

# Package ‘proBatch’

November 14, 2019

**Type** Package

**Title** Tools for Diagnostics and Corrections of Batch Effects in Proteomics

**Version** 1.3.0

**Author** Jelena Cuklina <chuklina.jelena@gmail.com>, Chloe H. Lee <chloe.h.lee94@gmail.com>, Patrick Pedrioli <pedrioli@gmail.com>

**Maintainer** Jelena Cuklina <chuklina.jelena@gmail.com>

**Description** These tools facilitate batch effects analysis and correction in high-throughput experiments. It was developed primarily for mass-spectrometry proteomics (DIA/SWATH), but could also be applicable to most omic data with minor adaptations. The package contains functions for diagnostics (proteome/genome-wide and feature-level), correction (normalization and batch effects correction) and quality control. Non-linear fitting based approaches were also included to deal with complex, mass spectrometry-specific signal drifts.

**biocViews** BatchEffect, Normalization, Preprocessing, Software, MassSpectrometry, Proteomics, QualityControl

**License** GPL-3

**URL** <https://github.com/symbioticMe/proBatch>

**BugReports** <https://github.com/symbioticMe/proBatch/issues>

**Depends** R (>= 3.6)

**Encoding** UTF-8

**LazyData** true

**Imports** Biobase, corrplot, dplyr, data.table, ggfortify, ggplot2, grDevices, lazyeval, lubridate, magrittr, pheatmap, preprocessCore, purrr, pvca, RColorBrewer, reshape2, rlang, scales, stats, sva, tidyr, tibble, tools, utils, viridis, wesanderson, WGCNA

**Suggests** knitr, rmarkdown, devtools, ggpubr, gtable, gridExtra, roxygen2, testthat (>= 2.1.0), spelling

**VignetteBuilder** knitr

**RoxygenNote** 6.1.1

**Language** en-US

**git\_url** <https://git.bioconductor.org/packages/proBatch>

**git\_branch** master

**git\_last\_commit** 6241f81

**git\_last\_commit\_date** 2019-10-29

**Date/Publication** 2019-11-13

## R topics documented:

calculate_feature_CV . . . . .	3
calculate_peptide_corr_distr . . . . .	4
calculate_PVCA . . . . .	5
calculate_sample_corr_distr . . . . .	6
check_sample_consistency . . . . .	7
correct_batch_effects . . . . .	8
create_peptide_annotation . . . . .	12
dates_to_posix . . . . .	13
date_to_sample_order . . . . .	14
define_sample_order . . . . .	15
example_peptide_annotation . . . . .	16
example_proteome . . . . .	16
example_proteome_matrix . . . . .	17
example_sample_annotation . . . . .	17
feature_level_diagnostics . . . . .	18
fit_nonlinear . . . . .	22
long_to_matrix . . . . .	23
matrix_to_long . . . . .	24
normalize . . . . .	25
plot_corr_matrix . . . . .	27
plot_CV_distr . . . . .	28
plot_CV_distr.df . . . . .	30
plot_heatmap_diagnostic . . . . .	30
plot_heatmap_generic . . . . .	32
plot_hierarchical_clustering . . . . .	34
plot_PCA . . . . .	36
plot_peptide_corr_distribution . . . . .	37
plot_protein_corrplot . . . . .	39
plot_PVCA . . . . .	40
plot_PVCA.df . . . . .	42
plot_sample_corr_distribution . . . . .	43
plot_sample_corr_heatmap . . . . .	44
plot_sample_mean_or_boxplot . . . . .	46
plot_split_violin_with_boxplot . . . . .	49
prepare_PVCA_df . . . . .	49
proBatch . . . . .	51
sample_annotation_to_colors . . . . .	53
transform_raw_data . . . . .	54
<b>Index</b>	<b>56</b>

---

calculate\_feature\_CV *Calculate CV distribution for each feature*

---

### Description

Calculate CV distribution for each feature

### Usage

```
calculate_feature_CV(df_long, sample_annotation = NULL,
  feature_id_col = "peptide_group_label",
  sample_id_col = "FullRunName", measure_col = "Intensity",
  batch_col = NULL, biospecimen_id_col = NULL, unlog = TRUE,
  log_base = 2, offset = 1)
```

### Arguments

df_long	data frame where each row is a single feature in a single sample. It minimally has a sample_id_col, a feature_id_col and a measure_col, but usually also an m_score (in OpenSWATH output result file). See help("example_proteome") for more details.
sample_annotation	data frame with: <ol style="list-style-type: none"> <li>1. sample_id_col (this can be repeated as row names)</li> <li>2. biological covariates</li> <li>3. technical covariates (batches etc)</li> </ol> . See help("example_sample_annotation")
feature_id_col	name of the column with feature/gene/peptide/protein ID used in the long format representation df_long. In the wide formatted representation data_matrix this corresponds to the row names.
sample_id_col	name of the column in sample_annotation table, where the filenames (col-names of the data_matrix are found).
measure_col	if df_long is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency.
batch_col	column in sample_annotation that should be used for batch comparison (or other, non-batch factor to be mapped to color in plots).
biospecimen_id_col	column in sample_annotation that defines a unique bio ID, which is usually a combination of conditions or groups. Tip: if such ID is absent, but can be defined from several columns, create new biospecimen_id column
unlog	(logical) whether to reverse log transformation of the original data
log_base	base of the logarithm for transformation
offset	small positive number to prevent 0 conversion to -Inf

### Value

data frame with Total CV for each feature & (optionally) per-batch CV

**Examples**

```
CV_df = calculate_feature_CV(example_proteome,
sample_annotation = example_sample_annotation,
measure_col = 'Intensity',
batch_col = 'MS_batch')
```

---

```
calculate_peptide_corr_distr
```

*Calculate peptide correlation between and within peptides of one protein*

---

**Description**

Calculate peptide correlation between and within peptides of one protein

**Usage**

```
calculate_peptide_corr_distr(data_matrix, peptide_annotation,
protein_col = "ProteinName", feature_id_col = "peptide_group_label")
```

**Arguments**

`data_matrix` features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example\_proteome\_matrix" for more details (to call the description, use `help("example_proteome_matrix")`)

`peptide_annotation` long format data frame with peptide ID and their corresponding protein and/or gene annotations. See `help("example_peptide_annotation")`.

`protein_col` column where protein names are specified

`feature_id_col` name of the column with feature/gene/peptide/protein ID used in the long format representation `df_long`. In the wide formatted representation `data_matrix` this corresponds to the row names.

**Value**

dataframe with peptide correlation coefficients that are suggested to use for plotting in [plot\\_peptide\\_corr\\_distribution](#) as `plot_param`:

**Examples**

```
selected_genes = c('BOVINE_A1ag', 'BOVINE_FetuinB', 'Cyfip1')
gene_filter = example_peptide_annotation$Gene %in% selected_genes
peptides_ann = example_peptide_annotation$peptide_group_label
selected_peptides = peptides_ann[gene_filter]
matrix_test = example_proteome_matrix[selected_peptides,]
pep_annotation_sel = example_peptide_annotation[gene_filter, ]
corr_distribution = calculate_peptide_corr_distr(matrix_test,
pep_annotation_sel, protein_col = 'Gene')
```

---

calculate_PVCA	<i>Calculate variance distribution by variable</i>
----------------	--

---

### Description

Calculate variance distribution by variable

### Usage

```
calculate_PVCA(data_matrix, sample_annotation,
  feature_id_col = "peptide_group_label",
  sample_id_col = "FullRunName", factors_for_PVCA = c("MS_batch",
  "digestion_batch", "Diet", "Sex", "Strain"), pca_threshold = 0.6,
  variance_threshold = 0.01, fill_the_missing = -1)
```

### Arguments

**data\_matrix** features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example\_proteome\_matrix" for more details (to call the description, use help("example\_proteome\_matrix"))

**sample\_annotation** data frame with:

1. sample\_id\_col (this can be repeated as row names)
2. biological covariates
3. technical covariates (batches etc)

. See help("example\_sample\_annotation")

**feature\_id\_col** name of the column with feature/gene/peptide/protein ID used in the long format representation df\_long. In the wide formatted representation data\_matrix this corresponds to the row names.

**sample\_id\_col** name of the column in sample\_annotation table, where the filenames (colnames of the data\_matrix are found).

**factors\_for\_PVCA** vector of factors from sample\_annotation, that are used in PVCA analysis

**pca\_threshold** the percentile value of the minimum amount of the variabilities that the selected principal components need to explain

**variance\_threshold** the percentile value of weight each of the factors needs to explain (the rest will be lumped together)

**fill\_the\_missing** numeric value determining how missing values should be substituted. If NULL, features with missing values are excluded.

### Value

data frame of weights of Principal Variance Components

**Examples**

```
matrix_test <- example_proteome_matrix[1:150, ]
pvca_df <- calculate_PVCA(matrix_test, example_sample_annotation,
  factors_for_PVCA = c('MS_batch', 'digestion_batch', "Diet", "Sex", "Strain"),
  pca_threshold = .6, variance_threshold = .01, fill_the_missing = -1)
```

---

```
calculate_sample_corr_distr
```

*Calculates correlation for all pairs of the samples in data matrix, labels as replicated/same\_batch/unrelated in output columns (see "Value").*

---

**Description**

Calculates correlation for all pairs of the samples in data matrix, labels as replicated/same\_batch/unrelated in output columns (see "Value").

**Usage**

```
calculate_sample_corr_distr(data_matrix, sample_annotation,
  repeated_samples = NULL, biospecimen_id_col = "EarTag",
  sample_id_col = "FullRunName", batch_col = "MS_batch")
```

**Arguments**

**data\_matrix** features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example\_proteome\_matrix" for more details (to call the description, use help("example\_proteome\_matrix"))

**sample\_annotation** data frame with:

1. sample\_id\_col (this can be repeated as row names)
2. biological covariates
3. technical covariates (batches etc)

. See help("example\_sample\_annotation")

**repeated\_samples** vector of sample IDs to evaluate, if NULL, all samples are taken into account for plotting

**biospecimen\_id\_col** column in sample\_annotation that defines a unique bio ID, which is usually a combination of conditions or groups. Tip: if such ID is absent, but can be defined from several columns, create new biospecimen\_id column

**sample\_id\_col** name of the column in sample\_annotation table, where the filenames (colnames of the data\_matrix are found).

**batch\_col** column in sample\_annotation that should be used for batch comparison (or other, non-batch factor to be mapped to color in plots).

**Value**

dataframe with the following columns, that are suggested to use for plotting in [plot\\_sample\\_corr\\_distribution](#) as plot\_param:

1. replicate
2. batch\_the\_same
3. batch\_replicate
4. batches

other columns are:

1. sample\_id\_1 & sample\_id\_2, both generated from sample\_id\_col variable
2. correlation - correlation of two corresponding samples
3. batch\_1 & batch\_2 or analogous, created the same as sample\_id\_1

**Examples**

```
corr_distribution = calculate_sample_corr_distr(data_matrix = example_proteome_matrix,
sample_annotation = example_sample_annotation,
batch_col = 'MS_batch', biospecimen_id_col = "EarTag")
```

---

check\_sample\_consistency

*Check if sample annotation is consistent with data matrix and join the two*

---

**Description**

Check if sample annotation is consistent with data matrix and join the two

**Usage**

```
check_sample_consistency(sample_annotation, sample_id_col, df_long,
batch_col = NULL, order_col = NULL, facet_col = NULL,
merge = TRUE)
```

**Arguments**

sample\_annotation

data frame with:

1. sample\_id\_col (this can be repeated as row names)
2. biological covariates
3. technical covariates (batches etc)

. See help("example\_sample\_annotation")

sample\_id\_col name of the column in sample\_annotation table, where the filenames (col-names of the data\_matrix are found).

df_long	data frame where each row is a single feature in a single sample. It minimally has a sample_id_col, a feature_id_col and a measure_col, but usually also an m_score (in OpenSWATH output result file). See help("example_proteome") for more details.
batch_col	column in sample_annotation that should be used for batch comparison (or other, non-batch factor to be mapped to color in plots).
order_col	column in sample_annotation that determines sample order. It is used for in initial assessment plots ( <a href="#">plot_sample_mean_or_boxplot</a> ) and feature-level diagnostics ( <a href="#">feature_level_diagnostics</a> ). Can be 'NULL' if sample order is irrelevant (e.g. in genomic experiments). For more details, order definition/inference, see <a href="#">define_sample_order</a> and <a href="#">date_to_sample_order</a>
facet_col	column in sample_annotation with a batch factor to separate plots into facets; usually 2nd to batch_col. Most meaningful for multi-instrument MS experiments (where each instrument has its own order-associated effects (see order_col) or simultaneous examination of two batch factors (e.g. preparation day and measurement day). For single-instrument case should be set to 'NULL'
merge	(logical) whether to merge df_long with sample_annotation or not

### Value

df\_long format data frame, merged with sample\_annotation using inner\_join (samples represented in both)

### Examples

```
df_test = check_sample_consistency(sample_annotation = example_sample_annotation,
df_long = example_proteome, sample_id_col = 'FullRunName',
batch_col = NULL, order_col = NULL, facet_col = NULL)
```

---

correct\_batch\_effects *Batch correction of normalized data*

---

### Description

Batch correction of normalized data. Batch correction brings each feature in each batch to the comparable shape. Currently the following batch correction functions are implemented:

1. Per-feature median centering: `center_feature_batch_medians_df()`. Median centering of the features (per batch median).
2. correction with ComBat: `correct_with_ComBat_df()`. Adjusts for discrete batch effects using ComBat. ComBat, described in Johnson et al. 2007. It uses either parametric or non-parametric empirical Bayes frameworks for adjusting data for batch effects. Users are returned an expression matrix that has been corrected for batch effects. The input data are assumed to be free of missing values and normalized before batch effect removal. Please note that missing values are common in proteomics, which is why in some cases corrections like `center_peptide_batch_medians_df` are more appropriate.



3. Continuous drift correction: `adjust_batch_trend_df()`. Adjust batch signal trend with the custom (continuous) fit. Should be followed by discrete corrections, e.g. `center_feature_batch_medians_df()` or `correct_with_ComBat_df()`.

Alternatively, one can call the correction function with `correct_batch_effects_df()` wrapper. Batch correction method allows correction of continuous signal drift within batch (if required) and adjustment for discrete difference across batches.

## Usage

```
center_feature_batch_medians_df(df_long, sample_annotation = NULL,
  sample_id_col = "FullRunName", batch_col = "MS_batch",
  feature_id_col = "peptide_group_label", measure_col = "Intensity",
  keep_all = "default", no_fit_imputed = TRUE, qual_col = NULL,
  qual_value = NULL)
```

```
center_feature_batch_medians_dm(data_matrix, sample_annotation,
  sample_id_col = "FullRunName", batch_col = "MS_batch",
  feature_id_col = "peptide_group_label", measure_col = "Intensity")
```

```
center_feature_batch_means_df(df_long, sample_annotation = NULL,
  sample_id_col = "FullRunName", batch_col = "MS_batch",
  feature_id_col = "peptide_group_label", measure_col = "Intensity",
  keep_all = "default", no_fit_imputed = TRUE, qual_col = NULL,
  qual_value = NULL)
```

```
center_feature_batch_means_dm(data_matrix, sample_annotation,
  sample_id_col = "FullRunName", batch_col = "MS_batch",
  feature_id_col = "peptide_group_label", measure_col = "Intensity")
```

```
adjust_batch_trend_df(df_long, sample_annotation = NULL,
  batch_col = "MS_batch", feature_id_col = "peptide_group_label",
  sample_id_col = "FullRunName", measure_col = "Intensity",
  order_col = "order", keep_all = "default",
  fit_func = "loess_regression", no_fit_imputed = TRUE,
  qual_col = NULL, qual_value = NULL, min_measurements = 8, ...)
```

```
adjust_batch_trend_dm(data_matrix, sample_annotation,
  batch_col = "MS_batch", feature_id_col = "peptide_group_label",
  sample_id_col = "FullRunName", measure_col = "Intensity",
  order_col = "order", fit_func = "loess_regression",
  return_fit_df = TRUE, min_measurements = 8, ...)
```

```
correct_with_ComBat_df(df_long, sample_annotation = NULL,
  feature_id_col = "peptide_group_label", measure_col = "Intensity",
  sample_id_col = "FullRunName", batch_col = "MS_batch",
  par.prior = TRUE, no_fit_imputed = TRUE, qual_col = NULL,
  qual_value = NULL, keep_all = "default")
```

```
correct_with_ComBat_dm(data_matrix, sample_annotation = NULL,
  feature_id_col = "peptide_group_label", measure_col = "Intensity",
  sample_id_col = "FullRunName", batch_col = "MS_batch",
  par.prior = TRUE)
```

```
correct_batch_effects_df(df_long, sample_annotation,
  continuous_func = NULL, discrete_func = c("MedianCentering",
  "MeanCentering", "ComBat"), batch_col = "MS_batch",
  feature_id_col = "peptide_group_label",
  sample_id_col = "FullRunName", measure_col = "Intensity",
  order_col = "order", keep_all = "default", no_fit_imputed = TRUE,
  qual_col = NULL, qual_value = NULL, min_measurements = 8, ...)
```

```
correct_batch_effects_dm(data_matrix, sample_annotation,
  continuous_func = NULL, discrete_func = c("MedianCentering",
  "ComBat"), batch_col = "MS_batch",
  feature_id_col = "peptide_group_label",
  sample_id_col = "FullRunName", measure_col = "Intensity",
  order_col = "order", min_measurements = 8, ...)
```

## Arguments

<code>df_long</code>	data frame where each row is a single feature in a single sample. It minimally has a <code>sample_id_col</code> , a <code>feature_id_col</code> and a <code>measure_col</code> , but usually also an <code>m_score</code> (in OpenSWATH output result file). See <code>help("example_proteome")</code> for more details.
<code>sample_annotation</code>	data frame with: <ol style="list-style-type: none"> <li>1. <code>sample_id_col</code> (this can be repeated as row names)</li> <li>2. biological covariates</li> <li>3. technical covariates (batches etc)</li> </ol> . See <code>help("example_sample_annotation")</code>
<code>sample_id_col</code>	name of the column in <code>sample_annotation</code> table, where the filenames (colnames of the <code>data_matrix</code> are found).
<code>batch_col</code>	column in <code>sample_annotation</code> that should be used for batch comparison (or other, non-batch factor to be mapped to color in plots).
<code>feature_id_col</code>	name of the column with feature/gene/peptide/protein ID used in the long format representation <code>df_long</code> . In the wide formatted representation <code>data_matrix</code> this corresponds to the row names.
<code>measure_col</code>	if <code>df_long</code> is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency.
<code>keep_all</code>	when transforming the data (normalize, correct) - acceptable values: all/default/minimal (which set of columns be kept).
<code>no_fit_imputed</code>	(logical) whether to use imputed (requant) values, as flagged in <code>qual_col</code> by <code>qual_value</code> for data transformation
<code>qual_col</code>	column to color point by certain value denoted by <code>color_by_qual_value</code> . Design with inferred/requant values in OpenSWATH output data, which means argument value has to be set to <code>m_score</code> .
<code>qual_value</code>	value in <code>qual_col</code> to color. For OpenSWATH data, this argument value has to be set to 2 (this is an <code>m_score</code> value for imputed values (requant values)).
<code>data_matrix</code>	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example_proteome_matrix" for more details (to call the description, use <code>help("example_proteome_matrix")</code> )

order_col	column in sample_annotation that determines sample order. It is used for in initial assessment plots ( <a href="#">plot_sample_mean_or_boxplot</a> ) and feature-level diagnostics ( <a href="#">feature_level_diagnostics</a> ). Can be 'NULL' if sample order is irrelevant (e.g. in genomic experiments). For more details, order definition/inference, see <a href="#">define_sample_order</a> and <a href="#">date_to_sample_order</a>
fit_func	function to fit the (non)-linear trend
min_measurements	the number of samples in a batch required for curve fitting.
...	other parameters, usually of <a href="#">adjust_batch_trend</a> , and <a href="#">fit_func</a> .
return_fit_df	(logical) whether to return the fit_df from <a href="#">adjust_batch_trend_dm</a> or only the data matrix
par.prior	use parametrical or non-parametrical prior
continuous_func	function to use for the fit (currently only <a href="#">loess_regression</a> available); if order-associated fix is not required, should be NULL.
discrete_func	function to use for adjustment of discrete batch effects ( <a href="#">MedianCentering</a> or <a href="#">ComBat</a> ).

### Value

the data in the same format as input ([data\\_matrix](#) or [df\\_long](#)). For [df\\_long](#) the data frame stores the original values of [measure\\_col](#) in another column called "preBatchCorr\_[[measure\\_col](#)]", and the normalized values in [measure\\_col](#) column.

The function [adjust\\_batch\\_trend\\_dm\(\)](#), if [return\\_fit\\_df](#) is TRUE returns list of two items:

1. [data\\_matrix](#)
2. [fit\\_df](#), used to examine the fitting curves

### See Also

[fit\\_nonlinear](#)

[fit\\_nonlinear](#), [plot\\_with\\_fitting\\_curve](#)

[fit\\_nonlinear](#), [plot\\_with\\_fitting\\_curve](#)

### Examples

```
#Median centering per feature per batch:
median_centered_df <- center_feature_batch_medians_df(
  example_proteome, example_sample_annotation)

#Correct with ComBat:
combat_corrected_df <- correct_with_ComBat_df(example_proteome,
  example_sample_annotation)

#Adjust the MS signal drift:
test_peptides = unique(example_proteome$peptide_group_label)[1:3]
test_peptide_filter = example_proteome$peptide_group_label %in% test_peptides
test_proteome = example_proteome[test_peptide_filter,]
adjusted_df <- adjust_batch_trend_df(test_proteome,
  example_sample_annotation, span = 0.7,
  min_measurements = 8)
```

```

plot_fit <- plot_with_fitting_curve(unique(adjusted_df$peptide_group_label),
df_long = adjusted_df, measure_col = 'preTrendFit_Intensity',
fit_df = adjusted_df, sample_annotation = example_sample_annotation)

#Correct the data in one go:
batch_corrected_matrix <- correct_batch_effects_df(example_proteome,
example_sample_annotation,
continuous_func = 'loess_regression',
discrete_func = 'MedianCentering',
batch_col = 'MS_batch',
span = 0.7, min_measurements = 8)

```

---

```
create_peptide_annotation
```

*Prepare peptide annotation from long format data frame Create light-weight peptide annotation data frame for selection of illustrative proteins*

---

## Description

Prepare peptide annotation from long format data frame

Create light-weight peptide annotation data frame for selection of illustrative proteins

## Usage

```

create_peptide_annotation(df_long,
  feature_id_col = "peptide_group_label",
  protein_col = c("ProteinName", "Gene"))

```

## Arguments

df_long	data frame where each row is a single feature in a single sample. It minimally has a sample_id_col, a feature_id_col and a measure_col, but usually also an m_score (in OpenSWATH output result file). See help("example_proteome") for more details.
feature_id_col	name of the column with feature/gene/peptide/protein ID used in the long format representation df_long. In the wide formatted representation data_matrix this corresponds to the row names.
protein_col	column where protein names are specified

## Value

data frame containing peptide annotations

## See Also

[plot\\_peptides\\_of\\_one\\_protein](#), [plot\\_protein\\_corrplot](#)

**Examples**

```
generated_peptide_annotation <- create_peptide_annotation(
  example_proteome, feature_id_col = "peptide_group_label",
  protein_col = c("Protein"))
```

---

dates\_to\_posix                      *Convert data/time to POSIXct*

---

**Description**

convert date/time column of sample\_annotation to POSIX format required to keep number-like behavior

**Usage**

```
dates_to_posix(sample_annotation, time_column = c("RunDate", "RunTime"),
  new_time_column = "DateTime", dateTimeFormat = c("%b_%d",
  "%H:%M:%S"), tz = "GMT")
```

**Arguments**

sample\_annotation  
                   data frame with:

1. sample\_id\_col (this can be repeated as row names)
2. biological covariates
3. technical covariates (batches etc)

. See help("example\_sample\_annotation")

time\_column        name of the column(s) where run date & time are specified. These will be used to determine the run order

new\_time\_column    name of the new column to which date&time will be converted to

dateTimeFormat    POSIX format of the date and time. See [as.POSIXct](#) from base R for details

tz                  for time zone

**Value**

sample annotation file with a new column new\_time\_column with POSIX-formatted date

**Examples**

```
date_to_posix <- dates_to_posix(example_sample_annotation,
  time_column = c('RunDate', 'RunTime'),
  new_time_column = 'DateTime',
  dateTimeFormat = c("%b_%d", "%H:%M:%S"))
```

---

date\_to\_sample\_order *Convert date/time to POSIXct and rank samples by it*

---

### Description

Converts date/time columns fo sample\_annotation to POSIXct format and calculates sample run rank in order column

### Usage

```
date_to_sample_order(sample_annotation, time_column = c("RunDate",
  "RunTime"), new_time_column = "DateTime",
  dateTimeFormat = c("%b_%d", "%H:%M:%S"),
  new_order_col = "order", instrument_col = "instrument")
```

### Arguments

sample\_annotation

data frame with:

1. sample\_id\_col (this can be repeated as row names)
2. biological covariates
3. technical covariates (batches etc)

. See help("example\_sample\_annotation")

time\_column name of the column(s) where run date & time are specified. These will be used to determine the run order

new\_time\_column

name of the new column to which date&time will be converted to

dateTimeFormat POSIX format of the date and time. See [as.POSIXct](#) from base R for details

new\_order\_col name of column with generated the order of sample run based on time columns

instrument\_col column, denoting different instrument used for measurements

### Value

sample annotation file with a new column new\_time\_column with POSIX-formatted date & new\_order\_col used in some diagnostic plots (e.g. [plot\\_iRT](#), [plot\\_sample\\_mean](#))

### Examples

```
sample_annotation_wOrder <- date_to_sample_order(
  example_sample_annotation,
  time_column = c('RunDate', 'RunTime'),
  new_time_column = 'new_DateTime',
  dateTimeFormat = c("%b_%d", "%H:%M:%S"),
  new_order_col = 'new_order',
  instrument_col = NULL)
```

---

define\_sample\_order     *Defining sample order internally*

---

## Description

Defining sample order internally

## Usage

```
define_sample_order(order_col, sample_annotation, facet_col, batch_col,
  df_long, sample_id_col, color_by_batch)
```

## Arguments

- |                   |   |
|-------------------|---|
| order_col         | column in sample_annotation that determines sample order. It is used for in initial assessment plots ( <a href="#">plot_sample_mean_or_boxplot</a> ) and feature-level diagnostics ( <a href="#">feature_level_diagnostics</a> ). Can be 'NULL' if sample order is irrelevant (e.g. in genomic experiments). For more details, order definition/inference, see <a href="#">define_sample_order</a> and <a href="#">date_to_sample_order</a> |
| sample_annotation | data frame with: <ol style="list-style-type: none"> <li>1. sample_id_col (this can be repeated as row names)</li> <li>2. biological covariates</li> <li>3. technical covariates (batches etc)</li> </ol> . See <code>help("example_sample_annotation")</code>   |
| facet_col         | column in sample_annotation with a batch factor to separate plots into facets; usually 2nd to batch_col. Most meaningful for multi-instrument MS experiments (where each instrument has its own order-associated effects (see order_col) or simultaneous examination of two batch factors (e.g. preparation day and measurement day). For single-instrument case should be set to 'NULL'  |
| batch_col         | column in sample_annotation that should be used for batch comparison (or other, non-batch factor to be mapped to color in plots).   |
| df_long           | data frame where each row is a single feature in a single sample. It minimally has a sample_id_col, a feature_id_col and a measure_col, but usually also an m_score (in OpenSWATH output result file). See <code>help("example_proteome")</code> for more details.  |
| sample_id_col     | name of the column in sample_annotation table, where the filenames (column names of the data_matrix are found).   |
| color_by_batch    | (logical) whether to color points and connecting lines by batch factor as defined by batch_col.   |

## Value

list of two items: order\_col new name and new df\_long

## See Also

[plot\\_sample\\_mean\\_or\\_boxplot](#), [feature\\_level\\_diagnostics](#)

**Examples**

```

sample_order = define_sample_order(order_col = 'order',
sample_annotation = example_sample_annotation,
facet_col = NULL, batch_col = 'MS_batch', df_long = example_proteome,
sample_id_col = 'FullRunName', color_by_batch = TRUE)
new_order_col = sample_order$order_col
df_long = sample_order$df_long

```

---

```

example_peptide_annotation
      Peptide annotation data

```

---

**Description**

This is data from Aging study annotated with gene names

**Usage**

```
example_peptide_annotation
```

**Format**

A data frame with 535 rows and 10 variables:

**peptide\_group\_label** peptide group label ID, identical to peptide\_group\_label in example\_proteome

**Gene** HUGO gene ID

**ProteinName** protein group name as specified in example\_proteome

---

```

example_proteome      Example protein data in long format

```

---

**Description**

This is OpenSWATH-output data from Aging study with all iRT, spike-in peptides, few representative peptides and proteins for signal improvement demonstration. Using matrix\_to\_long can be converted to example\_proteome\_matrix

**Usage**

```
example_proteome
```

**Format**

An object of class data.frame with 85278 rows and 5 columns.

**Source**

PRIDE ID will be added upon the publication of the dataset



---

`example_proteome_matrix`*Example protein data in matrix*

---

**Description**

This is measurement data from Aging study with columns representing samples and rows representing peptides. Generated by `long_to_matrix`

**Usage**`example_proteome_matrix`**Format**

A matrix with 535 rows and 233 columns:

**Source**

PRIDE ID will be added upon the publication of the dataset

---

`example_sample_annotation`*Sample annotation data version 1*

---

**Description**

This is data from BXD mouse population aging study with mock instruments to show how instrument-specific functionality works

**Usage**`example_sample_annotation`**Format**

A data frame with 233 rows and 11 variables:

**FullRunName** name of the file with the measurement for each sample, referred to as `sample_id_col`

**MS\_batch** mass-spectrometry batch: 4-level factor of manually annotated batches

**EarTag** mouse ID, i.e. ID of the biological object. Only 14 mice have been replicated, one mouse was profiled 7 times.

**Strain** mouse strain ID from BXD population set - biological covariate #1, 51 Strain represented

**Diet** diet, biological covariate #2 - either HFD = 'High Fat Diet' or CD = 'Chow Diet'

**Sex** mice sex - biological covariate #3

**RunDate** mass-spectrometry running date. In combination with `RunTime` used for running order determination. Vector of class "difftime" and "hms"

**RunTime** mass-spectrometry running time. In combination with RunDate used for running order determination. Vector of class "POSIXct" and "POSIXt"

**DateTime** numeric date and time generated by date\_to\_sample\_order

**order** order of samples generated by sorting DateTime in date\_to\_sample\_order

**digestion\_batch** peptide digestion batch: 4-level factor of manually annotated batches ...

feature\_level\_diagnostics

*Plotting peptide measurements*

## Description

Creates a peptide faceted ggplot2 plot of the value in measure\_col vs order\_col (if 'NULL', x-axis is simply a sample name order). Additionally, the resulting plot can also be colored either by batch factor, by quality factor (e.g. imputed/non-imputed) and, if needed, faceted by another batch factor, e.g. an instrument. If the non-linear curve was fit, this can also be added to the plot, see functions specific to each case below

## Usage

```
plot_single_feature(feature_name, df_long, sample_annotation = NULL,
  sample_id_col = "FullRunName", measure_col = "Intensity",
  feature_id_col = "peptide_group_label", geom = c("point", "line"),
  qual_col = NULL, qual_value = NULL, batch_col = "MS_batch",
  color_by_batch = FALSE, color_scheme = "brewer",
  order_col = "order", vline_color = "red", facet_col = NULL,
  filename = NULL, width = NA, height = NA, units = c("cm", "in",
  "mm"), plot_title = NULL, theme = "classic", ylimits = NULL)
```

```
plot_peptides_of_one_protein(protein_name, peptide_annotation = NULL,
  protein_col = "ProteinName", df_long, sample_annotation = NULL,
  sample_id_col = "FullRunName", measure_col = "Intensity",
  feature_id_col = "peptide_group_label", geom = c("point", "line"),
  qual_col = NULL, qual_value = NULL, batch_col = "MS_batch",
  color_by_batch = FALSE, color_scheme = "brewer",
  order_col = "order", vline_color = "red", facet_col = NULL,
  filename = NULL, width = NA, height = NA, units = c("cm", "in",
  "mm"), plot_title = sprintf("Peptides of %s protein", protein_name),
  theme = "classic")
```

```
plot_spike_in(spike_ins = "BOVIN", peptide_annotation = NULL,
  protein_col = "ProteinName", df_long, sample_annotation = NULL,
  sample_id_col = "FullRunName", measure_col = "Intensity",
  feature_id_col = "peptide_group_label", geom = c("point", "line"),
  qual_col = NULL, qual_value = NULL, batch_col = "MS_batch",
  color_by_batch = FALSE, color_scheme = "brewer",
  order_col = "order", vline_color = "red", facet_col = NULL,
  filename = NULL, width = NA, height = NA, units = c("cm", "in",
  "mm"), plot_title = sprintf("Spike-in %s plots", spike_ins),
  theme = "classic")
```

```

plot_iRT(irt_pattern = "iRT", peptide_annotation = NULL,
  protein_col = "ProteinName", df_long, sample_annotation = NULL,
  sample_id_col = "FullRunName", measure_col = "Intensity",
  feature_id_col = "peptide_group_label", geom = c("point", "line"),
  qual_col = NULL, qual_value = NULL, batch_col = "MS_batch",
  color_by_batch = FALSE, color_scheme = "brewer",
  order_col = "order", vline_color = "red", facet_col = NULL,
  filename = NULL, width = NA, height = NA, units = c("cm", "in",
  "mm"), plot_title = "iRT peptide profile", theme = "classic")

plot_with_fitting_curve(feature_name, fit_df, fit_value_col = "fit",
  df_long, sample_annotation = NULL, sample_id_col = "FullRunName",
  measure_col = "Intensity", feature_id_col = "peptide_group_label",
  geom = c("point", "line"), qual_col = NULL, qual_value = NULL,
  batch_col = "MS_batch", color_by_batch = FALSE,
  color_scheme = "brewer", order_col = "order", vline_color = "grey",
  facet_col = NULL, filename = NULL, width = NA, height = NA,
  units = c("cm", "in", "mm"),
  plot_title = sprintf("Fitting curve of %s \n
  paste(feature_name, collapse = " ")), theme = "classic")

```

peptide",

## Arguments

<code>feature_name</code>	name of the selected feature (e.g. peptide) for diagnostic profiling
<code>df_long</code>	data frame where each row is a single feature in a single sample. It minimally has a <code>sample_id_col</code> , a <code>feature_id_col</code> and a <code>measure_col</code> , but usually also an <code>m_score</code> (in OpenSWATH output result file). See <code>help("example_proteome")</code> for more details.
<code>sample_annotation</code>	data frame with: <ol style="list-style-type: none"> <li>1. <code>sample_id_col</code> (this can be repeated as row names)</li> <li>2. biological covariates</li> <li>3. technical covariates (batches etc)</li> </ol> . See <code>help("example_sample_annotation")</code>
<code>sample_id_col</code>	name of the column in <code>sample_annotation</code> table, where the filenames (column names of the <code>data_matrix</code> are found).
<code>measure_col</code>	if <code>df_long</code> is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency.
<code>feature_id_col</code>	name of the column with feature/gene/peptide/protein ID used in the long format representation <code>df_long</code> . In the wide formatted representation <code>data_matrix</code> this corresponds to the row names.
<code>geom</code>	whether to show the feature as points and/or connect by lines (accepted values are: 1. <code>point</code> , <code>line</code> and <code>c('point', 'line')</code> )
<code>qual_col</code>	column to color point by certain value denoted by <code>color_by_qual_value</code> . Design with inferred/requant values in OpenSWATH output data, which means argument value has to be set to <code>m_score</code> .
<code>qual_value</code>	value in <code>qual_col</code> to color. For OpenSWATH data, this argument value has to be set to 2 (this is an <code>m_score</code> value for imputed values (requant values)).

batch_col	column in sample_annotation that should be used for batch comparison (or other, non-batch factor to be mapped to color in plots).
color_by_batch	(logical) whether to color points and connecting lines by batch factor as defined by batch_col.
color_scheme	a named vector of colors to map to batch_col, names corresponding to the levels of the factor. For continuous variables, vector doesn't need to be named.
order_col	column in sample_annotation that determines sample order. It is used for in initial assessment plots ( <a href="#">plot_sample_mean_or_boxplot</a> ) and feature-level diagnostics ( <a href="#">feature_level_diagnostics</a> ). Can be 'NULL' if sample order is irrelevant (e.g. in genomic experiments). For more details, order definition/inference, see <a href="#">define_sample_order</a> and <a href="#">date_to_sample_order</a>
vline_color	color of vertical lines, typically separating different MS batches in ordered runs; should be 'NULL' for experiments without intrinsic order
facet_col	column in sample_annotation with a batch factor to separate plots into facets; usually 2nd to batch_col. Most meaningful for multi-instrument MS experiments (where each instrument has its own order-associated effects (see order_col) or simultaneous examination of two batch factors (e.g. preparation day and measurement day). For single-instrument case should be set to 'NULL'
filename	path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported
width	option determining the output image width
height	option determining the output image width
units	units: 'cm', 'in' or 'mm'
plot_title	title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))
theme	ggplot theme, by default classic. Can be easily overridden
ylimits	range of y-axis to plot feature-level trends
protein_name	name of the protein as defined in ProteinName
peptide_annotation	long format data frame with peptide ID and their corresponding protein and/or gene annotations. See <code>help("example_peptide_annotation")</code> .
protein_col	column where protein names are specified
spike_ins	name of feature(s), typically proteins that were spiked in for control
irt_pattern	substring used to identify iRT proteins in the column 'ProteinName'
fit_df	data frame output of <code>adjust_batch_trend_df</code> to be plotted with the line
fit_value_col	column in fit_df where the values for fitting trend are found

**Value**

ggplot2 type plot of measure\_col vs order\_col, faceted by feature\_name and (optionally) by batch\_col

**Examples**

```
single_feature_plot <- plot_single_feature(feature_name = "46213_NVGVSFYADKPEVTQEQK_2",
df_long = example_proteome, example_sample_annotation,
qual_col = NULL)

#color measurements by factor, related to order (MS_batch)
plot_single_feature(feature_name = "46213_NVGVSFYADKPEVTQEQK_2",
df_long = example_proteome, example_sample_annotation,
qual_col = NULL, color_by_batch = TRUE, batch_col = 'MS_batch')

#color measurements by factor, with order-unrelated factor
single_feature_plot <- plot_single_feature(feature_name = "46213_NVGVSFYADKPEVTQEQK_2",
df_long = example_proteome, example_sample_annotation,
qual_col = NULL, color_by_batch = TRUE, batch_col = 'Diet', geom = 'point',
vline_color = NULL)

#saving the plot
## Not run:
single_feature_plot <- plot_single_feature(feature_name = "46213_NVGVSFYADKPEVTQEQK_2",
df_long = example_proteome, example_sample_annotation,
qual_col = NULL, filename = 'test_peptide.png',
width = 28, height = 18, units = 'cm')

## End(Not run)

#to examine peptides of a single protein:
peptides_of_one_protein_plot <- plot_peptides_of_one_protein (
protein_name = "Haao", peptide_annotation = example_peptide_annotation,
protein_col = "Gene", df_long = example_proteome,
sample_annotation = example_sample_annotation,
order_col = 'order', sample_id_col = 'FullRunName',
batch_col = 'MS_batch')

#saving the peptides of one protein
## Not run:
peptides_of_one_protein_plot <- plot_peptides_of_one_protein (
protein_name = "Haao", peptide_annotation = example_peptide_annotation,
protein_col = "Gene", df_long = example_proteome,
sample_annotation = example_sample_annotation,
order_col = 'order', sample_id_col = 'FullRunName',
batch_col = 'MS_batch',
filename = 'test_protein.png', width = 14, height = 9, units = 'in')
## End(Not run)

#to illustrate spike-ins:
spike_in_plot <- plot_spike_in(spike_ins = "BOVINE_A1ag",
peptide_annotation = example_peptide_annotation, protein_col = 'Gene',
df_long = example_proteome, sample_annotation = example_sample_annotation,
sample_id_col = 'FullRunName',
plot_title = "Spike-in BOVINE protein peptides")

#to illustrate iRT peptides:
irt_plot <- plot_iRT(irt_pattern = "iRT",
peptide_annotation = example_peptide_annotation,
df_long = example_proteome, sample_annotation = example_sample_annotation,
protein_col = 'Gene')
```

```
#illustrate the fitting curve:
special_peptide = example_proteome$peptide_group_label == "10231_QDVDVWLWQQEGSSK_2"
loess_fit_70 <- adjust_batch_trend_df(example_proteome[special_peptide,],
example_sample_annotation, span = 0.7)

fitting_curve_plot <- plot_with_fitting_curve(feature_name = "10231_QDVDVWLWQQEGSSK_2",
df_long = example_proteome, sample_annotation = example_sample_annotation,
fit_df = loess_fit_70, plot_title = "Curve fitting with 70% span")

#with curves colored by the corresponding batch:
fitting_curve_plot <- plot_with_fitting_curve(feature_name = "10231_QDVDVWLWQQEGSSK_2",
df_long = example_proteome, sample_annotation = example_sample_annotation,
fit_df = loess_fit_70, plot_title = "Curve fitting with 70% span",
color_by_batch = TRUE, batch_col = 'MS_batch')
```

---

fit\_nonlinear

*Fit a non-linear trend (currently optimized for LOESS)*


---

## Description

Fit a non-linear trend (currently optimized for LOESS)

## Usage

```
fit_nonlinear(df_feature_batch, measure_col = "Intensity",
order_col = "order", feature_id = NULL, batch_id = NULL,
fit_func = "loess_regression", optimize_span = FALSE,
no_fit_imputed = TRUE, qual_col = "m_score", qual_value = 2,
min_measurements = 8, ...)
```

## Arguments

df_feature_batch	data frame containing response variable e.g. samples in order and explanatory variable e.g. measurement for a specific feature (peptide) in a specific batch
measure_col	if df_long is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency.
order_col	column in sample_annotation that determines sample order. It is used for in initial assessment plots ( <a href="#">plot_sample_mean_or_boxplot</a> ) and feature-level diagnostics ( <a href="#">feature_level_diagnostics</a> ). Can be 'NULL' if sample order is irrelevant (e.g. in genomic experiments). For more details, order definition/inference, see <a href="#">define_sample_order</a> and <a href="#">date_to_sample_order</a>
feature_id	the name of the feature, required for warnings
batch_id	the name of the batch, required for warnings
fit_func	function to use for the fit, e.g. loess_regression
optimize_span	logical, whether to specify span or optimize it (specific entirely for LOESS regression)
no_fit_imputed	(logical) whether to fit the imputed (requant) values

qual_col	column to color point by certain value denoted by color_by_qual_value. Design with inferred/requant values in OpenSWATH output data, which means argument value has to be set to m_score.
qual_value	value in qual_col to color. For OpenSWATH data, this argument value has to be set to 2 (this is an m_score value for imputed values (requant values)).
min_measurements	the absolute threshold to filter
...	additional parameters to be passed to the fitting function

**Value**

vector of fitted response values

**Examples**

```
test_peptide = example_proteome$peptide_group_label[1]
selected_peptide = example_proteome$peptide_group_label == test_peptide
df_selected = example_proteome[selected_peptide,]
selected_batch = example_sample_annotation$MS_batch == 'Batch_1'
batch_selected_df = example_sample_annotation[selected_batch,]
df_for_test = merge(df_selected, batch_selected_df, by = 'FullRunName')
fit_values = fit_nonlinear(df_for_test)

#for the case where are two many missing values, no curve is fit
selected_batch = example_sample_annotation$MS_batch == 'Batch_2'
batch_selected_df = example_sample_annotation[selected_batch,]
df_for_test = merge(df_selected, batch_selected_df, by = 'FullRunName')
fit_values = fit_nonlinear(df_for_test)
missing_values = df_for_test[['m_score']] == 2
all(fit_values[!is.na(fit_values)] == df_for_test[['Intensity']][!missing_values])
```

---

long\_to\_matrix

*Long to wide data format conversion*


---

**Description**

Convert from a long data frame representation to a wide matrix representation

**Usage**

```
long_to_matrix(df_long, feature_id_col = "peptide_group_label",
  measure_col = "Intensity", sample_id_col = "FullRunName",
  qual_col = NULL, qual_value = 2)
```

**Arguments**

df\_long data frame where each row is a single feature in a single sample. It minimally has a sample\_id\_col, a feature\_id\_col and a measure\_col, but usually also an m\_score (in OpenSWATH output result file). See help("example\_proteome") for more details.

feature_id_col	name of the column with feature/gene/peptide/protein ID used in the long format representation df_long. In the wide formatted representation data_matrix this corresponds to the row names.
measure_col	if df_long is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency.
sample_id_col	name of the column in sample_annotation table, where the filenames (colnames of the data_matrix are found).
qual_col	column to color point by certain value denoted by color_by_qual_value. Design with inferred/requant values in OpenSWATH output data, which means argument value has to be set to m_score.
qual_value	value in qual_col to color. For OpenSWATH data, this argument value has to be set to 2 (this is an m_score value for imputed values (requant values)).

**Value**

data\_matrix ([proBatch](#)) like matrix (features in rows, samples in columns)

**See Also**

Other matrix manipulation functions: [matrix\\_to\\_long](#)

**Examples**

```
proteome_matrix <- long_to_matrix(example_proteome)
```

---

matrix_to_long	<i>Wide to long conversion</i>
----------------	--------------------------------

---

**Description**

Convert from wide matrix to a long data frame representation

**Usage**

```
matrix_to_long(data_matrix, sample_annotation = NULL,
  feature_id_col = "peptide_group_label", measure_col = "Intensity",
  sample_id_col = "FullRunName", step = NULL)
```

**Arguments**

data\_matrix features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example\_proteome\_matrix" for more details (to call the description, use help("example\_proteome\_matrix"))

sample\_annotation data frame with:

1. sample\_id\_col (this can be repeated as row names)
2. biological covariates
3. technical covariates (batches etc)

. See help("example\_sample\_annotation")



feature_id_col	name of the column with feature/gene/peptide/protein ID used in the long format representation <code>df_long</code> . In the wide formatted representation <code>data_matrix</code> this corresponds to the row names.
measure_col	if <code>df_long</code> is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency.
sample_id_col	name of the column in <code>sample_annotation</code> table, where the filenames (column names of the <code>data_matrix</code> are found).
step	normalization step (e.g. Raw or Normalized. Useful if consecutive steps are compared in plots. Note that in plots these are usually ordered alphabetically, so it's worth naming with numbers, e.g. 1_raw, 2_quantile

**Value**

`df_long` ([proBatch](#)) like data frame

**See Also**

Other matrix manipulation functions: [long\\_to\\_matrix](#)

**Examples**

```
proteome_long <- matrix_to_long(example_proteome_matrix,
example_sample_annotation)
```

---

normalize

*Data normalization methods*

---

**Description**

Normalization of raw (usually log-transformed) data. Normalization brings the samples to the same scale. Currently the following normalization functions are implemented: #

1. Quantile normalization: `'quantile_normalize_dm()'`. Quantile normalization of the data.
2. Median normalization: `'normalize_sample_medians_dm()'`. Normalization by centering sample medians to global median of the data

Alternatively, one can call normalization function with `'normalize_data_dm()'` wrapper.

**Usage**

```
quantile_normalize_dm(data_matrix)
```

```
quantile_normalize_df(df_long, feature_id_col = "peptide_group_label",
sample_id_col = "FullRunName", measure_col = "Intensity",
no_fit_imputed = TRUE, qual_col = NULL, qual_value = 2,
keep_all = "default")
```

```
normalize_sample_medians_dm(data_matrix)
```

```
normalize_sample_medians_df(df_long,
```

```

feature_id_col = "peptide_group_label",
sample_id_col = "FullRunName", measure_col = "Intensity",
no_fit_imputed = FALSE, qual_col = NULL, qual_value = 2,
keep_all = "default")

normalize_data_dm(data_matrix, normalize_func = c("quantile",
"medianCentering"), log_base = NULL, offset = 1)

normalize_data_df(df_long, normalize_func = c("quantile",
"medianCentering"), log_base = NULL, offset = 1,
feature_id_col = "peptide_group_label",
sample_id_col = "FullRunName", measure_col = "Intensity",
no_fit_imputed = TRUE, qual_col = NULL, qual_value = 2,
keep_all = FALSE)

```

### Arguments

<code>data_matrix</code>	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example_proteome_matrix" for more details (to call the description, use <code>help("example_proteome_matrix")</code> )
<code>df_long</code>	data frame where each row is a single feature in a single sample. It minimally has a <code>sample_id_col</code> , a <code>feature_id_col</code> and a <code>measure_col</code> , but usually also an <code>m_score</code> (in OpenSWATH output result file). See <code>help("example_proteome")</code> for more details.
<code>feature_id_col</code>	name of the column with feature/gene/peptide/protein ID used in the long format representation <code>df_long</code> . In the wide formatted representation <code>data_matrix</code> this corresponds to the row names.
<code>sample_id_col</code>	name of the column in <code>sample_annotation</code> table, where the filenames (colnames of the <code>data_matrix</code> are found).
<code>measure_col</code>	if <code>df_long</code> is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency.
<code>no_fit_imputed</code>	(logical) whether to use imputed (requant) values, as flagged in <code>qual_col</code> by <code>qual_value</code> for data transformation
<code>qual_col</code>	column to color point by certain value denoted by <code>color_by_qual_value</code> . Design with inferred/requant values in OpenSWATH output data, which means argument value has to be set to <code>m_score</code> .
<code>qual_value</code>	value in <code>qual_col</code> to color. For OpenSWATH data, this argument value has to be set to 2 (this is an <code>m_score</code> value for imputed values (requant values)).
<code>keep_all</code>	when transforming the data (normalize, correct) - acceptable values: all/default/minimal (which set of columns be kept).
<code>normalize_func</code>	global batch normalization method ('quantile' or 'MedianCentering')
<code>log_base</code>	whether to log transform data matrix before normalization (e.g. 'NULL', '2' or '10')
<code>offset</code>	small positive number to prevent 0 conversion to $-\text{Inf}$

### Value

the data in the same format as input (`data_matrix` or `df_long`). For `df_long` the data frame stores the original values of `measure_col` in another column called "preNorm\_intensity" if "intensity", and the normalized values in `measure_col` column.

**Examples**

```
#Quantile normalization:
quantile_normalized_matrix <- quantile_normalize_dm(example_proteome_matrix)

#Median centering:
median_normalized_df <- normalize_sample_medians_df(example_proteome)

#Transform the data in one go:
quantile_normalized_matrix <- normalize_data_dm(example_proteome_matrix,
normalize_func = "quantile", log_base = 2, offset = 1)
```

---

plot_corr_matrix	<i>Visualise correlation matrix</i>
------------------	-------------------------------------

---

**Description**

recommended for heatmap-type visualisation of correlation matrix with <100 items. With >50 samples and ~10 replicate pairs distribution plots may be more informative.

**Usage**

```
plot_corr_matrix(corr_matrix, annotation = NULL,
  annotation_id_col = "FullRunName", factors_to_plot = NULL,
  cluster_rows = FALSE, cluster_cols = FALSE,
  heatmap_color = colorRampPalette(rev(brewer.pal(n = 7, name =
  "RdYlBu")))(100), color_list = NULL, filename = NULL, width = 7,
  height = 7, units = c("cm", "in", "mm"), plot_title = NULL, ...)
```

**Arguments**

corr_matrix	square correlation matrix
annotation	data frame with peptide_annotation for protein correlation heatmap or sample_annotation for sample correlation heatmap
annotation_id_col	feature_id_col for protein correlation heatmap or sample_id_col for sample correlation heatmap
factors_to_plot	vector of technical and biological covariates to be plotted in this diagnostic plot (assumed to be present in sample_annotation)
cluster_rows	boolean values determining if rows should be clustered or hclust object
cluster_cols	boolean values determining if columns should be clustered or hclust object
heatmap_color	vector of colors used in heatmap.
color_list	list, as returned by sample_annotation_to_colors, where each item contains a color vector for each factor to be mapped to the color.
filename	path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported

width	option determining the output image width
height	option determining the output image width
units	units: 'cm', 'in' or 'mm'
plot_title	title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))
...	parameters for the <a href="#">pheatmap</a> visualisation, for details see examples and help to corresponding functions

### Details

Plot correlation of selected samples or peptides

### Value

pheatmap object

### See Also

[pheatmap](#), [plot\\_sample\\_corr\\_distribution](#), [plot\\_peptide\\_corr\\_distribution](#)

### Examples

```
peptides <- c("10231_QDQDVWLWQQEGSSK_2", "10768_RLESELDGLR_2")
data_matrix_sub = example_proteome_matrix[peptides,]
corr_matrix = cor(t(data_matrix_sub), use = 'complete.obs')
corr_matrix_plot <- plot_corr_matrix(corr_matrix)
```

---

plot\_CV\_distr                      *Plot CV distribution to compare various steps of the analysis*

---

### Description

Plot CV distribution to compare various steps of the analysis

### Usage

```
plot_CV_distr(df_long, sample_annotation = NULL,
              feature_id_col = "peptide_group_label",
              sample_id_col = "FullRunName", measure_col = "Intensity",
              biospecimen_id_col = "EarTag", batch_col = NULL, unlog = TRUE,
              log_base = 2, offset = 1, plot_title = NULL, filename = NULL,
              theme = "classic")
```

**Arguments**

df_long	as in df_long for the rest of the package, but, when it has entries for intensity, represented in measure_col for several steps, e.g. raw, normalized, batch corrected data, as seen in column Step, then multi-step CV comparison can be carried out.
sample_annotation	data frame with: <ol style="list-style-type: none"> <li>1. sample_id_col (this can be repeated as row names)</li> <li>2. biological covariates</li> <li>3. technical covariates (batches etc)</li> </ol> . See help("example_sample_annotation")
feature_id_col	name of the column with feature/gene/peptide/protein ID used in the long format representation df_long. In the wide formatted representation data_matrix this corresponds to the row names.
sample_id_col	name of the column in sample_annotation table, where the filenames (column names of the data_matrix are found).
measure_col	if df_long is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency.
biospecimen_id_col	column in sample_annotation that defines a unique bio ID, which is usually a combination of conditions or groups. Tip: if such ID is absent, but can be defined from several columns, create new biospecimen_id column
batch_col	column in sample_annotation that should be used for batch comparison (or other, non-batch factor to be mapped to color in plots).
unlog	(logical) whether to reverse log transformation of the original data
log_base	base of the logarithm for transformation
offset	small positive number to prevent 0 conversion to -Inf
plot_title	title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))
filename	path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported
theme	ggplot theme, by default classic. Can be easily overridden

**Value**

ggplot object with the boxplot of CVs on one or several steps

**Examples**

```
CV_plot = plot_CV_distr(example_proteome,
  sample_annotation = example_sample_annotation,
  measure_col = 'Intensity', batch_col = 'MS_batch',
  plot_title = NULL, filename = NULL, theme = 'classic')
```

---

plot_CV_distr.df	<i>Plot the distribution (boxplots) of per-batch per-step CV of features</i>
------------------	--

---

**Description**

Plot the distribution (boxplots) of per-batch per-step CV of features

**Usage**

```
plot_CV_distr.df(CV_df, plot_title = NULL, filename = NULL,
  theme = "classic", log_y_scale = TRUE)
```

**Arguments**

CV_df	data frame with Total CV for each feature & (optionally) per-batch CV
plot_title	title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))
filename	path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported
theme	ggplot theme, by default classic. Can be easily overridden
log_y_scale	(logical) whether to display the CV on log-scale

**Value**

ggplot object

---

plot_heatmap_diagnostic	<i>Plot the heatmap of samples (cols) vs features (rows)</i>
-------------------------	--

---

**Description**

Plot the heatmap of samples (cols) vs features (rows)

**Usage**

```
plot_heatmap_diagnostic(data_matrix, sample_annotation = NULL,
  sample_id_col = "FullRunName", factors_to_plot = NULL,
  fill_the_missing = -1, color_for_missing = "black",
  heatmap_color = colorRampPalette(rev(brewer.pal(n = 7, name =
  "RdYlBu")))(100), cluster_rows = TRUE, cluster_cols = FALSE,
  color_list = NULL, peptide_annotation = NULL,
  feature_id_col = "peptide_group_label",
  factors_of_feature_ann = c("KEGG_pathway", "evolutionary_distance"),
  color_list_features = NULL, filename = NULL, width = 7,
  height = 7, units = c("cm", "in", "mm"), plot_title = NULL, ...)
```

**Arguments**

<code>data_matrix</code>	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example_proteome_matrix" for more details (to call the description, use <code>help("example_proteome_matrix")</code> )
<code>sample_annotation</code>	data frame with: <ol style="list-style-type: none"> <li>1. <code>sample_id_col</code> (this can be repeated as row names)</li> <li>2. biological covariates</li> <li>3. technical covariates (batches etc)</li> </ol> . See <code>help("example_sample_annotation")</code>
<code>sample_id_col</code>	name of the column in <code>sample_annotation</code> table, where the filenames (colnames of the <code>data_matrix</code> are found).
<code>factors_to_plot</code>	vector of technical and biological factors to be plotted in this diagnostic plot (assumed to be present in <code>sample_annotation</code> )
<code>fill_the_missing</code>	numeric value that the missing values are substituted with, or NULL if features with missing values are to be excluded.
<code>color_for_missing</code>	special color to make missing values. Usually black or white, depending on <code>heatmap_color</code>
<code>heatmap_color</code>	vector of colors used in heatmap (typicall a gradient)
<code>cluster_rows</code>	boolean value determining if rows should be clustered
<code>cluster_cols</code>	boolean value determining if columns should be clustered
<code>color_list</code>	list, as returned by <code>sample_annotation_to_colors</code> , where each item contains a color vector for each factor to be mapped to the color.
<code>peptide_annotation</code>	long format data frame with peptide ID and their corresponding protein and/or gene annotations. See <code>help("example_peptide_annotation")</code> .
<code>feature_id_col</code>	name of the column with feature/gene/peptide/protein ID used in the long format representation <code>df_long</code> . In the wide formatted representation <code>data_matrix</code> this corresponds to the row names.
<code>factors_of_feature_ann</code>	vector of factors that characterize features, as listed in <code>peptide_annotation</code>
<code>color_list_features</code>	list, as returned by <code>sample_annotation_to_colors</code> , but mapping <code>peptide_annotation</code> where each item contains a color vector for each factor to be mapped to the color.
<code>filename</code>	path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported
<code>width</code>	option determining the output image width
<code>height</code>	option determining the output image width
<code>units</code>	units: 'cm', 'in' or 'mm'
<code>plot_title</code>	title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))
<code>...</code>	other parameters of <code>link[pheatmap]{pheatmap}</code>

**Value**

object returned by `link[pheatmap]{pheatmap}`

**See Also**

[sample\\_annotation\\_to\\_colors](#), [pheatmap](#)

**Examples**

```
log_transformed_matrix = log_transform_dm(example_proteome_matrix)
heatmap_plot <- plot_heatmap_diagnostic(log_transformed_matrix,
example_sample_annotation,
factors_to_plot = c("MS_batch", "digestion_batch", "Diet", 'DateTime'),
cluster_cols = TRUE, cluster_rows = FALSE,
show_rownames = FALSE, show_colnames = FALSE)
```

```
color_list <- sample_annotation_to_colors (example_sample_annotation,
factor_columns = c('MS_batch', 'EarTag', "Strain",
"Diet", "digestion_batch", "Sex"),
numeric_columns = c('DateTime', 'order'))
```

```
log_transformed_matrix = log_transform_dm(example_proteome_matrix)
heatmap_plot <- plot_heatmap_diagnostic(log_transformed_matrix,
example_sample_annotation,
factors_to_plot = c("MS_batch", "digestion_batch", "Diet", 'DateTime'),
cluster_cols = TRUE, cluster_rows = FALSE,
color_list = color_list,
show_rownames = FALSE, show_colnames = FALSE)
```

---

`plot_heatmap_generic` *Plot the heatmap*

---

**Description**

Plot the heatmap

**Usage**

```
plot_heatmap_generic(data_matrix, column_annotation_df = NULL,
row_annotation_df = NULL, col_ann_id_col = "FullRunName",
row_ann_id_col = "peptide_group_label",
columns_for_cols = c("MS_batch", "Diet", "DateTime", "order"),
columns_for_rows = c("KEGG_pathway", "WGCNA_module",
"evolutionary_distance"), cluster_rows = FALSE, cluster_cols = TRUE,
annotation_color_cols = NULL, annotation_color_rows = NULL,
fill_the_missing = -1, color_for_missing = "black",
heatmap_color = colorRampPalette(rev(brewer.pal(n = 7, name =
"RdYlBu")))(100), filename = NULL, width = 7, height = 7,
units = c("cm", "in", "mm"), plot_title = NULL, ...)
```



**Arguments**

<code>data_matrix</code>	the matrix of data to be plotted
<code>column_annotation_df</code>	data frame annotating columns of <code>data_matrix</code>
<code>row_annotation_df</code>	data frame annotating rows of <code>data_matrix</code>
<code>col_ann_id_col</code>	column of <code>column_annotation_df</code> whose values are unique identifiers of columns in <code>data_matrix</code>
<code>row_ann_id_col</code>	column of <code>row_annotation_df</code> whose values are unique identifiers of rows in <code>data_matrix</code>
<code>columns_for_cols</code>	vector of factors (columns) of <code>column_annotation_df</code> that will be mapped to color annotation of heatmap columns
<code>columns_for_rows</code>	vector of factors (columns) of <code>row_annotation_df</code> that will be mapped to color annotation of heatmap rows
<code>cluster_rows</code>	boolean: whether the rows should be clustered
<code>cluster_cols</code>	boolean: whether the rows should be clustered
<code>annotation_color_cols</code>	list of color vectors for column annotation, for each factor to be plotted; for factor-like variables a named vector (names should correspond to the levels of factors). Advisable to supply here color list returned by <code>sample_annotation_to_colors</code>
<code>annotation_color_rows</code>	list of color vectors for row annotation, for each factor to be plotted; for factor-like variables a named vector (names should correspond to the levels of factors). Advisable to supply here color list returned by <code>sample_annotation_to_colors</code>
<code>fill_the_missing</code>	numeric value that the missing values are substituted with, or NULL if features with missing values are to be excluded.
<code>color_for_missing</code>	special color to make missing values. Usually black or white, depending on <code>heatmap_color</code>
<code>heatmap_color</code>	vector of colors used in heatmap (typicall a gradient)
<code>filename</code>	path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported
<code>width</code>	option determining the output image width
<code>height</code>	option determining the output image width
<code>units</code>	units: 'cm', 'in' or 'mm'
<code>plot_title</code>	title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))
...	other parameters of <code>link[pheatmap]{pheatmap}</code>

**Value**

pheatmap-type object

**Examples**

```
p <- plot_heatmap_generic(log_transform_dm(example_proteome_matrix),
  column_annotation_df = example_sample_annotation,
  columns_for_cols = c("MS_batch", "digestion_batch", "Diet", 'DateTime'),
  plot_title = 'test_heatmap',
  show_rownames = FALSE, show_colnames = FALSE)
```

---

plot\_hierarchical\_clustering

*cluster the data matrix to visually inspect which confounder dominates*

---

**Description**

cluster the data matrix to visually inspect which confounder dominates

**Usage**

```
plot_hierarchical_clustering(data_matrix, sample_annotation,
  sample_id_col = "FullRunName", color_list = NULL,
  factors_to_plot = NULL, fill_the_missing = 0,
  distance = "euclidean", agglomeration = "complete",
  label_samples = TRUE, label_font = 0.2, filename = NULL,
  width = 38, height = 25, units = c("cm", "in", "mm"),
  plot_title = NULL, ...)
```

**Arguments**

data_matrix	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example_proteome_matrix" for more details (to call the description, use help("example_proteome_matrix"))
sample_annotation	data frame with: <ol style="list-style-type: none"> <li>1. sample_id_col (this can be repeated as row names)</li> <li>2. biological covariates</li> <li>3. technical covariates (batches etc)</li> </ol> . See help("example_sample_annotation")
sample_id_col	name of the column in sample_annotation table, where the filenames (colnames of the data_matrix are found).
color_list	list, as returned by sample_annotation_to_colors, where each item contains a color vector for each factor to be mapped to the color.
factors_to_plot	vector of technical and biological covariates to be plotted in this diagnostic plot (assumed to be present in sample_annotation)
fill_the_missing	numeric value determining how missing values should be substituted. If NULL, features with missing values are excluded.
distance	distance metric used for clustering

agglomeration	agglomeration methods as used by hclust
label_samples	if TRUE sample IDs (column names of data_matrix) will be printed
label_font	size of the font. Is active if label_samples is TRUE, ignored otherwise
filename	path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported
width	option determining the output image width
height	option determining the output image width
units	units: 'cm', 'in' or 'mm'
plot_title	title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))
...	other parameters of plotDendroAndColors from WGCNA package

**Value**

No return

**See Also**

[hclust](#), [sample\\_annotation\\_to\\_colors](#), [plotDendroAndColors](#)

**Examples**

```

selected_batches = example_sample_annotation$MS_batch %in%
                    c('Batch_1', 'Batch_2')
selected_samples = example_sample_annotation$FullRunName[selected_batches]
test_matrix = example_proteome_matrix[,selected_samples]

hierarchical_clustering_plot <- plot_hierarchical_clustering(
  example_proteome_matrix, example_sample_annotation,
  factors_to_plot = c('MS_batch', 'Diet', 'DateTime'),
  color_list = NULL,
  distance = "euclidean", agglomeration = 'complete',
  label_samples = FALSE)

#with defined color scheme:
color_list <- sample_annotation_to_colors (example_sample_annotation,
  factor_columns = c('MS_batch', "Strain", "Diet", "digestion_batch"),
  numeric_columns = c('DateTime', 'order'))
hierarchical_clustering_plot <- plot_hierarchical_clustering(
  example_proteome_matrix, example_sample_annotation,
  factors_to_plot = c('MS_batch', "Strain", 'DateTime', "digestion_batch"),
  color_list = color_list,
  distance = "euclidean", agglomeration = 'complete',
  label_samples = FALSE)

```

plot\_PCA

*plot PCA plot***Description**

plot PCA plot

**Usage**

```
plot_PCA(data_matrix, sample_annotation,
         feature_id_col = "peptide_group_label",
         sample_id_col = "FullRunName", color_by = "MS_batch",
         PC_to_plot = c(1, 2), fill_the_missing = -1,
         color_scheme = "brewer", filename = NULL, width = NA,
         height = NA, units = c("cm", "in", "mm"), plot_title = NULL,
         theme = "classic")
```

**Arguments**

**data\_matrix** features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example\_proteome\_matrix" for more details (to call the description, use help("example\_proteome\_matrix"))

**sample\_annotation** data frame with:

1. sample\_id\_col (this can be repeated as row names)
2. biological covariates
3. technical covariates (batches etc)

. See help("example\_sample\_annotation")

**feature\_id\_col** name of the column with feature/gene/peptide/protein ID used in the long format representation df\_long. In the wide formatted representation data\_matrix this corresponds to the row names.

**sample\_id\_col** name of the column in sample\_annotation table, where the filenames (colnames of the data\_matrix are found).

**color\_by** column name (as in sample\_annotation) to color by

**PC\_to\_plot** principal component numbers for x and y axis

**fill\_the\_missing** numeric value determining how missing values should be substituted. If NULL, features with missing values are excluded. If NULL, features with missing values are excluded.

**color\_scheme** a named vector of colors to map to batch\_col, names corresponding to the levels of the factor. For continuous variables, vector doesn't need to be named.

**filename** path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported

**width** option determining the output image width

**height** option determining the output image width

**units** units: 'cm', 'in' or 'mm'

plot_title	title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))
theme	ggplot theme, by default classic. Can be easily overridden

**Value**

ggplot scatterplot colored by factor levels of column specified in factor\_to\_color

**See Also**

[autoplot.pca\\_common](#), [ggplot](#)

**Examples**

```
pca_plot <- plot_PCA(example_proteome_matrix, example_sample_annotation,
  color_by = 'MS_batch', plot_title = "PCA colored by MS batch")
pca_plot <- plot_PCA(example_proteome_matrix, example_sample_annotation,
  color_by = 'DateTime', plot_title = "PCA colored by DateTime")

color_list <- sample_annotation_to_colors (example_sample_annotation,
  factor_columns = c('MS_batch', 'digestion_batch'),
  numeric_columns = c('DateTime','order'))
pca_plot <- plot_PCA(example_proteome_matrix, example_sample_annotation,
  color_by = 'DateTime', color_scheme = color_list[['DateTime']])

## Not run:
pca_plot <- plot_PCA(example_proteome_matrix, example_sample_annotation,
  color_by = 'DateTime', plot_title = "PCA colored by DateTime",
  filename = 'test_PCA.png', width = 14, height = 9, units = 'cm')

## End(Not run)
```

---

plot\_peptide\_corr\_distribution

*Create violin plot of peptide correlation distribution*

---

**Description**

Plot distribution of peptide correlations within one protein and between proteins

**Usage**

```
plot_peptide_corr_distribution(data_matrix, peptide_annotation,
  protein_col = "ProteinName", feature_id_col = "peptide_group_label",
  filename = NULL, width = NA, height = NA, units = c("cm", "in",
  "mm"), plot_title = "Distribution of peptide correlation",
  theme = "classic")

plot_peptide_corr_distribution.corrDF(corr_distribution, filename = NULL,
  width = NA, height = NA, units = c("cm", "in", "mm"),
  plot_title = "Correlation of peptides", theme = "classic")
```

**Arguments**

<code>data_matrix</code>	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example_proteome_matrix" for more details (to call the description, use <code>help("example_proteome_matrix")</code> )
<code>peptide_annotation</code>	long format data frame with peptide ID and their corresponding protein and/or gene annotations. See <code>help("example_peptide_annotation")</code> .
<code>protein_col</code>	column where protein names are specified
<code>feature_id_col</code>	name of the column with feature/gene/peptide/protein ID used in the long format representation <code>df_long</code> . In the wide formatted representation <code>data_matrix</code> this corresponds to the row names.
<code>filename</code>	path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported
<code>width</code>	option determining the output image width
<code>height</code>	option determining the output image width
<code>units</code>	units: 'cm', 'in' or 'mm'
<code>plot_title</code>	title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))
<code>theme</code>	ggplot theme, by default <code>classic</code> . Can be easily overridden
<code>corr_distribution</code>	data frame with peptide correlation distribution

**Value**

ggplot object (violin plot of peptide correlation)

**See Also**

[calculate\\_peptide\\_corr\\_distr](#), [ggplot](#)

**Examples**

```
peptide_corr_distribution <- plot_peptide_corr_distribution(
  example_proteome_matrix,
  example_peptide_annotation, protein_col = 'Gene')

selected_genes = c('BOVINE_A1ag', 'BOVINE_FetuinB', 'Cyfip1')
gene_filter = example_peptide_annotation$Gene %in% selected_genes
peptides_ann = example_peptide_annotation$peptide_group_label
selected_peptides = peptides_ann[gene_filter]
matrix_test = example_proteome_matrix[selected_peptides,]
pep_annotation_sel = example_peptide_annotation[gene_filter, ]
corr_distribution = calculate_peptide_corr_distr(matrix_test,
  pep_annotation_sel, protein_col = 'Gene')
peptide_corr_distribution <- plot_peptide_corr_distribution.corrDF(corr_distribution)

## Not run:
peptide_corr_distribution <- plot_peptide_corr_distribution.corrDF(corr_distribution,
  filename = 'test_peptide.png',
  width = 28, height = 28, units = 'cm')
```

```
## End(Not run)
```

---

plot\_protein\_corrplot *Peptide correlation matrix (heatmap)*

---

## Description

Plots correlation plot of peptides from a single protein

## Usage

```
plot_protein_corrplot(data_matrix, protein_name,
  peptide_annotation = NULL, protein_col = "ProteinName",
  feature_id_col = "peptide_group_label",
  factors_to_plot = c("ProteinName"), cluster_rows = FALSE,
  cluster_cols = FALSE,
  heatmap_color = colorRampPalette(rev(brewer.pal(n = 7, name =
    "RdYlBu")))(100), color_list = NULL, filename = NULL, width = NA,
  height = NA, units = c("cm", "in", "mm"),
  plot_title = sprintf("Peptide correlation matrix of %s protein",
    protein_name), ...)
```

## Arguments

data_matrix	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example_proteome_matrix" for more details (to call the description, use help("example_proteome_matrix"))
protein_name	the name of the protein
peptide_annotation	long format data frame with peptide ID and their corresponding protein and/or gene annotations. See help("example_peptide_annotation").
protein_col	column where protein names are specified
feature_id_col	name of the column with feature/gene/peptide/protein ID used in the long format representation df_long. In the wide formatted representation data_matrix this corresponds to the row names.
factors_to_plot	vector of technical and biological covariates to be plotted in this diagnostic plot (assumed to be present in sample_annotation)
cluster_rows	boolean values determining if rows should be clustered or hclust object
cluster_cols	boolean values determining if columns should be clustered or hclust object
heatmap_color	vector of colors used in heatmap.
color_list	list, as returned by sample_annotation_to_colors, where each item contains a color vector for each factor to be mapped to the color.
filename	path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported

width	option determining the output image width
height	option determining the output image width
units	units: 'cm', 'in' or 'mm'
plot_title	title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))
...	parameters for the corrplot visualisation

**Value**

heatmap object

**Examples**

```
protein_corrplot_plot <- plot_protein_corrplot(example_proteome_matrix,
protein_name = 'Haa0', peptide_annotation = example_peptide_annotation,
protein_col = 'Gene')
```

```
protein_corrplot_plot <- plot_protein_corrplot(example_proteome_matrix,
protein_name = c('Haa0', 'Dhtkd1'),
peptide_annotation = example_peptide_annotation,
protein_col = 'Gene', factors_to_plot = 'Gene')
```

---

plot\_PVCA

*Plot variance distribution by variable*


---

**Description**

Plot variance distribution by variable

**Usage**

```
plot_PVCA(data_matrix, sample_annotation,
feature_id_col = "peptide_group_label",
sample_id_col = "FullRunName", technical_factors = c("MS_batch",
"instrument"), biological_factors = c("cell_line", "drug_dose"),
fill_the_missing = -1, pca_threshold = 0.6,
variance_threshold = 0.01, colors_for_bars = NULL, filename = NULL,
width = NA, height = NA, units = c("cm", "in", "mm"),
plot_title = NULL, theme = "classic")
```

**Arguments**

**data\_matrix** features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example\_proteome\_matrix" for more details (to call the description, use help("example\_proteome\_matrix"))

**sample\_annotation**

data frame with:

1. sample\_id\_col (this can be repeated as row names)
2. biological covariates



3. technical covariates (batches etc)  
 . See `help("example_sample_annotation")`

`feature_id_col` name of the column with feature/gene/peptide/protein ID used in the long format representation `df_long`. In the wide formatted representation `data_matrix` this corresponds to the row names.

`sample_id_col` name of the column in `sample_annotation` table, where the filenames (column names of the `data_matrix` are found).

`technical_factors`  
 vector `sample_annotation` column names that are technical covariates

`biological_factors`  
 vector `sample_annotation` column names, that are biologically meaningful covariates

`fill_the_missing`  
 numeric value determining how missing values should be substituted. If NULL, features with missing values are excluded. If NULL, features with missing values are excluded.

`pca_threshold` the percentile value of the minimum amount of the variabilities that the selected principal components need to explain

`variance_threshold`  
 the percentile value of weight each of the covariates needs to explain (the rest will be lumped together)

`colors_forBars`  
 four-item color vector, specifying colors for the following categories: `c('residual', 'biological', 'biol:techn', 'technical')`

`filename` path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported

`width` option determining the output image width

`height` option determining the output image width

`units` units: 'cm', 'in' or 'mm'

`plot_title` title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))

`theme` ggplot theme, by default `classic`. Can be easily overridden

**Value**

ggplot object with the plot

**See Also**

[sample\\_annotation\\_to\\_colors](#), [ggplot](#)

**Examples**

```
matrix_test <- example_proteome_matrix[1:150, ]
pvca_plot <- plot_PVCA(matrix_test, example_sample_annotation,
  technical_factors = c('MS_batch', 'digestion_batch'),
  biological_factors = c("Diet", "Sex", "Strain"))
```

## Not run:

```
pvca_plot <- plot_PVCA(matrix_test, example_sample_annotation,
  technical_factors = c('MS_batch', 'digestion_batch'),
  biological_factors = c("Diet", "Sex", "Strain"),
  filename = 'test_PVCA.png', width = 28, height = 22, units = 'cm')

## End(Not run)
```

---

plot\_PVCA.df

*plot PVCA, when the analysis is completed*

---

## Description

plot PVCA, when the analysis is completed

## Usage

```
plot_PVCA.df(pvca_res, colors_forBars = NULL, filename = NULL,
  width = NA, height = NA, units = c("cm", "in", "mm"),
  plot_title = NULL, theme = "classic")
```

## Arguments

pvca_res	data frame of weights of Principal Variance Components, result of calculate_PVCA
colors_forBars	four-item color vector, specifying colors for the following categories: c('residual', 'biological', 'biol:techn', 'technical')
filename	path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported
width	option determining the output image width
height	option determining the output image width
units	units: 'cm', 'in' or 'mm'
plot_title	title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))
theme	ggplot theme, by default classic. Can be easily overridden

## Value

ggplot object with bars as weights, colored by bio/tech factors

## Examples

```
matrix_test <- example_proteome_matrix[1:150, ]
pvca_df_res <- prepare_PVCA_df(matrix_test, example_sample_annotation,
  technical_factors = c('MS_batch', 'digestion_batch'),
  biological_factors = c("Diet", "Sex", "Strain"),
  pca_threshold = .6, variance_threshold = .01, fill_the_missing = -1)
colors_forBars = c('grey', 'green', 'blue', 'red')
names(colors_forBars) = c('residual', 'biological', 'biol:techn', 'technical')

pvca_plot <- plot_PVCA.df(pvca_df_res, colors_forBars)
```

---

 plot\_sample\_corr\_distribution

*Create violin plot of sample correlation distribution*


---

## Description

Useful to visualize within batch vs within replicate vs non-related sample correlation

## Usage

```
plot_sample_corr_distribution(data_matrix, sample_annotation,
  repeated_samples = NULL, sample_id_col = "FullRunName",
  batch_col = "MS_batch", biospecimen_id_col = "EarTag",
  filename = NULL, width = NA, height = NA, units = c("cm", "in",
  "mm"), plot_title = "Sample correlation distribution",
  plot_param = "batch_replicate", theme = "classic")
```

```
plot_sample_corr_distribution.corrDF(corr_distribution, filename = NULL,
  width = NA, height = NA, units = c("cm", "in", "mm"),
  plot_title = "Sample correlation distribution",
  plot_param = "batch_replicate", theme = "classic")
```

## Arguments

data_matrix	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example_proteome_matrix" for more details (to call the description, use help("example_proteome_matrix"))
sample_annotation	data frame with: <ol style="list-style-type: none"> <li>1. sample_id_col (this can be repeated as row names)</li> <li>2. biological covariates</li> <li>3. technical covariates (batches etc)</li> </ol> . See help("example_sample_annotation")
repeated_samples	if NULL, correlation of all samples is plotted
sample_id_col	name of the column in sample_annotation table, where the filenames (colnames of the data_matrix are found).
batch_col	column in sample_annotation that should be used for batch comparison (or other, non-batch factor to be mapped to color in plots).
biospecimen_id_col	column in sample_annotation that captures the biological sample, that (possibly) was profiled several times as technical replicates. Tip: if such ID is absent, but can be defined from several columns, create new biospecimen_id column
filename	path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported
width	option determining the output image width
height	option determining the output image width

units	units: 'cm', 'in' or 'mm'
plot_title	title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))
plot_param	columns, defined in correlation_df, which is output of calculate_sample_corr_distr, specifically, <ol style="list-style-type: none"> <li>1. replicate</li> <li>2. batch_the_same</li> <li>3. batch_replicate</li> <li>4. batches</li> </ol>
theme	ggplot theme, by default classic. Can be easily overridden
corr_distribution	data frame with correlation distribution, as returned by calculate_sample_corr_distr

**Value**

ggplot type object with violin plot for each plot\_param

**See Also**

[calculate\\_sample\\_corr\\_distr](#), [ggplot](#)

**Examples**

```

sample_corr_distribution_plot <- plot_sample_corr_distribution(
  example_proteome_matrix,
  example_sample_annotation, batch_col = 'MS_batch',
  biospecimen_id_col = "EarTag",
  plot_param = 'batch_replicate')

corr_distribution = calculate_sample_corr_distr(data_matrix = example_proteome_matrix,
  sample_annotation = example_sample_annotation,
  batch_col = 'MS_batch', biospecimen_id_col = "EarTag")
sample_corr_distribution_plot <- plot_sample_corr_distribution.corrDF(corr_distribution,
  plot_param = 'batch_replicate')

## Not run:
sample_corr_distribution_plot <- plot_sample_corr_distribution.corrDF(corr_distribution,
  plot_param = 'batch_replicate',
  filename = 'test_sampleCorr.png',
  width = 28, height = 28, units = 'cm')

## End(Not run)

```

---

plot\_sample\_corr\_heatmap

*Sample correlation matrix (heatmap)*

---

**Description**

Plot correlation of selected samples

**Usage**

```
plot_sample_corr_heatmap(data_matrix, samples_to_plot = NULL,
  sample_annotation = NULL, sample_id_col = "FullRunName",
  factors_to_plot = NULL, cluster_rows = FALSE, cluster_cols = FALSE,
  heatmap_color = colorRampPalette(rev(brewer.pal(n = 7, name =
  "RdYlBu")))(100), color_list = NULL, filename = NULL, width = NA,
  height = NA, units = c("cm", "in", "mm"),
  plot_title = sprintf("Correlation matrix of%s samples",
  ifelse(is.null(samples_to_plot), "", " selected")), ...)
```

**Arguments**

**data\_matrix** features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example\_proteome\_matrix" for more details (to call the description, use help("example\_proteome\_matrix"))

**samples\_to\_plot** string vector of samples in data\_matrix to be used in the plot

**sample\_annotation** data frame with:

1. sample\_id\_col (this can be repeated as row names)
2. biological covariates
3. technical covariates (batches etc)

. See help("example\_sample\_annotation")

**sample\_id\_col** name of the column in sample\_annotation table, where the filenames (colnames of the data\_matrix are found).

**factors\_to\_plot** vector of technical and biological covariates to be plotted in this diagnostic plot (assumed to be present in sample\_annotation)

**cluster\_rows** boolean values determining if rows should be clustered or hclust object

**cluster\_cols** boolean values determining if columns should be clustered or hclust object

**heatmap\_color** vector of colors used in heatmap.

**color\_list** list, as returned by sample\_annotation\_to\_colors, where each item contains a color vector for each factor to be mapped to the color.

**filename** path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported

**width** option determining the output image width

**height** option determining the output image width

**units** units: 'cm', 'in' or 'mm'

**plot\_title** title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))

**...** parameters for the [pheatmap](#) visualisation, for details see examples and help to corresponding functions

**Value**

pheatmap object

**See Also**[pheatmap](#)**Examples**

```

specified_samples = example_sample_annotation$FullRunName[
  which(example_sample_annotation$order %in% 110:115)]

sample_corr_heatmap <- plot_sample_corr_heatmap(example_proteome_matrix,
  samples_to_plot = specified_samples,
  factors_to_plot = c('MS_batch', 'Diet', 'DateTime', 'digestion_batch'),
  cluster_rows= FALSE, cluster_cols=FALSE,
  annotation_names_col = TRUE, annotation_legend = FALSE,
  show_colnames = FALSE)

color_list <- sample_annotation_to_colors (example_sample_annotation,
  factor_columns = c('MS_batch', 'EarTag', "Strain",
    "Diet", "digestion_batch", "Sex"),
  numeric_columns = c('DateTime', 'order'))
sample_corr_heatmap_annotated <- plot_sample_corr_heatmap(log_transform_dm(example_proteome_matrix),
  sample_annotation = example_sample_annotation,
  factors_to_plot = c('MS_batch', 'Diet', 'DateTime', 'digestion_batch'),
  cluster_rows= FALSE, cluster_cols=FALSE,
  annotation_names_col = TRUE,
  show_colnames = FALSE, color_list = color_list)

```

---

plot\_sample\_mean\_or\_boxplot

*Plot per-sample mean or boxplots for initial assessment*

---

**Description**

Plot per-sample mean or boxplots (showing median and quantiles). In ordered samples, e.g. consecutive MS runs, order-associated effects are visualised.

**Usage**

```

plot_sample_mean(data_matrix, sample_annotation = NULL,
  sample_id_col = "FullRunName", batch_col = "MS_batch",
  color_by_batch = FALSE, color_scheme = "brewer",
  order_col = "order", vline_color = "grey", facet_col = NULL,
  filename = NULL, width = NA, height = NA, units = c("cm", "in",
  "mm"), plot_title = NULL, theme = "classic", ylimits = NULL)

plot_boxplot(df_long, sample_annotation = NULL,
  sample_id_col = "FullRunName", measure_col = "Intensity",
  batch_col = "MS_batch", color_by_batch = TRUE,
  color_scheme = "brewer", order_col = "order", facet_col = NULL,
  filename = NULL, width = NA, height = NA, units = c("cm", "in",
  "mm"), plot_title = NULL, theme = "classic", ylimits = NULL,
  outliers = TRUE)

```

**Arguments**

<code>data_matrix</code>	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example_proteome_matrix" for more details (to call the description, use <code>help("example_proteome_matrix")</code> )
<code>sample_annotation</code>	data frame with: <ol style="list-style-type: none"> <li>1. <code>sample_id_col</code> (this can be repeated as row names)</li> <li>2. biological covariates</li> <li>3. technical covariates (batches etc)</li> </ol> . See <code>help("example_sample_annotation")</code>
<code>sample_id_col</code>	name of the column in <code>sample_annotation</code> table, where the filenames (colnames of the <code>data_matrix</code> are found).
<code>batch_col</code>	column in <code>sample_annotation</code> that should be used for batch comparison (or other, non-batch factor to be mapped to color in plots).
<code>color_by_batch</code>	(logical) whether to color points and connecting lines by batch factor as defined by <code>batch_col</code> .
<code>color_scheme</code>	named vector, names corresponding to unique batch values of <code>batch_col</code> in <code>sample_annotation</code> . Best created with <a href="#">sample_annotation_to_colors</a>
<code>order_col</code>	column in <code>sample_annotation</code> that determines sample order. It is used for in initial assessment plots ( <a href="#">plot_sample_mean_or_boxplot</a> ) and feature-level diagnostics ( <a href="#">feature_level_diagnostics</a> ). Can be 'NULL' if sample order is irrelevant (e.g. in genomic experiments). For more details, order definition/inference, see <a href="#">define_sample_order</a> and <a href="#">date_to_sample_order</a>
<code>vline_color</code>	color of vertical lines, typically denoting different MS batches in ordered runs; should be NULL for experiments without intrinsic order
<code>facet_col</code>	column in <code>sample_annotation</code> with a batch factor to separate plots into facets; usually 2nd to <code>batch_col</code> . Most meaningful for multi-instrument MS experiments (where each instrument has its own order-associated effects (see <code>order_col</code> ) or simultaneous examination of two batch factors (e.g. preparation day and measurement day). For single-instrument case should be set to 'NULL'
<code>filename</code>	path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported
<code>width</code>	option determining the output image width
<code>height</code>	option determining the output image width
<code>units</code>	units: 'cm', 'in' or 'mm'
<code>plot_title</code>	title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))
<code>theme</code>	ggplot theme, by default <code>classic</code> . Can be easily overridden
<code>ylimits</code>	range of y-axis to compare two plots side by side, if required.
<code>df_long</code>	data frame where each row is a single feature in a single sample. It minimally has a <code>sample_id_col</code> , a <code>feature_id_col</code> and a <code>measure_col</code> , but usually also an <code>m_score</code> (in OpenSWATH output result file). See <code>help("example_proteome")</code> for more details.
<code>measure_col</code>	if <code>df_long</code> is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency.
<code>outliers</code>	keep (default) or remove the boxplot outliers

**Details**

functions for quick visual assessment of trends associated, overall or specific covariate-associated (see `batch_col` and `facet_col`)

**Value**

ggplot2 class object. Thus, all aesthetics can be overridden

**See Also**

[ggplot](#), [date\\_to\\_sample\\_order](#)

**Examples**

```
mean_plot <- plot_sample_mean(example_proteome_matrix, example_sample_annotation,
  order_col = 'order', batch_col = "MS_batch")

color_list <- sample_annotation_to_colors (example_sample_annotation,
  factor_columns = c('MS_batch'),
  numeric_columns = c('DateTime', 'order'))
plot_sample_mean(example_proteome_matrix, example_sample_annotation,
  order_col = 'order', batch_col = "MS_batch", color_by_batch = TRUE,
  color_scheme = color_list[["MS_batch"]])

## Not run:
mean_plot <- plot_sample_mean(example_proteome_matrix,
  example_sample_annotation,
  order_col = 'order', batch_col = "MS_batch",
  filename = 'test_meanplot.png',
  width = 28, height = 18, units = 'cm')

## End(Not run)

boxplot <- plot_boxplot(log_transform_df(example_proteome),
  sample_annotation = example_sample_annotation,
  batch_col = "MS_batch")

color_list <- sample_annotation_to_colors (example_sample_annotation,
  factor_columns = c('MS_batch'),
  numeric_columns = c('DateTime', 'order'))
plot_boxplot(log_transform_df(example_proteome),
  sample_annotation = example_sample_annotation,
  batch_col = "MS_batch", color_scheme = color_list[["MS_batch"]])

## Not run:
boxplot <- plot_boxplot(log_transform_df(example_proteome),
  sample_annotation = example_sample_annotation,
  batch_col = "MS_batch", filename = 'test_boxplot.png',
  width = 14, height = 9, units = 'in')

## End(Not run)
```



---

 plot\_split\_violin\_with\_boxplot

*Plot split violin plot (convenient to compare distribution before and after)*

---

### Description

Plot split violin plot (convenient to compare distribution before and after)

### Usage

```
plot_split_violin_with_boxplot(df, y_col = "y", col_for_color = "m",
  col_for_box = "x", colors_for_plot = c("#8f1811", "#F8C333"),
  hlineintercept = NULL, plot_title = NULL, theme = "classic")
```

### Arguments

df	data.frame with y_col, col_for_color, col_for_box
y_col	value to explore the distribution of
col_for_color	column to use to map to two colors
col_for_box	column to use to do group comparison
colors_for_plot	colors to map to col_for_color
hlineintercept	NULL: no intercept line; non-null: intercept value
plot_title	title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))
theme	ggplot theme, by default classic. Can be easily overridden

### Value

ggplot object

---

prepare\_PVCA\_df      *prepare the weights of Principal Variance Components*

---

### Description

prepare the weights of Principal Variance Components

### Usage

```
prepare_PVCA_df(data_matrix, sample_annotation,
  feature_id_col = "peptide_group_label",
  sample_id_col = "FullRunName", technical_factors = c("MS_batch",
  "instrument"), biological_factors = c("cell_line", "drug_dose"),
  fill_the_missing = -1, pca_threshold = 0.6,
  variance_threshold = 0.01)
```

**Arguments**

- data\_matrix** features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example\_proteome\_matrix" for more details (to call the description, use help("example\_proteome\_matrix"))
- sample\_annotation** data frame with:
1. sample\_id\_col (this can be repeated as row names)
  2. biological covariates
  3. technical covariates (batches etc)
- . See help("example\_sample\_annotation")
- feature\_id\_col** name of the column with feature/gene/peptide/protein ID used in the long format representation df\_long. In the wide formatted representation data\_matrix this corresponds to the row names.
- sample\_id\_col** name of the column in sample\_annotation table, where the filenames (colnames of the data\_matrix are found).
- technical\_factors** vector sample\_annotation column names that are technical covariates
- biological\_factors** vector sample\_annotation column names, that are biologically meaningful covariates
- fill\_the\_missing** numeric value determining how missing values should be substituted. If NULL, features with missing values are excluded. If NULL, features with missing values are excluded.
- pca\_threshold** the percentile value of the minimum amount of the variabilities that the selected principal components need to explain
- variance\_threshold** the percentile value of weight each of the covariates needs to explain (the rest will be lumped together)

**Value**

data frame with weights and factors, combined in a way ready for plotting

**Examples**

```
matrix_test <- example_proteome_matrix[1:150, ]
pvca_df_res <- prepare_PVCA_df(matrix_test, example_sample_annotation,
  technical_factors = c('MS_batch', 'digestion_batch'),
  biological_factors = c("Diet", "Sex", "Strain"),
  pca_threshold = .6, variance_threshold = .01, fill_the_missing = -1)
```

---

proBatch                      *proBatch: A package for diagnostics and correction of batch effects, primarily in proteomics*

---

## Description

The proBatch package contains functions for analyzing and correcting batch effects (unwanted technical variation) from high-throughput experiments. Although the package has primarily been developed for mass spectrometry proteomics (DIA/SWATH), it has been designed to be applicable to most omic data with minor adaptations. It addresses the following needs:

- prepare the data for analysis
- Visualize batch effects in sample-wide and feature-level;
- Normalize and correct for batch effects.

## Arguments

df_long	data frame where each row is a single feature in a single sample. It minimally has a <code>sample_id_col</code> , a <code>feature_id_col</code> and a <code>measure_col</code> , but usually also an <code>m_score</code> (in OpenSWATH output result file). See <code>help("example_proteome")</code> for more details.
data_matrix	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example_proteome_matrix" for more details (to call the description, use <code>help("example_proteome_matrix")</code> )
sample_annotation	data frame with: <ol style="list-style-type: none"> <li>1. <code>sample_id_col</code> (this can be repeated as row names)</li> <li>2. biological covariates</li> <li>3. technical covariates (batches etc)</li> </ol> . See <code>help("example_sample_annotation")</code>
sample_id_col	name of the column in <code>sample_annotation</code> table, where the filenames (colnames of the <code>data_matrix</code> are found).
measure_col	if <code>df_long</code> is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency.
feature_id_col	name of the column with feature/gene/peptide/protein ID used in the long format representation <code>df_long</code> . In the wide formatted representation <code>data_matrix</code> this corresponds to the row names.
batch_col	column in <code>sample_annotation</code> that should be used for batch comparison (or other, non-batch factor to be mapped to color in plots).
order_col	column in <code>sample_annotation</code> that determines sample order. It is used for in initial assessment plots ( <code>plot_sample_mean_or_boxplot</code> ) and feature-level diagnostics ( <code>feature_level_diagnostics</code> ). Can be 'NULL' if sample order is irrelevant (e.g. in genomic experiments). For more details, order definition/inference, see <code>define_sample_order</code> and <code>date_to_sample_order</code>
facet_col	column in <code>sample_annotation</code> with a batch factor to separate plots into facets; usually 2nd to <code>batch_col</code> . Most meaningful for multi-instrument MS experiments (where each instrument has its own order-associated effects (see <code>order_col</code> ) or simultaneous examination of two batch factors (e.g. preparation day and measurement day). For single-instrument case should be set to 'NULL'

<code>color_by_batch</code>	(logical) whether to color points and connecting lines by batch factor as defined by <code>batch_col</code> .
<code>peptide_annotation</code>	long format data frame with peptide ID and their corresponding protein and/or gene annotations. See <code>help("example_peptide_annotation")</code> .
<code>color_scheme</code>	a named vector of colors to map to <code>batch_col</code> , names corresponding to the levels of the factor. For continuous variables, vector doesn't need to be named.
<code>color_list</code>	list, as returned by <code>sample_annotation_to_colors</code> , where each item contains a color vector for each factor to be mapped to the color.
<code>factors_to_plot</code>	vector of technical and biological covariates to be plotted in this diagnostic plot (assumed to be present in <code>sample_annotation</code> )
<code>protein_col</code>	column where protein names are specified
<code>no_fit_imputed</code>	(logical) whether to use imputed (requant) values, as flagged in <code>qual_col</code> by <code>qual_value</code> for data transformation
<code>qual_col</code>	column to color point by certain value denoted by <code>color_by_qual_value</code> . Design with inferred/requant values in OpenSWATH output data, which means argument value has to be set to <code>m_score</code> .
<code>qual_value</code>	value in <code>qual_col</code> to color. For OpenSWATH data, this argument value has to be set to 2 (this is an <code>m_score</code> value for imputed values (requant values)).
<code>plot_title</code>	title of the plot (e.g., processing step + representation level (fragments, transitions, proteins) + purpose (meanplot/corrplot etc))
<code>keep_all</code>	when transforming the data (normalize, correct) - acceptable values: all/default/minimal (which set of columns be kept).
<code>theme</code>	ggplot theme, by default <code>classic</code> . Can be easily overridden
<code>filename</code>	path where the results are saved. If null the object is returned to the active window; otherwise, the object is save into the file. Currently only pdf and png format is supported
<code>width</code>	option determining the output image width
<code>height</code>	option determining the output image width
<code>units</code>	units: 'cm', 'in' or 'mm'

## Details

To learn more about proBatch, start with the vignettes: `browseVignettes(package = "proBatch")`

## Section

Common arguments to the functions.

---

`sample_annotation_to_colors`*Generate colors for sample annotation*

---

## Description

Convert the sample annotation data frame to list of colors the list is named as columns included to use in plotting functions

## Usage

```
sample_annotation_to_colors(sample_annotation,
  sample_id_col = "FullRunName", factor_columns = c("MS_batch",
  "EarTag", "digestion_batch", "Strain", "Diet"),
  numeric_columns = c("DateTime", "order"),
  rare_categories_to_other = TRUE, guess_factors = FALSE,
  numeric_palette_type = "brewer")
```

## Arguments

`sample_annotation`

data frame with:

1. `sample_id_col` (this can be repeated as row names)
2. biological covariates
3. technical covariates (batches etc)

. See `help("example_sample_annotation")`

`sample_id_col` name of the column in `sample_annotation` table, where the filenames (column names of the `data_matrix` are found).

`factor_columns` columns of `sample_annotation` to be treated as factors. Sometimes categorical variables are depicted as integers (e.g. in column "Batch", values are 1, 2 and 3), specification here allows to map them correctly to qualitative palettes.

`numeric_columns`

columns of `sample_annotation` to be treated as continuous numeric values.

`rare_categories_to_other`

if True rare categories will be merged into the value "other"

`guess_factors` whether attempt which of the `factor_columns` are actually numeric

`numeric_palette_type`

palette to be used for numeric values coloring (can be 'brewer' and 'viridis')

## Value

list of three items:

1. list of colors;
2. data frame of colors;
3. new sample annotation (e.g. rare factor levels merged into "other")

**Examples**

```
color_scheme <- sample_annotation_to_colors (example_sample_annotation,
factor_columns = c('MS_batch', 'EarTag', "Strain",
"Diet", "digestion_batch", "Sex"),
numeric_columns = c('DateTime', 'order'))
```

---

transform_raw_data	<i>Functions to log transform raw data before normalization and batch correction</i>
--------------------	--

---

**Description**

Functions to log transform raw data before normalization and batch correction

Log transformation of the data

"Unlog" transformation of the data to pre-log form (for quantification, forcing log-transform)

**Usage**

```
log_transform_df(df_long, log_base = 2, offset = 1,
measure_col = "Intensity")
```

```
unlog_df(df_long, log_base = 2, offset = 1,
measure_col = "Intensity")
```

```
log_transform_dm(data_matrix, log_base = 2, offset = 1)
```

```
unlog_dm(data_matrix, log_base = 2, offset = 1)
```

**Arguments**

df_long	data frame where each row is a single feature in a single sample. It minimally has a sample_id_col, a feature_id_col and a measure_col, but usually also an m_score (in OpenSWATH output result file). See help("example_proteome") for more details.
log_base	base of the logarithm for transformation
offset	small positive number to prevent 0 conversion to -Inf
measure_col	if df_long is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency.
data_matrix	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. See "example_proteome_matrix" for more details (to call the description, use help("example_proteome_matrix"))

**Value**

'log\_transform\_df()' returns df\_long-size data frame, with measure\_col log transformed; with old value in another column called "beforeLog\_intensity" if "intensity" was the value of measure\_col; 'log\_transform\_dm()' returns data\_matrix format matrix

**Examples**

```
log_transformed_df <- log_transform_df(example_proteome)
```

```
log_transformed_matrix <- log_transform_dm(example_proteome_matrix,  
log_base = 10, offset = 1)
```

# Index

- \* **datasets**
  - example\_peptide\_annotation, 16
  - example\_proteome, 16
  - example\_proteome\_matrix, 17
  - example\_sample\_annotation, 17
- \* **date**
  - dates\_to\_posix, 13
- \* **matrix manipulation functions**
  - long\_to\_matrix, 23
  - matrix\_to\_long, 24
- adjust\_batch\_trend\_df
  - (correct\_batch\_effects), 8
- adjust\_batch\_trend\_dm
  - (correct\_batch\_effects), 8
- as.POSIXct, 13, 14
- autoplot.pca\_common, 37
- calculate\_feature\_CV, 3
- calculate\_peptide\_corr\_distr, 4, 38
- calculate\_PVCA, 5
- calculate\_sample\_corr\_distr, 6, 44
- center\_feature\_batch\_means\_df
  - (correct\_batch\_effects), 8
- center\_feature\_batch\_means\_dm
  - (correct\_batch\_effects), 8
- center\_feature\_batch\_medians\_df
  - (correct\_batch\_effects), 8
- center\_feature\_batch\_medians\_dm
  - (correct\_batch\_effects), 8
- check\_sample\_consistency, 7
- correct\_batch\_effects, 8
- correct\_batch\_effects\_df
  - (correct\_batch\_effects), 8
- correct\_batch\_effects\_dm
  - (correct\_batch\_effects), 8
- correct\_with\_ComBat\_df
  - (correct\_batch\_effects), 8
- correct\_with\_ComBat\_dm
  - (correct\_batch\_effects), 8
- create\_peptide\_annotation, 12
- date\_to\_sample\_order, 8, 11, 14, 15, 20, 22, 47, 48, 51
- dates\_to\_posix, 13
- define\_sample\_order, 8, 11, 15, 15, 20, 22, 47, 51
- example\_peptide\_annotation, 16
- example\_proteome, 16
- example\_proteome\_matrix, 17
- example\_sample\_annotation, 17
- feature\_level\_diagnostics, 8, 11, 15, 18, 20, 22, 47, 51
- fit\_nonlinear, 11, 22
- ggplot, 37, 38, 41, 44, 48
- hclust, 35
- log\_transform\_df (transform\_raw\_data), 54
- log\_transform\_dm (transform\_raw\_data), 54
- long\_to\_matrix, 23, 25
- matrix\_to\_long, 24, 24
- normalize, 25
- normalize\_data\_df (normalize), 25
- normalize\_data\_dm (normalize), 25
- normalize\_sample\_medians\_df
  - (normalize), 25
- normalize\_sample\_medians\_dm
  - (normalize), 25
- pheatmap, 28, 32, 45, 46
- plot\_boxplot
  - (plot\_sample\_mean\_or\_boxplot), 46
- plot\_corr\_matrix, 27
- plot\_CV\_distr, 28
- plot\_CV\_distr.df, 30
- plot\_heatmap\_diagnostic, 30
- plot\_heatmap\_generic, 32
- plot\_hierarchical\_clustering, 34
- plot\_iRT, 14
- plot\_iRT (feature\_level\_diagnostics), 18



plot\_PCA, 36  
plot\_peptide\_corr\_distribution, 4, 28, 37  
plot\_peptides\_of\_one\_protein, 12  
plot\_peptides\_of\_one\_protein  
    (feature\_level\_diagnostics), 18  
plot\_protein\_corrplot, 12, 39  
plot\_PVCA, 40  
plot\_PVCA.df, 42  
plot\_sample\_corr\_distribution, 7, 28, 43  
plot\_sample\_corr\_heatmap, 44  
plot\_sample\_mean, 14  
plot\_sample\_mean  
    (plot\_sample\_mean\_or\_boxplot), 46  
plot\_sample\_mean\_or\_boxplot, 8, 11, 15, 20, 22, 46, 47, 51  
plot\_single\_feature  
    (feature\_level\_diagnostics), 18  
plot\_spike\_in  
    (feature\_level\_diagnostics), 18  
plot\_split\_violin\_with\_boxplot, 49  
plot\_with\_fitting\_curve, 11  
plot\_with\_fitting\_curve  
    (feature\_level\_diagnostics), 18  
plotDendroAndColors, 35  
prepare\_PVCA\_df, 49  
proBatch, 24, 25, 51  
proBatch-package (proBatch), 51  
  
quantile\_normalize\_df (normalize), 25  
quantile\_normalize\_dm (normalize), 25  
  
sample\_annotation\_to\_colors, 32, 35, 41, 47, 53  
  
transform\_raw\_data, 54  
  
unlog\_df (transform\_raw\_data), 54  
unlog\_dm (transform\_raw\_data), 54