample2_C JD_06232014_sample3_C JD_06232014_sample4_C
## 3                       3                       1                       2
## 4                       1                       1                       1
## 5                       0                       1                       0
## 6                       2                       2                       2
```

One file is for annotation information, required to fill in `Condition` and `BioReplicate` for corresponding `Run` information. Users have to prepare as csv or txt file like 'Choi2017_DDA_Progenesis_annotation.csv', which includes `Run`, `Condition`, and `BioReplicate` information, and load it in R.

```
## Read in annotation including condition and biological replicates
annot <- read.csv("dda_progenesis/Choi2017_DDA_Progenesis_annotation.csv", header = TRUE)
annot
```

```
##                      Run  Condition BioReplicate
## 1  JD_06232014_sample1-A Condition1            1
## 2  JD_06232014_sample2_A Condition2            2
## 3  JD_06232014_sample4_B Condition4            4
## 4  JD_06232014_sample1_B Condition1            1
## 5  JD_06232014_sample1_C Condition1            1
## 6  JD_06232014_sample2_B Condition2            2
## 7  JD_06232014_sample2_C Condition2            2
## 8  JD_06232014_sample3_A Condition3            3
## 9  JD_06232014_sample3_B Condition3            3
## 10 JD_06232014_sample3_C Condition3            3
## 11 JD_06232014_sample4-A Condition4            4
## 12 JD_06232014_sample4_C Condition4            4
```

4.4.2 Preprocessing with DDA experiment from progenesis output

The output from Progenesis includes peptide ion-level quantification for each MS runs. `ProgenesistoMSstatsFormat` function helps pre-processing for making right format of MSstats input from Progenesis output. Basically, this function reformats wide format to long format. It provide 'Raw.abundance', 'Normalized.abundance' and 'Spectral count' columns. This converter uses 'Raw.abundance' columns for Intensity values. In addition, there are several steps to filter out or to modify the data in order to get required information.

Here is the summary of pre-processing steps in `ProgenesistoMSstatsFormat` function.

ProgenesistoMSstatsFormat

- Extract essential information(columns)
- Subset the rows with `use in quantitation = true`
- Rename column names
- Remove empty protein and peptide sequence
- Remove duplicated measurements
- Remove shared peptides
- Aggregate multiple measurements per feature and run
- Remove features with all missing values or with less than 3 measurements across MS runs
- Remove protein with only one feature
- **Add annotation for experimental design** : Group, biological replicate, fraction information per MS run

Options for `ProgenesistoMSstatsFormat`

- **input** : name of Progenesis output, which is wide-format. 'Accession', 'Sequence', 'Modification', 'Charge' and one column for each run are required. 'Accession' column is used for ProteinName. 'Raw.abundance' is used for Intensity.
- **annotation** :name of 'annotation.txt' or 'annotation.csv' data which includes Condition, BioReplicate, and Run information. It will be matched with the column name of input for MS runs.
- **useUniquePeptide** : TRUE (default) removes peptides that are assigned for more than one proteins. We assume to use unique peptide for each protein.
- **summaryforMultipleRows** : max(default), sum, or mean. MSstats assumes that there is only one measurement (peak intensity) for one feature and one run. When there are multiple measurements for certain feature and certain run, MSstats need to know which measurements need to be used for further analysis. Users can use highest(max), sum or mean among multiple measurements for one feature and one run.
- **fewMeasurement** : remove or keep the featurew with few measurements.
  - **'remove'** : (default) remove the features that have 1 or 2 measurements across runs.
  - **'keep'** : keep all the features. However, it could generate the error in the step for fitting the statistical model.
- **removeOxidationMpeptides** : TRUE will remove the peptides including 'oxidation (M)' in modification. FALSE is default.
- **removeProtein_with1Peptide** : TRUE will remove the proteins which have only 1 peptide and charge. FALSE is default.

For further details, visit the help file using the following code.

```
## check options for converting format
?ProgenesistoMSstatsFormat
```

Now, we use `ProgenesistoMSstatsFormat` function for this example dataset. We chose to remove the proteins only 1 peptide ion.

```
quant <- ProgenesistoMSstatsFormat(raw, annotation=annot,
                                   removeProtein_with1Peptide = TRUE)
```

This function shows the progress. The output of `ProgenesistoMSstatsFormat`, called `quant`, is ready for next step.

```
## now 'quant' is ready for MSstats
head(quant)
```

```
##   ProteinName PeptideSequence PrecursorCharge FragmentIon ProductCharge
## 1      P38998    (ac)AAVTLHLR               2          NA            NA
## 2      P38998    (ac)AAVTLHLR               2          NA            NA
## 3      P38998    (ac)AAVTLHLR               2          NA            NA
## 4      P38998    (ac)AAVTLHLR               2          NA            NA
## 5      P38998    (ac)AAVTLHLR               2          NA            NA
## 6      P38998    (ac)AAVTLHLR               2          NA            NA
##   IsotopeLabelType  Condition BioReplicate                   Run Fraction
## 1                L Condition1            1 JD_06232014_sample1-A        1
## 2                L Condition1            1 JD_06232014_sample1_B        1
## 3                L Condition1            1 JD_06232014_sample1_C        1
## 4                L Condition2            2 JD_06232014_sample2_A        1
## 5                L Condition2            2 JD_06232014_sample2_B        1
## 6                L Condition2            2 JD_06232014_sample2_C        1
##   Intensity
## 1        NA
## 2   8792900
## 3  10744000
## 4   7647300
## 5  11407000
## 6  10736000
```

4.4.3 Different options for Progenesis in dataProcess

Progenesis reports 0(zero) for missing values and does not have NA. Therefore,in `dataProcess`, users need to use `censoredInt='0'`. Users can used the same choice for other options.

```
progenesis.proposed <- dataProcess(quant,
                                   normalization='equalizeMedian',
                                   summaryMethod="TMP",
                                   cutoffCensored="minFeature",
                                   censoredInt="0", ## !! important
                                   MBimpute=TRUE,
                                   maxQuantileforCensored=0.999)
```

Further steps is the same as in general workflow (section 4.1).

4.5 Suggested workflow with Proteome Discoverer output for DDA

This section describes steps and considerations to properly format data processed by Proteome Discoverer, prior to the `MSstats` analysis. In the following example, another spike-in dataset processed by Proteome Discoverer is used to demonstrate. The datasets and details for data processing are available in MassIVE.quant, MSV000084181, Reanalysis : RMSV000000261.4

4.5.1 Load Proteome Discoverer output

The output from Proteome Discoverer includes several level of datasets. PSM sheet should be saved as csv as below. Here is the expected input for MSstats.

```
## Read PSM-level data
raw <- read.csv("dda_PD/ControlMixture_DDA_ProteomeDiscoverer_input.csv")
head(raw)
```

```
##   Confidence.Level Search.ID Processing.Node.No Sequence Unique.Sequence.ID
## 1             High         A                  4  AALGVLR                  2
## 2             High         A                  4   NLLLVK                  4
## 3             High         A                  4   LIVVEK                  5
## 4             High         A                  4   LLVDLK                  6
## 5             High         A                  4   IITLLK                  9
## 6             High         A                  4    HEFLR                 10
##   PSM.Ambiguity
## 1   Unambiguous
## 2   Unambiguous
## 3   Unambiguous
## 4   Unambiguous
## 5   Unambiguous
## 6   Unambiguous
##                                                                                                    Protein.Descrip
## 1     Glycine--tRNA ligase beta subunit OS=Escherichia coli (strain K12) GN=glyS PE=1 SV=4 - [SYGB_E
## 2               50S ribosomal protein L3 OS=Escherichia coli (strain K12) GN=rplC PE=1 SV=1 - [RL3_E
## 3               50S ribosomal protein L4 OS=Escherichia coli (strain K12) GN=rplD PE=1 SV=1 - [RL4_E
## 4 Peptidyl-prolyl cis-trans isomerase D OS=Escherichia coli (strain K12) GN=ppiD PE=1 SV=1 - [PPID_E
## 5             3-dehydroquinate synthase OS=Escherichia coli (strain K12) GN=aroB PE=1 SV=1 - [AROB_E
## 6                  GTP cyclohydrolase 1 OS=Escherichia coli (strain K12) GN=folE PE=1 SV=2 - [GCH1_E
##   X..Proteins X..Protein.Groups Protein.Group.Accessions Modifications
## 1           1                 1                   P00961
## 2           1                 1                   P60438
## 3           1                 1                   P60723
## 4           1                 1                   P0ADY1
## 5           1                 1                   P07639
## 6           1                 1                   P0A6T5
##   Activation.Type DeltaScore DeltaCn Rank Search.Engine.Rank Precursor.Area
## 1             CID     1.0000       0    1                  1       3.77e+07
## 2             CID     0.5455       0    1                  1       6.59e+08
## 3             CID     0.0000       0    1                  1       3.83e+08
## 4             CID     0.4062       0    1                  1       1.42e+07
## 5             CID     1.0000       0    1                  1       3.93e+07
## 6             CID     1.0000       0    1                  1       2.80e+07
##   QuanResultID Decoy.Peptides.Matched Exp.Value Homology.Threshold
## 1           NA                     11   0.00033                 13
## 2           NA                      6   0.00940                 13
## 3           NA                     17   0.20000                 13
## 4           NA                      4   0.01300                 13
## 5           NA                     NA   0.00860                 13
## 6           NA                      7   0.27000                 13
##   Identity.High Identity.Middle IonScore Peptides.Matched X..Missed.Cleavages
## 1            13              13       48                5                   0
## 2            13              13       33               11                   0
## 3            13              13       20               19                   0
## 4            13              13       32                6                   0
## 5            13              13       34                5                   0
## 6            13              13       19                4                   0
##   Isolation.Interference.... Ion.Inject.Time..ms. Intensity Charge m.z..Da.
## 1                         53                    4   1700000      2 350.2295
## 2                          8                    2   2520000      2 350.2417
## 3                         38                    5    739000      2 350.7340
## 4                         34                    3   1520000      2 350.7342
## 5                         13                    2   2480000      2 350.7520
## 6                         41                   70     53500      2 351.1900
##   MH...Da. Delta.Mass..Da. Delta.Mass..PPM. RT..min. First.Scan Last.Scan
## 1 699.4517               0             0.68    32.17       8180      8180
## 2 699.4761               0            -0.44    38.77      10907     10907
## 3 700.4607               0             0.41    27.49       6221      6221
## 4 700.4611               0             0.93    43.27      12766     12766
## 5 700.4968               0            -0.03    42.75      12552     12552
## 6 701.3728               0            -0.25    17.39       2693      2693
##   MS.Order Ions.Matched Matched.Ions Total.Ions
## 1      MS2       Jun-50            6         50
## 2      MS2       May-52            5         52
## 3      MS2       May-40            5         40
## 4      MS2       May-40            5         40
## 5      MS2       Apr-40            4         40
## 6      MS2       Apr-32            4         32
##                                       Spectrum.File Annotation
## 1 121219_S_CCES_01_01_LysC_Try_1to10_Mixt_1_1.raw         NA
## 2 121219_S_CCES_01_01_LysC_Try_1to10_Mixt_1_1.raw         NA
## 3 121219_S_CCES_01_01_LysC_Try_1to10_Mixt_1_1.raw         NA
## 4 121219_S_CCES_01_01_LysC_Try_1to10_Mixt_1_1.raw         NA
## 5 121219_S_CCES_01_01_LysC_Try_1to10_Mixt_1_1.raw         NA
## 6 121219_S_CCES_01_01_LysC_Try_1to10_Mixt_1_1.raw         NA
```

One file is for annotation information, required to fill in `Condition` and `BioReplicate` for corresponding `Run` information. Users have to prepare as csv or txt file like 'ControlMixture_DDA_ProteomeDiscoverer_annotation.csv', which includes `Run`, `Condition`, and `BioReplicate` information, and load it in R.

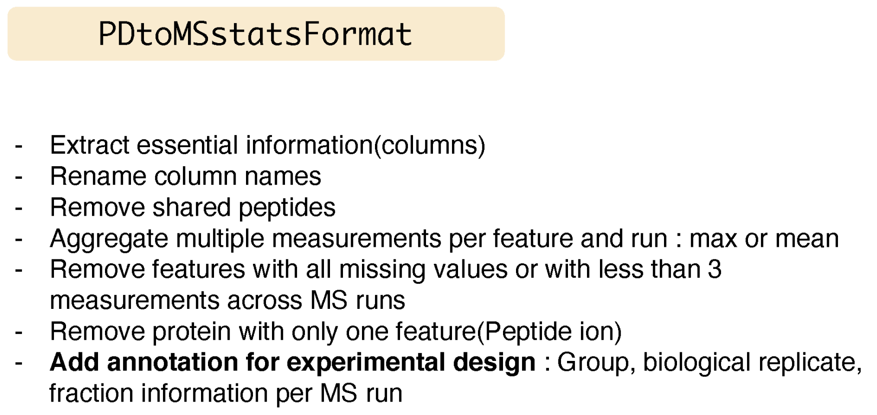
```
## Read in annotation including condition and biological replicates
annot <- read.csv("dda_PD/ControlMixture_DDA_ProteomeDiscoverer_annotation.csv", header = TRUE)
annot
```

```
##                                                Run  Condition BioReplicate
## 1  121219_S_CCES_01_01_LysC_Try_1to10_Mixt_1_1.raw Condition1            1
## 2  121219_S_CCES_01_02_LysC_Try_1to10_Mixt_1_2.raw Condition1            1
## 3  121219_S_CCES_01_03_LysC_Try_1to10_Mixt_1_3.raw Condition1            1
## 4  121219_S_CCES_01_04_LysC_Try_1to10_Mixt_2_1.raw Condition2            2
## 5  121219_S_CCES_01_05_LysC_Try_1to10_Mixt_2_2.raw Condition2            2
## 6  121219_S_CCES_01_06_LysC_Try_1to10_Mixt_2_3.raw Condition2            2
## 7  121219_S_CCES_01_07_LysC_Try_1to10_Mixt_3_1.raw Condition3            3
## 8  121219_S_CCES_01_08_LysC_Try_1to10_Mixt_3_2.raw Condition3            3
## 9  121219_S_CCES_01_09_LysC_Try_1to10_Mixt_3_3.raw Condition3            3
## 10 121219_S_CCES_01_10_LysC_Try_1to10_Mixt_4_1.raw Condition4            4
## 11 121219_S_CCES_01_11_LysC_Try_1to10_Mixt_4_2.raw Condition4            4
## 12 121219_S_CCES_01_12_LysC_Try_1to10_Mixt_4_3.raw Condition4            4
## 13 121219_S_CCES_01_13_LysC_Try_1to10_Mixt_5_1.raw Condition5            5
## 14 121219_S_CCES_01_14_LysC_Try_1to10_Mixt_5_2.raw Condition5            5
## 15 121219_S_CCES_01_15_LysC_Try_1to10_Mixt_5_3.raw Condition5            5
```

4.5.2 Preprocessing with DDA experiment from Proteome Discoverer output

`PDtoMSstatsFormat` function helps pre-processing for making right format of MSstats input from Proteome Discoverer output. `Protein.Group.Accessions` is used for `ProteinName`. The combination of `Sequence` and `Modifications` is used for `PeptideSequence`. `Charge` is used for `PrecursorCharge`. `Precursor.Area` is used for `Intensity`. In addition, there are several steps to filter out or to modify the data in order to get required information.

Here is the summary of pre-processing steps in `PDtoMSstatsFormat` function.



PDtoMSstatsFormat

- Extract essential information(columns)
- Rename column names
- Remove shared peptides
- Aggregate multiple measurements per feature and run : max or mean
- Remove features with all missing values or with less than 3 measurements across MS runs
- Remove protein with only one feature(Peptide ion)
- **Add annotation for experimental design** : Group, biological replicate, fraction information per MS run

Options for `PDtoMSstatsFormat`

- **input** : name of Proteome discover PSM output, which is long-format. "Protein.Group.Accessions", "#Proteins", "Sequence", "Modifications", "Charge", "Intensity", "Spectrum.File" are required.
- **annotation** : name of 'annotation.txt' or 'annotation.csv' data which includes Condition, BioReplicate, and Run information. 'Run' will be matched with 'Spectrum.File'.
- **useNumProteinsColumn** : TRUE removes peptides which have more than 1 in # Proteins column of PD output.
- **useUniquePeptide** : TRUE (default) removes peptides that are assigned for more than one proteins. We assume to use unique peptide for each protein.
- **summaryforMultipleRows** : max(default), sum, or mean. MSstats assumes that there is only one measurement (peak intensity) for one feature and one run. When there are multiple measurements for certain feature and certain run, MSstats need to know which measurements need to be used for further analysis. Users can use highest(max), sum or mean among multiple measurements for one feature and one run.
- **fewMeasurement** : remove or keep the featurew with few measurements.
  - **'remove'** : (default) remove the features that have 1 or 2 measurements across runs.
  - **'keep'** : keep all the features. However, it could generate the error in the step for fitting the statistical model.
- **removeOxidationMpeptides** : TRUE will remove the modified peptides including 'Oxidation (M)' in 'Modifications' column. FALSE is default.
- **removeProtein_with1Peptide** : TRUE will remove the proteins which have only 1 peptide and charge. FALSE is default.
- **which.quantification** : Use 'Precursor.Area'(default) column for quantified intensities. 'Intensity' or 'Area' can be used instead.
- **which.proteinid** : Use 'Protein.Accessions'(default) column for protein name. 'Master.Protein.Accessions' can be used instead.

- **which.sequence** : Use 'Sequence'(default) column for peptide sequence. 'Annotated.Sequence' can be used instead.

For further details, visit the help file using the following code.

```
## check options for converting format
?PDtoMSstatsFormat
```

Now, we use `PDtoMSstatsFormat` function for this example dataset. We chose to remove the proteins with only 1 peptide ion.

```
quant <- PDtoMSstatsFormat(raw,
                           annotation=annot,
                           which.proteinid = 'Protein.Group.Accessions',
                           removeProtein_with1Peptide=TRUE)
```

This function shows the progress. The output of `PDtoMSstatsFormat`, called `quant`, is ready for next step.

```
head(quant)
```

```
##   ProteinName PeptideSequence PrecursorCharge FragmentIon ProductCharge
## 1      P38998    (ac)AAVTLHLR               2          NA            NA
## 2      P38998    (ac)AAVTLHLR               2          NA            NA
## 3      P38998    (ac)AAVTLHLR               2          NA            NA
## 4      P38998    (ac)AAVTLHLR               2          NA            NA
## 5      P38998    (ac)AAVTLHLR               2          NA            NA
## 6      P38998    (ac)AAVTLHLR               2          NA            NA
##   IsotopeLabelType  Condition BioReplicate                   Run Fraction
## 1                L Condition1            1 JD_06232014_sample1-A        1
## 2                L Condition1            1 JD_06232014_sample1_B        1
## 3                L Condition1            1 JD_06232014_sample1_C        1
## 4                L Condition2            2 JD_06232014_sample2_A        1
## 5                L Condition2            2 JD_06232014_sample2_B        1
## 6                L Condition2            2 JD_06232014_sample2_C        1
##   Intensity
## 1        NA
## 2   8792900
## 3  10744000
## 4   7647300
## 5  11407000
## 6  10736000
```

4.5.3 Different options for Proteome Discoverer in dataProcess

Progenesis reports NA for missing values. Therefore,in `dataProcess`, users need to use `censoredInt='NA'`. Users can used the same choice for other options.

```
pd.proposed <- dataProcess(quant,
                           normalization='equalizeMedian',
                           summaryMethod="TMP",
                           cutoffCensored="minFeature",
                           censoredInt="NA", ## !! important
                           MBimpute=TRUE,
                           maxQuantileforCensored=0.999)
```

Further steps is the same as in general workflow (section 4.1).

4.6 Suggested workflow with OpenMS output for DDA

This section describes steps and considerations to properly format data processed by OpenMS, prior to the `MSstats` analysis. In the following example, the raw files for the benchmark dataset (Choi, M. and Eren-Dogu, Z. F. and Colangelo, C. and Cottrell, J. and Hoopmann, M. R. and Kapp, E. A. and Kim, S. and Lam, H. and Neubert, T. A. and Palmblad, M. and Phinney, B. S. and Weintraub, S. T. and MacLean, B. and Vitek, O. 2017) are used. The datasets and details for data processing are available in OpenMS webpage, iPRG 2015 example dataset

4.6.1 Load OpenMS output

Here is the expected input for MSstats, which is output of OpenMS.

```
## Read PSM-level data
raw <- read.csv("dda_openms/iPRG_lfq_input_MSstats.csv")
head(raw)
```

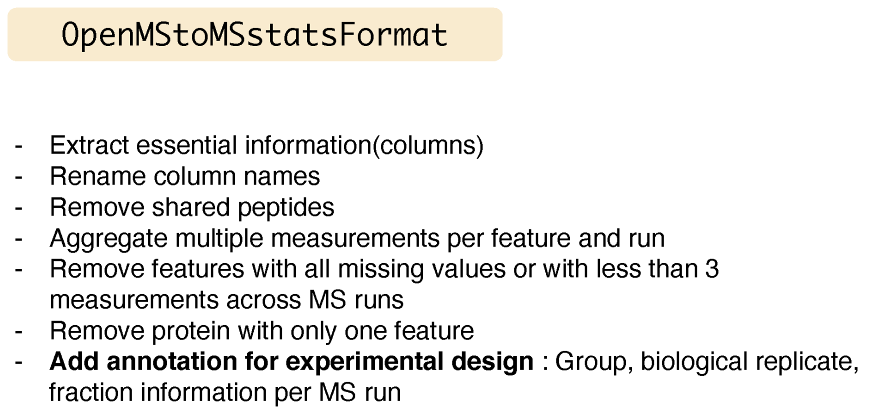
```
##            ProteinName PeptideSequence PrecursorCharge FragmentIon
## 1 sp|P09938|RIR2_YEAST    AAADALSDLEIK               2          NA
## 2 sp|P09938|RIR2_YEAST    AAADALSDLEIK               2          NA
## 3 sp|P09938|RIR2_YEAST    AAADALSDLEIK               2          NA
## 4 sp|P09938|RIR2_YEAST    AAADALSDLEIK               2          NA
## 5 sp|P09938|RIR2_YEAST    AAADALSDLEIK               2          NA
## 6 sp|P09938|RIR2_YEAST    AAADALSDLEIK               2          NA
##   ProductCharge IsotopeLabelType Condition BioReplicate Run Intensity
## 1             0                L         1            1   1 391797000
## 2             0                L         4           10  10 103656000
## 3             0                L         4           11  11 361107000
## 4             0                L         1            2   2 456756000
## 5             0                L         1            3   3 389268000
## 6             0                L         2            4   4 433488000
```

If you follow the workflow in OpenMS KNIME, annotation information should be already filled in.

4.6.2 Preprocessing with DDA experiment from OpenMS output

`OpenMStoMSstatsFormat` function helps pre-processing for making right format of MSstats input from OpenMS output.

Here is the summary of pre-processing steps in `OpenMStoMSstatsFormat` function.


OpenMStoMSstatsFormat

- Extract essential information(columns)
- Rename column names
- Remove shared peptides
- Aggregate multiple measurements per feature and run
- Remove features with all missing values or with less than 3 measurements across MS runs
- Remove protein with only one feature
- **Add annotation for experimental design** : Group, biological replicate, fraction information per MS run

Options for `OpenMStoMSstatsFormat`

- **input** : name of MSstats input report from OpenMS, which includes feature(peptide ion)-level data.

- **annotation** : name of 'annotation.txt' data which includes Raw.file, Condition, BioReplicate, and Run information. If annotation is already complete in OpenMS, use annotation=NULL (default). It will use the annotation information from input.
- **useUniquePeptide** : TRUE (default) removes peptides that are assigned for more than one proteins. We assume to use unique peptide for each protein.
- **summaryforMultipleRows** : max(default), sum, or mean. MSstats assumes that there is only one measurement (peak intensity) for one feature and one run. When there are multiple measurements for certain feature and certain run, MSstats need to know which measurements need to be used for further analysis. Users can use highest(max), sum or mean among multiple measurements for one feature and one run.
- **fewMeasurement** : remove or keep the featurew with few measurements.
  - **'remove'** : (default) remove the features that have 1 or 2 measurements across runs.
  - **'keep'** : keep all the features. However, it could generate the error in the step for fitting the statistical model.
- **removeProtein_with1Feature** : TRUE will remove the proteins which have only 1 peptide and charge. FALSE is default.

For further details, visit the help file using the following code.

```
## check options for converting format
?OpenMStoMSstatsFormat
```

Now, we use `OpenMStoMSstatsFormat` function for this example dataset. We chose to remove the proteins with shared peptides or only 1 peptide ion.

```
quant <- OpenMStoMSstatsFormat(raw,
                               removeProtein_with1Feature = TRUE)
```

```
## INFO  [2023-02-28 10:23:33] ** Raw data from OpenMS imported successfully.
## INFO  [2023-02-28 10:23:34] ** Raw data from OpenMS cleaned successfully.
## INFO  [2023-02-28 10:23:34] ** Using annotation extracted from quantification data.
## INFO  [2023-02-28 10:23:34] ** Run labels were standardized to remove symbols such as '.' or '%'.
## INFO  [2023-02-28 10:23:34] ** The following options are used:
##   - Features will be defined by the columns: PeptideSequence, PrecursorCharge, FragmentIon, ProductCh
##   - Shared peptides will be removed.
##   - Proteins with a single feature will be removed.
##   - Features with less than 3 measurements across runs will be removed.
## INFO  [2023-02-28 10:23:34] ** Features with all missing measurements across runs are removed.
## INFO  [2023-02-28 10:23:34] ** Shared peptides are removed.
## INFO  [2023-02-28 10:23:34] ** Multiple measurements in a feature and a run are summarized by summary
## INFO  [2023-02-28 10:23:34] ** Features with one or two measurements across runs are removed.
## INFO  [2023-02-28 10:23:34] Proteins with a single feature are removed.
## INFO  [2023-02-28 10:23:35] ** Run annotation merged with quantification data.
## INFO  [2023-02-28 10:23:35] ** Features with one or two measurements across runs are removed.
## INFO  [2023-02-28 10:23:35] ** Fractionation handled.
## INFO  [2023-02-28 10:23:35] ** Updated quantification data to make balanced design. Missing values a
## INFO  [2023-02-28 10:23:35] ** Finished preprocessing. The dataset is ready to be processed by the da
```

This function shows the progress. The output of `OpenMStoMSstatsFormat`, called `quant`, is ready for next step.

```
## now 'quant' is ready for MSstats
head(quant)
```

```
##              ProteinName PeptideSequence PrecursorCharge FragmentIon
## 1 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK              2        <NA>
## 2 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK              2        <NA>
## 3 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK              2        <NA>
## 4 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK              2        <NA>
## 5 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK              2        <NA>
## 6 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK              2        <NA>
##   ProductCharge IsotopeLabelType Condition BioReplicate Run Fraction Intensity
## 1             0                L         1            1   1        1  19165300
## 2             0                L         4           10  10        1  20805400
## 3             0                L         4           11  11        1        NA
## 4             0                L         4           12  12        1        NA
## 5             0                L         1            2   2        1        NA
## 6             0                L         1            3   3        1        NA
```

4.6.3 Different options for OpenMS in dataProcess

Progenesis reports NA for missing values. Therefore,in `dataProcess`, users need to use `censoredInt='NA'`. Users can used the same choice for other options.

```
openms.proposed <- dataProcess(quant,
                               normalization='equalizeMedian',
                               summaryMethod="TMP",
                               cutoffCensored="minFeature",
                               censoredInt="NA",
                               MBimpute=TRUE,
                               maxQuantileforCensored=0.999)
```

Further steps is the same as in general workflow (section 4.1).

---

5. DIA analysis with MSstats

5.1 Suggested workflow with Skyline output for DIA

The analysis for DIA with Skyline output is the same as the workflow with Skyline output for DDA. Please check section 4.2 and options form `SkylinetoMSstatsFormat`.

5.2 Suggested workflow with Spectronaut output for DIA

This section describes steps and considerations to properly format data processed by Spectronaut for SWATH/DIA experiments, prior to the `MSstats` analysis. In the following example, the raw files for the benchmark dataset (Bruderer et al. 2015) are quantified by Spectronaut. The datasets and details for data processing are available in MassIVE.quant, MSV000081828, Reanalysis : RMSV000000252.2

5.2.1 Load Spectronaut output

We first load and access the dataset processed by Spectronaut.

```
# Read output from Spectronaut
raw <- read.csv("dia_spectronaut/Bruderer2015_DIA_Spectronaut_input.xls", sep="\t")
```

One file is for annotation information, required to fill in `Condition` and `BioReplicate` for corresponding `Run` information. Users have to prepare as csv or txt file like 'Bruderer2015_DIA_Spectronaut_annotation.csv', which includes `Run`, `Condition`, and `BioReplicate` information, and load it in R.

```r
## Read in annotation including condition and biological replicates
annot <- read.csv("dia_spectronaut/Bruderer2015_DIA_Spectronaut_annotation.csv", header = TRUE)
annot
```

```
##                                  Run Condition BioReplicate
## 1  B_D140314_SGSDSsample1_R01_MHRM        S1           S1
## 2  B_D140314_SGSDSsample1_R02_MHRM        S1           S1
## 3  B_D140314_SGSDSsample1_R03_MHRM        S1           S1
## 4  B_D140314_SGSDSsample2_R01_MHRM        S2           S2
## 5  B_D140314_SGSDSsample2_R02_MHRM        S2           S2
## 6  B_D140314_SGSDSsample2_R03_MHRM        S2           S2
## 7  B_D140314_SGSDSsample3_R01_MHRM        S3           S3
## 8  B_D140314_SGSDSsample3_R02_MHRM        S3           S3
## 9  B_D140314_SGSDSsample3_R03_MHRM        S3           S3
## 10 B_D140314_SGSDSsample4_R01_MHRM        S4           S4
## 11 B_D140314_SGSDSsample4_R02_MHRM        S4           S4
## 12 B_D140314_SGSDSsample4_R03_MHRM        S4           S4
## 13 B_D140314_SGSDSsample5_R01_MHRM        S5           S5
## 14 B_D140314_SGSDSsample5_R02_MHRM        S5           S5
## 15 B_D140314_SGSDSsample5_R03_MHRM        S5           S5
## 16 B_D140314_SGSDSsample6_R01_MHRM        S6           S6
## 17 B_D140314_SGSDSsample6_R02_MHRM        S6           S6
## 18 B_D140314_SGSDSsample6_R03_MHRM        S6           S6
## 19 B_D140314_SGSDSsample7_R01_MHRM        S7           S7
## 20 B_D140314_SGSDSsample7_R02_MHRM        S7           S7
## 21 B_D140314_SGSDSsample7_R03_MHRM        S7           S7
## 22 B_D140314_SGSDSsample8_R01_MHRM        S8           S8
## 23 B_D140314_SGSDSsample8_R02_MHRM        S8           S8
## 24 B_D140314_SGSDSsample8_R03_MHRM        S8           S8
```

5.2.2 Preprocessing with DIA experiment from Spectronaut output

The output from Spectronaut should look like below.

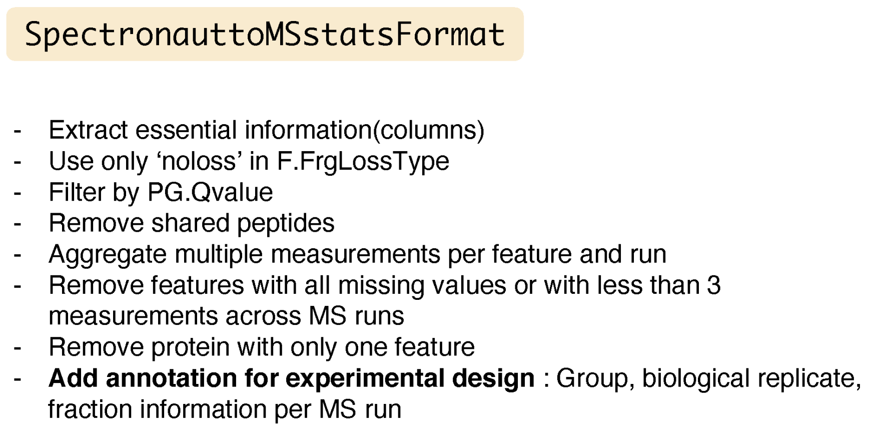
```r
head(raw)
```

```
##    R.Condition                      R.FileName R.Replicate PG.ProteinAccessions
## 1 SGSDSsample1 B_D140314_SGSDSsample1_R01_MHRM           1           A0A0B4J2A2
## 2 SGSDSsample1 B_D140314_SGSDSsample1_R01_MHRM           1           A0A0B4J2A2
## 3 SGSDSsample1 B_D140314_SGSDSsample1_R01_MHRM           1           A0A0B4J2A2
## 4 SGSDSsample1 B_D140314_SGSDSsample1_R01_MHRM           1           A0A0B4J2A2
## 5 SGSDSsample1 B_D140314_SGSDSsample1_R01_MHRM           1           A0A0B4J2A2
## 6 SGSDSsample1 B_D140314_SGSDSsample1_R01_MHRM           1           A0A0B4J2A2
##   PG.ProteinGroups PG.Qvalue PG.Quantity PEP.GroupingKey PEP.StrippedSequence
## 1       A0A0B4J2A2         0     4662586  IIPGFMCQGGDFTR       IIPGFMCQGGDFTR
## 2       A0A0B4J2A2         0     4662586  IIPGFMCQGGDFTR       IIPGFMCQGGDFTR
## 3       A0A0B4J2A2         0     4662586  IIPGFMCQGGDFTR       IIPGFMCQGGDFTR
## 4       A0A0B4J2A2         0     4662586  IIPGFMCQGGDFTR       IIPGFMCQGGDFTR
## 5       A0A0B4J2A2         0     4662586  IIPGFMCQGGDFTR       IIPGFMCQGGDFTR
## 6       A0A0B4J2A2         0     4662586  IIPGFMCQGGDFTR       IIPGFMCQGGDFTR
##   PEP.Quantity EG.iRTPredicted                   EG.Library
## 1      3303864        59.83870 S072_GSDS_DpD_Pulsar_Full.xls
## 2      3303864        59.83870 S072_GSDS_DpD_Pulsar_Full.xls
## 3      3303864        59.83870 S072_GSDS_DpD_Pulsar_Full.xls
## 4      3303864        59.83870 S072_GSDS_DpD_Pulsar_Full.xls
## 5      3303864        59.83870 S072_GSDS_DpD_Pulsar_Full.xls
## 6      3303864        45.20149 S072_GSDS_DpD_Pulsar_Full.xls
##               EG.ModifiedSequence                        EG.PrecursorId
## 1     _IIPGFMC[+C2+H3+N+O]QGGDFTR_     _IIPGFMC[+C2+H3+N+O]QGGDFTR_.2
## 2     _IIPGFMC[+C2+H3+N+O]QGGDFTR_     _IIPGFMC[+C2+H3+N+O]QGGDFTR_.2
## 3     _IIPGFMC[+C2+H3+N+O]QGGDFTR_     _IIPGFMC[+C2+H3+N+O]QGGDFTR_.2
## 4     _IIPGFMC[+C2+H3+N+O]QGGDFTR_     _IIPGFMC[+C2+H3+N+O]QGGDFTR_.2
## 5     _IIPGFMC[+C2+H3+N+O]QGGDFTR_     _IIPGFMC[+C2+H3+N+O]QGGDFTR_.2
## 6 _IIPGFM[+O]C[+C2+H3+N+O]QGGDFTR_ _IIPGFM[+O]C[+C2+H3+N+O]QGGDFTR_.2
##      EG.Qvalue FG.Charge
## 1 3.355320e-12         2
## 2 3.355320e-12         2
## 3 3.355320e-12         2
## 4 3.355320e-12         2
## 5 3.355320e-12         2
## 6 4.455933e-14         2
##
## 1      _IIPGFMC[+C2+H3+N+O]QGGDFTR_;A0A0B4J2A2.2;59.8387F:\\Data\\HRM_GS\\S072_GSDS_DpD_Pulsar_Full.
## 2      _IIPGFMC[+C2+H3+N+O]QGGDFTR_;A0A0B4J2A2.2;59.8387F:\\Data\\HRM_GS\\S072_GSDS_DpD_Pulsar_Full.
## 3      _IIPGFMC[+C2+H3+N+O]QGGDFTR_;A0A0B4J2A2.2;59.8387F:\\Data\\HRM_GS\\S072_GSDS_DpD_Pulsar_Full.
## 4      _IIPGFMC[+C2+H3+N+O]QGGDFTR_;A0A0B4J2A2.2;59.8387F:\\Data\\HRM_GS\\S072_GSDS_DpD_Pulsar_Full.
## 5      _IIPGFMC[+C2+H3+N+O]QGGDFTR_;A0A0B4J2A2.2;59.8387F:\\Data\\HRM_GS\\S072_GSDS_DpD_Pulsar_Full.
## 6 _IIPGFM[+O]C[+C2+H3+N+O]QGGDFTR_;A0A0B4J2A2.2;45.20149F:\\Data\\HRM_GS\\S072_GSDS_DpD_Pulsar_Full.
##   FG.PrecMz FG.Quantity F.Charge F.FrgIon F.FrgLossType  F.FrgMz F.FrgNum
## 1  799.8763   3027798.8        2      y12        noloss 686.7923       12
## 2  799.8763   3027798.8        1       y8        noloss 940.3942        8
## 3  799.8763   3027798.8        1       y6        noloss 652.3049        6
## 4  799.8763   3027798.8        1       y3        noloss 423.2350        3
## 5  799.8763   3027798.8        1       y7        noloss 780.3635        7
## 6  807.8738    257682.8        2      y12        noloss 694.7897       12
##   F.FrgType F.ExcludedFromQuantification F.NormalizedPeakArea
## 1         y                        False            1880069.5
## 2         y                        False             382481.4
## 3         y                        False             331427.1
## 4         y                        False             244019.3
## 5         y                        False             189801.4
## 6         y                        False             197765.4
##   F.NormalizedPeakHeight F.PeakArea F.PeakHeight
## 1              6854814.4  1687479.6    6152623.5
## 2              1320585.1   343300.9    1185307.5
## 3              1221639.0   297476.5    1096497.2
## 4               891761.8   219022.5     800411.9
## 5               673177.2   170358.6     604218.5
## 6               431701.5   174551.3     381027.4
```

The input data for `MSstats` is required to contain variables of `ProteinName`, `PeptideSequence`, `PrecursorCharge`, `FragmentIon`, `ProductCharge`, `IsotopeLabelType`, `Condition`, `BioReplicate`, `Run`, `Intensity`. These variable names should be fixed. Therefore, we need to get subset of useful columns and to rename them. Also several filtering steps are required. `SpectronauttoMSstatsFormat` function helps pre-processing for making right format of MSstats input from Spectronaut output. First, it uses only `noloss` from `F.FrgLossType`. If not, multiple measurements for each feature and run can be happend. Spectronaut provides the column named `F.ExcludedFromQuantification` based on XIC quality such as interference between chromatographies. Only features with `F.ExcludedFromQuantification == 'False'` should be used. `PG.ProteinGroups` is used for `ProteinName`. `EG.ModifiedSequence` is used for `PeptideSequence`.

FG.Charge is used for PrecursorCharge. F.FrgIon is used for FragmentIon. F.Charge is used for ProductCharge. F.PeakArea with default option is used for Intensity. Then several filtering steps will be performed.

Here is the summary of pre-processing steps for SWATH/DIA experiment in SpectronauttoMSstatsFormat function.


SpectronauttoMSstatsFormat

- Extract essential information(columns)
- Use only 'noloss' in F.FrgLossType
- Filter by PG.Qvalue
- Remove shared peptides
- Aggregate multiple measurements per feature and run
- Remove features with all missing values or with less than 3 measurements across MS runs
- Remove protein with only one feature
- **Add annotation for experimental design** : Group, biological replicate, fraction information per MS run

```
## check options for converting format
?SpectronauttoMSstatsFormat
```

```
quant <- SpectronauttoMSstatsFormat(raw,
                                    annotation = annot,
                                    filter_with_Qvalue = TRUE, ## same as default
                                    qvalue_cutoff = 0.01, ## same as default
                                    removeProtein_with1Feature = TRUE)
```

This function shows the progress. The output of SpectronauttoMSstatsFormat, called quant, is ready for next step.

```
## now 'quant' is ready for MSstats
head(quant)
```

```
##            ProteinName PeptideSequence PrecursorCharge FragmentIon
## 1 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK               2        <NA>
## 2 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK               2        <NA>
## 3 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK               2        <NA>
## 4 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK               2        <NA>
## 5 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK               2        <NA>
## 6 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK               2        <NA>
##   ProductCharge IsotopeLabelType Condition BioReplicate Run Fraction Intensity
## 1             0                L         1            1   1        1  19165300
## 2             0                L         4           10  10        1  20805400
## 3             0                L         4           11  11        1        NA
## 4             0                L         4           12  12        1        NA
## 5             0                L         1            2   2        1        NA
## 6             0                L         1            3   3        1        NA
```

5.2.3 Different options for Spectronaut output of DIA experiment in dataProcess

In dataProcess, users need to use censoredInt='0' for Spectronaut output. Spectronaut ouput generates very few number of NA. After applying Qvalue, zero intensities will be generated and those should be imputed.

```
spectronaut.proposed <- dataProcess(quant,
                                    normalization='equalizeMedian',
                                    summaryMethod="TMP",
                                    cutoffCensored="minFeature",
                                    censoredInt="0",
                                    MBimpute=TRUE,
                                    maxQuantileforCensored=0.999)
```

Further steps is the same as in general workflow (section 4.1).

5.3 Suggested workflow with DIA-Umpire output for DIA

This section describes steps and considerations to properly format data processed by DIA-Umpire for SWATH/DIA experiments, prior to the `MSstats` analysis. In the following example, the raw files for the benchmark dataset (Bruderer et al. 2015) are quantified by DIA-Umpire. The datasets and details for data processing are available in MassIVE.quant, MSV000081828, Reanalysis : RMSV000000252.3

5.3.1 Load DIA-Umpire output

We first load and access the dataset processed by DIA-Umpire.

```
# Read output from DIA_Umpire : three output from different levels.
raw.frag <- read.csv('dia_diaumpire/Bruderer2015_DIA_DIAumpire_input_FragSummary.xls', sep="\t")

raw.pep <- read.csv('dia_diaumpire/Bruderer2015_DIA_DIAumpire_input_PeptideSummary.xls', sep="\t")

raw.pro <- read.csv('dia_diaumpire/Bruderer2015_DIA_DIAumpire_input_ProtSummary.xls', sep="\t")
```

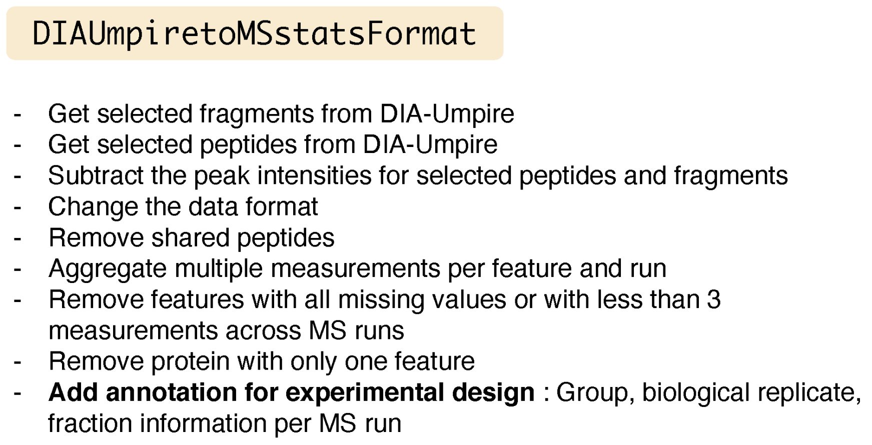
One file is for annotation information, required to fill in `Condition` and `BioReplicate` for corresponding `Run` information. Users have to prepare as csv or txt file like 'Bruderer2015_DIA_DIAumpire_annotation.csv', which includes `Run`, `Condition`, and `BioReplicate` information, and load it in R.

```
## Read in annotation including condition and biological replicates
annot <- read.csv("dia_diaumpire/Bruderer2015_DIA_DIAumpire_annotation.csv", header = TRUE)
annot
```

```
##       Condition BioReplicate                                 Run
## 1   SGSDSsample1 SGSDSsample1 B_D140314_SGSDSsample1_R01_MHRM_T0
## 2   SGSDSsample1 SGSDSsample1 B_D140314_SGSDSsample1_R02_MHRM_T0
## 3   SGSDSsample1 SGSDSsample1 B_D140314_SGSDSsample1_R03_MHRM_T0
## 4   SGSDSsample2 SGSDSsample2 B_D140314_SGSDSsample2_R01_MHRM_T0
## 5   SGSDSsample2 SGSDSsample2 B_D140314_SGSDSsample2_R02_MHRM_T0
## 6   SGSDSsample2 SGSDSsample2 B_D140314_SGSDSsample2_R03_MHRM_T0
## 7   SGSDSsample3 SGSDSsample3 B_D140314_SGSDSsample3_R01_MHRM_T0
## 8   SGSDSsample3 SGSDSsample3 B_D140314_SGSDSsample3_R02_MHRM_T0
## 9   SGSDSsample3 SGSDSsample3 B_D140314_SGSDSsample3_R03_MHRM_T0
## 10 SGSDSsample4 SGSDSsample4 B_D140314_SGSDSsample4_R01_MHRM_T0
## 11 SGSDSsample4 SGSDSsample4 B_D140314_SGSDSsample4_R02_MHRM_T0
## 12 SGSDSsample4 SGSDSsample4 B_D140314_SGSDSsample4_R03_MHRM_T0
## 13 SGSDSsample5 SGSDSsample5 B_D140314_SGSDSsample5_R01_MHRM_T0
## 14 SGSDSsample5 SGSDSsample5 B_D140314_SGSDSsample5_R02_MHRM_T0
## 15 SGSDSsample5 SGSDSsample5 B_D140314_SGSDSsample5_R03_MHRM_T0
## 16 SGSDSsample6 SGSDSsample6 B_D140314_SGSDSsample6_R01_MHRM_T0
## 17 SGSDSsample6 SGSDSsample6 B_D140314_SGSDSsample6_R02_MHRM_T0
## 18 SGSDSsample6 SGSDSsample6 B_D140314_SGSDSsample6_R03_MHRM_T0
## 19 SGSDSsample7 SGSDSsample7 B_D140314_SGSDSsample7_R01_MHRM_T0
## 20 SGSDSsample7 SGSDSsample7 B_D140314_SGSDSsample7_R02_MHRM_T0
## 21 SGSDSsample7 SGSDSsample7 B_D140314_SGSDSsample7_R03_MHRM_T0
## 22 SGSDSsample8 SGSDSsample8 B_D140314_SGSDSsample8_R01_MHRM_T0
## 23 SGSDSsample8 SGSDSsample8 B_D140314_SGSDSsample8_R02_MHRM_T0
## 24 SGSDSsample8 SGSDSsample8 B_D140314_SGSDSsample8_R03_MHRM_T0
```

5.3.2 Preprocessing with DIA experiment from DIA-Umpire output

Here is the summary of pre-processing steps for DIA experiment in `DIAUmpiretoMSstatsFormat` function.



DIAUmpiretoMSstatsFormat

- Get selected fragments from DIA-Umpire
- Get selected peptides from DIA-Umpire
- Subtract the peak intensities for selected peptides and fragments
- Change the data format
- Remove shared peptides
- Aggregate multiple measurements per feature and run
- Remove features with all missing values or with less than 3 measurements across MS runs
- Remove protein with only one feature
- **Add annotation for experimental design** : Group, biological replicate, fraction information per MS run

Options for `DIAUmpiretoMSstatsFormat`

- **raw.frag** : name of FragSummary_date.xls data, which includes feature-level data.
- **raw.pep** : name of PeptideSummary_date.xls data, which includes selected fragments information.
- **raw.pro** : name of ProteinSummary_date.xls data, which includes selected peptides information.
- **annotation** : name of 'annotation.txt' data which includes Raw.file, Condition, BioReplicate, and Run information.
- **useSelectedFrag** : TRUE (default) will use the selected fragment for each peptide. 'Selected_fragments' column is required.
- **useSelectedPep** : TRUE (default) will use the selected peptide for each protein. 'Selected_peptides' column is required.
- **summaryforMultipleRows** : max(default), sum, or mean. MSstats assumes that there is only one measurement (peak intensity) for one feature and one run. When there are multiple measurements for certain feature and certain run, MSstats need to know which measurements need to be used for further analysis. Users can use highest(max), sum or mean among multiple measurements for one feature and one run.
- **fewMeasurement** : remove or keep the featurew with few measurements.
  - **'remove'** : (default) remove the features that have 1 or 2 measurements across runs.
  - **'keep'** : keep all the features. However, it could generate the error in the step for fitting the statistical model.
- **removeProtein_with1Feature** : TRUE will remove the proteins which have only 1 feature, which is the combination of peptide, precursor charge, fragment and charge. FALSE is default.

For further details, visit the help file using the following code.

```
## check options for converting format
?DIAUmpiretoMSstatsFormat
```

```
quant <- DIAUmpiretoMSstatsFormat(raw.frag, raw.pep, raw.pro,
                                  annot,
                                  useSelectedFrag = TRUE,
                                  useSelectedPep = FALSE,
                                  removeProtein_with1Feature = TRUE)
```

This function shows the progress. The output of `DIAUmpiretoMSstatsFormat`, called `quant`, is ready for next step.

```
## now 'quant' is ready for MSstats
head(quant)
```

```
##              ProteinName PeptideSequence PrecursorCharge FragmentIon
## 1 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK               2        <NA>
## 2 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK               2        <NA>
## 3 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK               2        <NA>
## 4 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK               2        <NA>
## 5 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK               2        <NA>
## 6 sp|P09938|RIR2_YEAST AAADALSDLEIKDSK               2        <NA>
##   ProductCharge IsotopeLabelType Condition BioReplicate Run Fraction Intensity
## 1             0                L         1            1   1        1  19165300
## 2             0                L         4           10  10        1  20805400
## 3             0                L         4           11  11        1        NA
## 4             0                L         4           12  12        1        NA
## 5             0                L         1            2   2        1        NA
## 6             0                L         1            3   3        1        NA
```

5.3.3 Different options for DIA-Umpire output of DIA experiment in dataProcess

In `dataProcess`, users need to use `censoredInt='NA'` for DIA-Umpire output.

```
diaumpire.proposed <- dataProcess(quant,
                                  normalization='equalizeMedian',
                                  summaryMethod="TMP",
                                  cutoffCensored="minFeature",
                                  censoredInt="NA", ## !! important
                                  MBimpute=TRUE,
                                  maxQuantileforCensored=0.999)
```

Further steps is the same as in general workflow (section 4.1).

5.4 Suggested workflow with OpenSWATH output for SWATH

This section describes steps and considerations to properly format data processed by OpenSWATH for SWATH experiments, prior to the `MSstats` analysis. In the following example, the dataset processed and quantified by OpenSWATH and available as supplementary in (Röst et al. 2014) is used. The datasets and details for data processing are available in MassIVE.quant, MSV000081829, Reanalysis : RMSV000000253.2

5.4.1 Load OpenSWATH output

```
## Read fragment-level data
raw <- read.csv("dia_openswath/Rost2014_DIA_OpenSWATH_input.txt", sep="\t")
head(raw)
```

```
##                                                                                              transition_group_
## 1          AQUA4SWATH_YeastB_GSMADVPK(UniMod:259)/2_run0_split_napedro_L120417_001_SW_combined.featureI
## 2 AQUA4SWATH_HMLangeA_APIPTALDTDSSK(UniMod:259)/2_run0_split_napedro_L120417_001_SW_combined.featureI
## 3   AQUA4SWATH_HMLangeA_DITAFDETLFR(UniMod:267)/2_run0_split_napedro_L120417_001_SW_combined.featureI
## 4   AQUA4SWATH_HMLangeA_LNTIYQNDLTK(UniMod:259)/2_run0_split_napedro_L120417_001_SW_combined.featureI
## 5 AQUA4SWATH_HMLangeB_GDSSLLLAVTEVK(UniMod:259)/2_run0_split_napedro_L120417_001_SW_combined.featureI
## 6 AQUA4SWATH_HMLangeB_ITVDDSDQGANAK(UniMod:259)/2_run0_split_napedro_L120417_001_SW_combined.featureI
##   decoy main_var_xx_swath_prelim_score var_bseries_score
## 1 FALSE                      1.2878970                 1
## 2 FALSE                      0.1706103                 5
## 3 FALSE                      1.1339196                 2
## 4 FALSE                      1.7598289                 2
## 5 FALSE                     -0.4115174                 2
## 6 FALSE                      1.0441584                 1
##   var_elution_model_fit_score var_intensity_score var_isotope_correlation_score
## 1                   0.9342550         0.453363758                     0.9621812
## 2                   0.8913419         0.049083725                     0.6132449
## 3                   0.9999998         0.009851618                     0.3554128
## 4                   0.9159949         0.095405653                     0.9065832
## 5                   0.9574775         0.021453197                     0.9215580
## 6                   0.9746000         0.022167918                     0.4878706
##   var_isotope_overlap_score var_library_corr var_library_rmsd var_log_sn_score
## 1                0.00000000       -0.1055161       0.27253090        2.6731795
## 2                0.06965649       -0.5652690       0.30030363        0.4186867
## 3                0.22701794       -0.6904735       0.07683261        0.7115797
## 4                0.10497624        0.9654186       0.21724993        2.2435553
## 5                0.02678354       -0.2470041       0.30010703        0.3742150
## 6                0.84251969        0.8373865       0.08390017        1.5353173
##   var_massdev_score var_massdev_score_weighted var_norm_rt_score
## 1         10.605493                   8.722201       0.037829778
## 2          3.506636                   1.525251       0.053104479
## 3          1.922673                   1.770099       0.059778618
## 4          6.326571                   7.671188       0.037207495
## 5         10.654442                  13.899227       0.027236154
## 6          5.657834                   3.023133       0.007901597
##   var_xcorr_coelution var_xcorr_coelution_weighted var_xcorr_shape
## 1           3.3763883                    1.9254634       0.8278485
## 2           2.9055453                    0.9019211       0.7211294
## 3           0.9163978                    0.4743998       0.8000000
## 4           2.6757296                    1.1959883       0.8199334
## 5           1.8944289                    0.7742008       0.8040568
## 6           1.8595018                    0.2477762       0.8121000
##   var_xcorr_shape_weighted var_yseries_score
## 1                0.7850388                 2
## 2                0.7406344                 3
## 3                0.7628001                 3
## 4                0.7884498                 2
## 5                0.8094922                 0
## 6                0.8723227                 0
##                                                                                              transition_group_
## 1        AQUA4SWATH_YeastB_GSMADVPK(UniMod:259)/2_run0_split_napedro_L120417_001_SW_combined.featureX
## 2 AQUA4SWATH_HMLangeA_APIPTALDTDSSK(UniMod:259)/2_run0_split_napedro_L120417_001_SW_combined.featureX
## 3   AQUA4SWATH_HMLangeA_DITAFDETLFR(UniMod:267)/2_run0_split_napedro_L120417_001_SW_combined.featureX
## 4   AQUA4SWATH_HMLangeA_LNTIYQNDLTK(UniMod:259)/2_run0_split_napedro_L120417_001_SW_combined.featureX
## 5 AQUA4SWATH_HMLangeB_GDSSLLLAVTEVK(UniMod:259)/2_run0_split_napedro_L120417_001_SW_combined.featureX
## 6 AQUA4SWATH_HMLangeB_ITVDDSDQGANAK(UniMod:259)/2_run0_split_napedro_L120417_001_SW_combined.featureX
##   run_id                                         filename       RT
## 1      0 split_napedro_L120417_001_SW_combined.featureXML 1024.232
## 2      0 split_napedro_L120417_001_SW_combined.featureXML 2525.123
## 3      0 split_napedro_L120417_001_SW_combined.featureXML 4923.724
## 4      0 split_napedro_L120417_001_SW_combined.featureXML 2396.661
## 5      0 split_napedro_L120417_001_SW_combined.featureXML 4217.872
## 6      0 split_napedro_L120417_001_SW_combined.featureXML 1058.414
##                     id      Sequence           FullPeptideName Charge     m.z
## 1  f_1353009549277083696      GSMADVPK      GSMADVPK(UniMod:259)      2 406.707
## 2 f_17169785622655779335 APIPTALDTDSSK APIPTALDTDSSK(UniMod:259)      2 662.348
## 3 f_14843615568932246264   DITAFDETLFR   DITAFDETLFR(UniMod:267)      2 669.334
## 4  f_2705275134670444755   LNTIYQNDLTK   LNTIYQNDLTK(UniMod:259)      2 665.858
## 5  f_9381457823485960609 GDSSLLLAVTEVK GDSSLLLAVTEVK(UniMod:259)      2 670.382
## 6  f_4286219334306522917 ITVDDSDQGANAK ITVDDSDQGANAK(UniMod:259)      2 671.322
##   Intensity         ProteinName assay_rt   delta_rt leftWidth    norm_RT
## 1    228484   AQUA4SWATH_YeastB 1160.081 -135.84854   1003.92 -19.18298
## 2     11528 AQUA4SWATH_HMLangeA 2343.868  181.25584   2513.89  24.41045
## 3      1784 AQUA4SWATH_HMLangeA 4711.441  212.28267   4920.69  94.07786
## 4     41457 AQUA4SWATH_HMLangeA 2525.725 -129.06375   2384.16  20.67925
## 5      4107 AQUA4SWATH_HMLangeB 4306.552  -88.67945   4214.01  73.57638
## 6       762 AQUA4SWATH_HMLangeB 1091.456  -33.04119   1045.90 -18.19016
##   nr_peaks peak_apices_sum rightWidth  rt_score  sn_ratio total_xic
## 1        4           23302    1061.95 3.7829778 14.485954    503975
## 2        4            1673    2537.79 5.3104479  1.519964    234864
## 3        4             902    4924.10 5.9778618  2.037207    181087
## 4        4            8532    2411.47 3.7207495  9.426787    434534
## 5        4            1065    4227.67 2.7236154  1.453850    191440
## 6        4             291    1062.97 0.7901597  4.642799     34374
##   dotprod_score library_dotprod library_manhattan manhatt_score
## 1     0.7223189       0.7550611         0.6275367     0.7230707
## 2     0.6545768       0.7396598         0.6218496     0.8495160
## 3     0.7349210       0.9816014         0.1471980     0.7705343
## 4     0.7527736       0.8719077         0.5787663     0.7852044
## 5     0.6493268       0.7414075         0.6934909     0.8909054
## 6     0.4477453       0.9752496         0.2175255     1.0719205
##   xx_lda_prelim_score xx_swath_prelim_score
## 1            3.488009                     0
## 2            1.007469                     0
## 3            2.232398                     0
## 4            3.770504                     0
## 5            1.713495                     0
## 6            3.736117                     0
##                                     aggr_Peak_Area aggr_Peak_Apex
## 1 153339.000000;67621.000000;5210.000000;2314.000000    NA;NA;NA;NA
## 2     803.000000;6614.000000;2073.000000;2038.000000    NA;NA;NA;NA
## 3        520.000000;403.000000;405.000000;456.000000    NA;NA;NA;NA
## 4    1849.000000;37105.000000;410.000000;2093.000000    NA;NA;NA;NA
## 5        904.000000;110.000000;3073.000000;20.000000    NA;NA;NA;NA
## 6        120.000000;180.000000;140.000000;322.000000    NA;NA;NA;NA
##
## 1                              AQUA4SWATH_YeastB_GSMADVPK(UniMod:259)/2_y4;AQUA4SWATH_YeastB_GSMADVI
## 2  AQUA4SWATH_HMLangeA_APIPTALDTDSSK(UniMod:259)/2_y10;AQUA4SWATH_HMLangeA_APIPTALDTDSSK(UniMod:259)/
## 3           AQUA4SWATH_HMLangeA_DITAFDETLFR(UniMod:267)/2_y6;AQUA4SWATH_HMLangeA_DITAFDETLFR(UniMod:2
## 4           AQUA4SWATH_HMLangeA_LNTIYQNDLTK(UniMod:259)/2_y8;AQUA4SWATH_HMLangeA_LNTIYQNDLTK(UniMod:2
## 5   AQUA4SWATH_HMLangeB_GDSSLLLAVTEVK(UniMod:259)/2_y7;AQUA4SWATH_HMLangeB_GDSSLLLAVTEVK(UniMod:259)/
## 6 AQUA4SWATH_HMLangeB_ITVDDSDQGANAK(UniMod:259)/2_y9;AQUA4SWATH_HMLangeB_ITVDDSDQGANAK(UniMod:259)/2
##   log10_total_xic        LD1 peak_group_rank    d_score   m_score
## 1        5.702409 -1.9493868               1 -0.6243745 0.4629052
## 2        5.370816 -2.4461163               1 -1.1333458 0.5062006
## 3        5.257887 -1.0880511               1  0.2581884 0.3345616
## 4        5.638024 -0.9814508               1  0.3674158 0.3138670
## 5        5.282033 -2.4287808               1 -1.1155831 0.5038482
## 6        4.536230 -1.7118092               1 -0.3809420 0.4358001
```

One file is for annotation information, required to fill in `Condition` and `BioReplicate` for corresponding `Run` information. Users have to prepare as csv or txt file like 'Rost2014_DIA_OpenSWATH_annotation.csv', which includes `Run`, `Condition`, and `BioReplicate` information, and load it in R.

```
## Read in annotation including condition and biological replicates: ControlMixture_DDA_ProteomeDiscove
annot <- read.csv("dia_openswath/Rost2014_DIA_OpenSWATH_annotation.csv", header = TRUE)
annot
```

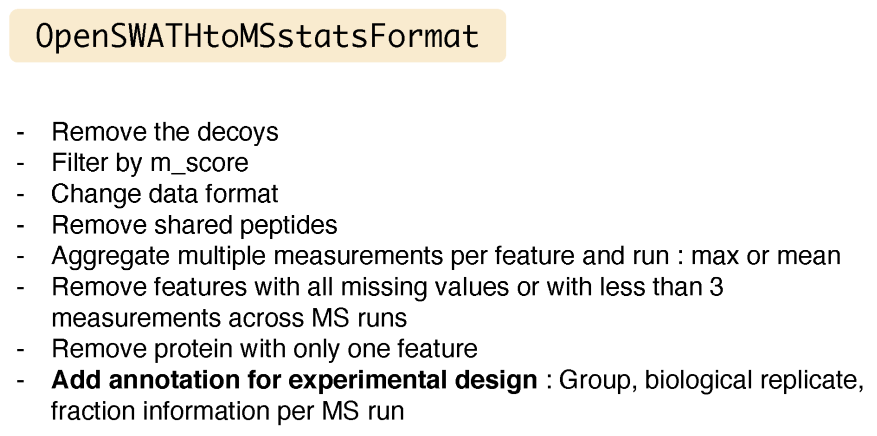
```
##                                         Filename Condition BioReplicate
## 1  split_napedro_L120417_001_SW_combined.featureXML       512          512
## 2  split_napedro_L120417_002_SW_combined.featureXML       256          256
## 3  split_napedro_L120417_003_SW_combined.featureXML       128          128
## 4  split_napedro_L120417_004_SW_combined.featureXML        64           64
## 5  split_napedro_L120417_005_SW_combined.featureXML        32           32
## 6  split_napedro_L120417_006_SW_combined.featureXML        16           16
## 7  split_napedro_L120417_007_SW_combined.featureXML         8            8
## 8  split_napedro_L120417_008_SW_combined.featureXML         4            4
## 9  split_napedro_L120417_009_SW_combined.featureXML         2            2
## 10 split_napedro_L120417_010_SW_combined.featureXML         1            1
## 11 split_napedro_L120419_001_SW_combined.featureXML       512          512
## 12 split_napedro_L120419_002_SW_combined.featureXML       256          256
## 13 split_napedro_L120419_003_SW_combined.featureXML       128          128
## 14 split_napedro_L120419_004_SW_combined.featureXML        64           64
## 15 split_napedro_L120419_005_SW_combined.featureXML        32           32
## 16 split_napedro_L120419_006_SW_combined.featureXML        16           16
## 17 split_napedro_L120419_007_SW_combined.featureXML         8            8
## 18 split_napedro_L120419_008_SW_combined.featureXML         4            4
## 19 split_napedro_L120419_009_SW_combined.featureXML         2            2
## 20 split_napedro_L120419_010_SW_combined.featureXML         1            1
## 21 split_napedro_L120420_001_SW_combined.featureXML       512          512
## 22 split_napedro_L120420_002_SW_combined.featureXML       256          256
## 23 split_napedro_L120420_003_SW_combined.featureXML       128          128
## 24 split_napedro_L120420_004_SW_combined.featureXML        64           64
## 25 split_napedro_L120420_005_SW_combined.featureXML        32           32
## 26 split_napedro_L120420_006_SW_combined.featureXML        16           16
## 27 split_napedro_L120420_007_SW_combined.featureXML         8            8
## 28 split_napedro_L120420_008_SW_combined.featureXML         4            4
## 29 split_napedro_L120420_009_SW_combined.featureXML         2            2
## 30 split_napedro_L120420_010_SW_combined.featureXML         1            1
##                                                  Run
## 1  split_napedro_L120417_001_SW_combined.featureXML
## 2  split_napedro_L120417_002_SW_combined.featureXML
## 3  split_napedro_L120417_003_SW_combined.featureXML
## 4  split_napedro_L120417_004_SW_combined.featureXML
## 5  split_napedro_L120417_005_SW_combined.featureXML
## 6  split_napedro_L120417_006_SW_combined.featureXML
## 7  split_napedro_L120417_007_SW_combined.featureXML
## 8  split_napedro_L120417_008_SW_combined.featureXML
## 9  split_napedro_L120417_009_SW_combined.featureXML
## 10 split_napedro_L120417_010_SW_combined.featureXML
## 11 split_napedro_L120419_001_SW_combined.featureXML
## 12 split_napedro_L120419_002_SW_combined.featureXML
## 13 split_napedro_L120419_003_SW_combined.featureXML
## 14 split_napedro_L120419_004_SW_combined.featureXML
## 15 split_napedro_L120419_005_SW_combined.featureXML
## 16 split_napedro_L120419_006_SW_combined.featureXML
## 17 split_napedro_L120419_007_SW_combined.featureXML
## 18 split_napedro_L120419_008_SW_combined.featureXML
## 19 split_napedro_L120419_009_SW_combined.featureXML
## 20 split_napedro_L120419_010_SW_combined.featureXML
## 21 split_napedro_L120420_001_SW_combined.featureXML
## 22 split_napedro_L120420_002_SW_combined.featureXML
## 23 split_napedro_L120420_003_SW_combined.featureXML
## 24 split_napedro_L120420_004_SW_combined.featureXML
## 25 split_napedro_L120420_005_SW_combined.featureXML
## 26 split_napedro_L120420_006_SW_combined.featureXML
## 27 split_napedro_L120420_007_SW_combined.featureXML
## 28 split_napedro_L120420_008_SW_combined.featureXML
## 29 split_napedro_L120420_009_SW_combined.featureXML
## 30 split_napedro_L120420_010_SW_combined.featureXML
```

5.4.2 Preprocessing with DIA experiment from OpenSWATH output

The output from OpenSWATH should look like below.

```
head(raw)
```

Here is the summary of pre-processing steps for SWATH/DIA experiment in `OpenSWATHtoMSstatsFormat` function.



OpenSWATHtoMSstatsFormat

- Remove the decoys
- Filter by m_score
- Change data format
- Remove shared peptides
- Aggregate multiple measurements per feature and run : max or mean
- Remove features with all missing values or with less than 3 measurements across MS runs
- Remove protein with only one feature
- **Add annotation for experimental design** : Group, biological replicate, fraction information per MS run

Options for `OpenSWATHtoMSstatsFormat`

- **input** : name of MSstats input report from OpenSWATH, which includes feature-level data.
- **annotation** :name of 'annotation.txt' data which includes Raw.file, Condition, BioReplicate, and Run information. Run should be the same as 'filename'.
- **filter_with_mscore** : TRUE (default) will filter out the features that have greater than `mscore_cutoff` in m_score column. Those features will be removed.
- **mscore_cutoff** : Cutoff for `m_score`. default is 0.01.
- **useUniquePeptide** : TRUE (default) removes peptides that are assigned for more than one proteins. We assume to use unique peptide for each protein.
- **summaryforMultipleRows** : max(default), sum, or mean. MSstats assumes that there is only one measurement (peak intensity) for one feature and one run. When there are multiple measurements for certain feature and certain run, MSstats need to know which measurements need to be used for further analysis. Users can use highest(max), sum or mean among multiple measurements for one feature and one run.
- **fewMeasurement** : remove or keep the featurew with few measurements.
  - **'remove'** : (default) remove the features that have 1 or 2 measurements across runs.
  - **'keep'** : keep all the features. However, it could generate the error in the step for fitting the statistical model.
- **removeProtein_with1Feature** : TRUE will remove the proteins which have only 1 peptide and charge. FALSE is default.

For further details, visit the help file using the following code.

```
## check options for converting format
?OpenSWATHtoMSstatsFormat
```

Now, we use `OpenSWATHtoMSstatsFormat` function for this example dataset. We chose to remove the proteins with only 1 feature.

```
quant <- OpenSWATHtoMSstatsFormat(raw,
                                  annotation = annot,
                                  filter_with_mscore = TRUE, ## same as default
                                  mscore_cutoff = 0.01, ## same as default
                                  removeProtein_with1Feature = TRUE)
```

```
## INFO  [2023-02-28 10:31:22] ** Raw data from OpenSWATH imported successfully.
## INFO  [2023-02-28 10:31:22] ** Raw data from OpenSWATH cleaned successfully.
## INFO  [2023-02-28 10:31:22] ** Using provided annotation.
## INFO  [2023-02-28 10:31:22] ** Run labels were standardized to remove symbols such as '.' or '%'.
## INFO  [2023-02-28 10:31:22] ** The following options are used:
##   - Features will be defined by the columns: PeptideSequence, PrecursorCharge, FragmentIon
##   - Shared peptides will be removed.
##   - Proteins with a single feature will be removed.
##   - Features with less than 3 measurements across runs will be removed.
## INFO  [2023-02-28 10:31:22] ** Rows with values of decoy equal to 1 are removed
## INFO  [2023-02-28 10:31:22] ** Rows with values not smaller than 0.01 in m_score are removed
## INFO  [2023-02-28 10:31:22] ** Features with all missing measurements across runs are removed.
## INFO  [2023-02-28 10:31:22] ** Shared peptides are removed.
```

```
## Warning in aggregator(Intensity, na.rm = TRUE): no non-missing arguments to
## max; returning -Inf
## Warning in aggregator(Intensity, na.rm = TRUE): no non-missing arguments to
## max; returning -Inf

## Warning in aggregator(Intensity, na.rm = TRUE): no non-missing arguments to
## max; returning -Inf

## Warning in aggregator(Intensity, na.rm = TRUE): no non-missing arguments to
## max; returning -Inf

## Warning in aggregator(Intensity, na.rm = TRUE): no non-missing arguments to
## max; returning -Inf

## Warning in aggregator(Intensity, na.rm = TRUE): no non-missing arguments to
## max; returning -Inf

## Warning in aggregator(Intensity, na.rm = TRUE): no non-missing arguments to
## max; returning -Inf

## Warning in aggregator(Intensity, na.rm = TRUE): no non-missing arguments to
## max; returning -Inf

## INFO  [2023-02-28 10:31:22] ** Multiple measurements in a feature and a run are summarized by summary
## INFO  [2023-02-28 10:31:22] ** Features with one or two measurements across runs are removed.
## INFO  [2023-02-28 10:31:22] Proteins with a single feature are removed.
## INFO  [2023-02-28 10:31:22] ** Run annotation merged with quantification data.
## INFO  [2023-02-28 10:31:22] ** Features with one or two measurements across runs are removed.
## INFO  [2023-02-28 10:31:22] ** Fractionation handled.
## INFO  [2023-02-28 10:31:22] ** Updated quantification data to make balanced design. Missing values ar
## INFO  [2023-02-28 10:31:22] ** Finished preprocessing. The dataset is ready to be processed by the da
```

This function shows the progress. The output of `OpenMStoMSstatsFormat`, called `quant`, is ready for next step.

```
## now 'quant' is ready for MSstats
head(quant)
```

```
##           ProteinName       PeptideSequence PrecursorCharge
## 1 AQUA4SWATH_HMLangeE AAEDFTLLVK(UniMod259)               2
## 2 AQUA4SWATH_HMLangeE AAEDFTLLVK(UniMod259)               2
## 3 AQUA4SWATH_HMLangeE AAEDFTLLVK(UniMod259)               2
## 4 AQUA4SWATH_HMLangeE AAEDFTLLVK(UniMod259)               2
## 5 AQUA4SWATH_HMLangeE AAEDFTLLVK(UniMod259)               2
## 6 AQUA4SWATH_HMLangeE AAEDFTLLVK(UniMod259)               2
##                                  FragmentIon ProductCharge IsotopeLabelType
## 1 AQUA4SWATHHMLangeEAAEDFTLLVK(UniMod259)/2y5          <NA>                L
## 2 AQUA4SWATHHMLangeEAAEDFTLLVK(UniMod259)/2y5          <NA>                L
## 3 AQUA4SWATHHMLangeEAAEDFTLLVK(UniMod259)/2y5          <NA>                L
## 4 AQUA4SWATHHMLangeEAAEDFTLLVK(UniMod259)/2y5          <NA>                L
## 5 AQUA4SWATHHMLangeEAAEDFTLLVK(UniMod259)/2y5          <NA>                L
## 6 AQUA4SWATHHMLangeEAAEDFTLLVK(UniMod259)/2y5          <NA>                L
##   Condition BioReplicate                                            Run
## 1       512          512 split_napedro_L120417_001_SW_combinedfeatureXML
## 2       256          256 split_napedro_L120417_002_SW_combinedfeatureXML
## 3       128          128 split_napedro_L120417_003_SW_combinedfeatureXML
## 4        64           64 split_napedro_L120417_004_SW_combinedfeatureXML
## 5        32           32 split_napedro_L120417_005_SW_combinedfeatureXML
## 6        16           16 split_napedro_L120417_006_SW_combinedfeatureXML
##   Fraction Intensity
## 1        1         0
## 2        1         0
## 3        1         0
## 4        1         0
## 5        1         0
## 6        1         0
```

5.4.3 Different options for OpenSWATH output of DIA experiment in dataProcess

In `dataProcess`, users need to use `censoredInt='0'` for OpenSWATH output.

```
goldstandard.proposed <- dataProcess(quant,
                                     normalization='equalizeMedian',
                                     summaryMethod="TMP",
                                     cutoffCensored="minFeature",
                                     censoredInt="0",
                                     MBimpute=TRUE,
                                     maxQuantileforCensored=0.999)
```

Further steps is the same as in general workflow (section 4.1).

---

6. SRM analysis with MSstats

6.1 Suggested workflow for SRM

This section describes a typical workflow for SRM experiments with heavy labeled-isotope peptides. The example dataset, `SRMRawData` in MSstats is used for demonstration.

6.1.1 Preparing the data for MSstats input

The first step in using the `MSstats` is to format the data as described in Section 2. `SRMRawData` is already formatted for `MSstats` input.

```
# Check the first 6 rows in SRMRawData
head(SRMRawData)
```

```
##     ProteinName PeptideSequence PrecursorCharge FragmentIon ProductCharge
## 243        IDHC   ATDVIVPEEGELR               2          y7            NA
## 244        IDHC   ATDVIVPEEGELR               2          y7            NA
## 245        IDHC   ATDVIVPEEGELR               2          y8            NA
## 246        IDHC   ATDVIVPEEGELR               2          y8            NA
## 247        IDHC   ATDVIVPEEGELR               2          y9            NA
## 248        IDHC   ATDVIVPEEGELR               2          y9            NA
##     IsotopeLabelType Condition BioReplicate Run   Intensity
## 243                H         1        ReplA   1 84361.08350
## 244                L         1        ReplA   1   215.13526
## 245                H         1        ReplA   1 29778.10188
## 246                L         1        ReplA   1    98.02134
## 247                H         1        ReplA   1 17921.29255
## 248                L         1        ReplA   1    60.47029
```

6.1.2 Processing the data

It is the same workflow as described in section 4.1.2. Only difference is the normalization with heavy labeled isotope peptides.

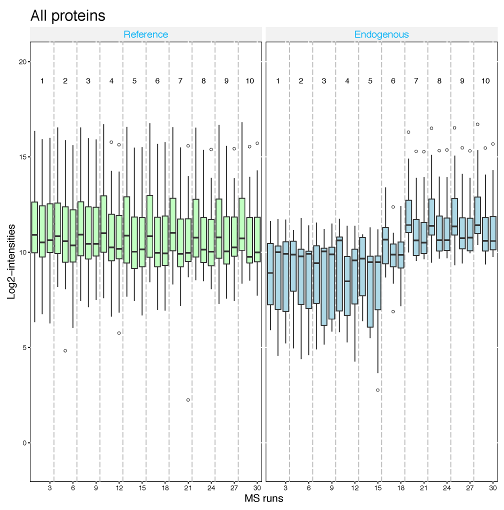
###$ Different normalization option Let's see the different normalization effect with SRM dataset including two proteins.

```
unique(SRMRawData$ProteinName)
```

```
## [1] IDHC PMG2
## 45 Levels: ACEA ACH1 ACON ADH1 ADH2 ADH4 ALDH6 ALF CISY1 CISY2 DHSA ... SUCB
```

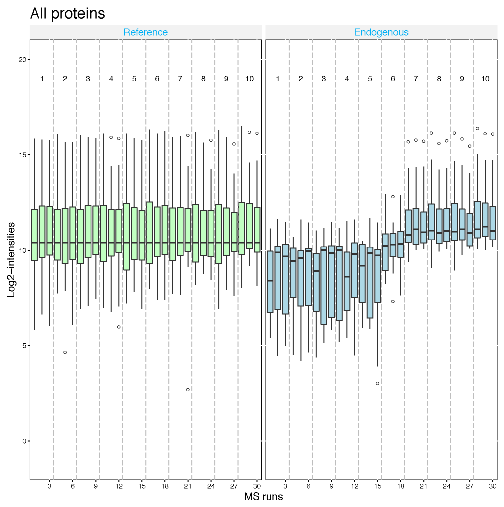
No normalization No normalization is performed. If you had your own normalization before `MSstats`, you should use like below.

```
srm.nonorm <- dataProcess(SRMRawData, normalization=FALSE)
dataProcessPlots(srm.nonorm, type='QCplot', address='srm_noNorm_')
```



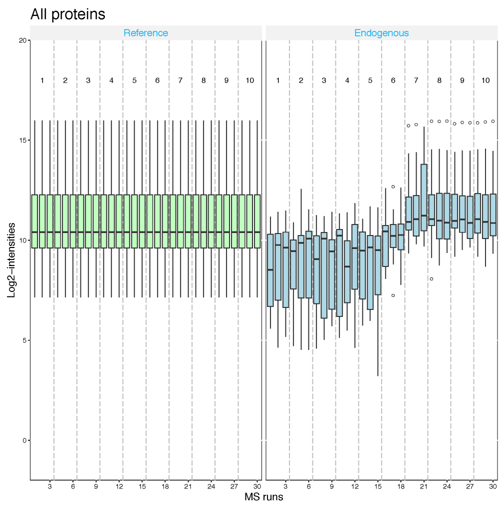
Equalize medians normalization The default option for normalization is **'equalizeMedians'**, where all the intensities in a run are shifted by a constant, to equalize the median of intensities across runs for label-free experiment. This normalization method is appropriate **when we can assume that the majority of proteins do not change across runs**. Be cautious when using the `equalizeMedians` option for a label-free dataset with only a small number of proteins. For label based experiment,`equalizeMedians` equalizes the median of reference intensities across runs and is generally proper even for a dataset with a small number of proteins.

```
srm.equalmed <- dataProcess(SRMRawData, normalization = 'equalizeMedians')
dataProcessPlots(srm.equalmed, type='QCplot', address='srm_equalM_')
```



Quantile normalization The distribution of all the intensities in each run will become the same across runs for label-free experiment. For label-based experiment, the distribution of all the reference intensities will be become the same across runs and all the endogenous intensities are shifted by a constant corresponding to reference intensities.

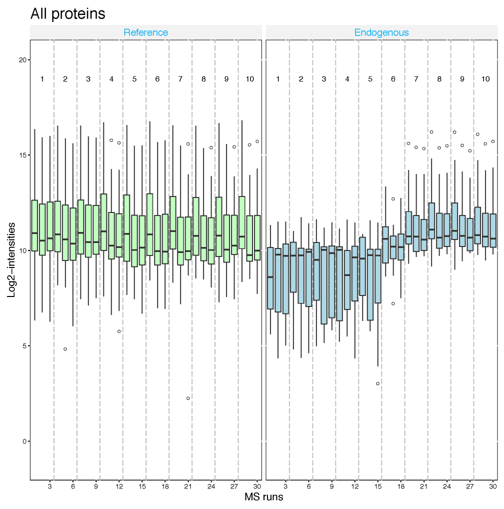
```
srm.quantile <- dataProcess(SRMRawData, normalization='quantile')
dataProcessPlots(srm.quantile, type='QCplot', address='srm_quantile_')
```



Global standards normalization : example 1 If you have a spiked in standard across all MS runs, you may set this to `globalStandards` and define the standard with `nameStandards` option. Global standard peptide or Protein names, which you can assume that they have the same abundance across MS runs, should be assigned in the vector for this option.

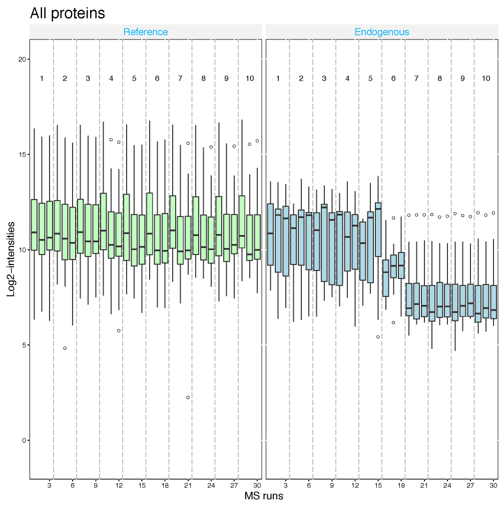
First, let's assume that `PMG2` proteins is the spike-in protein and shoule be equal amount across MS runs.

```
srm.global.pmg2 <- dataProcess(SRMRawData, normalization ='globalStandards',
                                     nameStandards = 'PMG2')
dataProcessPlots(srm.global.pmg2, type='QCplot', address='srm_global_PMG2_')
```



Second, let's assume that `IDHC` proteins is the spike-in protein and shoule be equal amount across MS runs.

```
srm.global.idhc <- dataProcess(SRMRawData, normalization ='globalStandards',
                               nameStandards = 'IDHC')
dataProcessPlots(srm.global.idhc, type='QCplot', address='srm_global_IDHC_')
```



Global standards normalization : example 2

Further steps is the same as in general workflow (section 4.1).

Reference

Benjamini, Y., and Y. Hochberg. 1955. "Controlling the false discovery rate: a practical and powerful approach to multiple testing." *J.R. Statist. Soc. B* 57 (1): 289–300.

Bruderer, R., O. M. Bernhardt, T. Gandhi, S. M Miladinović, L.-Y. Cheng, S. Messner, T. Ehrenberger, et al. 2015. "Extending the limits of quantitative proteome profiling with Data-Independent Acquisition and application to acetaminophen-treated three-dimensional liver microtissues." *Mol. Cell. Proteomics* 14 (5): 1400–1410.

Choi, M. and Eren-Dogu, Z. F. and Colangelo, C. and Cottrell, J. and Hoopmann, M. R. and Kapp, E. A. and Kim, S. and Lam, H. and Neubert, T. A. and Palmblad, M. and Phinney, B. S. and Weintraub, S. T. and MacLean, B. and Vitek, O. 2017. "ABRF Proteome Informatics Research Group (iPRG) 2015

Study: Detection of differentially abundant proteins in label-free quantitative LC-MS/MS experiments." *J. Proteome Res.* https://doi.org/10.1021/acs.jproteome.6b00881.

Cox, Jürgen, and Matthias Mann. 2008. "MaxQuant enables high peptide identification rates, individualized p.p.b.-range mass accuracies and proteome-wide protein quantification." *Nature Biotechnology* 26 (12): 1367–72.

MacLean, B., D. M. Tomazela, N. Shulman, M. Chambers, G. Finney, B. Frewen, R. Kern, D. L Tabb, D. C. Liebler, and M. J. MacCoss. 2010. "Skyline: An open source document editor for creating and analyzing targeted proteomics experiments." *Bioinformatics* 26–27: 966.

Mueller, L. N., O. Rinner, A. Schmidt, S. Letarte, B. Bodenmiller, M.-Y. Brusniak, O. Vitek, R. Aebersold, and M. Müller. 2007. "SuperHirn - a novel tool for high resolution LC-MS-based peptide/protein profiling." *Proteomics* 7: 3470–80.

Röst, H. L., G. Rosenberger, P. Navarro, L. Gillet, S. M. Miladinović, O. T. Schubert, W. Wolski, et al. 2014. "OpenSWATH enables automated, targeted analysis of data- independent acquisition MS data." *Nat. Biotechnol.* 32 (3): 219–23.

Sturm, Marc, Andreas Bertsch, Clemens Gröpl, Andreas Hildebrandt, Rene Hussong, Eva Lange, Nico Pfeifer, et al. 2008. "OpenMS – An open-source software framework for mass spectrometry." *BMC Bioinformatics* 9 (1): 163.

Tsou, C.-C., D. Avtonomov, B. Larsen, M. Tucholska, H. Choi, A.-C. Gingras, and A. I. Nesvizhskii. 2015. "DIA-Umpire: comprehensive computational framework for data-independent acquisition proteomics." *Nat. Methods* 12 (3): 258–64.