Package ‘proBatch’

June 1, 2018

Type Package

Title Tools for Batch Effects Diagnostics and Correction

Version 1.1.0

Author Jelena Čuklina <chuklina.jelena@gmail.com>

Maintainer The package maintainer <chuklina.jelena@gmail.com>

Description The proBatch package contains functions for diagnosing and removing
batch effects and other unwanted variation in high-thoughput experiment, primarily
designed for DIA proteomics data.
The diagnostic part of the package can be broadly divided in (1) Genome-wide
and (2) Gene-specific functions, explained in corresponding vignettes. Since
the diagnostic part for batch effects does require batch effect removal, here
we provide a few convenience wrappers for common batch-effect removal approaches,
namely, ComBat (Johnson et al. 2007 Biostatistics) and mean/median centering.
However, proteomics data may require more complicated technical artifact
correction approaches like non-linear fitting, which is also found in ``normalization''
section of this package.
The approaches are described in (Čuklina et al. 2019, MCP)

License What license is it under?

Depends R (>= 3.4.0),
dplyr (>= 0.7.4),
ggplot2

Encoding UTF-8

LazyData true

Imports Biobase,
corrplot,
ggfortify,
lazyeval,
lubridate,
magrittr,
pheatmap,
preprocessCore,
pvca,
RColorBrewer,
readr,
reshape2,
rlang,
scales,

sva,
tibble,
tidyverse (>= 1.2.1),
wesanderson,
WGCNA

Suggests knitr,
rmarkdown,
SWATH2stats

VignetteBuilder knitr

RoxygenNote 6.0.1

R topics documented:

`boxplot_all_steps` *Plot boxplots to compare various data normalization steps/approaches WARNING: extremely slow for big dataframes*

Description

Plot boxplots to compare various data normalization steps/approaches WARNING: extremely slow for big dataframes

Usage

```
boxplot_all_steps(list_of_dfs, sample_annotation, batch_column, step = NULL)
```

Arguments

`list_of_dfs`	list of data frames of format, specified in ‘plot_boxplot‘
`sample_annotation`	data matrix with 1) ‘sample_id_col‘ (this can be repeated as row names) 2) biological and 3) technical covariates (batches etc)
`batch_column`	column in ‘sample_annotation‘ that should be used for batch comparison
`step`	normalization step (e.g. ‘Raw‘ or ‘Quantile_normalized‘ or ‘qNorm_ComBat‘). 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

ggplot object

See Also

`plot_boxplot`

`clean_requants` *Remove requanted and sparse features*

Description

Cleans dataset `df_long` (proBatch) by removing requanted features and features not meeting user define sparsness criterias.

Usage

```
clean_requants(df_long, sample_annotation, batch_column = "MS_batch.final",
  feature_id_column = "peptide_group_label", threshold_batch = 0.3,
  threshold_global = 0.3)
```

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_column` and a `measure_column`, but usually also an `m_score` (in OpenSWATH output result file)
`sample_annotation`	data matrix with: 1. `sample_id_col` (this can be repeated as row names) 2. biological covariates 3. technical covariates (batches etc)
`batch_column`	column in `sample_annotation` that should be used for batch comparison
`feature_id_column`	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.
`threshold_batch`	maximally tollerated fraction of missing values for a feature in a batch
`threshold_global`	maximally tollerated fraction of globally missing values for a feature

Value

`df_long` (proBatch) like data frame filtered as follow:

- remove requant values
- remove features not meeting batch or global sparsness thresholds

See Also

Other dataset cleaning functions: `remove_peptides_with_missing_batch`, `summarize_peptides`

`color_list_to_df` *Color list to data frame*

Description

Turn color list to df (some plotting functions require the latter)

Usage

```
color_list_to_df(color_list, sample_annotation)
```

Arguments

`sample_annotation`

Value

a data frame representation of the input color list

config	*Data normalization and batch adjustment methods*

Description

Data normalization and batch adjustment methods

Median normalization of the data (per batch median)

Median normalization of the data (global)

normalize with the custom (continuous) fit

Standardized input-output ComBat normalization ComBat allows users to adjust for batch effects in datasets where the batch covariate is known, using methodology 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 cleaned and normalized before batch effect removal.

Usage

```
normalize_medians_batch(data_long, sample_annotation = NULL,
  sample_id_column = "FullRunName", batch_column = "MS_batch.final",
  feature_id_col = "peptide_group_label", measure_column = "Intensity")

normalize_medians_global(data_long, sample_id_column = "FullRunName",
  measure_column = "Intensity")

normalize_custom_fit(data_matrix, sample_annotation,
  batch_column = "MS_batch.final", feature_id_col = "peptide_group_label",
  sample_id_column = "FullRunName", measure_col = "Intensity",
  sample_order_col = "order", fit_func = fit_nonlinear, return_long = F,
  ...)

correct_with_ComBat(data_matrix, sample_annotation,
  batch_column = "MS_batch.final", par.prior = TRUE)
```

Arguments

`sample_annotation`	data matrix with 1) ‘sample_id_col‘ (this can be repeated as row names) 2) biological and 3) technical covariates (batches etc)
`batch_column`	column in ‘sample_annotation‘ that should be used for batch comparison
`measure_column`	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. Usually the log transformed version of the original data
`sample_order_col`	column, determining the order of sample MS run, used as covariate to fit the non-linear fit
`fit_func`	function to fit the (non)-linear trend

`return_long`	whether the result should be the "long" data frame (as ‘df_long‘) or "wide" (as ‘data_matrix‘)
`...`	other parameters, usually those of the ‘fit_func‘
`par.prior`	
`sample_id_col`	name of the column in sample_annotation file, where the filenames (colnames of the data matrix are found)

Value

‘data_matrix‘-size data matrix with batch-effect corrected by ‘ComBat‘

`convert_to_matrix` *Long to wide conversion*

Description

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

Usage

```
convert_to_matrix(df_long, feature_id_column = "peptide_group_label",
  measure_column = "Intensity", sample_id_col = "FullRunName")
```

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_column` and a `measure_column`, but usually also an `m_score` (in OpenSWATH output result file)
`feature_id_column`	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_column`	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 file, where the filenames (colnames of the data matrix are found)

Value

`data_matrix` (proBatch) like matrix (features in rows, samples in columns)

See Also

Other matrix manipulation functions: `join_data_matrices`, `matrix_to_long`

dates_to_posix *Convert data/time to POSIXct*

Description

convert date/time column of sample_annotation to POSIX format required to keep number-like behaviour

Usage

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

Arguments

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

Value

sample annotation file with column names as 'new_time_column' with POSIX-formatted date

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"), order_column = "order", instrument_col = "instrument")
```

Arguments

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

Value

sample annotation file with column names as 'new_time_column' with POSIX-formatted date & `order_column` used in some diagnostic plots (e.g. `plot_iRTs`, `plot_sample_mean`)

`define_batches_by_MS_pauses`
Batch by date/time and instrument

Description

Identify long stretches of time between samples on a per instrument basis and split them into batches.

Usage

```
define_batches_by_MS_pauses(sample_annotation, threshold,
  runtime_col = "RunDateTime", minimal_batch_size = 5,
  instrument_col = "instr", batch_name = "MS_batch")
```

Arguments

`sample_annotation`	data matrix with: 1. `sample_id_col` (this can be repeated as row names) 2. biological covariates 3. technical covariates (batches etc)
`threshold`	time difference that would mean there was an interruption
`runtime_col`	POSIX or numeric-like column corresponding to the sample MS profile acquisition timepoint
`minimal_batch_size`	minimal number of samples in a batch
`instrument_col`	column specifying MS instrument used to acquired data (to account for the presence of multiple instruments in `sample_annotation`)
`batch_name`	string with a self-explanatory name for the batch (e.g. 'MS_batch' for MS-proteomics) to which batch number will be added

Value

`sample_annotation` data matrix with an additional column to indicate sample batching by MS run time and instrument.

`define_batches_by_MS_pauses_within_instrument` *Batch by date/time*

Description

Identify long stretches of time between samples and split them into batches. Most users are going to want to call define_batches_by_MS_pauses, rather then this function

Usage

```
define_batches_by_MS_pauses_within_instrument(date_vector, threshold,
  minimal_batch_size = 5, batch_name = "MS_batch")
```

Arguments

`date_vector`	POSIX or numeric-like vector corresponding to the sample MS profile acquisition timepoint
`threshold`	time difference that would mean there was an interruption
`minimal_batch_size`	minimal number of samples in a batch
`batch_name`	string with a self-explanatory name for the batch (e.g. 'MS_batch' for MS-proteomics) to which batch number will be added

Value

vector of batches for each sample

`distribution_of_cor` *Plot distribution of correlations*

Description

Plot distribution of correlations

Usage

```
distribution_of_cor(data_matrix_sub, facet_var = NULL, theme = "classic")
```

Arguments

`data_matrix_sub`

`example_peptide_annotation` *Peptide annotation data*

Description

This is data from Evan's aging study annotated with gene names

Usage

```
example_peptide_annotation
```

Format

A data frame with 200625 rows and 8 variables:

Peptide 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' ... other parameters determined by 'summarize_peptides' function

`example_proteome` *Example protein data*

Description

This is data from Evan's aging study with all iRT, spike-in peptides, few random peptides and QTL proteins for biological signal improvement demonstration

Usage

```
example_proteome
```

Format

A data frame with 200625 rows and 8 variables:

peptide_group_label peptide ID, which is regular feature level. This column is mostly used as 'feature_id_col'

RT retention time. Relevant to identify retention time related bias

Intensity peptide group intensity in given sample. Used in function as 'measure_col'

ProteinName Protein group ID, specified as N/UniProtID1|UniProtID2|..., where N is number of protein peptide group maps to. If 1/UniProtID, then this is proteotypic peptide

assay_rt retention time as in DIA library

m_score peptide group identification FDR as determined by pyProphet

FullRunName name of the file, in most functions used for 'sample_id_col' #' ...

Source

PRIDE ID will be added in future

`example_sample_annotation1` *Sample annotation data version 1*

Description

This is data from Evan's aging study with mock instruments to show how instrument-specific functionality works

Usage

```
example_sample_annotation1
```

Format

A data frame with 375 rows and 18 variables:

FullRunName name of the file, in most functions used for 'sample_id_col'

MS_batch.final mass-spectrometry batch: 7-level factor of manually annotated batches

EarTag mouse ID, i.e. ID of the biological object

Strain mouse strain ID - biological covariate #1

Diet diet - either 'HFD' = 'High Fat Diet' or 'CD' = 'Chow Diet'. 'Mix' stands for mixture of several samples

Sex mice sex - 3-level biological covariate. Possible values - "

Age_Days mice age at sampling - numeric biological covariate

RunDate mass-spectrometry running date. In combination with 'RunTime' used for running order determination

RunTime mass-spectrometry running time. In combination with 'RunDate' used for running order determination

SacrificeDate date of mouse sacrifice - technical covariate

ProteinPrepDate date of protein preparation - technical covariate ...

`fit_nonlinear` *Fit a non-linear trend*

Description

Fit a non-linear trend

Usage

```
fit_nonlinear(dataDF, response.var = "y", expl.var = "x",
  noFitRequants = F, fitFunc = "kernel_smooth", with_df = F, ...)
```

Arguments

`response.var`	the name of the column in dataDF with the response variable
`expl.var`	the name of the column in dataDF with the explanatory variable
`noFitRequants`	(logical) whether to fit requanted values
`fitFunc`	function to use for the fit (‘kernel_smooth‘, ‘smooth_spline‘, or ‘loess_regression‘)
`with_df`	
`...`	additional paramters to be passed to the fitting function

Value

vector of fitted response values

`generate_colors_for_numeric`

Generates color list

Description

Generates a list of colors for a vector of numeric, POSIXct (i.e. the (signed) number of seconds since the beginning of 1970 , or factors

Usage

```
generate_colors_for_numeric(num_col, palette_type = "brewer", i = 1,
  granularity = 10)
```

Arguments

`num_col`	a vector of type numeric of factor to generate colors for
`palette_type`	’brewer’ or ’viridis’
`i`	if `palette_type` is ’brewer’ the palette argument to `brewer_pal`. If `palette_type` is ’viridis’ the option argument to virids_pal ()
`granularity`	the breaks to use when generating colors for num_col

`get_sample_corr_distrib`

Calculates correlation distribution for all pairs of the replicated samples

Description

Calculates correlation distribution for all pairs of the replicated samples

Usage

```
get_sample_corr_distrib(cor_proteome, sample_annotation,
  sample_id_col = "FullRunName", biospecimen_id_col = "EarTag",
  batch_col = "MS_batch.final")
```

Arguments

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

`sample_id_col`
: name of the column in sample_annotation file, where the filenames (colnames of the data matrix are found)

`batch_col`

`join_data_matrices` *Join data matrices*

Description

Joins 2 or more data matrices

Usage

```
join_data_matrices(matrices_list, step, sample_annotation,
  measure_column = "Intensity")
```

Arguments

`matrices_list`
: list of matrices in `data_matrix` (proBatch) format to be joined

`step`
: normalization step (e.g. ‘Raw‘ or ‘Quantile_normalized‘ or ‘qNorm_ComBat‘). 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‘

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

`measure_column`
: if `df_long` is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency

Value

`df_long` (proBatch) like data frame with a row, having entries for:

1. `feature_id_col` (e.g. peptide name)
2. `sample_id_col` (e.g. filename)
3. `measure_col` (e.g. intensity/expression)
4. `step` (e.g. ‘raw‘, ‘quantile_norm‘)

See Also

Other matrix manipulation functions: `convert_to_matrix`, `matrix_to_long`

`matrix_to_long` *Wide to long conversion*

Description

Convert from wide matrix to a long data frame representation

Usage

```
matrix_to_long(data_matrix, sample_annotation,
  feature_id_column = "peptide_group_label", measure_column = "Intensity",
  sample_id_col = "FullRunName", step = NA)
```

Arguments

`data_matrix`	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. Usually the log transformed version of the original data
`sample_annotation`	data matrix with: 1. `sample_id_col` (this can be repeated as row names) 2. biological covariates 3. technical covariates (batches etc)
`feature_id_column`	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_column`	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 file, where the filenames (colnames of the data matrix are found)
`step`	normalization step (e.g. ‘Raw‘ or ‘Quantile_normalized‘ or ‘qNorm_ComBat‘). 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: `convert_to_matrix`, `join_data_matrices`

`merge_rare_levels` *Replaces rare levels with other*

Description

Replaces levels with a maximal occurence of 1 with other

Usage

```
merge_rare_levels(column)
```

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

Description

cluster the data matrix to visually inspect which confounder dominates

Usage

```
plot_clustering(data_matrix, color_df, title = "Clustering of raw samples",
  distance = "euclidean", agglomeration = "complete", label_samples = T,
  label_font = 0.2, 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. in most function, it is assumed that this is the log transformed version of the original data
`color_df`	data frame of colors, as created by ‘sample_annotation_to_colors‘
`title`	Title of the plot (usually, processing step + representation level (fragments, transitions, proteins))
`distance`	distance metric used for clustering
`agglomeration`	agglomeration methods as used by ‘hclust‘
`...`	other parameters of ‘plotDendroAndColors‘ from ‘WGCNA‘ package

See Also

`hclust`, `sample_annotation_to_colors`, `plotDendroAndColors`

plot_corr_between_samples

Sample correlation plot

Description

Plot correlation of selected samples

Usage

```
plot_corr_between_samples(data_matrix, samples_to_plot, flavor = "corrplot",
  ...)
```

Arguments

`data_matrix`	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. Usually the log transformed version of the original data
`samples_to_plot`	string vector of samples in `data_matrix` to be used in the plot
`...`	parameters for the corrplot visualisation

plot_corr_plot_protein

Protein correlation plot

Description

Plots correlation plot of a single protein

Usage

```
plot_corr_plot_protein(data_matrix, protein_name, peptide_annotation,
  prot.column = "ProteinName", peptide_col_name = "peptide_group_label",
  title = NULL, ...)
```

Arguments

`data_matrix`	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. Usually the log transformed version of the original data
`protein_name`	the name of the protein
`prot.column`	the column name in `peptide_annotation` with protein names
`peptide_col_name`	the column name in `peptide_annotation` with peptide names
`title`	The title of the plot
`...`	parameters for the corrplot visualisation

Examples

```
plot_corr_plot(q_norm_proteome, protein_name = 'Haao',
               peptide_annotation = peptide_annotation, prot.column = 'Gene',
               title = 'Haao protein peptides after quantile norm',
               number.cex=0.75, tl.cex = .75
               mar=c(0,0,1,0))
```

`plot_heatmap` *Plot the heatmap of samples*

Description

Plot the heatmap of samples

Usage

```
plot_heatmap(data_matrix, sample_annotation = NULL, fill_the_missing = T,
  cluster_rows = F, cluster_cols = F, annotation_color_list = NA,
  filename = NA, ...)
```

Arguments

`data_matrix`	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. in most function, it is assumed that this is the log transformed version of the original data
`sample_annotation`	data matrix with 1) ‘sample_id_col‘ (this can be repeated as row names) 2) biological and 3) technical covariates (batches etc)
`fill_the_missing`	boolean value determining if missing values should be substituted with -1 (and colored with black)
`cluster_rows`	boolean value determining if rows should be clustered
`cluster_cols`	boolean value determining if columns should be clustered
`annotation_color_list`	list specifying colors for columns (samples). Best created by ‘sample_annotation_to_colors‘
`filename`	filepath where to save the image
`...`	other parameters of pheatmap

Value

object returned by ‘pheatmap‘

See Also

`sample_annotation_to_colors`, `pheatmap` plot_iRTs *Plot iRT measurements*

Description

Creates a iRT facetted ggplot2 plot of the value in measure_column vs order_column using plot_peptide_level. Additionally, the resulting plot can also be facetted by batch.

Usage

```
plot_iRTs(df_long, sample_annotation, order_column = NULL,
  irt_pattern = "iRT", batch_column = "MS_batch.final",
  sample_id_col = "FullRunName", feature_id_column = "peptide_group_label",
  measure_column = "Intensity", title = "iRT peptide profile", ...)
```

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_column and a measure_column, but usually also an m_score (in OpenSWATH output result file)
sample_annotation	data matrix with: 1. sample_id_col (this can be repeated as row names) 2. biological covariates 3. technical covariates (batches etc)
order_column	column in sample_annotation that determines sample order. It is used for certain diagnostics and normalisations.
irt_pattern	substring used to identify irts proteins in the column 'ProteinName'
batch_column	column in sample_annotation that should be used for batch comparison
sample_id_col	name of the column in sample_annotation file, where the filenames (colnames of the data matrix are found)
feature_id_column	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_column	if df_long is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency
title	the string indicating the source of the peptides
...	additional arguments to plot_peptide_level function

Value

ggplot2 type plot of measure_column vs order_column, faceted by irt_pattern containing proteins and (optionally) by batch_column

See Also

Other feature-level diagnostic functions: plot_peptide_level, plot_spike_ins, plot_with_fitting_curve

plot_pca *plot PCA plot*

Description

plot PCA plot

Usage

```
plot_pca(data_matrix, sample_annotation, color_by = "MS_batch",
  PC_to_plot = c(1, 2), colors_for_factor = 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. in most function, it is assumed that this is the log transformed version of the original data
sample_annotation	data matrix with 1) ‘sample_id_col‘ (this can be repeated as row names) 2) biological and 3) technical covariates (batches etc)
color_by	column name (as in ‘sample_annotation‘) to color by
PC_to_plot	principal component numbers for x and y axis
colors_for_factor	named vector of colors for the ‘color_by‘ variable
theme	ggplot theme, by default ‘classic‘. Can be easily overriden (see examples)

Value

ggplot scatterplot colored by factor levels of column specified in ‘factor_to_color‘

plot_peptide_level *Plot peptide measurements*

Description

Creates a peptide facetted ggplot2 plot of the value in `measure_column` vs `order_column`. Additionally, the resulting plot can also be facetted by batch.

Usage

```
plot_peptide_level(pep_name, df_long, sample_annotation, order_column = NULL,
  sample_id_col = "FullRunName", batch_column = "MS_batch.final",
  measure_column = "Intensity", feature_id_column = "peptide_group_label",
  geom = c("point", "line"), color_by_batch = F, facet_by_batch = F,
  title = NULL, requant = NULL, theme = "classic")
```

Arguments

`pep_name`	name of the peptide for diagnostic profiling
`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_column` and a `measure_column`, but usually also an `m_score` (in OpenSWATH output result file)
`sample_annotation`	data matrix with: 1. `sample_id_col` (this can be repeated as row names) 2. biological covariates 3. technical covariates (batches etc)
`order_column`	column in `sample_annotation` that determines sample order. It is used for certain diagnostics and normalisations.
`sample_id_col`	name of the column in sample_annotation file, where the filenames (colnames of the data matrix are found)
`batch_column`	column in `sample_annotation` that should be used for batch comparison
`measure_column`	if `df_long` is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency
`feature_id_column`	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.
`geom`	whether to show the feature as points and/or connect by lines
`color_by_batch`	(logical) whether to color points by batch
`facet_by_batch`	(logical) whether to plot each batch in its own facet
`title`	the string indicating the source of the peptides
`requant`	if data frame: requant values; if logical: whether to indicate requant values (requires 'requant' or 'm_score' column in `df_long`)
`theme`	plot theme (default is 'classical'; other options not implemented)

Value

ggplot2 type plot of `measure_column` vs `order_column`, faceted by `pep_name` and (optionally) by `batch_column`

See Also

Other feature-level diagnostic functions: `plot_iRTs`, `plot_spike_ins`, `plot_with_fitting_curve`

`plot_pvca` *Plot variance distribution by variable*

Description

Plot variance distribution by variable

Usage

```
plot_pvca(data_matrix, sample_annotation, sample_id_column = "FullRunName",
  technical_covariates = c("MS_batch", "instrument"),
  biological_covariates = c("cell_line", "drug_dose"), title = NULL,
  colors_for_bars = NULL, threshold_pca = 0.6, threshold_var = 0.01,
  theme = "classic")
```

Arguments

`data_matrix`	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. in most function, it is assumed that this is the log transformed version of the original data
`sample_annotation`	data matrix with 1) ‘sample_id_col‘ (this can be repeated as row names) 2) biological and 3) technical covariates (batches etc)
`technical_covariates`	vector ‘sample_annotation‘ column names that are technical covariates
`biological_covariates`	vector ‘sample_annotation‘ column names, that are biologically meaningful covariates
`title`	Title of the plot (usually, processing step + representation level (fragments, transitions, proteins))
`colors_for_bars`	four-item color vector, specifying colors for the following categories: c(’residual’, ’biological’, ’biol:techn’, ’technical’)
`threshold_pca`	the percentile value of the minimum amount of the variabilities that the selected principal components need to explain
`threshold_var`	the percentile value of weight each of the covariates needs to explain (the rest will be lumped together)

Value

list of two items: plot =gg, df = pvca_res

See Also

`sample_annotation_to_colors`

`plot_sample_corr_distribution`
Create violin plot of correlation distribution Typically to visualize within batch vs within replicate vs non-related sample correlation

---

Description

Create violin plot of correlation distribution Typically 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.final", covariate = "EarTag",
  title = "Correlation_distribution", plot_param = "batch_the_same")
```

Arguments

`plot_param`

`plot_sample_means_or_boxplots`
Plot per-sample average or boxplot (distribution) vs order (if the real running order available)

Description

Plot per-sample average or boxplot (distribution) vs order (if the real running order available)

Usage

```
plot_sample_mean(data_matrix, sample_annotation = NULL,
  sample_id_col = "FullRunName", order_column = "order",
  batch_column = NULL, facet_column = "instrument", color_by_batch = F,
  color_scheme = "brewer", theme = "classic", title = NULL,
  order_per_facet = F)

plot_boxplot(df_long, sample_annotation = NULL,
  sample_id_column = "FullRunName", measure_col = "Intensity",
  order_column = "order", batch_column = "MS_batch.final",
  facet_column = "instrument", color_by_batch = T,
  color_scheme = "brewer", theme = "classic", title = NULL,
  order_per_facet = F)
```

Arguments

`data_matrix`	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. in most function, it is assumed that this is the log transformed version of the original data
`sample_annotation`	data matrix with 1) 'sample_id_col' (this can be repeated as row names) 2) biological and 3) technical covariates (batches etc)
`sample_id_col`	name of the column in sample_annotation file, where the filenames (colnames of the data matrix are found)
`order_column`	column where running order is specified.
`batch_column`	column in 'sample_annotation' that should be used for batch comparison
`facet_column`	recommended if more than one batch covariate is present. Faceting is most suited to examine instruments separately
`color_by_batch`	should the each batch be represented with its own color?
`color_scheme`	named vector, names corresponding to unique batch values as specified in 'sample_annotation'
`theme`	ggplot theme, by default 'classic'. Can be easily overriden (see examples)
`title`	Title of the plot (usually, processing step + representation level (fragments, transitions, proteins))
`order_per_facet`	if order is defined ignoring facets (usually instrument), re-define order per-batch
`df_long`	data frame where each row is a single feature in a single sample, thus it has minimally, 'sample_id_col', 'feature_id_column' and 'measure_column', but usually also 'm_score' (in OpenSWATH output result file)
`measure_column`	if 'df_long' is among the parameters, it is the column with expression/abundance/intensity, otherwise, it is used internally for consistency

Details

functions for quick visual assessment of trends associated, overall or specific covariate-associated (see 'batch_column' and 'facet_column')

Value

ggplot2 class object. Thus, all aesthetics can be overriden

`plot_spike_ins` *Plot spike-in measurements*

Description

Creates a spike-in facetted ggplot2 plot of the value in `measure_column` vs `order_column` using `plot_peptide_level`. Additionally, the resulting plot can also be facetted by batch.

Usage

```
plot_spike_ins(df_long, sample_annotation, order_column = "order",
  spike_ins = "BOVIN", sample_id_column = "FullRunName",
  batch_column = "MS_batch", measure_column = "Intensity",
  feature_id_column = "peptide_group_label",
  title = "Spike-in BOVINE protein peptides", ...)
```

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_column` and a `measure_column`, but usually also an `m_score` (in OpenSWATH output result file)
`sample_annotation`	data matrix with: 1. `sample_id_col` (this can be repeated as row names) 2. biological covariates 3. technical covariates (batches etc)
`order_column`	column in `sample_annotation` that determines sample order. It is used for certain diagnostics and normalisations.
`spike_ins`	substring used to identify spike-in proteins in the column 'ProteinName'
`batch_column`	column in `sample_annotation` that should be used for batch comparison
`measure_column`	if `df_long` is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency
`feature_id_column`	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.
`title`	the string indicating the source of the peptides
`...`	additional arguments to `plot_peptide_level` function

Value

ggplot2 type plot of `measure_column` vs `order_column`, faceted by `spike_ins` containing proteins and (optionally) by `batch_column`

See Also

Other feature-level diagnostic functions: `plot_iRTs`, `plot_peptide_level`, `plot_with_fitting_curve`

`plot_with_fitting_curve` *Plot peptide measurements across multi-step analysis*

Description

Plot Intensity of a few representative peptides for each step of the analysis including the fitting curve

Usage

```
plot_with_fitting_curve(pep_name, data_df_all_steps, sample_annotation, fit_df,
  fit_value_var = "fit", fit_step = "3_loess_fit", order_column = NULL,
  sample_id_col = "FullRunName", batch_column = "MS_batch",
  measure_column = "Intensity", feature_id_column = "peptide_group_label",
  geom = c("point", "line"), color_by_batch = F, facet_by_batch = F,
  title = NULL, requant = NULL, theme = "classic")
```

Arguments

`pep_name`	name of the peptide for diagnostic profiling
`data_df_all_steps`	data frame, similar to `df_long` proBatch, where each row is a single feature in a single sample, at a certain step of the analysis (minimally raw and after linear normalization) thus it has minimally the following columns: `sample_id_col`, `feature_id_column`, `measure_column`, and `fit_step`, but usually also `m_score`
`sample_annotation`	data matrix with: 1. `sample_id_col` (this can be repeated as row names) 2. biological covariates 3. technical covariates (batches etc)
`order_column`	column in `sample_annotation` that determines sample order. It is used for certain diagnostics and normalisations.
`sample_id_col`	name of the column in sample_annotation file, where the filenames (colnames of the data matrix are found)
`batch_column`	column in `sample_annotation` that should be used for batch comparison
`measure_column`	if `df_long` is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency
`feature_id_column`	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.
`geom`	for the intensity `measure_col` profile:
`color_by_batch`	(logical) whether to color points by batch
`facet_by_batch`	(logical) whether to plot each batch in its own facet
`title`	the string indicating the source of the peptides
`requant`	if data frame: requant values; if logical: whether to indicate requant values (requires 'requant' or 'm_score' column in `df_long`)
`theme`	plot theme (default is 'classical'; other options not implemented)

Value

ggplot-class plot with minimally two facets (before and after non-linear fit) with `measure_column` (Intensity) vs `order_column` (injection order) for selected peptides (specified in `pep_name`)

See Also

Other feature-level diagnostic functions: `plot_iRTs`, `plot_peptide_level`, `plot_spike_ins`

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

Description

It adresses the following needs:

- prepare the original data (e.g. OpenSWATH output matrix and sample annotation file) for analysis. However, you might need to use 'SWATH2stats' additionally
- Diagnose batch effects, sample-wide and feature-level
- Correct for batch effects (normalize the data). Other useful package for this purpose is 'Normalyzer'.

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_column` and a `measure_column`, but usually also an `m_score` (in OpenSWATH output result file)
`data_matrix`	features (in rows) vs samples (in columns) matrix, with feature IDs in rownames and file/sample names as colnames. Usually the log transformed version of the original data
`sample_annotation`	data matrix with: 1. `sample_id_col` (this can be repeated as row names) 2. biological covariates 3. technical covariates (batches etc)
`sample_id_col`	name of the column in sample_annotation file, where the filenames (colnames of the data matrix are found)
`batch_column`	column in `sample_annotation` that should be used for batch comparison
`order_column`	column in `sample_annotation` that determines sample order. It is used for certain diagnostics and normalisations.
`measure_column`	if `df_long` is among the parameters, it is the column with expression/abundance/intensity; otherwise, it is used internally for consistency
`feature_id_column`	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.

Details

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

`quantile_normalize` *Quantile normalization of the data, ensuring that the row and column names are retained*

Description

Quantile normalization of the data, ensuring that the row and column names are retained

Usage

```
quantile_normalize(data_matrix)
```

Arguments

`data_matrix`	log transformed data matrix (features in rows and samples in columns)

Value

'data_matrix'-size matrix, with columns quantile-normalized

`remove_peptides_with_missing_batch`
Remove features missing in at least one batch

Description

Cleans dataset `df_long` (proBatch) by removing all features that are not present in every batch

Usage

```
remove_peptides_with_missing_batch(df_long, batch_column = "MS_batch.final",
  feature_id_column = "peptide_group_label")
```

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_column` and a `measure_column`, but usually also an `m_score` (in OpenSWATH output result file)
`batch_column`	column in `sample_annotation` that should be used for batch comparison
`feature_id_column`	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.

Details

useful for some downstream functions as ComBat normalization, that would not work otherwise

Value

`df_long` (proBatch) like data frame freed of features that were not detected in each batch

See Also

Other dataset cleaning functions: `clean_requants`, `summarize_peptides`

`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 potting functions

Usage

```
sample_annotation_to_colors(sample_annotation, columns_for_plotting = NULL,
  sample_id_column = NULL, factor_columns = NULL,
  not_factor_columns = NULL, rare_categories_to_other = T,
  numerics_to_log = F, numeric_palette_type = "brewer", granularity = 10)
```

Arguments

`sample_annotation`	data matrix with: 1. `sample_id_col` (this can be repeated as row names) 2. biological covariates 3. technical covariates (batches etc)
`columns_for_plotting`	only consider these columns from sample_annotation
`factor_columns`	columns of sample_annotation to be treated as factors. Note that factor and character columns are treated as factors by default.
`not_factor_columns`	don't treat these columns as factors. This can be used to override the default behaviour of considering factors and character columns as factors.
`rare_categories_to_other`	if True rare categories will be merged as other
`numerics_to_log`	NOT IMPLEMENTED!
`numeric_palette_type`	palette to be used for numeric values coloring
`granularity`	number of colors to map to the number vector (equally spaced between minimum and maximum)

Value

list of colors

`sample_random_peptides` *sample random peptides for diagnostics*

Description

sample random peptides for diagnostics

Usage

```
sample_random_peptides(proteome, seed = 1, pep_per_group = 3,
  groups_RT = 10, groups_intensity = 5)
```

Arguments

`summarized_proteome`

`summarize_peptides` *Summarize run features*

Description

Summarizes various peptide properties on a per sample basis. By default will summarize RT, Intensity and m_score. If your feature does not have some of these set them to NULL when calling.

Usage

```
summarize_peptides(df_long, sample_id_col = "FullRunName",
  feature_id_column = "peptide_group_label", RT = "RT",
  Intensity = "Intensity", m_score = "m_score")
```

Details

summarize peptides by sample (ranking) and on the contrary, across peptide-wise across samples

Value

a data frame summarizing features in a dataset on a per sample basis. The following columns are returned: 'RT_mean', 'Int_mean', 'numb_requants', 'median_m_score', 'mean_m_score', 'median_good_m_score' (median of 'm_score' excluding requants)

See Also

Other dataset cleaning functions: `clean_requants`, `remove_peptides_with_missing_batch`

Index