Package ‘notameViz’

October 24, 2025

Type Package

Title Workflow for non-targeted LC-MS metabolic profiling

Version 0.99.6

Description Provides visualization functionality for untargeted LC-MS metabolomics research. Includes quality control visualizations, feature-wise visualizations and results visualizations.

License MIT + file LICENSE

Encoding UTF-8

biocViews BiomedicalInformatics, Metabolomics, DataImport, MassSpectrometry, BatchEffect, MultipleComparison, Normalization, QualityControl, Visualization, Preprocessing

Depends R (>= 4.5.0), ggplot2, SummarizedExperiment

Imports BiocGenerics, cowplot, devEMF, dplyr, ggbeeswarm, ggdendro, ggrepel, grDevices, methods, notame, scales, stringr, stats, tibble, tidyr, utils

Suggests batchCorr, BiocStyle, igraph, knitr, notameStats, pcaMethods, Rtsne, testthat

URL `https://github.com/hanhineva-lab/notameViz`

BugReports `https://github.com/hanhineva-lab/notameViz/issues`

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/testthat/parallel true

git_url https://git.bioconductor.org/packages/notameViz

git_branch devel

git_last_commit 47f9fcb

git_last_commit_date 2025-10-20

Repository Bioconductor 3.23

Date/Publication 2025-10-24

Author Anton Klåvus [aut, cph] (ORCID:
<https://orcid.org/0000-0003-2612-0230>),
Jussi Paananen [aut, cph] (ORCID:
<https://orcid.org/0000-0001-5100-4907>),
Oskari Timonen [aut, cph] (ORCID:
<https://orcid.org/0000-0002-6317-6260>),
Atte Lihtamo [aut],
Vilhelm Suksi [aut, cre] (ORCID:
<https://orcid.org/0009-0005-1108-518X>),
Retu Haikonen [aut] (ORCID: <https://orcid.org/0000-0003-0830-3850>),
Leo Lahti [aut] (ORCID: <https://orcid.org/0000-0001-5537-637X>),
Kati Hanhineva [aut] (ORCID: <https://orcid.org/0000-0001-6834-7375>),
Ville Koistinen [ctb] (ORCID: <https://orcid.org/0000-0003-1587-8361>),
Olli Kärkkäinen [ctb] (ORCID: <https://orcid.org/0000-0003-0825-4956>),
Artur Sannikov [ctb]

Maintainer Vilhelm Suksi <vksuks@utu.fi>

Contents

`manhattan_plot` *Manhattan plot*

Description

Draws a (directed) Manhattan plot of p-values and versus e.g. retention time or mass-to-charge ratio. If effect size and direction is supplied, the - log10(p-value) on the y-axis will be multiplied by the direction (sign) of the effect, so part of the points will "drop" from the p = 1 (-log10(p) = 0) line. This results in a so-called directed Manhattan plot.

Usage

```
manhattan_plot(
  object,
  x,
  p,
  effect = NULL,
  p_fdr = NULL,
  color = NULL,
  p_breaks = c(0.05, 0.01, 0.001, 1e-04),
  fdr_limit = 0.05,
  x_lim = NULL,
  y_lim = NULL,
  color_scale = getOption("notame.color_scale_con"),
  title = "Manhattan plot",
  subtitle = NULL,
  ...
)
```

Arguments

`object`	a `SummarizedExperiment` object or a data frame like object. Feature data is used.
`x, p`	the column names of x-axis and p-values
`effect`	column name of effect size (should have negative and positive values).
`p_fdr`	column name of FDR corrected p-values, used to draw a line showing the fdr-corrected significance level
`color`	column name used to color the plots
`p_breaks`	a numerical vector of the p_values to show on the y-axis
`fdr_limit`	the significance level used in the experiment
`x_lim, y_lim`	numerical vectors of length 2 for manually setting the axis limits
`color_scale`	the color scale as returned by a ggplot function
`title, subtitle`	the title and subtitle of the plot
`...`	parameters passed to `geom_point`, such as shape and alpha values. New aesthetics can also be passed using `mapping = aes(...)`.

Value

A ggplot object.

Examples

```
data(toy_notame_set, package = "notame")
# naturally, this looks messy as there are not enough p-values
lm_results <- notameStats::perform_lm(notame::drop_qcs(toy_notame_set),
  formula_char = "Feature ~ Group")
lm_data <- dplyr::left_join(as.data.frame(rowData(toy_notame_set)),
  lm_results)
# Traditional Manhattan plot from data frame
manhattan_plot(lm_data,
  x = "Average_Mz",
  p = "GroupB.p.value", p_fdr = "GroupB.p.value_FDR",
  fdr_limit = 0.1
)
# Directed Manhattan plot from SummarizedExperiment
with_results <- notame::join_rowData(toy_notame_set, lm_results)
manhattan_plot(with_results,
  x = "Average_Mz", effect = "GroupB.estimate",
  p = "GroupB.p.value", p_fdr = "GroupB.p.value_FDR",
  fdr_limit = 0.1
)
```

`mz_rt_plot` *Plot m/z vs retention time plot (cloud plot)*

Description

Plots a scatter plot of results of statistical tests, where each point represents a feature. The plot has retention time on x-axis, m/z on y-axis and the size of the points is scaled based on p-value

Usage

```
mz_rt_plot(
  object,
  p_col = NULL,
  p_limit = NULL,
  mz_col = NULL,
  rt_col = NULL,
  color = NULL,
  title = "m/z vs retention time",
  subtitle = NULL,
  color_scale = getOption("notame.color_scale_con"),
  all_features = FALSE,
  ...
)
```

Arguments

`object`	a `SummarizedExperiment` object or a data frame. Feature data is used. If x is a data frame, it is used as is.
`p_col`	the column name containing p-values. This is used to scale the size of the points.
`p_limit`	numeric, limits plotted features by p-values. If NULL, plots all features.
`mz_col`, `rt_col`	the column names for m/z and retention time. If NULL, automatic detection is attempted.
`color`	the column name used to color the points
`title`	The plot title
`subtitle`	The plot subtitle
`color_scale`	color scale as returned by a ggplot function. Defaults to current continuous color scale.
`all_features`	logical, should all features be retained? Should be used only if x is a SummarizedExperiment object.
`...`	parameters passed to `geom_point`, such as shape and alpha values. New aesthetics can also be passed using `mapping = aes(...)`.

Value

A ggplot object.

Examples

```
data(toy_notame_set, package = "notame")
# Compute results from a linear model
lm_results <- notameStats::perform_lm(toy_notame_set,
  formula_char = "Feature ~ Group")
with_results <- notame::join_rowData(toy_notame_set, lm_results)

# Plot from the SummarizedExperiment object
# automatically facet by analytical mode in variable Split
mz_rt_plot(with_results, p_col = "GroupB.p.value", color = "GroupB.estimate")

# Plot the results from the results dataframe
lm_data <- dplyr::left_join(as.data.frame(rowData(toy_notame_set)),
                            lm_results)
mz_rt_plot(lm_data, p_col = "GroupB.p.value", color = "GroupB.estimate")
```

`plot_dendrogram` *Sample dendrogram*

Description

Draws a dendrogram of a hierarchical clustering applied to the samples of an experiment.

Usage

```
plot_dendrogram(
  object,
  all_features = FALSE,
  color,
  dist_method = "euclidean",
  clust_method = "ward.D2",
  center = TRUE,
  scale = "uv",
  title = "Dendrogram of hierarchical clustering",
  subtitle = NULL,
  color_scale = getOption("notame.color_scale_dis"),
  assay.type = NULL
)
```

Arguments

`object`	a `SummarizedExperiment` object
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`color`	character, name of the column used for coloring the sample labels
`dist_method`	distance method used in clustering as in `dist`
`clust_method`	method used in clustering as in `hclust`
`center`	logical, should the data be centered?
`scale`	scaling used, as in `prep`. Default is "uv" for unit variance
`title`	The plot title
`subtitle`	The plot subtitle
`color_scale`	the color scale as returned by a ggplot function.
`assay.type`	character, assay to be used in case of multiple assays

Value

A ggplot object.

See Also

`dist` `hclust`

Examples

```
data(toy_notame_set, package = "notame")
plot_dendrogram(toy_notame_set, color = "Group")
```

`plot_dist_density` *Plot distance density*

Description

Plot density of distances between samples in QC samples and actual samples.

Usage

```
plot_dist_density(
  object,
  all_features = FALSE,
  dist_method = "euclidean",
  center = TRUE,
  scale = "uv",
  color_scale = getOption("notame.color_scale_dis"),
  fill_scale = getOption("notame.fill_scale_dis"),
  title = paste("Density plot of", dist_method, "distances between samples"),
  subtitle = NULL,
  assay.type = NULL
)
```

Arguments

`object`	a `SummarizedExperiment` object
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`dist_method`	method for calculating the distances, passed to `dist`
`center`	logical, should the data be centered?
`scale`	scaling used, as in `prep` Default is "uv" for unit variance
`color_scale`	a scale for the color of the edge of density curves, as returned by a ggplot function
`fill_scale`	a scale for the fill of the density curves, as returned by a ggplot function
`title`	the plot title
`subtitle`	the plot subtitle
`assay.type`	character, assay to be used in case of multiple assays

Value

A ggplot object.

See Also

`dist`

Examples

```
data(toy_notame_set, package = "notame")
plot_dist_density(toy_notame_set)
# Drift correction tightens QCs together
plot_dist_density(notame::correct_drift(toy_notame_set))
```

`plot_effect_heatmap` *Heatmap of effects between variables, such as correlations*

Description

Draws a heatmap of e.g. correlations between variables (see perform_correlation_tests). It is possible to draw only the lower triangular of the heatmap, order rows and columns with hierarchical clustering, and add circles for p-values.

Usage

```
plot_effect_heatmap(
  data,
  x,
  y,
  effect,
  p = NULL,
  p_limit = 0.1,
  point_size_range = c(1, 6),
  log2_effect = FALSE,
  discretize_effect = FALSE,
  breaks = 5,
  clustering = TRUE,
  dist_method = "euclidean",
  clust_method = "ward.D2",
  lower_tri = FALSE,
  reverse_y = TRUE,
  use_coord_fixed = TRUE,
  symmetric_aspect_ratio = TRUE,
  title = NULL,
  subtitle = NULL,
  fill_scale = NA
)
```

Arguments

`data`	a data frame with x and y variables and the effect
`x, y`	the column names of data with the x and y variables
`effect`	the column name of the effect, e.g. correlation

`p`	optional, the column name with p-values. If provided, points that scale by p-value are drawn on top of the heatmap tiles
`p_limit`	numeric, only p-values below the limit are plotted as points
`point_size_range`	a numeric vector of length 2. The upper and lower limits for the point sizes. This needs to be adjusted to make the point size look good when compared to the tiles
`log2_effect`	logical, whether the effect should be plotted on a logarithmic scale (in case of fold change etc.)
`discretize_effect`	logical, whether the effect range should be divided into discrete levels instead of using a continuous scale. Can sometimes make patterns more visible, but the hard limits can blur the big picture as well.
`breaks`	if `discretize_effect` = `TRUE`, either the number of breaks or the points where to cut for the levels, see `cut`
`clustering`	logical, whether the order of rows and columns should be ordered by hierarchical clustering?
`dist_method`	distance method used in clustering, see `dist`
`clust_method`	clustering method used in clustering, see `hclust`
`lower_tri`	logical, should only the lower triangular be plotted?
`reverse_y`	logical, if `clustering` = `FALSE`, `lower_tri` = `FALSE`, should the order of the y-axis be reversed so that the diagonal is from top left to bottom right?
`use_coord_fixed`	logical, should the heatmap tiles be squares? If yes, this uses `coord_fixed`
`symmetric_aspect_ratio`	logical, should the plot panel be a square? If yes, uses ggplot2::theme(aspect.ratio = 1).
`title`, `subtitle`	the title and subtitle of the plot
`fill_scale`	fill scale for the heatmap as returned by a ggplot function. Set to NA to choose the appropriate scale based on the class of the effect variable.

Details

All missing effects between variables are replaced by 0 before clustering, since `hclust` can't deal with missing values.

Value

A ggplot object.

See Also

`cut` for discretizing the effect, `dist` for distance calculation for clustering, `hclust` for hierarchical clustering.

Examples

```
data(toy_notame_set, package = "notame")
# Compute correlations between variables
correlations <- notameStats::perform_correlation_tests(toy_notame_set,
  x = rownames(toy_notame_set),
  duplicates = TRUE
)

# Minimal example
plot_effect_heatmap(correlations,
x = "X", y = "Y", effect = "Correlation_coefficient")

# Lower triangular with discrete effect and p-value dots
plot_effect_heatmap(correlations,
  x = "X", y = "Y", effect = "Correlation_coefficient",
  p = "Correlation_P", point_size_range = c(2, 8),
  discretize_effect = TRUE, breaks = 7, lower_tri = TRUE
)
```

`plot_injection_lm` *Estimate the magnitude of drift*

Description

Plots histograms of p-values from linear regression model, where each feature is predicted by injection order alone. The expected uniform distribution is represented by a dashed red line.

Usage

```
plot_injection_lm(object, all_features = FALSE, assay.type = NULL)
```

Arguments

`object`	a `SummarizedExperiment` object
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`assay.type`	character, assay to be used in case of multiple assays

Value

A ggplot object.

See Also

`plot_p_histogram`

Examples

```
data(toy_notame_set, package = "notame")
plot_injection_lm(toy_notame_set)
```

`plot_pca` *PCA scatter plot*

Description

Computes PCA using one of the methods provided in the Bioconductor package pcaMethods and plots the two first principal components.

Usage

```
plot_pca(
  object,
  pcs = c(1, 2),
  all_features = FALSE,
  center = TRUE,
  scale = "uv",
  color = NULL,
  shape = color,
  label = NULL,
  density = FALSE,
  title = "PCA",
  subtitle = NULL,
  color_scale = NA,
  shape_scale = getOption("notame.shape_scale"),
  fill_scale = getOption("notame.fill_scale_dis"),
  text_base_size = 14,
  point_size = 2,
  assay.type = NULL,
  ...
)
```

Arguments

`object`	a `SummarizedExperiment` object
`pcs`	numeric vector of length 2, the principal components to plot
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`center`	logical, should the data be centered prior to PCA? (usually yes)
`scale`	scaling used, as in `prep`. Default is "uv" for unit variance
`color`	character, name of the column used for coloring the points. Set to NULL for black color.

`shape`	character, name of the column used for shape. Set to NULL for uniform round shapes.
`label`	character, name of the column used for point labels
`density`	logical, whether to include density plots to both axes. The density curves will be split and colored by the 'color' variable.
`title`, `subtitle`	the titles of the plot
`color_scale`	the color scale as returned by a ggplot function. Set to NA to choose the appropriate scale based on the class of the coloring variable.
`shape_scale`	the shape scale as returned by a ggplot function
`fill_scale`	the fill scale used for density curves. If a continuous variable is used as color, density curve will be colorless.
`text_base_size`	numeric, base size for text
`point_size`	numeric, size of the points
`assay.type`	character, assay to be used in case of multiple assays
`...`	additional arguments passed to `pca`

Value

A ggplot object. If `density` is `TRUE`, the plot will consist of multiple parts and is harder to modify.

See Also

`pca`

Examples

```
data(toy_notame_set, package = "notame")
plot_pca(toy_notame_set, color = "Injection_order", shape = "Group")
```

`plot_pca_arrows` *PCA plot with arrows*

Description

Plots changes in PCA space according to time. All the observations of a single subject are connected by an arrow ending at the last observation.

Usage

```
plot_pca_arrows(
  object,
  pcs = c(1, 2),
  all_features = FALSE,
  center = TRUE,
  scale = "uv",
  color,
  time,
  subject,
  alpha = 0.6,
  arrow_style = arrow(),
  title = "PCA changes",
  subtitle = NULL,
  color_scale = getOption("notame.color_scale_dis"),
  text_base_size = 14,
  line_width = 0.5,
  assay.type = NULL,
  ...
)
```

Arguments

`object`	a `SummarizedExperiment` object
`pcs`	numeric vector of length 2, the principal components to plot
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`center`	logical, should the data be centered prior to PCA? (usually yes)
`scale`	scaling used, as in `prep`. Default is "uv" for unit variance
`color`	character, name of the column used for coloring the arrows
`time`	character, name of the column containing timepoints
`subject`	character, name of the column containing subject identifiers
`alpha`	numeric, value for the alpha parameter of the arrows (transparency)
`arrow_style`	a description of arrow heads, the size and angle can be modified, see `?arrow`
`title, subtitle`	the titles of the plot
`color_scale`	the color scale as returned by a ggplot function
`text_base_size`	the base size of the text
`line_width`	the width of the arrows
`assay.type`	character, assay to be used in case of multiple assays
`...`	additional arguments passed to `pca`

Value

A ggplot object.

See Also

pca

Examples

```
data(toy_notame_set, package = "notame")
plot_pca_arrows(notame::drop_qcs(toy_notame_set), color = "Group",
  time = "Time", subject = "Subject_ID")
# If the sample size is large, plot groups separately
plot_pca_arrows(notame::drop_qcs(toy_notame_set), color = "Group",
                time = "Time", subject = "Subject_ID") +
  facet_wrap(~Group)
```

`plot_pca_hexbin` *PCA hexbin plot*

Description

Computes PCA using one of the methods provided in the Bioconductor package pcaMethods and plots the two first principal components as hexagonal bins, where the value of the coloring variable is summarised for each bin, by default as the mean of the values inside the bin.

Usage

```
plot_pca_hexbin(
  object,
  pcs = c(1, 2),
  all_features = FALSE,
  center = TRUE,
  scale = "uv",
  fill = "Injection_order",
  summary_fun = "mean",
  bins = 10,
  title = "PCA",
  subtitle = NULL,
  fill_scale = getOption("notame.fill_scale_con"),
  assay.type = NULL,
  ...
)
```

Arguments

`object`	a SummarizedExperiment object
`pcs`	numeric vector of length 2, the principal components to plot
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.

`center`	logical, should the data be centered prior to PCA? (usually yes)
`scale`	scaling used, as in `prep`. Default is "uv" for unit variance
`fill`	character, name of the column used for coloring the hexagons
`summary_fun`	the function used to compute the value for each hexagon
`bins`	the number of bins in x and y axes
`title, subtitle`	the titles of the plot
`fill_scale`	the fill scale as returned by a ggplot function
`assay.type`	character, assay to be used in case of multiple assays
`...`	additional arguments passed to `pca`

Value

A ggplot object.

See Also

`pca`

Examples

```
data(toy_notame_set, package = "notame")
plot_pca_hexbin(toy_notame_set)
```

`plot_pca_loadings` *PCA loadings plot*

Description

Computes PCA using one of the methods provided in the Bioconductor package pcaMethods and plots the loadings of first principal components.

Usage

```
plot_pca_loadings(
  object,
  pcs = c(1, 2),
  all_features = FALSE,
  center = TRUE,
  scale = "uv",
  n_features = c(10, 10),
  title = "PCA loadings",
  subtitle = NULL,
  text_base_size = 14,
  point_size = 2,
```

```
    label_text_size = 4,
    assay.type = NULL,
    ...
  )
```

Arguments

`object`	a `SummarizedExperiment` object
`pcs`	numeric vector of length 2, the principal components to plot
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`center`	logical, should the data be centered prior to PCA? (usually yes)
`scale`	scaling used, as in `prep`. Default is "uv" for unit variance
`n_features`	numeric vector of length two, number of top feature to plot for each principal component
`title, subtitle`	the titles of the plot
`text_base_size`	numeric, base size for text
`point_size`	numeric, size of the points
`label_text_size`	numeric, size of the labels
`assay.type`	character, assay to be used in case of multiple assays
`...`	additional arguments passed to `prep`

Value

A ggplot object.

See Also

`pca`

Examples

```
data(toy_notame_set, package = "notame")
plot_pca_loadings(toy_notame_set, n_features = c(2, 4))
```

`plot_p_histogram` *Histogram of p-values*

Description

Draws histograms of p-values with expected uniform distribution represented by a dashed red line.

Usage

```
plot_p_histogram(p_values, hline = TRUE, combine = TRUE, x_label = "p-value")
```

Arguments

`p_values`	list or data frame, each element/column is a vector of p- values. The list names are used as plot titles
`hline`	logical, whether a horizontal line representing uniform distribution should be plotted
`combine`	logical, whether plots of individual p-value vectors should be combined into a single object. Set to FALSE if you want to add other plots to the list before plotting
`x_label`	the x-axis label

Value

If combine = TRUE, a ggplot object. Otherwise a list of ggplot objects.

Examples

```
data(toy_notame_set, package = "notame")
lm_sample <- notameStats::perform_lm(notame::drop_qcs(toy_notame_set),
  "Feature ~ Injection_order")
p_values <- list("Biological samples" = lm_sample$Injection_order.p.value)
plot_p_histogram(p_values)
```

`plot_quality` *Plot quality metrics*

Description

Plots distribution of each quality metric, and a distribution of the flags.

Usage

```
plot_quality(
  object,
  all_features = FALSE,
  plot_flags = TRUE,
  assay.type = NULL
)
```

Arguments

`object`	a `SummarizedExperiment` object
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`plot_flags`	logical, should the distribution of flags be added as a barplot?
`assay.type`	character, assay to be used in case of multiple assays and no quality metrics are present in feature data

Value

A ggplot object.

Examples

```
data(toy_notame_set, package = "notame")
plot_quality(toy_notame_set)
```

`plot_sample_boxplots` *Plot a boxplot for each sample*

Description

Plots a boxplot of the distribution of the metabolite values for each sample. The boxplots can be ordered and filled by any combination of columns in the pheno data. By default, order and fill are both determined by the combination of group and time columns.

Usage

```
plot_sample_boxplots(
  object,
  all_features = FALSE,
  order_by,
  fill_by,
  title = "Boxplot of samples",
  subtitle = NULL,
  fill_scale = getOption("notame.fill_scale_dis"),
```

```
  zoom_boxplot = TRUE,
  assay.type = NULL
)
```

Arguments

`object`	a `SummarizedExperiment` object
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`order_by`	character vector, names of columns used to order the samples
`fill_by`	character vector, names of columns used to fill the boxplots
`title, subtitle`	character, title and subtitle of the plot
`fill_scale`	a scale for the fill of the boxplots, as returned by a ggplot function
`zoom_boxplot`	logical, whether outliers should be left outside the plot and only the boxplots shown. Defaults to TRUE.
`assay.type`	character, assay to be used in case of multiple assays

Value

A ggplot object.

Examples

```
data(toy_notame_set, package = "notame")
plot_sample_boxplots(toy_notame_set, order_by = "Group", fill_by = "Group")
```

`plot_sample_heatmap` *Sample heatmap*

Description

Draws a heatmap of the distances between the samples of an experiment, the samples are ordered by hierarchical clustering.

Usage

```
plot_sample_heatmap(
  object,
  all_features = FALSE,
  dist_method = "euclidean",
  clust_method = "ward.D2",
  center = TRUE,
  scale = "uv",
  group_bar = TRUE,
  group = NULL,
```

```
  title = "Heatmap of distances between samples",
  subtitle = NULL,
  fill_scale_con = getOption("notame.fill_scale_con"),
  fill_scale_dis = getOption("notame.fill_scale_dis"),
  assay.type = NULL
)
```

Arguments

`object`	a `SummarizedExperiment` object
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`dist_method`	distance method used in clustering as in `dist`
`clust_method`	method used in clustering as in `hclust`
`center`	logical, should the data be centered?
`scale`	scaling used, as in `prep`. Default is "uv" for unit variance
`group_bar`	logical, should a bar showing the groups be drawn under the heat map?
`group`	character, name of the column used for coloring the group bar
`title`	The plot title
`subtitle`	The plot subtitle
`fill_scale_con`	Continuous fill scale for the heatmap as returned by a ggplot function
`fill_scale_dis`	Discrete fill scale for the group bar as returned by a ggplot function
`assay.type`	character, assay to be used in case of multiple assays

Value

A ggplot object. If `group_bar` is `TRUE`, the plot will consist of multiple parts and is harder to modify.

See Also

`dist` `hclust`

Examples

```
data(toy_notame_set, package = "notame")
plot_sample_heatmap(toy_notame_set, group = "Group")
```

`plot_tsne` *t-SNE scatter plot*

Description

Computes t-SNE into two dimensions and plots the map points. In case there are missing values, PCA is performed using the nipals method of `pca`, the method can be changed to "ppca" if nipals fails.

Usage

```
plot_tsne(
  object,
  all_features = FALSE,
  center = TRUE,
  scale = "uv",
  perplexity = 30,
  pca_method = "nipals",
  color = NULL,
  shape = color,
  label = NULL,
  density = FALSE,
  title = "t-SNE",
  subtitle = paste("Perplexity:", perplexity),
  color_scale = NA,
  shape_scale = getOption("notame.shape_scale"),
  fill_scale = getOption("notame.fill_scale_dis"),
  text_base_size = 14,
  point_size = 2,
  assay.type = NULL,
  ...
)
```

Arguments

`object`	a `SummarizedExperiment` object
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`center`	logical, should the data be centered prior to PCA? (usually yes)
`scale`	scaling used, as in `prep`. Default is '"uv" for unit variance
`perplexity`	the perplexity used in t-SNE
`pca_method`	the method used in PCA if there are missing values
`color`	character, name of the column used for coloring the points. Set to NULL for black color.

`shape`	character, name of the column used for shape. Set to NULL for uniform round shapes.
`label`	character, name of the column used for point labels
`density`	logical, whether to include density plots to both axes. The density curves will be split and colored by the 'color' variable.
`title, subtitle`	the titles of the plot
`color_scale`	the color scale as returned by a ggplot function. Set to NA to choose the appropriate scale based on the class of the coloring variable.
`shape_scale`	the shape scale as returned by a ggplot function
`fill_scale`	the fill scale used for density curves. If a continuous variable is used as color, density curve will be colorless.
`text_base_size`	numeric, base size for text
`point_size`	numeric, size of the points
`assay.type`	character, assay to be used in case of multiple assays
`...`	additional arguments passed to `Rtsne`

Value

A ggplot object. If `density` is `TRUE`, the plot will consist of multiple parts and is harder to modify.

See Also

`Rtsne`

Examples

```
data(toy_notame_set, package = "notame")
plot_tsne(toy_notame_set, color = "Time", shape = "Group", perplexity = 10)
```

`plot_tsne_arrows` *t-SNE plot with arrows*

Description

Computes t-SNE into two dimensions and plots changes according to time. All the observations of a single subject are connected by an arrow ending at the last observation. In case there are missing values, PCA is performed using the nipals method of `pca`, the method can be changed to "ppca" if nipals fails.

Usage

```
plot_tsne_arrows(
  object,
  all_features = FALSE,
  center = TRUE,
  scale = "uv",
  perplexity = 30,
  pca_method = "nipals",
  color,
  time,
  subject,
  alpha = 0.6,
  arrow_style = arrow(),
  title = "t-SNE changes",
  subtitle = paste("Perplexity:", perplexity),
  color_scale = getOption("notame.color_scale_dis"),
  text_base_size = 14,
  line_width = 0.5,
  assay.type = NULL,
  ...
)
```

Arguments

`object`	a SummarizedExperiment object
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`center`	logical, should the data be centered prior to PCA? (usually yes)
`scale`	scaling used, as in `prep`. Default is "uv" for unit variance
`perplexity`	the perplexity used in t-SNE
`pca_method`	the method used in PCA if there are missing values
`color`	character, name of the column used for coloring the points
`time`	character, name of the column containing timepoints
`subject`	character, name of the column containing subject identifiers
`alpha`	numeric, value for the alpha parameter of the arrows (transparency)
`arrow_style`	a description of arrow heads, the size and angle can be modified, see `?arrow`
`title, subtitle`	the titles of the plot
`color_scale`	the color scale as returned by a ggplot function
`text_base_size`	the base size of the text
`line_width`	the width of the arrows
`assay.type`	character, assay to be used in case of multiple assays
`...`	additional arguments passed to `Rtsne`

Value

A ggplot object. If `density` is `TRUE`, the plot will consist of multiple parts and is harder to modify.

See Also

`Rtsne`

Examples

```
data(toy_notame_set, package = "notame")
plot_tsne_arrows(notame::drop_qcs(toy_notame_set), perplexity = 10,
  color = "Group", time = "Time", subject = "Subject_ID")
# If the sample size is large, plot groups separately
plot_tsne_arrows(notame::drop_qcs(toy_notame_set), perplexity = 10,
  color = "Group", time = "Time", subject = "Subject_ID") +
    facet_wrap(~Group)
```

`plot_tsne_hexbin` *t-SNE hexbin plot*

Description

Computes t-SNE into two dimensions and plots the map as hexagonal bins, where the value of the coloring variable is summarised for each bin, by default as the mean of the values inside the bin. In case there are missing values, PCA is performed using the nipals method of `pca`, the method can be changed to "ppca" if nipals fails.

Usage

```
plot_tsne_hexbin(
  object,
  all_features = FALSE,
  center = TRUE,
  scale = "uv",
  pca_method = "nipals",
  perplexity = 30,
  fill = "Injection_order",
  summary_fun = "mean",
  bins = 10,
  title = "t-SNE",
  subtitle = paste("Perplexity:", perplexity),
  fill_scale = getOption("notame.fill_scale_con"),
  assay.type = NULL,
  ...
)
```

Arguments

`object`	a SummarizedExperiment object
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`center`	logical, should the data be centered prior to PCA? (usually yes)
`scale`	scaling used, as in `prep`. Default is "uv" for unit variance
`pca_method`	the method used in PCA if there are missing values
`perplexity`	the perplexity used in t-SNE
`fill`	character, name of the column used for coloring the hexagons
`summary_fun`	the function used to compute the value for each hexagon
`bins`	the number of bins in x and y axes
`title, subtitle`	the titles of the plot
`fill_scale`	the fill scale as returned by a ggplot function
`assay.type`	character, assay to be used in case of multiple assays
`...`	additional arguments passed to `Rtsne`

Value

A ggplot object.

See Also

`Rtsne`

Examples

```
data(toy_notame_set, package = "notame")
plot_tsne_hexbin(toy_notame_set, perplexity = 10)
```

`save_batch_plots` *Save batch correction plots*

Description

Saves plots of each feature showing the effect of batch correction. Plots show QC samples and regular samples inside each batch, plus the batch mean for biological samples and QC samples as a horizontal line. The dashed line represents QC mean, the filled line represents biological sample mean. NOTE: if you change the shape variable, be sure to set a shape scale as well, the default scale only has 2 values, so it can only accomodate 2 shapes.

Usage

```
save_batch_plots(
  orig,
  corrected,
  file,
  save = TRUE,
  width = 14,
  height = 10,
  batch = "Batch",
  color = "Batch",
  shape = "QC",
  color_scale = getOption("notame.color_scale_dis"),
  shape_scale = scale_shape_manual(values = c(15, 21)),
  assay.type1 = NULL,
  assay.type2 = NULL
)
```

Arguments

`orig, corrected`	`SummarizedExperiment` objects before and after batch effect correction
`file`	path to the PDF file where the plots will be saved
`save`	logical, if false, the plots are not saved but returned as a list
`width, height`	width and height of the plots in inches
`batch, color, shape`	column names of pheno data for batch labels, and column used for coloring and shaping points (by default batch and QC)
`color_scale, shape_scale`	scales for color and scale as returned by ggplot functions.
`assay.type1`	character, assay of orig to be used in case of multiple assays.
`assay.type2`	character, assay of corrected to be used in case of multiple assays. If corrected is not supplied, this argument selects another assay from orig.

Value

None, the function is invoked for its plot-saving side effect.

Examples

```
data(toy_notame_set, package = "notame")
# Batch correction
batch_corrected <- batchCorr::normalizeBatches(toy_notame_set,
  assay.type = 1, batches = "Batch", sampleGroup = "Group", refGroup = "QC",
  population = "all", name = "normalized")
# Plots of each feature
save_batch_plots(
  orig = toy_notame_set[1:10], corrected = batch_corrected[1:10],
  file = "batch_plots.pdf", assay.type2 = "normalized"
```

```
)
```

`save_beeswarm_plots` *Save beeswarm plots of each feature by group*

Description

Draws a beeswarm plot of feature abundances in each group. A separate plot is drawn and saved for each feature.

Usage

```
save_beeswarm_plots(
  object,
  all_features = FALSE,
  save = TRUE,
  file_path = NULL,
  format = "emf",
  x,
  add_boxplots = FALSE,
  title = "Feature_ID",
  subtitle = NULL,
  color,
  color_scale = getOption("notame.color_scale_dis"),
  text_base_size = 14,
  cex = 2,
  size = 2,
  title_line_length = 40,
  theme = theme_bw(base_size = text_base_size),
  assay.type = NULL,
  ...
)
```

Arguments

`object`	a `SummarizedExperiment` object
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`save`	logical, if false, the plots are not saved but returned as a list
`file_path`	character, a file path for PDF or prefix added to the file paths for other formats
`format`	character, format in which the plots should be saved
`x`	character, name of the column to be used as x-axis
`add_boxplots`	logical, should boxplots be added to the figure?

`title, subtitle`	column names from feature data to use as plot title/filename and subtitle. Set to NULL for no title/subtitle, this creates running numbered filenames
`color`	character, name of the column to be used for coloring
`color_scale`	the color scale as returned by a ggplot function
`text_base_size`	integer, base size for text in figures
`cex`	numeric, scaling for adjusting point spacing
`size`	numeric, size of points
`title_line_length`	integer, maximum length of the title line in characters, passed to `str_wrap`
`theme`	a ggplot theme to be added to the plot
`assay.type`	character, assay to be used in case of multiple assays
`...`	other arguments to graphic device functions, like width and height

Value

By default, the function is invoked for its plot-saving side effect. The function returns a list of plots when `save = FALSE`.

See Also

`save_plot`

Examples

```
data(toy_notame_set, package = "notame")
# Default beeswarms by group
save_beeswarm_plots(notame::drop_qcs(toy_notame_set)[1:10],
  file_path = "./beeswarm_plots.pdf",
  format = "pdf", x = "Group", color = "Group"
)
# x and color can be a different variable
save_beeswarm_plots(notame::drop_qcs(toy_notame_set)[1:10],
  file_path = "./beeswarm_plots/",
  format = "png",
  x = "Time",
  color = "Group"
)

# Plot one feature
save_beeswarm_plots(notame::drop_qcs(toy_notame_set)[1, ], save = FALSE,
x = "Group", color = "Group")
```

`save_dc_plots` *Save drift correction plots*

Description

Plots the data before and after drift correction, with the regression line drawn with the original data. If the drift correction was done on log-transformed data, then plots of both the original and log-transformed data before and after correction are drawn. The plot shows 2 standard deviation spread for both QC samples and regular samples.

Usage

```
save_dc_plots(
  orig,
  dc,
  file,
  save = TRUE,
  log_transform = TRUE,
  width = 16,
  height = 8,
  color = "QC",
  shape = color,
  color_scale = getOption("notame.color_scale_dis"),
  shape_scale = scale_shape_manual(values = c(15, 16)),
  assay.orig = NULL,
  assay.dc = NULL
)
```

Arguments

`orig`	a SummarizedExperiment object with assay before drift correction
`dc`	a SummarizedExperiment object with assay after drift correction
`file`	path to the PDF file where the plots should be saved
`save`	logical, if false, the plots are not saved but returned as a list
`log_transform`	logical, was the drift correction done on log- transformed data?
`width, height`	width and height of the plots in inches
`color`	character, name of the column used for coloring the points
`shape`	character, name of the column used for shape
`color_scale`	the color scale as returned by a ggplot function
`shape_scale`	the shape scale as returned by a ggplot function
`assay.orig`	character, name of assay with abundances before correction
`assay.dc`	character, name of assay after correction

Details

By default, the column used for color is also used for shape.

Value

None, the function is invoked for its plot-saving side effect.

See Also

`correct_drift`

Examples

```
data(toy_notame_set, package = "notame")

toy_notame_set <- notame::mark_nas(toy_notame_set, value = 0)
dc <- notame::correct_drift(toy_notame_set, assay.type = 1,
                            name = "corrected")
save_dc_plots(toy_notame_set[1, ], dc[1, ],
  file = "drift_plots.pdf",
  assay.orig = 1, assay.dc = "corrected")
```

`save_group_boxplots` *Save box plots of each feature by group*

Description

Draws a boxplot of feature abundances in each group. A separate plot is drawn and saved for each feature.

Usage

```
save_group_boxplots(
  object,
  all_features = FALSE,
  save = TRUE,
  file_path = NULL,
  format = "emf",
  x,
  color,
  title = "Feature_ID",
  subtitle = NULL,
  color_scale = getOption("notame.color_scale_dis"),
  text_base_size = 14,
  box_width = 0.8,
  line_width = 0.5,
  point_size = 3,
```

```
  title_line_length = 40,
  theme = theme_bw(base_size = text_base_size),
  assay.type = NULL,
  ...
)
```

Arguments

`object`	a `SummarizedExperiment` object
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`save`	logical, if false, the plots are not saved but returned as a list
`file_path`	character, a file path for PDF or prefix added to the file paths for other formats
`format`	character, format in which the plots should be saved
`x`	character, name of the column to be used as x-axis
`color`	character, name of the column to be used for coloring
`title, subtitle`	column names from feature data to use as plot title/filename and subtitle. Set to NULL for no title/subtitle, this creates running numbered filenames
`color_scale`	the color scale as returned by a ggplot function
`text_base_size`	integer, base size for text in figures
`box_width`	numeric, width of the boxes
`line_width`	numeric, width of the lines
`point_size`	numeric, size of the mean points
`title_line_length`	integer, maximum length of the title line in characters, passed to `str_wrap`
`theme`	a ggplot theme to be added to the plot
`assay.type`	character, assay to be used in case of multiple assays
`...`	other arguments to graphic device functions, like width and height

Value

By default, the function is invoked for its plot-saving side effect. The function returns a list of plots when `save = FALSE`.

See Also

`save_plot`

Examples

```
data(toy_notame_set, package = "notame")
# Default boxplots by group
save_group_boxplots(notame::drop_qcs(toy_notame_set)[1:10],
  file_path = "./group_boxplots.pdf",
  format = "pdf", x = "Group", color = "Group"
```

```
)
# x and color can be a different variable
save_group_boxplots(notame::drop_qcs(toy_notame_set)[1:10],
  file_path = "./time_boxplots/",
  format = "emf",
  x = "Time",
  color = "Group"
)
# Plot one feature
save_group_boxplots(notame::drop_qcs(toy_notame_set)[1, ], save = FALSE,
  x = "Group", color = "Group")
```

`save_group_lineplots` *Save line plots with errorbars by group*

Description

Plots the change in the feature abundances as a function of e.g. time. A line is drawn for each group and error bars are added. A separate plot is drawn for each feature.

Usage

```
save_group_lineplots(
  object,
  all_features = FALSE,
  save = TRUE,
  file_path = NULL,
  format = "emf",
  x,
  group,
  title = "Feature_ID",
  subtitle = NULL,
  fun.data = "mean_cl_boot",
  fun = NULL,
  fun.min = NULL,
  fun.max = NULL,
  position_dodge_amount = 0.2,
  color_scale = getOption("notame.color_scale_dis"),
  text_base_size = 14,
  line_width = 0.5,
  point_size = 4,
  title_line_length = 40,
  theme = theme_bw(base_size = text_base_size),
  assay.type = NULL,
  ...
)
```

Arguments

`object`	a `SummarizedExperiment` object
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization
`save`	logical, if false, the plots are not saved but returned as a list
`file_path`	character, a file path for PDF or prefix added to the file paths for other formats
`format`	character, format in which the plots should be saved
`x`	character, name of the column to be used as x-axis
`group`	character, name of the column containing group information, used for coloring
`title, subtitle`	column names from feature data to use as plot title/filename and subtitle. Set to NULL for no title/subtitle, this creates running numbered filenames
`fun.data`	passed to `stat_summary` and used for errorbars, "A function that is given the complete data and should return a data frame with variables ymin, y, and ymax."
`fun.min, fun, fun.max`	Alternative to fun.data, passed to `stat_summary`, "supply three individual functions that are each passed a vector of x's and should return a single number"
`position_dodge_amount`	numeric: how much the group mean points should dodge away from each other
`color_scale`	the color scale as returned by a ggplot function
`text_base_size`	integer, base size for text in figures
`line_width`	numeric, width of the lines
`point_size`	numeric, size of the points
`title_line_length`	integer, maximum length of the title line in characters, passed to `str_wrap`
`theme`	a ggplot theme to be added to the plot
`assay.type`	character, assay to be used in case of multiple assays
`...`	other arguments to graphic device functions, like width and height

Value

By default, the function is invoked for its plot-saving side effect. The function returns a list of plots when `save = FALSE`.

See Also

`save_plot`, `stat_summary`

Examples

```
data(toy_notame_set, package = "notame")
save_group_lineplots(notame::drop_qcs(toy_notame_set)[1:10],
  file_path = "./group_line_plots.pdf",
  format = "pdf", x = "Time", group = "Group"
)
```

```
save_group_lineplots(notame::drop_qcs(toy_notame_set)[1:10],
  file_path = "./group_line_plots/",
  format = "png", x = "Time", group = "Group"
)
# Plot one feature
save_group_lineplots(notame::drop_qcs(toy_notame_set[1, ]), save = FALSE,
x = "Time", group = "Group")
```

`save_plot` *Save plot to chosen format*

Description

Saves the given plot to a file. Supports pdf, svg, emf, png and tiff formats. If an error occurs with the plot, an empty file is created.

Usage

```
save_plot(p, file, ...)
```

Arguments

`p`	a ggplot object
`file`	the file path
`...`	other arguments to plot function, like width and height

Value

None, the function is invoked for its plot-saving side effect.

See Also

`pdf`, `emf`, `svg`, `png`, `tiff`

Examples

```
data(toy_notame_set, package = "notame")

p <- plot_sample_heatmap(toy_notame_set, group = "Group")

save_plot(p, file = "test.pdf")
```

`save_QC_plots` *Write all relevant pretreatment visualizations to pdf*

Description

A wrapper around all the major visualization functions, used for visualizing data between major steps of data preprocessing. Saves all visualizations as PDFs with a set prefix on filenames.

Usage

```
save_QC_plots(
  object,
  prefix,
  format = "pdf",
  perplexity = 30,
  merge = FALSE,
  remove_singles = FALSE,
  group = NULL,
  time = NULL,
  id = NULL,
  color = NULL,
  assay.type = NULL
)
```

Arguments

`object`	a `SummarizedExperiment` object
`prefix`	character, a file path prefix added to the file paths
`format`	character, format in which the plots should be saved, DOES NOT support raster formats
`perplexity`	perplexity for t-SNE plots
`merge`	logical, whether the files should be merged to a single PDF, see Details
`remove_singles`	logical, whether to remove single plot files after merging. Only used if `merge = TRUE`
`group`	character, name of pheno data column containing the group labels
`time`	character, name of pheno data column containing timepoints
`id`	character, name of pheno data column containing subject identifiers
`color`	character, name of pheno data column used for coloring sample labels for dendrograms
`assay.type`	character, assay to be used in case of multiple assays

Details

If `merge` is `TRUE` and `format` is `pdf`, then a file containing all the visualizations named `prefix.pdf` will be created. NOTE: on Windows this requires installation of pdftk (`https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/`) and on Linux you need to have pdfunite installed. On MacOS, no external software is needed. Note that at least on Windows, prefix should be a path from the root, so that the underlying system command will find the files. The type of visualizations to be saved depends on the type of object. Here is a comprehensive list of the visualizations:

- Distribution of quality metrics and flags `plot_quality`
- Boxplots of each sample in injection order `plot_sample_boxplots`
- PCA scores plot of samples colored by injection order `plot_pca`
- t-SNE plot of samples colored by injection order `plot_tsne`
- If the object has over 60 samples, hexbin versions of the PCA and t- SNE plots above `plot_pca_hexbin`, `plot_tsne_hexbin`
- Dendrogram of samples ordered by hierarchical clustering, sample labels colored by group if present `plot_dendrogram`
- heat map of intersample distances, ordered by hierarchical clustering `plot_sample_heatmap`
- If the object has QC samples:
  - Density function of the intersample distances in both QCs and biological samples `plot_dist_density`
  - Histograms of p-values from linear regression of features against injection order in both QCs and biological samples `plot_p_histogram`
- If the object has a group column:
  - PCA and tSNE plots with points shaped and colored by group `plot_pca`, `plot_tsne`
- If the object has a time column:
  - PCA and tSNE plots with points shaped and colored by time '`plot_pca`, `plot_tsne`
  - Dendrogram of samples ordered by hierarchical clustering, sample labels colored by time point `plot_dendrogram`
- If the object has a group column OR a time column:
  - Boxplots of samples ordered and colored by group and/or time `plot_sample_boxplots`
- If the object has a group column AND a time column:
  - PCA and tSNE plots with points shaped by group and colored by time `plot_pca`, `plot_tsne`
- If the object has a time column AND a subject column:
  - PCA and tSNE plots with arrows connecting the samples of each subject in time point order `plot_pca_arrows`, `plot_tsne_arrows`

Value

None, the function is invoked for its plot-saving side effect.

See Also

`save_plot`

Examples

```
data(toy_notame_set, package = "notame")
rp_neg_set <- toy_notame_set[rowData(toy_notame_set)$Split == "RP_neg", ]
save_QC_plots(rp_neg_set, prefix="figures/RP_neg", perplexity=5,
              group = "Group", color = "Group", time = "Time",
              id = "Subject_ID")
```

`save_scatter_plots` *Save scatter plots of each feature against a set variable*

Description

Draws a scatterplots with a feature on y-axis and another variable on x-axis. A separate plot is drawn and saved for each feature.

Usage

```
save_scatter_plots(
  object,
  x = "Injection_order",
  save = TRUE,
  file_path = NULL,
  format = "emf",
  all_features = FALSE,
  color = NULL,
  color_scale = NA,
  shape = NULL,
  title = "Feature_ID",
  subtitle = NULL,
  shape_scale = getOption("notame.shape_scale"),
  text_base_size = 14,
  point_size = 2,
  title_line_length = 40,
  theme = theme_bw(base_size = text_base_size),
  assay.type = NULL,
  ...
)
```

Arguments

`object`	a `SummarizedExperiment` object
`x`	character, name of the column to be used as x-axis
`save`	logical, if false, the plots are not saved but returned as a list
`file_path`	character, a file path for PDF or prefix added to the file paths for other formats

`format`	character, format in which the plots should be saved
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`color`	character, name of the column to be used for coloring
`color_scale`	the color scale as returned by a ggplot function. Set to NA to choose the appropriate scale based on the class of the coloring variable.
`shape`	character, name of the column used for shape
`title`, `subtitle`	column names from feature data to use as plot title/filename and subtitle. Set to NULL for no title/subtitle, this creates running numbered filenames
`shape_scale`	the shape scale as returned by a ggplot function
`text_base_size`	integer, base size for text in figures
`point_size`	numeric, size of the points
`title_line_length`	integer, maximum length of the title line in characters, passed to `str_wrap`
`theme`	a ggplot theme to be added to the plot
`assay.type`	character, assay to be used in case of multiple assays
`...`	other arguments to graphic device functions, like width and height

Value

By default, the function is invoked for its plot-saving side effect. The function returns a list of plots when `save = FALSE`.

See Also

`save_plot`

Examples

```
data(toy_notame_set, package = "notame")
# Against injection order, colored by group
save_scatter_plots(
  object = toy_notame_set[1:10],
  x = "Injection_order",
  color = "Group",
  file_path = "./scatter_plots.pdf",
  format = "pdf"
)
# Plot one feature
save_scatter_plots(toy_notame_set[1, ], save = FALSE)
```

---

save_subject_line_plots

Save line plots with mean

Description

Plots the change in the feature abundances as a function of e.g. time. A line is drawn for each subject and a mean line is added. A separate plot is drawn and saved for each feature.

Usage

```
save_subject_line_plots(
  object,
  all_features = FALSE,
  save = TRUE,
  file_path = NULL,
  format = "emf",
  x,
  id,
  title = "Feature_ID",
  subtitle = NULL,
  color = NULL,
  color_scale = getOption("notame.color_scale_dis"),
  facet = NULL,
  text_base_size = 14,
  line_width = 0.3,
  mean_line_width = 1.2,
  title_line_length = 40,
  theme = theme_bw(base_size = text_base_size),
  assay.type = NULL,
  ...
)
```

Arguments

`object`	a `SummarizedExperiment` object
`all_features`	logical, should all features be used? If FALSE (the default), flagged features are removed before visualization.
`save`	logical, if false, the plots are not saved but returned as a list
`file_path`	character, a file path for PDF or prefix added to the file paths for other formats
`format`	character, format in which the plots should be saved
`x`	character, name of the column to be used as x-axis
`id`	character, name of the column containing subject IDs
`title, subtitle`	column names from feature data to use as plot title/filename and subtitle. Set to NULL for no title/subtitle, this creates running numbered filenames

`color`	character, the column name to color the lines by (optional)
`color_scale`	the color scale as returned by a ggplot function
`facet`	character, the column name to facet by (optional, usually same as color)
`text_base_size`	integer, base size for text in figures
`line_width`	numeric, width of the lines
`mean_line_width`	numeric, width of the mean line
`title_line_length`	integer, maximum length of the title line in characters, passed to `str_wrap`
`theme`	a ggplot theme to be added to the plot
`assay.type`	character, assay to be used in case of multiple assays
`...`	other arguments to graphic device functions, like width and height

Value

By default, the function is invoked for its plot-saving side effect. The function returns a list of plots when `save = FALSE`.

See Also

`save_plot`

Examples

```
data(toy_notame_set, package = "notame")
save_subject_line_plots(notame::drop_qcs(toy_notame_set)[1:10], x = "Time",
  id = "Subject_ID", file_path = "./subject_line_plots.pdf",
  format = "emf", title = NULL)

# Plot one feature
save_subject_line_plots(notame::drop_qcs(toy_notame_set[1, ]), save = FALSE,
  x = "Time", id = "Subject_ID")
```

`visualize_clusters` *Visualize clusters of features*

Description

Draws multiple visualizations of each cluster, creating a separate file for each cluster.

Usage

```
visualize_clusters(
  object,
  min_size = 3,
  rt_window = 1/60,
  n_clust_col = "Cluster_size",
  clust_col = "Cluster_features",
  mpa_col = "MPA",
  mz_col = NULL,
  rt_col = NULL
)
```

Arguments

`object`	a `SummarizedExperiment` object with clustering metadata
`min_size`	the minimum number of features a cluster needs to have to be plotted
`rt_window`	numeric, the retention time window to use in linking features. NOTE you need to use the same unit as in the retention time column
`n_clust_col`	character, name of the column that contains the features included in cluster, separated by semicolon
`clust_col`	character, name of the column that contains the features in a cluster
`mpa_col`	character, name of column that contains median peak area of features
`mz_col`	character, name of the column in features that contains mass-to-charge ratios
`rt_col`	character, name of the column in features that contains retention times

Details

Note that the input data has been assigned clusters but has not yet been compressed, for example by retaining the feature with the highest median peak area.

Value

A list with clusters containing two plots, a heatmap

Examples

```
data(toy_notame_set, package = "notame")
# The parameters are really weird because example data is imaginary
clustered <- notame::cluster_features(toy_notame_set, rt_window = 1,
                                      corr_thresh = 0.5, d_thresh = 0.6)

cluster_plots <- visualize_clusters(clustered, rt_window = 1)
```

`volcano_plot` *Volcano plot*

Description

Draws a volcano plot of effect size and p-values.

Usage

```
volcano_plot(
  object,
  x,
  p,
  p_fdr = NULL,
  color = NULL,
  p_breaks = c(0.05, 0.01, 0.001, 1e-04),
  fdr_limit = 0.05,
  log2_x = FALSE,
  center_x_axis = TRUE,
  x_lim = NULL,
  label = NULL,
  label_limit = 0.05,
  color_scale = getOption("notame.color_scale_con"),
  title = "Volcano plot",
  subtitle = NULL,
  text_base_size = 14,
  label_text_size = 4,
  ...
)
```

Arguments

`object`	a `SummarizedExperiment` object or a data frame. Feature data is used. If x is a data frame, it is used as is.
`x, p`	the column names of effect size (x-axis) and p-values
`p_fdr`	column name of FDR corrected p-values, used to draw a line showing the fdr-corrected significance level
`color`	column name used to color the plots
`p_breaks`	a numerical vector of the p_values to show on the y-axis
`fdr_limit`	the significance level used in the experiment
`log2_x`	logical, whether effect size should be plotted on a log2 axis.
`center_x_axis`	logical, whether x-axis should be centered. If `TRUE`, the "zero-effect" will be on the middle of the plot. The "zero effect" is 0 if `log2_x = FALSE` and 1 if `log2_x = TRUE`
`x_lim`	numerical vector of length 2 for manually setting the x-axis limits

`label`	column name used to label the plots
`label_limit`	numeric, p-value which is used to limit label plotting. Defaults to 0.05.
`color_scale`	the color scale as returned by a ggplot function
`title`, `subtitle`	the title and subtitle of the plot
`text_base_size`	integer, base size for text in figures
`label_text_size`	numeric, size of the labels
`...`	parameters passed to `geom_point`, such as shape and alpha values. New aesthetics can also be passed using `mapping = aes(...)`.

Value

A ggplot object.

Examples

```
data(toy_notame_set, package = "notame")
# naturally, this looks messy as there are not enough p-values
lm_results <- notameStats::perform_lm(notame::drop_qcs(toy_notame_set),
  formula_char = "Feature ~ Group")
volcano_plot(lm_results,
  x = "GroupB.estimate",
  p = "GroupB.p.value", p_fdr = "GroupB.p.value_FDR",
  label = "Feature_ID",
  fdr_limit = 0.1
)
```

Index