Package 'miaViz'

April 12, 2022

Title Microbiome Analysis Plotting and Visualization

Version 1.2.1

Description
The miaViz package implements plotting function to work with TreeSummarizedExperiment and related objects in a context of microbiome analysis. Among others this includes plotting tree, graph and microbiome series data. The package is part of the broader miaverse framework.

biocViews Microbiome, Software, Visualization

License Artistic-2.0 | file LICENSE

Encoding UTF-8

LazyData false

Depends R (>= 4.0), SummarizedExperiment, TreeSummarizedExperiment, mia (>= 0.99), ggplot2, ggraph (>= 2.0)

Imports methods, stats, S4Vectors, BiocGenerics, BiocParallel, DelayedArray, scater, ggtree, ggnewscale, viridis, tibble, tidytree, tidygraph, rlang, purrr, tidyr, dplyr, ape, DirichletMultinomial

Suggests knitr, rmarkdown, BiocStyle, testthat, patchwork, microbiomeDataSets

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

VignetteBuilder knitr

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

git_branch RELEASE_3_14

git_last_commit 093001e

git_last_commit_date 2021-12-31

Date/Publication 2022-04-12

Author Felix G.M. Ernst [aut, cre] (<https://orcid.org/0000-0001-5064-0928>), Tuomas Borman [aut] (<https://orcid.org/0000-0002-8563-8884>), Leo Lahti [aut] (<https://orcid.org/0000-0001-5537-637X>)

Maintainer Felix G.M. Ernst <felix.gm.ernst@outlook.com>

R topics documented:

`miaViz-package` *miaViz - Microbiome Analysis Plotting and Visualization*

Description

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

See Also

mia class

`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 10.

An object of class tbl_graph (inherits from igraph) of length 10.

An object of class tbl_graph (inherits from igraph) of length 10.

Details

For *_graph data:

1. "Jaccard" distances were calculated via calculateDistance(genus,FUN = vegan::vegdist,method = "jaccard",exprs_values = "relabundance"), either using transposed assay data or not to calculate distances for samples or features.
2. "Jaccard" dissimilarites 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], specifying the transparency of the tree edges. Defaults to 1.

line_width: Numeric scalar, specifying the default width of an edge. Defaults to NULL to use default of the ggtree package

line_width_range: Two numeric values, the range for plotting dynamic edge widths in. Defaults to c(0.5,3).

point_alpha: Numeric scalar in [0,1], specifying the transparency of the tips. Defaults to 1.

point_size: Numeric scalar, specifying the default size of tips Defaults to 2..

point_size_range: Two numeric values, the range for plotting dynamic tip sizes in. Defaults to c(1,4).

label_font_size: Numeric scalar, font size for the tip and node labels. Default to 3.

highlight_font_size: Numeric scalar, font size for the highlight labels. Default to 3.

Graph plotting

`line_alpha`: Numeric scalar in `[0,1]`, specifying the transparency of the tree edges. Defaults to `1`.

`line_width`: Numeric scalar, specifying the default width of an edge. Defaults to NULL to use default of the `ggraph` package

`line_width_range`: Two numeric values, the range for plotting dynamic edge widths in. Defaults to `c(0.5,3)`.

`point_alpha`: Numeric scalar in `[0,1]`, specifying the transparency of the tips. Defaults to `1`.

`point_size`: Numeric scalar, specifying the default size of tips Defaults to `2.`.

`point_size_range`: Two numeric values, the range for plotting dynamic tip sizes in. Defaults to `c(1,4)`.

Abundance plotting

`flipped`: Logical scalar. Should the plot be flipped? Defaults to `FALSE`.

`add_legend`: Logical scalar. Should legends be plotted? Defaults to `TRUE`.

`add_x_text`: Logical scalar. Should x tick labels be plotted? Defaults to `FALSE`.

`add_border`: Logical scalar. Should border of bars be plotted? Defaults to `FALSE`.

`bar_alpha`: Numeric scalar in `[0,1]`, specifying the transparency of the bars. Defaults to `1`.

`point_alpha`: Numeric scalar in `[0,1]`, specifying the transparency of the points. Defaults to `1`.

`point_size`: Numeric scalar, specifying the default size of points. Defaults to `2.`.

Abundance density plotting

`add_legend`: Logical scalar. Should legends be plotted? Defaults to `TRUE`.

`point_shape`: Numeric scalar setting the shape of points. Defaults to `21`.

`point_colour`: Character scalar, specifying the default colour of points. Defaults to `2.`.

`point_size`: Numeric scalar, specifying the default size of points. Defaults to `2.`.

`point_alpha`: Numeric scalar in `[0,1]`, specifying the transparency of the points. Defaults to `1`.

`flipped`: Logical scalar. Should the plot be flipped? Defaults to `FALSE`.

`scales_free`: Logical scalar. Should `scales = "free"` be set for faceted plots? Defaults to `TRUE`.

`angle_x_text`: Logical scalar. Should x tick labels be plotted? Defaults to `FALSE`.

Prevalence plotting

`flipped`: Logical scalar, specifying whether the plot should be flipped. Defaults to `FALSE`.

`add_legend`: Logical scalar. Should legends be plotted? Defaults to `TRUE`.

`point_alpha`: Numeric scalar in `[0,1]`, specifying the transparency of the tips. Defaults to `1`.

`point_size`: Numeric scalar, specifying the default size of tips Defaults to `2.`.

`line_alpha`: Numeric scalar in `[0,1]`, specifying the transparency of the tree edges. Defaults to `1`.

`line_type`: Numeric scalar, specifying the default line type. Defaults to NULL to use default of the `ggplot2` package

`line_size`: Numeric scalar, specifying the default width of a line. Defaults to NULL to use default of the `ggplot2` package

Series plotting

`add_legend`: Logical scalar. Should legends be plotted? Defaults to `TRUE`.

`line_alpha`: Numeric scalar in `[0,1]`, specifying the transparency of the tree edges. Defaults to `1`.

`line_type`: Numeric scalar, specifying the default line type. Defaults to NULL to use default of the `ggplot2` package

`line_width`: Numeric scalar, specifying the default width of a line. Defaults to NULL to use default of the `ggplot2` package

`line_width_range`: Two numeric values, the range for plotting dynamic line widths in. Defaults to `c(0.5,3)`.

`ribbon_alpha`: Numeric scalar in `[0,1]`, specifying the transparency of the ribbon. Defaults to `0.3`.

Tile plotting

`add_legend`: Logical scalar. Should legends be plotted? Defaults to `TRUE`.

`rect_alpha`: Numeric scalar in `[0,1]`, specifying the transparency of the areas. Defaults to `1`.

`rect_colour`: Character scalar, specfiying the colour to use for colouring the borders of the areas. Defaults to `"black"`.

`na.value`: Character scalar, specfiying the colour to use for `NA` values. Defaults to `"grey80"`.

---

plotAbundance *Plotting abundance data*

Description

`plotAbundance` plots the abundance on a selected taxonomic rank. Since this probably makes sense only for relative abundance data, the assay used by default is expected to be in the slot 'relabundance'. If only 'counts' is present, the relative abundance is computed.

Usage

```
plotAbundance(x, ...)

## S4 method for signature 'SummarizedExperiment'
plotAbundance(
  x,
  rank = taxonomyRanks(x)[1],
  features = NULL,
  order_rank_by = c("name", "abund", "revabund"),
  order_sample_by = NULL,
  decreasing = TRUE,
  use_relative = TRUE,
  layout = c("bar", "point"),
```

```
  one_facet = TRUE,
  ncol = 2,
  scales = "fixed",
  abund_values = "counts",
  ...
)
```

Arguments

x	a SummarizedExperiment object.
...	additional parameters for plotting. See mia-plot-args for more details i.e. call help("mia-plot-args")
rank	a single character value defining the taxonomic rank to use. Must be a value of taxonomyRanks(x).
features	a single character value defining 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: features = NULL)
order_rank_by	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").
order_sample_by	A single character value from the chosen rank of abundance data or from colData to select values to order the abundance plot by. (default: order_sample_by = NULL)
decreasing	TRUE or FALSE: If the order_sample_by is defined and the values are numeric, should the values used to order in decreasing or increasing fashion? (default: decreasing = FALSE)
use_relative	TRUE or FALSE: Should the relative values be calculated? (default: use_relative = TRUE)
layout	Either "bar" or "point".
one_facet	Should the plot be returned in on facet or split into different facet, one facet per different value detect in rank. If features or order_sample_by is not NULL, this setting will be disregarded.
ncol, scales	if one_facet = FALSE, ncol defines many columns should be for plotting the different facets and scales is used to define the behavior of the scales of each facet. Both values are passed onto facet_wrap.
abund_values	a character value defining which assay data to use. (default: abund_values = "relabundance")

Details

Subsetting to rows of interested and ordering of those is expected to be done outside of this functions, e.g. x[1:2,]. This will plot data of all features present.

Value

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

Examples

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

## Plotting abundance using the first taxonomic rank as default
plotAbundance(se, abund_values="counts")

## Using "Phylum" as rank
plotAbundance(se, abund_values="counts", rank = "Phylum", add_legend = FALSE)

## If rank is set to NULL plotAbundance behaves like plotExpression
plotAbundance(se, abund_values="counts", rank = NULL,
           features = head(rownames(se)))

## A feature from colData or taxon from chosen rank can be used for ordering samples.
plotAbundance(se, abund_values="counts", rank = "Phylum",
           order_sample_by = "Bacteroidetes")

## Features from colData can be plotted together with abundance plot.
# Returned object is a list that includes two plot; other visualizes abundance
# other features.
plot <- plotAbundance(se, abund_values = "counts", rank = "Phylum",
                  features = "SampleType")

# These two plots can be combined with wrap_plots function from patchwork package
library(patchwork)
wrap_plots(plot, ncol = 1)


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

# Getting top taxa on a Phylum level
se <- relAbundanceCounts(se)
se_phylum <- agglomerateByRank(se, rank ="Phylum", onRankOnly=TRUE)
top_taxa <- getTopTaxa(se_phylum,top = 5, abund_values = "relabundance")

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

# Compositional barplot
plotAbundance(se, abund_values="relabundance", rank = "Phylum",
           order_rank_by="abund", order_sample_by = "Bacteroidetes")
```

`plotAbundanceDensity` *Plot abundance density*

Description

This function plots abundance of the most abundant taxa.

Usage

```
plotAbundanceDensity(object, ...)

## S4 method for signature 'SummarizedExperiment'
plotAbundanceDensity(
  object,
  layout = c("jitter", "density", "point"),
  abund_values = "counts",
  n = min(nrow(object), 25L),
  colour_by = NULL,
  shape_by = NULL,
  size_by = NULL,
  order_descending = TRUE,
  ...
)
```

Arguments

`object`	a `SummarizedExperiment` object.
`...`	additional parameters for plotting. • xlab a single character value for selecting the x-axis label. (default: `xlab = abund_values`) • ylab a single character value for selecting the y-axis label. `ylab` is disabled when `layout = "density"`. (default: `ylab = "Taxa"`) • point_alpha a numeric value from range 0 to 1 selecting the transparency of colour in `jitter` and `point` plot. (default: `point_alpha = 0.6`) • point_shape a positive integer value selecting the shape of point in `jitter` and `point` plot. (default: `point_shape = 21`) • point_size a positive numeric value selecting the size of point in `jitter` and `point` plot. (default: `point_size = 2`) • add_legend a boolean value selecting if legend is added. (default: `add_legend = TRUE`) • flipped a boolean value selecting if the orientation of plot is changed so that x-axis and y-axis are swapped. (default `flipped = FALSE`) • add_x_text a boolean value selecting if text that represents values is included in x-axis. (default: `add_x_text = TRUE`) See `mia-plot-args` for more details i.e. call `help("mia-plot-args")`

layout	a single character value for selecting the layout of the plot. There are three different options: jitter, density, and point plot. (default: layout = "jitter")
abund_values	a single character value for selecting the assay to be plotted. (default: abund_values = "counts")
n	a positive integer specifying the number of the most abundant taxa to show. (default: n = min(nrow(object),25L))
colour_by	a single character value defining a column from colData, that is used to color plot. Must be a value of colData() function. (default: colour_by = NULL)
shape_by	a single character value defining a column from colData, that is used to group observations to different point shape groups. Must be a value of colData() function. shape_by is disabled when layout = "density". (default: shape_by = NULL)
size_by	a single character value defining a column from colData, that is used to group observations to different point size groups. Must be a value of colData() function. size_by is disabled when layout = "density". (default: size_by = NULL)
order_descending	TRUE, FALSE orNA: Should the results be ordered in a descending order? If NA is given the order as found in object for the n most abundant taxa is used. (default: order_descending = TRUE)

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

Author(s)

Leo Lahti and Tuomas Borman. Contact: microbiome.github.io

Examples

```
tse <- microbiomeDataSets::atlas1006()

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

# Counts relative abundances
tse <- transformSamples(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", abund_values = "relabundance",
                     n = 10, colour_by = "nationality") +
    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", abund_values = "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", abund_values = "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", abund_values = "relabundance", n = 10,
                     shape_by = "sex", size_by = "time", point_size=1)

# Ordering via order_descending
plotAbundanceDensity(tse, abund_values = "relabundance",
                     order_descending = FALSE)

# for custom ordering set order_descending = NA and order the input object
# to your wishes
plotAbundanceDensity(tse, abund_values = "relabundance",
                     order_descending = NA)
```

`plotColTile` *Plot factor data as tiles*

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	String specifying 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	String specifying 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 help("mia-plot-args")

Value

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

Examples

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

---

plotDMN *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	the name to store the result in metadata
type	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

```
data(dmn_se, package = "mia")
names(metadata(dmn_se))

# plot the fit
plotDMNFit(dmn_se, type = "laplace")
```

---

plotGraph *Plotting igraph objects with information from a* SummarizedExperiment

---

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 = FALSE,
  add_legend = TRUE,
  layout = "kk",
  edge_type = c("fan", "link", "arc", "parallel"),
  edge_colour_by = NULL,
  edge_width_by = NULL,
  colour_by = NULL,
  shape_by = NULL,
  size_by = NULL,
  by_exprs_values = "counts",
  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 = FALSE,
  add_legend = TRUE,
  layout = "kk",
  edge_type = c("fan", "link", "arc", "parallel"),
  edge_colour_by = NULL,
  edge_width_by = NULL,
  colour_by = NULL,
  shape_by = NULL,
  size_by = NULL,
  by_exprs_values = "counts",
  other_fields = list(),
  ...
)

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

Arguments

x, y	a graph object and a SummarizedExperiment object or just a SummarizedExperiment. For the latter object a graph object must be stored in metadata(x)$name.
...	additional arguments for plotting. See mia-plot-args for more details i.e. call help("mia-plot-args")
show_label	logical (scalar), integer 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: show_label = FALSE)
add_legend	logical scalar. Should legends be plotted? (default: add_legend = TRUE)
layout	layout for the plotted graph. See ggraph for details. (default: layout = "kk")
edge_type	type of edge plotted on the graph. See geom_edge_fan for details and other available geoms. (default: edge_type = "fan")
edge_colour_by	Specification of a edge metadata field to use for setting colours of the edges.
edge_width_by	Specification of a edge metadata field to use for setting width of the edges.
colour_by	Specification of a column metadata field or a feature to colour graph nodes by, see the by argument in ?retrieveCellInfo for possible values.

shape_by — Specification of a column metadata field or a feature to shape graph nodes by, see the by argument in ?retrieveCellInfo for possible values.

size_by — Specification of a column metadata field or a feature to size graph nodes by, see the by argument in ?retrieveCellInfo for possible values.

by_exprs_values — A string or integer scalar specifying which assay to obtain expression values from, for use in point aesthetics - see the exprs_values argument in ?retrieveCellInfo.

other_fields — Additional fields to include in the node information without plotting them.

name — If x is a SummarizedExperiment the key for subsetting the metadata(x) to a graph object.

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"))

# labls 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))

# labls 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)
```

plotPrevalence *Plot prevalence information*

Description

plotPrevalence and plotTaxaPrevalence visualize prevalence information.

Usage

```
plotPrevalence(x, ...)

## S4 method for signature 'SummarizedExperiment'
plotPrevalence(
  x,
  detections = c(0.01, 0.1, 1, 2, 5, 10, 20)/100,
  prevalences = seq(0.1, 1, 0.1),
  abund_values = "counts",
  as_relative = TRUE,
  rank = NULL,
  BPPARAM = BiocParallel::SerialParam(),
  ...
)

plotPrevalentAbundance(x, ...)

## S4 method for signature 'SummarizedExperiment'
plotPrevalentAbundance(
  x,
  rank = taxonomyRanks(x)[1L],
  abund_values = "counts",
  as_relative = TRUE,
  colour_by = NULL,
  size_by = NULL,
  shape_by = NULL,
  label = NULL,
  facet_by = NULL,
  ...
)

plotTaxaPrevalence(x, ...)

## S4 method for signature 'SummarizedExperiment'
plotTaxaPrevalence(
  x,
  rank = taxonomyRanks(x)[1L],
  abund_values = "counts",
  detections = NULL,
```

```
  ndetections = 20,
  as_relative = TRUE,
  min_prevalence = 0,
  BPPARAM = BiocParallel::SerialParam(),
  ...
)
```

Arguments

x	a SummarizedExperiment object.
detections	Detection thresholds for absence/presence. Either an absolutes value compared directly to the values of x or a relative value between 0 and 1, if as_relative = TRUE.
prevalences	Prevalence thresholds (in 0 to 1). The required prevalence is strictly greater by default. To include the limit, set include_lowest to TRUE.
abund_values	a character value defining which assay data to use. (default: abund_values = "relabundance")
as_relative	logical scalar: Should the detection threshold be applied on compositional (relative) abundances? Passed onto getPrevalence. (default: TRUE)
rank, ...	additional arguments • If !is.null(rank) 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 help("mia-plot-args")
BPPARAM	A BiocParallelParam object specifying whether the UniFrac calculation should be parallelized.
colour_by	Specification of a feature to colour points by, see the by argument in ?retrieveFeatureInfo for possible values. Only used with layout = "point".
size_by	Specification of a feature to size points by, see the by argument in ?retrieveFeatureInfo for possible values. Only used with layout = "point".
shape_by	Specification of a feature to shape points by, see the by argument in ?retrieveFeatureInfo for possible values. Only used with layout = "point".
label	a logical, character or integer vector for selecting labels from the rownames of x. If rank is not NULL the rownames might change. (default: label = NULL)
facet_by	Taxonomic rank to facet the plot by. Value must be of taxonomyRanks(x) Argument can only be used in function plotPrevalentAbundance.
ndetections	If detections is NULL, a number of breaks are calculated automatically. as_relative is then also regarded as TRUE.
min_prevalence	a single numeric value to apply as a threshold for plotting. The threshold is applied per row and column. (default: min_prevalence = 0)

Details

Whereas plotPrevalence procudes a line plot, plotTaxaPrevalence 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 getPrevalentTaxa or agglomerateByPrevalence.

Value

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

See Also

getPrevalence, agglomerateByPrevalence, agglomerateByRank

Examples

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

# 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 detectio thresholds as heatmap
plotTaxaPrevalence(GlobalPatterns, rank = "Phylum")

# by default a continous scale is used for different detections levels,
# but this can be adjusted
plotTaxaPrevalence(GlobalPatterns, rank = "Phylum",
                   detections = c(0, 0.001, 0.01, 0.1, 0.2))

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

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

---

plotSeries *Plot Series*

Description

This function plots series data.

Usage

```
plotSeries(
  object,
  x,
  y = NULL,
  rank = NULL,
  colour_by = NULL,
  size_by = NULL,
  linetype_by = NULL,
  abund_values = "counts",
  ...
)

## S4 method for signature 'SummarizedExperiment'
plotSeries(
  object,
  x,
  y = NULL,
  rank = NULL,
  colour_by = NULL,
  size_by = NULL,
  linetype_by = NULL,
  abund_values = "counts",
  ...
)
```

Arguments

object	a SummarizedExperiment object.
x	a single character value for selecting the column from ColData that will specify values of x-axis.
y	a single character value for selecting the taxa from rownames. This parameter specifies taxa whose abundances will be plotted.
rank	a single character value defining a taxonomic rank, that is used to agglomerate the data. Must be a value of taxonomicRanks() function.
colour_by	a single character value defining a taxonomic rank, that is used to color plot. Must be a value of taxonomicRanks() function.
size_by	a single character value defining a taxonomic rank, that is used to divide taxa to different line size types. Must be a value of taxonomicRanks() function.
linetype_by	a single character value defining a taxonomic rank, that is used to divide taxa to different line types. Must be a value of taxonomicRanks() function.
abund_values	a single character value for selecting the assay to be plotted. (default: abund_values = "counts")
...	additional parameters for plotting. See mia-plot-args for more details i.e. call help("mia-plot-args")

Details

This function creates series plot, where x-axis includes e.g. time points, and y-axis abundances of selected taxa.

Value

A ggplot2 object

Author(s)

Leo Lahti and Tuomas Borman. Contact: microbiome.github.io

Examples

```
library(mia)
object <- microbiomeDataSets::SilvermanAGutData()
# Plots 2 most abudant taxa, which are colore by their family
plotSeries(object,
           x = "DAY_ORDER",
           y = getTopTaxa(object, 2),
           colour_by = "Family")

# Counts relative abundances
object <- transformCounts(object, method = "relabundance")

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

# Plots relative abundances of phylums
plotSeries(object[taxa,],
           x = "DAY_ORDER",
           colour_by = "Family",
           linetype_by = "Phylum",
           abund_values = "relabundance")

# In addition to 'colour_by' and 'linetype_by', 'size_by' can also be used to group taxa.
plotSeries(object,
           x = "DAY_ORDER",
           y = getTopTaxa(object, 5),
           colour_by = "Family",
           size_by = "Phylum",
           abund_values = "counts")
```

---

plotTree *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(object, ...)

plotColTree(object, ...)

## S4 method for signature 'TreeSummarizedExperiment'
plotColTree(
  object,
  relabel_tree = FALSE,
  order_tree = FALSE,
  remove_levels = FALSE,
  show_label = FALSE,
  show_highlights = FALSE,
  show_highlight_label = FALSE,
  abbr_label = FALSE,
  add_legend = TRUE,
  layout = "circular",
  edge_colour_by = NULL,
  edge_size_by = NULL,
  tip_colour_by = NULL,
  tip_shape_by = NULL,
  tip_size_by = NULL,
  node_colour_by = NULL,
  node_shape_by = NULL,
  node_size_by = NULL,
  colour_highlights_by = NULL,
  by_exprs_values = "counts",
  other_fields = list(),
  ...
)

## S4 method for signature 'TreeSummarizedExperiment'
plotRowTree(
  object,
  relabel_tree = FALSE,
  order_tree = FALSE,
  remove_levels = FALSE,
  show_label = FALSE,
  show_highlights = FALSE,
  show_highlight_label = FALSE,
  abbr_label = FALSE,
  add_legend = TRUE,
  layout = "circular",
  edge_colour_by = NULL,
  edge_size_by = NULL,
  tip_colour_by = NULL,
  tip_shape_by = NULL,
  tip_size_by = NULL,
```

```
  node_colour_by = NULL,
  node_shape_by = NULL,
  node_size_by = NULL,
  colour_highlights_by = NULL,
  by_exprs_values = "counts",
  other_fields = list(),
  ...
)
```

Arguments

object	a TreeSummarizedExperiment object.
...	additional arguments for plotting. See mia-plot-args for more details i.e. call help("mia-plot-args")
relabel_tree	logical scalar, Should the tip labels be relabeled using the output of getTaxonomyLabels(object,with_r = TRUE)? (default: relabel_tree = FALSE)
order_tree	logical scalar, Should the tree be ordered based on alphabetic order of taxonomic levels? (default: order_tree = FALSE)
remove_levels	logical scalar, Should taxonomic level information be removed from labels? (default: relabel_tree = FALSE)
show_label, show_highlights, show_highlight_label, abbr_label	logical (scalar), integer 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: add_legend = TRUE)
layout	layout for the plotted tree. See ggtree for details.
edge_colour_by	Specification of a column metadata field or a feature to colour tree edges by, see the by argument in ?retrieveCellInfo for possible values.
edge_size_by	Specification of a column metadata field or a feature to size tree edges by, see the by argument in ?retrieveCellInfo for possible values.
tip_colour_by	Specification of a column metadata field or a feature to colour tree tips by, see the by argument in ?retrieveCellInfo for possible values.
tip_shape_by	Specification of a column metadata field or a feature to shape tree tips by, see the by argument in ?retrieveCellInfo for possible values.
tip_size_by	Specification of a column metadata field or a feature to size tree tips by, see the by argument in ?retrieveCellInfo for possible values.
node_colour_by	Specification of a column metadata field or a feature to colour tree nodes by. Must be a field from other_fields.
node_shape_by	Specification of a column metadata field or a feature to shape tree nodes by. Must be a field from other_fields.

node_size_by Specification of a column metadata field or a feature to size tree nodes by. Must be a field from other_fields.

colour_highlights_by
Should the highlights be colour differently? If show_highlights = TRUE, colour_highlights will be set to TRUE as default. (default: colour_highlights = FALSE)

by_exprs_values
A string or integer scalar specifying which assay to obtain expression values from, for use in point aesthetics - see the exprs_values argument in ?retrieveCellInfo.

other_fields Additional fields to include in the node information without plotting them.

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

splitByRanks

Examples

```
library(scater)
library(mia)
# preparation of some data
data(GlobalPatterns)
altExps(GlobalPatterns) <- splitByRanks(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 <- getTopTaxa(altExp(GlobalPatterns,"Genus"),
                        method="mean",
                        top=100L,
                        abund_values="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,],
            edge_colour_by = "Phylum",
            edge_size_by = "detected",
            tip_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 <- getTopTaxa(altExp(GlobalPatterns,"Phylum"),
                        method="mean",
                        top=10L,
                        abund_values="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
              y
          })
x <- unsplitByRanks(GlobalPatterns)
x <- addTaxonomyTree(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")

plotRowTree(x[rowData(x)$Phylum %in% top_phyla,],
            edge_colour_by = "Phylum",
            edge_size_by = "detected",
            tip_colour_by = "log_mean",
            node_colour_by = "log_mean")
```

treeData *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) <- value

colTreeData(x) <- value

combineTreeData(x, other_fields = list())

combineTreeData(x, other_fields = list())

## S4 method for signature 'TreeSummarizedExperiment'
colTreeData(x)

## S4 method for signature 'TreeSummarizedExperiment'
rowTreeData(x)

## S4 replacement method for signature 'TreeSummarizedExperiment'
colTreeData(x) <- value

## S4 replacement method for signature 'TreeSummarizedExperiment'
rowTreeData(x) <- value

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

## S4 method for signature 'treedata'
combineTreeData(x, other_fields = list())
```

Arguments

x	a TreeSummarizedExperiment object.
...	additional arguments, currently not used.
other_fields, value	a data.frame or coercible to one, with at least one type of id information. See details.

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