Package 'rbiom'

January 11, 2025

Type Package

Title Read/Write, Analyze, and Visualize 'BIOM' Data

Version 2.0.7

Date 2025-01-10

Description A toolkit for working with Biological Observation Matrix ('BIOM') files. Read/write all 'BIOM' formats. Compute rarefaction, alpha diversity, and beta diversity (including 'UniFrac'). Summarize counts by taxonomic level. Subset based on metadata. Generate visualizations and statistical analyses. CPU intensive operations are coded in C++ for speed.

URL https://cmmr.github.io/rbiom/, https://github.com/cmmr/rbiom

BugReports https://github.com/cmmr/rbiom/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.2.0)

RoxygenNote 7.3.2

Config/Needs/website rmarkdown, phyloseq, npregfast

Config/testthat/edition 3

Config/testthat/parallel false

LinkingTo Rcpp, RcppThread

Imports cluster, methods, mgcv, stats, utils, ape, broom, dplyr, emmeans, fillpattern, ggnewscale, ggplot2, ggtext, jsonlite, magrittr, pillar, plyr, readr, readxl, slam, tidyplots, vegan

Suggests cli, crayon, ggbeeswarm, ggdensity, ggrepel, glue, labeling, lifecycle, openxlsx, optparse, patchwork, pkgconfig, prettycode, R6, rhdf5, rlang, scales, testthat, tibble, tsne, uwot

NeedsCompilation yes

Author Daniel P. Smith [aut, cre] (<https://orcid.org/0000-0002-2479-2044>), Alkek Center for Metagenomics and Microbiome Research [cph, fnd]

Maintainer Daniel P. Smith <dansmith01@gmail.com>

Repository CRAN

Date/Publication 2025-01-11 15:10:02 UTC

Contents

`adiv_boxplot` *Visualize alpha diversity with boxplots.*

Description

Visualize alpha diversity with boxplots.

Usage

```
adiv_boxplot(
  biom,
  x = NULL,
  adiv = "Shannon",
  layers = "x",
  stat.by = x,
  facet.by = NULL,
  colors = TRUE,
  shapes = TRUE,
  patterns = FALSE,
  flip = FALSE,
  stripe = NULL,
  ci = "ci",
  level = 0.95,
  p.adj = "fdr",
  outliers = NULL,
  xlab.angle = "auto",
  p.label = 0.05,
  transform = "none",
  caption = TRUE,
  ...
)
```

Arguments

biom	An rbiom object, such as from as_rbiom(). Any value accepted by as_rbiom() can also be given here.
x	A categorical metadata column name to use for the x-axis. Or NULL, which groups all samples into a single category.
adiv	Alpha diversity metric(s) to use. Options are: "OTUs", "Shannon", "Chao1", "Simpson", and/or "InvSimpson". Set adiv=".all" to use all metrics. Multiple/abbreviated values allowed. Default: "Shannon"
layers	One or more of c("bar", "box" ("x"), "violin", "dot", "strip", "crossbar", "errorbar", "linerange", "pointrange"). Single letter abbreviations are also accepted. For instance, c("box", "dot") is equivalent to c("x", "d") and "xd". Default: "x"
stat.by	Dataset field with the statistical groups. Must be categorical. Default: NULL
facet.by	Dataset field(s) to use for faceting. Must be categorical. Default: NULL
colors	How to color the groups. Options are: TRUE - Automatically select colorblind-friendly colors. FALSE or NULL - Don't use colors. **a palette name -** Auto-select colors from this set. E.g. "okabe" **character vector -** Custom colors to use. E.g. c("red", "#00FF00") **named character vector -** Explicit mapping. E.g. c(Male = "blue", Female = "red") See "Aesthetics" section below for additional information. Default: TRUE
shapes	Shapes for each group. Options are similar to colors's: TRUE, FALSE, NULL, shape names (typically integers 0 - 17), or a named vector mapping groups to specific shape names. See "Aesthetics" section below for additional information. Default: TRUE
patterns	Patterns for each group. Options are similar to colors's: TRUE, FALSE, NULL, pattern names ("brick", "chevron", "fish", "grid", etc), or a named vector mapping groups to specific pattern names. See "Aesthetics" section below for additional information. Default: FALSE
flip	Transpose the axes, so that taxa are present as rows instead of columns. Default: FALSE
stripe	Shade every other x position. Default: *same as flip*
ci	How to calculate min/max of the **crossbar**, **errorbar**, **linerange**, and **pointrange** layers. Options are: "ci" (confidence interval), "range", "sd" (standard deviation), "se" (standard error), and "mad" (median absolute deviation). The center mark of **crossbar** and **pointrange** represents the mean, except for "mad" in which case it represents the median. Default: "ci"
level	The confidence level for calculating a confidence interval. Default: 0.95
p.adj	Method to use for multiple comparisons adjustment of p-values. Run p.adjust.methods for a list of available options. Default: "fdr"
outliers	Show boxplot outliers? TRUE to always show. FALSE to always hide. NULL to only hide them when overlaying a dot or strip chart. Default: NULL

xlab.angle	Angle of the labels at the bottom of the plot. Options are "auto", '0', '30', and '90'. Default: "auto".
p.label	Minimum adjusted p-value to display on the plot with a bracket. p.label = 0.05 - Show p-values that are <= 0.05. p.label = 0 - Don't show any p-values on the plot. p.label = 1 - Show all p-values on the plot. If a numeric vector with more than one value is provided, they will be used as breaks for asterisk notation. Default: 0.05
transform	Transformation to apply. Options are: c("none", "rank", "log", "log1p", "sqrt", "percent"). "rank" is useful for correcting for non-normally distributions before applying regression statistics. Default: "none"
caption	Add methodology caption beneath the plot. Default: TRUE
...	Additional parameters to pass along to ggplot2 functions. Prefix a parameter name with a layer name to pass it to only that layer. For instance, d.size = 2 ensures only the points on the **dot** layer have their size set to 2.

Value

A ggplot2 plot. The computed data points, ggplot2 command, stats table, and stats table commands are available as $data, $code, $stats, and $stats$code, respectively.

Aesthetics

All built-in color palettes are colorblind-friendly. The available categorical palette names are: "okabe", "carto", "r4", "polychrome", "tol", "bright", "light", "muted", "vibrant", "tableau", "classic", "alphabet", "tableau20", "kelly", and "fishy".

Patterns are added using the fillpattern R package. Options are "brick", "chevron", "fish", "grid", "herringbone", "hexagon", "octagon", "rain", "saw", "shingle", "rshingle", "stripe", and "wave", optionally abbreviated and/or suffixed with modifiers. For example, "hex10_sm" for the hexagon pattern rotated 10 degrees and shrunk by 2x. See fillpattern::fill_pattern() for complete documentation of options.

Shapes can be given as per base R - numbers 0 through 17 for various shapes, or the decimal value of an ascii character, e.g. a-z = 65:90; A-Z = 97:122 to use letters instead of shapes on the plot. Character strings may used as well.

See Also

Other alpha_diversity: adiv_corrplot(), adiv_stats(), adiv_table()

Other visualization: adiv_corrplot(), bdiv_boxplot(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), plot_heatmap(), rare_corrplot(), rare_multiplot(), rare_stacked(), stats_boxplot(), stats_corrplot(), taxa_boxplot(), taxa_corrplot(), taxa_heatmap(), taxa_stacked()

Examples

```
library(rbiom)

biom <- rarefy(hmp50)

adiv_boxplot(biom, x="Body Site", stat.by="Body Site")

adiv_boxplot(biom, x="Sex", stat.by="Body Site", adiv=c("otu", "shan"), layers = "bld")

adiv_boxplot(biom, x="body", stat.by="sex", adiv=".all", flip=TRUE, layers="p")


# Each plot object includes additional information.
fig <- adiv_boxplot(biom, x="Body Site")

## Computed Data Points -------------------
fig$data

## Statistics Table -----------------------
fig$stats

## ggplot2 Command ------------------------
fig$code
```

`adiv_corrplot` *Visualize alpha diversity with scatterplots and trendlines.*

Description

Visualize alpha diversity with scatterplots and trendlines.

Usage

```
adiv_corrplot(
  biom,
  x,
  adiv = "Shannon",
  layers = "tc",
  stat.by = NULL,
  facet.by = NULL,
  colors = TRUE,
  shapes = TRUE,
  test = "emmeans",
  fit = "gam",
  at = NULL,
  level = 0.95,
  p.adj = "fdr",
```

```
  transform = "none",
  alt = "!=",
  mu = 0,
  caption = TRUE,
  check = FALSE,
  ...
)
```

Arguments

biom	An rbiom object, such as from as_rbiom(). Any value accepted by as_rbiom() can also be given here.
x	Dataset field with the x-axis values. Equivalent to the regr argument in stats_table(). Required.
adiv	Alpha diversity metric(s) to use. Options are: "OTUs", "Shannon", "Chao1", "Simpson", and/or "InvSimpson". Set adiv=".all" to use all metrics. Multiple/abbreviated values allowed. Default: "Shannon"
layers	One or more of c("trend", "confidence", "point", "name", "residual"). Single letter abbreviations are also accepted. For instance, c("trend", "point") is equivalent to c("t", "p") and "tp". Default: "tc"
stat.by	Dataset field with the statistical groups. Must be categorical. Default: NULL
facet.by	Dataset field(s) to use for faceting. Must be categorical. Default: NULL
colors	How to color the groups. Options are: TRUE - Automatically select colorblind-friendly colors. FALSE or NULL - Don't use colors. **a palette name -** Auto-select colors from this set. E.g. "okabe" **character vector -** Custom colors to use. E.g. c("red", "#00FF00") **named character vector -** Explicit mapping. E.g. c(Male = "blue", Female = "red") See "Aesthetics" section below for additional information. Default: TRUE
shapes	Shapes for each group. Options are similar to colors's: TRUE, FALSE, NULL, shape names (typically integers 0 - 17), or a named vector mapping groups to specific shape names. See "Aesthetics" section below for additional information. Default: TRUE
test	Method for computing p-values: 'none', 'emmeans', or 'emtrends'. Default: 'emmeans'
fit	How to fit the trendline. 'lm', 'log', or 'gam'. Default: 'gam'
at	Position(s) along the x-axis where the means or slopes should be evaluated. Default: NULL, which samples 100 evenly spaced positions and selects the position where the p-value is most significant.
level	The confidence level for calculating a confidence interval. Default: 0.95
p.adj	Method to use for multiple comparisons adjustment of p-values. Run p.adjust.methods for a list of available options. Default: "fdr"

transform	Transformation to apply. Options are: c("none", "rank", "log", "log1p", "sqrt", "percent"). "rank" is useful for correcting for non-normally distributions before applying regression statistics. Default: "none"
alt	Alternative hypothesis direction. Options are '!=' (two-sided; not equal to mu), '<' (less than mu), or '>' (greater than mu). Default: '!='
mu	Reference value to test against. Default: 0
caption	Add methodology caption beneath the plot. Default: TRUE
check	Generate additional plots to aid in assessing data normality. Default: FALSE
...	Additional parameters to pass along to ggplot2 functions. Prefix a parameter name with a layer name to pass it to only that layer. For instance, p.size = 2 ensures only the points have their size set to 2.

Value

A ggplot2 plot. The computed data points, ggplot2 command, stats table, and stats table commands are available as $data, $code, $stats, and $stats$code, respectively.

Aesthetics

All built-in color palettes are colorblind-friendly. The available categorical palette names are: "okabe", "carto", "r4", "polychrome", "tol", "bright", "light", "muted", "vibrant", "tableau", "classic", "alphabet", "tableau20", "kelly", and "fishy".

Shapes can be given as per base R - numbers 0 through 17 for various shapes, or the decimal value of an ascii character, e.g. a-z = 65:90; A-Z = 97:122 to use letters instead of shapes on the plot. Character strings may used as well.

See Also

Other alpha_diversity: adiv_boxplot(), adiv_stats(), adiv_table()

Other visualization: adiv_boxplot(), bdiv_boxplot(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), plot_heatmap(), rare_corrplot(), rare_multiplot(), rare_stacked(), stats_boxplot(), stats_corrplot(), taxa_boxplot(), taxa_corrplot(), taxa_heatmap(), taxa_stacked()

Examples

```
library(rbiom)

p <- adiv_corrplot(babies, "age", stat.by = "deliv", fit = "gam")

p

p$stats

p$code
```

`adiv_matrix` *Create a matrix of samples x alpha diversity metrics.*

Description

Create a matrix of samples x alpha diversity metrics.

Usage

```
adiv_matrix(biom, transform = "none")
```

Arguments

`biom`	An rbiom object, such as from `as_rbiom()`. Any value accepted by `as_rbiom()` can also be given here.
`transform`	Transformation to apply. Options are: `c("none", "rank", "log", "log1p", "sqrt", "percent")`. `"rank"` is useful for correcting for non-normally distributions before applying regression statistics. Default: `"none"`

Value

A numeric matrix with samples as rows and columns named **Depth**, **OTUs**, **Shannon**, **Chao1**, **Simpson**, and **InvSimpson**.

Examples

```
library(rbiom)

biom <- slice_head(hmp50, n = 5)

adiv_matrix(biom)
```

`adiv_stats` *Test alpha diversity for associations with metadata.*

Description

A convenience wrapper for `adiv_table()` + `stats_table()`.

Usage

```
adiv_stats(
  biom,
  regr = NULL,
  stat.by = NULL,
  adiv = "Shannon",
  split.by = NULL,
  transform = "none",
  test = "emmeans",
  fit = "gam",
  at = NULL,
  level = 0.95,
  alt = "!=",
  mu = 0,
  p.adj = "fdr"
)
```

Arguments

biom	An rbiom object, such as from as_rbiom(). Any value accepted by as_rbiom() can also be given here.
regr	Dataset field with the x-axis (independent; predictive) values. Must be numeric. Default: NULL
stat.by	Dataset field with the statistical groups. Must be categorical. Default: NULL
adiv	Alpha diversity metric(s) to use. Options are: "OTUs", "Shannon", "Chao1", "Simpson", and/or "InvSimpson". Set adiv=".all" to use all metrics. Multiple/abbreviated values allowed. Default: "Shannon"
split.by	Dataset field(s) that the data should be split by prior to any calculations. Must be categorical. Default: NULL
transform	Transformation to apply. Options are: c("none", "rank", "log", "log1p", "sqrt", "percent"). "rank" is useful for correcting for non-normally distributions before applying regression statistics. Default: "none"
test	Method for computing p-values: 'wilcox', 'kruskal', 'emmeans', or 'emtrends'. Default: 'emmeans'
fit	How to fit the trendline. 'lm', 'log', or 'gam'. Default: 'gam'
at	Position(s) along the x-axis where the means or slopes should be evaluated. Default: NULL, which samples 100 evenly spaced positions and selects the position where the p-value is most significant.
level	The confidence level for calculating a confidence interval. Default: 0.95
alt	Alternative hypothesis direction. Options are '!=' (two-sided; not equal to mu), '<' (less than mu), or '>' (greater than mu). Default: '!='
mu	Reference value to test against. Default: 0
p.adj	Method to use for multiple comparisons adjustment of p-values. Run p.adjust.methods for a list of available options. Default: "fdr"

Value

A tibble data.frame with fields from the table below. This tibble object provides the `$code` operator to print the R code used to generate the statistics.

Field	Description
.mean	Estimated marginal mean. See `emmeans::emmeans()`.
.mean.diff	Difference in means.
.slope	Trendline slope. See `emmeans::emtrends()`.
.slope.diff	Difference in slopes.
.h1	Alternate hypothesis.
.p.val	Probability that null hypothesis is correct.
.adj.p	`.p.val` after adjusting for multiple comparisons.
.effect.size	Effect size. See `emmeans::eff_size()`.
.lower	Confidence interval lower bound.
.upper	Confidence interval upper bound.
.se	Standard error.
.n	Number of samples.
.df	Degrees of freedom.
.stat	Wilcoxon or Kruskal-Wallis rank sum statistic.
.t.ratio	`.mean` / `.se`
.r.sqr	Percent of variation explained by the model.
.adj.r	`.r.sqr`, taking degrees of freedom into account.
.aic	Akaike Information Criterion (predictive models).
.bic	Bayesian Information Criterion (descriptive models).
.loglik	Log-likelihood goodness-of-fit score.
.fit.p	P-value for observing this fit by chance.

See Also

Other alpha_diversity: `adiv_boxplot()`, `adiv_corrplot()`, `adiv_table()`

Other stats_tables: `bdiv_stats()`, `distmat_stats()`, `stats_table()`, `taxa_stats()`

Examples

```
library(rbiom)

biom <- rarefy(hmp50)

adiv_stats(biom, stat.by = "Sex")[,1:6]

adiv_stats(biom, stat.by = "Sex", split.by = "Body Site")[,1:6]

adiv_stats(biom, stat.by = "Body Site", test = "kruskal")
```

adiv_table *Calculate the alpha diversity of each sample.*

Description

Calculate the alpha diversity of each sample.

Usage

```
adiv_table(biom, adiv = "Shannon", md = ".all", transform = "none")
```

Arguments

biom	An rbiom object, such as from as_rbiom(). Any value accepted by as_rbiom() can also be given here.
adiv	Alpha diversity metric(s) to use. Options are: "OTUs", "Shannon", "Chao1", "Simpson", and/or "InvSimpson". Set adiv=".all" to use all metrics. Multiple/abbreviated values allowed. Default: "Shannon"
md	Dataset field(s) to include in the output data frame, or '.all' to include all metadata fields. Default: '.all'
transform	Transformation to apply. Options are: c("none", "rank", "log", "log1p", "sqrt", "percent"). "rank" is useful for correcting for non-normally distributions before applying regression statistics. Default: "none"

Value

A data frame of alpha diversity values. Each combination of sample/depth/adiv has its own row. Column names are **.sample**, **.depth**, **.adiv**, and **.diversity**, followed by any metadata fields requested by md.

See Also

Other alpha_diversity: adiv_boxplot(), adiv_corrplot(), adiv_stats()

Examples

```
library(rbiom)

# Subset to 10 samples.
biom <- slice(hmp50, 1:10)
adiv_table(biom)

biom <- rarefy(biom)
adiv_table(biom, adiv = ".all", md = NULL)
```

as.list.rbiom *Convert an rbiom object to a base R list.*

Description

Convert an rbiom object to a base R list.

Usage

```
## S3 method for class 'rbiom'
as.list(x, ...)
```

Arguments

x	An rbiom object, such as from as_rbiom().
...	Not used.

Value

A list with names c('counts', 'metadata', 'taxonomy', 'tree', 'sequences', 'id', 'comment', 'date', 'generated_by').

See Also

Other conversion: as.matrix.rbiom()

as.matrix.rbiom *Convert an rbiom object to a simple count matrix.*

Description

Identical to running as.matrix(biom$counts).

Usage

```
## S3 method for class 'rbiom'
as.matrix(x, ...)
```

Arguments

x	An rbiom object, such as from as_rbiom().
...	Not used.

Value

A base R matrix with OTUs as rows and samples as columns.

See Also

Other conversion: `as.list.rbiom()`

Examples

```
library(rbiom)

as.matrix(hmp50)[1:5,1:5]
```

`as_rbiom` *Convert a variety of data types to an rbiom object.*

Description

Construct an rbiom object. The returned object is an R6 reference class. Use `b <- a$clone()` to create copies, not `b <- a`.

Usage

```
as_rbiom(biom, ...)
```

Arguments

`biom`	Object which can be coerced to an rbiom-class object. For example: ***file* -** Filepath or URL to a biom file. ***matrix* -** An abundance matrix with OTUs in rows and samples in columns. `phyloseq`**-class object -** From the phyloseq Bioconductor R package. ***list* -** With `counts` and optionally `metadata`, `taxonomy`, `tree`, etc (see details).
`...`	Properties to overwrite in biom: `metadata`, `taxonomy`, `tree`, etc (see details).

Value

An rbiom object.

Examples

```
library(rbiom)

# create a simple matrix ------------------------
mtx <- matrix(
  data     = floor(runif(24) * 1000),
  nrow     = 6,
  dimnames = list(paste0("OTU", 1:6), paste0("Sample", 1:4)) )
mtx

# and some sample metadata ----------------------
```

```
df <- data.frame(
  .sample   = paste0("Sample", 1:4),
  treatment = c("A", "B", "A", "B"),
  days      = c(12, 3, 7, 8) )

# convert data set to rbiom ---------------------
biom <- as_rbiom(mtx, metadata = df, id = "My BIOM")
biom
```

`babies` *Longitudinal Stool Samples from Infants (n = 2,684)*

Description

Longitudinal Stool Samples from Infants (n = 2,684)

Usage

```
babies
```

Format

An rbiom object with 2,684 samples. Includes metadata and taxonomy.

Subject ID - ID1, ID2, ..., ID12

Sex - Male or Female

Age (days) - 1 - 266

Child's diet - "Breast milk", "Breast milk and formula", or "Formula"

Sample collection - "Frozen upon collection" or "Stored in alcohol"

Antibiotic exposure - Yes or No

Antifungal exposure - Yes or No

Delivery mode - Cesarean or Vaginal

Solid food introduced (Age) - 116 - 247

Source

https://www.nature.com/articles/s41467-018-04641-7 and doi:10.1038/s41467017019738

bdiv_boxplot *Visualize BIOM data with boxplots.*

Description

Visualize BIOM data with boxplots.

Usage

```
bdiv_boxplot(
  biom,
  x = NULL,
  bdiv = "Bray-Curtis",
  layers = "x",
  weighted = TRUE,
  tree = NULL,
  within = NULL,
  between = NULL,
  stat.by = x,
  facet.by = NULL,
  colors = TRUE,
  shapes = TRUE,
  patterns = FALSE,
  flip = FALSE,
  stripe = NULL,
  ci = "ci",
  level = 0.95,
  p.adj = "fdr",
  outliers = NULL,
  xlab.angle = "auto",
  p.label = 0.05,
  transform = "none",
  caption = TRUE,
  ...
)
```

Arguments

biom	An rbiom object, such as from as_rbiom(). Any value accepted by as_rbiom() can also be given here.
x	A categorical metadata column name to use for the x-axis. Or NULL, which groups all samples into a single category.
bdiv	Beta diversity distance algorithm(s) to use. Options are: "Bray-Curtis", "Manhattan", "Euclidean", "Jaccard", and "UniFrac". For "UniFrac", a phylogenetic tree must be present in biom or explicitly provided via tree=. Multiple/abbreviated values allowed. Default: "Bray-Curtis"

layers — One or more of c("bar", "box" ("x"), "violin", "dot", "strip", "crossbar", "errorbar", "linerange", "pointrange"). Single letter abbreviations are also accepted. For instance, c("box", "dot") is equivalent to c("x", "d") and "xd". Default: "x"

weighted — Take relative abundances into account. When weighted=FALSE, only presence/absence is considered. Multiple values allowed. Default: TRUE

tree — A phylo object representing the phylogenetic relationships of the taxa in biom. Only required when computing UniFrac distances. Default: biom$tree

within, between — Dataset field(s) for intra- or inter- sample comparisons. Alternatively, dataset field names given elsewhere can be prefixed with '==' or '!=' to assign them to within or between, respectively. Default: NULL

stat.by — Dataset field with the statistical groups. Must be categorical. Default: NULL

facet.by — Dataset field(s) to use for faceting. Must be categorical. Default: NULL

colors — How to color the groups. Options are:

TRUE **-** Automatically select colorblind-friendly colors.

FALSE **or** NULL **-** Don't use colors.

a palette name - Auto-select colors from this set. E.g. "okabe"

character vector - Custom colors to use. E.g. c("red", "#00FF00")

named character vector - Explicit mapping. E.g. c(Male = "blue", Female = "red")

See "Aesthetics" section below for additional information. Default: TRUE

shapes — Shapes for each group. Options are similar to colors's: TRUE, FALSE, NULL, shape names (typically integers 0 - 17), or a named vector mapping groups to specific shape names. See "Aesthetics" section below for additional information. Default: TRUE

patterns — Patterns for each group. Options are similar to colors's: TRUE, FALSE, NULL, pattern names ("brick", "chevron", "fish", "grid", etc), or a named vector mapping groups to specific pattern names. See "Aesthetics" section below for additional information. Default: FALSE

flip — Transpose the axes, so that taxa are present as rows instead of columns. Default: FALSE

stripe — Shade every other x position. Default: *same as flip*

ci — How to calculate min/max of the **crossbar**, **errorbar**, **linerange**, and **pointrange** layers. Options are: "ci" (confidence interval), "range", "sd" (standard deviation), "se" (standard error), and "mad" (median absolute deviation). The center mark of **crossbar** and **pointrange** represents the mean, except for "mad" in which case it represents the median. Default: "ci"

level — The confidence level for calculating a confidence interval. Default: 0.95

p.adj — Method to use for multiple comparisons adjustment of p-values. Run p.adjust.methods for a list of available options. Default: "fdr"

outliers — Show boxplot outliers? TRUE to always show. FALSE to always hide. NULL to only hide them when overlaying a dot or strip chart. Default: NULL

xlab.angle	Angle of the labels at the bottom of the plot. Options are "auto", '0', '30', and '90'. Default: "auto".
p.label	Minimum adjusted p-value to display on the plot with a bracket. p.label = 0.05 - Show p-values that are <= 0.05. p.label = 0 - Don't show any p-values on the plot. p.label = 1 - Show all p-values on the plot. If a numeric vector with more than one value is provided, they will be used as breaks for asterisk notation. Default: 0.05
transform	Transformation to apply. Options are: c("none", "rank", "log", "log1p", "sqrt", "percent"). "rank" is useful for correcting for non-normally distributions before applying regression statistics. Default: "none"
caption	Add methodology caption beneath the plot. Default: TRUE
...	Additional parameters to pass along to ggplot2 functions. Prefix a parameter name with a layer name to pass it to only that layer. For instance, d.size = 2 ensures only the points on the **dot** layer have their size set to 2.

Value

A ggplot2 plot. The computed data points, ggplot2 command, stats table, and stats table commands are available as $data, $code, $stats, and $stats$code, respectively.

Aesthetics

All built-in color palettes are colorblind-friendly. The available categorical palette names are: "okabe", "carto", "r4", "polychrome", "tol", "bright", "light", "muted", "vibrant", "tableau", "classic", "alphabet", "tableau20", "kelly", and "fishy".

Patterns are added using the fillpattern R package. Options are "brick", "chevron", "fish", "grid", "herringbone", "hexagon", "octagon", "rain", "saw", "shingle", "rshingle", "stripe", and "wave", optionally abbreviated and/or suffixed with modifiers. For example, "hex10_sm" for the hexagon pattern rotated 10 degrees and shrunk by 2x. See fillpattern::fill_pattern() for complete documentation of options.

Shapes can be given as per base R - numbers 0 through 17 for various shapes, or the decimal value of an ascii character, e.g. a-z = 65:90; A-Z = 97:122 to use letters instead of shapes on the plot. Character strings may used as well.

See Also

Other beta_diversity: bdiv_clusters(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), bdiv_ord_table(), bdiv_stats(), bdiv_table(), distmat_stats()

Other visualization: adiv_boxplot(), adiv_corrplot(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), plot_heatmap(), rare_corrplot(), rare_multiplot(), rare_stacked(), stats_boxplot(), stats_corrplot(), taxa_boxplot(), taxa_corrplot(), taxa_heatmap(), taxa_stacked()

Examples

```
library(rbiom)

biom <- rarefy(hmp50)

bdiv_boxplot(biom, x="==Body Site", bdiv="UniFrac", stat.by="Body Site")
```

`bdiv_clusters` *Define sample PAM clusters from beta diversity.*

Description

Define sample PAM clusters from beta diversity.

Usage

```
bdiv_clusters(
  biom,
  bdiv = "Bray-Curtis",
  weighted = TRUE,
  tree = NULL,
  k = 5,
  ...
)
```

Arguments

`biom`	An rbiom object, such as from `as_rbiom()`. Any value accepted by `as_rbiom()` can also be given here.
`bdiv`	Beta diversity distance algorithm(s) to use. Options are: `"Bray-Curtis"`, `"Manhattan"`, `"Euclidean"`, `"Jaccard"`, and `"UniFrac"`. For `"UniFrac"`, a phylogenetic tree must be present in `biom` or explicitly provided via `tree=`. Multiple/abbreviated values allowed. Default: `"Bray-Curtis"`
`weighted`	Take relative abundances into account. When `weighted=FALSE`, only presence/absence is considered. Multiple values allowed. Default: `TRUE`
`tree`	A `phylo` object representing the phylogenetic relationships of the taxa in `biom`. Only required when computing UniFrac distances. Default: `biom$tree`
`k`	Number of clusters. Default: `5L`
`...`	Passed on to `cluster::pam()`.

Value

A numeric factor assigning samples to clusters.

See Also

Other beta_diversity: bdiv_boxplot(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), bdiv_ord_table(), bdiv_stats(), bdiv_table(), distmat_stats()

Other clustering: taxa_clusters()

Examples

```
library(rbiom)

biom <- rarefy(hmp50)
biom$metadata$bray_cluster <- bdiv_clusters(biom)

pull(biom, 'bray_cluster')[1:10]

bdiv_ord_plot(biom, stat.by = "bray_cluster")
```

---

bdiv_corrplot *Visualize beta diversity with scatterplots and trendlines.*

Description

Visualize beta diversity with scatterplots and trendlines.

Usage

```
bdiv_corrplot(
  biom,
  x,
  bdiv = "Bray-Curtis",
  layers = "tc",
  weighted = TRUE,
  tree = NULL,
  within = NULL,
  between = NULL,
  stat.by = NULL,
  facet.by = NULL,
  colors = TRUE,
  shapes = TRUE,
  test = "emmeans",
  fit = "gam",
  at = NULL,
  level = 0.95,
  p.adj = "fdr",
  transform = "none",
  ties = "random",
  seed = 0,
  alt = "!=",
```

```
  mu = 0,
  caption = TRUE,
  check = FALSE,
  ...
)
```

Arguments

biom	An rbiom object, such as from as_rbiom(). Any value accepted by as_rbiom() can also be given here.
x	Dataset field with the x-axis values. Equivalent to the regr argument in stats_table(). Required.
bdiv	Beta diversity distance algorithm(s) to use. Options are: "Bray-Curtis", "Manhattan", "Euclidean", "Jaccard", and "UniFrac". For "UniFrac", a phylogenetic tree must be present in biom or explicitly provided via tree=. Multiple/abbreviated values allowed. Default: "Bray-Curtis"
layers	One or more of c("trend", "confidence", "point", "name", "residual"). Single letter abbreviations are also accepted. For instance, c("trend", "point") is equivalent to c("t", "p") and "tp". Default: "tc"
weighted	Take relative abundances into account. When weighted=FALSE, only presence/absence is considered. Multiple values allowed. Default: TRUE
tree	A phylo object representing the phylogenetic relationships of the taxa in biom. Only required when computing UniFrac distances. Default: biom$tree
within, between	Dataset field(s) for intra- or inter- sample comparisons. Alternatively, dataset field names given elsewhere can be prefixed with '==' or '!=' to assign them to within or between, respectively. Default: NULL
stat.by	Dataset field with the statistical groups. Must be categorical. Default: NULL
facet.by	Dataset field(s) to use for faceting. Must be categorical. Default: NULL
colors	How to color the groups. Options are: TRUE - Automatically select colorblind-friendly colors. FALSE or NULL - Don't use colors. **a palette name -** Auto-select colors from this set. E.g. "okabe" **character vector -** Custom colors to use. E.g. c("red", "#00FF00") **named character vector -** Explicit mapping. E.g. c(Male = "blue", Female = "red") See "Aesthetics" section below for additional information. Default: TRUE
shapes	Shapes for each group. Options are similar to colors's: TRUE, FALSE, NULL, shape names (typically integers 0 - 17), or a named vector mapping groups to specific shape names. See "Aesthetics" section below for additional information. Default: TRUE
test	Method for computing p-values: 'none', 'emmeans', or 'emtrends'. Default: 'emmeans'
fit	How to fit the trendline. 'lm', 'log', or 'gam'. Default: 'gam'

at	Position(s) along the x-axis where the means or slopes should be evaluated. Default: NULL, which samples 100 evenly spaced positions and selects the position where the p-value is most significant.
level	The confidence level for calculating a confidence interval. Default: 0.95
p.adj	Method to use for multiple comparisons adjustment of p-values. Run p.adjust.methods for a list of available options. Default: "fdr"
transform	Transformation to apply. Options are: c("none", "rank", "log", "log1p", "sqrt", "percent"). "rank" is useful for correcting for non-normally distributions before applying regression statistics. Default: "none"
ties	When transform="rank", how to rank identical values. Options are: c("average", "first", "last", "random", "max", "min"). See rank() for details. Default: "random"
seed	Random seed for permutations. Default: 0
alt	Alternative hypothesis direction. Options are '!=' (two-sided; not equal to mu), '<' (less than mu), or '>' (greater than mu). Default: '!='
mu	Reference value to test against. Default: 0
caption	Add methodology caption beneath the plot. Default: TRUE
check	Generate additional plots to aid in assessing data normality. Default: FALSE
...	Additional parameters to pass along to ggplot2 functions. Prefix a parameter name with a layer name to pass it to only that layer. For instance, p.size = 2 ensures only the points have their size set to 2.

Value

A ggplot2 plot. The computed data points, ggplot2 command, stats table, and stats table commands are available as $data, $code, $stats, and $stats$code, respectively.

Aesthetics

All built-in color palettes are colorblind-friendly. The available categorical palette names are: "okabe", "carto", "r4", "polychrome", "tol", "bright", "light", "muted", "vibrant", "tableau", "classic", "alphabet", "tableau20", "kelly", and "fishy".

Shapes can be given as per base R - numbers 0 through 17 for various shapes, or the decimal value of an ascii character, e.g. a-z = 65:90; A-Z = 97:122 to use letters instead of shapes on the plot. Character strings may used as well.

See Also

Other beta_diversity: bdiv_boxplot(), bdiv_clusters(), bdiv_heatmap(), bdiv_ord_plot(), bdiv_ord_table(), bdiv_stats(), bdiv_table(), distmat_stats()

Other visualization: adiv_boxplot(), adiv_corrplot(), bdiv_boxplot(), bdiv_heatmap(), bdiv_ord_plot(), plot_heatmap(), rare_corrplot(), rare_multiplot(), rare_stacked(), stats_boxplot(), stats_corrplot(), taxa_boxplot(), taxa_corrplot(), taxa_heatmap(), taxa_stacked()

Examples

```
library(rbiom)

biom <- rarefy(hmp50)
bdiv_corrplot(biom, "Age", stat.by = "Sex", layers = "tcp")
```

`bdiv_heatmap` *Display beta diversities in an all vs all grid.*

Description

Display beta diversities in an all vs all grid.

Usage

```
bdiv_heatmap(
  biom,
  bdiv = "Bray-Curtis",
  weighted = TRUE,
  tree = NULL,
  tracks = NULL,
  grid = "devon",
  label = TRUE,
  label_size = NULL,
  rescale = "none",
  clust = "complete",
  trees = TRUE,
  asp = 1,
  tree_height = 10,
  track_height = 10,
  legend = "right",
  title = TRUE,
  xlab.angle = "auto",
  ...
)
```

Arguments

`biom`	An rbiom object, such as from `as_rbiom()`. Any value accepted by `as_rbiom()` can also be given here.
`bdiv`	Beta diversity distance algorithm(s) to use. Options are: `"Bray-Curtis"`, `"Manhattan"`, `"Euclidean"`, `"Jaccard"`, and `"UniFrac"`. For `"UniFrac"`, a phylogenetic tree must be present in `biom` or explicitly provided via `tree=`. Multiple/abbreviated values allowed. Default: `"Bray-Curtis"`
`weighted`	Take relative abundances into account. When `weighted=FALSE`, only presence/absence is considered. Multiple values allowed. Default: `TRUE`

`tree` A phylo object representing the phylogenetic relationships of the taxa in biom. Only required when computing UniFrac distances. Default: `biom$tree`

`tracks` A character vector of metadata fields to display as tracks at the top of the plot. Or, a list as expected by the tracks argument of `plot_heatmap()`. Default: NULL

`grid` Color palette name, or a list with entries for label, colors, range, bins, na.color, and/or guide. See the Track Definitions section for details. Default: "devon"

`label` Label the matrix rows and columns. You can supply a list or logical vector of length two to control row labels and column labels separately, for example label = c(rows = TRUE, cols = FALSE), or simply label = c(TRUE, FALSE). Other valid options are "rows", "cols", "both", "bottom", "right", and "none". Default: TRUE

`label_size` The font size to use for the row and column labels. You can supply a numeric vector of length two to control row label sizes and column label sizes separately, for example c(rows = 20, cols = 8), or simply c(20, 8). Default: NULL, which computes: pmax(8, pmin(20, 100 / dim(mtx)))

`rescale` Rescale rows or columns to all have a common min/max. Options: "none", "rows", or "cols". Default: "none"

`clust` Clustering algorithm for reordering the rows and columns by similarity. You can supply a list or character vector of length two to control the row and column clustering separately, for example clust = c(rows = "complete", cols = NA), or simply clust = c("complete", NA). Options are:

FALSE **or** NA **-** Disable reordering.

An hclust **class object** E.g. from `stats::hclust()`.

A method name - "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median", or "centroid".

Default: "complete"

`trees` Draw a dendrogram for rows (left) and columns (top). You can supply a list or logical vector of length two to control the row tree and column tree separately, for example trees = c(rows = TRUE, cols = FALSE), or simply trees = c(TRUE, FALSE). Other valid options are "rows", "cols", "both", "left", "top", and "none". Default: TRUE

`asp` Aspect ratio (height/width) for entire grid. Default: 1 (square)

`tree_height, track_height`
The height of the dendrogram or annotation tracks as a percentage of the overall grid size. Use a numeric vector of length two to assign c(top, left) independently. Default: 10 (10% of the grid's height)

`legend` Where to place the legend. Options are: "right" or "bottom". Default: "right"

`title` Plot title. Set to TRUE for a default title, NULL for no title, or any character string. Default: TRUE

`xlab.angle` Angle of the labels at the bottom of the plot. Options are "auto", '0', '30', and '90'. Default: "auto".

`...` Additional arguments to pass on to ggplot2::theme(). For example, labs.subtitle = "Plot subtitle".

Value

A ggplot2 plot. The computed data points and ggplot command are available as $data and $code, respectively.

Annotation Tracks

Metadata can be displayed as colored tracks above the heatmap. Common use cases are provided below, with more thorough documentation available at https://cmmr.github.io/rbiom .

```
## Categorical ----------------------------
tracks = "Body Site"
tracks = list('Body Site' = "bright")
tracks = list('Body Site' = c('Stool' = "blue", 'Saliva' = "green"))

## Numeric --------------------------------
tracks = "Age"
tracks = list('Age' = "reds")

## Multiple Tracks ------------------------
tracks = c("Body Site", "Age")
tracks = list('Body Site' = "bright", 'Age' = "reds")
tracks = list(
  'Body Site' = c('Stool' = "blue", 'Saliva' = "green"),
  'Age'       = list('colors' = "reds") )
```

The following entries in the track definitions are understood:

`colors` - A pre-defined palette name or custom set of colors to map to.

`range` - The c(min,max) to use for scale values.

`label` - Label for this track. Defaults to the name of this list element.

`side` - Options are `"top"` (default) or `"left"`.

`na.color` - The color to use for `NA` values.

`bins` - Bin a gradient into this many bins/steps.

`guide` - A list of arguments for guide_colorbar() or guide_legend().

All built-in color palettes are colorblind-friendly.

Categorical palette names: `"okabe"`, `"carto"`, `"r4"`, `"polychrome"`, `"tol"`, `"bright"`, `"light"`, `"muted"`, `"vibrant"`, `"tableau"`, `"classic"`, `"alphabet"`, `"tableau20"`, `"kelly"`, and `"fishy"`.

Numeric palette names: `"reds"`, `"oranges"`, `"greens"`, `"purples"`, `"grays"`, `"acton"`, `"bamako"`, `"batlow"`, `"bilbao"`, `"buda"`, `"davos"`, `"devon"`, `"grayC"`, `"hawaii"`, `"imola"`, `"lajolla"`, `"lapaz"`, `"nuuk"`, `"oslo"`, `"tokyo"`, `"turku"`, `"bam"`, `"berlin"`, `"broc"`, `"cork"`, `"lisbon"`, `"roma"`, `"tofino"`, `"vanimo"`, and `"vik"`.

See Also

Other beta_diversity: bdiv_boxplot(), bdiv_clusters(), bdiv_corrplot(), bdiv_ord_plot(), bdiv_ord_table(), bdiv_stats(), bdiv_table(), distmat_stats()

Other visualization: adiv_boxplot(), adiv_corrplot(), bdiv_boxplot(), bdiv_corrplot(), bdiv_ord_plot(), plot_heatmap(), rare_corrplot(), rare_multiplot(), rare_stacked(), stats_boxplot(), stats_corrplot(), taxa_boxplot(), taxa_corrplot(), taxa_heatmap(), taxa_stacked()

Examples

```
library(rbiom)

# Keep and rarefy the 10 most deeply sequenced samples.
hmp10 <- rarefy(hmp50, n = 10)

bdiv_heatmap(hmp10, tracks=c("Body Site", "Age"))

bdiv_heatmap(hmp10, bdiv="uni", weighted=c(TRUE,FALSE), tracks="sex")
```

`bdiv_ord_plot` *Ordinate samples and taxa on a 2D plane based on beta diversity distances.*

Description

Ordinate samples and taxa on a 2D plane based on beta diversity distances.

Usage

```
bdiv_ord_plot(
  biom,
  bdiv = "Bray-Curtis",
  ord = "PCoA",
  weighted = TRUE,
  layers = "petm",
  stat.by = NULL,
  facet.by = NULL,
  colors = TRUE,
  shapes = TRUE,
  tree = NULL,
  test = "adonis2",
  seed = 0,
  permutations = 999,
  rank = -1,
  taxa = 4,
  p.top = Inf,
  p.adj = "fdr",
```

```
  unc = "singly",
  caption = TRUE,
  ...
)
```

Arguments

biom	An rbiom object, such as from as_rbiom(). Any value accepted by as_rbiom() can also be given here.
bdiv	Beta diversity distance algorithm(s) to use. Options are: "Bray-Curtis", "Manhattan", "Euclidean", "Jaccard", and "UniFrac". For "UniFrac", a phylogenetic tree must be present in biom or explicitly provided via tree=. Multiple/abbreviated values allowed. Default: "Bray-Curtis"
ord	Method for reducing dimensionality. Options are: "PCoA" - Principal coordinate analysis; ape::pcoa(). "UMAP" - Uniform manifold approximation and projection; uwot::umap(). "NMDS" - Nonmetric multidimensional scaling; vegan::metaMDS(). "tSNE" - t-distributed stochastic neighbor embedding; tsne::tsne(). Multiple/abbreviated values allowed. Default: "PCoA"
weighted	Take relative abundances into account. When weighted=FALSE, only presence/absence is considered. Multiple values allowed. Default: TRUE
layers	One or more of c("point", "spider", "ellipse", "name", "mean", "taxon", "arrow"). The first four are sample-centric; the last three are taxa-centric. Single letter abbreviations are also accepted. For instance, c("point", "ellipse") is equivalent to c("p", "e") and "pe". Default: "pe"
stat.by	The categorical or numeric metadata field over which statistics should be calculated. Required.
facet.by	Dataset field(s) to use for faceting. Must be categorical. Default: NULL
colors	How to color the groups. Options are: TRUE - Automatically select colorblind-friendly colors. FALSE or NULL - Don't use colors. **a palette name -** Auto-select colors from this set. E.g. "okabe" **character vector -** Custom colors to use. E.g. c("red", "#00FF00") **named character vector -** Explicit mapping. E.g. c(Male = "blue", Female = "red") See "Aesthetics" section below for additional information. Default: TRUE
shapes	Shapes for each group. Options are similar to colors's: TRUE, FALSE, NULL, shape names (typically integers 0 - 17), or a named vector mapping groups to specific shape names. See "Aesthetics" section below for additional information. Default: TRUE
tree	A phylo object representing the phylogenetic relationships of the taxa in biom. Only required when computing UniFrac distances. Default: biom$tree
test	Permutational test for accessing significance. Options are:

"adonis2" - Permutational MANOVA; vegan::adonis2().

"mrpp" - Multiple response permutation procedure; vegan::mrpp().

"none" - Don't run any statistics.

Abbreviations are allowed. Default: "adonis2"

seed — Random seed for permutations. Default: 0

permutations — Number of random permutations to use. Default: 999

rank — What rank(s) of taxa to display. E.g. "Phylum", "Genus", ".otu", etc. An integer vector can also be given, where 1 is the highest rank, 2 is the second highest, -1 is the lowest rank, -2 is the second lowest, and 0 is the OTU "rank". Run biom$ranks to see all options for a given rbiom object. Default: -1.

taxa — Which taxa to display. An integer value will show the top n most abundant taxa. A value 0 <= n < 1 will show any taxa with that mean abundance or greater (e.g. 0.1 implies >= 10%). A character vector of taxa names will show only those named taxa. Default: 6.

p.top — Only display taxa with the most significant differences in abundance. If p.top is >= 1, then the p.top most significant taxa are displayed. If p.top is less than one, all taxa with an adjusted p-value <= p.top are displayed. Recommended to be used in combination with the taxa parameter to set a lower bound on the mean abundance of considered taxa. Default: Inf

p.adj — Method to use for multiple comparisons adjustment of p-values. Run p.adjust.methods for a list of available options. Default: "fdr"

unc — How to handle unclassified, uncultured, and similarly ambiguous taxa names. Options are:

"singly" - Replaces them with the OTU name.

"grouped" - Replaces them with a higher rank's name.

"drop" - Excludes them from the result.

"asis" - To not check/modify any taxa names.

Abbreviations are allowed. Default: "singly"

caption — Add methodology caption beneath the plot. Default: TRUE

... — Parameters for layer geoms (e.g. ggplot2::geom_point()). Prefixing parameter names with a layer name ensures that a particular parameter is passed to, and only to, that layer. For instance, point.size = 2 or p.size = 2 ensures only the points have their size set to 2. Points can also be controlled with the pt. prefix.

Value

A ggplot2 plot. The computed sample coordinates and ggplot command are available as $data and $code respectively. If stat.by is given, then $stats and $stats$code are set. If rank is given, then $data$taxa_coords, $taxa_stats, and $taxa_stats$code are set.

See Also

Other beta_diversity: bdiv_boxplot(), bdiv_clusters(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_table(), bdiv_stats(), bdiv_table(), distmat_stats()

Other ordination: bdiv_ord_table(), distmat_ord_table()

Other visualization: adiv_boxplot(), adiv_corrplot(), bdiv_boxplot(), bdiv_corrplot(), bdiv_heatmap(), plot_heatmap(), rare_corrplot(), rare_multiplot(), rare_stacked(), stats_boxplot(), stats_corrplot(), taxa_boxplot(), taxa_corrplot(), taxa_heatmap(), taxa_stacked()

Examples

```
library(rbiom)

biom <- rarefy(hmp50)

bdiv_ord_plot(biom, layers="pemt", stat.by="Body Site", rank="g")
```

---

bdiv_ord_table *Calculate PCoA and other ordinations, including taxa biplots and statistics.*

Description

The biplot parameters (taxa, unc, p.top, and p.adj) only only have an effect when rank is not NULL.

Usage

```
bdiv_ord_table(
  biom,
  bdiv = "Bray-Curtis",
  ord = "PCoA",
  weighted = TRUE,
  md = NULL,
  k = 2,
  stat.by = NULL,
  split.by = NULL,
  tree = NULL,
  test = "adonis2",
  seed = 0,
  permutations = 999,
  rank = NULL,
  taxa = 6,
  p.top = Inf,
  p.adj = "fdr",
  unc = "singly",
  ...
)
```

Arguments

biom	An rbiom object, such as from as_rbiom(). Any value accepted by as_rbiom() can also be given here.
bdiv	Beta diversity distance algorithm(s) to use. Options are: "Bray-Curtis", "Manhattan", "Euclidean", "Jaccard", and "UniFrac". For "UniFrac", a phylogenetic tree must be present in biom or explicitly provided via tree=. Multiple/abbreviated values allowed. Default: "Bray-Curtis"
ord	Method for reducing dimensionality. Options are: "PCoA" - Principal coordinate analysis; ape::pcoa(). "UMAP" - Uniform manifold approximation and projection; uwot::umap(). "NMDS" - Nonmetric multidimensional scaling; vegan::metaMDS(). "tSNE" - t-distributed stochastic neighbor embedding; tsne::tsne(). Multiple/abbreviated values allowed. Default: "PCoA"
weighted	Take relative abundances into account. When weighted=FALSE, only presence/absence is considered. Multiple values allowed. Default: TRUE
md	Dataset field(s) to include in the output data frame, or '.all' to include all metadata fields. Default: '.all'
k	Number of ordination dimensions to return. Either 2L or 3L. Default: 2L
stat.by	The categorical or numeric metadata field over which statistics should be calculated. Required.
split.by	Dataset field(s) that the data should be split by prior to any calculations. Must be categorical. Default: NULL
tree	A phylo object representing the phylogenetic relationships of the taxa in biom. Only required when computing UniFrac distances. Default: biom$tree
test	Permutational test for accessing significance. Options are: "adonis2" - Permutational MANOVA; vegan::adonis2(). "mrpp" - Multiple response permutation procedure; vegan::mrpp(). "none" - Don't run any statistics. Abbreviations are allowed. Default: "adonis2"
seed	Random seed for permutations. Default: 0
permutations	Number of random permutations to use. Default: 999
rank	What rank(s) of taxa to compute biplot coordinates and statistics for, or NULL to disable. E.g. "Phylum", "Genus", ".otu", etc. An integer vector can also be given, where 1 is the highest rank, 2 is the second highest, -1 is the lowest rank, -2 is the second lowest, and 0 is the OTU "rank". Run biom$ranks to see all options for a given rbiom object. Default: NULL.
taxa	Which taxa to display. An integer value will show the top n most abundant taxa. A value 0 <= n < 1 will show any taxa with that mean abundance or greater (e.g. 0.1 implies >= 10%). A character vector of taxa names will show only those named taxa. Default: 6.

p.top — Only display taxa with the most significant differences in abundance. If p.top is >= 1, then the p.top most significant taxa are displayed. If p.top is less than one, all taxa with an adjusted p-value <= p.top are displayed. Recommended to be used in combination with the taxa parameter to set a lower bound on the mean abundance of considered taxa. Default: Inf

p.adj — Method to use for multiple comparisons adjustment of p-values. Run p.adjust.methods for a list of available options. Default: "fdr"

unc — How to handle unclassified, uncultured, and similarly ambiguous taxa names. Options are:

"singly" - Replaces them with the OTU name.

"grouped" - Replaces them with a higher rank's name.

"drop" - Excludes them from the result.

"asis" - To not check/modify any taxa names.

Abbreviations are allowed. Default: "singly"

... — Additional arguments to pass on to uwot::umap(), ape::pcoa(), vegan::metaMDS(), or tsne::tsne().

Value

A data.frame with columns .sample, .weighted, .bdiv, .ord, .x, .y, and (optionally) .z. Any columns given by md, split.by, and stat.by are included as well. If stat.by is given, then $stats and $stats$code) are set. If rank is given, then $taxa_coords, $taxa_stats, and $taxa_stats$code are set.

See Also

Other beta_diversity: bdiv_boxplot(), bdiv_clusters(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), bdiv_stats(), bdiv_table(), distmat_stats()

Other ordination: bdiv_ord_plot(), distmat_ord_table()

Examples

```
library(rbiom)

ord <- bdiv_ord_table(hmp50, "bray", "pcoa", stat.by="Body Site", rank="g")
head(ord)

ord$stats

ord$taxa_stats
```

bdiv_stats *Test beta diversity for associations with metadata.*

Description

A convenience wrapper for `bdiv_table()` + `stats_table()`.

Usage

```
bdiv_stats(
  biom,
  regr = NULL,
  stat.by = NULL,
  bdiv = "Bray-Curtis",
  weighted = TRUE,
  tree = NULL,
  within = NULL,
  between = NULL,
  split.by = NULL,
  transform = "none",
  test = "emmeans",
  fit = "gam",
  at = NULL,
  level = 0.95,
  alt = "!=",
  mu = 0,
  p.adj = "fdr"
)
```

Arguments

`biom`	An rbiom object, such as from `as_rbiom()`. Any value accepted by `as_rbiom()` can also be given here.
`regr`	Dataset field with the x-axis (independent; predictive) values. Must be numeric. Default: `NULL`
`stat.by`	Dataset field with the statistical groups. Must be categorical. Default: `NULL`
`bdiv`	Beta diversity distance algorithm(s) to use. Options are: `"Bray-Curtis"`, `"Manhattan"`, `"Euclidean"`, `"Jaccard"`, and `"UniFrac"`. For `"UniFrac"`, a phylogenetic tree must be present in `biom` or explicitly provided via `tree=`. Multiple/abbreviated values allowed. Default: `"Bray-Curtis"`
`weighted`	Take relative abundances into account. When `weighted=FALSE`, only presence/absence is considered. Multiple values allowed. Default: `TRUE`
`tree`	A `phylo` object representing the phylogenetic relationships of the taxa in `biom`. Only required when computing UniFrac distances. Default: `biom$tree`

within, between	Dataset field(s) for intra- or inter- sample comparisons. Alternatively, dataset field names given elsewhere can be prefixed with '==' or '!=' to assign them to within or between, respectively. Default: NULL
split.by	Dataset field(s) that the data should be split by prior to any calculations. Must be categorical. Default: NULL
transform	Transformation to apply. Options are: c("none", "rank", "log", "log1p", "sqrt", "percent"). "rank" is useful for correcting for non-normally distributions before applying regression statistics. Default: "none"
test	Method for computing p-values: 'wilcox', 'kruskal', 'emmeans', or 'emtrends'. Default: 'emmeans'
fit	How to fit the trendline. 'lm', 'log', or 'gam'. Default: 'gam'
at	Position(s) along the x-axis where the means or slopes should be evaluated. Default: NULL, which samples 100 evenly spaced positions and selects the position where the p-value is most significant.
level	The confidence level for calculating a confidence interval. Default: 0.95
alt	Alternative hypothesis direction. Options are '!=' (two-sided; not equal to mu), '<' (less than mu), or '>' (greater than mu). Default: '!='
mu	Reference value to test against. Default: 0
p.adj	Method to use for multiple comparisons adjustment of p-values. Run p.adjust.methods for a list of available options. Default: "fdr"

Value

A tibble data.frame with fields from the table below. This tibble object provides the $code operator to print the R code used to generate the statistics.

Field	Description
.mean	Estimated marginal mean. See emmeans::emmeans().
.mean.diff	Difference in means.
.slope	Trendline slope. See emmeans::emtrends().
.slope.diff	Difference in slopes.
.h1	Alternate hypothesis.
.p.val	Probability that null hypothesis is correct.
.adj.p	.p.val after adjusting for multiple comparisons.
.effect.size	Effect size. See emmeans::eff_size().
.lower	Confidence interval lower bound.
.upper	Confidence interval upper bound.
.se	Standard error.
.n	Number of samples.
.df	Degrees of freedom.
.stat	Wilcoxon or Kruskal-Wallis rank sum statistic.
.t.ratio	.mean / .se
.r.sqr	Percent of variation explained by the model.
.adj.r	.r.sqr, taking degrees of freedom into account.
.aic	Akaike Information Criterion (predictive models).
.bic	Bayesian Information Criterion (descriptive models).

.loglik	Log-likelihood goodness-of-fit score.
.fit.p	P-value for observing this fit by chance.

See Also

Other beta_diversity: bdiv_boxplot(), bdiv_clusters(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), bdiv_ord_table(), bdiv_table(), distmat_stats()

Other stats_tables: adiv_stats(), distmat_stats(), stats_table(), taxa_stats()

Examples

```
library(rbiom)

biom <- rarefy(hmp50)

bdiv_stats(biom, stat.by = "Sex", bdiv = c("bray", "unifrac"))[,1:7]

bdiv_stats(biom, stat.by = "Body Site", split.by = "==Sex")[,1:6]
```

---

bdiv_table *Distance / dissimilarity between samples.*

Description

Distance / dissimilarity between samples.

Usage

```
bdiv_table(
  biom,
  bdiv = "Bray-Curtis",
  weighted = TRUE,
  tree = NULL,
  md = ".all",
  within = NULL,
  between = NULL,
  delta = ".all",
  transform = "none",
  ties = "random",
  seed = 0,
  cpus = -1
)

bdiv_matrix(
  biom,
  bdiv = "Bray-Curtis",
```

```
  weighted = TRUE,
  tree = NULL,
  within = NULL,
  between = NULL,
  transform = "none",
  ties = "random",
  seed = 0,
  cpus = -1
)

bdiv_distmat(
  biom,
  bdiv = "Bray-Curtis",
  weighted = TRUE,
  tree = NULL,
  within = NULL,
  between = NULL,
  transform = "none",
  cpus = -1
)
```

Arguments

biom	An rbiom object, such as from as_rbiom(). Any value accepted by as_rbiom() can also be given here.
bdiv	Beta diversity distance algorithm(s) to use. Options are: "Bray-Curtis", "Manhattan", "Euclidean", "Jaccard", and "UniFrac". For "UniFrac", a phylogenetic tree must be present in biom or explicitly provided via tree=. Multiple/abbreviated values allowed. Default: "Bray-Curtis"
weighted	Take relative abundances into account. When weighted=FALSE, only presence/absence is considered. Multiple values allowed. Default: TRUE
tree	A phylo object representing the phylogenetic relationships of the taxa in biom. Only required when computing UniFrac distances. Default: biom$tree
md	Dataset field(s) to include in the output data frame, or '.all' to include all metadata fields. Default: '.all'
within, between	Dataset field(s) for intra- or inter- sample comparisons. Alternatively, dataset field names given elsewhere can be prefixed with '==' or '!=' to assign them to within or between, respectively. Default: NULL
delta	For numeric metadata, report the absolute difference in values for the two samples, for instance 2 instead of "10 vs 12". Default: TRUE
transform	Transformation to apply. Options are: c("none", "rank", "log", "log1p", "sqrt", "percent"). "rank" is useful for correcting for non-normally distributions before applying regression statistics. Default: "none"
ties	When transform="rank", how to rank identical values. Options are: c("average", "first", "last", "random", "max", "min"). See rank() for details. Default: "random"

seed	Random seed for permutations. Default: 0
cpus	The number of CPUs to use. Set to -1 to use all available, or to 0 to not use threaded computing. Default: -1

Value

bdiv_matrix() - An R matrix of samples x samples.

bdiv_distmat() - A dist-class distance matrix.

bdiv_table() - A tibble data.frame with columns names .sample1, .sample2, .weighted, .bdiv, .distance, and any fields requested by md. Numeric metadata fields will be returned as abs(x - y); categorical metadata fields as "x", "y", or "x vs y".

Metadata Comparisons

Prefix metadata fields with == or != to limit comparisons to within or between groups, respectively. For example, stat.by = '==Sex' will run calculations only for intra-group comparisons, returning "Male" and "Female", but NOT "Female vs Male". Similarly, setting stat.by = '!=Body Site' will only show the inter-group comparisons, such as "Saliva vs Stool", "Anterior nares vs Buccal mucosa", and so on.

The same effect can be achieved by using the within and between parameters. stat.by = '==Sex' is equivalent to stat.by = 'Sex', within = 'Sex'.

See Also

Other beta_diversity: bdiv_boxplot(), bdiv_clusters(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), bdiv_ord_table(), bdiv_stats(), distmat_stats()

Examples

```
library(rbiom)

# Subset to four samples
biom <- hmp50$clone()
biom$counts <- biom$counts[,c("HMP18", "HMP19", "HMP20", "HMP21")]

# Return in long format with metadata
bdiv_table(biom, 'unifrac', md = ".all")

# Only look at distances among the stool samples
bdiv_table(biom, 'unifrac', md = c("==Body Site", "Sex"))

# Or between males and females
bdiv_table(biom, 'unifrac', md = c("Body Site", "!=Sex"))

# All-vs-all matrix
bdiv_matrix(biom, 'unifrac')

# All-vs-all distance matrix
dm <- bdiv_distmat(biom, 'unifrac')
dm
```

```
plot(hclust(dm))
```

---

bdply *Apply a function to each subset of an rbiom object.*

Description

`blply()` and `bdply()` let you divide your biom dataset into smaller pieces, run a function on those smaller rbiom objects, and return the results as a data.frame or list.

Usage

```
bdply(biom, vars, FUN, ..., iters = list(), prefix = FALSE)

blply(biom, vars, FUN, ..., iters = list(), prefix = FALSE)
```

Arguments

`biom`	An rbiom object, such as from `as_rbiom()`. Any value accepted by `as_rbiom()` can also be given here.
`vars`	A character vector of metadata fields. Each unique combination of values in these columns will be used to create a subsetted rbiom object to pass to `FUN`. If `NULL`, `biom` will be passed to `FUN` unaltered. Unambiguous abbreviations of metadata fields are also accepted.
`FUN`	The function to execute on each subset of `biom`. For `bdply()`, the returned value will be coerced to a data.frame. For `blply()`, any returned value is unmodified.
`...`	Additional arguments to pass on to `FUN`.
`iters`	A named list of values to pass to `FUN`. Unlike `...`, these will be iterated over in all combinations. Default: `list()`
`prefix`	When `TRUE`, prefixes the names in in `iters` with a '.' in the final data.frame or 'split_labels' attribute. Default: `FALSE`

Details

You can also specify additional variables for your function to iterate over in unique combinations.

Calls `plyr::ddply()` or `plyr::dlply()` internally.

Value

For `bdply()`, a tibble data.frame comprising the accumulated outputs of `FUN`, along with the columns specified by `vars` and `iters`. For `blply()`, a named list that has details about `vars` and `iters` in `attr(,'split_labels')`.

See Also

Other metadata: glimpse.rbiom()

Other biom: biom_merge()

Examples

```
library(rbiom)

bdply(hmp50, "Sex", `$`, 'n_samples')

blply(hmp50, "Sex", `$`, 'n_samples') %>% unlist()

bdply(hmp50, c("Body Site", "Sex"), function (b) {
  adm <- adiv_matrix(b)[,c("Shannon", "Simpson")]
  apply(adm, 2L, mean)
})

iters <- list(w = c(TRUE, FALSE), d = c("bray", "euclid"))
bdply(hmp50, "Sex", iters = iters, function (b, w, d) {
  r <- range(bdiv_distmat(biom = b, bdiv = d, weighted = w))
  round(data.frame(min = r[[1]], max = r[[2]]))
})
```

---

biom_merge *Combine several rbiom objects into one.*

Description

WARNING: It is generally ill-advised to merge BIOM datasets, as OTUs mappings are dependent on upstream clustering and are not equivalent between BIOM files.

Usage

```
biom_merge(
  ...,
  metadata = NA,
  taxonomy = NA,
  tree = NULL,
  sequences = NA,
  id = NA,
  comment = NA
)
```

Arguments

...	Any number of rbiom objects (e.g. from as_rbiom()), lists of rbiom objects, or valid arguments to the biom parameter of as_rbiom() (for instance file names).
metadata, taxonomy, tree, sequences, id, comment	Replace the corresponding data in the merged rbiom object with these values. Set to NULL to not inherit a particular component. The default, NA, will attempt to create the component based on ... values. The merged phylogenetic tree cannot be inferred.

Value

An rbiom object.

See Also

Other biom: bdply()

Examples

```
library(rbiom)

b1 <- as_rbiom(hmp50$counts[,1:4])
b2 <- as_rbiom(hmp50$counts[,5:8])

biom <- biom_merge(b1, b2)
print(biom)

biom$tree     <- hmp50$tree
biom$metadata <- hmp50$metadata
print(biom)
```

---

distmat_ord_table *Run ordinations on a distance matrix.*

Description

Run ordinations on a distance matrix.

Usage

```
distmat_ord_table(dm, ord = "PCoA", k = 2L, ...)
```

Arguments

dm	A dist-class distance matrix, as returned from bdiv_distmat() or stats::dist(). Required.
ord	Method for reducing dimensionality. Options are:

	"PCoA" - Principal coordinate analysis; ape::pcoa().
	"UMAP" - Uniform manifold approximation and projection; uwot::umap().
	"NMDS" - Nonmetric multidimensional scaling; vegan::metaMDS().
	"tSNE" - t-distributed stochastic neighbor embedding; tsne::tsne().
	Multiple/abbreviated values allowed. Default: "PCoA"
k	Number of ordination dimensions to return. Either 2L or 3L. Default: 2L
...	Additional arguments for ord.

Value

A data.frame with columns .sample, .ord, .x, .y, and (optionally) .z.

See Also

Other ordination: bdiv_ord_plot(), bdiv_ord_table()

Examples

```
library(rbiom)

dm  <- bdiv_distmat(hmp50, "bray")
ord <- distmat_ord_table(dm, "PCoA")
head(ord)
```

---

distmat_stats *Run statistics on a distance matrix vs a categorical or numeric variable.*

Description

Run statistics on a distance matrix vs a categorical or numeric variable.

Usage

```
distmat_stats(dm, groups, test = "adonis2", seed = 0, permutations = 999)
```

Arguments

dm	A dist-class distance matrix, as returned from bdiv_distmat() or stats::dist(). Required.
groups	A named vector of grouping values. The names should correspond to attr(dm, 'Labels'). Values can be either categorical or numeric. Required.
test	Permutational test for accessing significance. Options are:
	"adonis2" - Permutational MANOVA; vegan::adonis2().
	"mrpp" - Multiple response permutation procedure; vegan::mrpp().

	`"none"` - Don't run any statistics.
	Abbreviations are allowed. Default: `"adonis2"`
`seed`	Random seed for permutations. Default: `0`
`permutations`	Number of random permutations to use. Default: `999`

Value

A data.frame with summary statistics from `vegan::permustats()`. The columns are:

.n - The size of the distance matrix.

.stat - The observed statistic. For mrpp, this is the overall weighted mean of group mean distances.

.z - The difference of observed statistic and mean of permutations divided by the standard deviation of permutations (also known as z-values). Evaluated from permuted values without observed statistic.

.p.val - Probability calculated by `test`.

R commands for reproducing the results are in `$code`.

See Also

Other beta_diversity: `bdiv_boxplot()`, `bdiv_clusters()`, `bdiv_corrplot()`, `bdiv_heatmap()`, `bdiv_ord_plot()`, `bdiv_ord_table()`, `bdiv_stats()`, `bdiv_table()`

Other stats_tables: `adiv_stats()`, `bdiv_stats()`, `stats_table()`, `taxa_stats()`

Examples

```
library(rbiom)

hmp10        <- hmp50$clone()
hmp10$counts <- hmp10$counts[,1:10]

dm <- bdiv_distmat(hmp10, 'unifrac')

distmat_stats(dm, groups = pull(hmp10, 'Body Site'))

distmat_stats(dm, groups = pull(hmp10, 'Age'))

# See the R code used to calculate these statistics:
stats <- distmat_stats(dm, groups = pull(hmp10, 'Age'))
stats$code
```

gems *Global Enteric Multicenter Study (n = 1,006)*

Description

Global Enteric Multicenter Study (n = 1,006)

Usage

```
gems
```

Format

An rbiom object with 1,006 samples. Includes metadata and taxonomy.

diarrhea - Case or Control

age - 0 - 4.8 (years old)

country - Bangladesh, Gambia, Kenya, or Mali

Source

doi:10.1186/gb2014156r76 and doi:10.1093/nar/gkx1027

glimpse.rbiom *Get a glimpse of your metadata.*

Description

Get a glimpse of your metadata.

Usage

```
## S3 method for class 'rbiom'
glimpse(x, width = NULL, ...)
```

Arguments

x	An rbiom object, such as from as_rbiom().
width	Width of output. See pillar::glimpse() documentation. Default: NULL
...	Not used.

Value

The original biom, invisibly.

See Also

Other metadata: `bdply()`

Examples

```
library(rbiom)

glimpse(hmp50)
```

`hmp50` *Human Microbiome Project - demo dataset (n = 50)*

Description

Human Microbiome Project - demo dataset (n = 50)

Usage

```
hmp50
```

Format

An rbiom object with 50 samples. Includes metadata, taxonomy, phylogeny, and sequences.

Sex - Male or Female

Body Site - Anterior nares, Buccal mucosa, Mid vagina, Saliva, or Stool

Age - 21 - 40

BMI - 19 - 32

Source

https://portal.hmpdacc.org

modify_metadata *Create, modify, and delete metadata fields.*

Description

mutate() creates new fields in `$metadata` that are functions of existing metadata fields. It can also modify (if the name is the same as an existing field) and delete fields (by setting their value to NULL).

Usage

```
## S3 method for class 'rbiom'
mutate(.data, ..., clone = TRUE)

## S3 method for class 'rbiom'
rename(.data, ..., clone = TRUE)
```

Arguments

`.data`	An rbiom object, such as from `as_rbiom()`.
`...`	Passed on to `dplyr::mutate()` or `dplyr::rename()`.
`clone`	Create a copy of `biom` before modifying. If `FALSE`, `biom` is modified in place as a side-effect. See speed ups for use cases. Default: `TRUE`

Value

An rbiom object.

See Also

Other transformations: `rarefy()`, `rarefy_cols()`, `slice_metadata`, `subset()`, `with()`

Examples

```
library(rbiom)

biom <- slice_max(hmp50, BMI, n = 6)
biom$metadata

# Add a new field to the metadata
biom <- mutate(biom, Obsese = BMI >= 30)
biom$metadata

# Rename a metadata field
biom <- rename(biom, 'Age (years)' = "Age")
biom$metadata
```

plot_heatmap *Create a heatmap with tracks and dendrograms from any matrix.*

Description

Create a heatmap with tracks and dendrograms from any matrix.

Usage

```
plot_heatmap(
  mtx,
  grid = list(label = "Grid Value", colors = "imola"),
  tracks = NULL,
  label = TRUE,
  label_size = NULL,
  rescale = "none",
  trees = TRUE,
  clust = "complete",
  dist = "euclidean",
  asp = 1,
  tree_height = 10,
  track_height = 10,
  legend = "right",
  title = NULL,
  xlab.angle = "auto",
  ...
)
```

Arguments

mtx	A numeric matrix with named rows and columns.
grid	Color palette name, or a list with entries for label, colors, range, bins, na.color, and/or guide. See the Track Definitions section for details. Default: list(label = "Grid Value", colors = "imola")
tracks	List of track definitions. See details below. Default: NULL.
label	Label the matrix rows and columns. You can supply a list or logical vector of length two to control row labels and column labels separately, for example label = c(rows = TRUE, cols = FALSE), or simply label = c(TRUE, FALSE). Other valid options are "rows", "cols", "both", "bottom", "right", and "none". Default: TRUE
label_size	The font size to use for the row and column labels. You can supply a numeric vector of length two to control row label sizes and column label sizes separately, for example c(rows = 20, cols = 8), or simply c(20, 8). Default: NULL, which computes: pmax(8, pmin(20, 100 / dim(mtx)))
rescale	Rescale rows or columns to all have a common min/max. Options: "none", "rows", or "cols". Default: "none"

`trees` Draw a dendrogram for rows (left) and columns (top). You can supply a list or logical vector of length two to control the row tree and column tree separately, for example `trees = c(rows = TRUE, cols = FALSE)`, or simply `trees = c(TRUE, FALSE)`. Other valid options are `"rows"`, `"cols"`, `"both"`, `"left"`, `"top"`, and `"none"`. Default: `TRUE`

`clust` Clustering algorithm for reordering the rows and columns by similarity. You can supply a list or character vector of length two to control the row and column clustering separately, for example `clust = c(rows = "complete", cols = NA)`, or simply `clust = c("complete", NA)`. Options are:

`FALSE` **or** `NA` **-** Disable reordering.

An `hclust` **class object** E.g. from `stats::hclust()`.

A method name - `"ward.D"`, `"ward.D2"`, `"single"`, `"complete"`, `"average"`, `"mcquitty"`, `"median"`, or `"centroid"`.

Default: `"complete"`

`dist` Distance algorithm to use when reordering the rows and columns by similarity. You can supply a list or character vector of length two to control the row and column clustering separately, for example `dist = c(rows = "euclidean", cols = "maximum")`, or simply `dist = c("euclidean", "maximum")`. Options are:

A `dist` **class object** E.g. from `stats::dist()` or `bdiv_distmat()`.

A method name - `"euclidean"`, `"maximum"`, `"manhattan"`, `"canberra"`, `"binary"`, or `"minkowski"`.

Default: `"euclidean"`

`asp` Aspect ratio (height/width) for entire grid. Default: `1` (square)

`tree_height, track_height`
The height of the dendrogram or annotation tracks as a percentage of the overall grid size. Use a numeric vector of length two to assign `c(top, left)` independently. Default: `10` (10% of the grid's height)

`legend` Where to place the legend. Options are: `"right"` or `"bottom"`. Default: `"right"`

`title` Plot title. Default: `NULL`.

`xlab.angle` Angle of the labels at the bottom of the plot. Options are `"auto"`, `'0'`, `'30'`, and `'90'`. Default: `"auto"`.

`...` Additional arguments to pass on to ggplot2::theme().

Value

A `ggplot2` plot. The computed data points and ggplot command are available as `$data` and `$code`, respectively.

Track Definitions

One or more colored tracks can be placed on the left and/or top of the heatmap grid to visualize associated metadata values.

```
## Categorical ----------------------------
cat_vals <- sample(c("Male", "Female"), 10, replace = TRUE)
tracks   <- list('Sex' = cat_vals)
tracks   <- list('Sex' = list(values = cat_vals, colors = "bright"))
tracks   <- list('Sex' = list(
  values = cat_vals,
  colors = c('Male' = "blue", 'Female' = "red")) )

## Numeric --------------------------------
num_vals <- sample(25:40, 10, replace = TRUE)
tracks   <- list('Age' = num_vals)
tracks   <- list('Age' = list(values = num_vals, colors = "greens"))
tracks   <- list('Age' = list(values = num_vals, range = c(0,50)))
tracks   <- list('Age' = list(
  label  = "Age (Years)",
  values = num_vals,
  colors = c("azure", "darkblue", "darkorchid") ))

## Multiple Tracks ------------------------
tracks <- list('Sex' = cat_vals, 'Age' = num_vals)
tracks <- list(
  list(label = "Sex", values = cat_vals, colors = "bright"),
  list(label = "Age", values = num_vals, colors = "greens") )


mtx           <- matrix(sample(1:50), ncol = 10)
dimnames(mtx) <- list(letters[1:5], LETTERS[1:10])
plot_heatmap(mtx = mtx, tracks = tracks)
```

The following entries in the track definitions are understood:

`values` - The metadata values. When unnamed, order must match matrix.

`range` - The c(min,max) to use for scale values.

`label` - Label for this track. Defaults to the name of this list element.

`side` - Options are `"top"` (default) or `"left"`.

`colors` - A pre-defined palette name or custom set of colors to map to.

`na.color` - The color to use for `NA` values.

`bins` - Bin a gradient into this many bins/steps.

`guide` - A list of arguments for guide_colorbar() or guide_legend().

All built-in color palettes are colorblind-friendly. See Mapping Metadata to Aesthetics for images of the palettes.

Categorical palette names: `"okabe"`, `"carto"`, `"r4"`, `"polychrome"`, `"tol"`, `"bright"`, `"light"`, `"muted"`, `"vibrant"`, `"tableau"`, `"classic"`, `"alphabet"`, `"tableau20"`, `"kelly"`, and `"fishy"`.

Numeric palette names: `"reds"`, `"oranges"`, `"greens"`, `"purples"`, `"grays"`, `"acton"`, `"bamako"`, `"batlow"`, `"bilbao"`, `"buda"`, `"davos"`, `"devon"`, `"grayC"`, `"hawaii"`, `"imola"`, `"lajolla"`, `"lapaz"`, `"nuuk"`, `"oslo"`, `"tokyo"`, `"turku"`, `"bam"`, `"berlin"`, `"broc"`, `"cork"`, `"lisbon"`, `"roma"`, `"tofino"`, `"vanimo"`, and `"vik"`.

See Also

Other visualization: adiv_boxplot(), adiv_corrplot(), bdiv_boxplot(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), rare_corrplot(), rare_multiplot(), rare_stacked(), stats_boxplot(), stats_corrplot(), taxa_boxplot(), taxa_corrplot(), taxa_heatmap(), taxa_stacked()

Examples

```
library(rbiom)

set.seed(123)
mtx <- matrix(runif(5*8), nrow = 5, dimnames = list(LETTERS[1:5], letters[1:8]))

plot_heatmap(mtx)
plot_heatmap(mtx, grid="oranges")
plot_heatmap(mtx, grid=list(colors = "oranges", label = "Some %", bins = 5))

tracks <- list(
  'Number' = sample(1:ncol(mtx)),
  'Person' = list(
    values = factor(sample(c("Alice", "Bob"), ncol(mtx), TRUE)),
    colors = c('Alice' = "purple", 'Bob' = "darkcyan") ),
  'State' = list(
    side   = "left",
    values = sample(c("TX", "OR", "WA"), nrow(mtx), TRUE),
    colors = "bright" )
)

plot_heatmap(mtx, tracks=tracks)
```

---

pull.rbiom *Map sample names to metadata field values.*

Description

Map sample names to metadata field values.

Usage

```
## S3 method for class 'rbiom'
pull(.data, var = -1, name = ".sample", ...)
```

Arguments

.data	An rbiom object, such as from as_rbiom().
var	The metadata field name specified as: • The metadata field name to retrieve. Can be abbreviated.

- A positive integer, giving the position counting from the left.
- A negative integer, giving the position counting from the right.

Default: `-1`

`name` The column to be used as names for a named vector. Specified in a similar manner as var. Default: `".sample"`

`...` Not used.

Value

A vector of metadata values, named with sample names.

See Also

`taxa_map()`

Other samples: `sample_sums()`

Examples

```
library(rbiom)

pull(hmp50, 'Age') %>% head()

pull(hmp50, 'bod') %>% head(4)
```

---

rarefy *Rarefy OTU counts.*

Description

Sub-sample OTU observations such that all samples have an equal number. If called on data with non-integer abundances, values will be re-scaled to integers between 1 and `depth` such that they sum to `depth`.

Usage

```
rarefy(biom, depth = 0.1, n = NULL, seed = 0, clone = TRUE)
```

Arguments

`biom` An rbiom object, such as from `as_rbiom()`. Any value accepted by `as_rbiom()` can also be given here.

`depth` How many observations to keep per sample. When `0 < depth < 1`, it is taken as the minimum percentage of the dataset's observations to keep. Ignored when `n` is specified. Default: `0.1`

n	The number of samples to keep. When `0 < n < 1`, it is taken as the percentage of samples to keep. If negative, that number or percentage of samples is dropped. If `0`, all samples are kept. If `NULL`, `depth` is used instead. Default: `NULL`
seed	An integer seed for randomizing which observations to keep or drop. If you need to create different random rarefactions of the same data, set the seed to a different number each time.
clone	Create a copy of `biom` before modifying. If `FALSE`, `biom` is modified in place as a side-effect. See speed ups for use cases. Default: `TRUE`

Value

An rbiom object.

See Also

Other rarefaction: `rare_corrplot()`, `rare_multiplot()`, `rare_stacked()`, `rarefy_cols()`, `sample_sums()`

Other transformations: `modify_metadata`, `rarefy_cols()`, `slice_metadata`, `subset()`, `with()`

Examples

```
library(rbiom)

sample_sums(hmp50) %>% head()

biom <- rarefy(hmp50)
sample_sums(biom) %>% head()
```

`rarefy_cols` *Transform a counts matrix.*

Description

Rarefaction subset counts so that all samples have the same number of observations. Rescaling rows or cols scales the matrix values so that row sums or column sums equal 1.

Usage

```
rarefy_cols(mtx, depth = 0.1, n = NULL, seed = 0)

rescale_cols(mtx)

rescale_rows(mtx)
```

Arguments

mtx	A matrix-like object.
depth	How many observations to keep per sample. When `0 < depth < 1`, it is taken as the minimum percentage of the dataset's observations to keep. Ignored when n is specified. Default: `0.1`
n	The number of samples to keep. When `0 < n < 1`, it is taken as the percentage of samples to keep. If negative, that number or percentage of samples is dropped. If `0`, all samples are kept. If `NULL`, depth is used instead. Default: `NULL`
seed	An integer to use for seeding the random number generator. If you need to create different random rarefactions of the same matrix, set this seed value to a different number each time.

Value

The rarefied or rescaled matrix.

See Also

Other rarefaction: `rare_corrplot()`, `rare_multiplot()`, `rare_stacked()`, `rarefy()`, `sample_sums()`

Other transformations: `modify_metadata`, `rarefy()`, `slice_metadata`, `subset()`, `with()`

Examples

```
library(rbiom)

# rarefy_cols ----------------------------------------
biom <- hmp50$clone()
sample_sums(biom) %>% head(10)

biom$counts %<>% rarefy_cols(depth=1000)
sample_sums(biom) %>% head(10)


# rescaling ------------------------------------------
mtx <- matrix(sample(1:20), nrow=4)
mtx

rowSums(mtx)
rowSums(rescale_rows(mtx))

colSums(mtx)
colSums(rescale_cols(mtx))
```

`rare_corrplot` *Visualize rarefaction curves with scatterplots and trendlines.*

Description

Visualize rarefaction curves with scatterplots and trendlines.

Usage

```
rare_corrplot(
  biom,
  adiv = "Shannon",
  layers = "tc",
  rline = TRUE,
  stat.by = NULL,
  facet.by = NULL,
  colors = TRUE,
  shapes = TRUE,
  test = "none",
  fit = "log",
  at = NULL,
  level = 0.95,
  p.adj = "fdr",
  transform = "none",
  alt = "!=",
  mu = 0,
  caption = TRUE,
  check = FALSE,
  ...
)
```

Arguments

`biom`	An rbiom object, such as from `as_rbiom()`. Any value accepted by `as_rbiom()` can also be given here.
`adiv`	Alpha diversity metric(s) to use. Options are: `"OTUs"`, `"Shannon"`, `"Chao1"`, `"Simpson"`, and/or `"InvSimpson"`. Set `adiv=".all"` to use all metrics. Multiple/abbreviated values allowed. Default: `"Shannon"`
`layers`	One or more of `c("trend", "confidence", "point", "name", "residual")`. Single letter abbreviations are also accepted. For instance, `c("trend", "point")` is equivalent to `c("t", "p")` and `"tp"`. Default: `"tc"`
`rline`	Where to draw a horizontal line on the plot, intended to show a particular rarefaction depth. Set to `TRUE` to show an auto-selected rarefaction depth or `FALSE` to not show a line. Default: `NULL`
`stat.by`	Dataset field with the statistical groups. Must be categorical. Default: `NULL`

`facet.by` Dataset field(s) to use for faceting. Must be categorical. Default: `NULL`

`colors` How to color the groups. Options are:

`TRUE` - Automatically select colorblind-friendly colors.

`FALSE` **or** `NULL` - Don't use colors.

a palette name - Auto-select colors from this set. E.g. `"okabe"`

character vector - Custom colors to use. E.g. `c("red", "#00FF00")`

named character vector - Explicit mapping. E.g. `c(Male = "blue", Female = "red")`

See "Aesthetics" section below for additional information. Default: `TRUE`

`shapes` Shapes for each group. Options are similar to `colors`'s: `TRUE`, `FALSE`, `NULL`, shape names (typically integers 0 - 17), or a named vector mapping groups to specific shape names. See "Aesthetics" section below for additional information. Default: `TRUE`

`test` Method for computing p-values: `'none'`, `'emmeans'`, or `'emtrends'`. Default: `'emmeans'`

`fit` How to fit the trendline. Options are `'lm'`, `'log'`, and `'gam'`. Default: `'log'`

`at` Position(s) along the x-axis where the means or slopes should be evaluated. Default: `NULL`, which samples 100 evenly spaced positions and selects the position where the p-value is most significant.

`level` The confidence level for calculating a confidence interval. Default: `0.95`

`p.adj` Method to use for multiple comparisons adjustment of p-values. Run `p.adjust.methods` for a list of available options. Default: `"fdr"`

`transform` Transformation to apply. Options are: `c("none", "rank", "log", "log1p", "sqrt", "percent")`. `"rank"` is useful for correcting for non-normally distributions before applying regression statistics. Default: `"none"`

`alt` Alternative hypothesis direction. Options are `'!='` (two-sided; not equal to `mu`), `'<'` (less than `mu`), or `'>'` (greater than `mu`). Default: `'!='`

`mu` Reference value to test against. Default: `0`

`caption` Add methodology caption beneath the plot. Default: `TRUE`

`check` Generate additional plots to aid in assessing data normality. Default: `FALSE`

`...` Additional parameters to pass along to ggplot2 functions. Prefix a parameter name with a layer name to pass it to only that layer. For instance, `p.size = 2` ensures only the points have their size set to 2.

Value

A `ggplot2` plot. The computed data points, ggplot2 command, stats table, and stats table commands are available as `$data`, `$code`, `$stats`, and `$stats$code`, respectively.

Aesthetics

All built-in color palettes are colorblind-friendly. The available categorical palette names are: `"okabe"`, `"carto"`, `"r4"`, `"polychrome"`, `"tol"`, `"bright"`, `"light"`, `"muted"`, `"vibrant"`, `"tableau"`, `"classic"`, `"alphabet"`, `"tableau20"`, `"kelly"`, and `"fishy"`.

Shapes can be given as per base R - numbers 0 through 17 for various shapes, or the decimal value of an ascii character, e.g. a-z = 65:90; A-Z = 97:122 to use letters instead of shapes on the plot. Character strings may used as well.

See Also

Other rarefaction: rare_multiplot(), rare_stacked(), rarefy(), rarefy_cols(), sample_sums()

Other visualization: adiv_boxplot(), adiv_corrplot(), bdiv_boxplot(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), plot_heatmap(), rare_multiplot(), rare_stacked(), stats_boxplot(), stats_corrplot(), taxa_boxplot(), taxa_corrplot(), taxa_heatmap(), taxa_stacked()

Examples

```
library(rbiom)

biom <- subset(hmp50, `Body Site` %in% c('Saliva', 'Stool'))
rare_corrplot(biom, stat.by = "body", adiv = c("sh", "o"), facet.by = "Sex")
```

---

rare_multiplot *Combines rare_corrplot and rare_stacked into a single figure.*

Description

Combines rare_corrplot and rare_stacked into a single figure.

Usage

```
rare_multiplot(
  biom,
  adiv = "Shannon",
  layers = "tc",
  rline = TRUE,
  stat.by = NULL,
  facet.by = NULL,
  colors = TRUE,
  shapes = TRUE,
  test = "none",
  fit = "log",
  at = NULL,
  level = 0.95,
  p.adj = "fdr",
  transform = "none",
  alt = "!=",
  mu = 0,
  caption = TRUE,
```

```
  check = FALSE,
  ...
)
```

Arguments

biom	An rbiom object, such as from as_rbiom(). Any value accepted by as_rbiom() can also be given here.
adiv	Alpha diversity metric(s) to use. Options are: "OTUs", "Shannon", "Chao1", "Simpson", and/or "InvSimpson". Set adiv=".all" to use all metrics. Multiple/abbreviated values allowed. Default: "Shannon"
layers	One or more of c("trend", "confidence", "point", "name", "residual"). Single letter abbreviations are also accepted. For instance, c("trend", "point") is equivalent to c("t", "p") and "tp". Default: "tc"
rline	Where to draw a horizontal line on the plot, intended to show a particular rarefaction depth. Set to TRUE to show an auto-selected rarefaction depth or FALSE to not show a line. Default: NULL
stat.by	Dataset field with the statistical groups. Must be categorical. Default: NULL
facet.by	Dataset field(s) to use for faceting. Must be categorical. Default: NULL
colors	How to color the groups. Options are: TRUE - Automatically select colorblind-friendly colors. FALSE or NULL - Don't use colors. **a palette name -** Auto-select colors from this set. E.g. "okabe" **character vector -** Custom colors to use. E.g. c("red", "#00FF00") **named character vector -** Explicit mapping. E.g. c(Male = "blue", Female = "red") See "Aesthetics" section below for additional information. Default: TRUE
shapes	Shapes for each group. Options are similar to colors's: TRUE, FALSE, NULL, shape names (typically integers 0 - 17), or a named vector mapping groups to specific shape names. See "Aesthetics" section below for additional information. Default: TRUE
test	Method for computing p-values: 'none', 'emmeans', or 'emtrends'. Default: 'emmeans'
fit	How to fit the trendline. Options are 'lm', 'log', and 'gam'. Default: 'log'
at	Position(s) along the x-axis where the means or slopes should be evaluated. Default: NULL, which samples 100 evenly spaced positions and selects the position where the p-value is most significant.
level	The confidence level for calculating a confidence interval. Default: 0.95
p.adj	Method to use for multiple comparisons adjustment of p-values. Run p.adjust.methods for a list of available options. Default: "fdr"
transform	Transformation to apply. Options are: c("none", "rank", "log", "log1p", "sqrt", "percent"). "rank" is useful for correcting for non-normally distributions before applying regression statistics. Default: "none"

alt	Alternative hypothesis direction. Options are '!=' (two-sided; not equal to mu), '<' (less than mu), or '>' (greater than mu). Default: '!='
mu	Reference value to test against. Default: 0
caption	Add methodology caption beneath the plot. Default: TRUE
check	Generate additional plots to aid in assessing data normality. Default: FALSE
...	Additional parameters to pass along to ggplot2 functions. Prefix a parameter name with a layer name to pass it to only that layer. For instance, p.size = 2 ensures only the points have their size set to 2.

Value

A ggplot2 plot. The computed data points, ggplot2 command, stats table, and stats table commands are available as $data, $code, $stats, and $stats$code, respectively.

Aesthetics

All built-in color palettes are colorblind-friendly. The available categorical palette names are: "okabe", "carto", "r4", "polychrome", "tol", "bright", "light", "muted", "vibrant", "tableau", "classic", "alphabet", "tableau20", "kelly", and "fishy".

Shapes can be given as per base R - numbers 0 through 17 for various shapes, or the decimal value of an ascii character, e.g. a-z = 65:90; A-Z = 97:122 to use letters instead of shapes on the plot. Character strings may used as well.

See Also

Other rarefaction: rare_corrplot(), rare_stacked(), rarefy(), rarefy_cols(), sample_sums()

Other visualization: adiv_boxplot(), adiv_corrplot(), bdiv_boxplot(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), plot_heatmap(), rare_corrplot(), rare_stacked(), stats_boxplot(), stats_corrplot(), taxa_boxplot(), taxa_corrplot(), taxa_heatmap(), taxa_stacked()

Examples

```
library(rbiom)

rare_multiplot(hmp50, stat.by = "Body Site")
```

---

rare_stacked *Visualize the number of observations per sample.*

Description

Visualize the number of observations per sample.

Usage

```
rare_stacked(
  biom,
  rline = TRUE,
  counts = TRUE,
  labels = TRUE,
  y.transform = "log10",
  ...
)
```

Arguments

biom	An rbiom object, such as from as_rbiom(). Any value accepted by as_rbiom() can also be given here.
rline	Where to draw a horizontal line on the plot, intended to show a particular rarefaction depth. Set to TRUE to show an auto-selected rarefaction depth, FALSE to not show a line, or an integer for a custom position. Default: TRUE.
counts	Display the number of samples and reads remaining after rarefying to rline reads per sample. Default: TRUE.
labels	Show sample names under each bar. Default: TRUE.
y.transform	Y-axis transformation. Options are "log10" or "none". Default: "log10". Use xaxis.transform or yaxis.transform to pass custom values directly to ggplot2's scale_* functions.
...	Additional parameters to pass along to ggplot2 functions. Prefix a parameter name with r. to ensure it gets passed to (and only to) geom_hline. For instance, r.color = "black" ensures only the horizontal rarefaction line has its color set to "black".

Value

A ggplot2 plot. The computed data points and ggplot command are available as $data and $code, respectively.

See Also

Other rarefaction: rare_corrplot(), rare_multiplot(), rarefy(), rarefy_cols(), sample_sums()

Other visualization: adiv_boxplot(), adiv_corrplot(), bdiv_boxplot(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), plot_heatmap(), rare_corrplot(), rare_multiplot(), stats_boxplot(), stats_corrplot(), taxa_boxplot(), taxa_corrplot(), taxa_heatmap(), taxa_stacked()

Examples

```
library(rbiom)

rare_stacked(hmp50)
```

```
rare_stacked(hmp50, rline = 500, r.linewidth = 2, r.linetype = "twodash")

fig <- rare_stacked(hmp50, counts = FALSE)
fig$code
```

`read_biom` *Parse counts, metadata, taxonomy, and phylogeny from a BIOM file.*

Description

Parse counts, metadata, taxonomy, and phylogeny from a BIOM file.

Usage

```
read_biom(src, ...)
```

Arguments

`src`	Input data as either a file path, URL, or JSON string. BIOM files can be formatted according to version 1.0 (JSON) or 2.1 (HDF5) specifications, or as classical tabular format. URLs must begin with `http://`, `https://`, `ftp://`, or `ftps://`. JSON files must have `{` as their first character. Compressed (gzip or bzip2) BIOM files are also supported. NOTE: to read HDF5 formatted BIOM files, the BioConductor R package `rhdf5` must be installed.
`...`	Properties to set in the new rbiom object, for example, `metadata`, `id`, `comment`, or `tree`.

Value

An rbiom object.

See Also

`as_rbiom()`

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read_biom(infile)

print(biom)

# Taxa Abundances
biom$counts[1:4,1:10] %>% as.matrix()

biom$taxonomy %>% head()
```

```
# Metadata
biom$metadata %>% head()

table(biom$metadata$Sex, biom$metadata$`Body Site`)

sprintf("Mean age: %.1f", mean(biom$metadata$Age))

# Phylogenetic tree
biom$tree %>%
  tree_subset(1:10) %>%
  plot()
```

`read_fasta` *Parse a fasta file into a named character vector.*

Description

Parse a fasta file into a named character vector.

Usage

```
read_fasta(file, ids = NULL)
```

Arguments

`file`	A file/URL with fasta-formatted sequences. Can optionally be compressed with gzip, bzip2, xz, or lzma.
`ids`	Character vector of IDs to retrieve. The default, `NULL`, will retrieve everything.

Value

A named character vector in which names are the fasta headers and values are the sequences.

`read_tree` *Read a newick formatted phylogenetic tree.*

Description

A phylogenetic tree is required for computing UniFrac distance matrices. You can load a tree from a file or by providing the tree string directly. This tree must be in Newick format, also known as parenthetic format and New Hampshire format.

Usage

```
read_tree(src)
```

Arguments

src	Input data as either a file path, URL, or Newick string. Compressed (gzip or bzip2) files are also supported.

Value

A phylo class object representing the tree.

See Also

Other phylogeny: tree_subset()

Examples

```
library(rbiom)

infile <- system.file("extdata", "newick.tre", package = "rbiom")
tree <- read_tree(infile)

tree <- read_tree("
    (t9:0.99,((t5:0.87,t2:0.89):0.51,(((t10:0.16,(t7:0.83,t4:0.96)
    :0.94):0.69,(t6:0.92,(t3:0.62,t1:0.85):0.54):0.23):0.74,t8:0.1
    2):0.43):0.67);")
```

---

sample_sums *Summarize the taxa observations in each sample.*

Description

Summarize the taxa observations in each sample.

Usage

```
sample_sums(biom, rank = -1, sort = NULL, unc = "singly")

sample_apply(biom, FUN, rank = -1, sort = NULL, unc = "singly", ...)
```

Arguments

biom	An rbiom object, such as from as_rbiom(). Any value accepted by as_rbiom() can also be given here.
rank	What rank(s) of taxa to display. E.g. "Phylum", "Genus", ".otu", etc. An integer vector can also be given, where 1 is the highest rank, 2 is the second highest, -1 is the lowest rank, -2 is the second lowest, and 0 is the OTU "rank". Run biom$ranks to see all options for a given rbiom object. Default: -1.

sort	Sort the result. Options: NULL - don't sort; "asc" - in ascending order (smallest to largest); "desc" - in descending order (largest to smallest). Ignored when the result is not a simple numeric vector. Default: NULL
unc	How to handle unclassified, uncultured, and similarly ambiguous taxa names. Options are: "singly" - Replaces them with the OTU name. "grouped" - Replaces them with a higher rank's name. "drop" - Excludes them from the result. "asis" - To not check/modify any taxa names. Abbreviations are allowed. Default: "singly"
FUN	The function to apply to each column of taxa_matrix().
...	Optional arguments to FUN.

Value

For sample_sums, A named numeric vector of the number of observations in each sample. For sample_apply, a named vector or list with the results of FUN. The names are the taxa IDs.

See Also

Other samples: pull.rbiom()

Other rarefaction: rare_corrplot(), rare_multiplot(), rare_stacked(), rarefy(), rarefy_cols()

Other taxa_abundance: taxa_boxplot(), taxa_clusters(), taxa_corrplot(), taxa_heatmap(), taxa_stacked(), taxa_stats(), taxa_sums(), taxa_table()

Examples

```
library(rbiom)
library(ggplot2)

sample_sums(hmp50, sort = 'asc') %>% head()

# Unique OTUs and "cultured" classes per sample
nnz <- function (x) sum(x > 0) # number of non-zeroes
sample_apply(hmp50, nnz, 'otu') %>% head()
sample_apply(hmp50, nnz, 'class', unc = 'drop') %>% head()

# Number of reads in each sample's most abundant family
sample_apply(hmp50, base::max, 'f', sort = 'desc') %>% head()

ggplot() + geom_histogram(aes(x=sample_sums(hmp50)), bins = 20)
```

slice_metadata *Subset to a specific number of samples.*

Description

Subset to a specific number of samples.

Usage

```
## S3 method for class 'rbiom'
slice(.data, ..., .by = NULL, .preserve = FALSE, clone = TRUE)

## S3 method for class 'rbiom'
slice_head(.data, n, prop, by = NULL, clone = TRUE, ...)

## S3 method for class 'rbiom'
slice_tail(.data, n, prop, by = NULL, clone = TRUE, ...)

## S3 method for class 'rbiom'
slice_min(
  .data,
  order_by,
  n,
  prop,
  by = NULL,
  with_ties = TRUE,
  na_rm = FALSE,
  clone = TRUE,
  ...
)

## S3 method for class 'rbiom'
slice_max(
  .data,
  order_by,
  n,
  prop,
  by = NULL,
  with_ties = TRUE,
  na_rm = FALSE,
  clone = TRUE,
  ...
)

## S3 method for class 'rbiom'
slice_sample(
  .data,
```

```
  n,
  prop,
  by = NULL,
  weight_by = NULL,
  replace = FALSE,
  clone = TRUE,
  ...
)
```

Arguments

.data	An rbiom object, such as from as_rbiom().
...	For slice(), integer row indexes. For other slice_*() functions, not used. See dplyr::slice().
.by, by	**[Experimental]** <tidy-select> Optionally, a selection of columns to group by for just this operation, functioning as an alternative to group_by(). For details and examples, see ?dplyr_by.
.preserve	Relevant when the .data input is grouped. If .preserve = FALSE (the default), the grouping structure is recalculated based on the resulting data, otherwise the grouping is kept as is.
clone	Create a copy of biom before modifying. If FALSE, biom is modified in place as a side-effect. See speed ups for use cases. Default: TRUE
n, prop	Provide either n, the number of rows, or prop, the proportion of rows to select. If neither are supplied, n = 1 will be used. If n is greater than the number of rows in the group (or prop > 1), the result will be silently truncated to the group size. prop will be rounded towards zero to generate an integer number of rows. A negative value of n or prop will be subtracted from the group size. For example, n = -2 with a group of 5 rows will select 5 - 2 = 3 rows; prop = -0.25 with 8 rows will select 8 * (1 - 0.25) = 6 rows.
order_by	<data-masking> Variable or function of variables to order by. To order by multiple variables, wrap them in a data frame or tibble.
with_ties	Should ties be kept together? The default, TRUE, may return more rows than you request. Use FALSE to ignore ties, and return the first n rows.
na_rm	Should missing values in order_by be removed from the result? If FALSE, NA values are sorted to the end (like in arrange()), so they will only be included if there are insufficient non-missing values to reach n/prop.
weight_by	<data-masking> Sampling weights. This must evaluate to a vector of non-negative numbers the same length as the input. Weights are automatically standardised to sum to 1.
replace	Should sampling be performed with (TRUE) or without (FALSE, the default) replacement.

Value

An rbiom object.

See Also

Other transformations: modify_metadata, rarefy(), rarefy_cols(), subset(), with()

Examples

```
library(rbiom)

# The last 3 samples in the metadata table.
biom <- slice_tail(hmp50, n = 3)
biom$metadata

# The 3 oldest subjects sampled.
biom <- slice_max(hmp50, Age, n = 3)
biom$metadata

# Pick 3 samples at random.
biom <- slice_sample(hmp50, n = 3)
biom$metadata
```

---

stats_boxplot *Visualize categorical metadata effects on numeric values.*

Description

Visualize categorical metadata effects on numeric values.

Usage

```
stats_boxplot(
  df,
  x = NULL,
  y = attr(df, "response"),
  layers = "x",
  stat.by = x,
  facet.by = NULL,
  colors = TRUE,
  shapes = TRUE,
  patterns = FALSE,
  test = "auto",
  flip = FALSE,
  stripe = NULL,
  ci = "ci",
  level = 0.95,
  p.adj = "fdr",
  p.top = Inf,
  outliers = NULL,
  xlab.angle = "auto",
```

```
  p.label = 0.05,
  caption = TRUE,
  ...
)
```

Arguments

df	The dataset (data.frame or tibble object). "Dataset fields" mentioned below should match column names in df. Required.
x	A categorical metadata column name to use for the x-axis. Or NULL, which groups all samples into a single category.
y	A numeric metadata column name to use for the y-axis. Default: attr(df, 'response')
layers	One or more of c("bar", "box" ("x"), "violin", "dot", "strip", "crossbar", "errorbar", "linerange", "pointrange"). Single letter abbreviations are also accepted. For instance, c("box", "dot") is equivalent to c("x", "d") and "xd". Default: "x"
stat.by	Dataset field with the statistical groups. Must be categorical. Default: NULL
facet.by	Dataset field(s) to use for faceting. Must be categorical. Default: NULL
colors	How to color the groups. Options are: TRUE - Automatically select colorblind-friendly colors. FALSE or NULL - Don't use colors. **a palette name -** Auto-select colors from this set. E.g. "okabe" **character vector -** Custom colors to use. E.g. c("red", "#00FF00") **named character vector -** Explicit mapping. E.g. c(Male = "blue", Female = "red") See "Aesthetics" section below for additional information. Default: TRUE
shapes	Shapes for each group. Options are similar to colors's: TRUE, FALSE, NULL, shape names (typically integers 0 - 17), or a named vector mapping groups to specific shape names. See "Aesthetics" section below for additional information. Default: TRUE
patterns	Patterns for each group. Options are similar to colors's: TRUE, FALSE, NULL, pattern names ("brick", "chevron", "fish", "grid", etc), or a named vector mapping groups to specific pattern names. See "Aesthetics" section below for additional information. Default: FALSE
test	Method for computing p-values: 'auto' or 'none'. 'auto' will choose Wilcox or Kruskal-Wallis depending on the number of groups.
flip	Transpose the axes, so that taxa are present as rows instead of columns. Default: FALSE
stripe	Shade every other x position. Default: *same as flip*
ci	How to calculate min/max of the **crossbar**, **errorbar**, **linerange**, and **pointrange** layers. Options are: "ci" (confidence interval), "range", "sd" (standard deviation), "se" (standard error), and "mad" (median absolute deviation). The center mark of **crossbar** and **pointrange** represents the mean, except for "mad" in which case it represents the median. Default: "ci"

level	The confidence level for calculating a confidence interval. Default: 0.95
p.adj	Method to use for multiple comparisons adjustment of p-values. Run p.adjust.methods for a list of available options. Default: "fdr"
p.top	Only display taxa with the most significant differences in abundance. If p.top is >= 1, then the p.top most significant taxa are displayed. If p.top is less than one, all taxa with an adjusted p-value <= p.top are displayed. Recommended to be used in combination with the taxa parameter to set a lower bound on the mean abundance of considered taxa. Default: Inf
outliers	Show boxplot outliers? TRUE to always show. FALSE to always hide. NULL to only hide them when overlaying a dot or strip chart. Default: NULL
xlab.angle	Angle of the labels at the bottom of the plot. Options are "auto", '0', '30', and '90'. Default: "auto".
p.label	Minimum adjusted p-value to display on the plot with a bracket. p.label = 0.05 - Show p-values that are <= 0.05. p.label = 0 - Don't show any p-values on the plot. p.label = 1 - Show all p-values on the plot. If a numeric vector with more than one value is provided, they will be used as breaks for asterisk notation. Default: 0.05
caption	Add methodology caption beneath the plot. Default: TRUE
...	Additional parameters to pass along to ggplot2 functions. Prefix a parameter name with a layer name to pass it to only that layer. For instance, d.size = 2 ensures only the points on the **dot** layer have their size set to 2.

Value

A ggplot2 plot. The computed data points, ggplot2 command, stats table, and stats table commands are available as $data, $code, $stats, and $stats$code, respectively.

Aesthetics

All built-in color palettes are colorblind-friendly. The available categorical palette names are: "okabe", "carto", "r4", "polychrome", "tol", "bright", "light", "muted", "vibrant", "tableau", "classic", "alphabet", "tableau20", "kelly", and "fishy".

Patterns are added using the fillpattern R package. Options are "brick", "chevron", "fish", "grid", "herringbone", "hexagon", "octagon", "rain", "saw", "shingle", "rshingle", "stripe", and "wave", optionally abbreviated and/or suffixed with modifiers. For example, "hex10_sm" for the hexagon pattern rotated 10 degrees and shrunk by 2x. See fillpattern::fill_pattern() for complete documentation of options.

Shapes can be given as per base R - numbers 0 through 17 for various shapes, or the decimal value of an ascii character, e.g. a-z = 65:90; A-Z = 97:122 to use letters instead of shapes on the plot. Character strings may used as well.

See Also

Other visualization: adiv_boxplot(), adiv_corrplot(), bdiv_boxplot(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), plot_heatmap(), rare_corrplot(), rare_multiplot(), rare_stacked(), stats_corrplot(), taxa_boxplot(), taxa_corrplot(), taxa_heatmap(), taxa_stacked()

Examples

```
library(rbiom)

df <- adiv_table(rarefy(hmp50))
stats_boxplot(df, x = "Body Site")
stats_boxplot(df, x = "Sex", stat.by = "Body Site", layers = "be")
```

---

stats_corrplot *Visualize regression with scatterplots and trendlines.*

Description

Visualize regression with scatterplots and trendlines.

Usage

```
stats_corrplot(
  df,
  x,
  y = attr(df, "response"),
  layers = "tc",
  stat.by = NULL,
  facet.by = NULL,
  colors = TRUE,
  shapes = TRUE,
  test = "emmeans",
  fit = "gam",
  at = NULL,
  level = 0.95,
  p.adj = "fdr",
  p.top = Inf,
  alt = "!=",
  mu = 0,
  caption = TRUE,
  check = FALSE,
  ...
)
```

Arguments

df	The dataset (data.frame or tibble object). "Dataset fields" mentioned below should match column names in df. Required.
x	Dataset field with the x-axis values. Equivalent to the regr argument in stats_table(). Required.
y	A numeric metadata column name to use for the y-axis. Default: attr(df, 'response')
layers	One or more of c("trend", "confidence", "point", "name", "residual"). Single letter abbreviations are also accepted. For instance, c("trend", "point") is equivalent to c("t", "p") and "tp". Default: "tc"
stat.by	Dataset field with the statistical groups. Must be categorical. Default: NULL
facet.by	Dataset field(s) to use for faceting. Must be categorical. Default: NULL
colors	How to color the groups. Options are: TRUE - Automatically select colorblind-friendly colors. FALSE or NULL - Don't use colors. **a palette name -** Auto-select colors from this set. E.g. "okabe" **character vector -** Custom colors to use. E.g. c("red", "#00FF00") **named character vector -** Explicit mapping. E.g. c(Male = "blue", Female = "red") See "Aesthetics" section below for additional information. Default: TRUE
shapes	Shapes for each group. Options are similar to colors's: TRUE, FALSE, NULL, shape names (typically integers 0 - 17), or a named vector mapping groups to specific shape names. See "Aesthetics" section below for additional information. Default: TRUE
test	Method for computing p-values: 'none', 'emmeans', or 'emtrends'. Default: 'emmeans'
fit	How to fit the trendline. 'lm', 'log', or 'gam'. Default: 'gam'
at	Position(s) along the x-axis where the means or slopes should be evaluated. Default: NULL, which samples 100 evenly spaced positions and selects the position where the p-value is most significant.
level	The confidence level for calculating a confidence interval. Default: 0.95
p.adj	Method to use for multiple comparisons adjustment of p-values. Run p.adjust.methods for a list of available options. Default: "fdr"
p.top	Only display taxa with the most significant differences in abundance. If p.top is >= 1, then the p.top most significant taxa are displayed. If p.top is less than one, all taxa with an adjusted p-value <= p.top are displayed. Recommended to be used in combination with the taxa parameter to set a lower bound on the mean abundance of considered taxa. Default: Inf
alt	Alternative hypothesis direction. Options are '!=' (two-sided; not equal to mu), '<' (less than mu), or '>' (greater than mu). Default: '!='
mu	Reference value to test against. Default: 0
caption	Add methodology caption beneath the plot. Default: TRUE

check	Generate additional plots to aid in assessing data normality. Default: FALSE
...	Additional parameters to pass along to ggplot2 functions. Prefix a parameter name with a layer name to pass it to only that layer. For instance, p.size = 2 ensures only the points have their size set to 2.

Value

A ggplot2 plot. The computed data points, ggplot2 command, stats table, and stats table commands are available as $data, $code, $stats, and $stats$code, respectively.

Aesthetics

All built-in color palettes are colorblind-friendly. The available categorical palette names are: "okabe", "carto", "r4", "polychrome", "tol", "bright", "light", "muted", "vibrant", "tableau", "classic", "alphabet", "tableau20", "kelly", and "fishy".

Shapes can be given as per base R - numbers 0 through 17 for various shapes, or the decimal value of an ascii character, e.g. a-z = 65:90; A-Z = 97:122 to use letters instead of shapes on the plot. Character strings may used as well.

See Also

Other visualization: adiv_boxplot(), adiv_corrplot(), bdiv_boxplot(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), plot_heatmap(), rare_corrplot(), rare_multiplot(), rare_stacked(), stats_boxplot(), taxa_boxplot(), taxa_corrplot(), taxa_heatmap(), taxa_stacked()

Examples

```
library(rbiom)

biom <- subset(hmp50, `Body Site` %in% c('Saliva', 'Stool'))
df   <- adiv_table(rarefy(biom))
stats_corrplot(df, "age", stat.by = "body")
stats_corrplot(
  df       = df,
  x        = "Age",
  stat.by  = "Body Site",
  facet.by = "Sex",
  layers   = "trend" )
```

---

stats_table *Run non-parametric statistics on a data.frame.*

Description

A simple interface to lower-level statistics functions, including stats::wilcox.test(), stats::kruskal.test(), emmeans::emmeans(), and emmeans::emtrends().

Usage

```
stats_table(
  df,
  regr = NULL,
  resp = attr(df, "response"),
  stat.by = NULL,
  split.by = NULL,
  test = "emmeans",
  fit = "gam",
  at = NULL,
  level = 0.95,
  alt = "!=",
  mu = 0,
  p.adj = "fdr"
)
```

Arguments

df	The dataset (data.frame or tibble object). "Dataset fields" mentioned below should match column names in df. Required.
regr	Dataset field with the x-axis (independent; predictive) values. Must be numeric. Default: NULL
resp	Dataset field with the y-axis (dependent; response) values, such as taxa abundance or alpha diversity. Default: attr(df, 'response')
stat.by	Dataset field with the statistical groups. Must be categorical. Default: NULL
split.by	Dataset field(s) that the data should be split by prior to any calculations. Must be categorical. Default: NULL
test	Method for computing p-values: 'wilcox', 'kruskal', 'emmeans', or 'emtrends'. Default: 'emmeans'
fit	How to fit the trendline. 'lm', 'log', or 'gam'. Default: 'gam'
at	Position(s) along the x-axis where the means or slopes should be evaluated. Default: NULL, which samples 100 evenly spaced positions and selects the position where the p-value is most significant.
level	The confidence level for calculating a confidence interval. Default: 0.95
alt	Alternative hypothesis direction. Options are '!=' (two-sided; not equal to mu), '<' (less than mu), or '>' (greater than mu). Default: '!='
mu	Reference value to test against. Default: 0
p.adj	Method to use for multiple comparisons adjustment of p-values. Run p.adjust.methods for a list of available options. Default: "fdr"

Value

A tibble data.frame with fields from the table below. This tibble object provides the $code operator to print the R code used to generate the statistics.

Field	Description
.mean	Estimated marginal mean. See `emmeans::emmeans()`.
.mean.diff	Difference in means.
.slope	Trendline slope. See `emmeans::emtrends()`.
.slope.diff	Difference in slopes.
.h1	Alternate hypothesis.
.p.val	Probability that null hypothesis is correct.
.adj.p	`.p.val` after adjusting for multiple comparisons.
.effect.size	Effect size. See `emmeans::eff_size()`.
.lower	Confidence interval lower bound.
.upper	Confidence interval upper bound.
.se	Standard error.
.n	Number of samples.
.df	Degrees of freedom.
.stat	Wilcoxon or Kruskal-Wallis rank sum statistic.
.t.ratio	`.mean / .se`
.r.sqr	Percent of variation explained by the model.
.adj.r	`.r.sqr`, taking degrees of freedom into account.
.aic	Akaike Information Criterion (predictive models).
.bic	Bayesian Information Criterion (descriptive models).
.loglik	Log-likelihood goodness-of-fit score.
.fit.p	P-value for observing this fit by chance.

See Also

Other stats_tables: `adiv_stats()`, `bdiv_stats()`, `distmat_stats()`, `taxa_stats()`

Examples

```
library(rbiom)

biom <- rarefy(hmp50)

df <- taxa_table(biom, rank = "Family")
stats_table(df, stat.by = "Body Site")[,1:6]

df <- adiv_table(biom)
stats_table(df, stat.by = "Sex", split.by = "Body Site")[,1:7]
```

`subset` *Subset an rbiom object by sample names, OTU names, metadata, or taxonomy.*

Description

Dropping samples or OTUs will lead to observations being removed from the OTU matrix (`biom$counts`). OTUs and samples with zero observations are automatically removed from the rbiom object.

Usage

```
## S3 method for class 'rbiom'
subset(x, subset, clone = TRUE, ...)

## S3 method for class 'rbiom'
x[i, j, ..., clone = TRUE, drop = FALSE]

## S3 method for class 'rbiom'
na.omit(object, fields = ".all", clone = TRUE, ...)

subset_taxa(x, subset, clone = TRUE, ...)
```

Arguments

x	An rbiom object, such as from as_rbiom().
subset	Logical expression for rows to keep. See base::subset().
clone	Create a copy of biom before modifying. If FALSE, biom is modified in place as a side-effect. See speed ups for use cases. Default: TRUE
...	Not used.
i, j	The sample or OTU names to keep. Or a logical/integer vector indicating which sample names from biom$samples or biom$otus to keep. Subsetting with [i] takes i as samples, whereas [i,j] takes i as otus and j as samples (corresponding to [rows, cols] in the underlying biom$counts matrix).
drop	Not used
object	An rbiom object, such as from as_rbiom().
fields	Which metadata field(s) to check for NAs, or ".all" to check all metadata fields.

Value

An rbiom object.

See Also

Other transformations: modify_metadata, rarefy(), rarefy_cols(), slice_metadata, with()

Examples

```
library(rbiom)
library(dplyr)

# Subset to specific samples
biom <- hmp50[c('HMP20', 'HMP42', 'HMP12')]
biom$metadata

# Subset to specific OTUs
biom <- hmp50[c('LtbAci52', 'UncO2012'),] # <- Trailing ,
biom$taxonomy
```

```
# Subset to specific samples and OTUs
biom <- hmp50[c('LtbAci52', 'UncO2012'), c('HMP20', 'HMP42', 'HMP12')]
as.matrix(biom)

# Subset samples according to metadata
biom <- subset(hmp50, `Body Site` %in% c('Saliva') & Age < 25)
biom$metadata

# Subset OTUs according to taxonomy
biom <- subset_taxa(hmp50, Phylum == 'Cyanobacteria')
biom$taxonomy

# Remove samples with NA metadata values
biom <- mutate(hmp50, BS2 = na_if(`Body Site`, 'Saliva'))
biom$metadata
biom <- na.omit(biom)
biom$metadata
```

`taxa_boxplot` *Visualize BIOM data with boxplots.*

Description

Visualize BIOM data with boxplots.

Usage

```
taxa_boxplot(
  biom,
  x = NULL,
  rank = -1,
  layers = "x",
  taxa = 6,
  unc = "singly",
  other = FALSE,
  p.top = Inf,
  stat.by = x,
  facet.by = NULL,
  colors = TRUE,
  shapes = TRUE,
  patterns = FALSE,
  flip = FALSE,
  stripe = NULL,
  ci = "ci",
  level = 0.95,
  p.adj = "fdr",
  outliers = NULL,
```

```
  xlab.angle = "auto",
  p.label = 0.05,
  transform = "none",
  y.transform = "sqrt",
  caption = TRUE,
  ...
)
```

Arguments

biom	An rbiom object, such as from as_rbiom(). Any value accepted by as_rbiom() can also be given here.
x	A categorical metadata column name to use for the x-axis. Or NULL, which puts taxa along the x-axis. Default: NULL
rank	What rank(s) of taxa to display. E.g. "Phylum", "Genus", ".otu", etc. An integer vector can also be given, where 1 is the highest rank, 2 is the second highest, -1 is the lowest rank, -2 is the second lowest, and 0 is the OTU "rank". Run biom$ranks to see all options for a given rbiom object. Default: -1.
layers	One or more of c("bar", "box" ("x"), "violin", "dot", "strip", "crossbar", "errorbar", "linerange", "pointrange"). Single letter abbreviations are also accepted. For instance, c("box", "dot") is equivalent to c("x", "d") and "xd". Default: "x"
taxa	Which taxa to display. An integer value will show the top n most abundant taxa. A value 0 <= n < 1 will show any taxa with that mean abundance or greater (e.g. 0.1 implies >= 10%). A character vector of taxa names will show only those named taxa. Default: 6.
unc	How to handle unclassified, uncultured, and similarly ambiguous taxa names. Options are: "singly" - Replaces them with the OTU name. "grouped" - Replaces them with a higher rank's name. "drop" - Excludes them from the result. "asis" - To not check/modify any taxa names. Abbreviations are allowed. Default: "singly"
other	Sum all non-itemized taxa into an "Other" taxa. When FALSE, only returns taxa matched by the taxa argument. Specifying TRUE adds "Other" to the returned set. A string can also be given to imply TRUE, but with that value as the name to use instead of "Other". Default: FALSE
p.top	Only display taxa with the most significant differences in abundance. If p.top is >= 1, then the p.top most significant taxa are displayed. If p.top is less than one, all taxa with an adjusted p-value <= p.top are displayed. Recommended to be used in combination with the taxa parameter to set a lower bound on the mean abundance of considered taxa. Default: Inf
stat.by	Dataset field with the statistical groups. Must be categorical. Default: NULL
facet.by	Dataset field(s) to use for faceting. Must be categorical. Default: NULL
colors	How to color the groups. Options are:

TRUE - Automatically select colorblind-friendly colors.

FALSE or NULL - Don't use colors.

a palette name - Auto-select colors from this set. E.g. "okabe"

character vector - Custom colors to use. E.g. c("red", "#00FF00")

named character vector - Explicit mapping. E.g. c(Male = "blue", Female = "red")

See "Aesthetics" section below for additional information. Default: TRUE

shapes — Shapes for each group. Options are similar to colors's: TRUE, FALSE, NULL, shape names (typically integers 0 - 17), or a named vector mapping groups to specific shape names. See "Aesthetics" section below for additional information. Default: TRUE

patterns — Patterns for each group. Options are similar to colors's: TRUE, FALSE, NULL, pattern names ("brick", "chevron", "fish", "grid", etc), or a named vector mapping groups to specific pattern names. See "Aesthetics" section below for additional information. Default: FALSE

flip — Transpose the axes, so that taxa are present as rows instead of columns. Default: FALSE

stripe — Shade every other x position. Default: *same as flip*

ci — How to calculate min/max of the **crossbar**, **errorbar**, **linerange**, and **pointrange** layers. Options are: "ci" (confidence interval), "range", "sd" (standard deviation), "se" (standard error), and "mad" (median absolute deviation). The center mark of **crossbar** and **pointrange** represents the mean, except for "mad" in which case it represents the median. Default: "ci"

level — The confidence level for calculating a confidence interval. Default: 0.95

p.adj — Method to use for multiple comparisons adjustment of p-values. Run p.adjust.methods for a list of available options. Default: "fdr"

outliers — Show boxplot outliers? TRUE to always show. FALSE to always hide. NULL to only hide them when overlaying a dot or strip chart. Default: NULL

xlab.angle — Angle of the labels at the bottom of the plot. Options are "auto", '0', '30', and '90'. Default: "auto".

p.label — Minimum adjusted p-value to display on the plot with a bracket.

p.label = 0.05 - Show p-values that are <= 0.05.

p.label = 0 - Don't show any p-values on the plot.

p.label = 1 - Show all p-values on the plot.

If a numeric vector with more than one value is provided, they will be used as breaks for asterisk notation. Default: 0.05

transform — Transformation to apply. Options are: c("none", "rank", "log", "log1p", "sqrt", "percent"). "rank" is useful for correcting for non-normally distributions before applying regression statistics. Default: "none"

y.transform — The transformation to apply to the y-axis. Visualizing differences of both high- and low-abundance taxa is best done with a non-linear axis. Options are:

"sqrt" - square-root transformation

	`"log1p"` - log(y + 1) transformation `"none"` - no transformation These methods allow visualization of both high- and low-abundance taxa simultaneously, without complaint about 'zero' count observations. Default: `"sqrt"` Use `xaxis.transform` or `yaxis.transform` to pass custom values directly to ggplot2's `scale_*` functions.
`caption`	Add methodology caption beneath the plot. Default: `TRUE`
`...`	Additional parameters to pass along to ggplot2 functions. Prefix a parameter name with a layer name to pass it to only that layer. For instance, `d.size = 2` ensures only the points on the **dot** layer have their size set to 2.

Value

A `ggplot2` plot. The computed data points, ggplot2 command, stats table, and stats table commands are available as `$data`, `$code`, `$stats`, and `$stats$code`, respectively.

Aesthetics

All built-in color palettes are colorblind-friendly. The available categorical palette names are: `"okabe"`, `"carto"`, `"r4"`, `"polychrome"`, `"tol"`, `"bright"`, `"light"`, `"muted"`, `"vibrant"`, `"tableau"`, `"classic"`, `"alphabet"`, `"tableau20"`, `"kelly"`, and `"fishy"`.

Patterns are added using the fillpattern R package. Options are `"brick"`, `"chevron"`, `"fish"`, `"grid"`, `"herringbone"`, `"hexagon"`, `"octagon"`, `"rain"`, `"saw"`, `"shingle"`, `"rshingle"`, `"stripe"`, and `"wave"`, optionally abbreviated and/or suffixed with modifiers. For example, `"hex10_sm"` for the hexagon pattern rotated 10 degrees and shrunk by 2x. See `fillpattern::fill_pattern()` for complete documentation of options.

Shapes can be given as per base R - numbers 0 through 17 for various shapes, or the decimal value of an ascii character, e.g. a-z = 65:90; A-Z = 97:122 to use letters instead of shapes on the plot. Character strings may used as well.

See Also

Other taxa_abundance: `sample_sums()`, `taxa_clusters()`, `taxa_corrplot()`, `taxa_heatmap()`, `taxa_stacked()`, `taxa_stats()`, `taxa_sums()`, `taxa_table()`

Other visualization: `adiv_boxplot()`, `adiv_corrplot()`, `bdiv_boxplot()`, `bdiv_corrplot()`, `bdiv_heatmap()`, `bdiv_ord_plot()`, `plot_heatmap()`, `rare_corrplot()`, `rare_multiplot()`, `rare_stacked()`, `stats_boxplot()`, `stats_corrplot()`, `taxa_corrplot()`, `taxa_heatmap()`, `taxa_stacked()`

Examples

```
library(rbiom)

biom <- rarefy(hmp50)

taxa_boxplot(biom, stat.by = "Body Site", stripe = TRUE)
taxa_boxplot(biom, layers = "bed", rank = c("Phylum", "Genus"), flip = TRUE)
taxa_boxplot(
```

```
  biom    = subset(biom, `Body Site` %in% c('Saliva', 'Stool')),
  taxa    = 3,
  layers  = "ps",
  stat.by = "Body Site",
  colors  = c('Saliva' = "blue", 'Stool' = "red") )
```

`taxa_clusters` *Define sample kmeans clusters from taxa abundances.*

Description

Define sample kmeans clusters from taxa abundances.

Usage

```
taxa_clusters(biom, rank = ".otu", k = 5, ...)
```

Arguments

`biom`	An rbiom object, such as from `as_rbiom()`. Any value accepted by `as_rbiom()` can also be given here.
`rank`	Which taxa rank to use. E.g. `"Phylum"`, `"Genus"`, `".otu"`, etc. An integer can also be given, where `1` is the highest rank, `2` is the second highest, `-1` is the lowest rank, `-2` is the second lowest, and `0` is the OTU "rank". Run `biom$ranks` to see all options for a given rbiom object. Default: `.otu`.
`k`	Number of clusters. Default: `5L`
`...`	Passed on to `stats::kmeans()`.

Value

A numeric factor assigning samples to clusters.

See Also

Other taxa_abundance: `sample_sums()`, `taxa_boxplot()`, `taxa_corrplot()`, `taxa_heatmap()`, `taxa_stacked()`, `taxa_stats()`, `taxa_sums()`, `taxa_table()`

Other clustering: `bdiv_clusters()`

Examples

```
library(rbiom)

biom <- rarefy(hmp50)
biom$metadata$otu_cluster <- taxa_clusters(biom)

pull(biom, 'otu_cluster')[1:10]

bdiv_ord_plot(biom, layers = "p", stat.by = "otu_cluster")
```

`taxa_corrplot` *Visualize taxa abundance with scatterplots and trendlines.*

Description

Visualize taxa abundance with scatterplots and trendlines.

Usage

```
taxa_corrplot(
  biom,
  x,
  rank = -1,
  layers = "tc",
  taxa = 6,
  lineage = FALSE,
  unc = "singly",
  other = FALSE,
  stat.by = NULL,
  facet.by = NULL,
  colors = TRUE,
  shapes = TRUE,
  test = "emmeans",
  fit = "gam",
  at = NULL,
  level = 0.95,
  p.adj = "fdr",
  transform = "none",
  ties = "random",
  seed = 0,
  alt = "!=",
  mu = 0,
  caption = TRUE,
  check = FALSE,
  ...
)
```

Arguments

`biom`	An rbiom object, such as from `as_rbiom()`. Any value accepted by `as_rbiom()` can also be given here.
`x`	Dataset field with the x-axis values. Equivalent to the `regr` argument in `stats_table()`. Required.
`rank`	What rank(s) of taxa to display. E.g. `"Phylum"`, `"Genus"`, `".otu"`, etc. An integer vector can also be given, where `1` is the highest rank, `2` is the second highest, `-1` is the lowest rank, `-2` is the second lowest, and `0` is the OTU "rank". Run `biom$ranks` to see all options for a given rbiom object. Default: `-1`.

layers — One or more of `c("trend", "confidence", "point", "name", "residual")`. Single letter abbreviations are also accepted. For instance, `c("trend", "point")` is equivalent to `c("t", "p")` and `"tp"`. Default: `"tc"`

taxa — Which taxa to display. An integer value will show the top n most abundant taxa. A value 0 <= n < 1 will show any taxa with that mean abundance or greater (e.g. `0.1` implies >= 10%). A character vector of taxa names will show only those named taxa. Default: `6`.

lineage — Include all ranks in the name of the taxa. For instance, setting to `TRUE` will produce `Bacteria; Actinobacteria; Coriobacteriia; Coriobacteriales`. Otherwise the taxa name will simply be `Coriobacteriales`. You want to set this to TRUE when `unc = "asis"` and you have taxa names (such as *Incertae_Sedis*) that map to multiple higher level ranks. Default: `FALSE`

unc — How to handle unclassified, uncultured, and similarly ambiguous taxa names. Options are:

`"singly"` - Replaces them with the OTU name.

`"grouped"` - Replaces them with a higher rank's name.

`"drop"` - Excludes them from the result.

`"asis"` - To not check/modify any taxa names.

Abbreviations are allowed. Default: `"singly"`

other — Sum all non-itemized taxa into an "Other" taxa. When `FALSE`, only returns taxa matched by the `taxa` argument. Specifying `TRUE` adds "Other" to the returned set. A string can also be given to imply `TRUE`, but with that value as the name to use instead of "Other". Default: `FALSE`

stat.by — Dataset field with the statistical groups. Must be categorical. Default: `NULL`

facet.by — Dataset field(s) to use for faceting. Must be categorical. Default: `NULL`

colors — How to color the groups. Options are:

`TRUE` - Automatically select colorblind-friendly colors.

`FALSE` **or** `NULL` - Don't use colors.

a palette name - Auto-select colors from this set. E.g. `"okabe"`

character vector - Custom colors to use. E.g. `c("red", "#00FF00")`

named character vector - Explicit mapping. E.g. `c(Male = "blue", Female = "red")`

See "Aesthetics" section below for additional information. Default: `TRUE`

shapes — Shapes for each group. Options are similar to `colors`'s: `TRUE`, `FALSE`, `NULL`, shape names (typically integers 0 - 17), or a named vector mapping groups to specific shape names. See "Aesthetics" section below for additional information. Default: `TRUE`

test — Method for computing p-values: `'none'`, `'emmeans'`, or `'emtrends'`. Default: `'emmeans'`

fit — How to fit the trendline. `'lm'`, `'log'`, or `'gam'`. Default: `'gam'`

at — Position(s) along the x-axis where the means or slopes should be evaluated. Default: `NULL`, which samples 100 evenly spaced positions and selects the position where the p-value is most significant.

level	The confidence level for calculating a confidence interval. Default: 0.95
p.adj	Method to use for multiple comparisons adjustment of p-values. Run p.adjust.methods for a list of available options. Default: "fdr"
transform	Transformation to apply. Options are: c("none", "rank", "log", "log1p", "sqrt", "percent"). "rank" is useful for correcting for non-normally distributions before applying regression statistics. Default: "none"
ties	When transform="rank", how to rank identical values. Options are: c("average", "first", "last", "random", "max", "min"). See rank() for details. Default: "random"
seed	Random seed for permutations. Default: 0
alt	Alternative hypothesis direction. Options are '!=' (two-sided; not equal to mu), '<' (less than mu), or '>' (greater than mu). Default: '!='
mu	Reference value to test against. Default: 0
caption	Add methodology caption beneath the plot. Default: TRUE
check	Generate additional plots to aid in assessing data normality. Default: FALSE
...	Additional parameters to pass along to ggplot2 functions. Prefix a parameter name with a layer name to pass it to only that layer. For instance, p.size = 2 ensures only the points have their size set to 2.

Value

A ggplot2 plot. The computed data points, ggplot2 command, stats table, and stats table commands are available as $data, $code, $stats, and $stats$code, respectively.

Aesthetics

All built-in color palettes are colorblind-friendly. The available categorical palette names are: "okabe", "carto", "r4", "polychrome", "tol", "bright", "light", "muted", "vibrant", "tableau", "classic", "alphabet", "tableau20", "kelly", and "fishy".

Shapes can be given as per base R - numbers 0 through 17 for various shapes, or the decimal value of an ascii character, e.g. a-z = 65:90; A-Z = 97:122 to use letters instead of shapes on the plot. Character strings may used as well.

See Also

Other taxa_abundance: sample_sums(), taxa_boxplot(), taxa_clusters(), taxa_heatmap(), taxa_stacked(), taxa_stats(), taxa_sums(), taxa_table()

Other visualization: adiv_boxplot(), adiv_corrplot(), bdiv_boxplot(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), plot_heatmap(), rare_corrplot(), rare_multiplot(), rare_stacked(), stats_boxplot(), stats_corrplot(), taxa_boxplot(), taxa_heatmap(), taxa_stacked()

Examples

```
library(rbiom)

biom <- rarefy(subset(hmp50, `Body Site` %in% c('Buccal mucosa', 'Saliva')))
taxa_corrplot(biom, x = "BMI", stat.by = "Body Site", taxa = 'Streptococcus')
```

`taxa_heatmap` *Display taxa abundances as a heatmap.*

Description

Display taxa abundances as a heatmap.

Usage

```
taxa_heatmap(
  biom,
  rank = -1,
  taxa = 6,
  tracks = NULL,
  grid = "bilbao",
  other = FALSE,
  unc = "singly",
  lineage = FALSE,
  label = TRUE,
  label_size = NULL,
  rescale = "none",
  trees = TRUE,
  clust = "complete",
  dist = "euclidean",
  asp = 1,
  tree_height = 10,
  track_height = 10,
  legend = "right",
  title = TRUE,
  xlab.angle = "auto",
  ...
)
```

Arguments

`biom`	An rbiom object, such as from `as_rbiom()`. Any value accepted by `as_rbiom()` can also be given here.
`rank`	What rank(s) of taxa to display. E.g. `"Phylum"`, `"Genus"`, `".otu"`, etc. An integer vector can also be given, where `1` is the highest rank, `2` is the second highest, `-1` is the lowest rank, `-2` is the second lowest, and `0` is the OTU "rank". Run `biom$ranks` to see all options for a given rbiom object. Default: `-1`.

`taxa` Which taxa to display. An integer value will show the top n most abundant taxa. A value 0 <= n < 1 will show any taxa with that mean abundance or greater (e.g. `0.1` implies >= 10%). A character vector of taxa names will show only those named taxa. Default: `6`.

`tracks` A character vector of metadata fields to display as tracks at the top of the plot. Or, a list as expected by the `tracks` argument of `plot_heatmap()`. Default: `NULL`

`grid` Color palette name, or a list as expected `plot_heatmap()`. Default: `"bilbao"`

`other` Sum all non-itemized taxa into an "Other" taxa. When `FALSE`, only returns taxa matched by the `taxa` argument. Specifying `TRUE` adds "Other" to the returned set. A string can also be given to imply `TRUE`, but with that value as the name to use instead of "Other". Default: `FALSE`

`unc` How to handle unclassified, uncultured, and similarly ambiguous taxa names. Options are:

`"singly"` - Replaces them with the OTU name.

`"grouped"` - Replaces them with a higher rank's name.

`"drop"` - Excludes them from the result.

`"asis"` - To not check/modify any taxa names.

Abbreviations are allowed. Default: `"singly"`

`lineage` Include all ranks in the name of the taxa. For instance, setting to `TRUE` will produce `Bacteria; Actinobacteria; Coriobacteriia; Coriobacteriales`. Otherwise the taxa name will simply be `Coriobacteriales`. You want to set this to TRUE when `unc = "asis"` and you have taxa names (such as *Incertae_Sedis*) that map to multiple higher level ranks. Default: `FALSE`

`label` Label the matrix rows and columns. You can supply a list or logical vector of length two to control row labels and column labels separately, for example `label = c(rows = TRUE, cols = FALSE)`, or simply `label = c(TRUE, FALSE)`. Other valid options are `"rows"`, `"cols"`, `"both"`, `"bottom"`, `"right"`, and `"none"`. Default: `TRUE`

`label_size` The font size to use for the row and column labels. You can supply a numeric vector of length two to control row label sizes and column label sizes separately, for example `c(rows = 20, cols = 8)`, or simply `c(20, 8)`. Default: `NULL`, which computes: `pmax(8, pmin(20, 100 / dim(mtx)))`

`rescale` Rescale rows or columns to all have a common min/max. Options: `"none"`, `"rows"`, or `"cols"`. Default: `"none"`

`trees` Draw a dendrogram for rows (left) and columns (top). You can supply a list or logical vector of length two to control the row tree and column tree separately, for example `trees = c(rows = TRUE, cols = FALSE)`, or simply `trees = c(TRUE, FALSE)`. Other valid options are `"rows"`, `"cols"`, `"both"`, `"left"`, `"top"`, and `"none"`. Default: `TRUE`

`clust` Clustering algorithm for reordering the rows and columns by similarity. You can supply a list or character vector of length two to control the row and column clustering separately, for example `clust = c(rows = "complete", cols = NA)`, or simply `clust = c("complete", NA)`. Options are:

`FALSE` **or** `NA` **-** Disable reordering.

An `hclust` **class object** E.g. from `stats::hclust()`.

A method name - `"ward.D"`, `"ward.D2"`, `"single"`, `"complete"`, `"average"`, `"mcquitty"`, `"median"`, or `"centroid"`.

Default: `"complete"`

`dist` Distance algorithm to use when reordering the rows and columns by similarity. You can supply a list or character vector of length two to control the row and column clustering separately, for example `dist = c(rows = "euclidean", cols = "maximum")`, or simply `dist = c("euclidean", "maximum")`. Options are:

A `dist` **class object** E.g. from `stats::dist()` or `bdiv_distmat()`.

A method name - `"euclidean"`, `"maximum"`, `"manhattan"`, `"canberra"`, `"binary"`, or `"minkowski"`.

Default: `"euclidean"`

`asp` Aspect ratio (height/width) for entire grid. Default: `1` (square)

`tree_height, track_height`
The height of the dendrogram or annotation tracks as a percentage of the overall grid size. Use a numeric vector of length two to assign `c(top, left)` independently. Default: `10` (10% of the grid's height)

`legend` Where to place the legend. Options are: `"right"` or `"bottom"`. Default: `"right"`

`title` Plot title. Set to `TRUE` for a default title, `NULL` for no title, or any character string. Default: `TRUE`

`xlab.angle` Angle of the labels at the bottom of the plot. Options are `"auto"`, `'0'`, `'30'`, and `'90'`. Default: `"auto"`.

`...` Additional arguments to pass on to ggplot2::theme().

Value

A `ggplot2` plot. The computed data points and ggplot command are available as `$data` and `$code`, respectively.

Annotation Tracks

Metadata can be displayed as colored tracks above the heatmap. Common use cases are provided below, with more thorough documentation available at https://cmmr.github.io/rbiom .

```
## Categorical ----------------------------
tracks = "Body Site"
tracks = list('Body Site' = "bright")
tracks = list('Body Site' = c('Stool' = "blue", 'Saliva' = "green"))

## Numeric --------------------------------
tracks = "Age"
tracks = list('Age' = "reds")
```

```
## Multiple Tracks ------------------------
tracks = c("Body Site", "Age")
tracks = list('Body Site' = "bright", 'Age' = "reds")
tracks = list(
  'Body Site' = c('Stool' = "blue", 'Saliva' = "green"),
  'Age'       = list('colors' = "reds") )
```

The following entries in the track definitions are understood:

`colors` - A pre-defined palette name or custom set of colors to map to.

`range` - The c(min,max) to use for scale values.

`label` - Label for this track. Defaults to the name of this list element.

`side` - Options are `"top"` (default) or `"left"`.

`na.color` - The color to use for `NA` values.

`bins` - Bin a gradient into this many bins/steps.

`guide` - A list of arguments for guide_colorbar() or guide_legend().

All built-in color palettes are colorblind-friendly.

Categorical palette names: `"okabe"`, `"carto"`, `"r4"`, `"polychrome"`, `"tol"`, `"bright"`, `"light"`, `"muted"`, `"vibrant"`, `"tableau"`, `"classic"`, `"alphabet"`, `"tableau20"`, `"kelly"`, and `"fishy"`.

Numeric palette names: `"reds"`, `"oranges"`, `"greens"`, `"purples"`, `"grays"`, `"acton"`, `"bamako"`, `"batlow"`, `"bilbao"`, `"buda"`, `"davos"`, `"devon"`, `"grayC"`, `"hawaii"`, `"imola"`, `"lajolla"`, `"lapaz"`, `"nuuk"`, `"oslo"`, `"tokyo"`, `"turku"`, `"bam"`, `"berlin"`, `"broc"`, `"cork"`, `"lisbon"`, `"roma"`, `"tofino"`, `"vanimo"`, and `"vik"`.

See Also

Other taxa_abundance: `sample_sums()`, `taxa_boxplot()`, `taxa_clusters()`, `taxa_corrplot()`, `taxa_stacked()`, `taxa_stats()`, `taxa_sums()`, `taxa_table()`

Other visualization: `adiv_boxplot()`, `adiv_corrplot()`, `bdiv_boxplot()`, `bdiv_corrplot()`, `bdiv_heatmap()`, `bdiv_ord_plot()`, `plot_heatmap()`, `rare_corrplot()`, `rare_multiplot()`, `rare_stacked()`, `stats_boxplot()`, `stats_corrplot()`, `taxa_boxplot()`, `taxa_corrplot()`, `taxa_stacked()`

Examples

```
library(rbiom)

# Keep and rarefy the 10 most deeply sequenced samples.
hmp10 <- rarefy(hmp50, n = 10)

taxa_heatmap(hmp10, rank = "Phylum", tracks = "Body Site")

taxa_heatmap(hmp10, rank = "Genus", tracks = c("sex", "bo"))

taxa_heatmap(hmp10, rank = "Phylum", tracks = list(
  'Sex'       = list(colors = c(m = "#0000FF", f = "violetred")),
  'Body Site' = list(colors = "muted", label = "Source") ))
```

taxa_map *Map OTUs names to taxa names at a given rank.*

Description

Map OTUs names to taxa names at a given rank.

Usage

```
taxa_map(
  biom,
  rank = NULL,
  taxa = Inf,
  unc = "singly",
  lineage = FALSE,
  other = FALSE
)
```

Arguments

biom	An rbiom object, such as from `as_rbiom()`. Any value accepted by `as_rbiom()` can also be given here.
rank	When `NULL`, the entire biom$taxonomy data.frame is returned, transformed as per `unc`. Alternatively, a single taxonomic rank (rank name or integer position in `biom$ranks`) which returns a named character vector for mapping OTUs to taxa names.
taxa	Which taxa to display. An integer value will show the top n most abundant taxa. A value 0 <= n < 1 will show any taxa with that mean abundance or greater (e.g. `0.1` implies >= 10%). A character vector of taxa names will show only those named taxa. Default: `6`.
unc	How to handle unclassified, uncultured, and similarly ambiguous taxa names. Options are: `"singly"` - Replaces them with the OTU name. `"grouped"` - Replaces them with a higher rank's name. `"drop"` - Excludes them from the result. `"asis"` - To not check/modify any taxa names. Abbreviations are allowed. Default: `"singly"`
lineage	Include all ranks in the name of the taxa. For instance, setting to `TRUE` will produce `Bacteria; Actinobacteria; Coriobacteriia; Coriobacteriales`. Otherwise the taxa name will simply be `Coriobacteriales`. You want to set this to TRUE when `unc = "asis"` and you have taxa names (such as *Incertae_Sedis*) that map to multiple higher level ranks. Default: `FALSE`
other	Sum all non-itemized taxa into an "Other" taxa. When `FALSE`, only returns taxa matched by the `taxa` argument. Specifying `TRUE` adds "Other" to the returned set. A string can also be given to imply `TRUE`, but with that value as the name to use instead of "Other". Default: `FALSE`

Value

A tibble data.frame when `rank=NULL`, or a character vector named with the OTU names.

See Also

`pull.rbiom()`

Examples

```
library(rbiom)
library(dplyr, warn.conflicts = FALSE)

# In $taxonomy, .otu is the first column (like a row identifier)  -----
hmp50$taxonomy %>% head(4)

# In taxa_map, .otu is the last column (most precise rank)  -----------
taxa_map(hmp50) %>% head(4)

# Generate an OTU to Genus mapping  ----------------------------------
taxa_map(hmp50, "Genus") %>% head(4)

# Sometimes taxonomic names are incomplete ---------------------------
otus <- c('GemAsacc', 'GcbBacte', 'Unc58411')
taxa_map(hmp50, unc = "asis") %>% filter(.otu %in% otus) %>% select(Phylum:.otu)

# rbiom can replace them with unique placeholders --------------------
taxa_map(hmp50, unc = "singly") %>% filter(.otu %in% otus) %>% select(Class:.otu)

# Or collapse them into groups ---------------------------------------
taxa_map(hmp50, unc = "grouped") %>% filter(.otu %in% otus) %>% select(Class:Genus)
```

---

taxa_stacked *Display taxa abundances as a stacked bar graph.*

Description

Display taxa abundances as a stacked bar graph.

Usage

```
taxa_stacked(
  biom,
  rank = -1,
  taxa = 6,
  colors = TRUE,
  patterns = FALSE,
  label.by = NULL,
  order.by = NULL,
```

```
  facet.by = NULL,
  dist = "euclidean",
  clust = "complete",
  other = TRUE,
  unc = "singly",
  lineage = FALSE,
  xlab.angle = 90,
  ...
)
```

Arguments

`biom`	An rbiom object, such as from `as_rbiom()`. Any value accepted by `as_rbiom()` can also be given here.
`rank`	What rank(s) of taxa to display. E.g. `"Phylum"`, `"Genus"`, `".otu"`, etc. An integer vector can also be given, where `1` is the highest rank, `2` is the second highest, `-1` is the lowest rank, `-2` is the second lowest, and `0` is the OTU "rank". Run `biom$ranks` to see all options for a given rbiom object. Default: `-1`.
`taxa`	Which taxa to display. An integer value will show the top n most abundant taxa. A value 0 <= n < 1 will show any taxa with that mean abundance or greater (e.g. `0.1` implies >= 10%). A character vector of taxa names will show only those named taxa. Default: `6`.
`colors`, `patterns`	A character vector of colors or patterns to use in the graph. A named character vector can be used to map taxon names to specific colors or patterns. Set to `TRUE` to auto-select colors or patterns, or to `FALSE` to disable per-taxa colors or patterns. Default: `colors=TRUE`, `patterns=FALSE`.
`label.by`, `order.by`	What metadata column to use for labeling and/or sorting the samples across the x-axis. Set `label.by='.sample'` to display sample names. When `order.by=NULL`, samples are arranged based on `dist` and `clust`, below. Default: `label.by=NULL`, `order.by=NULL`.
`facet.by`	Dataset field(s) to use for faceting. Must be categorical. Default: `NULL`
`dist`, `clust`	Distance (`stats::dist()`) and clustering (`stats::hclust()`) methods to use for automatically arranging samples along the x-axis to put samples with similar composition near one another. Default: `dist="euclidean"`, `clust="complete"`.
`other`	Sum all non-itemized taxa into an "Other" taxa. When `FALSE`, only returns taxa matched by the `taxa` argument. Specifying `TRUE` adds "Other" to the returned set. A string can also be given to imply `TRUE`, but with that value as the name to use instead of "Other". Default: `FALSE`
`unc`	How to handle unclassified, uncultured, and similarly ambiguous taxa names. Options are: `"singly"` - Replaces them with the OTU name. `"grouped"` - Replaces them with a higher rank's name. `"drop"` - Excludes them from the result. `"asis"` - To not check/modify any taxa names.

Abbreviations are allowed. Default: "singly"

lineage — Include all ranks in the name of the taxa. For instance, setting to TRUE will produce Bacteria; Actinobacteria; Coriobacteriia; Coriobacteriales. Otherwise the taxa name will simply be Coriobacteriales. You want to set this to TRUE when unc = "asis" and you have taxa names (such as *Incertae_Sedis*) that map to multiple higher level ranks. Default: FALSE

xlab.angle — Angle of the labels at the bottom of the plot. Options are "auto", '0', '30', and '90'. Default: "auto".

... — Parameters for underlying functions. Prefixing parameter names with a layer name ensures that a particular parameter is passed to, and only to, that layer.

Value

A ggplot2 plot. The computed data points and ggplot command are available as $data and $code, respectively.

See Also

Other taxa_abundance: sample_sums(), taxa_boxplot(), taxa_clusters(), taxa_corrplot(), taxa_heatmap(), taxa_stats(), taxa_sums(), taxa_table()

Other visualization: adiv_boxplot(), adiv_corrplot(), bdiv_boxplot(), bdiv_corrplot(), bdiv_heatmap(), bdiv_ord_plot(), plot_heatmap(), rare_corrplot(), rare_multiplot(), rare_stacked(), stats_boxplot(), stats_corrplot(), taxa_boxplot(), taxa_corrplot(), taxa_heatmap()

Examples

```
library(rbiom)

biom <- rarefy(hmp50)

taxa_stacked(biom, rank="Phylum")

taxa_stacked(biom, rank = "genus", facet.by = "body site")
```

---

taxa_stats *Test taxa abundances for associations with metadata.*

Description

A convenience wrapper for taxa_table() + stats_table().

Usage

```
taxa_stats(
  biom,
  regr = NULL,
  stat.by = NULL,
  rank = -1,
  taxa = 6,
  lineage = FALSE,
  unc = "singly",
  other = FALSE,
  split.by = NULL,
  transform = "none",
  test = "emmeans",
  fit = "gam",
  at = NULL,
  level = 0.95,
  alt = "!=",
  mu = 0,
  p.adj = "fdr"
)
```

Arguments

biom	An rbiom object, such as from `as_rbiom()`. Any value accepted by `as_rbiom()` can also be given here.
regr	Dataset field with the x-axis (independent; predictive) values. Must be numeric. Default: `NULL`
stat.by	Dataset field with the statistical groups. Must be categorical. Default: `NULL`
rank	What rank(s) of taxa to display. E.g. `"Phylum"`, `"Genus"`, `".otu"`, etc. An integer vector can also be given, where `1` is the highest rank, `2` is the second highest, `-1` is the lowest rank, `-2` is the second lowest, and `0` is the OTU "rank". Run `biom$ranks` to see all options for a given rbiom object. Default: `-1`.
taxa	Which taxa to display. An integer value will show the top n most abundant taxa. A value 0 <= n < 1 will show any taxa with that mean abundance or greater (e.g. `0.1` implies >= 10%). A character vector of taxa names will show only those named taxa. Default: `6`.
lineage	Include all ranks in the name of the taxa. For instance, setting to `TRUE` will produce `Bacteria; Actinobacteria; Coriobacteriia; Coriobacteriales`. Otherwise the taxa name will simply be `Coriobacteriales`. You want to set this to TRUE when `unc = "asis"` and you have taxa names (such as *Incertae_Sedis*) that map to multiple higher level ranks. Default: `FALSE`
unc	How to handle unclassified, uncultured, and similarly ambiguous taxa names. Options are: `"singly"` - Replaces them with the OTU name. `"grouped"` - Replaces them with a higher rank's name. `"drop"` - Excludes them from the result.

	`"asis"` - To not check/modify any taxa names. Abbreviations are allowed. Default: `"singly"`
`other`	Sum all non-itemized taxa into an "Other" taxa. When `FALSE`, only returns taxa matched by the `taxa` argument. Specifying `TRUE` adds "Other" to the returned set. A string can also be given to imply `TRUE`, but with that value as the name to use instead of "Other". Default: `FALSE`
`split.by`	Dataset field(s) that the data should be split by prior to any calculations. Must be categorical. Default: `NULL`
`transform`	Transformation to apply. Options are: `c("none", "rank", "log", "log1p", "sqrt", "percent")`. `"rank"` is useful for correcting for non-normally distributions before applying regression statistics. Default: `"none"`
`test`	Method for computing p-values: `'wilcox'`, `'kruskal'`, `'emmeans'`, or `'emtrends'`. Default: `'emmeans'`
`fit`	How to fit the trendline. `'lm'`, `'log'`, or `'gam'`. Default: `'gam'`
`at`	Position(s) along the x-axis where the means or slopes should be evaluated. Default: `NULL`, which samples 100 evenly spaced positions and selects the position where the p-value is most significant.
`level`	The confidence level for calculating a confidence interval. Default: `0.95`
`alt`	Alternative hypothesis direction. Options are `'!='` (two-sided; not equal to `mu`), `'<'` (less than `mu`), or `'>'` (greater than `mu`). Default: `'!='`
`mu`	Reference value to test against. Default: `0`
`p.adj`	Method to use for multiple comparisons adjustment of p-values. Run `p.adjust.methods` for a list of available options. Default: `"fdr"`

Value

A tibble data.frame with fields from the table below. This tibble object provides the `$code` operator to print the R code used to generate the statistics.

Field	Description
.mean	Estimated marginal mean. See `emmeans::emmeans()`.
.mean.diff	Difference in means.
.slope	Trendline slope. See `emmeans::emtrends()`.
.slope.diff	Difference in slopes.
.h1	Alternate hypothesis.
.p.val	Probability that null hypothesis is correct.
.adj.p	`.p.val` after adjusting for multiple comparisons.
.effect.size	Effect size. See `emmeans::eff_size()`.
.lower	Confidence interval lower bound.
.upper	Confidence interval upper bound.
.se	Standard error.
.n	Number of samples.
.df	Degrees of freedom.
.stat	Wilcoxon or Kruskal-Wallis rank sum statistic.
.t.ratio	`.mean` / `.se`

.r.sqr	Percent of variation explained by the model.
.adj.r	`.r.sqr`, taking degrees of freedom into account.
.aic	Akaike Information Criterion (predictive models).
.bic	Bayesian Information Criterion (descriptive models).
.loglik	Log-likelihood goodness-of-fit score.
.fit.p	P-value for observing this fit by chance.

See Also

Other taxa_abundance: `sample_sums()`, `taxa_boxplot()`, `taxa_clusters()`, `taxa_corrplot()`, `taxa_heatmap()`, `taxa_stacked()`, `taxa_sums()`, `taxa_table()`

Other stats_tables: `adiv_stats()`, `bdiv_stats()`, `distmat_stats()`, `stats_table()`

Examples

```
library(rbiom)

biom <- rarefy(hmp50)

taxa_stats(biom, stat.by = "Body Site", rank = "Family")[,1:6]
```

`taxa_sums` *Get summary taxa abundances.*

Description

Get summary taxa abundances.

Usage

```
taxa_sums(
  biom,
  rank = -1,
  sort = NULL,
  lineage = FALSE,
  unc = "singly",
  transform = "none"
)

taxa_means(
  biom,
  rank = -1,
  sort = NULL,
  lineage = FALSE,
  unc = "singly",
  transform = "none"
```

```
)

taxa_apply(
  biom,
  FUN,
  rank = -1,
  sort = NULL,
  lineage = FALSE,
  unc = "singly",
  transform = "none",
  ...
)
```

Arguments

biom	An rbiom object, such as from as_rbiom(). Any value accepted by as_rbiom() can also be given here.
rank	What rank(s) of taxa to display. E.g. "Phylum", "Genus", ".otu", etc. An integer vector can also be given, where 1 is the highest rank, 2 is the second highest, -1 is the lowest rank, -2 is the second lowest, and 0 is the OTU "rank". Run biom$ranks to see all options for a given rbiom object. Default: -1.
sort	Sort the result. Options: NULL, "asc", or "desc", where NULL will not sort the result. "asc" will sort in ascending order (smallest to largest), and "desc" in descending order (largest to smallest). Ignored when the result is not a simple numeric vector. Default: NULL
lineage	Include all ranks in the name of the taxa. For instance, setting to TRUE will produce Bacteria; Actinobacteria; Coriobacteriia; Coriobacteriales. Otherwise the taxa name will simply be Coriobacteriales. You want to set this to TRUE when unc = "asis" and you have taxa names (such as *Incertae_Sedis*) that map to multiple higher level ranks. Default: FALSE
unc	How to handle unclassified, uncultured, and similarly ambiguous taxa names. Options are: "singly" - Replaces them with the OTU name. "grouped" - Replaces them with a higher rank's name. "drop" - Excludes them from the result. "asis" - To not check/modify any taxa names. Abbreviations are allowed. Default: "singly"
transform	Transformation to apply. Options are: c("none", "rank", "log", "log1p", "sqrt", "percent"). "rank" is useful for correcting for non-normally distributions before applying regression statistics. Default: "none"
FUN	The function to apply to each row of the taxa_matrix().
...	Optional arguments to FUN.

Value

For taxa_sums and taxa_means, a named numeric vector. For taxa_apply, a named vector or list with the results of FUN. The names are the taxa IDs.

See Also

Other taxa_abundance: sample_sums(), taxa_boxplot(), taxa_clusters(), taxa_corrplot(), taxa_heatmap(), taxa_stacked(), taxa_stats(), taxa_table()

Examples

```
library(rbiom)

taxa_sums(hmp50) %>% head(4)

taxa_means(hmp50, 'Family') %>% head(5)

taxa_apply(hmp50, max) %>% head(5)

taxa_apply(hmp50, fivenum) %>% head(5)
```

---

taxa_table *Taxa abundances per sample.*

Description

`taxa_matrix()` - Accepts a single `rank` and returns a matrix.

`taxa_table()` - Can accept more than one `rank` and returns a tibble data.frame.

Usage

```
taxa_table(
  biom,
  rank = -1,
  taxa = 6,
  lineage = FALSE,
  md = ".all",
  unc = "singly",
  other = FALSE,
  transform = "none",
  ties = "random",
  seed = 0
)

taxa_matrix(
  biom,
  rank = -1,
  taxa = NULL,
  lineage = FALSE,
  sparse = FALSE,
  unc = "singly",
  other = FALSE,
```

```
  transform = "none",
  ties = "random",
  seed = 0
)
```

Arguments

biom	An rbiom object, such as from as_rbiom(). Any value accepted by as_rbiom() can also be given here.
rank	What rank(s) of taxa to display. E.g. "Phylum", "Genus", ".otu", etc. An integer vector can also be given, where 1 is the highest rank, 2 is the second highest, -1 is the lowest rank, -2 is the second lowest, and 0 is the OTU "rank". Run biom$ranks to see all options for a given rbiom object. Default: -1.
taxa	Which taxa to display. An integer value will show the top n most abundant taxa. A value 0 <= n < 1 will show any taxa with that mean abundance or greater (e.g. 0.1 implies >= 10%). A character vector of taxa names will show only those named taxa. Default: 6.
lineage	Include all ranks in the name of the taxa. For instance, setting to TRUE will produce Bacteria; Actinobacteria; Coriobacteriia; Coriobacteriales. Otherwise the taxa name will simply be Coriobacteriales. You want to set this to TRUE when unc = "asis" and you have taxa names (such as *Incertae_Sedis*) that map to multiple higher level ranks. Default: FALSE
md	Dataset field(s) to include in the output data frame, or '.all' to include all metadata fields. Default: '.all'
unc	How to handle unclassified, uncultured, and similarly ambiguous taxa names. Options are: "singly" - Replaces them with the OTU name. "grouped" - Replaces them with a higher rank's name. "drop" - Excludes them from the result. "asis" - To not check/modify any taxa names. Abbreviations are allowed. Default: "singly"
other	Sum all non-itemized taxa into an "Other" taxa. When FALSE, only returns taxa matched by the taxa argument. Specifying TRUE adds "Other" to the returned set. A string can also be given to imply TRUE, but with that value as the name to use instead of "Other". Default: FALSE
transform	Transformation to apply. Options are: c("none", "rank", "log", "log1p", "sqrt", "percent"). "rank" is useful for correcting for non-normally distributions before applying regression statistics. Default: "none"
ties	When transform="rank", how to rank identical values. Options are: c("average", "first", "last", "random", "max", "min"). See rank() for details. Default: "random"
seed	Random seed for permutations. Default: 0
sparse	If TRUE, returns a slam::simple_triplet_matrix(), otherwise returns a normal R matrix object. Default: FALSE

Value

`taxa_matrix()` - A numeric matrix with taxa as rows, and samples as columns.

`taxa_table()` - A tibble data frame with column names .sample, .taxa, .abundance, and any requested by `md`.

See Also

Other taxa_abundance: `sample_sums()`, `taxa_boxplot()`, `taxa_clusters()`, `taxa_corrplot()`, `taxa_heatmap()`, `taxa_stacked()`, `taxa_stats()`, `taxa_sums()`

Examples

```
library(rbiom)

hmp50$ranks

taxa_matrix(hmp50, 'Phylum')[1:4,1:6]

taxa_table(hmp50, 'Phylum')
```

`tree_subset` *Create a subtree by specifying tips to keep.*

Description

Create a subtree by specifying tips to keep.

Usage

```
tree_subset(tree, tips)
```

Arguments

`tree`	A phylo object, as returned from `read_tree()`.
`tips`	A character, numeric, or logical vector of tips to keep.

Value

A `phylo` object for the subtree.

See Also

Other phylogeny: `read_tree()`

Examples

```
library(rbiom)

infile <- system.file("extdata", "newick.tre", package = "rbiom")
tree <- read_tree(infile)
tree

subtree <- tree_subset(tree, tips = head(tree$tip.label))
subtree
```

`with` *Evaluate expressions on metadata.*

Description

`with()` will return the result of your expression. `within()` will return an rbiom object.

Usage

```
## S3 method for class 'rbiom'
with(data, expr, ...)

## S3 method for class 'rbiom'
within(data, expr, clone = TRUE, ...)
```

Arguments

`data`	An rbiom object, such as from `as_rbiom()`.
`expr`	Passed on to `base::with()` or `base::within()`.
`...`	Not used.
`clone`	Create a copy of `biom` before modifying. If `FALSE`, `biom` is modified in place as a side-effect. See speed ups for use cases. Default: `TRUE`

Value

See description.

See Also

Other transformations: `modify_metadata`, `rarefy()`, `rarefy_cols()`, `slice_metadata`, `subset()`

Examples

```
library(rbiom)

with(hmp50, table(`Body Site`, Sex))

biom <- within(hmp50, {
  age_bin = cut(Age, 5)
  bmi_bin = cut(BMI, 5)
})
biom$metadata
```

`write_biom` *Save an rbiom object to a file.*

Description

Automatically creates directories and adds compression based on file name.

`write_biom()` - According to BIOM format specification.

`write_xlsx()` - Raw data and summary tables in Excel file format. See details.

`write_fasta()` - Sequences only in fasta format. `biom` may also be a named character vector.

`write_tree()` - Phylogenetic tree only in newick format. `biom` may also be a phylo object.

`write_counts()`, `write_metadata()`, `write_taxonomy()` - Tab-separated values.

Usage

```
write_biom(biom, file, format = "json")

write_metadata(biom, file, quote = FALSE, sep = "\t", ...)

write_counts(biom, file, quote = FALSE, sep = "\t", ...)

write_taxonomy(biom, file, quote = FALSE, sep = "\t", ...)

write_fasta(biom, file = NULL)

write_tree(biom, file = NULL)

write_xlsx(biom, file, depth = 0.1, n = NULL, seed = 0, unc = "singly")
```

Arguments

`biom`	An rbiom object, such as from `as_rbiom()`. Any value accepted by `as_rbiom()` can also be given here.

file	Path to the output file. File names ending in .gz or .bz2 will be compressed accordingly. Setting file=NULL for write_fasta(), write_tree(), and write_biom(format='json'), and returns a string of the output which would have been written. For write_biom(format='tab'), file=NULL returns the tibble that would have been written.
format	Options are "tab", "json", and "hdf5", corresponding to classic tabular format, BIOM format version 1.0 and biom version 2.1, respectively. NOTE: to write HDF5 formatted BIOM files, the BioConductor R package rhdf5 must be installed. Default: "json"
quote, sep, ...	Parameters passed on to write.table(). Default: quote=FALSE, sep="\t"
depth, n	Passed on to rarefy_cols(). For write_xlsx() only, depth=0 disables rarefaction. Default: depth=0.1, n=NULL
seed	Random seed to use in rarefying. See rarefy_cols() function for details. Default: 0
unc	How to handle unclassified, uncultured, and similarly ambiguous taxa names. Options are: "singly" - Replaces them with the OTU name. "grouped" - Replaces them with a higher rank's name. "drop" - Excludes them from the result. "asis" - To not check/modify any taxa names. Abbreviations are allowed. Default: "singly"

Details

For write_xlsx(), attributes(biom) are saved as additional worksheets if the attribute is a data frame, matrix, or dist -class object. An attribute named 'Reads Per Step' is treated specially and merged with the usual 'Reads Per Sample' tab.

Value

The normalized filepath that was written to (invisibly), unless file=NULL (see file argument above).

Examples

```
library(rbiom)

write_tree(hmp50) %>% substr(1, 50)

if (FALSE) {

  hmp10        <- hmp50$clone()
  hmp10$counts <- hmp10$counts[,1:10] %>% rarefy_cols()

  attr(hmp10, "Weighted UniFrac")   <- bdiv_distmat(hmp10, 'unifrac')
 attr(hmp10, "Unweighted Jaccard") <- bdiv_distmat(hmp10, 'jaccard', weighted=FALSE)

  outfile <- write_xlsx(hmp10, tempfile(fileext = ".xlsx"))
}
```

Index