

Package: miaViz (via r-universe)

October 2, 2026

Title Microbiome Analysis Plotting and Visualization

Version 1.21.6

Description The miaViz package implements functions to visualize TreeSummarizedExperiment objects especially in the context of microbiome analysis. Part of the mia family of R/Bioconductor packages.

biocViews Microbiome, Software, Visualization

License Artistic-2.0 | file LICENSE

Encoding UTF-8

LazyData false

Depends R (>= 4.1), ggplot2, ggraph (>= 2.0), mia (>= 1.13.0), SummarizedExperiment, TreeSummarizedExperiment

Imports ape, BiocGenerics, BiocParallel, DelayedArray, DirichletMultinomial, dplyr, ggnewscale, ggrepel, ggtree, grid, methods, patchwork, rlang, S4Vectors, scales, scater, SingleCellExperiment, stats, tibble, tidygraph, tidyr, tidytext, tidytree, viridis

Suggests beeswarm, BiocStyle, bluster, circlize, ComplexHeatmap, ggh4x, ggpubr, knitr, maaslin3, mediation, miaTime, patchwork, rmarkdown, rstatix, shadowtext, testthat, vegan, vipor

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

URL <https://github.com/microbiome/miaViz>

BugReports <https://github.com/microbiome/miaViz/issues>

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libglpk-dev make libgsl0-dev libicu-dev libpng-dev libuv1-dev libxml2-dev zlib1g-dev

Repository <https://bioc.r-universe.dev>

Date/Publication 2026-10-02 05:56:50 UTC

RemoteUrl <https://github.com/bioc/miaViz>

RemoteRef HEAD
RemoteSha 7a584efdf3bfc07fe6fed12fd9e9e1aa75cea4d1

Contents

miaViz-package	2
deprecate	3
getNeatOrder	4
mia-datasets	6
mia-plot-args	6
plotAbundance	9
plotAbundanceDensity	11
plotBoxplot	15
plotCCA	19
plotColGraph	21
plotColTile	26
plotDMNFit	27
plotForest	28
plotHeatmap	32
plotHistogram	34
plotJointRPCA	37
plotLoadings	39
plotMediation	41
plotNMDS	42
plotOrdination	43
plotRowPrevalence	47
plotRowTree	50
plotScree	54
plotSeries	56
rowTreeData	58
Index	61

miaViz-package	<i>miaViz - Microbiome Analysis Plotting and Visualization</i>
----------------	--

Description

The scope of this package is the plotting and visualization of microbiome data. The main class for interfacing is the TreeSummarizedExperiment class.

Author(s)

Maintainer: Tuomas Borman <tuomas.v.borman@utu.fi> ([ORCID](#))

Authors:

- Tuomas Borman <tuomas.v.borman@utu.fi> ([ORCID](#))
- Felix G.M. Ernst ([ORCID](#))
- Leo Lahti <leo.lahti@iki.fi> ([ORCID](#))

Other contributors:

- Basil Courbayre [contributor]
- Giulio Benedetti ([ORCID](#)) [contributor]
- Théotime Pralas [contributor]
- Chouaib Benchraka [contributor]
- Sam Hillman [contributor]
- Geraldson Muluh [contributor]
- Noah De Gunst [contributor]
- Ely Seraidarian [contributor]
- Himmi Lindgren [contributor]
- Akewak Jeba ([ORCID](#)) [contributor]
- Vivian Ikeh [contributor]

See Also

[mia](#) class

deprecate	<i>These functions will be deprecated. Please use other functions instead.</i>
-----------	--

Description

These functions will be deprecated. Please use other functions instead.

Usage

```
plotTaxaPrevalence(x, ...)  
  
## S4 method for signature 'ANY'  
plotTaxaPrevalence(x, ...)  
  
plotFeaturePrevalence(x, ...)  
  
## S4 method for signature 'ANY'  
plotFeaturePrevalence(x, ...)
```

Arguments

x •
... •

getNeatOrder

Sorting by radial theta angle

Description

getNeatOrder sorts already ordinated data by the radial theta angle. This method is useful for organizing data points based on their angular position in a 2D space, typically after an ordination technique such as PCA or NMDS has been applied.

The function takes in a matrix of ordinated data, optionally centers the data using specified methods (mean, median, or NULL), and then calculates the angle (theta) for each point relative to the centroid. The data points are then sorted based on these theta values in ascending order.

One significant application of this sorting method is in plotting heatmaps. By using radial theta sorting, the relationships between data points can be preserved according to the ordination method's spatial configuration, rather than relying on hierarchical clustering, which may distort these relationships. This approach allows for a more faithful representation of the data's intrinsic structure as captured by the ordination process.

Usage

```
getNeatOrder(x, centering = "mean", ...)
```

```
## S4 method for signature 'matrix'
getNeatOrder(x, centering = "mean", ...)
```

Arguments

x	A matrix containing the ordinated data to be sorted. Columns should represent the principal components (PCs) and rows should represent the entities being analyzed (e.g. features or samples). There should be 2 columns only representing 2 PCs.
centering	Character scalar. Specifies the method to center the data. Options are "mean", "median", or NULL if your data is already centered. (Default: "mean")
...	Additional arguments passed to other methods.

Details

It's important to note that the **sechm** package does actually have the functionality for plotting a heatmap using this radial theta angle ordering, though only by using an MDS ordination.

That being said, the getNeatOrder function is more modular and separate to the plotting, and can be applied to any kind of ordinated data which can be valuable depending on the use case.

Rajaram & Oono (2010) NeatMap - non-clustering heat map alternatives in R outlines this in more detail.

Value

A character vector of row indices in the sorted order.

Examples

```
# Load the required libraries and dataset
library(mia)
library(scater)
library(ComplexHeatmap) |> suppressPackageStartupMessages()
library(circlize) |> suppressPackageStartupMessages()
data(peerj13075)

# Group data by taxonomic order
tse <- agglomerateByRank(peerj13075, rank = "order", onRankOnly = TRUE)

# Transform the samples into relative abundances using CLR
tse <- transformAssay(
  tse, assay.type = "counts", method="clr", MARGIN = "cols",
  name="clr", pseudocount = TRUE)

# Transform the features (taxa) into zero mean, unit variance
# (standardize transformation)
tse <- transformAssay(
  tse, assay.type="clr", method="standardize", MARGIN = "rows")

# Perform PCA using calculatePCA
res <- calculatePCA(tse, assay.type = "standardize", ncomponents = 10)

# Sort by radial theta and sort the original assay data
sorted_order <- getNeatOrder(res[, c(1,2)], centering = "mean")
tse <- tse[, sorted_order]

# Define the color function and cap the colors at [-5, 5]
col_fun <- colorRamp2(c(-5, 0, 5), c("blue", "white", "red"))

# Create the heatmap
heatmap <- Heatmap(assay(tse, "standardize"),
  name = "NeatMap",
  col = col_fun,
  cluster_rows = FALSE, # Do not cluster rows
  cluster_columns = FALSE, # Do not cluster columns
  show_row_dend = FALSE,
  show_column_dend = FALSE,
  row_names_gp = gpar(fontsize = 4),
  column_names_gp = gpar(fontsize = 6),
  heatmap_width = unit(20, "cm"),
  heatmap_height = unit(15, "cm")
)
```

 mia-datasets

 miaViz example data

Description

These example data objects were prepared to serve as examples. See the details for more information.

Usage

```
data(col_graph)
```

```
data(row_graph)
```

```
data(row_graph_order)
```

Format

An object of class `tbl_graph` (inherits from `igraph`) of length 26.

An object of class `tbl_graph` (inherits from `igraph`) of length 996.

An object of class `tbl_graph` (inherits from `igraph`) of length 110.

Details

For `*_graph` data:

1. “Jaccard” distances were calculated via `getDissimilarity(genus, FUN = vegan::vegdist, method = "jaccard", assay.type = "relabundance")`, either using transposed assay data or not to calculate distances for samples or features. NOTE: the function `mia::calculateDistance` is now deprecated.
2. “Jaccard” dissimilarities were converted to similarities and values above a threshold were used to construct a graph via `graph.adjacency(mode = "lower", weighted = TRUE)`.
3. The `igraph` object was converted to `tbl_graph` via `as_tbl_graph` from the `tidygraph` package.

 mia-plot-args

 Additional arguments for plotting

Description

To be able to fine tune plotting, several additional plotting arguments are available. These are described on this page.

Tree plotting

`line.alpha`: Numeric scalar in $[0, 1]$, Specifies the transparency of the tree edges. (Default: 1)

`line.width`: Numeric scalar. Specifies the default width of an edge. (Default: NULL) to use default of the ggtree package.

`line.width.range`: Numeric vector. The range for plotting dynamic edge widths in. (Default: `c(0.5, 3)`)

`point.alpha`: Numeric scalar in $[0, 1]$. Specifies the transparency of the tips. (Defaults: 1)

`point.size`: Numeric scalar. Specifies the default size of tips. (Defaults: 2)

`point.size.range`: Numeric vector. Specifies the range for plotting dynamic tip sizes in. (Defaults: `c(1, 4)`)

`label.font.size`: Numeric scalar. Font size for the tip and node labels. (Default: 3)

`highlight.font.size`: Numeric scalar. Font size for the highlight labels. (Default: 3)

Graph plotting

`line.alpha`: Numeric scalar in $[0, 1]$. Specifies the transparency of the tree edges. (Default: 1)

`line.width`: Numeric scalar. Specifies the default width of an edge. (Default: NULL) to use default of the ggtree package.

`line.width.range`: Numeric vector. The range for plotting dynamic edge widths in. (Default: `c(0.5, 3)`)

`point.alpha`: Numeric scalar in $[0, 1]$. Specifies the transparency of the tips. (Default: 1)

`point.size`: Numeric scalar. Specifies the default size of tips. (Default: 2.)

`point.size.range`: Numeric vector. The range for plotting dynamic tip sizes in. (Default: `c(1, 4)`)

Abundance plotting

`flipped`: Logical scalar. Should the plot be flipped? (Default: FALSE)

`add.legend`: Logical scalar. Should legends be plotted? (Default: TRUE)

`add.x.text`: Logical scalar. Should x tick labels be plotted? (Default: FALSE)

`add.border`: Logical scalar. Should border of bars be plotted? (Default: FALSE)

`bar.alpha`: Numeric scalar in $[0, 1]$. Specifies the transparency of the bars. (Default: 1)

`point.alpha`: Numeric scalar in $[0, 1]$. Specifies the transparency of the points. (Default: 1)

`point.size`: Numeric scalar. Specifies the default size of points. (Default: 2)

Abundance density plotting

`add.legend`: Logical scalar. Should legends be plotted? (Default: TRUE)

`point.shape`: Numeric scalar. Sets the shape of points. (Default: 21)

`point.colour`: Character scalar. Specifies the default colour of points. (Default: 2)

`point.size`: Numeric scalar. Specifies the default size of points. (Default: 2)
`point.alpha`: Numeric scalar in $[0, 1]$. Specifies the transparency of the points. (Default: 1)
`flipped`: Logical scalar. Should the plot be flipped? (Default: FALSE)
`scales.free`: Logical scalar. Should `scales = "free"` be set for faceted plots? (Default: TRUE)
`angle.x.text`: Logical scalar. Should x tick labels be plotted? (Default: FALSE)

Prevalence plotting

`flipped`: Logical scalar. Specifies whether the plot should be flipped. (Default: FALSE)
`add.legend`: Logical scalar. Should legends be plotted? (Default: TRUE)
`point.alpha`: Numeric scalar in $[0, 1]$. Specifies the transparency of the tips. (Default: 1)
`point.size`: Numeric scalar. Specifies the default size of tips. (Default: 2.)
`line.alpha`: Numeric scalar in $[0, 1]$. Specifies the transparency of the tree edges. (Default: 1)
`line.type`: Numeric scalar. Specifies the default line type. (Default: NULL) to use default of the ggplot2 package.
`line.size`: Numeric scalar. Specifies the default width of a line. (Default: NULL) to use default of the ggplot2 package.

Series plotting

`add.legend`: Logical scalar. Should legends be plotted? (Default: TRUE)
`line.alpha`: Numeric scalar in $[0, 1]$. Specifies the transparency of the tree edges. (Default: 1)
`line.type`: Numeric scalar. Specifies the default line type. (Default: NULL) to use default of the ggplot2 package.
`line.width`: Numeric scalar. Specifies the default width of a line. (Default: NULL) to use default of the ggplot2 package.
`line.width.range`: Numeric vector. The range for plotting dynamic line widths in. (Default: `c(0.5, 3)`)
`ribbon.alpha`: Numeric scalar in $[0, 1]$. Specifies the transparency of the ribbon. (Default: 0.3)

Tile plotting

`add.legend`: Logical scalar. Should legends be plotted? (Default: TRUE)
`rect.alpha`: Numeric scalar in $[0, 1]$. Specifies the transparency of the areas. (Default: 1)
`rect.colour`: Character scalar. Specifies the colour to use for colouring the borders of the areas. (Default: "black")
`na.value`: Character scalar. Specifies the colour to use for NA values. (Default: "grey80")

plotAbundance	<i>Plotting abundance data</i>
---------------	--------------------------------

Description

plotAbundance() creates a barplot of feature abundances, typically used to visualize the relative abundance of features at a specific taxonomy rank.

Usage

```
plotAbundance(x, ...)

## S4 method for signature 'SummarizedExperiment'
plotAbundance(
  x,
  assay.type = assay_name,
  assay_name = "counts",
  layout = "bar",
  ...
)
```

Arguments

- | | |
|-----|---|
| x | a SummarizedExperiment object. |
| ... | additional parameters for plotting. <ul style="list-style-type: none"> • group: Character scalar. Specifies the group for agglomeration. Must be a value from colnames(rowData(x)). If NULL, agglomeration is not applied. (Default: NULL) • as.relative: Character scalar. Should the relative values be calculated? (Default: FALSE) • col.var: Character scalar. Selects a column from colData to be plotted below the abundance plot. Continuous numeric values will be plotted as point, whereas factors and character will be plotted as colour-code bar. (Default: NULL) • order.row.by: Character scalar. How to order abundance value. By name ("name") for sorting the taxonomic labels alphabetically, by abundance ("abund") to sort by abundance values or by a reverse order of abundance values ("revabund"). (Default: "name") • row.levels: Character vector. Specifies order of rows in a plot. Can be combined with order.row.by to control order of only certain rows. If NULL, the order follows order.row.by. (Default: NULL) • order.col.by: Character scalar. from the chosen rank of abundance data or from colData to select values to order the abundance plot by. (Default: NULL) |

- `col.levels`: Character vector. Specifies order of columns in a plot. Can be combined with `order.col.by` to control order of only certain columns. If NULL, the order follows `order.col.by`. (Default: NULL)
- `decreasing`: Logical scalar. If the `order.col.by` is defined and the values are numeric, should the values used to order in decreasing or increasing fashion? (Default: FALSE)
- `facet.rows`: Logical scalar. Should the rows in the plot be spitted into facets? (Default: FALSE)
- `facet.cols`: Logical scalar. Should the columns in the plot be spitted into facets? (Default: TRUE)
- `ncol`: Numeric scalar. if facets are applied, `ncol` defines many columns should be for plotting the different facets. (Default: 2)
- `scales`: Character scalar. Defines the behavior of the scales of each facet. The value is passed into `facet_wrap`. (Default: "free_x")

See `mia-plot-args` for more details i.e. call `help("mia-plot-args")`

<code>assay.type</code>	Character scalar value defining which assay data to use. (Default: "relabundance")
<code>assay_name</code>	Deprecate. Use <code>assay.type</code> instead.
<code>layout</code>	Character scalar. Either "bar" or "point".

Details

It is recommended to handle subsetting, agglomeration, and transformation outside this function. However, agglomeration and relative transformation can be applied using the `group` and `as.relative` parameters, respectively. If one of the `TAXONOMY_RANKS` is selected via `group`, `mia::agglomerateByRank()` is used, otherwise `agglomerateByVariable()` is applied.

Value

a `ggplot` object or list of two `ggplot` objects, if `col.var` are added to the plot.

Examples

```
data(GlobalPatterns, package="mia")
tse <- GlobalPatterns

# If rank is set to NULL (default), agglomeration is not done. However, note
# that there is maximum number of rows that can be plotted. That is why
# we take sample from the data.
set.seed(26348)
sample <- sample(rownames(tse), 20)
tse_sub <- tse[sample, ]
# Apply relative transformation
tse_sub <- transformAssay(tse_sub, method = "relabundance")
plotAbundance(tse_sub, assay.type = "relabundance")

# Plotting counts using the first taxonomic rank as default
plotAbundance(
  tse, assay.type="counts", group = "Phylum") +
  labs(y="Counts")
```

```

# Using "Phylum" as rank. Apply relative transformation to "counts" assay.
plotAbundance(
  tse, assay.type="counts", group = "Phylum", add_legend = FALSE,
  as.relative = TRUE)

# Apply relative transform
tse <- transformAssay(tse, method = "relabundance")

# A feature from colData or taxon from chosen rank can be used for ordering
# samples.
plotAbundance(
  tse, assay.type="relabundance", group = "Phylum",
  order.col.by = "Bacteroidetes")

# col.var from colData can be plotted together with abundance plot.
# Returned object is a list that includes two plot; other visualizes
## abundance other col.var.
plot <- plotAbundance(
  tse, assay.type = "relabundance", group = "Phylum",
  col.var = "SampleType")

# Create a plot with sample grouping, and showing sample IDs as labels for
# the x axis on the top plot.
plot <- plotAbundance(
  tse, assay.type = "relabundance",
  group = "Phylum", col.var = "SampleType", add.legend = FALSE,
  add.x.text = TRUE) +
  theme(axis.text.x = element_text(angle = 90))
plot

# Compositional barplot with top 5 taxa and samples sorted by
# "Bacteroidetes"

# Getting top taxa on a Phylum level
tse <- transformAssay(tse, method = "relabundance")
tse_phylum <- agglomerateByRank(tse, rank = "Phylum")
top_taxa <- getTop(tse_phylum, top = 5, assay.type = "relabundance")

# Renaming the "Phylum" rank to keep only top taxa and the rest to "Other"
phylum_renamed <- lapply(rowData(tse)$Phylum, function(x){
  if (x %in% top_taxa) {x} else {"Other"}})
rowData(tse)$Phylum <- as.character(phylum_renamed)

# Compositional barplot
plotAbundance(
  tse, assay.type="relabundance", group = "Phylum",
  order.row.by="abund", order.col.by = "Bacteroidetes")

```

Description

This function plots abundance of the most abundant taxa.

Usage

```
plotAbundanceDensity(x, ...)

## S4 method for signature 'SummarizedExperiment'
plotAbundanceDensity(
  x,
  layout = c("jitter", "density", "point"),
  assay.type = assay_name,
  assay_name = "counts",
  n = min(nrow(x), 25L),
  colour.by = colour_by,
  colour_by = NULL,
  shape.by = shape_by,
  shape_by = NULL,
  size.by = size_by,
  size_by = NULL,
  decreasing = order_descending,
  order_descending = TRUE,
  ...
)
```

Arguments

- | | |
|-----|--|
| x | a SummarizedExperiment object. |
| ... | additional parameters for plotting. <ul style="list-style-type: none"> • xlab Character scalar. Selects the x-axis label. (Default: assay.type) • ylab Character scalar. Selects the y-axis label. ylab is disabled when layout = "density". (Default: "Taxa") • point.alpha Numeric scalar. From range 0 to 1. Selects the transparency of colour in jitter and point plot. (Default: 0.6) • point.shape Positive integer scalar. Value selecting the shape of point in jitter and point plot. (Default: 21) • point.size Positive integer scalar. Selects the size of point in jitter and point plot. (Default: 2) • add_legend Logical scalar. Determines if legend is added. (Default: TRUE) • flipped Logical scalar. Determines if the orientation of plot is changed so that x-axis and y-axis are swapped. (Default: FALSE) • add_x_text Logical scalar. Determines if text that represents values is included in x-axis. (Default: TRUE) • jitter.height Numeric scalar. Controls jitter in a jitter plot. (Default: 0.25) |

	• <code>jitter.width</code> Numeric scalar. Controls jitter in a jitter plot. (Default: NULL)
	See mia-plot-args for more details i.e. call <code>help("mia-plot-args")</code>
<code>layout</code>	Character scalar. Selects the layout of the plot. There are three different options: jitter, density, and point plot. (default: <code>layout = "jitter"</code>)
<code>assay.type</code>	Character scalar value defining which assay data to use. (Default: <code>"relabundance"</code>)
<code>assay_name</code>	Deprecated. Use <code>assay.type</code> instead.
<code>n</code>	Integer scalar. Specifies the number of the most abundant taxa to show. (Default: <code>min(nrow(x), 25L)</code>)
<code>colour.by</code>	Character scalar. Defines a column from <code>colData</code> , that is used to color plot. Must be a value of <code>colData()</code> function. (Default: NULL)
<code>colour_by</code>	Deprecated. Use <code>colour.by</code> instead.
<code>shape.by</code>	Character scalar. Defines a column from <code>colData</code> , that is used to group observations to different point shape groups. Must be a value of <code>colData()</code> function. <code>shape.by</code> is disabled when <code>layout = "density"</code> . (Default: NULL)
<code>shape_by</code>	Deprecated. Use <code>shape.by</code> instead.
<code>size.by</code>	Character scalar. Defines a column from <code>colData</code> , that is used to group observations to different point size groups. Must be a value of <code>colData()</code> function. <code>size.by</code> is disabled when <code>layout = "density"</code> . (Default: NULL)
<code>size_by</code>	Deprecated. Use <code>size.by</code> instead.
<code>decreasing</code>	Logical scalar. Indicates whether the results should be ordered in a descending order or not. If NA is given the order as found in <code>x</code> for the <code>n</code> most abundant taxa is used. (Default: TRUE)
<code>order_descending</code>	Deprecated. Use <code>order.descending</code> instead.

Details

This function plots abundance of the most abundant taxa. Abundance can be plotted as a jitter plot, a density plot, or a point plot. By default, x-axis represents abundance and y-axis taxa. In a jitter and point plot, each point represents abundance of individual taxa in individual sample. Most common abundances are shown as a higher density.

A density plot can be seen as a smoothened bar plot. It visualized distribution of abundances where peaks represent most common abundances.

Value

A `ggplot2` object

See Also

[scater::plotExpression](#)

Examples

```

data("peerj13075", package = "mia")
tse <- peerj13075

# Plots the abundances of 25 most abundant taxa. Jitter plot is the default
# option.
plotAbundanceDensity(tse, assay.type = "counts")

# Counts relative abundances
tse <- transformAssay(tse, method = "relabundance")

# Plots the relative abundance of 10 most abundant taxa.
# "nationality" information is used to color the points. X-axis is
# log-scaled.
plotAbundanceDensity(
  tse, layout = "jitter", assay.type = "relabundance", n = 10,
  colour.by = "Geographical_location") +
  scale_x_log10()

# Plots the relative abundance of 10 most abundant taxa as a density plot.
# X-axis is log-scaled
plotAbundanceDensity(
  tse, layout = "density", assay.type = "relabundance", n = 10 ) +
  scale_x_log10()

# Plots the relative abundance of 10 most abundant taxa as a point plot.
# Point shape is changed from default (21) to 41.
plotAbundanceDensity(
  tse, layout = "point", assay.type = "relabundance", n = 10,
  point.shape = 41)

# Plots the relative abundance of 10 most abundant taxa as a point plot.
# In addition to colour, groups can be visualized by size and shape in point
# plots, and adjusted for point size
plotAbundanceDensity(
  tse, layout = "point", assay.type = "relabundance", n = 10,
  shape.by = "Geographical_location", size.by = "Age", point.size=1)

# Ordering via decreasing
plotAbundanceDensity(
  tse, assay.type = "relabundance", decreasing = FALSE)

# for custom ordering set decreasing = NA and order the input object
# to your wishes
plotAbundanceDensity(
  tse, assay.type = "relabundance", decreasing = NA)

# Box plots and violin plots are supported by scater::plotExpression.
# Plots the relative abundance of 5 most abundant taxa as a violin plot.
library(scater)
top <- getTop(tse, top = 5)
plotExpression(tse, features = top, assay.type = "relabundance") +

```

```

ggplot2::coord_flip()

# Plots the relative abundance of 5 most abundant taxa as a box plot.
plotExpression(tse, features = top, assay.type = "relabundance",
  show_violin = FALSE, show_box = TRUE) + ggplot2::coord_flip()

```

plotBoxplot	Create boxplot of assay, rowData or colData.
-------------	--

Description

This methods visualizes abundances or variables from rowData or colData.

Usage

```

plotBoxplot(object, ...)

## S4 method for signature 'SummarizedExperiment'
plotBoxplot(
  object,
  assay.type = NULL,
  row.var = NULL,
  col.var = NULL,
  x = NULL,
  features = NULL,
  group.by = NULL,
  ...
)

```

Arguments

object	a SummarizedExperiment object.
...	Additional parameters for plotting. • <code>point.offset</code>: Character scalar. Utilized method for offsetting points. The available options include: "center", "compactswarm", "hex", "square", "swarm" (see beeswarm::beeswarm() for details), "frowney", "maxout", "minout", "pseudorandom", "quasirandom", "smiley", "tukey", "tukeyDense" (see vipor::offsetSingleGroup() for details), "jitter", and "none", If "none", ofsetting is not applied. (Default: "jitter") • <code>colour.by</code>: NULL or character scalar. Specifies a variable from <code>colData(x)</code> or <code>rowData(x)</code> which is used to colour observations. (Default: NULL) • <code>fill.by</code>: NULL or character scalar. Specifies a variable from <code>colData(x)</code> or <code>rowData(x)</code> which is used to colour observations. (Default: NULL) • <code>size.by</code>: NULL or character scalar. Specifies a variable from <code>colData(x)</code> or <code>rowData(x)</code> which is used to scale observation points. (Default: NULL)

- `shape.by`: NULL or character scalar. Specifies a variable from `colData(x)` or `rowData(x)` which is used to shape observation points. (Default: NULL)
 - `facet.by`: NULL or character scalar. Specifies a variable from `colData(x)` or `rowData(x)` which is used to facet or group observations. (Default: NULL)
 - `pair.by`: NULL or character scalar. Specifies a variable from `colData(x)` which is used to pair observation points. (Default: NULL)
 - `add.chance`: Logical scalar. Whether to visualize chance of paired observations by the color of line. (Default: FALSE)
 - `add.box`: Logical scalar. Whether to add a boxplot layout. (Default: TRUE)
 - `add.points`: Logical scalar. Whether to add a point layout. (Default: TRUE)
 - `add.proportion`: Logical scalar. Whether to add a barplot layout denoting the proportion of observations above threshold. (Default: FALSE)
 - `add.threshold`: Logical scalar. Whether to add a threshold as horizontal line when `add.proportion = TRUE` is specified. (Default: TRUE)
 - `threshold`: Numeric scalar. Specifies threshold for the barplots. (Default: 0)
 - `jitter.width`: Numeric scalar. Width of jitter. (Default: 0.3)
 - `jitter.height`: Numeric scalar. Height of jitter. (Default: 0)
 - `dodge.width`: Numeric scalar. Width of dodge. How far apart the groups are plotted? (Default: 0)
 - `beeswarm.corral`: Character scalar. Beeswarm's "corral" method. Fed to function `beeswarm::beeswarm()`. (Default: "none")
 - `scales`: Character scalar. Adjust scales of facets. (Default: "fixed")
 - `box.alpha`: Numeric scalar. Transparency of the boxplot layer. (Default: 0.5)
 - `point.alpha`: Numeric scalar. Transparency of the point layer. (Default: 0.65)
 - `line.alpha`: Numeric scalar. Transparency of the line layer. (Default: 0.5)
 - `point.shape`: Numeric scalar. Shape of points. (Default: 21)
 - `point.size`: Numeric scalar. Size of points. (Default: 2)
 - `point.colour`: Character scalar. Colour of points. (Default: "grey70")
 - `linetype`: Numeric scalar. Type of lines. (Default: 1)
 - `linewidth`: Numeric scalar. Width of lines. (Default: 1)
 - `line.colour`: Character scalar. Colour of lines. (Default: "grey70")
 - `box.width`: Numeric scalar. Width of boxes. (Default: 0.75)
 - `bar.width`: Numeric scalar. Width of proportion bars. By default, it is calculated based so that the width matches with the width of boxes.
- `assay.type` NULL or character scalar. Specifies the abundance table to plot. (Default: NULL)
- `row.var` NULL or character scalar. Specifies a variable from `rowData(x)` to visualize. (Default: NULL)

col.var	NULL or character scalar Specifies a variable from colData(x) to visualize. (Default: NULL)
x	NULL or character vector. Specifies a variable from colData(x) or rowData(x) to visualize in x axis. (Default: NULL)
features	NULL or character vector. If assay.type is specified, this specifies rows to visualize in different facets. If NULL, whole data is visualized as a whole. (Default: NULL)
group.by	NULL or character vector. Specifies a variable from colData(x) or rowData(x) to group observations. (Default: NULL)

Details

A box plot is standard visualization technique to compare numeric values, such as abundance, between categorical values, such as sample groups. `plotBoxplot()` streamlines creation of box plots, and it offers multiple options for visualization.

Value

A ggplot2 object.

See Also

- [scater::plotExpression](#)
- [scater::plotRowData](#)
- [scater::plotColData](#)

Examples

```
data("Tito2024QMP")
tse <- Tito2024QMP

tse <- transformAssay(tse, method = "relabundance")
tse <- addAlpha(tse, index = "shannon")

# Visualize alpha diversity
plotBoxplot(tse, col.var = "shannon", x = "diagnosis")

# Visualize relative abundance of top features
tse <- tse[getTop(tse, 6), ]

plotBoxplot(
  tse, assay.type = "relabundance",
  x = "diagnosis", fill.by = "diagnosis",
  features = rownames(tse), facet.by = "rownames",
  ncol = 2
)

# Add proportion bar
plotBoxplot(
  tse, assay.type = "relabundance",
```

```

    x = "diagnosis", fill.by = "diagnosis",
    features = rownames(tse), facet.by = "rownames",
    add.proportion = TRUE, threshold = 0.1
  )

# Visualize only with beeswarm
plotBoxplot(
  tse, assay.type = "relabundance",
  x = "diagnosis", group.by = "diagnosis",
  colour.by = "colonoscopy",
  features = rownames(tse), facet.by = "rownames",
  point.offset = "swarm", add.box = FALSE
)

# Do not add points
plotBoxplot(
  tse, assay.type = "relabundance",
  fill.by = "diagnosis",
  features = rownames(tse), facet.by = "rownames",
  add.points = FALSE
)

# Calculate statistical significance with Wilcoxon test
plotBoxplot(
  tse,
  col.var = "shannon",
  fill.by = "diagnosis",
  add.significance = TRUE
)

## Not run:
# Add pre-calculated p-values
# Calculate p-values
df <- meltSE(tse, add.col = TRUE)
pvals <- df |>
  rstatix::t_test(shannon ~ diagnosis) |>
  ungroup() |>
  as.data.frame()
# Add them with p.value argument
plotBoxplot(
  tse,
  x = "diagnosis",
  col.var = "shannon",
  p.value = pvals
)

## End(Not run)

## Not run:
library(microbiomeDataSets)

mae <- microbiomeDataSets::peerj32()
tse <- getWithColData(mae, 1)

```

```

tse[["time_point"]] <- as.character(tse[["time"]])
# Create a plot showing chance between time points in abundance of
# Akkermansia
plotBoxplot(
  tse, x = "time_point", assay.type = "counts", fill.by = "group",
  features = "Akkermansia", pair.by = "subject",
  add.chance = TRUE, scales = "free"
)

## End(Not run)

```

plotCCA

Plot RDA or CCA object

Description

plotRDA and plotCCA create an RDA/CCA plot starting from the output of [CCA](#) and [RDA](#) functions, two common methods for supervised ordination of microbiome data.

Usage

```

plotCCA(x, ...)

plotRDA(x, ...)

## S4 method for signature 'SingleCellExperiment'
plotCCA(x, dimred, ...)

## S4 method for signature 'SingleCellExperiment'
plotRDA(x, dimred, ...)

```

Arguments

x	a TreeSummarizedExperiment object.
...	additional parameters for plotting, inherited from plotReducedDim , geom_label and geom_label_repel . • <code>add.ellipse</code>: One of <code>c(TRUE, FALSE, "fill", "colour")</code>, indicating whether ellipses should be present, absent, filled or colored. (default: <code>ellipse.fill = TRUE</code>) • <code>ellipse.alpha</code>: Numeric scalar. Between 0 and 1. Adjusts the opacity of ellipses. (Default: 0.2) • <code>ellipse.linewidth</code>: Numeric scalar. Specifies the size of ellipses. (Default: 0.1) • <code>ellipse.linetype</code>: Integer scalar. Specifies the style of ellipses. (Default: 1)

- `confidence.level`: Numeric scalar. Between 0 and 1. Adjusts confidence level. (Default: 0.95)
- `add.vectors`: Logical scalar or character vector. If boolean, should vectors appear in the plot. If character, selects vectors that are showed. The matching is done with regular expression. (Default: TRUE)
- `vec.size`: Numeric scalar. Specifies the size of vectors. (Default: 0.5)
- `vec.colour`: Character scalar. Specifies the colour of vectors. (Default: "black")
- `vec.linetype`: Integer scalar. Specifies the style of vector lines. (Default: 1)
- `arrow.size`: Numeric scalar. Specifies the size of arrows. (Default: `arrow.size = 0.25`)
- `label.size`: Numeric scalar. Specifies the size of text and labels. (Default: 4)
- `label.colour`: Character scalar. Specifies the colour of text and labels. (Default: "black")
- `sep.group`: Character scalar. Specifies the separator used in the labels. (Default: "\U2014")
- `repl.underscore`: Character scalar. Used to replace underscores in the labels. (Default: " ")
- `vec.text`: Logical scalar. Should text instead of labels be used to label vectors. (Default: TRUE)
- `repel.labels`: Logical scalar. Should labels be repelled. (Default: TRUE)
- `parse.labels`: Logical scalar. Should labels be parsed. (Default: TRUE)
- `add.significance`: Logical scalar. Should explained variance and p-value appear in the labels. (Default: TRUE)
- `add.expl.var`: Logical scalar. Should explained variance appear on the coordinate axes. (Default: FALSE)
- `add.centroids`: Logical scalar. Should centroids of variables be added. (Default: FALSE)
- `add.species`: Logical scalar. Should species scores be added. (Default: FALSE)

`dimred` Character scalar or integer scalar. Determines the reduced dimension to plot. This is the output of `addRDA` and resides in `reducedDim(tse, dimred)`.

Details

`plotRDA` and `plotCCA` create an RDA/CCA plot starting from the output of `CCA` and `RDA` functions, two common methods for supervised ordination of microbiome data. Either a `TreeSummarizedExperiment` or a matrix object is supported as input. When the input is a `TreeSummarizedExperiment`, this should contain the output of `addRDA` in the `reducedDim` slot and the argument `dimred` needs to be defined. When the input is a matrix, this should be returned as output from `getRDA`. However, the first method is recommended because it provides the option to adjust aesthetics to the `colData` variables through the arguments inherited from `plotReducedDim`.

Value

A ggplot2 object

See Also

- [plotOrdination](#)

Examples

```
# Load dataset
data("enterotype", package = "mia")
tse <- enterotype

# Run RDA and store results into TreeSE
tse <- transformAssay(tse, method = "relabundance")
tse <- addRDA(
  tse,
  assay.type = "relabundance",
  formula = assay ~ ClinicalStatus + Gender + Age,
  distance = "bray",
  na.action = na.exclude
)

suppressWarnings({
# Create RDA plot coloured by variable
plotRDA(tse, "RDA", colour.by = "ClinicalStatus")

# Create RDA plot with ellipses
plotRDA(
  tse, "RDA", colour.by = "ClinicalStatus", fill.by = "ClinicalStatus",
  add.ellipse = TRUE
)

# Create RDA plot with text encased in labels
plotRDA(tse, "RDA", colour.by = "ClinicalStatus", vec.text = FALSE)

# Create RDA plot without repelling text
plotRDA(tse, "RDA", colour.by = "ClinicalStatus", repel.labels = FALSE)

# Create RDA plot without vectors
plotRDA(tse, "RDA", colour.by = "ClinicalStatus", add.vectors = FALSE)
})
```

Description

plotGraph plots an igraph object with additional information matched from a SummarizedExperiment object for the nodes only. Information on the edges have to provided manually.

Usage

```
plotColGraph(x, y, ...)
```

```
plotRowGraph(x, y, ...)
```

```
## S4 method for signature 'ANY,SummarizedExperiment'
plotColGraph(
  x,
  y,
  show.label = show_label,
  show_label = FALSE,
  add.legend = add_legend,
  add_legend = TRUE,
  layout = "kk",
  edge.type = edge_type,
  edge_type = c("fan", "link", "arc", "parallel"),
  edge.colour.by = edge_colour_by,
  edge_colour_by = NULL,
  edge.width.by = edge_width_by,
  edge_width_by = NULL,
  colour.by = colour_by,
  colour_by = NULL,
  shape.by = shape_by,
  shape_by = NULL,
  size.by = size_by,
  size_by = NULL,
  assay.type = by_exprs_values,
  by_exprs_values = "counts",
  other.fields = other_fields,
  other_fields = list(),
  ...
)

## S4 method for signature 'SummarizedExperiment,missing'
plotColGraph(x, y, name = "graph", ...)

## S4 method for signature 'ANY,SummarizedExperiment'
plotRowGraph(
  x,
  y,
  show.label = show_label,
  show_label = FALSE,
  add.legend = add_legend,
```

```

    add_legend = TRUE,
    layout = "kk",
    edge.type = edge_type,
    edge_type = c("fan", "link", "arc", "parallel"),
    edge.colour.by = edge_colour_by,
    edge_colour_by = NULL,
    edge.width.by = edge_width_by,
    edge_width_by = NULL,
    colour.by = colour_by,
    colour_by = NULL,
    shape.by = shape_by,
    shape_by = NULL,
    size.by = NULL,
    assay.type = by_exprs_values,
    by_exprs_values = "counts",
    other.fields = other_fields,
    other_fields = list(),
    ...
)

## S4 method for signature 'SummarizedExperiment,missing'
plotRowGraph(x, y, name = "graph", ...)

```

Arguments

<code>x, y</code>	a graph object and a SummarizedExperiment object or just a SummarizedExperiment . For the latter object a graph object must be stored in <code>metadata(x)\$name</code> .
<code>...</code>	additional arguments for plotting. See mia-plot-args for more details i.e. call <code>help("mia-plot-args")</code>
<code>show.label</code>	Logical scalar, integer vector or character vector If a logical scalar is given, should tip labels be plotted or if a logical vector is provided, which labels should be shown? If an integer or character vector is provided, it will be converted to a logical vector. The integer values must be in the range of 1 and number of nodes, whereas the values of a character vector must match values of a label or name column in the node data. In case of a character vector only values corresponding to actual labels will be plotted and if no labels are provided no labels will be shown. (Default: FALSE)
<code>show_label</code>	Deprecated. Use <code>show.label</code> instead.
<code>add.legend</code>	Logical scalar. Should legends be plotted? (Default: TRUE)
<code>add_legend</code>	Deprecated. Use <code>add.legend</code> instead.
<code>layout</code>	Character scalar. Layout for the plotted graph. See ggraph for details. (Default: "kk")
<code>edge.type</code>	Character scalar. Type of edge plotted on the graph. See geom_edge_fan for details and other available geoms. (Default: "fan")
<code>edge_type</code>	Deprecated. Use <code>edge.type</code> instead.
<code>edge.colour.by</code>	Character scalar. Specification of an edge metadata field to use for setting colours of the edges. (Default: NULL)

edge_colour_by	Deprecated. Use <code>edge.colour.by</code> instead.
edge.width.by	Character scalar. Specification of an edge metadata field to use for setting width of the edges. (Default: NULL)
edge_width_by	Deprecated. Use <code>edge.width.by</code> instead.
colour.by	Character scalar. Specification of a column metadata field or a feature to colour graph nodes by, see the <code>by</code> argument in ?retrieveCellInfo for possible values. (Default: NULL)
colour_by	Deprecated. Use <code>colour.by</code> instead.
shape.by	Character scalar. Specification of a column metadata field or a feature to shape graph nodes by, see the <code>by</code> argument in ?retrieveCellInfo for possible values. (Default: NULL)
shape_by	Deprecated. Use <code>shape.by</code> instead.
size.by	Character scalar. Specification of a column metadata field or a feature to size graph nodes by, see the <code>by</code> argument in ?retrieveCellInfo for possible values. (Default: NULL)
size_by	Deprecated. Use <code>size.by</code> instead.
assay.type	Character scalar. or integer scalar. Specifies which assay to obtain expression values from, for use in point aesthetics - see the <code>exprs_values</code> argument in ?retrieveCellInfo . (Default: "counts")
by_exprs_values	Deprecated. Use <code>assay.type</code> instead.
other.fields	Additional fields to include in the node information without plotting them.
other_fields	Deprecated. Use <code>other.fields</code> instead.
name	Character scalar. If <code>x</code> is a SummarizedExperiment the key for subsetting the <code>metadata(x)</code> to a graph object. (Default: "graph")

Details

Internally `tidygraph` and `ggraph` are used. Therefore, all graph types which can be converted by `tidygraph::as_tbl_graph` can be used.

Value

a [ggtree](#) plot

Examples

```
# data setup
library(mia)
data(GlobalPatterns)
data(col_graph)
data(row_graph)
data(row_graph_order)
metadata(GlobalPatterns)$col_graph <- col_graph

genus <- agglomerateByRank(GlobalPatterns, "Genus", na.rm=TRUE)
```

```

metadata(genus)$row_graph <- row_graph
order <- agglomerateByRank(genus,"Order",na.rm=TRUE)
metadata(order)$row_graph <- row_graph_order

# plot a graph independently
plotColGraph(col_graph,
             genus,
             colour.by = "SampleType",
             edge.colour.by = "weight",
             edge.width.by = "weight",
             show.label = TRUE)

# plot the graph stored in the object
plotColGraph(genus,
             name = "col_graph",
             colour.by = "SampleType",
             edge.colour.by = "weight",
             edge.width.by = "weight")

# plot a graph independently
plotRowGraph(row_graph,
             genus,
             colour.by = "Kingdom",
             edge.colour.by = "weight",
             edge.width.by = "weight")

# plot the graph stored in the object
plotRowGraph(genus,
             name = "row_graph",
             colour.by = "Phylum",
             edge.colour.by = "weight",
             edge.width.by = "weight")

# plot a graph independently
plotRowGraph(row_graph_order,
             order,
             colour.by = "Kingdom",
             edge.colour.by = "weight",
             edge.width.by = "weight")

# plot the graph stored in the object and include some labels
plotRowGraph(order,
             name = "row_graph",
             colour.by = "Phylum",
             edge.colour.by = "weight",
             edge.width.by = "weight",
             show.label = c("Sulfolobales","Spirochaetales",
                           "Verrucomicrobiales"))

# labels can also be included via selecting specific rownames of x/y
plotRowGraph(order,

```

```

        name = "row_graph",
        colour.by = "Phylum",
        edge.colour.by = "weight",
        edge.width.by = "weight",
        show.label = c(1,10,50))

# labels can also be included via a logical vector, which has the same length
# as nodes are present
label_select <- rep(FALSE,nrow(order))
label_select[c(1,10,50)] <- TRUE
plotRowGraph(order,
             name = "row_graph",
             colour.by = "Phylum",
             edge.colour.by = "weight",
             edge.width.by = "weight",
             show.label = label_select)

```

plotColTile	<i>Plot factor data as tiles</i>
-------------	----------------------------------

Description

Relative relations of two grouping can be visualized by plotting tiles with relative sizes. plotColTile and plotRowTile can be used for this.

Usage

```

plotColTile(object, x, y, ...)

plotRowTile(object, x, y, ...)

## S4 method for signature 'SummarizedExperiment'
plotColTile(object, x, y, ...)

## S4 method for signature 'SummarizedExperiment'
plotRowTile(object, x, y, ...)

```

Arguments

object	a SummarizedExperiment object.
x	Character scalar. Specifies the column-level metadata field to show on the x-axis. Alternatively, an AsIs vector or data.frame, see ?retrieveFeatureInfo or ?retrieveCellInfo . Must result in a returned character or factor vector.
y	Character scalar. Specifies the column-level metadata to show on the y-axis. Alternatively, an AsIs vector or data.frame, see ?retrieveFeatureInfo or ?retrieveCellInfo . Must result in a returned character or factor vector.
...	additional arguments for plotting. See mia-plot-args for more details i.e. call <code>help("mia-plot-args")</code>

Value

A ggplot2 object or plotly object, if more than one prevalences was defined.

Examples

```
data(GlobalPatterns)
se <- GlobalPatterns
plotColTile(se, "SampleType", "Primer")
```

plotDMNFit

Plotting Dirichlet-Multinomial Mixture Model data

Description

To plot DMN fits generated with mia use plotDMNFit.

Usage

```
plotDMNFit(x, name = "DMN", type = c("laplace", "AIC", "BIC"), ...)

## S4 method for signature 'SummarizedExperiment'
plotDMNFit(x, name = "DMN", type = c("laplace", "AIC", "BIC"))
```

Arguments

x	a SummarizedExperiment object contain the DMN data in metadata.
name	Character scalar. The name to store the result in metadata (Default: "DMN")
type	Character scalar. The type of measure for access the goodness of fit. One of 'laplace', 'AIC' or 'BIC'.
...	optional arguments not used.

Value

plotDMNFit returns a ggplot2 plot.

See Also

[calculateDMN](#)

Examples

```

library(mia)
library(bluster)

# Get dataset
data("peerj13075", package = "mia")
tse <- peerj13075

# Cluster the samples
tse <- addCluster(
  tse,
  assay.type = "counts",
  DmmParam(k = 1:4),
  name = "DMM",
  full = TRUE
)

# Plot the fit
plotDMNFit(tse, name = "DMM", type = "laplace")

```

plotForest

Visualize estimated results with forest plots

Description

plotForest() creates a feature- or sample-wise forest plot, showing estimated results from a statistical test with their confidence intervals. Additionally, the plot can be enriched with the tree structure and labelled with Confidence Intervals (CIs), p-values and other side information.

Usage

```

plotForest(x, ...)

## S4 method for signature 'TreeSummarizedExperiment'
plotForest(
  x,
  by = 1L,
  effect.var = "effect",
  ci.lower.var = "lower",
  ci.upper.var = "upper",
  err.var = NULL,
  pval.var = "pval",
  id.var = "rownames",
  label.by = NULL,
  order.by = NULL,
  facet.by = NULL,
  color.by = colour.by,

```

```

    colour.by = NULL,
    conf.level = 0.95,
    tree.name = "phylo",
    show.tree = TRUE,
    ...
)

## S4 method for signature 'SummarizedExperiment'
plotForest(
  x,
  by = 1L,
  effect.var = "effect",
  ci.lower.var = "lower",
  ci.upper.var = "upper",
  err.var = NULL,
  pval.var = "pval",
  id.var = "rownames",
  label.by = NULL,
  order.by = NULL,
  facet.by = NULL,
  color.by = colour.by,
  colour.by = NULL,
  conf.level = 0.95
)

## S4 method for signature 'data.frame'
plotForest(
  x,
  effect.var = "effect",
  ci.lower.var = "lower",
  ci.upper.var = "upper",
  err.var = NULL,
  pval.var = "pval",
  id.var = "rownames",
  label.by = NULL,
  order.by = NULL,
  facet.by = NULL,
  color.by = colour.by,
  colour.by = NULL,
  conf.level = 0.95
)

```

Arguments

x	a SummarizedExperiment object, or a <code>data.frame</code> object containing statistical estimates.
...	additional parameters passed to plotRowTree .
by	Character scalar. Determines whether features or samples data is used for the

	plot. (Default: "rows")
effect.var	Character scalar. Specifies the variable of x which corresponds to the effects or estimated results. (Default: "effect")
ci.lower.var	Character scalar. Specifies the variable of x which corresponds to the lower CI boundaries. (Default: "lower")
ci.upper.var	Character scalar. Specifies the variable of x which corresponds to the upper CI boundaries. (Default: "upper")
err.var	Character scalar. Specifies the variable of x which corresponds to the standard errors associated with effect.var. When defined, it overwrites ci.lower.var and ci.upper.var. (Default: "pval")
pval.var	Character scalar. Specifies the variable of x which corresponds to the p-values associated with effect.var. (Default: "pval")
id.var	Character scalar. Specifies the variable of x which corresponds to the observation identifiers. When "rownames"), the object rownames are used. (Default: "rownames")
label.by	Character vector. Specifies the variables of x or row/colData(x) by which the plot should be labelled. "CI" and "P-Value" are special entries which require either effect.var, ci.lower.var and ci.upper.var or pval.var to be specified, respectively. (Default: NULL)
order.by	Character scalar. Specifies the variable of x by which observations are ordered. If NULL, the observations are ordered by the tree structure if available. (Default: NULL)
facet.by	Character scalar. Specifies the variable of x by which observations are divided into horizontal facets. (Default: NULL)
color.by	Character scalar. Alias to colour.by.
colour.by	Character scalar. Specifies the variable of x by which observations are coloured. (Default: NULL)
conf.level	Numeric scalar. Specifies the confidence level of the interval when inferred from err.var. It is ignored when ci.lower.var and ci.upper.var are defined. (Default: 0.95)
tree.name	Character scalar. Specifies a row/colTree from x. (Default: "phylo")
show.tree	Logical scalar. Should the tree structure of the data be shown next to the forest plot?

Value

a [ggplot](#) object.

Examples

```
library(mia)
library(maaslin3)

# Import dataset
data("Tengeler2020", package = "mia")
```

```

tse <- Tengeler2020

# Agglomerate by genus and subset by prevalence
tse <- subsetByPrevalent(tse, rank = "Genus", prevalence = 10/100)

# Transform count assay to relative abundances
tse <- transformAssay(tse, assay.type = "counts", method = "relabundance")

# Run maaslin3
maaslin3_out <- maaslin3(
  input_data = tse,
  output = "maaslin_results",
  formula = "~ patient_status",
)

# Retrieve abundance results
maaslin3_abund <- maaslin3_out$fit_data_abundance$results
maaslin3_abund <- maaslin3_abund[!is.na(maaslin3_abund$coef), ]

# Visualize abundance results
plotForest(
  maaslin3_abund,
  effect.var = "coef",
  err.var = "stderr",
  pval.var = "qval_joint",
  id.var = "feature",
  label.by = c("CI", "P-Value"),
  order.by = "coef"
)

# Add abundance results to TreeSE rowData
rownames(maaslin3_abund) <- maaslin3_abund$feature
tax_order <- match(rownames(tse), rownames(maaslin3_abund))
rowData(tse) <- cbind(rowData(tse), maaslin3_abund[tax_order, ])

# Visualise abundance results with tree structure
plotForest(
  tse,
  effect.var = "coef",
  err.var = "stderr",
  pval.var = "qval_joint",
  label.by = c("CI", "P-Value")
)

# Retrieve prevalence results
maaslin3_prev <- maaslin3_out$fit_data_prevalence$results
maaslin3_prev <- maaslin3_prev[!is.na(maaslin3_prev$coef), ]

# Combine abundance and prevalence results
maaslin3_res <- rbind(maaslin3_abund, maaslin3_prev)

maaslin3_res$association <- c(
  rep("Abundance", nrow(maaslin3_abund)),

```

```

    rep("Prevalence", nrow(maaslin3_prev))
  )

  # Visualize combined results
  plotForest(
    maaslin3_res,
    effect.var = "coef",
    err.var = "stderr",
    pval.var = "qval_joint",
    id.var = "feature",
    label.by = c("CI", "P-Value"),
    order.by = "coef",
    facet.by = "association",
    colour.by = "association"
  )

```

plotHeatmap

Visualize assay values as a heatmap

Description

Creates a heatmap from an assay stored in a SummarizedExperiment or TreeSummarizedExperiment. Features are shown as rows and samples as columns.

Usage

```

plotHeatmap(x, ...)

## S4 method for signature 'TreeSummarizedExperiment'
plotHeatmap(
  x,
  assay.type = NULL,
  row.var = NULL,
  col.var = NULL,
  scale = FALSE,
  center = FALSE,
  tree.name = "phylo",
  show.tree = FALSE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
plotHeatmap(
  x,
  assay.type = NULL,
  row.var = NULL,
  col.var = NULL,

```

```

    scale = FALSE,
    center = FALSE,
    ...
  )

```

Arguments

<code>x</code>	a SummarizedExperiment object.
<code>...</code>	Additional parameters controlling the visualization. When <code>show.tree = TRUE</code> , additional arguments are passed to plotRowTree to control the appearance of the tree. Additional parameters include, for example: <code>scales</code>: Facet scaling passed to <code>ggplot2::facet_grid()</code> or <code>ggh4x::facet_nested()</code>. (Default: "free") <code>tree.width</code>: Relative width of the tree panel when <code>show.tree = TRUE</code>. (Default: 0.2)
<code>assay.type</code>	Character scalar value defining which assay data to use. (Default: "relabundance")
<code>row.var</code>	NULL or character vector. Variables from <code>rowData(x)</code> used for row faceting. (Default: NULL)
<code>col.var</code>	NULL or character vector. Variables from <code>colData(x)</code> used for column faceting. (Default: NULL)
<code>scale</code>	Logical scalar. Should assay values be scaled for each feature across samples? (Default: FALSE)
<code>center</code>	Logical scalar. Should assay values be centered for each feature across samples? (Default: FALSE)
<code>tree.name</code>	Character scalar. Name of the row tree to display when <code>x</code> is a <code>TreeSummarizedExperiment</code> . (Default: "phylo")
<code>show.tree</code>	Logical scalar. Should the row tree be displayed? Only available for <code>TreeSummarizedExperiment</code> . (Default: TRUE)

Details

`plotHeatmap` visualizes values from an assay as a heatmap. Values can be optionally centered and/or scaled across samples for each feature, which is useful for highlighting relative abundance patterns rather than absolute abundances.

Additional variables from `rowData(x)` and `colData(x)` can be used to facet the heatmap. When multiple row or column variables are provided, nested facets are created using **ggh4x**.

For `TreeSummarizedExperiment` objects, a row tree can be displayed alongside the heatmap. When a tree is shown, only leaf nodes are plotted and the heatmap rows are reordered to match the tree tip order.

Value

A `ggplot2` object. If `show.tree = TRUE`, the returned object is a combined patchwork object containing the row tree and the heatmap.

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

```
data(GlobalPatterns)
tse <- GlobalPatterns

tse <- agglomerateByPrevalence(tse, rank = "Class")

# Plot raw counts
plotHeatmap(
  tse,
  assay.type = "counts"
)

# Scale and center each feature
plotHeatmap(
  tse,
  assay.type = "counts",
  scale = TRUE,
  center = TRUE
)

# Facet samples by metadata
plotHeatmap(
  tse,
  assay.type = "counts",
  col.var = "SampleType"
)

# Add phylogeny
plotHeatmap(
  tse,
  assay.type = "counts",
  show.tree = TRUE
)
```

plotHistogram*Create histogram or barplot of assay, rowData or colData*

Description

This methods visualizes abundances or variables from rowData or colData.

Usage

```

plotHistogram(x, ...)

plotBarplot(x, ...)

## S4 method for signature 'SummarizedExperiment'
plotHistogram(
  x,
  assay.type = NULL,
  features = NULL,
  row.var = NULL,
  col.var = NULL,
  ...
)

## S4 method for signature 'SummarizedExperiment'
plotBarplot(
  x,
  assay.type = NULL,
  features = NULL,
  row.var = NULL,
  col.var = NULL,
  ...
)

```

Arguments

x	a SummarizedExperiment object.
...	Additional parameters for plotting. • layout: Character scalar. Specifies the layout of plot. Must be either "histogram" or "density". (Default: "histogram") • facet.by: Character vector. Specifies variables from colData(x) or rowData(x) used for facetting. (Default: NULL) • fill.by: Character scalar. Specifies variable from colData(x) or rowData(x) used for coloring. (Default: NULL)
assay.type	NULL or character scalar. Specifies the abundance table to plot. (Default: NULL)
features	NULL or character vector. If assay.type is specified, this specifies rows to visualize in different facets. If NULL, whole data is visualized as a whole. (Default: NULL)
row.var	NULL or character vector. Specifies a variable from rowData(x) to visualize. (Default: NULL)
col.var	NULL or character vector. Specifies a variable from colData(x) to visualize. (Default: NULL)

Details

Histogram and bar plot are a basic visualization techniques in quality control. It helps to visualize the distribution of data. `plotAbundance` allows researcher to visualise the abundance from assay, or variables from `rowData` or `colData`. For visualizing categorical values, one can utilize `plotBarplot`.

`plotAbundanceDensity` function is related to `plotHistogram`. However, the former visualizes the most prevalent features, while the latter can be used more freely to explore the distributions.

Value

A `ggplot2` object.

See Also

- `plotAbundanceDensity`
- `scater::plotExpression`
- `scater::plotRowData`
- `scater::plotColData`

Examples

```
data(GlobalPatterns)
tse <- GlobalPatterns

# Visualize the counts data. There are lots of zeroes.
plotHistogram(tse, assay.type = "counts")

# Apply transformation
tse <- transformAssay(tse, method = "clr", pseudocount = TRUE)
# And plot specified rows
plotHistogram(
  tse,
  assay.type = "clr",
  features = rownames(tse)[1:5],
  facet.by = "rownames"
)

# Calculate shannon diversity and visualize its distribution with density
# plot. Different sample types are separated with color.
tse <- addAlpha(tse, index = "shannon")
plotHistogram(
  tse,
  col.var = "shannon",
  layout = "density",
  fill.by = "SampleType"
)

# For categorical values, one can utilize a bar plot
plotBarplot(tse, col.var = "SampleType")
```

plotJointRPCA	<i>Visualize Joint-RPCA results</i>
---------------	-------------------------------------

Description

Creates a two-dimensional ordination plot from Joint-RPCA results stored in a `SingleCellExperiment` or `MultiAssayExperiment`. In addition to the sample ordination, feature loadings can be visualized as vectors.

Usage

```
plotJointRPCA(x, ...)

## S4 method for signature 'MultiAssayExperiment'
plotJointRPCA(x, dimred, ...)

## S4 method for signature 'SingleCellExperiment'
plotJointRPCA(x, dimred, add.vectors = TRUE, ...)
```

Arguments

<code>x</code>	A SummarizedExperiment or MultiAssayExperiment object containing Joint-RPCA results.
<code>...</code>	Additional arguments passed to plotOrdination and to the feature-vector plotting. Commonly used Joint-RPCA-specific arguments include: <code>ntop</code>: integer(1) or NULL. Maximum number of loading vectors shown per data layer. The vectors with the largest lengths are retained. Set to NULL to display all vectors. (Default: 10)
<code>dimred</code>	character(1): Specifies the name of the Joint-RPCA result to plot.
<code>add.vectors</code>	logical(1) or character. If TRUE, feature loading vectors are added. Alternatively, a character vector can be supplied to display only features whose names match the given pattern(s). (Default: TRUE)

Details

This function is a wrapper around [plotOrdination](#) for Joint-RPCA results. Consequently, most graphical parameters are passed directly to `plotOrdination()`, including options for colouring, grouping, faceting and adding ellipses. See [plotOrdination](#) for a complete description of these arguments.

Feature loadings are plotted as vectors originating from the origin. By default, only the longest loading vectors are shown for each data layer to improve readability.

Value

A `ggplot2` object.

See Also

- [plotOrdination](#)

Examples

```
data("HintikkaXOData")
mae <- HintikkaXOData

mae[[1]] <- transformAssay(
  mae[[1]],
  assay.type = "counts",
  method = "rclr",
  impute = FALSE
)
mae[[2]] <- transformAssay(
  mae[[2]],
  assay.type = "nmr",
  method = "log10"
)

mae <- addJointRPCA(
  mae,
  experiments = c(1, 2),
  assay.types = c("rclr", "log10")
)

# Basic Joint-RPCA plot
plotJointRPCA(mae, "JointRPCA")

# Colour samples by metadata
plotJointRPCA(mae, "JointRPCA", colour.by = "Fat")

# Add confidence ellipses
plotJointRPCA(
  mae,
  "JointRPCA",
  colour.by = "Fat",
  add.ellipse = TRUE
)

# Show all loading vectors
plotJointRPCA(
  mae,
  "JointRPCA",
  ntop = NULL
)

# Display only selected feature vectors
plotJointRPCA(
  mae,
  "JointRPCA",
  add.vectors = c("Bacteroides", "Roseburia")
)
```

```

)

# Use boxed labels without repelling
plotJointRPCA(
  mae,
  "JointRPCA",
  text.labels = FALSE,
  repel.labels = FALSE
)

```

plotLoadings	<i>Plot feature loadings for TreeSummarizedExperiment objects or feature loadings numeric matrix.</i>
--------------	---

Description

This function is used after performing a reduction method. If TreeSE is given it retrieves the feature loadings matrix to plot values. A tree from rowTree can be added to heatmap layout.

Usage

```

plotLoadings(x, ...)

## S4 method for signature 'TreeSummarizedExperiment'
plotLoadings(
  x,
  dimred,
  layout = "barplot",
  ncomponents = 5,
  tree.name = "phylo",
  row.var = NULL,
  add.tree = FALSE,
  ...
)

## S4 method for signature 'SingleCellExperiment'
plotLoadings(x, dimred, layout = "barplot", ncomponents = 5, ...)

## S4 method for signature 'matrix'
plotLoadings(x, layout = "barplot", ncomponents = 5, ...)

```

Arguments

x	a TreeSummarizedExperiment .
...	additional parameters for plotting. n: Integer scalar. Number of features to be plotted. Applicable when layout="barplot". (Default: min(nrow(x), 10L))

	• <code>absolute.scale</code>: ("barplot", "lollipop") Logical scalar. Specifies whether a barplot or a lollipop plot should be visualized in absolute scale. (Default: TRUE)
<code>dimred</code>	Character scalar. Determines the reduced dimension to plot.
<code>layout</code>	Character scalar. Determines the layout of plot. Must be either "barplot", "heatmap", or "lollipop". (Default: "barplot")
<code>ncomponents</code>	Numeric scalar. Number of components must be lower or equal to the number of components chosen in the reduction method. (Default: 5)
<code>tree.name</code>	Character scalar. Specifies a rowTree/colTree from x. (Default: <code>tree.name = "phylo"</code>)
<code>row.var</code>	NULL or Character scalar. Specifies a variable from rowData to plot with tree heatmap layout. (Default: NULL)
<code>add.tree</code>	Logical scalar. Whether to add tree to heatmap layout. (Default: FALSE)

Details

These method visualize feature loadings of dimension reduction results. Inspired by the `plotASVcircular` method using `phyloseq`. `TreeSummarizedExperiment` object is expected to have content in `reducedDim` slot calculated with standardized methods from `mia` or `scater` package.

Value

A `ggplot2` object.

Examples

```
library(mia)
library(scater)
data("GlobalPatterns", package = "mia")
tse <- GlobalPatterns

# Calculate PCA
tse <- agglomerateByPrevalence(tse, rank="Phylum", update.tree = TRUE)
tse <- transformAssay(tse, method = "clr", pseudocount = 1)
tse <- runPCA(tse, ncomponents = 5, assay.type = "clr")

# Plotting feature loadings
plotLoadings(tse, dimred = "PCA", layout = "heatmap", add.tree = FALSE) |>
  # Remove this line to see messages
  suppressMessages()

# Plotting matrix as a barplot
loadings_matrix <- attr(reducedDim(tse, "PCA"), "rotation")
plotLoadings(loadings_matrix)

# Plotting more features but less components
plotLoadings(tse, dimred = "PCA", ncomponents = 2, n = 12)
```

```
# Plotting matrix as heatmap without tree
plotLoadings(loadings_matrix, layout = "heatmap")

# Plot with less components
plotLoadings(tse, "PCA", layout = "heatmap", ncomponents = 2)
```

plotMediation

Visualize mediation

Description

plotMediation() generates a heatmap from the results of mediation analysis produced with mia:getMediation() or mia:addMediation(). It displays effect size and significance of the Actual Causal Mediation Effect (ACME) and the Actual Direct Effect (ADE) for each mediator included in the object x.

Usage

```
plotMediation(x, ...)

## S4 method for signature 'SummarizedExperiment'
plotMediation(x, name = "mediation", ...)

## S4 method for signature 'data.frame'
plotMediation(x, layout = "heatmap", ...)
```

Arguments

x	a SummarizedExperiment object or a data.frame, returned as output from addMediation or getMediation , respectively.
...	Additional parameters for plotting. add.significance: Character scalar. Controls how p-values are displayed in the heatmap. Options include "symbol" (p-values displayed as symbols), "numeric" (p-values displayed as numeric values) and NULL (p-values not displayed). (Default: "symbol")
name	Character scalar value defining which mediation data to use. (Default: "mediation")
layout	Character scalar Determines the layout of plot. Must be either "heatmap" or "forest". (Default: "heatmap")

Details

plotMediation creates a heatmap starting from the output of the [mediation](#) functions, which are mia wrappers for the basic [mediate](#) function. Either a [SummarizedExperiment](#) or a data.frame object is supported as input. When the input is a SummarizedExperiment, this should contain the output of addMediation in the metadata slot and the argument name needs to be defined. When the input is a data.frame, this should be returned as output from getMediation.

Value

A ggplot2 object.

Examples

```
## Not run:
library(mia)
library(scater)

# Load dataset
data(hitchip1006, package = "miaTime")
tse <- hitchip1006

# Agglomerate features by family (merely to speed up execution)
tse <- agglomerateByRank(tse, rank = "Phylum")
# Convert BMI variable to numeric
tse[["bmi_group"]] <- as.numeric(tse[["bmi_group"]])

# Apply clr transformation to counts assay
tse <- transformAssay(tse, method = "clr", pseudocount = 1)

# Analyse mediated effect of nationality on BMI via clr-transformed features
# 100 permutations were done to speed up execution, but ~1000 are recommended
tse <- addMediation(
  tse, name = "assay_mediation",
  outcome = "bmi_group",
  treatment = "nationality",
  assay.type = "clr",
  covariates = c("sex", "age"),
  treat.value = "Scandinavia",
  control.value = "CentralEurope",
  boot = TRUE, sims = 100,
  p.adj.method = "fdr"
)

# Visualise results as heatmap
plotMediation(tse, "assay_mediation")

# Visualise results as forest plot
plotMediation(tse, "assay_mediation", layout = "forest")

## End(Not run)
```

plotNMDS

Wrapper for scater::plotReducedDim()

Description

Wrapper for scater::plotReducedDim()

Usage

```
plotNMDS(x, ..., ncomponents = 2)
```

Arguments

x a [SummarizedExperiment](#) object.

... additional arguments passed to `scater::plotReducedDim()`.

ncomponents Numeric scalar. indicating the number of dimensions to plot, starting from the first dimension. Alternatively, a numeric vector specifying the dimensions to be plotted. (Default: 2)

plotOrdination	<i>Visualize ordination results</i>
----------------	-------------------------------------

Description

Creates a two-dimensional ordination plot from a reduced dimension result stored in a [SingleCellExperiment](#) object. Samples can be coloured, filled, shaped, sized, grouped or faceted using sample metadata or feature abundances, and additional graphical elements such as confidence ellipses, centroids, vectors and density estimates can be added.

Usage

```
plotOrdination(x, ...)

## S4 method for signature 'SingleCellExperiment'
plotOrdination(x, dimred, ...)
```

Arguments

x a [SummarizedExperiment](#) object.

... Additional parameters for plotting.

- ncomponents**: integer vector of length 2 or integer scalar. Specifies which ordination components are plotted. If a scalar is provided, the first two components are used. (Default: 2L)
- colour.by**: NULL or character scalar. Specifies a variable from `colData(x)` or a feature from `rownames(x)` used to colour observations. Feature abundances are taken from `assay.type`. (Default: NULL)
- fill.by**: NULL or character scalar. Specifies a variable from `colData(x)` or a feature from `rownames(x)` used to fill observations or ellipses. Feature abundances are taken from `assay.type`. Cannot be used together with `add.density = TRUE`. (Default: NULL)
- shape.by**: NULL or character scalar. Specifies a categorical variable from `colData(x)` used for point shapes. (Default: NULL)

- `size.by`: NULL or character scalar. Specifies a variable from `colData(x)` used for point sizes. (Default: NULL)
- `group.by`: NULL or character scalar. Specifies a categorical variable from `colData(x)` used for grouping when drawing ellipses, centroids and centroid vectors. (Default: NULL)
- `linetype.by`: NULL or character scalar. Specifies a categorical variable from `colData(x)` used for ellipse line types. (Default: NULL)
- `pair.by`: NULL or character scalar. Specifies a variable from `colData(x)` identifying observations that should be connected by lines. (Default: NULL)
- `sort.by`: NULL or character scalar. Specifies a variable from `colData(x)` used to order observations before drawing connecting lines. (Default: NULL)
- `facet.by`: NULL or character scalar. Specifies a categorical variable from `colData(x)` used to split the plot into facets. (Default: NULL)
- `assay.type`: character scalar. Name of the assay used when `colour.by` or `fill.by` specifies a feature. (Default: "counts")
- `add.points`: logical scalar. Whether to draw sample points. (Default: TRUE)
- `add.ellipse`: logical scalar. Whether to draw confidence ellipses around groups. (Default: FALSE)
- `add.density`: logical scalar. Whether to draw a two-dimensional density estimate in the background. (Default: FALSE)
- `add.centroids`: logical scalar. Whether to draw group centroids. (Default: FALSE)
- `add.centroids.lines`: logical scalar. Whether to connect observations to their group centroids. (Default: FALSE)
- `add.vectors`: logical scalar. Whether to draw vectors from the global centroid to group centroids. (Default: FALSE)
- `add.rotation`: logical scalar. Whether to draw rotation (species score) coordinates if available in the ordination result. (Default: `add.species`)
- `add.species`: logical scalar. Alias for `add.rotation`. (Default: FALSE)
- `add.expl.var`: logical scalar. Whether to append the percentage of explained variance to the axis labels when available. (Default: FALSE)
- `scales`: character scalar. Scaling used for faceted plots. Passed to `ggplot2::facet_wrap()`. (Default: "fixed")
- `xlab, ylab`: character scalar. Axis labels. Defaults to the ordination component names.
- `panel.by.eigen`: logical scalar. Whether to scale the panel aspect ratio according to the eigenvalues of the ordination when available. (Default: TRUE)
- `point.shape`: Shape used for points. (Default: 19)
- `point.alpha`: numeric scalar. Transparency of points. Must be between 0 and 1. (Default: 0.4)
- `ellipse.alpha`: numeric scalar. Transparency of ellipse fills. Must be between 0 and 1. (Default: 0.2)
- `ellipse.linewidth`: numeric scalar. Line width of ellipse borders. (Default: 0.5 or 0 when `fill.by` is specified.)

- `ellipse.linetype`: Integer specifying the ellipse line type. (Default: 1)
 - `confidence.level`: numeric scalar. Confidence level used for ellipse calculation. Must be between 0 and 1. (Default: 0.95)
 - `adjust`: numeric scalar. Multiplicative adjustment for the bandwidth used in the background density estimate. (Default: 1)
- `dimred` character scalar. Name of the reduced dimension result stored in `reducedDim(x)` to visualize.

Details

This function provides a unified interface for visualizing ordination methods such as PCA, PCoA, MDS, t-SNE, UMAP, RDA and CCA. The plotted coordinates are retrieved from a reduced dimension result stored in `reducedDim(x)`.

Value

A `ggplot2` object.

See Also

- [scatter::plotReducedDim](#)
- [plotCCA](#)

Examples

```
data("Tito2024QMP")
tse <- Tito2024QMP

# Compute relative abundances and an MDS ordination
tse <- transformAssay(tse, method = "relabundance")
tse <- addMDS(tse, assay.type = "relabundance", method = "bray", ncomponents = 50)

# Basic ordination plot
plotOrdination(tse, "MDS")

# Colour samples by a sample-level variable
plotOrdination(tse, "MDS", colour.by = "diagnosis")

# Colour samples by the abundance of a single feature
plotOrdination(
  tse, "MDS",
  colour.by = rownames(tse)[1],
  assay.type = "relabundance")

# Use multiple aesthetics simultaneously
plotOrdination(
  tse, "MDS",
  colour.by = "diagnosis",
  shape.by = "colonoscopy"
)
```

```
# Add confidence ellipses
plotOrdination(
  tse, "MDS",
  colour.by = "diagnosis",
  group.by = "diagnosis",
  add.ellipse = TRUE
)

# Fill ellipses instead of colouring them
plotOrdination(
  tse, "MDS",
  fill.by = "diagnosis",
  group.by = "diagnosis",
  add.ellipse = TRUE
)

# Show explained variance in the axis labels
plotOrdination(
  tse, "MDS",
  colour.by = "diagnosis",
  add.expl.var = TRUE
)

# Add group centroids
plotOrdination(
  tse, "MDS",
  colour.by = "diagnosis",
  group.by = "diagnosis",
  add.centroids = TRUE
)

# Connect samples to their group centroids
plotOrdination(
  tse, "MDS",
  colour.by = "diagnosis",
  group.by = "diagnosis",
  add.centroids.lines = TRUE
)

# Show vectors from the global centroid to each group centroid
plotOrdination(
  tse, "MDS",
  colour.by = "diagnosis",
  group.by = "diagnosis",
  add.vectors = TRUE
)

# Draw a background density estimate
plotOrdination(
  tse, "MDS",
  add.density = TRUE
)
```

```

# Split the plot into facets
plotOrdination(
  tse, "MDS",
  colour.by = "diagnosis",
  facet.by = "colonoscopy"
)

# Plot different ordination components
plotOrdination(
  tse, "MDS",
  ncomponents = c(2, 3)
)

# Customize point appearance
plotOrdination(
  tse, "MDS",
  colour.by = "diagnosis",
  point.shape = 17,
  point.alpha = 0.8
)

```

plotRowPrevalence	<i>Plot prevalence information</i>
-------------------	------------------------------------

Description

plotPrevalence and plotRowPrevalence visualize prevalence information.

Usage

```

plotRowPrevalence(x, ...)

plotPrevalentAbundance(x, ...)

plotPrevalence(x, ...)

## S4 method for signature 'SummarizedExperiment'
plotPrevalence(
  x,
  detection = detections,
  detections = c(0.01, 0.1, 1, 2, 5, 10, 20),
  prevalence = prevalences,
  prevalences = seq(0.1, 1, 0.1),
  assay.type = assay_name,
  assay_name = "counts",
  rank = NULL,
  BPPARAM = BiocParallel::SerialParam(),
  ...

```

```

)

## S4 method for signature 'SummarizedExperiment'
plotPrevalentAbundance(
  x,
  rank = NULL,
  assay.type = assay_name,
  assay_name = "counts",
  colour.by = colour_by,
  colour_by = NULL,
  size.by = size_by,
  size_by = NULL,
  shape.by = shape_by,
  shape_by = NULL,
  show.label = label,
  label = NULL,
  facet.by = facet_by,
  facet_by = NULL,
  ...
)

## S4 method for signature 'SummarizedExperiment'
plotRowPrevalence(
  x,
  rank = NULL,
  assay.type = assay_name,
  assay_name = "counts",
  detection = detections,
  detections = c(0.01, 0.1, 1, 2, 5, 10, 20),
  min.prevalence = min_prevalence,
  min_prevalence = 0,
  BPPARAM = BiocParallel::SerialParam(),
  ...
)

```

Arguments

<code>x</code>	a SummarizedExperiment object.
<code>detection</code>	Numeric scalar. Detection thresholds for absence/presence. Either an absolute value compared directly to the values of <code>x</code> or a relative value between 0 and 1, if <code>TRUE</code> .
<code>detections</code>	Deprecated. Use <code>detection</code> instead.
<code>prevalence</code>	Numeric scalar. Prevalence thresholds (in 0 to 1). The required prevalence is strictly greater by default. To include the limit, set <code>include.lowest</code> to <code>TRUE</code> .
<code>prevalences</code>	Deprecated. Use <code>prevalence</code> instead.
<code>assay.type</code>	Character scalar. Defines which assay data to use. (Default: "relabundance")
<code>assay_name</code>	Deprecated. Use <code>assay.type</code> instead.

rank, ...	additional arguments • <code>as.relative</code> Logical scalar. Should the relative values be calculated? (Default: FALSE) • <code>ndetection</code> Integer scalar. Determines the number of breaks calculated detection thresholds when <code>detection=NULL</code>. When TRUE, <code>as_relative</code> is then also regarded as TRUE. (Default: 20) • If <code>!is.null(rank)</code> matching arguments are passed on to agglomerateByRank. See ?agglomerateByRank for more details. • additional arguments for plotting. See mia-plot-args for more details i.e. call <code>help("mia-plot-args")</code>
BPPARAM	A BiocParallelParam object specifying whether the UniFrac calculation should be parallelized.
colour.by	Character scalar. Specification of a feature to colour points by, see the by argument in ?retrieveFeatureInfo for possible values. Only used with layout = "point". (Default: NULL)
colour_by	Deprecated. Use <code>colour.by</code> instead.
size.by	Character scalar. Specification of a feature to size points by, see the by argument in ?retrieveFeatureInfo for possible values. Only used with layout = "point". (Default: NULL)
size_by	Deprecated. Use <code>size.by</code> instead.
shape.by	Character scalar. Specification of a feature to shape points by, see the by argument in ?retrieveFeatureInfo for possible values. Only used with layout = "point". (Default: NULL)
shape_by	Deprecated. Use <code>shape.by</code> instead.
show.label	Logical scalar, character scalar or integer vector for selecting labels from the rownames of x. If rank is not NULL the rownames might change. (Default: NULL)
label	Deprecated. Use <code>show.label</code> instead.
facet.by	Character scalar. Taxonomic rank to facet the plot by. Value must be of <code>taxonomyRanks(x)</code> Argument can only be used in function <code>plotPrevalentAbundance</code> .
facet_by	Deprecated. Use <code>facet.by</code> instead.
min.prevalence	Numeric scalar. Applied as a threshold for plotting. The threshold is applied per row and column. (Default: 0)
min_prevalence	Deprecated. Use <code>min.prevalence</code> instead.

Details

Whereas `plotPrevalence` produces a line plot, `plotRowPrevalence` returns a heatmap.

Agglomeration on different taxonomic levels is available through the rank argument.

To exclude certain taxa, preprocess x to your liking, for example with subsetting via `getPrevalent` or `agglomerateByPrevalence`.

Value

A ggplot2 object or plotly object, if more than one prevalence was defined.

See Also

[getPrevalence](#), [agglomerateByPrevalence](#), [agglomerateByRank](#)

Examples

```
data(GlobalPatterns, package = "mia")

# Apply relative transformation
GlobalPatterns <- transformAssay(GlobalPatterns, method = "relabundance")

# plotting N of prevalence exceeding taxa on the Phylum level
plotPrevalence(GlobalPatterns, rank = "Phylum")
plotPrevalence(GlobalPatterns, rank = "Phylum") + scale_x_log10()

# plotting prevalence per taxa for different detection thresholds as heatmap
plotRowPrevalence(GlobalPatterns, rank = "Phylum")

# by default a continuous scale is used for different detection levels,
# but this can be adjusted
plotRowPrevalence(
  GlobalPatterns, rank = "Phylum", assay.type = "relabundance",
  detection = c(0, 0.001, 0.01, 0.1, 0.2))

# point layout for plotRowPrevalence can be used to visualize by additional
# information
plotPrevalentAbundance(
  GlobalPatterns, rank = "Family", colour.by = "Phylum") +
  scale_x_log10()

# When using function plotPrevalentAbundance, it is possible to create facets
# with 'facet.by'.
plotPrevalentAbundance(
  GlobalPatterns, rank = "Family",
  colour.by = "Phylum", facet.by = "Kingdom") +
  scale_x_log10()
```

plotRowTree

Plotting tree information enriched with information

Description

Based on the stored data in a TreeSummarizedExperiment a tree can be plotted. From the rowData, the assays as well as the colData information can be taken for enriching the tree plots with additional information.

Usage

```

plotRowTree(x, ...)

plotColTree(x, ...)

## S4 method for signature 'TreeSummarizedExperiment'
plotColTree(x, tree.name = "phylo", ...)

## S4 method for signature 'TreeSummarizedExperiment'
plotRowTree(x, tree.name = "phylo", ...)

```

Arguments

x a [TreeSummarizedExperiment](#).

... additional arguments for plotting.

- **layout**: layout for the plotted tree. See [ggtree](#) for details.
- **relabel.tree**: Logical scalar. Should the tip labels be relabeled using the output of `getTaxonomyLabels(x, with_rank = TRUE)`? (Default: FALSE)
- **order.tree**: Logical scalar. Should the tree be ordered based on alphabetic order of taxonomic levels? (Default: FALSE)
- **levels.rm**: Logical scalar. Should taxonomic level information be removed from labels? (Default: FALSE)
- **show.label**, **show.highlights**, **show.highlight.label**, **abbr.label** logical vector, integer vector. or character vector. If a logical scalar is given, should tip labels be plotted or if a logical vector is provided, which labels should be shown? If an integer or character vector is provided, it will be converted to a logical vector. The integer values must be in the range of 1 and number of nodes, whereas the values of a character vector must match values of the label column in the node data. In case of a character vector only values corresponding to actual labels will be plotted and if no labels are provided no labels will be shown. (Default: FALSE)
- **add.legend**: Logical scalar. Should legends be plotted? (Default: TRUE)
- **edge.colour.by**: Character scalar. Specification of a column metadata field or a feature to colour tree edges by. (Default: NULL)
- **edge.size.by**: Character scalar. Specification of a column metadata field or a feature to size tree edges by. (Default: NULL)
- **colour.by**: Character scalar. Specification of a column metadata field or a feature to colour tree nodes by. (Default: NULL)
- **shape.by**: Character scalar. Specification of a column metadata field or a feature to shape tree nodes by. (Default: NULL)
- **size.by**: Character scalar. Specification of a column metadata field or a feature to size tree tips by. (Default: NULL)
- **show.tips**: Logical scalar. Whether to show tip points. (Default: FALSE)
- **show.nodes**: Logical scalar. Whether to show node points. (Default: FALSE)

- `colour.highlights.by`: Logical scalar. Should the highlights be colour differently? If `show.highlights = TRUE`, `colour_highlights` will be set to TRUE as default. (Default: FALSE)
 - `assay.type`: Character scalar. Specifies which assay to obtain expression values from, for use in point aesthetics. (Default: "counts")
 - `other.fields`: Character vector. Additional fields to include in the node information without plotting them. (Default: NULL)
- `tree.name` Character scalar. Specifies a rowTree/colTree from x. (Default: `tree.name = "phylo"`)

Details

If `show.label` or `show.highlight.label` have the same length as the number of nodes, the vector will be used to relabel the nodes.

Value

a [ggtree](#) plot

See Also

[agglomerateByRanks](#)

Examples

```
library(scater)
library(mia)
# preparation of some data
data(GlobalPatterns)
GlobalPatterns <- agglomerateByRanks(GlobalPatterns)
altExp(GlobalPatterns, "Genus") <- addPerFeatureQC(
  altExp(GlobalPatterns, "Genus"))
rowData(altExp(GlobalPatterns, "Genus"))$log_mean <- log(
  rowData(altExp(GlobalPatterns, "Genus"))$mean)
rowData(altExp(GlobalPatterns, "Genus"))$detected <- rowData(
  altExp(GlobalPatterns, "Genus"))$detected / 100
top_genus <- getTop(
  altExp(GlobalPatterns, "Genus"),
  method = "mean",
  top = 100L,
  assay.type = "counts"
)
#
x <- altExp(GlobalPatterns, "Genus")
plotRowTree(
  x[rownames(x) %in% top_genus,],
  tip.colour.by = "log_mean", tip.size.by = "detected"
)

# plot with tip labels
plotRowTree(
```

```

    x[rownames(x) %in% top_genus,],
    tip.colour.by = "log_mean",
    tip.size.by = "detected",
    show.label = TRUE
  )
  # plot with selected labels
  labels <- c("Genus:Providencia", "Genus:Morganella", "0.961.60")
  plotRowTree(
    x[rownames(x) %in% top_genus,],
    tip.colour.by = "log_mean",
    tip.size.by = "detected",
    show.label = labels,
    layout = "rectangular"
  )

  # plot with labeled edges
  plotRowTree(
    x[rownames(x) %in% top_genus,],
    edge.colour.by = "Phylum",
    tip.colour.by = "log_mean"
  )
  # if edges are sized, colours might disappear depending on plotting device
  plotRowTree(
    x[rownames(x) %in% top_genus,],
    node.colour.by = "Phylum",
    edge.size.by = "detected",
    edge.colour.by = "log_mean"
  )

  # aggregating data over the taxonomic levels for plotting a taxonomic tree
  # please note that the original tree of GlobalPatterns is dropped by
  # unsplitByRanks
  altExps(GlobalPatterns) <- splitByRanks(GlobalPatterns)
  top_phyla <- getTop(
    altExp(GlobalPatterns, "Phylum"),
    method = "mean",
    top = 10L,
    assay.type = "counts"
  )
  altExps(GlobalPatterns) <- lapply(altExps(GlobalPatterns), addPerFeatureQC)
  altExps(GlobalPatterns) <- lapply(
    altExps(GlobalPatterns), function(y){
      rowData(y)$log_mean <- log(rowData(y)$mean)
      rowData(y)$detected <- rowData(y)$detected / 100
      return(y)
    })
  x <- unsplitByRanks(GlobalPatterns)
  x <- addHierarchyTree(x)

  highlights <- c(
    "Phylum:Firmicutes", "Phylum:Bacteroidetes",
    "Family:Pseudomonadaceae", "Order:Bifidobacteriales")
  plotRowTree(

```

```

x[rowData(x)$Phylum %in% top_phyla,],
tip.colour.by = "log_mean",
node.colour.by = "log_mean",
show.highlights = highlights,
show.highlight.label = highlights,
colour.highlights.by = "Phylum"
)

# If you do not want to show internal nodes
plotRowTree(
  x[rowData(x)$Phylum %in% top_phyla,],
  edge.colour.by = "Phylum",
  edge.size.by = "detected",
  node.colour.by = "log_mean",
  show.nodes = FALSE
)

```

plotScree

Create a scree plot

Description

plotScree generates a scree plot to visualize the eigenvalues. The eigenvalues can be provided either as a part of a TreeSummarizedExperiment object or as a separate vector. This plot illustrates the decline in eigenvalues across components, helping to assess the importance of each component.

Usage

```

plotScree(x, ...)

## S4 method for signature 'SingleCellExperiment'
plotScree(x, dimred, ...)

## S4 method for signature 'ANY'
plotScree(x, ...)

```

Arguments

x	a TreeSummarizedExperiment eigenvals or a vector.
...	additional parameters for plotting • show.barplot: Logical scalar. Whether to show a barplot. (Default: TRUE) • show.points: Logical scalar. Whether to show a points. (Default: TRUE) • show.line: Logical scalar. Whether to show a line. (Default: TRUE) • show.labels: Logical scalar. Whether to show a labels for each point. (Default: FALSE)

- `add.proportion`: Logical scalar. Whether to show proportion of explained variance, i.e., raw eigenvalues. (Default: TRUE)
 - `add.cumulative`: Logical scalar. Whether to show cumulative explained variance calculated from eigenvalues. (Default: FALSE)
 - `n`: Integer scalar. Number of eigenvalues to plot. If NULL, all eigenvalues are plotted. (Default: NULL)
 - `show.names`: Logical scalar. Whether to show names of components in x-axis. If FALSE, the index of component is shown instead of names. (Default: FALSE)
 - `eig.name`: Character scalar. The name of the attribute in `reducedDim(x, dimred)` that contains the eigenvalues. (Default: `c("eig", "varExplained")`)
- `dimred` Character scalar or integer scalar. Determines the reduced dimension to plot. This is used when `x` is a `TreeSummarizedExperiment` to extract the eigenvalues from `reducedDim(x, dimred)`.

Details

`plotScree` generates a scree plot to visualize the relative importance of components in dimensionality reduction techniques such as Principal Component Analysis (PCA) or Principal Coordinate Analysis (PCoA). If the input is a `TreeSummarizedExperiment` object, the function extracts eigenvalues from the specified reduced dimension slot, which requires that dimensionality reduction has been performed beforehand using a dedicated function. Alternatively, if the input is a vector or an `eigenvals` object, these values are directly used as eigenvalues for the plot.

The plot can include a combination of barplot, points, connecting lines, and labels, which can be controlled using the `show.*` parameters.

An option to show cumulative explained variance is also available by setting `add.cumulative = TRUE`.

Value

A `ggplot2` object

Examples

```
library(miaViz)
library(scater)

data("enterotype", package = "mia")
tse <- enterotype

# Run PCA and store results into TreeSE
tse <- transformAssay(tse, method = "clr", pseudocount = TRUE)
tse <- runPCA(tse, assay.type = "clr")

# Plot scree plot
plotScree(tse, "PCA", add.cumulative = TRUE)
```

plotSeries

*Plot Series***Description**

This function plots time series data.

Usage

```
plotSeries(x, ...)

## S4 method for signature 'SummarizedExperiment'
plotSeries(
  x,
  time.col,
  assay.type = NULL,
  col.var = NULL,
  features = NULL,
  facet.by = NULL,
  ...
)
```

Arguments

x	a SummarizedExperiment object.
...	additional parameters for plotting. • rank Character scalar. A taxonomic rank, that is used to agglomerate the data. (Default: NULL) • colour.by Character scalar. A column name from <code>rowData(x)</code> or <code>colData(x)</code>, that is used to divide observations to different colors. If NULL, this is not applied. (Default: NULL) • linetype.by Character scalar. A column name from <code>rowData(x)</code> or <code>colData(x)</code>, that is used to divide observations to different line types. If NULL, this is not applied. (Default: NULL) • size.by: Character scalar. A column name from <code>rowData(x)</code> or <code>colData(x)</code>, that is used to divide observations to different size types. If NULL, this is not applied. (Default: NULL) • ncol: Numeric scalar. if facets are applied, ncol defines many columns should be for plotting the different facets. (Default: 1L) • scales Character scalar. Defines the behavior of the scales of each facet. The value is passed into facet_wrap. (Default: "fixed") See mia-plot-args for more details i.e. call <code>help("mia-plot-args")</code>
time.col	Character scalar. Selecting the column from colData that will specify values of x-axis.
assay.type	Character scalar. Specifies the assay to be plotted.

col.var	Character scalar. Selecting the column from <code>colData</code> that will be plotted. This can be used instead of <code>assay.type</code> for visualizing temporal changes in sample metadata variable.
features	Character scalar. Selects the taxa from <code>rownames</code> . This parameter specifies taxa whose abundances will be plotted.
facet.by	Character scalar. Specifies a sample grouping. Must be value from <code>rowData</code> or <code>colData</code> . If NULL, grouping is not applied. (Default: NULL)

Details

This function creates series plot, where x-axis includes e.g. time points, and y-axis abundances of selected taxa. If there are multiple observations for single system and time point, mean and standard deviation is plotted.

Value

A ggplot2 object

Examples

```
## Not run:
library(mia)
# Load data from miaTime package
library("miaTime")
data(SilvermanAGutData)
tse <- SilvermanAGutData

# Plots 2 most abundant taxa, which are colored by their family
plotSeries(
  tse,
  assay.type = "counts",
  time.col = "DAY_ORDER",
  features = getTop(tse, 2),
  colour.by = "Family"
)

# Counts relative abundances
tse <- transformAssay(tse, method = "relabundance")

# Selects taxa
taxa <- c("seq_1", "seq_2", "seq_3", "seq_4", "seq_5")

# Plots relative abundances of phylums
plotSeries(
  tse[taxa,],
  time.col = "DAY_ORDER",
  colour.by = "Family",
  linetype.by = "Phylum",
  assay.type = "relabundance"
)
```

```

# In addition to 'colour.by' and 'linetype.by', 'size.by' can also be used
# to group taxa.
plotSeries(
  tse,
  time.col = "DAY_ORDER",
  features = getTop(tse, 5),
  colour.by = "Family",
  size.by = "Phylum",
  assay.type = "counts"
)

# If the data includes multiple systems, e.g., patients or bioreactors,
# one can plot each system separately
plotSeries(
  tse,
  time.col = "DAY_ORDER",
  assay.type = "relabundance",
  features = getTop(tse, 5),
  facet.by = "Vessel",
  colour.by = "rownames", colour.lab = "Feature",
  linetype.by = "Pre_Post_Challenge",
  scales = "free"
)

# One can visualize colData variables by specifying col.var
# First calculate alpha diversity index to visualize.
tse <- addAlpha(tse, index = "shannon")
# Then create a plot
plotSeries(
  tse,
  col.var = "shannon",
  time.col = "DAY_ORDER",
  facet.by = "Vessel",
)

## End(Not run)

```

rowTreeData

Adding information to tree data in TreeSummarizedExperiment

Description

To facilitate the dressing of the tree data stored in a `TreeSummarizedExperiment` object, `rowTreeData` and `colTreeData` can be used.

Usage

```
rowTreeData(x, ...)
```

```

colTreeData(x, ...)

rowTreeData(x, tree.name = tree_name, tree_name = "phylo") <- value

colTreeData(x, tree.name = tree_name, tree_name = "phylo") <- value

combineTreeData(x, other.fields = other_fields, other_fields = list())

combineTreeData(x, other.fields = other_fields, other_fields = list())

## S4 method for signature 'TreeSummarizedExperiment'
colTreeData(x, tree.name = tree_name, tree_name = "phylo")

## S4 method for signature 'TreeSummarizedExperiment'
rowTreeData(x, tree.name = tree_name, tree_name = "phylo")

## S4 replacement method for signature 'TreeSummarizedExperiment'
colTreeData(x, tree.name = tree_name, tree_name = "phylo") <- value

## S4 replacement method for signature 'TreeSummarizedExperiment'
rowTreeData(x, tree.name = tree_name, tree_name = "phylo") <- value

## S4 method for signature 'phylo'
combineTreeData(x, other.fields = other_fields, other_fields = list())

## S4 method for signature 'treedata'
combineTreeData(x, other.fields = other_fields, other_fields = list())

```

Arguments

<code>x</code>	a TreeSummarizedExperiment object.
<code>...</code>	additional arguments, currently not used.
<code>tree.name</code>	Character scalar. Specifies a rowTree/colTree from x. (Default: "phylo")
<code>tree_name</code>	Deprecated. Use <code>tree.name</code> instead.
<code>other.fields, value</code>	a data.frame or coercible to one, with at least one type of id information. See details.(Default: list())
<code>other_fields</code>	Deprecated. Use <code>other.fields</code> instead.

Details

To match information to nodes, the id information in `other.fields` are used. These can either be a column, named 'node' or 'label' ('node' taking precedent), or rownames. If all rownames can be coerced to integer, they are considered as 'node' values, otherwise as 'label' values. The id information must be unique and match available values of `rowTreeData(c)`

The result of the accessors, `rowTreeData` and `colTreeData`, contain at least a 'node' and 'label' column.

Value

a `data.frame` for the accessor and the modified `TreeSummarizedExperiment` object

Examples

```
data(GlobalPatterns)
td <- rowTreeData(GlobalPatterns)
td
td$test <- rnorm(nrow(td))
rowTreeData(GlobalPatterns) <- td
rowTreeData(GlobalPatterns)
combineTreeData(rowTree(GlobalPatterns), td)
```

Index

- * **datasets**
 - mia-datasets, [6](#)
- ?agglomerateByRank, [49](#)
- ?retrieveCellInfo, [24](#)
- ?retrieveFeatureInfo, [49](#)
- addMediation, [41](#)
- addRDA, [20](#)
- agglomerateByPrevalence, [50](#)
- agglomerateByRank, [49](#), [50](#)
- agglomerateByRanks, [52](#)
- AsIs, [26](#)
- assay, [56](#)
- beeswarm::beeswarm(), [15](#)
- BiocParallelParam, [49](#)
- calculateDMN, [27](#)
- col_graph (mia-datasets), [6](#)
- colData, [56](#), [57](#)
- colTreeData (rowTreeData), [58](#)
- colTreeData, TreeSummarizedExperiment-method (rowTreeData), [58](#)
- colTreeData<- (rowTreeData), [58](#)
- colTreeData<- , TreeSummarizedExperiment-method (rowTreeData), [58](#)
- combineTreeData (rowTreeData), [58](#)
- combineTreeData, phylo-method (rowTreeData), [58](#)
- combineTreeData, treedata-method (rowTreeData), [58](#)
- deprecate, [3](#)
- eigenvals, [54](#)
- facet_wrap, [10](#), [56](#)
- geom_edge_fan, [23](#)
- geom_label, [19](#)
- geom_label_repel, [19](#)
- getMediation, [41](#)
- getNeatOrder, [4](#)
- getNeatOrder, matrix-method (getNeatOrder), [4](#)
- getPrevalence, [50](#)
- ggplot, [10](#), [30](#)
- ggraph, [23](#)
- ggtree, [24](#), [51](#), [52](#)
- mediate, [41](#)
- mediation, [41](#)
- metadata, [27](#)
- mia, [3](#)
- mia-datasets, [6](#)
- mia-plot-args, [6](#)
- miaViz (miaViz-package), [2](#)
- miaViz-package, [2](#)
- MultiAssayExperiment, [37](#)
- plotAbundance, [9](#)
- plotAbundance, SummarizedExperiment-method (plotAbundance), [9](#)
- plotAbundanceDensity, [11](#), [36](#)
- plotAbundanceDensity, SummarizedExperiment-method (plotAbundanceDensity), [11](#)
- plotBarplot (plotHistogram), [34](#)
- plotBarplot, SummarizedExperiment-method (plotHistogram), [34](#)
- plotBoxplot, [15](#)
- plotBoxplot, SummarizedExperiment-method (plotBoxplot), [15](#)
- plotCCA, [19](#), [45](#)
- plotCCA, SingleCellExperiment-method (plotCCA), [19](#)
- plotColGraph, [21](#)
- plotColGraph, ANY, SummarizedExperiment-method (plotColGraph), [21](#)
- plotColGraph, SummarizedExperiment, missing-method (plotColGraph), [21](#)
- plotColTile, [26](#)

- plotColTile, SummarizedExperiment-method
(plotColTile), 26
- plotColTree (plotRowTree), 50
- plotColTree, TreeSummarizedExperiment-method
(plotRowTree), 50
- plotDMN (plotDMNFit), 27
- plotDMNFit, 27
- plotDMNFit, SummarizedExperiment-method
(plotDMNFit), 27
- plotFeaturePrevalence (deprecate), 3
- plotFeaturePrevalence, ANY-method
(deprecate), 3
- plotForest, 28
- plotForest, data.frame-method
(plotForest), 28
- plotForest, SummarizedExperiment-method
(plotForest), 28
- plotForest, TreeSummarizedExperiment-method
(plotForest), 28
- plotGraph (plotColGraph), 21
- plotHeatmap, 32
- plotHeatmap, SummarizedExperiment-method
(plotHeatmap), 32
- plotHeatmap, TreeSummarizedExperiment-method
(plotHeatmap), 32
- plotHistogram, 34
- plotHistogram, SummarizedExperiment-method
(plotHistogram), 34
- plotJointRPCA, 37
- plotJointRPCA, MultiAssayExperiment-method
(plotJointRPCA), 37
- plotJointRPCA, SingleCellExperiment-method
(plotJointRPCA), 37
- plotLoadings, 39
- plotLoadings, matrix-method
(plotLoadings), 39
- plotLoadings, SingleCellExperiment-method
(plotLoadings), 39
- plotLoadings, TreeSummarizedExperiment-method
(plotLoadings), 39
- plotMediation, 41
- plotMediation, data.frame-method
(plotMediation), 41
- plotMediation, SummarizedExperiment-method
(plotMediation), 41
- plotNMDS, 42
- plotOrdination, 21, 37, 38, 43
- plotOrdination, SingleCellExperiment-method
(plotOrdination), 43
- plotPrevalence (plotRowPrevalence), 47
- plotPrevalence, SummarizedExperiment-method
(plotRowPrevalence), 47
- plotPrevalentAbundance
(plotRowPrevalence), 47
- plotPrevalentAbundance, SummarizedExperiment-method
(plotRowPrevalence), 47
- plotRDA (plotCCA), 19
- plotRDA, SingleCellExperiment-method
(plotCCA), 19
- plotReducedDim, 19, 20
- plotRowGraph (plotColGraph), 21
- plotRowGraph, ANY, SummarizedExperiment-method
(plotColGraph), 21
- plotRowGraph, SummarizedExperiment, missing-method
(plotColGraph), 21
- plotRowPrevalence, 47
- plotRowPrevalence, SummarizedExperiment-method
(plotRowPrevalence), 47
- plotRowTile (plotColTile), 26
- plotRowTile, SummarizedExperiment-method
(plotColTile), 26
- plotRowTree, 29, 33, 34, 50
- plotRowTree, TreeSummarizedExperiment-method
(plotRowTree), 50
- plotScree, 54
- plotScree, ANY-method (plotScree), 54
- plotScree, SingleCellExperiment-method
(plotScree), 54
- plotSeries, 56
- plotSeries, SummarizedExperiment-method
(plotSeries), 56
- plotTaxaPrevalence (deprecate), 3
- plotTaxaPrevalence, ANY-method
(deprecate), 3
- plotTree (plotRowTree), 50
- retrieveCellInfo, 26
- retrieveFeatureInfo, 26
- row_graph (mia-datasets), 6
- row_graph_order (mia-datasets), 6
- rowData, 57
- rownames, 57
- rowTreeData, 58
- rowTreeData, TreeSummarizedExperiment-method
(rowTreeData), 58
- rowTreeData<- (rowTreeData), 58

`rowTreeData<-`, `TreeSummarizedExperiment`-method
(`rowTreeData`), 58

`scater::plotColData`, 17, 36

`scater::plotExpression`, 13, 17, 36

`scater::plotReducedDim`, 45

`scater::plotRowData`, 17, 36

`SingleCellExperiment`, 43

`SummarizedExperiment`, 9, 12, 15, 23, 24, 26,
27, 29, 33, 35, 37, 41, 43, 48, 56

`treeData` (`rowTreeData`), 58

`TreeSummarizedExperiment`, 19, 20, 39, 51,
54, 59, 60

`vipor::offsetSingleGroup`(), 15