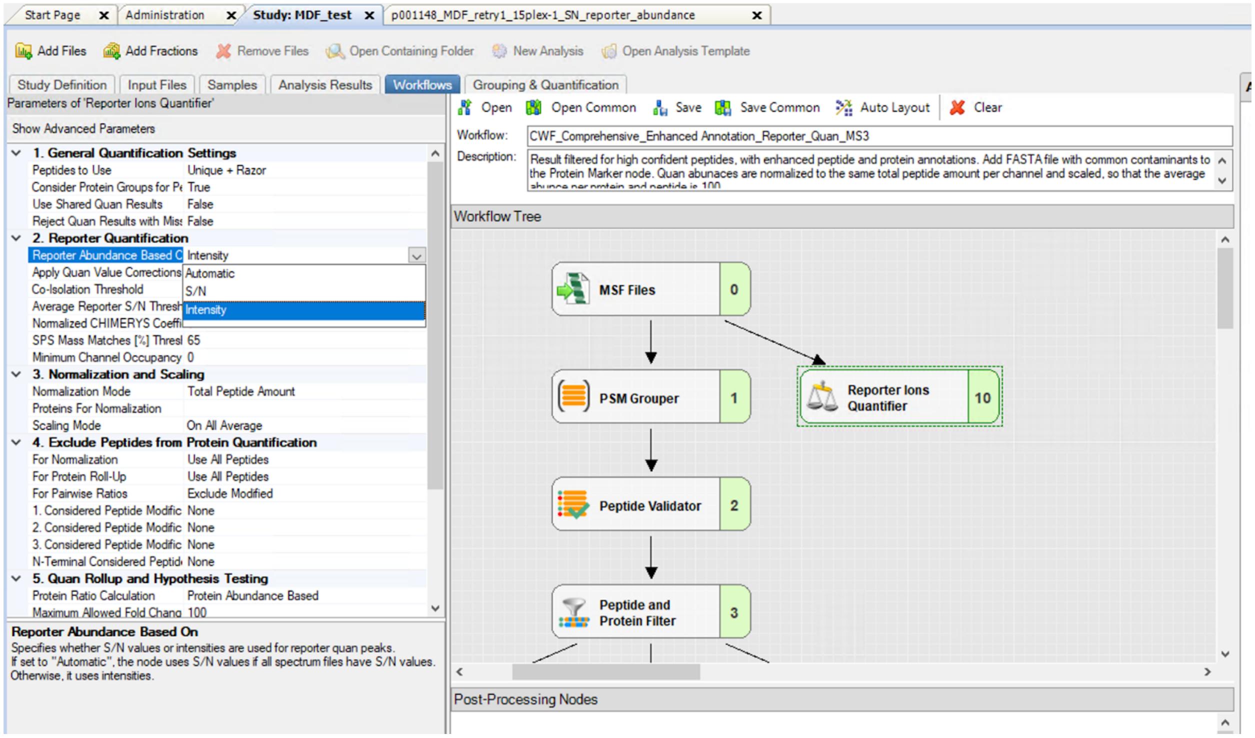
Use Proteome Discoverer with msTrawler
05/09/2023

Key parameters when running Proteome Discoverer

1. Proteome Discoverer (PD) has three modes for reporter quantitation: automatic, S/N or Intensity. To use PD with msTrawler, **choose Intensity mode**. The PD converter in msTrawler will use intensity and “Average Reporter S/N” in the output of PD to calculate S/N.



Parameters of 'Reporter Ions Quantifier'

Show Advanced Parameters

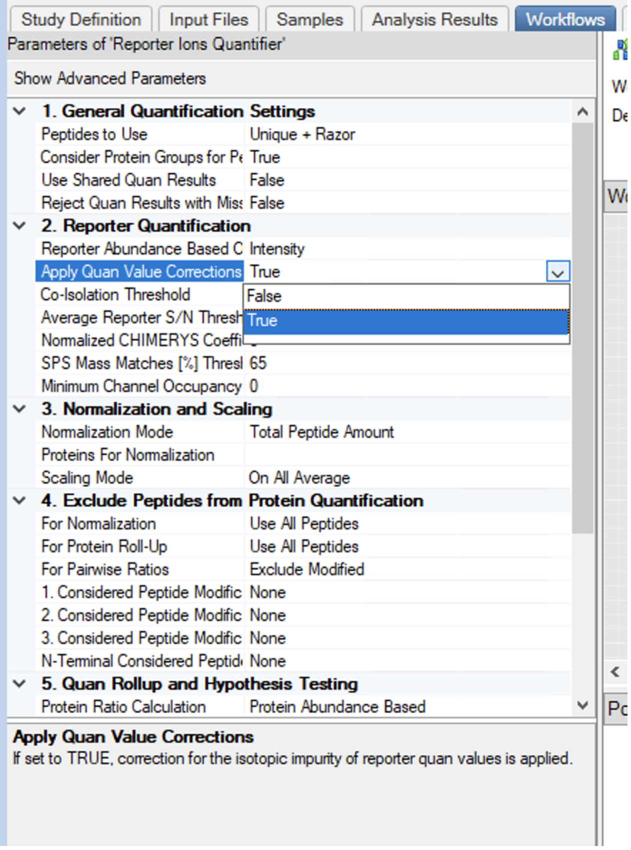
1. General Quantification Settings	
Peptides to Use	Unique + Razor
Consider Protein Groups for Pe	True
Use Shared Quan Results	False
Reject Quan Results with Miss	False
2. Reporter Quantification	
Reporter Abundance Based C	Intensity
Apply Quan Value Corrections	Automatic S/N Intensity
Co-Isolation Threshold	
Average Reporter S/N Thresh	
Normalized CHIMERYS Coeffi	
SPS Mass Matches [%] Thresl	65
Minimum Channel Occupancy	0
3. Normalization and Scaling	
Normalization Mode	Total Peptide Amount
Proteins For Normalization	
Scaling Mode	On All Average
4. Exclude Peptides from Protein Quantification	
For Normalization	Use All Peptides
For Protein Roll-Up	Use All Peptides
For Pairwise Ratios	Exclude Modified
1. Considered Peptide Modific	None
2. Considered Peptide Modific	None
3. Considered Peptide Modific	None
N-Terminal Considered Peptid	None
5. Quan Rollup and Hypothesis Testing	
Protein Ratio Calculation	Protein Abundance Based
Maximum Allowed Fold Chang	100

Reporter Abundance Based On
Specifies whether S/N values or intensities are used for reporter quan peaks.
If set to "Automatic", the node uses S/N values if all spectrum files have S/N values.
Otherwise, it uses intensities.

Workflow: CWF_Comprehensive_Enhanced Annotation_Reporter_Quan_MS3

Description: Result filtered for high confident peptides, with enhanced peptide and protein annotations. Add FASTA file with common contaminants to the Protein Marker node. Quan abunaces are normalized to the same total peptide amount per channel and scaled, so that the average abunce per protein and peptide is 100

2. Quan Value Correction could be set to False, but ideally it should be true. If it is set to True, you will be asked by the software to set impurity manually in quant method: https://assets.thermofisher.com/TFS-Assets/LSG/certificate/Certificates-of-Analysis/A44520_UI292951.PDF



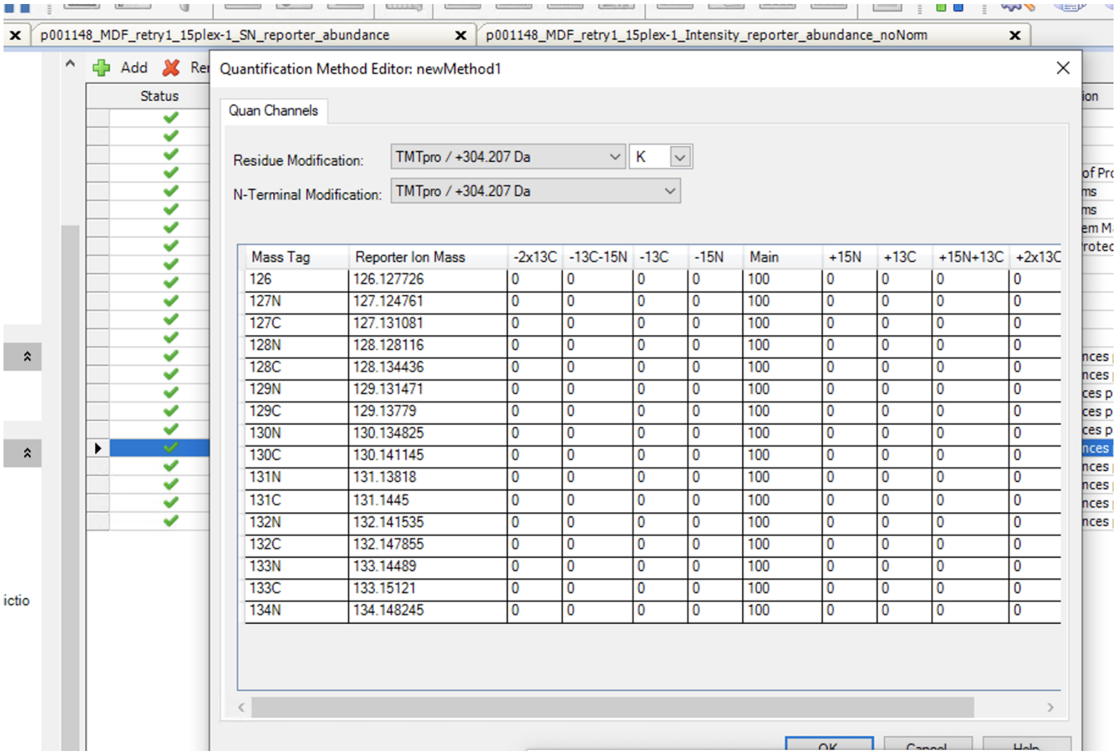
Parameters of 'Reporter Ions Quantifier'

Show Advanced Parameters

1. General Quantification Settings	
Peptides to Use	Unique + Razor
Consider Protein Groups for Pe	True
Use Shared Quan Results	False
Reject Quan Results with Miss	False
2. Reporter Quantification	
Reporter Abundance Based C	Intensity
Apply Quan Value Corrections	True
Co-Isolation Threshold	False
Average Reporter S/N Thresh	True
Normalized CHIMERYS Coeffi	
SPS Mass Matches [%] Threshl	65
Minimum Channel Occupancy	0
3. Normalization and Scaling	
Normalization Mode	Total Peptide Amount
Proteins For Normalization	
Scaling Mode	On All Average
4. Exclude Peptides from Protein Quantification	
For Normalization	Use All Peptides
For Protein Roll-Up	Use All Peptides
For Pairwise Ratios	Exclude Modified
1. Considered Peptide Modific	None
2. Considered Peptide Modific	None
3. Considered Peptide Modific	None
N-Terminal Considered Peptide	None
5. Quan Rollup and Hypothesis Testing	
Protein Ratio Calculation	Protein Abundance Based

Apply Quan Value Corrections
If set to TRUE, correction for the isotopic impurity of reporter quan values is applied.

The impurity values could be changed in the dialog below:


p001148_MDF_retry1_15plex-1_SN_reporter_abundance
p001148_MDF_retry1_15plex-1_Intensity_reporter_abundance_noNorm
Add
Status
OK
Cancel
Help

Quantification Method Editor: newMethod1

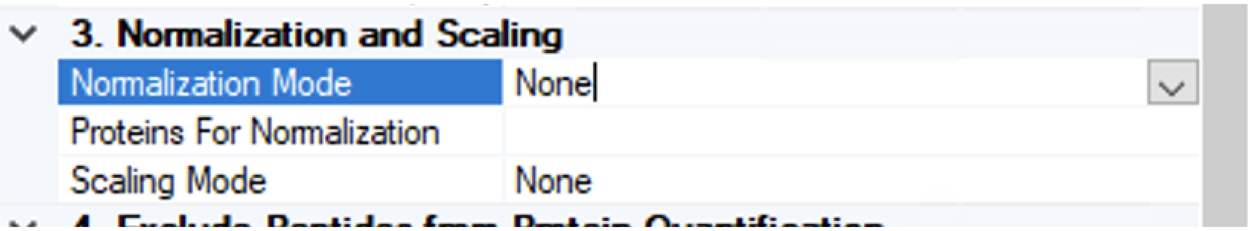
Quan Channels

Residue Modification: TMTpro / +304.207 Da K

N-Terminal Modification: TMTpro / +304.207 Da

Mass Tag	Reporter Ion Mass	-2x13C	-13C-15N	-13C	-15N	Main	+15N	+13C	+15N+13C	+2x13C
126	126.127726	0	0	0	0	100	0	0	0	0
127N	127.124761	0	0	0	0	100	0	0	0	0
127C	127.131081	0	0	0	0	100	0	0	0	0
128N	128.128116	0	0	0	0	100	0	0	0	0
128C	128.134436	0	0	0	0	100	0	0	0	0
129N	129.131471	0	0	0	0	100	0	0	0	0
129C	129.13779	0	0	0	0	100	0	0	0	0
130N	130.134825	0	0	0	0	100	0	0	0	0
130C	130.141145	0	0	0	0	100	0	0	0	0
131N	131.13818	0	0	0	0	100	0	0	0	0
131C	131.1445	0	0	0	0	100	0	0	0	0
132N	132.141535	0	0	0	0	100	0	0	0	0
132C	132.147855	0	0	0	0	100	0	0	0	0
133N	133.14489	0	0	0	0	100	0	0	0	0
133C	133.15121	0	0	0	0	100	0	0	0	0
134N	134.148245	0	0	0	0	100	0	0	0	0

3. Change Normalization mode to None because msTrawler will perform normalization as part of the algorithm.

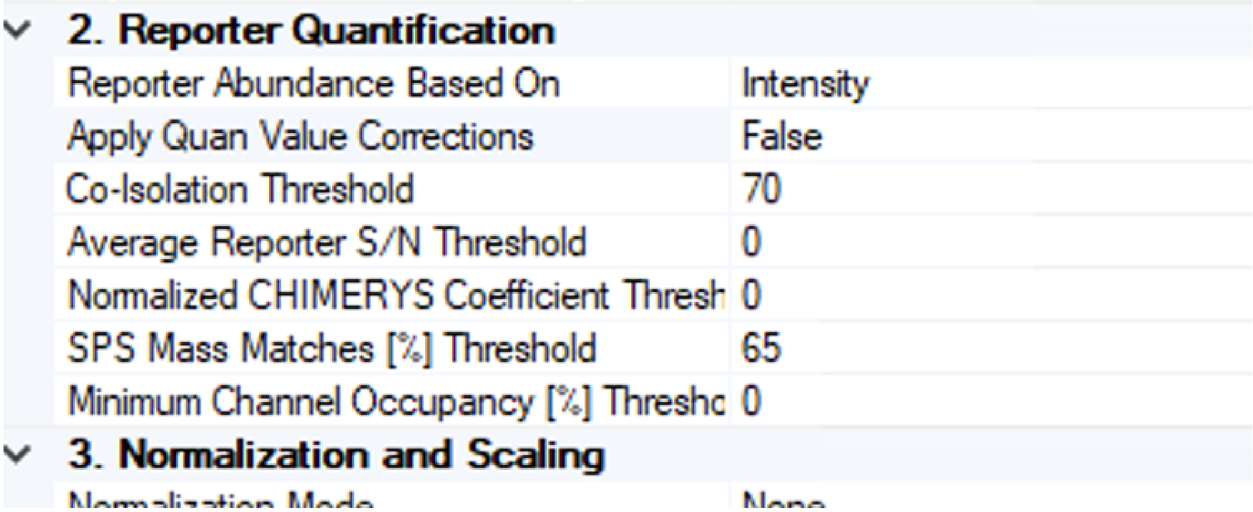


3. Normalization and Scaling

Normalization Mode	None
Proteins For Normalization	
Scaling Mode	None

4. Exclude Peptides from Protein Quantification

4. Average reporter SN threshold should be set to 0:



2. Reporter Quantification

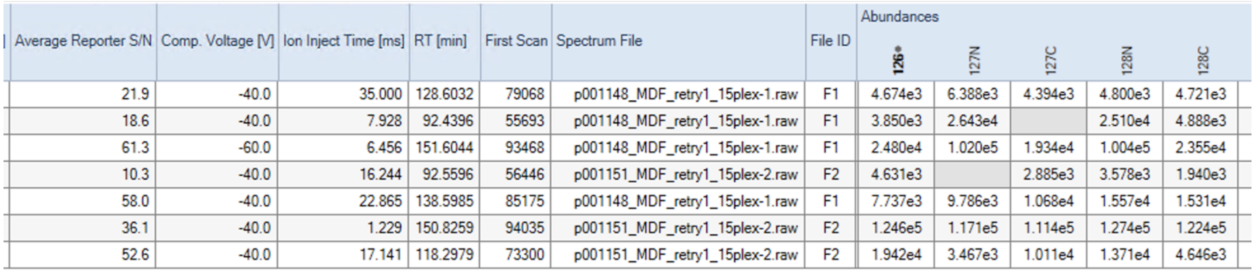
Reporter Abundance Based On	Intensity
Apply Quan Value Corrections	False
Co-Isolation Threshold	70
Average Reporter S/N Threshold	0
Normalized CHIMERYS Coefficient Thresh	0
SPS Mass Matches [%] Threshold	65
Minimum Channel Occupancy [%] Thresho	0

3. Normalization and Scaling

Normalization Mode	None

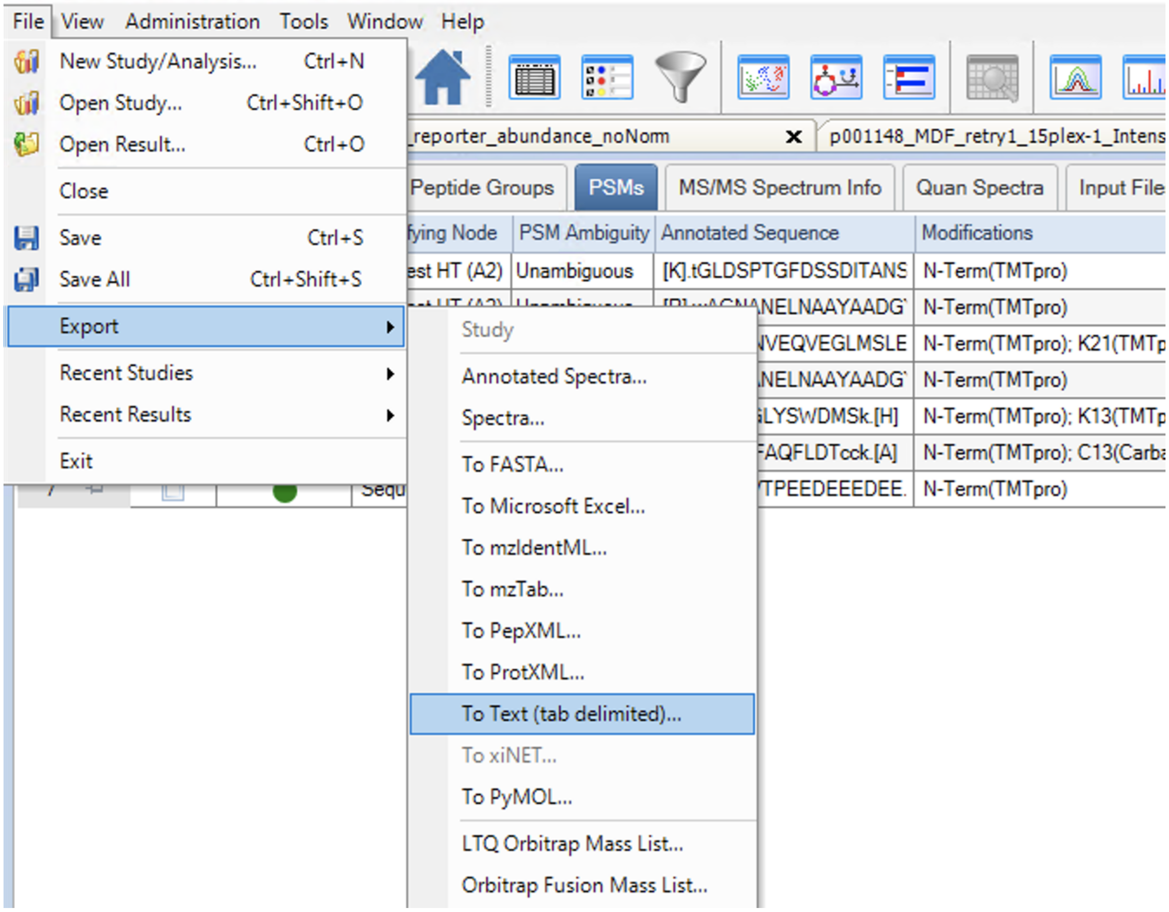
Export Proteome Discoverer Results

After the PD run, you will see each row in the peptide or PSM table corresponds to only 1 plex/file (File ID column below)

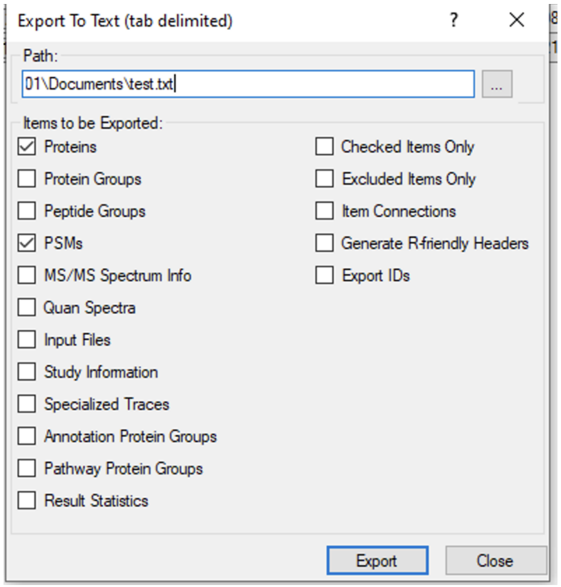


Average Reporter S/N	Comp. Voltage [V]	Ion Inject Time [ms]	RT [min]	First Scan	Spectrum File	File ID	Abundances				
							126	127N	127C	128N	128C
21.9	-40.0	35.000	128.6032	79068	p001148_MDF_retry1_15plex-1.raw	F1	4.674e3	6.388e3	4.394e3	4.800e3	4.721e3
18.6	-40.0	7.928	92.4396	55693	p001148_MDF_retry1_15plex-1.raw	F1	3.850e3	2.643e4		2.510e4	4.888e3
61.3	-60.0	6.456	151.6044	93468	p001148_MDF_retry1_15plex-1.raw	F1	2.480e4	1.020e5	1.934e4	1.004e5	2.355e4
10.3	-40.0	16.244	92.5596	56446	p001151_MDF_retry1_15plex-2.raw	F2	4.631e3		2.885e3	3.578e3	1.940e3
58.0	-40.0	22.865	138.5985	85175	p001148_MDF_retry1_15plex-1.raw	F1	7.737e3	9.786e3	1.068e4	1.557e4	1.531e4
36.1	-40.0	1.229	150.8259	94035	p001151_MDF_retry1_15plex-2.raw	F2	1.246e5	1.171e5	1.114e5	1.274e5	1.224e5
52.6	-40.0	17.141	118.2979	73300	p001151_MDF_retry1_15plex-2.raw	F2	1.942e4	3.467e3	1.011e4	1.371e4	4.646e3

To export results from PD, select “File” -> “Export” -> “To Text(tab delimited)..”:



“PSMs” export is required. To include gene information, “Proteins” could also be exported.



Export To Text (tab delimited)

Path:
01\Documents\test.txt

Items to be Exported:

☑ Proteins
□ Protein Groups
□ Peptide Groups
☑ PSMs
□ MS/MS Spectrum Info
□ Quan Spectra
□ Input Files
□ Study Information
□ Specialized Traces
□ Annotation Protein Groups
□ Pathway Protein Groups
□ Result Statistics

□ Checked Items Only
□ Excluded Items Only
□ Item Connections
□ Generate R-friendly Headers
□ Export IDs

Export | Close

Run msTrawler with PD files

1. Convert PD file

NOTE: Different versions of PD have slightly different file format. The columns below are required to run the conversion. If any columns have a different header, please change the header before running it.

Required column headers in exported PSM file (case sensitive):

- "Average Reporter S/N" or "Average Reporter SN"
- Master Protein Accessions
- Annotated Sequence
- Modifications
- File ID
- "Abundance: XXX" or "Abundance XXX"

Required column headers in exported protein file (case sensitive). Note that this file is optional. Without this file, Gene name will be missing but it does not affect data processing.

- Gene.Symbol
- Accession

After msTrawler is installed, you can use the following commands to run the converter:

```
# Load msTrawler library
> library('msTrawler')

# See usage of the converter
> ?convertPdFile

# Run PD converter and save the results in a dataframe
df <- convertPdFile("/PATH/TO/PSM/FILE", "/PATH/TO/PROTEIN/FILE")
```

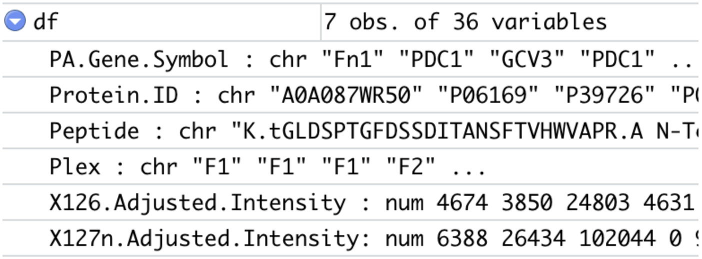
Example data are in the data folder of msTrawler code:

```
setwd("/FOLDER/PATH/msTrawler/data/tutorial_example")
df <- convertPdFile("pd_example_export_PSMs.txt", "pd_example_export_Proteins.txt")
```

Alternatively without protein file:

```
df <- convertPdFile("pd_example_export_PSMs.txt")
```

The df could be used as the dataframe in *msTrawl* or *protPrep* function. It should look similar to the example below. A copy of the df is also saved in the same folder with a “ _converted.csv” suffix.



```
df                          7 obs. of 36 variables
   PA.Gene.Symbol : chr "Fn1" "PDC1" "GCV3" "PDC1" ..
   Protein.ID : chr "A0A087WR50" "P06169" "P39726" "P
   Peptide : chr "K.tGLDSPTGFDSSDITANSFTVHWVAPR.A N-T
   Plex : chr "F1" "F1" "F1" "F2" ...
   X126.Adjusted.Intensity : num 4674 3850 24803 4631
   X127n.Adjusted.Intensity: num 6388 26434 102044 0
```

Please note that some protein ID from PD may contain multiple protein accessions due to peptide redundancy. Users who want to simplify protein group labels should try some algorithms at this stage to keep a single protein accession number. MsTrawler does not alter protein group assignment.

2. Load covariate file and sample file

Besides the data df created in previous step, msTrawler needs a covariate file and sample file which conveys the experimental design by defining factors, time series, etc. and corresponding sample level assignments. Refer to the example covariateFile.csv and sampleFile.csv in the data folder. Section **“Create covariate file and sample file”** discusses how to create these files.

```
setwd("/FOLDER/PATH/msTrawler/data/tutorial_example")

# read covariate_file.csv
covariateFile <- read.csv(file = 'covariate_file.csv', stringsAsFactors = FALSE)

# read sample_file.csv
sampleFile <- read.csv(file = 'sample_file.csv',stringsAsFactors = FALSE)
```

3. Run msTrawler with a single process

There are two ways to run msTrawler: as a single process or multiple processes with parallel processing. For multiple batches of data, especially when using the multi-level modeling options, parallel processing will greatly reduce the processing time. Please refer to Section 4 for parallel processing.

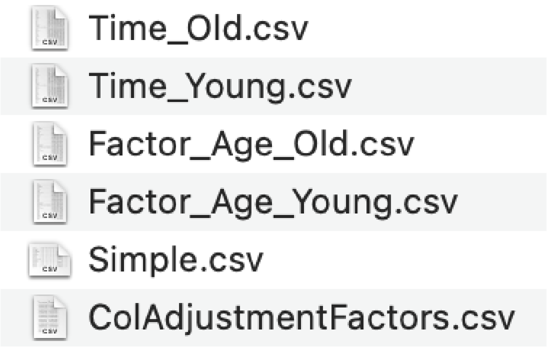
Example below is using files in the data/tutorial_example folder.

Function msTrawl is used for single process:

```
# See usage of the msTrawl function
> ?msTrawl

# run msTrawl function. N_SUM set to 9999 to trigger the use of msTrawler-SUM instead of the multilevel model; outlierCutoff is set to NULL because the example data is only for demonstration purposes.
  msTrawl(   df,
            sampleFile = sampleFile,
            covariateFile = covariateFile,
            scaleSN = 1,
            lod = 0.01,
            imputePenalty = 1,
            minAbove = 4,
            ssnFilter = 20,
            outlierCutoff = NULL,
            N_SUM = 9999,
            swapProtein = FALSE,
            maxPep = 25,
            colAdjust = 0.5,
            colRatios = NULL,
            dropContam = TRUE,
            dropReverse = TRUE,
            peptideAnalysis = FALSE,
            minRE = 5,
            timeDiff = TRUE)
```

After running the function, multiple output files will be generated in the same folder.



Time_Old.csv
Time_Young.csv
Factor_Age_Old.csv
Factor_Age_Young.csv
Simple.csv
ColAdjustmentFactors.csv

Refer to Section “Explanation of Export” for information on each file.

4. Parallel Preprocessing with protPrep and miniTrawl Function

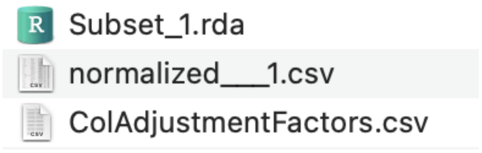
Alternative to running the msTrawl function, we can split the dataset in multiple subsets and run with parallel processing, which is especially useful for large datasets. Two functions are involved in this process: the protPrep function performs pre-processing and splitting data into subsets, and the miniTrawl function processes each subset.

```
# See usage and explain of each parameter of the protPrep
> ?protPrep
```

With protPrep function you can subset the data for parallel processing in subsequent steps. The main parameter is setSize (the number of proteins in each set). The typical setSize used is 50, meaning each result set contains 50 proteins.

```
# run protPrep funcion
   protPrep( df,
             setSize = 50,
             sampleFile = sampleFile,
             covariateFile = covariateFile,
             scaleSN = 1,
             lod = 0.01,
             imputePenalty = 1,
             minAbove = 3,
             ssnFilter = 20,
             outlierCutoff = NULL,
             N_SUM = 9999,
             swapProtein = FALSE,
             maxPep = 25,
             colAdjust = 0.5,
             colRatios = NULL,
             dropContam = TRUE,
             dropReverse = TRUE,
             peptideAnalysis = FALSE,
             minRE = 5,
             timeDiff = TRUE)
```

After running the protPrep function, several subset files will be generated in the input file folder. Each protein is randomly assigned a subset label, with the number of labels determined by the setSize parameter. If there are 150 proteins, there will be 3 sets of files like Subset_1.rda, Subset_2.rda, Subset_3.rda.



Subset_1.rda
normalized___1.csv
ColAdjustmentFactors.csv

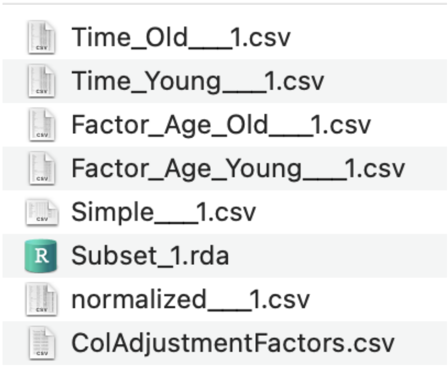
- Subset_X.rda file: contains all variables and results needed for the next step. Loading the rda file in R will populate an environment with all the variables required for the miniTrawl() function
- normalized___X.csv: a copy of input file after normalization
- ColAdjustmentFactors.csv: a copy of calculated column adjustment factors

For each subset file, run miniTrawl function

```
# See usage and explain of each parameter of the miniTrawl
> ?miniTrawl
```

```
# Run miniTrawl
> miniTrawl("/FOLDER/PATH/Subset_1.rda")
```

The output from miniTrawl is for each subset. To combine results from each subset, simply merge files from different subsets, e.g. merge Time_Old___1.csv, Time_Old___2.csv, Time_Old___2.csv together (usually this is done programmably).



Time_Old___1.csv
Time_Young___1.csv
Factor_Age_Old___1.csv
Factor_Age_Young___1.csv
Simple___1.csv
Subset_1.rda
normalized___1.csv
ColAdjustmentFactors.csv

Refer to Section “Explanation of Export” for information of each file.

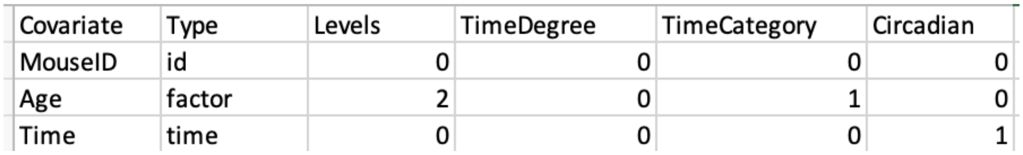
Create covariate file and sample file

MStrawler uses covariateFile and sampleFile to define factors, time series, etc. and corresponding sample level assignments. Example covariateFile and sampleFile are in the data folder.

This sample covariate file describes a longitudinal circadian proteomics study that allows us to estimate circadian patterns in both young and old mice (not real data!) and test for differences between them. The row names in the covariateFile must match to column names in the sample file. Covariate information is assigned to each sample in the sampleFile, while the covariateFile is used to form the logic around how that data will be handled.

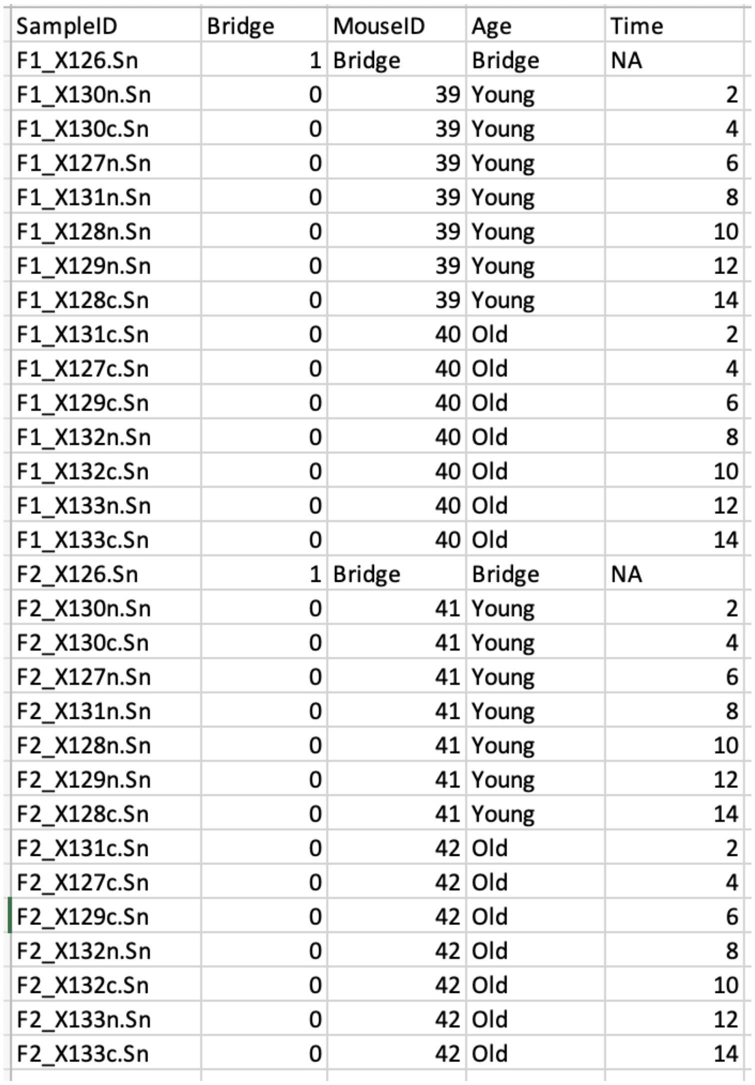
In the below example, there is one factor for age (“Young” or “Old”) which will generate a fixed effect parameter to estimate a difference in abundance between these groups. There is another variable classified with Type = “time”. This will be treated as a continuous covariate, but time variables have some special features in our software. Specifically, entering values of 1-3 into the covariateFile, under the TimeDegree column, would specify the creation of either a linear, quadratic or cubic time trend, respectively. Adding a 1 to the “Circadian” column, as we have done in this example, creates a two parameter sinusoidal timetrend, adding terms for sin((2 * pi / 24 *t) + cos((2 * pi / 24) * t) to the model. Additionally, a 1 can be entered in the “TimeCategory” column for any row representing a Type=factor variable. If this occurs, then the timetrend (linear, quadratic, cubic or circadian) will be estimated separately for each level of the selected factor. Hypothesis tests for differential time trends will be performed for each pair of factor levels. Finally, there is also an “id” variable, which will create a random intercept for each unique MouseID which can be used to create a correlation structure between samples from the same individual.

Example covariate_file.csv:



Covariate	Type	Levels	TimeDegree	TimeCategory	Circadian
MouseID	id	0	0	0	0
Age	factor	2	0	1	0
Time	time	0	0	0	1

Example sample_file.csv:



SampleID	Bridge	MouseID	Age	Time
F1_X126.Sn	1	Bridge	Bridge	NA
F1_X130n.Sn	0	39	Young	2
F1_X130c.Sn	0	39	Young	4
F1_X127n.Sn	0	39	Young	6
F1_X131n.Sn	0	39	Young	8
F1_X128n.Sn	0	39	Young	10
F1_X129n.Sn	0	39	Young	12
F1_X128c.Sn	0	39	Young	14
F1_X131c.Sn	0	40	Old	2
F1_X127c.Sn	0	40	Old	4
F1_X129c.Sn	0	40	Old	6
F1_X132n.Sn	0	40	Old	8
F1_X132c.Sn	0	40	Old	10
F1_X133n.Sn	0	40	Old	12
F1_X133c.Sn	0	40	Old	14
F2_X126.Sn	1	Bridge	Bridge	NA
F2_X130n.Sn	0	41	Young	2
F2_X130c.Sn	0	41	Young	4
F2_X127n.Sn	0	41	Young	6
F2_X131n.Sn	0	41	Young	8
F2_X128n.Sn	0	41	Young	10
F2_X129n.Sn	0	41	Young	12
F2_X128c.Sn	0	41	Young	14
F2_X131c.Sn	0	42	Old	2
F2_X127c.Sn	0	42	Old	4
F2_X129c.Sn	0	42	Old	6
F2_X132n.Sn	0	42	Old	8
F2_X132c.Sn	0	42	Old	10
F2_X133n.Sn	0	42	Old	12
F2_X133c.Sn	0	42	Old	14

Creating covariate file:

Bridge: specify the bridge to be 1 or 0 in the csv file
For the bridge, the covariate columns can be blank or whatever input => won't affect the model

Other than the bridge, don't have any missing value in the parameter files
Covariate: the exact name as the column name in the csv file

Type:

- Factor: for categorical variables (for e.g. GF vs. SPF, Day vs. Night)
- Continuous: a numeric data type (for e.g. BMI, weight)
  - will be centered
  - no polynomial fit

- Time: a time value. In the sample file, the corresponding column name must spell **“Time”** (make sure capitalized **T**)
    - start from 0 and will not be centered
    - polynomial fit
    - time trend can be fitted for all levels of a “Factor” covariate => allow hypothesis testing between the fit models
    - Can be used for “non-time” continuous variables whenever separate associations are wanted for each level of a factor
- ID (only for longitudinal samples): to make sure we account for the variance of sample from the same individual
    - Cross sectional time course vs. longitudinal
        - longitudinal: same individual over time
        - cross sectional: different times sample from a population
    - Only use when we have repeating measurements for all the ids
    - Repeating measurement is not technically replicate - repeating measurements are samples from the same individual at different times.

Levels: does not do anything
Time degree (only for “Time” type covariate): the degree of fitting model to fit; 1 is linear, 2 is quadratic, 3 is cubic
Time Category (only relevant when there is a “Time” covariate and we would like to estimate different time trends for different levels of a categorical variable - binary (0 or 1)

Circadian: binary 0 or 1

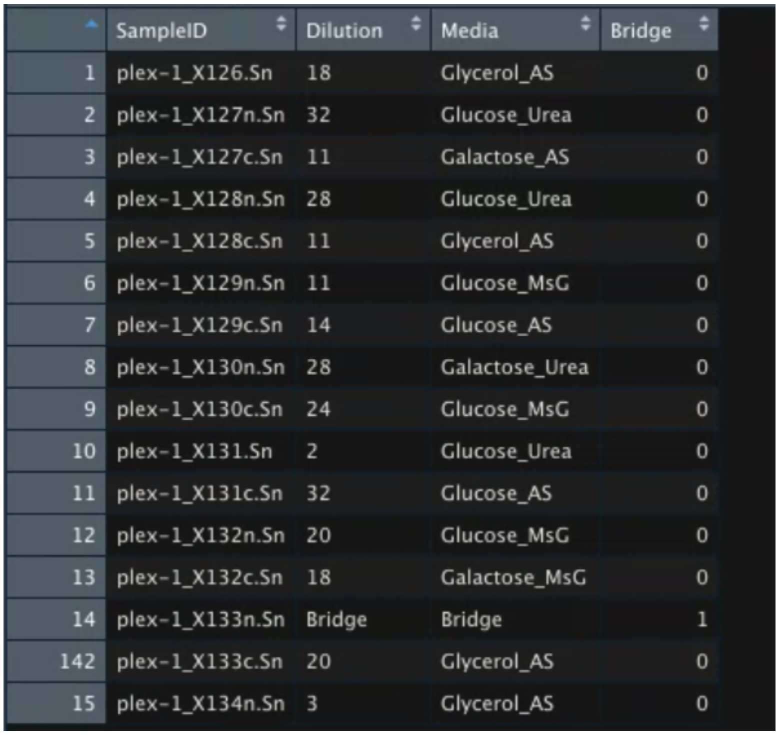
Creating Sample file:

Important notes for creating a new sample file:

- File always contains **SampleID** column that has to match samples in the PD output
    - There is always an X before the channel name (“X126”)
    - SampleID prefix is the plex name and must always be included
        - For pd the column “File ID” is used as plex name
    - SampleID channel names also need to match the channels in PD abundance columns or else it will fail (will pull 0s if done incorrectly)
        - Use lowercase letters for channel, e.g. 133n rather than 133N
- Sample Parameter File is used to assign sample specific **covariates**
    - Each covariate is its own column
- **Bridge**: If plex contains a bridge, create a Bridge column and assign 0s and 1s
    - Assign **1** to the bridge
    - Assign 0 to all other channels
    - In the absence of a “Bridge” column in the sampleFile, row normalization will be performed on every scan. This is not recommended for multi-batch experiments with any variation in the design across batches. However, we do not prevent such applications. Use at your own risk!

- **Missing channels**: To exclude unused or unwanted channels, remove the channel from the sample parameter file (delete row)
- Sample Parameter File is always a **csv**

Additional example: Plex-1 of 16-plex design containing 2 covariates (dilution, media) and a bridge



	SampleID	Dilution	Media	Bridge
1	plex-1_X126.Sn	18	Glycerol_AS	0
2	plex-1_X127n.Sn	32	Glucose_Urea	0
3	plex-1_X127c.Sn	11	Galactose_AS	0
4	plex-1_X128n.Sn	28	Glucose_Urea	0
5	plex-1_X128c.Sn	11	Glycerol_AS	0
6	plex-1_X129n.Sn	11	Glucose_MsG	0
7	plex-1_X129c.Sn	14	Glucose_AS	0
8	plex-1_X130n.Sn	28	Galactose_Urea	0
9	plex-1_X130c.Sn	24	Glucose_MsG	0
10	plex-1_X131.Sn	2	Glucose_Urea	0
11	plex-1_X131c.Sn	32	Glucose_AS	0
12	plex-1_X132n.Sn	20	Glucose_MsG	0
13	plex-1_X132c.Sn	18	Galactose_MsG	0
14	plex-1_X133n.Sn	Bridge	Bridge	1
142	plex-1_X133c.Sn	20	Glycerol_AS	0
15	plex-1_X134n.Sn	3	Glycerol_AS	0

Explanation of Export

Exported csv files

- Normalized file:
  - Protein quant data after column/row normalization.
  - An explanation of “SNR” vs. “lIntensity” (i.e. 126Sn vs 126 Adjusted Intensity):

    * “SNR” Reflects the quality of the data and is used to add weights to the models. These values may have been reduced if the intensities fell below the specified LOQ. They are never column normalized.

    * “lIntensity” These are the log2, isotope purity adjusted intensities after LOQ handling and column normalization
  - “techVar” = 1/SNR
- Factor_[factor name]_[factor level] file:

    - Factor_Age_Old means this file will contain all estimates and p-values for changes in abundance between levels of “factor name” using “factor level” as the reference.
    - The estimated log2 difference can be found in the Est_[Level] column
    - There is a lot of redundancy between the various tables from the same factor as differences only occur due to missing data
- Time_[factor name] file:
    - Time series results for a certain factor, e.g. Time_old, will show estimated time trends, a test for a non-zero time trend (joint test on all parameters if TimeDegree > 1), and differential time trends across levels of TimeCategory relative to the “old” condition.
- Simple file:
    - An overall summary table for the log fold change for all samples.
    - In the presence of a bridge channel these can be interpreted as the average weighted difference in log2 intensities between the bridge and each sample. Without a bridge, this provides the average weighted, row normalized, log2 intensity in each sample.
    - This table is not used in any statistical analyses in msTrawler. We primarily generate these numbers to aid in creating visualizations.