Package 'proBatch'

April 24, 2018

Type Package

Title Tools for Batch Effects Diagnostics and Correction

Version 1.0.0

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 (>= 3.4.1)

Encoding UTF-8

LazyData true

Imports tidyverse (>= 1.2.1),
reshape2,
lazyeval,
readr,
WGCNA,
rlang

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*

Description

Plot boxplots to compare various data normalization steps/approaches

Usage

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

Arguments

`steps`

clean_requants *Remove peptides with too many missing peptides*

Description

Remove peptides with too many missing peptides

Usage

```
clean_requants(df_long, sample_annotation, batch_column,
  feature_id_column = "peptide_group_label", threshold_batch = 0.7,
  threshold_global = 0.5)
```

Arguments

threshold_global

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

Description

cluster the data matrix to visually inspect which confounder dominates

Usage

```
cluster_samples(data_matrix, color_df, plot_title, ...)
```

Arguments

title

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

Description

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

Usage

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

Arguments

sample_annotation

convert_to_matrix *Convert from long data frame to data matrix (features in rows, samples in columns)*

Description

Convert from long data frame to data matrix (features in rows, samples in columns)

Usage

```
convert_to_matrix(data_df_long, feature_id_column = "peptide_group_label",
  measure_column = "Intensity", sample_id_column = "FullRunName")
```

Arguments

sample_id_column

correct_with_ComBat *Standardized input-output ComBat normalization*

Description

Standardized input-output ComBat normalization

Usage

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

Arguments

data_matrix

dates_to_posix *convert date/time column to POSIX format required to keep number-like behaviour*

Description

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

Usage

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

Arguments

dateTimeFormat

`date_to_sample_order` *convert date to order*

Description

convert date to order

Usage

```
date_to_sample_order(sample_annotation, time_column,
  new_time_column = "DateTime", dateTimeFormat = c("%b_%d",
  "%H:%M:%S"), order_column = "order")
```

Arguments

`order_column`

`define_batches_by_MS_pauses` *Identify stretches of time between runs that are long and split a batches by them*

Description

Identify stretches of time between runs that are long and split a batches by them

Usage

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

Arguments

`batch_name`

`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`

`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

```
...
```

`generate_colors_for_numeric` *generate a list of colors for the dataframe with all columns numeric (or date)*

Description

generate a list of colors for the dataframe with all columns numeric (or date)

Usage

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

Arguments

```
i
```

`get_sample_corr_distrib` *calculate correlation distribution for all pairs of the replicated samples*

Description

calculate 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

batch_col

---

gg_boxplot *plot boxplot of data, optionally colored by batch*

Description

plot boxplot of data, optionally colored by batch

Usage

```
gg_boxplot(data_df_long, sample_annotation, batch_column,
  order_column = "order", measure_col = "Intensity", fill_batch = T,
  theme = "classic", title = NULL)
```

Arguments

batch_column

---

join_data_matrices *join list of matrices from different transformation steps into joined data frame*

Description

join list of matrices from different transformation steps into joined data frame

Usage

```
join_data_matrices(matrix_list, Step, sample_annotation,
  measure.col = "Intensity")
```

Arguments

measure.col

matrix_to_long *Convert the features x samples data matrix to a long format (e.g. for plotting)*

Description

Convert the features x samples data matrix to a long format (e.g. for plotting)

Usage

```
matrix_to_long(data_matrix, sample_annotation, measure_col = "Intensity",
  step)
```

Arguments

sample_annotation

median_normalization *Median normalization of the data*

Description

Median normalization of the data

Usage

```
median_normalization(data_matrix, sample_annotation, batch_column,
  measure_column)
```

Arguments

data_matrix

normalize_custom_fit *normalize with the custom (continuous) fit*

Description

normalize with the custom (continuous) fit

Usage

```
normalize_custom_fit(data_matrix, sample_annotation, batch_col, feature_id_col,
  sample_id_column, measure_col, sample_order_col, fit_func, return_long = F,
  ...)
```

Arguments

...	other parameters, usually those of the 'fit_func'

plot_corr_between_samples

Plot correlation of selected samples

Description

Plot correlation of selected samples

Usage

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

Arguments

data_matrix	features x samples matrix, with sample IDs as colnames
samples_to_plot	string vector of samples from data_matrix
...	parameters for the corrplot visualisation

plot_corr_plot_protein

plot correlation plot of a single protein

Description

plot 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

title	

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_iRTs *Plot iRT peptides*

Description

Plot iRT peptides

Usage

```
plot_iRTs(data_df_long, sample_annotation, batch_column = "MS_batch.final",
  sample_id_column = "FullRunName",
  feature_id_column = "peptide_group_label", measurement.col = "Intensity",
  order_column = "order", ...)
```

Arguments

data_df_long	- "openSWATH" format data frame
...	additional arguments to plot_peptide_level function

plot_peptide_level *plot a single peptide or several peptides each in its own facet*

Description

plot a single peptide or several peptides each in its own facet

Usage

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

Arguments

theme

plot_sample_corr_distribution *Title*

Description

Title

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_mean *Plot the sample average*

Description

Plot the sample average

Usage

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

Arguments

color_scheme

plot_spike_ins *Plot Spike-in peptides/proteins*

Description

Plot Spike-in peptides/proteins

Usage

```
plot_spike_ins(data_df_long, sample_annotation, spike_ins = "BOVIN",
  order_column = "order", measurement.col = "Intensity",
  batch_column = "MS_batch", sample_id_column = "FullRunName",
  feature_id_column = "peptide_group_label", ...)
```

Arguments

...	additional arguments to plot_peptide_level function

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

Description

Plot Intensity for 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, fit_df, sample_annotation,
  fit_value_var = "fit", fit_step = "3_loess_fit",
  batch_column = "MS_batch", feature_id_column = "peptide_group_label",
  measurement.col = "Intensity", sample_id_column = "FullRunName",
  order_column = NULL, geom = c("point", "line"), color_by_batch = F,
  facet_by_batch = F, title = NULL, requant = NULL, theme = "classic",
  color_var = "fit")
```

Arguments

color_var

`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)

`remove_peptides_with_missing_batch`
remove peptides that are missing in the whole batch useful for some downstream functions as ComBat normalization, that would "choke"

Description

remove peptides that are missing in the whole batch useful for some downstream functions as ComBat normalization, that would "choke"

Usage

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

Value

data frame free of peptides that were not detected across all batches

`sample_annotation_to_colors`
convert the sample annotation data frame to list of colors the list is named as columns included to use in potting functions

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`

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

Description

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

Usage

```
summarize_peptides(proteome, sample_id = "FullRunName",
  feature_id = "peptide_group_label")
```

Arguments

`proteome`

Index