

Protein significance analysis of mass spectrometry-based proteomics experiments with R and MSstats (v3.0.9)

Meena Choi, Tsung-Heng Tsai & Erik Verschueren

September 24, 2015

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.

Applicability

MSstats version 3.0 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 currently not applicable to experiments that compare multiple metabolically labeled endogenous samples within a same run. It is not applicable to experiments with iTRAQ labeling. These experiments will be supported in the future.

Statistical functionalities

MSstats version 3.0 and above performs three analysis steps. The first step, *data processing and visualization*, transforms and normalizes the intensities of the peaks, 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, the type of spectral acquisition strategy, and the scope of conclusions (e.g. restricted to the subjects, or expanded to the underlying populations), 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 datasets, and calculates the minimal number of replicates necessary in the future experiment to achieve a pre-specified statistical power.

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 SuperHirn(Mueller et al. 2007), MaxQuant(Cox and Mann 2008), Progenesis, MultiQuant, OpenMS(Sturm et al. 2008) or OpenSWATH.

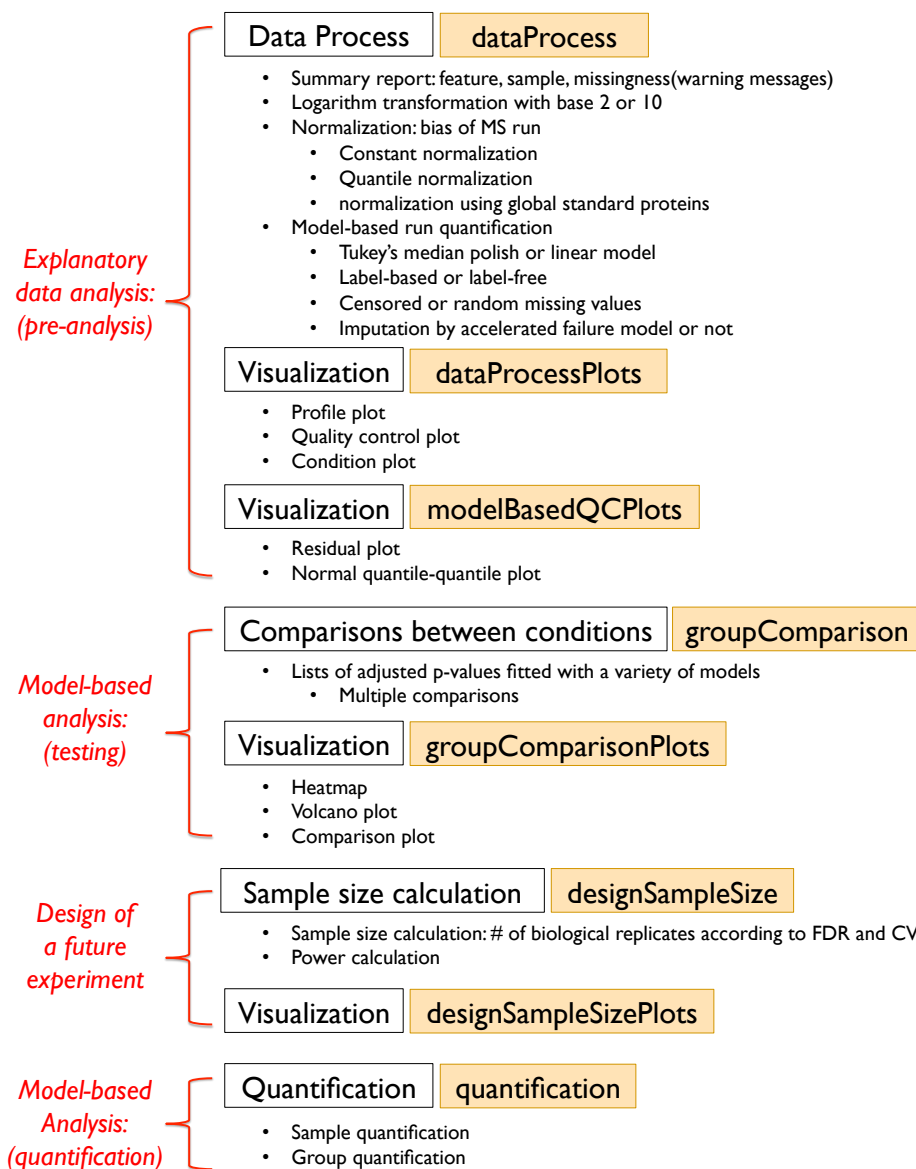
For statistics experts, **MSstats** 3.0 and above satisfies the interoperability requirements of Bioconductor, and takes as input data in the **MSnSet** format (Gatto and Lilley 2012). The command line-based workflow is partitioned into a series of independent steps, that facilitate the development and testing of alternative statistical approaches. It complies with the maintenance and documentation requirements of Bioconductor.

Finally, **MSstats** 3.0 and above is available as an external tool within Skyline (MacLean et al. 2010). 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.

Availability

MSstats is available under the Artistic-2.0 license at msstats.org. **MSstats** as an external tool 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.

Overview of the functionalities



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-MSstatsV2.1.6.pdf` on the **Installation** page of 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. The file should contain the following variables: **ProteinName**, **PeptideSequence**, **PrecursorCharge**, **FragmentIon**, **ProductCharge**, **IsotopeLabelType**, **Condition**, **BioReplicate**, **Run**, **Intensity**. The variable names are fixed, but are not case-sensitive. This required input data is generated automatically when using **MSstats** report format in Skyline.

- (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.
- (c) **IsotopeLabelType**: This column indicates whether this measurement is based on the endogenous peptides (use L) or labeled reference peptides (use H).
- (d) **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).
- (e) **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.
- (f) **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**.
- (g) **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 input dataset is shown. Example dataset 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	TDSEATLISSTID	2	y10	0	H	1	ReplA	1	43506.5425
9	ACEA	TDSEATLISSTID	2	y10	0	L	1	ReplA	1	217.203553
10	ACEA	TDSEATLISSTID	2	y7	0	H	1	ReplA	1	68023.0377
11	ACEA	TDSEATLISSTID	2	y7	0	L	1	ReplA	1	725.284308
12	ACEA	TDSEATLISSTID	2	y8	0	H	1	ReplA	1	68276.0489
13	ACEA	TDSEATLISSTID	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

Condition	BioReplicate	Run
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 *pared 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 appearance. 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

Condition	BioReplicate	Run
BiopsyTumor	PatientA	5
BiopsyTumor	PatientA	6
BiopsyHealthy	PatientB	7
BiopsyHealthy	PatientB	8
BiopsyHealthy	PatientB	9
BiopsyTumor	PatientB	10
BiopsyTumor	PatientB	11
BiopsyTumor	PatientB	12

2.1.3 MSnSet format

MSstats also allows data to be in the format of **MSnSet**, which is consistent with the requirements of Bioconductor. The **MSnSet** format has several components, of which the most commonly accessed are **assayData**, **phenoData**, and **featureData**. **assayData** is a matrix of intensities, where each row corresponds a transition, and the columns correspond to sample ids. **phenoData** contains columns that describe the biological samples, conditions in the experiment. **featureData** contains columns describing the peptide features, such as the name or id of the underlying protein and information of features.

If the data are stored in the format **expressionSet**, group labels information is required. If more than one variable is listed in the argument **group**, then a concatenated variable is created based on all of the specified **group** variables. The remaining information (peptide feature ids, biological replicate ids, and abundance) can be extracted from the rows and columns of **featureData** and **phenoData**, or the users can assign them based on their experimental design.

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	TPPAVLLK	2	y7	109401	L	2	1	3	257486
350748	TPPAVLLK	2	y7	109401	L	2	2	4	141159
350748	TPPAVLLK	2	y7	109401	L	1	1	1	452908
350748	TPPAVLLK	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

Set the working directory to where you saved files. Note that you should a different path on your computer from the example.

```
setwd('/Users/Meena/Dropbox/MSstats_GitHub_document')
```

You can check your working directory by:

```
getwd()
```

```
## [1] "/Users/Meena/Dropbox/MSstats_GitHub_document"
```

If you didn't have MSstats installed so far, you can download the package from the [MSstats installation page](#) to your working directory and install it as follows:

```
install.packages(pkgs = 'MSstats_3.0.9.tar.gz', repos = NULL, type = 'source')
```

Load MSstats into the R session and verify that you have the correct version (3.0.9) loaded.

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


4. DDA analysis with MSstats

Controlled mixture DDA data will be used for demonstration. This dataset is available in [MSstats material github](#) in the folder named 'example dataset/DDA_controlledMixture2009'. It is processed by Superhirn. [original reference link](#)

4.1 Preparing the data for MSstats input

Let's start by reading in data.

```
DDA2009.superhirn <- read.csv("RawData.DDA.csv")
head(DDA2009.superhirn)
```

```
## ProteinName PeptideSequence PrecursorCharge FragmentIon ProductCharge
## 1 bovine S.PVDIDTK_5 NA 5 NA
## 2 bovine S.PVDIDTK_5 NA 5 NA
## 3 bovine S.PVDIDTK_5 NA 5 NA
## 4 bovine S.PVDIDTK_5 NA 5 NA
## 5 bovine S.PVDIDTK_5 NA 5 NA
## 6 bovine S.PVDIDTK_5 NA 5 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
```

4.2 Processing the data

4.2.1 Normalizing and summarizing data with dataProcess

?dataProcess

4.2.1.1 Default normalization and summarization options (! Always pay attention to the default options)

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 works when the assumption that the majority of proteins do not change across runs is held. Therefore if your label-free DDA dataset has a small number of proteins, please be careful to use the `equalizeMedians` option. Instead, if you have a spiked in standard then you want to set this to `globalStandards` and define the standard with `nameStandards`. For label based experiment, `equalizeMedians` equalizes the median of reference intensities across runs for label based experiment. Therefore, it works for dataset with a small number of proteins.

Usually label-free DDA datasets have many missing values and noisy features with outliers. MSstats supports several ways to deal with this. The default option for summarization is TMP (Tukey's median polish, 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 and a run (`cutoffCensored="minFeatureNRun"`). Here the NA values

represent censored measurements, which are below the cutoff and not detected. This process handles missing values through imputation and reduces the influence of the outliers using the TMP estimation. However, those runs with no measurements at all will be removed and not be used for any calculation.

default option

```
DDA2009.TMP <- dataProcess(raw = DDA2009.superhirn, fillIncompleteRows = TRUE,
                           normalization = 'equalizeMedians',
                           summaryMethod = 'TMP',
                           censoredInt = "NA", cutoffCensored="minFeatureNRun",
                           MBimpute = TRUE)
```

```
## * Use all features that the dataset originally has.
```

```
##
```

```
## Summary of Features :
```

```
## count
## # of Protein 6
## # of Peptides/Protein 11-32
## # of Transitions/Peptide 1-1
```

```
##
```

```
## Summary of Samples :
```

```
## C1 C2 C3 C4 C5 C6
## # of MS runs 3 3 3 3 3 3
## # of Biological Replicates 1 1 1 1 1 1
## # of Technical Replicates 3 3 3 3 3 3
```

```
##
```

```
## Summary of Missingness :
```

```
##
```

```
## # transitions are completely missing in one condition: 90
```

```
##
```

```
## -> D.GPLTGTyr_23_NA_23_NA, F.HFWGSSDDQGSEHTVDR_402_NA_402_NA, G.PLTGTyr_8_NA_8_NA, H.SFNVEYDDSQ
```

```
##
```

```
## # run with 75% missing observations: 0
```

```
##
```

```
##
```

```
## == Start the summarization per subplot...
```

```
## Getting the summarization by Tukey's median polish per subplot for protein bovine ( 1 of 6 )
```

```
## Getting the summarization by Tukey's median polish per subplot for protein chicken ( 2 of 6 )
```

```
## Getting the summarization by Tukey's median polish per subplot for protein cyc_horse ( 3 of 6 )
```

```
## Getting the summarization by Tukey's median polish per subplot for protein myg_horse ( 4 of 6 )
```

```
## Getting the summarization by Tukey's median polish per subplot for protein rabbit ( 5 of 6 )
```

```
## Getting the summarization by Tukey's median polish per subplot for protein yeast ( 6 of 6 )
```

```
##
```

```
## == the summarization per subplot is done.
```

If MBimpute=FALSE, imputation will not be performed and censored intensities (here NAs) will be just replaced with cutoff value assigned in cutoffCensored="minFeatureNRun". With this option, those runs with substantial missing measurements will be biased by the cutoff value. In such case, you can remove the runs that have more than 50% missing values from your analysis with remove50missing=TRUE.

```
names(DDA2009.TMP)
```

```
## [1] "ProcessedData"      "RunlevelData"      "SummaryMethod"
## [4] "ModelQC"           "PredictBySurvival"
```

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

```
##      PROTEIN                                PEPTIDE TRANSITION
## 55   bovine                                D.GPLTGTyr_23_NA      23_NA
## 937  bovine F.HFWGSSDDQGSEHTVDR_402_NA      402_NA
## 1628 bovine F.HWGSSDDQGSEHTVDR_229_NA      229_NA
## 19   bovine                                G.PLTGTyr_8_NA       8_NA
## 1081 bovine H.SFNVEYDDSQDK_465_NA          465_NA
## 469  bovine K.AVVQDPALKPL_156_NA          156_NA
##                                     FEATURE LABEL GROUP_ORIGINAL
## 55                                D.GPLTGTyr_23_NA_23_NA      L      C1
## 937 F.HFWGSSDDQGSEHTVDR_402_NA_402_NA      L      C1
## 1628 F.HWGSSDDQGSEHTVDR_229_NA_229_NA      L      C1
## 19                                G.PLTGTyr_8_NA_8_NA      L      C1
## 1081 H.SFNVEYDDSQDK_465_NA_465_NA      L      C1
## 469 K.AVVQDPALKPL_156_NA_156_NA      L      C1
##      SUBJECT_ORIGINAL RUN GROUP SUBJECT SUBJECT_NESTED INTENSITY ABUNDANCE
## 55                    1  1    1        1              1.1  757400.1  19.79517
## 937                    1  1    1        1              1.1  2087125.8  21.25756
## 1628                    1  1    1        1              1.1  1485145.8  20.76665
## 19                      1  1    1        1              1.1  4986404.0  22.51404
## 1081                    1  1    1        1              1.1  2488141.2  21.51111
## 469                    1  1    1        1              1.1  7519322.0  23.10664
##      METHOD
## 55      1
## 937      1
## 1628      1
## 19        1
## 1081      1
## 469      1
```

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

```
##      RUN  Protein LogIntensities more50missing GROUP GROUP_ORIGINAL
## 1  1    bovine      21.25800                FALSE      1      C1
## 2  1  myg_horse     22.77386                FALSE      1      C1
## 3  1   rabbit      12.54786                TRUE       1      C1
## 4  1    yeast      16.44018                TRUE       1      C1
## 5  1   chicken     18.29509                FALSE      1      C1
## 6  1  cyc_horse     19.49460                FALSE      1      C1
##      SUBJECT_ORIGINAL SUBJECT_NESTED SUBJECT
## 1                    1              1.1    1
## 2                    1              1.1    1
## 3                    1              1.1    1
## 4                    1              1.1    1
```

```
## 5          1          1.1          1
## 6          1          1.1          1
```

```
# Since this is not model-based, no model summary (here DDA skyline.quant$ModelQC=NULL).
# Only with 'summaryMethod="linear"'
head(DDA2009.TMP$ModelQC)
```

```
## NULL
```

```
# here 'TMP'
head(DDA2009.TMP$SummaryMethod)
```

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

```
# predict values by AFT with 'MBimpute=TRUE'.
# These values are matching with rownames of DDA2009.TMP$ProcessedData
head(DDA2009.TMP$PredictBySurvival)
```

```
##          55          937          1628          19          1081          469
## 19.63438 22.40369 21.67132 23.52286 21.55629 23.11901
```

If `censoredInt=NULL`, we assume that all intensities are randomly missing and there is no action for missing values.

```
# no action for missing values.
DDA2009.TMP.random <- dataProcess(raw = DDA2009.superhirn, fillIncompleteRows = TRUE,

                                normalization = 'equalizeMedians',
                                summaryMethod = 'TMP',
                                censoredInt=NULL)
```

```
## * Use all features that the dataset originally has.
```

```
##
## Summary of Features :
## count
## # of Protein          6
## # of Peptides/Protein 11-32
## # of Transitions/Peptide 1-1
##
## Summary of Samples :
## C1 C2 C3 C4 C5 C6
## # of MS runs          3 3 3 3 3 3
## # of Biological Replicates 1 1 1 1 1 1
## # of Technical Replicates 3 3 3 3 3 3
```

```
##
## Summary of Missingness :
##
## # transitions are completely missing in one condition: 90
##
```

```
##      -> D.GPLTGTyr_23_NA_23_NA, F.HFWGSSDDQGSEHTVDR_402_NA_402_NA, G.PLTGTyr_8_NA_8_NA, H.SFNVEYDDSQ
##
##      # run with 75% missing observations: 0
##
##
##      == Start the summarization per subplot...
## Getting the summarization by Tukey's median polish per subplot for protein bovine ( 1 of 6 )
## Getting the summarization by Tukey's median polish per subplot for protein chicken ( 2 of 6 )
## Getting the summarization by Tukey's median polish per subplot for protein cyc_horse ( 3 of 6 )
## Getting the summarization by Tukey's median polish per subplot for protein myg_horse ( 4 of 6 )
## Getting the summarization by Tukey's median polish per subplot for protein rabbit ( 5 of 6 )
## Getting the summarization by Tukey's median polish per subplot for protein yeast ( 6 of 6 )
##
##      == the summarization per subplot is done.
```

4.2.1.2 Different summarization options Besides summarizing observations with the median polish method, MSstats also offers a summarization option using linear model, and one based on the sum of log-intensities, which is the default Skyline behaviour. Summarization using the option `summaryMethod="linear"` with `censoredInt=NULL` assumes that all NAs are randomly missing and uses `lm` or `lmer` for parameter estimation. On the other hand, `summaryMethod="linear"` with `censoredInt="NA"` assumes that NA intensities are censored, not detected because intensities are below the cutoff. AFT model is then applied with left-censored for parameter estimation. ‘cutoffCensored’ is the same as `summaryMethod="TMP"`. For `summaryMethod="linear"`, AFT model is only used for parameter estimation, not for imputation. Therefore `MBimpute=TRUE` with `summaryMethod="linear"` doesn’t work (no imputation before parameter estimation, the same with `MBimpute=FALSE`).

```
# linear mixed model (lm or lmer) with run and feature
DDA2009.linear <- dataProcess(raw = DDA2009.superhirn,
                             summaryMethod = "linear", censoredInt = NULL)

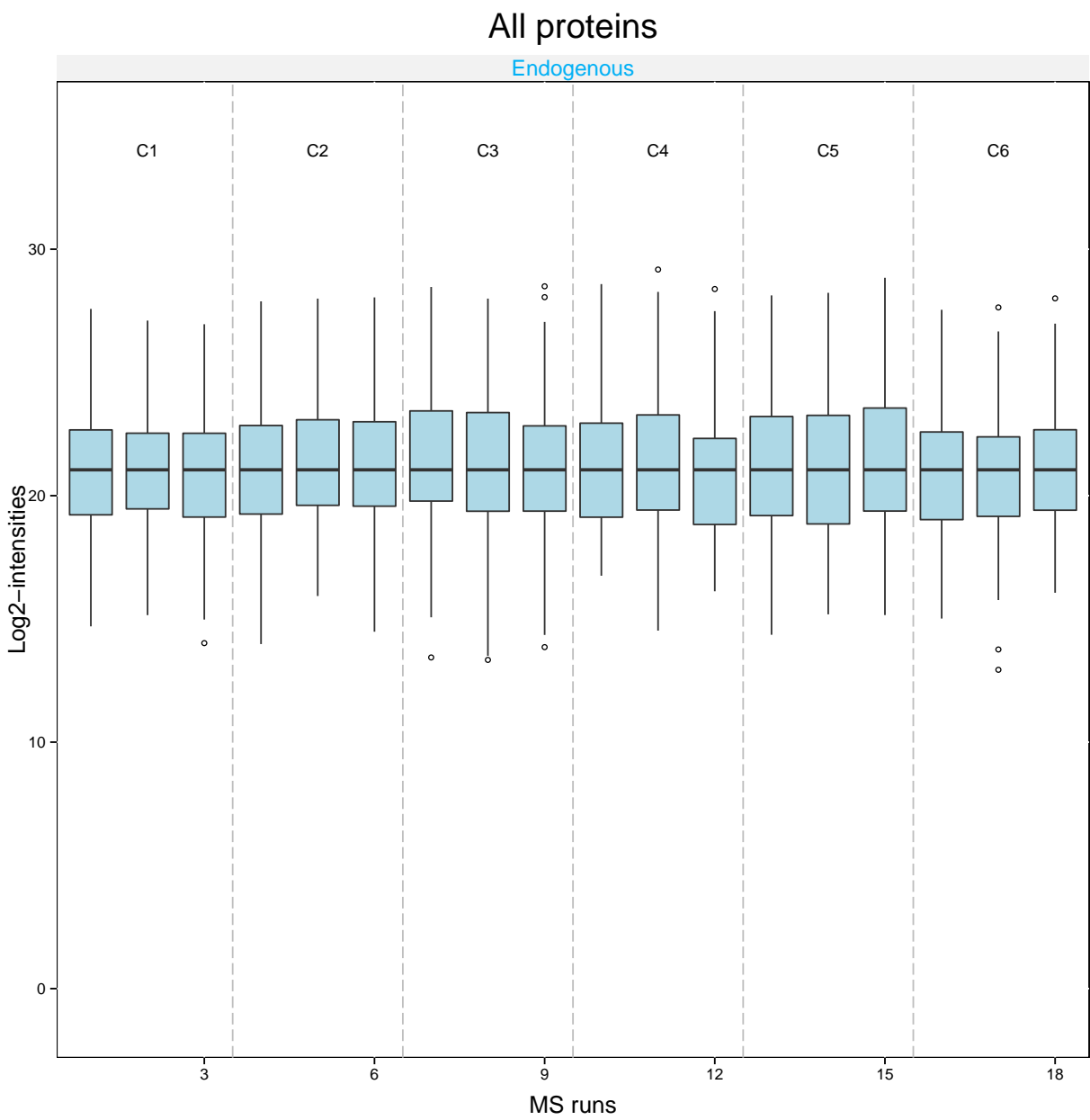
# accelerated failure model with left-censored. NA intensities are assumed as censored
DDA2009.linear.censored <- dataProcess(raw = DDA2009.superhirn,
                                       summaryMethod = "linear", censoredInt = "NA")
```

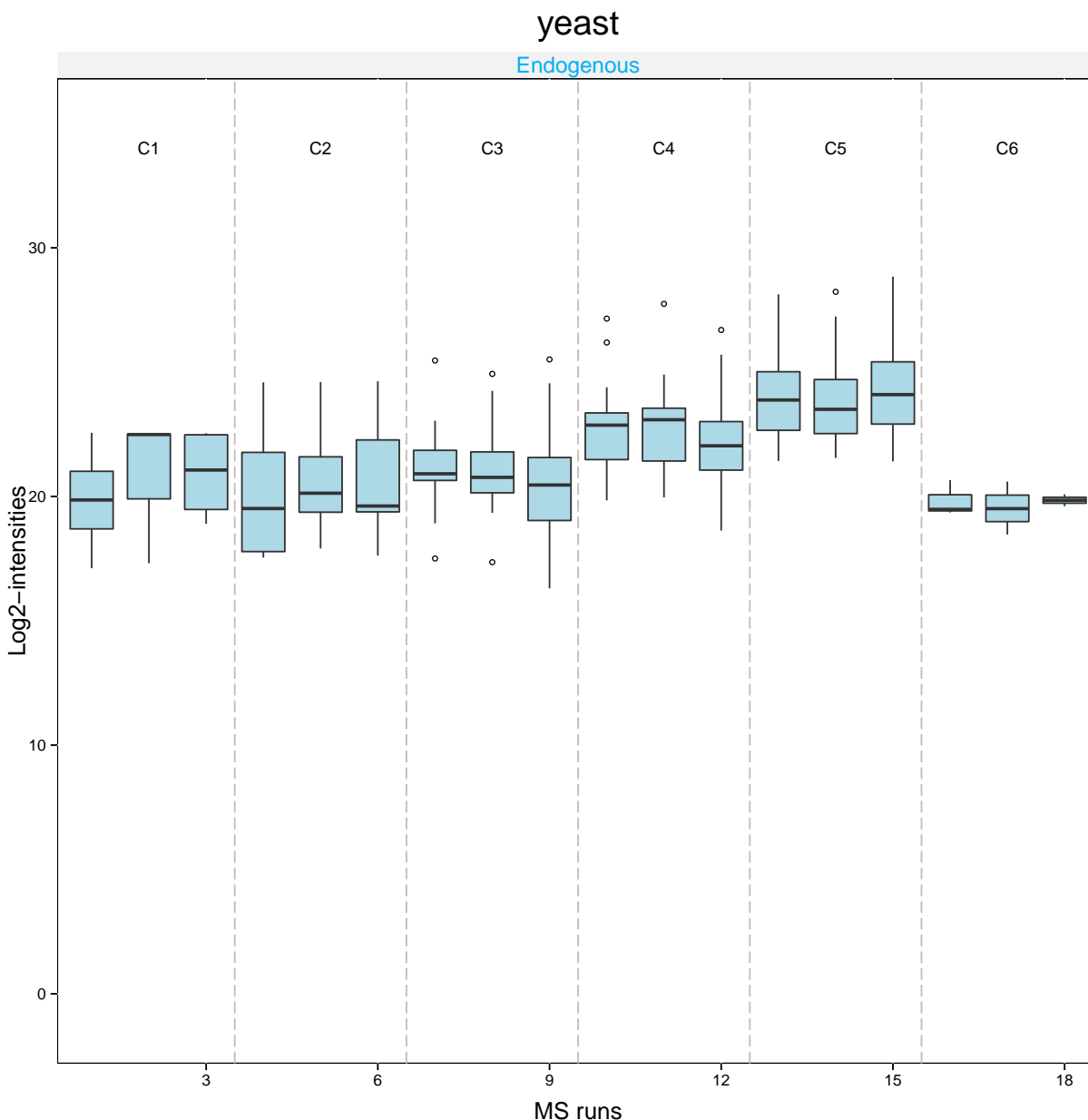
4.2.2 Visualization of processed data

4.2.2.1 Quality control and Normalization effects Now let’s look at what the equalize medians procedure did to our data. We can generate QC plots for all proteins or for single proteins at a time if we have a big dataset. Let’s look at the ‘Kininogen-1’ (Kng1/NP_036828) protein in these example data.

```
# use type="QCplot" with all proteins and change the upper limit of y-axis=35
dataProcessPlots(data = DDA2009.TMP, type="QCplot", ylimUp=35)

# QCplot for only 'yeast' protein
dataProcessPlots(data = DDA2009.TMP, type="QCplot", ylimUp=35,
                 which.Protein="yeast", address="yeast_eqmedians_")
```





NOTE Don't worry about warning messages as below. It means they will not plot with NA values, which we want.

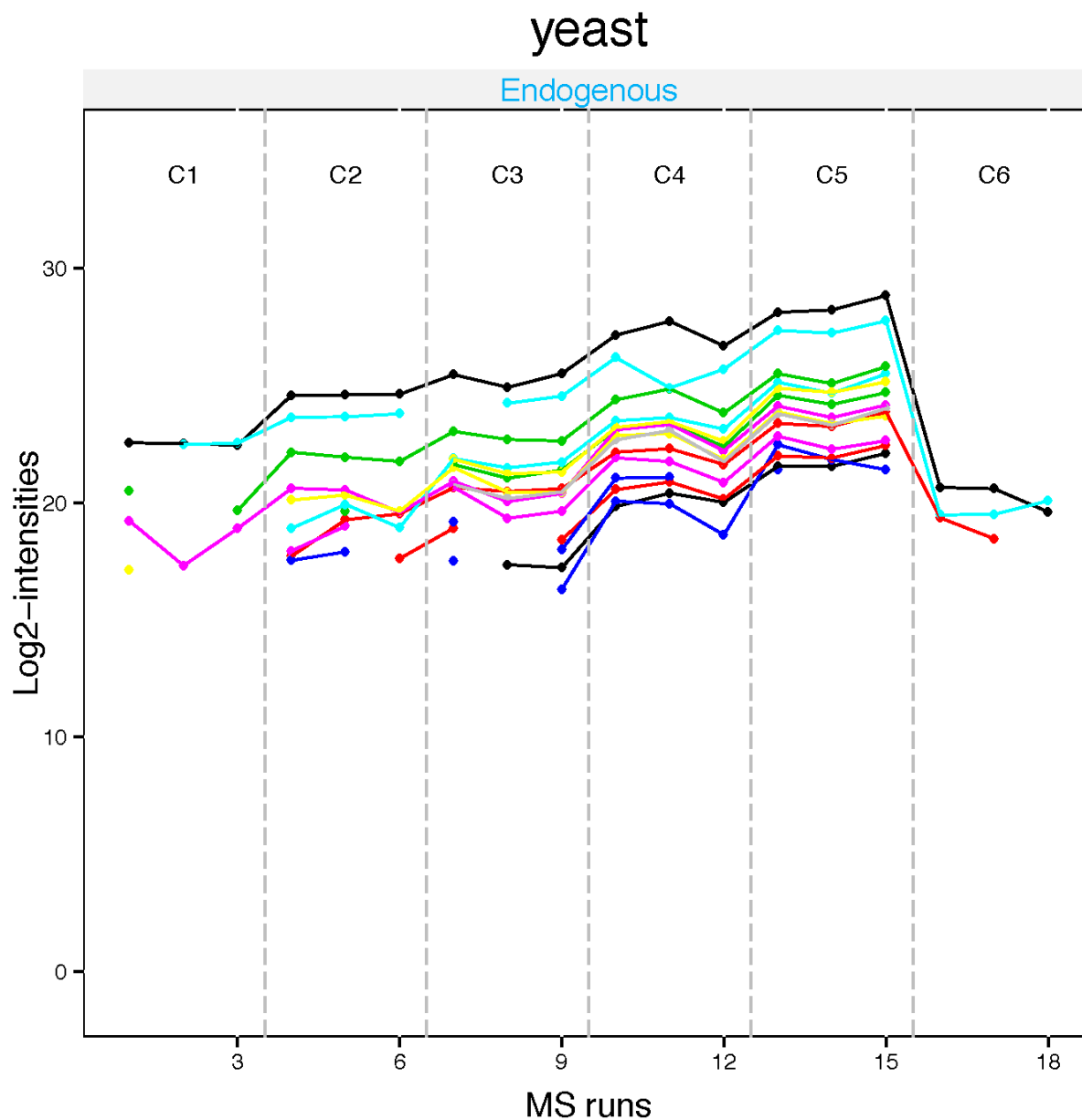
Warning messages: 1: In loop_apply(n, do.ply) : Removed 698 rows containing non-finite values (stat_boxplot).

4.2.2.2 Summarization effects

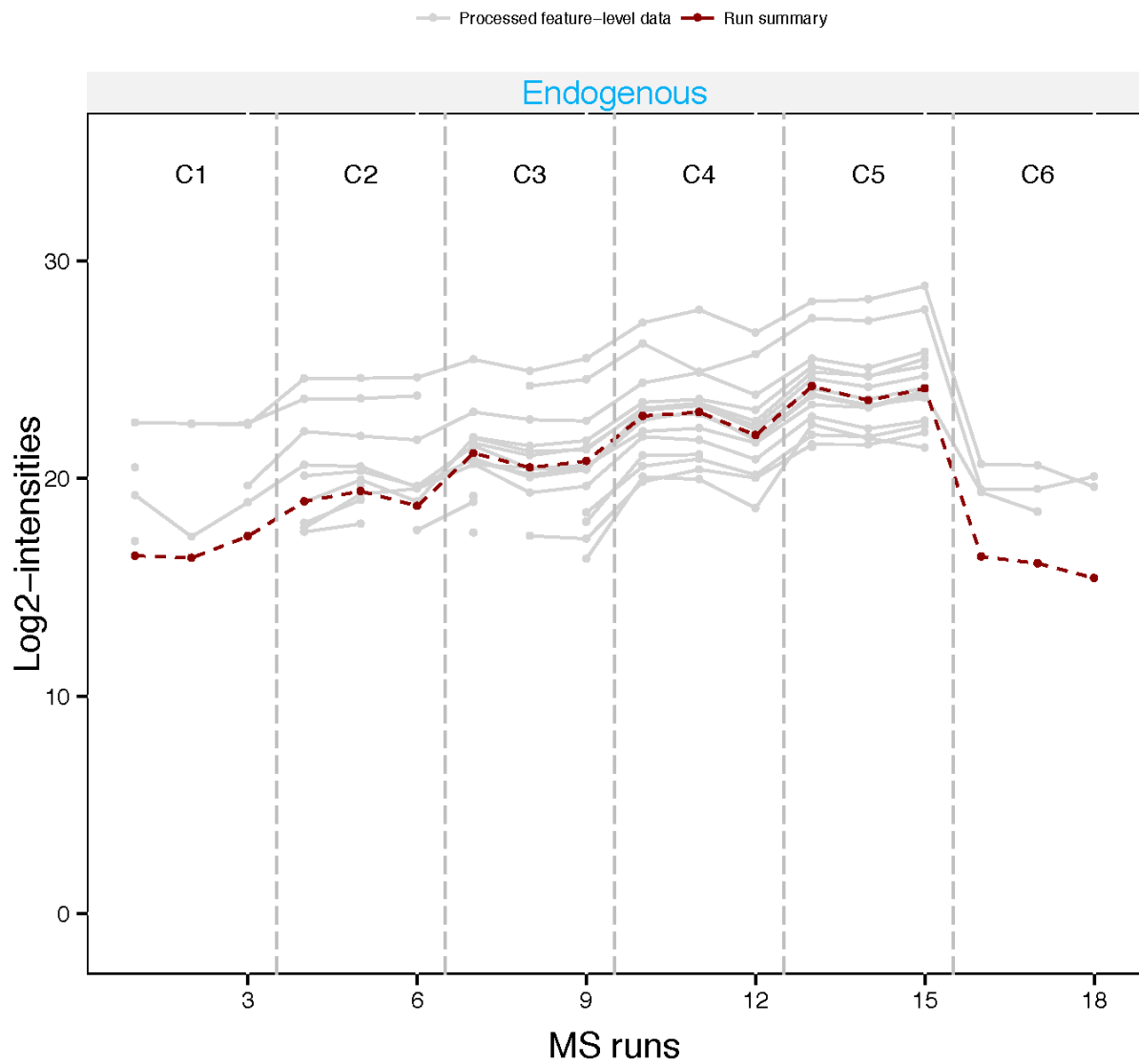
Profile plots Let's look at how the original data look like. Profile plot shows all individual measurements per protein. With `type="profileplot"`, two pdfs will be generated. The first pdf includes plots (per protein) to show individual measurement for each peptide (peptide for DDA, transition for SRM or DIA) across runs, grouped per condition. Each peptide has a different color/type layout. If you don't want to generate 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 overlaid in red.

```
dataProcessPlots(data = DDA2009.TMP, type="Profileplot", ylimUp=35,
  featureName="NA", width=7, height=7, address="DDA2009_TMP_")
```



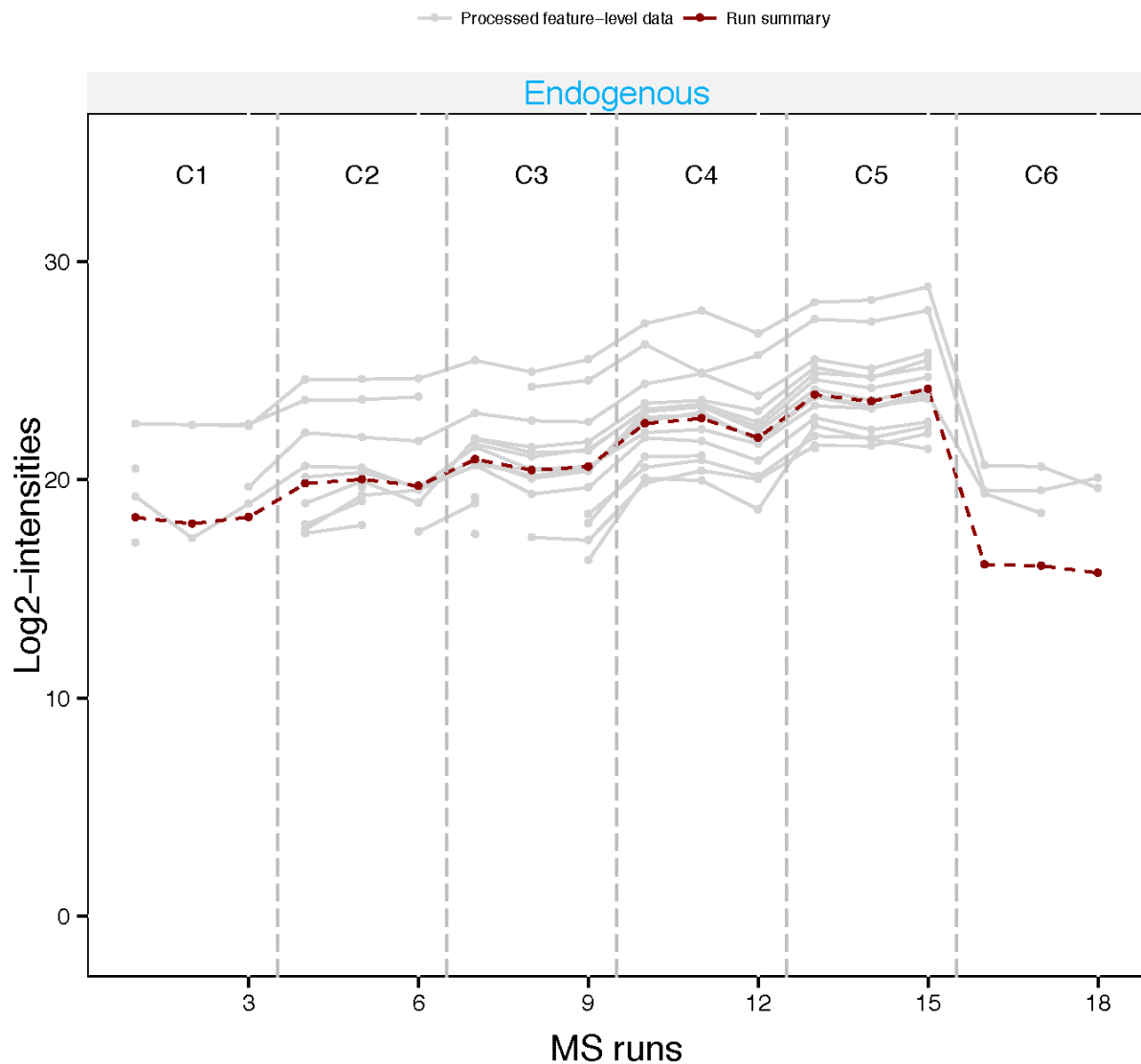
yeast



Have a look at the profile plots to compare the different options (e.g., TMP with or without considering missing values for summarization in `dataProcess`.) Original profile plots are the same. But, summarization plots are different, especially for conditions 'C1' and 'C2' in 'yeast' protein, which have many missing values.

```
dataProcessPlots(data = DDA2009.TMP.random, type="Profileplot", ylimUp=35,
  featureName="NA", width=7, height=7,
  originalPlot=FALSE, summaryPlot=TRUE, address="DDA2009_TMP_random_")
```

yeast



4.3 Finding differentially abundant proteins across conditions

4.3.1 Comparing conditions with groupComparison

After we normalized the data and summarized each protein's behaviour across conditions with one of the `dataProcess` summarization methods, we are all set to compare protein changes between groups of conditions. Within MSstats we can do this with the `groupComparison` function, which takes as input the output of the `dataProcess` function.

```
?groupComparison
```

We need to provide the `groupComparison` function with 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**:

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

```
## [1] "C1" "C2" "C3" "C4" "C5" "C6"
```

It shows the order of condition, which MSstats can recognize. Then, we add a row with entries of 0, 1, or -1 for every comparison we would like to make between groups of conditions.

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. -1 is for conditions we would like to put in the denominator 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. Also, combining multiple groups for comparison is 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")
```

We're ready to go! let's compare our two populations.

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

Let's inspect the results to see what proteins change significantly between groups.

```
# 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
names(DDA2009.comparisons$ComparisonResult)
```

```
## [1] "Protein"      "Label"        "log2FC"       "SE"           "Tvalue"
## [6] "DF"           "pvalue"       "adj.pvalue"
```

You can get a shorter list of the differentially abundant proteins based on the comparison results:

```
# get only significant proteins and comparisons among all comparisons
SignificantProteins =
  DDA2009.comparisons$ComparisonResult[DDA2009.comparisons$ComparisonResult$adj.pvalue < 0.05 ,]
nrow(SignificantProteins)
```

```
## [1] 33
```

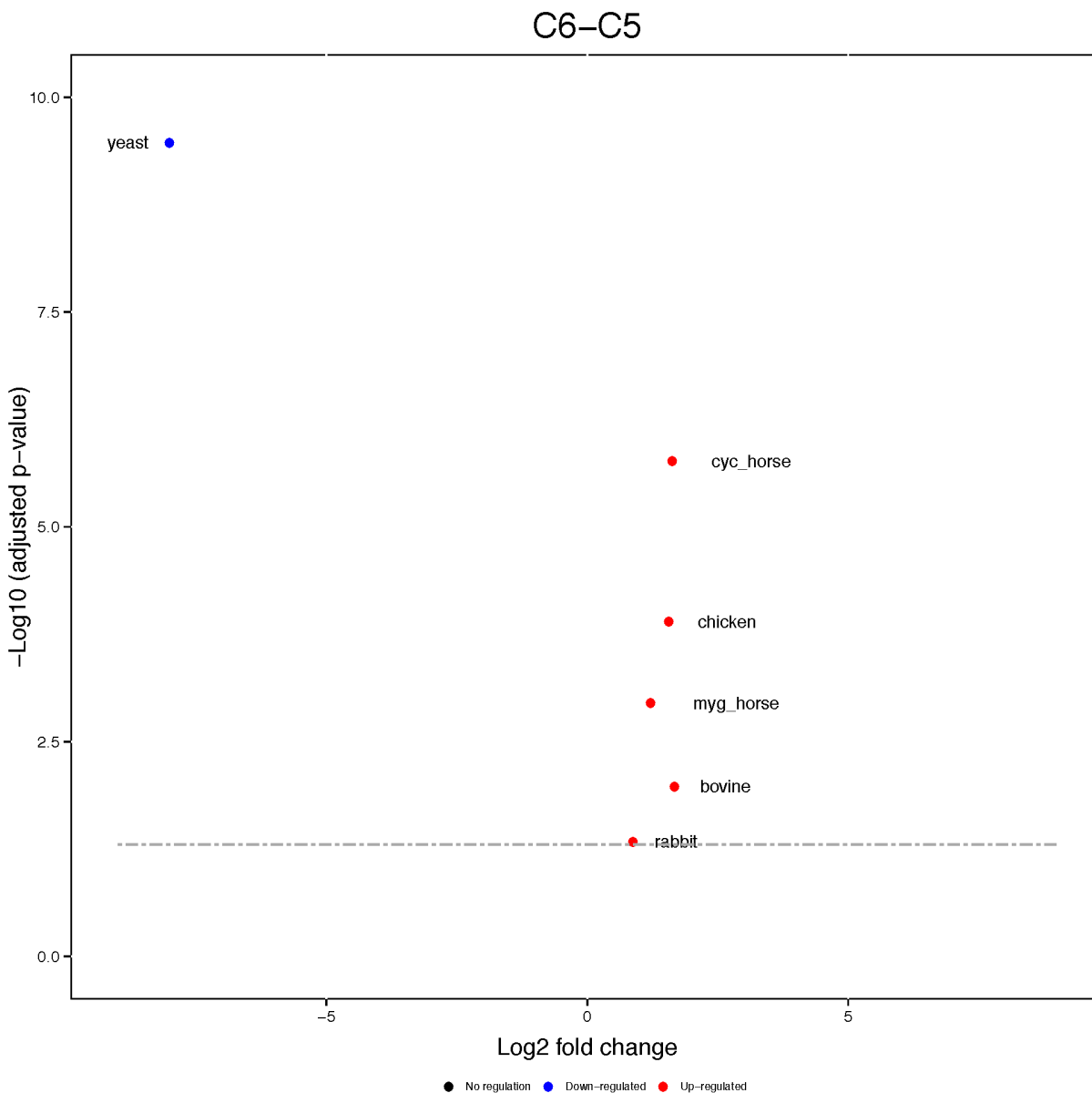
4.3.2 Visualization of differentially abundant proteins

`?groupComparisonPlots`

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, and represents statistical significance. The horizontal dashed line represents the FDR cutoff. The points above the FDR cutoff line are statistically significant differentially abundant proteins. These points are colored in red for upregulated proteins, and in blue for downregulated proteins. The x-axis is the model-based estimate of log-fold change (the base of logarithm transform is the same as specified in the logTrans option of the dataProcess step), 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 judged as not differentially abundant (and will be colored in black). The practical significance cutoff can 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')
```

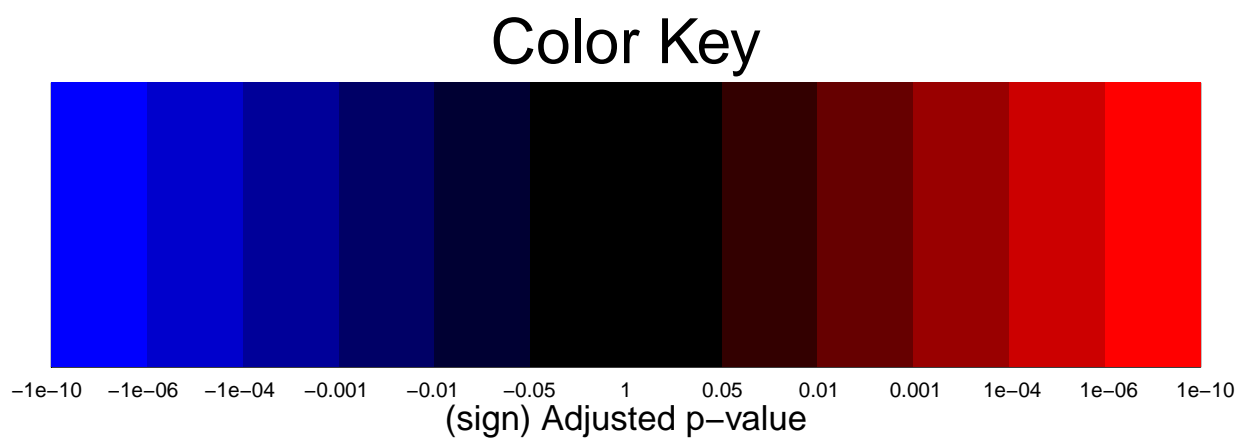
Then, 'VolcanoPlot.pdf' will be saved under the folder you assigned. It has the plots per comparison you set in contrast.matrix. Below is one of volcano plot, for comparison 'C6-C5'. Please check `?groupComparisonPlots` for detail, such as labelling protein names, size of dots, font sizes, etc.

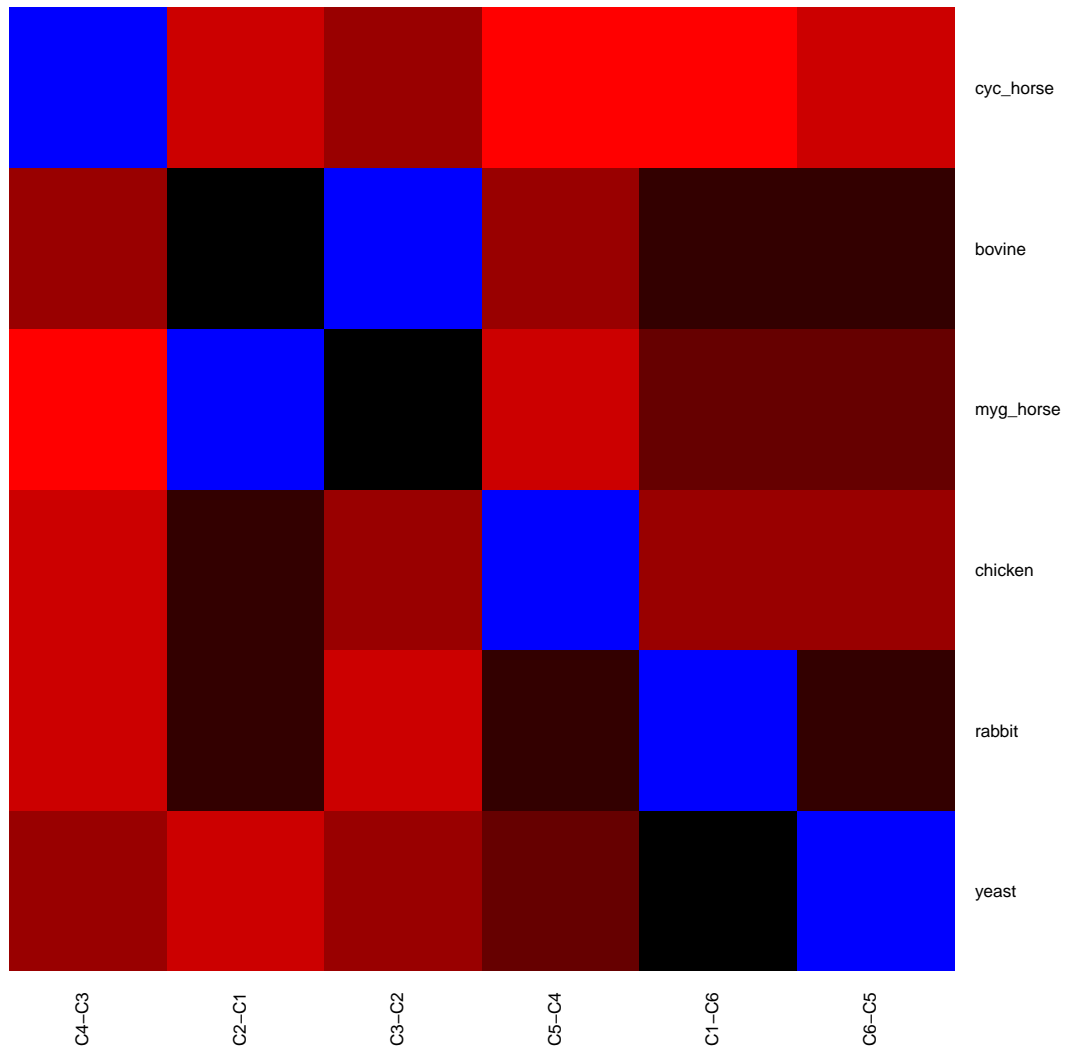


Heatmaps illustrate the patterns of up- and down-regulation of proteins in several comparisons. Columns in the heatmaps are comparison of conditions you assigned in `contrast.matrix`, and rows are proteins. The heatmaps display signed FDR-adjusted p-values of the tests, colored in red/blue for significantly 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 are not significantly differentially abundant.

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

Then, 'VolcanoPlot.pdf' will be saved under the folder you assigned. It has the plots per comparison you set in `contrast.matrix`. Below is one of volcano plot, for comparison 'C6-C5'. Please check `?groupComparisonPlots` for detail, such as labelling protein names, size of dots, font sizes, etc.





5. Convert skyline output to MSstats required format

If you use Skyline output, please follow this section. The same original raw dataset from previous section is used, but it is processed by Skyline.

5.1 Preparing the data for MSstats input

Let's read the data.

```
raw <- read.csv(file = "ControlMixerMSstatsInputfromskyline.csv")
head(raw)
```

```
## ProteinName PeptideSequence PrecursorCharge FragmentIon ProductCharge
## 1 bovine HWGSSDDQGSEHTVDR 2 precursor 2
## 2 bovine HWGSSDDQGSEHTVDR 2 precursor 2
## 3 bovine HWGSSDDQGSEHTVDR 2 precursor 2
## 4 bovine HWGSSDDQGSEHTVDR 2 precursor 2
## 5 bovine HWGSSDDQGSEHTVDR 2 precursor 2
## 6 bovine HWGSSDDQGSEHTVDR 2 precursor 2
## IsotopeLabelType Condition BioReplicate FileName Area
## 1 light 1 1 B06-8004_c.mzXML 1015466
## 2 light 3 2 B06-8006_c.mzXML 907841
## 3 light 5 3 B06-8008_c.mzXML 1263905
## 4 light 2 4 B06-8010_c.mzXML 2457121
## 5 light 4 5 B06-8012_c.mzXML 958204
## 6 light 6 6 B06-8014_c.mzXML 788090
## StandardType Truncated
## 1 NA False
## 2 NA False
## 3 NA False
## 4 NA False
## 5 NA False
## 6 NA False
```

The raw data (input data for MSstats) is required to contain variable of ProteinName, PeptideSequence, PrecursorCharge, FragmentIon, ProductCharge, IsotopeLabelType, Condition, BioReplicate, Run, Intensity. The variable names should be fixed. MSstats input from Skyline adapts the column scheme of the dataset so that it fits MSstats input format. However you should change FileName to Run and Area to Intensity.

```
colnames(raw)[9] <- 'Run'
colnames(raw)[10] <- 'Intensity'
head(raw)
```

```
## ProteinName PeptideSequence PrecursorCharge FragmentIon ProductCharge
## 1 bovine HWGSSDDQGSEHTVDR 2 precursor 2
## 2 bovine HWGSSDDQGSEHTVDR 2 precursor 2
## 3 bovine HWGSSDDQGSEHTVDR 2 precursor 2
## 4 bovine HWGSSDDQGSEHTVDR 2 precursor 2
## 5 bovine HWGSSDDQGSEHTVDR 2 precursor 2
## 6 bovine HWGSSDDQGSEHTVDR 2 precursor 2
## IsotopeLabelType Condition BioReplicate Run Intensity
## 1 light 1 1 B06-8004_c.mzXML 1015466
## 2 light 3 2 B06-8006_c.mzXML 907841
## 3 light 5 3 B06-8008_c.mzXML 1263905
## 4 light 2 4 B06-8010_c.mzXML 2457121
## 5 light 4 5 B06-8012_c.mzXML 958204
## 6 light 6 6 B06-8014_c.mzXML 788090
## StandardType Truncated
## 1 NA False
## 2 NA False
```



```
## 3      NA      False
## 4      NA      False
## 5      NA      False
## 6      NA      False
```

The difference between output from Skyline and other spectral processing tool is that Skyline can distinguish random missing (NA) and censored missing (zero). The output from Skyline can have NA (expect small number of NAs or none of them) and 0 (zero). Then, we can use zero values in intensity are censored.

```
sum(is.na(raw$Intensity))
```

```
## [1] 0
```

```
sum(raw$Intensity==0)
```

```
## [1] 4326
```

5.2 Preprocessing with DDA experiment from Skyline output

The output from Skyline for DDA experiment needs one extra step before using MSstats. It has several peak area from the monoisotopic, M+1 and M+2 peaks. To get a robust measure of peptide intensity, we can sum per peptide or use highest peak per peptide. Here we will sum per peptide.

```
library(reshape2)

raw$pepprecursor<-paste(raw$PeptideSequence, raw$PrecursorCharge, sep="_")

data_w = dcast( Run ~ pepprecursor, data=raw, value.var='Intensity', fun.aggregate=sum, fill=NULL)

newdata = melt(data_w, id.vars=c('Run'))
colnames(newdata)[colnames(newdata) %in% c("variable","value")]<-c('pepprecursor','Intensity')

uniinfo<-unique(raw[,c("ProteinName","PeptideSequence","PrecursorCharge","pepprecursor")])
newraw<-merge(newdata,uniinfo, by="pepprecursor")

uniinfo<-unique(raw[,c("Run","BioReplicate","Condition")])
newraw<-merge(newraw,uniinfo, by="Run")

newraw$BioReplicate<-1 # it should be change based on your experiment.
newraw$FragmentIon<-"sum"
newraw$ProductCharge<-NA
newraw$IsotopeLabelType<-"L"

raw<-newraw
# now 'raw' is ready to use MSstats
```

5.3 Different options for Skyline in dataProcess

Additional options in `dataProcess` need to be specified for processing Skyline output are `skylineReport=TRUE` (remove `Truncated=TRUE` rows and handle `intensity=0`) and `censoredInt="0"` (use `intensity=0` as censored to handle missing values and in this case, NA values are assumed as random missing).

6. Convert MaxQuant output to MSstats required input

If you have MaxQuant output, please follow this section. Here controlled mixture data with dynamic range benchmark (Cox, 2014 in MCP) is used to demonstrate. This dataset is available in [MSstats material github](#) in the folder named 'example dataset/DDA_controlledMixture20014'.

```
## 1. First, get protein ID information
proteinGroups <- read.table("DDA2014_proteinGroups.txt", sep = "\t", header = TRUE)

## 2. Read in annotation including condition and biological replicates: annotation.csv
annot <- read.csv("DDA2014_annotation.csv", header = TRUE)

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

## 4. Reformat for MSstats required input
## check options for converting format
?MaxQtoMSstatsFormat

msstats.raw <- MaxQtoMSstatsFormat(evidence=infile, annotation=annot, proteinGroups=proteinGroups)

## now 'msstats.raw' is ready for MSstats
head(msstats.raw)
```

	ProteinName	PeptideSequence	PrecursorCharge	FragmentIon
## 1	A5A614	QVAESTPDIPK	2	NA
## 2	000762ups	DPAATSVAAAR	2	NA
## 3	000762ups	FLTPCYHPNVDTQGNICLDILK	2	NA
## 4	000762ups	FLTPCYHPNVDTQGNICLDILK	3	NA
## 5	000762ups	GAEPSGGAAR	2	NA
## 6	000762ups	GISAFPESDNLFK	2	NA
##	ProductCharge	IsotopeLabelType	Condition	BioReplicate
## 1	NA	L	UPS1	1
## 2	NA	L	UPS1	1
## 3	NA	L	UPS1	1
## 4	NA	L	UPS1	1
## 5	NA	L	UPS1	1
## 6	NA	L	UPS1	1
##	Run	Intensity		
## 1	20130510_EXQ1_IgPa_QC_UPS1_01	NA		
## 2	20130510_EXQ1_IgPa_QC_UPS1_01	1144800000		
## 3	20130510_EXQ1_IgPa_QC_UPS1_01	32793000		
## 4	20130510_EXQ1_IgPa_QC_UPS1_01	566960000		
## 5	20130510_EXQ1_IgPa_QC_UPS1_01	58709000		
## 6	20130510_EXQ1_IgPa_QC_UPS1_01	861090000		

Reference

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) (November): 1367–1372.

- Gatto, L., and K. S. Lilley. 2012. “MSnbase-an R/Bioconductor Package for Isobaric Tagged Mass Spectrometry Data Visualization, Processing and Quantitation.” *Bioinformatics* 28: 288–289.
- 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: 3470–3480.
- 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.