Package: comparpq (via r-universe)

June 8, 2026

Type Package

Title An extension to {MiscMetabar} and {phyloseq} package to compare phyloseq object

Version 0.1.3

Description Compare phyloseq object from the R phyloseq package. This package provides tools and functions to facilitate comparison and analysis of multiple phyloseq objects.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 2.10), MiscMetabar, phyloseq, S7

Imports dplyr, ggplot2, tibble, formattable

URL https://github.com/adrientaudiere/comparpq, https://adrientaudiere.github.io/comparpq/

BugReports https://github.com/adrientaudiere/comparpq/issues

Suggests ALDEx2, ANCOMBC, ComplexUpset, dabestr, testthat (>= 3.0.0), ggrain, indicspecies, ggtext, ggVennDiagram, patchwork, gtsummary, robservable, knitr, permute, rmarkdown, MicrobiomeBenchmarkData, packcircles

LazyData true

Config/testthat/edition 3

Config/testthat/parallel true

VignetteBuilder knitr

Language en-US

Config/pak/sysreqs cmake libglpk-dev make libbz2-dev libicu-dev libjpeg-dev liblzma-dev libpng-dev libuv1-dev libxml2-dev xz-utils zlib1g-dev

Repository https://adrientaudiere.r-universe.dev

Date/Publication 2026-03-13 14:23:10 UTC

RemoteUrl https://github.com/adrientaudiere/comparpq

RemoteRef HEAD

RemoteSha 9be45e69b2880681d8583db9de74bef54151b7f2

Contents

add_external_seq_pq . . . 3
add_phyloseq . . . 4
add_shuffle_seq_pq . . . 4
adonis_lpq . . . 5
aldex_lpq . . . 7
ancombc_lpq . . . 8
apply_to_lpq . . . 10
bubbles_pq . . . 11
compare_refseq . . . 13
div_pq . . . 15
estim_cor_lpq . . . 16
estim_cor_pq . . . 18
estim_diff_lpq . . . 19
estim_diff_pq . . . 20
factor_formatter . . . 22
filter_common_lpq . . . 23
formattable_lpq . . . 24
formattable_lpq_full . . . 25
gg_aldex_plot . . . 26
gg_bubbles_pq . . . 28
gg_hill_lpq . . . 31
gg_maaslin3_plot . . . 33
glmulti_lpq . . . 37
list_phyloseq . . . 39
maaslin3_pq . . . 41
merge_lpq . . . 43
midasim_pq . . . 45
multipatt_lpq . . . 48
multiply_counts_pq . . . 50
n_levels_lpq . . . 53
permute_da_pq . . . 54
rainplot_taxo_na . . . 56
remove_phyloseq . . . 57
rename_ranks_pq . . . 58
resolve_taxo_conflict . . . 59
select_ranks_pq . . . 60
shared_mod_lpq . . . 61
simple_venn_pq . . . 62
taxo2tree . . . 65
taxtab_replace_pattern_by_NA . . . 67
tc_bar . . . 68

tc_circle 69
tc_congruence_metrics 70
tc_df_pq 72
tc_linked_trees 73
tc_metrics_mock 75
tc_metrics_mock_vec 76
tc_points_matrix 78
tc_sankey 79
update_list_phyloseq 80
upset_lpq 81

Index **83**

`add_external_seq_pq` *Add external sequences to a phyloseq object*

Description

Useful to compute true negative values with functions `tc_metrics_mock()` and `tc_metrics_mock_vec()`.

Note the that the tax_table for additional sequences is full of NA and that the corresponding otu_table is full of 0.

Usage

```
add_external_seq_pq(physeq, ext_seqs, prefix = "external_")
```

Arguments

`physeq`	(required) A `phyloseq-class` object obtained using the `phyloseq` package.
`ext_seqs`	(DNAStringSet, required) A DNAStringSet object containing external sequences to add.
`prefix`	(character, default "external_") A prefix to add to the taxa name.

Value

A phyloseq object

Author(s)

Adrien Taudière

Examples

```
add_external_seq_pq(
  data_fungi_mini,
  Biostrings::readDNAStringSet(
    system.file("extdata/ex_little.fasta", package = "MiscMetabar")
  )
)
```

add_phyloseq *Add a phyloseq object to a list_phyloseq*

Description

Add a phyloseq object to a list_phyloseq

Usage

```
add_phyloseq(x, physeq, name = NULL)
```

Arguments

x	(required) A list_phyloseq object.
physeq	(required) A phyloseq object to add.
name	(character, default NULL) Optional name for the new phyloseq object. If NULL, a name is generated automatically.

Value

A new list_phyloseq object with the added phyloseq

add_shuffle_seq_pq *Add fake sequences by shuffling existing ones in a phyloseq object*

Description

Useful to compute true negative values with functions `tc_metrics_mock()` and `tc_metrics_mock_vec()`.

Note the that the tax_table for additional sequences is full of NA and that the corresponding otu_table is full of 0.

Usage

```
add_shuffle_seq_pq(physeq, n_fake = NULL, prop_fake = NULL, prefix = "fake_")
```

Arguments

physeq	(required) A `phyloseq-class` object obtained using the `phyloseq` package.
n_fake	(integer, default NULL) A number of fake taxa to add. Must specify either `n_fake` or `prop_fake`, not both.
prop_fake	(numeric, default NULL) A proportion of fake taxa to add. Must specify either `n_fake` or `prop_fake`, not both.
prefix	(character, default "fake_") A prefix to add to the taxa name.

Value

A phyloseq object

Author(s)

Adrien Taudière

Examples

```
d_fake_F <- data_fungi_mini |>
  add_shuffle_seq_pq(prop_fake = 0.1)
ntaxa(d_fake_F) - ntaxa(data_fungi_mini)
```

---

adonis_lpq *PERMANOVA analysis on each phyloseq object in a list_phyloseq*

Description

Performs a PERMANOVA analysis using `MiscMetabar::adonis_pq()` on each phyloseq object in a list_phyloseq and returns a summary table with the results.

Usage

```
adonis_lpq(
  x,
  formula,
  dist_method = "bray",
  na_remove = FALSE,
  correction_for_sample_size = FALSE,
  rarefy_nb_seqs = FALSE,
  by = "margin",
  verbose = TRUE,
  ...
)
```

Arguments

x	(required) A list_phyloseq object.
formula	(character, required) The right part of a formula for `vegan::adonis2()`. Variables must be present in the `sample_data` slot of **all** phyloseq objects. The formula should contain variables that are in the shared modalities.
dist_method	(character, default "bray") The distance method to use. See `phyloseq::distance()` for available methods.
na_remove	(logical, default FALSE) If TRUE, samples with NA values in the formula variables are removed before analysis.

`correction_for_sample_size`
: (logical, default FALSE) If TRUE, adds library size to the formula following Weiss et al. 2017 recommendations.

`rarefy_nb_seqs`
: (logical, default FALSE) If TRUE, rarefy each sample before computing distances.

`by`
: (character, default "margin") The by argument passed to `vegan::adonis2()`. Options are "terms", "margin", or NULL.

`verbose`
: (logical, default TRUE) If TRUE, print progress messages.

`...`
: Additional arguments passed to `MiscMetabar::adonis_pq()`.

Details

This function requires that the list_phyloseq type is NOT `SEPARATE_ANALYSIS`, as the formula must contain variables that are common across all phyloseq objects.

The function is a wrapper around `MiscMetabar::adonis_pq()`, which itself wraps `vegan::adonis2()`.

Value

A tibble with one row per phyloseq object and the following columns:

name Name of the phyloseq object

term The term(s) from the formula (one row per term if multiple)

Df Degrees of freedom

SumOfSqs Sum of squares

R2 R-squared value

F F statistic

Pr(>F) P-value

partial_R2 Partial R-squared (if available)

See Also

`MiscMetabar::adonis_pq()`, `vegan::adonis2()`

Examples

```
## Not run:
lpq <- list_phyloseq(
  list(
    fungi = data_fungi,
    fungi_clust = postcluster_pq(data_fungi)
  ),
  same_bioinfo_pipeline = FALSE
)

# Run PERMANOVA on each phyloseq object
results <- adonis_lpq(lpq, formula = "Height+Time", na_remove = TRUE)
results
```

```
# With Jaccard distance
results_jaccard <- adonis_lpq(lpq, formula = "Height", dist_method = "jaccard")

## End(Not run)
```

`aldex_lpq` *ALDEx2 analysis on each phyloseq object in a list_phyloseq*

Description

Performs ALDEx2 differential abundance analysis using `MiscMetabar::aldex_pq()` on each phyloseq object in a list_phyloseq and returns a combined result table.

Usage

```
aldex_lpq(x, bifactor, modalities = NULL, gamma = 0.5, verbose = TRUE, ...)
```

Arguments

`x`	(required) A list_phyloseq object.
`bifactor`	(character, required) The name of a dichotomous column in `sample_data` to use as the grouping factor. Must be present in **all** phyloseq objects.
`modalities`	(character vector, default NULL) Two modalities of `bifactor` to compare. If NULL, uses the two levels present in the data.
`gamma`	(numeric, default 0.5) The gamma parameter for ALDEx2.
`verbose`	(logical, default TRUE) If TRUE, print progress messages.
`...`	Additional arguments passed to `MiscMetabar::aldex_pq()`.

Details

This function requires that the list_phyloseq type is NOT `SEPARATE_ANALYSIS`, as the bifactor must be common across all phyloseq objects.

When no common taxa exist across the phyloseq objects, taxa names are suffixed with the phyloseq object name to make them distinguishable.

Value

A tibble with the combined ALDEx2 results from all phyloseq objects. Each row corresponds to one taxon in one phyloseq object, with columns from ALDEx2 output plus `taxon` (from rownames) and `name` (identifying the source phyloseq object).

See Also

`MiscMetabar::aldex_pq()`, `ancombc_lpq()`, `multipatt_lpq()`

Examples

```
## Not run:
lpq <- list_phyloseq(
  list(
    fungi = data_fungi,
    fungi_clust = postcluster_pq(data_fungi)
  ),
  same_bioinfo_pipeline = FALSE
)

results <- aldex_lpq(lpq,
  bifactor = "Height",
  modalities = c("Low", "High")
)
results

ALDEx2::aldex.plot(filter(results, name == "fungi"), type = "volcano")
ALDEx2::aldex.plot(filter(results, name == "fungi_clust"), type = "volcano")

ggplot(results, aes(y = taxon, x = effect, col = wi.eBH)) +
  geom_point()

## End(Not run)
```

`ancombc_lpq` *ANCOM-BC analysis on each phyloseq object in a list_phyloseq*

Description

Performs ANCOM-BC differential abundance analysis using `MiscMetabar::ancombc_pq()` on each phyloseq object in a list_phyloseq and returns a combined result table.

Usage

```
ancombc_lpq(x, fact, levels_fact = NULL, tax_level = NULL, verbose = TRUE, ...)
```

Arguments

`x`	(required) A list_phyloseq object.
`fact`	(character, required) The name of a column in `sample_data` to use as the grouping factor. Must be present in **all** phyloseq objects.
`levels_fact`	(character vector, default NULL) Levels of the factor to include. If NULL, all levels are used.
`tax_level`	(character, default "Class") Taxonomic level for agglomeration before analysis.
`verbose`	(logical, default TRUE) If TRUE, print progress messages.
`...`	Additional arguments passed to `MiscMetabar::ancombc_pq()`.

Details

This function requires that the list_phyloseq type is NOT `SEPARATE_ANALYSIS`, as the factor must be common across all phyloseq objects.

When no common taxa exist across the phyloseq objects, taxa names are suffixed with the phyloseq object name to make them distinguishable.

Value

A tibble with the combined ANCOM-BC results from all phyloseq objects. Contains the `$res` data frame from each ANCOM-BC run, with an additional `name` column identifying the source phyloseq object.

See Also

`MiscMetabar::ancombc_pq()`, `aldex_lpq()`, `multipatt_lpq()`

Examples

```
data_fungi_high <- multiply_counts_pq(data_fungi, "Height", "High", 2)

lpq <- list_phyloseq(
  list(
    fungi = data_fungi,
    fungi_height = data_fungi_high
  )
)

results <- ancombc_lpq(lpq, fact = "Height")
results

results |>
  filter(diff_HeightLow) |>
  ggplot(aes(y = taxon, x = lfc_HeightLow, color = name, shape = q_HeightLow < 0.1)) +
  geom_point(size = 4) +
  geom_vline(xintercept = 0) +
  theme_minimal()

results_Genus <- ancombc_lpq(lpq, fact = "Height", tax_level = "Genus")

results_Genus |>
  filter(diff_HeightLow) |>
  ggplot(aes(y = taxon, x = lfc_HeightLow, color = name, shape = q_HeightLow < 0.1)) +
  geom_point(size = 4) +
  geom_vline(xintercept = 0) +
  theme_minimal()
```

apply_to_lpq *Apply a function to all phyloseq objects in a list_phyloseq*

Description

Applies a function to each phyloseq object in a list_phyloseq and returns a new list_phyloseq with the transformed objects. This is useful for applying cleaning, filtering, or transformation functions uniformly across all phyloseq objects.

Usage

```
apply_to_lpq(x, .f, ..., verbose = TRUE)
```

Arguments

x	(required) A list_phyloseq object.
.f	(function, required) A function to apply to each phyloseq object. The function must take a phyloseq object as its first argument and return a phyloseq object.
...	Additional arguments passed to .f.
verbose	(logical, default TRUE) If TRUE, print information about the transformation applied to each phyloseq object.

Details

Common functions to apply include:

- MiscMetabar::clean_pq(): Remove empty samples and taxa
- phyloseq::taxa_as_rows(): Ensure taxa are rows in otu_table
- phyloseq::rarefy_even_depth(): Rarefy to even depth
- phyloseq::transform_sample_counts(): Transform counts (e.g., relative abundance)
- phyloseq::subset_taxa(): Filter taxa based on criteria
- phyloseq::subset_samples(): Filter samples based on criteria

Value

A new list_phyloseq object with the transformed phyloseq objects. The comparison parameters (same_primer_seq_tech, same_bioinfo_pipeline) are preserved from the original object.

Author(s)

Adrien Taudière

See Also

filter_common_lpq(), update_list_phyloseq()

Examples

```
lpq <- list_phyloseq(list(fungi = data_fungi, fungi_mini = data_fungi_mini))

# Apply clean_pq to all phyloseq objects
lpq_clean <- apply_to_lpq(lpq, MiscMetabar::clean_pq)

# Apply taxa_as_rows
lpq_rows <- apply_to_lpq(lpq, MiscMetabar::taxa_as_rows)

# Apply rarefy_even_depth with a specific rngseed
lpq_rar <- apply_to_lpq(lpq, rarefy_even_depth, rngseed = 21)

lpq_rar

# Transform to relative abundance
lpq_rel <- apply_to_lpq(
  lpq,
  phyloseq::transform_sample_counts,
  function(x) x / sum(x)
)

# Chain multiple transformations
lpq_processed <- lpq |>
  apply_to_lpq(MiscMetabar::clean_pq) |>
  apply_to_lpq(MiscMetabar::taxa_as_rows)
```

---

bubbles_pq *Bubble plot of phyloseq object with observablehq*

Description

Creates an interactive bubble plot visualization of taxa abundances from a phyloseq object using an Observable notebook.

Usage

```
bubbles_pq(
  physeq,
  rank_label = "Taxa",
  rank_color = "Family",
  categorical_scheme = "d3.schemeCategory10",
  label_color = "grey10",
  value_color = "grey20",
  label_size = 8,
  log1ptransform = FALSE,
  min_nb_seq = 0,
  randomize = FALSE,
  seed = 32,
```

```
  width = 600,
  include = c("TitleCell", "key", "chart"),
  notebook = "https://observablehq.com/d/d755af3197af2320",
  return_dataframe = FALSE,
  title = "",
  title_size = "22px"
)
```

Arguments

physeq	(required) A phyloseq-class object obtained using the phyloseq package.
rank_label	(character, default "Taxa") The name of the column in the @tax_table slot to label the points. If set to "Taxa", the taxa name is used.
rank_color	(character, default "Family") The name of the column in the @tax_table slot to color the points.
categorical_scheme	(character, default "d3.schemeCategory10") A color scheme from d3js.
label_color	(character, default "grey10") Color for the label text.
value_color	(character, default "grey20") Color for the value text.
label_size	(integer, default 8) Font size for the label.
log1ptransform	(logical, default FALSE) If TRUE, the number of sequences is log1p transformed.
min_nb_seq	(integer, default 0) Minimum number of sequences to filter out taxa with low abundance.
randomize	(logical, default FALSE) If TRUE, shuffles the order of the taxa.
seed	(integer, default 32) Seed for the randomization.
width	(integer, default 600) The notebook width for visualization in pixels.
include	(character vector, default c("TitleCell", "key", "chart")) You can modify include to remove the title and/or the legend, for example using include = c("chart").
notebook	(character, default "https://observablehq.com/d/d755af3197af2320") You can change the notebook url if you know what you are doing.
return_dataframe	(logical, default FALSE) If TRUE, the plot is not returned, but the resulting dataframe to plot is returned.
title	(character, default "") Title of the plot.
title_size	(character, default "22px") Font size for the title.

Value

A htmlwidget

Author(s)

Adrien Taudière

Examples

```
bubbles_pq(physeq = data_fungi, rank_color = "Class", randomize = TRUE)
bubbles_pq(physeq = data_fungi, min_nb_seq = 10000, categorical_scheme = "d3.schemePastel1")
bubbles_pq(physeq = data_fungi, min_nb_seq = 1000, categorical_scheme = "d3.schemeTableau10", label_color = "purpl
bubbles_pq(physeq = data_fungi, rank_color = "Order", include = c("chart"), log1ptransform = TRUE)
bubbles_pq(physeq = data_fungi, rank_label = "Genus", rank_color = "Class", include = c("chart"), randomize = TRUE,
bubbles_pq(physeq = as_binary_otu_table(data_fungi), min_nb_seq = 20, title = "Nb of samples per OTU. <br> Only OTU
```

`compare_refseq` *Compare refseq slots between two phyloseq objects*

Description

Performs a detailed comparison of the reference sequences (`refseq` slot) between two phyloseq objects. Identifies shared and unique ASVs/OTUs both by taxa name and by actual DNA sequence content.

This is useful to detect subtle differences between two phyloseq objects that may share the same samples but differ in their ASV/OTU composition, e.g. after different bioinformatics pipelines or filtering steps.

When unique (non-shared) sequences exist, the mean nearest-neighbor k-mer distance from each unique sequence to the other set is computed. This gives a sense of how different the unmatched sequences are from their closest counterpart in the other object.

Usage

```
compare_refseq(
  physeq1,
  physeq2 = NULL,
  name1 = NULL,
  name2 = NULL,
  k = 5,
  max_seqs = 500,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

`physeq1`	(phyloseq or list_phyloseq, required) First phyloseq object, which must have a `refseq` slot. Alternatively, a list_phyloseq object; in that case the first two phyloseq objects are used (with their names) and `physeq2` is ignored.
`physeq2`	(phyloseq, default NULL) Second phyloseq object. Must have a `refseq` slot. Ignored when `physeq1` is a list_phyloseq.
`name1`	(character, default NULL) Label for the first phyloseq object in the output. If NULL, inferred from the list_phyloseq names or defaults to `"physeq1"`.

name2	(character, default NULL) Label for the second phyloseq object in the output. If NULL, inferred from the list_phyloseq names or defaults to "physeq2".
k	(integer, default 5) k-mer size for nearest-neighbor distance computation on unique sequences.
max_seqs	(integer, default 500) Maximum number of unique sequences per object to use for distance computation. If there are more, a random sample is drawn. Set to Inf to use all (may be slow).
seed	(integer, default NULL) Random seed for reproducible sampling when max_seqs is exceeded.
verbose	(logical, default TRUE) If TRUE, print a summary of the comparison.

Value

A list of class "compare_refseq" with components:

n_taxa Named integer vector with the number of taxa in each object.

shared_names Character vector of taxa names present in both objects.

unique_names_1 Taxa names only in physeq1.

unique_names_2 Taxa names only in physeq2.

shared_seqs Character vector of unique DNA sequences found in both objects (regardless of taxa name).

unique_seqs_1 DNA sequences only in physeq1.

unique_seqs_2 DNA sequences only in physeq2.

same_name_diff_seq A data frame of taxa with the same name but different sequences across objects. Columns: taxa_name, seq_physeq1, seq_physeq2.

same_seq_diff_name A data frame of sequences shared between objects under different names. Columns: sequence, name_physeq1, name_physeq2.

mean_nn_kmer_dist_1 Mean nearest-neighbor k-mer distance from sequences unique to physeq1 to the closest sequence in physeq2. NA if no unique sequences.

mean_nn_kmer_dist_2 Mean nearest-neighbor k-mer distance from sequences unique to physeq2 to the closest sequence in physeq1. NA if no unique sequences.

Author(s)

Adrien Taudiere

Examples

```
# Compare a phyloseq object with itself (all shared)
res <- compare_refseq(data_fungi_mini, data_fungi_mini)
res

# Compare with a subset
sub <- prune_taxa(taxa_names(data_fungi_mini)[1:20], data_fungi_mini)
res2 <- compare_refseq(data_fungi_mini, sub,
  name1 = "full", name2 = "subset"
```

```
)
res2

# From a list_phyloseq (uses the first two objects and their names)
lpq <- list_phyloseq(list(full = data_fungi_mini, subset = sub))
res3 <- compare_refseq(lpq)
res3
```

`div_pq` *Diversity indices per sample, optionally grouped by a modality*

Description

Computes alpha-diversity indices (via `vegan::diversity()`) and/or Hill numbers / Rényi entropy (via `vegan::renyi()`) for each sample in a phyloseq object. When `modality` is supplied the computation is done separately for each level of that grouping variable and the level label is appended as an extra column. `NA` values in the modality column are kept as a distinct group so that samples with missing metadata are never silently dropped. When `modality = NULL` all samples are analysed together and no grouping column is added to the result.

Usage

```
div_pq(
  physeq,
  modality = NULL,
  indices = "shannon",
  scales = NULL,
  hill = TRUE,
  aggregate = FALSE,
  funs = list(mean = mean, sd = sd)
)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object.
`modality`	(character or NULL, default NULL) Name of a column in `sample_data(physeq)` used to split samples into groups. `NA` values are treated as a separate group. When `NULL`, indices are computed for the whole dataset without grouping.
`indices`	(character, default `"shannon"`) One or more index names accepted by `vegan::diversity()` (e.g. `"shannon"`, `"simpson"`, `"invsimpson"`). Set to `NULL` to skip classical diversity indices.
`scales`	(numeric or NULL, default NULL) Scale values passed to `vegan::renyi()`. When `NULL`, Hill / Rényi computation is skipped.
`hill`	(logical, default TRUE) If `TRUE`, return Hill numbers; if `FALSE`, return Rényi entropy. Passed to `vegan::renyi()`. Only relevant when `scales` is not `NULL`.
`aggregate`	(logical, default FALSE) If `TRUE` and `modality` is not `NULL`, summarise per-group results using `funs`.

`funs` (named list, default `list(mean = mean, sd = sd)`) Summary functions applied when `aggregate = TRUE`. Passed to `dplyr::across()`.

Details

Value

A data frame with one row per sample. Columns correspond to the requested `indices` and/or Hill / Rényi scales. When `modality` is supplied, an additional column named after `modality` identifies the group (possibly `NA`). When `aggregate = TRUE` rows are collapsed to one per group with summary statistics.

Author(s)

Adrien Taudière

See Also

`vegan::diversity()`, `vegan::renyi()`

Examples

```
div_pq(data_fungi_mini, indices = c("shannon", "simpson"))

div_pq(
  data_fungi_mini,
  modality = "Height",
  indices = c("shannon", "simpson"),
  scales = c(0, 1, 2),
  hill = TRUE
)

div_pq(
  data_fungi_mini,
  modality = "Height",
  indices = "shannon",
  aggregate = TRUE
)
```

---

estim_cor_lpq *Estimation statistics for numeric correlation on a list_phyloseq*

Description

Applies `estim_cor_pq()` to each phyloseq object in a list_phyloseq and combines the results.

Usage

```
estim_cor_lpq(x, variable, ..., verbose = TRUE)
```

Arguments

x	(required) A list_phyloseq object.
variable	(character, required) The name of a numeric column in sample_data. Must be present in **all** phyloseq objects.
...	Additional arguments passed to estim_cor_pq().
verbose	(logical, default TRUE) If TRUE, print progress messages.

Details

Value

A list of class "estim_cor_lpq_result" with components:

results A named list of estim_cor_pq_result objects (one per phyloseq)

correlations A tibble combining all correlations with an additional name column

regressions A tibble combining all regressions with an additional name column

See Also

estim_cor_pq(), estim_diff_lpq()

Examples

```
## Not run:
lpq <- list_phyloseq(
  list(
    fungi = data_fungi,
    fungi_clust = postcluster_pq(data_fungi)
  ),
  same_bioinfo_pipeline = FALSE
)

# Assuming a numeric variable exists in sample_data
results <- estim_cor_lpq(lpq, variable = "lib_size")
results$correlations

## End(Not run)
```

estim_cor_pq *Estimation statistics for numeric variable correlation on a phyloseq object*

Description

Computes diversity metrics (Hill numbers by default) per sample and assesses their relationship with a numeric variable using bootstrap confidence intervals for correlation coefficients and regression slopes.

Usage

```
estim_cor_pq(
  physeq,
  variable,
  hill_scales = c(0, 1, 2),
  custom_fn = NULL,
  method = "pearson",
  resamples = 5000,
  ci = 95,
  na_remove = TRUE
)
```

Arguments

physeq	(phyloseq, required) A phyloseq object.
variable	(character, required) The name of a numeric column in sample_data.
hill_scales	(numeric vector, default c(0, 1, 2)) The q values for Hill number computation.
custom_fn	(function, default NULL) An optional custom diversity function (see estim_diff_pq() for details).
method	(character, default "pearson") Correlation method. One of "pearson", "spearman", "kendall".
resamples	(integer, default 5000) Number of bootstrap resamples.
ci	(numeric, default 95) Confidence interval level (0-100).
na_remove	(logical, default TRUE) If TRUE, samples with NA in variable are removed.

Details

Value

A list of class "estim_cor_pq_result" with components:

data The diversity data.frame used for analysis

correlations A tibble with columns: `metric`, `estimate`, `ci_lower`, `ci_upper`, `method`, `pvalue`

regressions A tibble with columns: `metric`, `intercept`, `slope`, `slope_ci_lower`, `slope_ci_upper`

plots A named list of ggplot2 scatter plots with regression line and bootstrap CI ribbon (one per metric)

See Also

`estim_diff_pq()`, `estim_cor_lpq()`

Examples

```
## Not run:
library(phyloseq)
data("data_fungi", package = "MiscMetabar")

# Add a numeric variable for demonstration
sam <- sample_data(data_fungi)
sam$lib_size <- sample_sums(data_fungi)
sample_data(data_fungi) <- sam

res <- estim_cor_pq(data_fungi, variable = "lib_size")
res
res$plots$Hill_0
res$correlations

## End(Not run)
```

---

estim_diff_lpq *Estimation statistics for categorical comparisons on a list_phyloseq*

Description

Applies `estim_diff_pq()` to each phyloseq object in a list_phyloseq and combines the results. This allows comparing estimation statistics across different bioinformatic pipelines or parameter settings.

Usage

```
estim_diff_lpq(x, fact, ..., verbose = TRUE)
```

Arguments

`x`	(required) A list_phyloseq object.
`fact`	(character, required) The name of a categorical column in `sample_data` to use as the grouping factor. Must be present in **all** phyloseq objects.
`...`	Additional arguments passed to `estim_diff_pq()`.
`verbose`	(logical, default TRUE) If TRUE, print progress messages.

Details

Value

A list of class "estim_diff_lpq_result" with components:

results A named list of estim_diff_pq_result objects (one per phyloseq)

summary A tibble combining all summaries with an additional name column identifying the source phyloseq

See Also

estim_diff_pq(), estim_cor_lpq(), adonis_lpq()

Examples

```
lpq <- list_phyloseq(
  list(
    fungi = data_fungi,
    fungi_clust = postcluster_pq(data_fungi),
    fungi_rarefy = rarefy_even_depth(data_fungi),
    fungi_with_less_otu_in_High =  multiply_counts_pq(data_fungi,
      fact = "Height", prop=0.8,
      conditions = "High",
      multipliers = 0)
  ),
  same_bioinfo_pipeline = FALSE
)

results <- estim_diff_lpq(lpq, fact = "Height")
results$summary

# Plot results for two phyloseq objects
 ggplot(results$summary, aes(x = metric, y = effect_size, color=name)) +
   facet_wrap(~comparison) +
   geom_point(position=position_dodge(width=0.5)) +
  geom_errorbar(aes(ymin = ci_lower, ymax = ci_upper), width = 0.2, position=position_dodge(width=0.5 ))
```

---

estim_diff_pq *Estimation statistics for categorical comparisons on a phyloseq object*

Description

Computes diversity metrics (Hill numbers by default) per sample and compares them across groups defined by a categorical variable using estimation statistics (effect sizes + bootstrap confidence intervals) via the **dabestr** package.

This approach replaces traditional p-value-based hypothesis testing with Gardner-Altman or Cumming estimation plots, following the estimation statistics framework (Ho et al. 2019).

Usage

```
estim_diff_pq(
  physeq,
  fact,
  hill_scales = c(0, 1, 2),
  custom_fn = NULL,
  effect_type = "cohens_d",
  idx = NULL,
  ci = 95,
  resamples = 5000,
  na_remove = TRUE,
  ...
)
```

Arguments

physeq	(phyloseq, required) A phyloseq object.
fact	(character, required) The name of a categorical column in sample_data to use as the grouping factor.
hill_scales	(numeric vector, default c(0, 1, 2)) The q values for Hill number computation: 0 = richness, 1 = Shannon exponential, 2 = inverse Simpson.
custom_fn	(function, default NULL) An optional custom diversity function. Must take a phyloseq object and return a named numeric vector (names = sample names) or a data.frame with one row per sample. If provided, hill_scales is ignored.
effect_type	(character, default "mean_diff") The type of effect size to compute. One of: "mean_diff", "median_diff", "cohens_d", "hedges_g", "cliffs_delta".
idx	(list or character vector, default NULL) The group ordering for comparisons. If NULL, uses factor levels with first level as control. For 2 groups: c("Control", "Treatment"). For 3+ groups: list(c("Ctrl", "T1", "T2")).
ci	(numeric, default 95) Confidence interval level (0-100).
resamples	(integer, default 5000) Number of bootstrap resamples.
na_remove	(logical, default TRUE) If TRUE, samples with NA in fact are removed before analysis.
...	Additional arguments passed to dabestr plotting functions.

Details

The function uses the **dabestr** package to produce estimation plots. For two groups, Gardner-Altman plots are produced (float_contrast = TRUE). For three or more groups, Cumming plots are used (float_contrast = FALSE).

Value

A list of class "estim_diff_pq_result" with components:

data The diversity data.frame used for analysis

dabest_objects A named list of dabestr objects (one per metric)

plots A named list of dabestr plots (one per metric)

summary A tibble summarizing all effect sizes and CIs with columns: `metric`, `comparison`, `effect_size`, `ci_lower`, `ci_upper`, `pvalue_permtest`, `pvalue_welch`, `pvalue_mann_whitney`

effect_type The effect size type used

References

Ho, J., Tumkaya, T., Aryal, S., Choi, H., & Claridge-Chang, A. (2019). Moving beyond P values: data analysis with estimation graphics. *Nature Methods*, 16(7), 565-566.

See Also

`estim_cor_pq()`, `estim_diff_lpq()`, `adonis_lpq()`

Examples

```
library(phyloseq)
data("data_fungi", package = "MiscMetabar")

pq <- subset_samples(data_fungi, !is.na(Height))
pq <- clean_pq(pq)

res <- estim_diff_pq(pq, fact = "Height")
res
res$plots$Hill_0
res$summary
```

---

factor_formatter *Format factor columns with funky colored backgrounds*

Description

Format factor columns with funky colored backgrounds

Usage

```
factor_formatter(x)
```

Arguments

x	A factor or character vector

Value

A formattable object with colored backgrounds

`filter_common_lpq` *Filter phyloseq objects to keep only shared samples and/or taxa*

Description

Filters each phyloseq object in a list_phyloseq to retain only the samples and/or taxa that are common across all objects. This is useful for making direct comparisons on a common basis.

Usage

```
filter_common_lpq(
  x,
  filter_samples = TRUE,
  filter_taxa = FALSE,
  clean = TRUE,
  verbose = TRUE
)
```

Arguments

`x`	(required) A list_phyloseq object.
`filter_samples`	(logical, default TRUE) If TRUE, filter to keep only samples present in all phyloseq objects.
`filter_taxa`	(logical, default FALSE) If TRUE, filter to keep only taxa present in all phyloseq objects.
`clean`	(logical, default TRUE) If TRUE, apply `MiscMetabar::clean_pq()` after filtering to remove empty samples/taxa.
`verbose`	(logical, default TRUE) If TRUE, print information about the filtering process.

Details

This function is particularly useful for:

- **NESTED_ROBUSTNESS** comparisons: filter to common samples when comparing original vs rarefied data
- **EXPLORATION** comparisons: filter to common samples/taxa when comparing different sample groups
- Any comparison where you need to ensure all phyloseq objects contain the same samples or taxa

Value

A new list_phyloseq object with filtered phyloseq objects

Examples

```
lpq <- list_phyloseq(list(
  original = data_fungi,
  mini = data_fungi_mini,
  rarefied = rarefy_even_depth(data_fungi)
))

# Filter to keep only common samples (useful for nested comparisons)
lpq_filtered <- filter_common_lpq(lpq, filter_samples = TRUE, verbose = FALSE)

# Filter to keep only common taxa
lpq_filtered <- filter_common_lpq(lpq, filter_samples = FALSE, filter_taxa = TRUE)

# Filter both samples and taxa
lpq_filtered <- filter_common_lpq(lpq, filter_samples = TRUE, filter_taxa = TRUE)
```

---

formattable_lpq *Formattable visualization for list_phyloseq summary*

Description

Create a visualization table to display the summary_table from a list_phyloseq object using the formattable package with colored bars.

Display the summary_table from a list_phyloseq object as a formattable table with colored bars for numeric columns and colored indicators for logical columns.

Usage

```
formattable_lpq(
  x,
 columns = c("name", "n_samples", "n_taxa", "n_sequences", "mean_seq_per_sample",
  "mean_seq_per_taxon", "has_sam_data", "has_tax_table", "has_refseq", "has_phy_tree"),
  bar_colors = NULL,
  round_digits = 1,
  void_style = FALSE,
  log10_transform = TRUE,
  ...
)
```

Arguments

x	(required) A list_phyloseq object.
columns	(character vector, default selection of key columns) Character vector of column names to display. If NULL, displays a curated selection of columns.
bar_colors	(named list, default NULL) Named list of colors for numeric columns with bars. Names should match column names. Default colors are provided for common columns.

round_digits	(integer, default 1) Number of decimal places for rounding numeric columns.
void_style	(logical, default FALSE) If TRUE, returns a formattable without any custom styling.
log10_transform	(logical, default TRUE) If TRUE, applies log10 transformation to numeric columns with a range greater than 1000.
...	Additional arguments passed to formattable::formattable().

Details

This function is inspired by MiscMetabar::formattable_pq(). Numeric columns are displayed with proportional colored bars. Logical columns (has_sam_data, has_tax_table, etc.) are displayed with checkmarks or X marks with colored backgrounds.

Value

A formattable object

Examples

```
## Not run:
lpq <- list_phyloseq(list(data1 = data_fungi, data2 = data_fungi_mini))
formattable_lpq(lpq)

# Custom columns
formattable_lpq(lpq,
  columns = c("name", "n_samples", "n_taxa", "n_sequences")
)

# Custom colors
formattable_lpq(lpq, bar_colors = list(
  n_samples = "steelblue",
  n_taxa = "darkgreen",
  n_sequences = "purple"
))

## End(Not run)
```

---

formattable_lpq_full *Extended formattable for list_phyloseq with comparison info*

Description

Create an extended formattable table that also displays comparison characteristics from a list_phyloseq object.

Usage

```
formattable_lpq_full(x, show_summary = TRUE, show_comparison = TRUE, ...)
```

Arguments

x	(required) A list_phyloseq object.
show_summary	(logical, default TRUE) If TRUE, show the summary table.
show_comparison	(logical, default TRUE) If TRUE, show comparison info.
...	Additional arguments passed to formattable_lpq().

Details

Value

A list containing formattable objects for summary and comparison, or a single formattable if only one is requested.

---

gg_aldex_plot *ggplot2 version of ALDEx2 diagnostic plots*

Description

Creates ggplot2 versions of the four ALDEx2 diagnostic plot types: MW (Bland-Altman style), MA, volcano, and volcano.var. Supports faceting by a name column, making it suitable for output from aldex_lpq().

Usage

```
gg_aldex_plot(
  x,
  type = c("MW", "MA", "volcano", "volcano.var"),
  test = c("welch", "wilcox", "effect", "both"),
  cutoff.pval = 0.05,
  cutoff.effect = 1,
  rare = 0,
  all.col = "grey30",
  called.col = "red",
  rare.col = "black",
  point.size = 1.5,
  point.alpha = 0.4
)
```

Arguments

x	(data.frame or tibble, required) ALDEx2 results, typically from ALDEx2::aldex() or aldex_lpq(). Must contain columns diff.btw, diff.win, rab.all, and effect. When test is "welch", column we.eBH is required; when "wilcox", column wi.eBH is required.
type	(character, default "MW") Plot type. One of "MW" (dispersion vs difference), "MA" (abundance vs difference), "volcano" (difference vs -log10 q-value), or "volcano.var" (dispersion vs -log10 q-value).
test	(character, default "welch") Statistical test used for significance calling. One of "welch", "wilcox", "effect", or "both" (effect + welch).
cutoff.pval	(numeric, default 0.05) q-value threshold for significance.
cutoff.effect	(numeric, default 1) Effect size threshold (must be >= 0.5 when test is "effect" or "both").
rare	(numeric, default 0) Abundance threshold below which taxa are marked as rare (only used for MW and MA plots).
all.col	(character, default "grey30") Color for non-significant points.
called.col	(character, default "red") Color for significant points.
rare.col	(character, default "black") Color for rare taxa points.
point.size	(numeric, default 1.5) Size of points.
point.alpha	(numeric, default 0.4) Alpha transparency of non-significant points.

Details

This function reimplements ALDEx2::aldex.plot() using ggplot2, providing a more customizable and composable output. The four plot types correspond to the original ALDEx2 types:

MW Bland-Altman style: within-condition dispersion (x) vs between-condition difference (y), with +/-1 effect size lines.

MA MA plot: median log2 relative abundance (x) vs median log2 difference (y).

volcano Volcano plot: median log2 difference (x) vs -log10 q-value (y).

volcano.var Variance volcano: median log2 dispersion (x) vs -log10 q-value (y).

Value

A ggplot2 object. If x contains a name column (e.g., from aldex_lpq()), the plot is faceted by name.

See Also

aldex_lpq(), ALDEx2::aldex.plot()

Examples

```
data_fungi_high <- multiply_counts_pq(data_fungi, "Height", "High",
  4,
  prop_taxa = 0.1, seed = 42
)

aldex_pq(data_fungi_high,
  bifactor = "Height",
  modalities = c("Low", "High")
) |>
  gg_aldex_plot()

lpq <- list_phyloseq(
  list(
    fungi = data_fungi,
    fungi_height = data_fungi_high
  ),
  same_bioinfo_pipeline = FALSE
)
# From aldex_lpq (faceted by name)
lpq_res <- aldex_lpq(lpq,
  bifactor = "Height",
  modalities = c("Low", "High")
)
gg_aldex_plot(lpq_res, type = "MA")
gg_aldex_plot(lpq_res, type = "MW", test = "wilcox")

gg_aldex_plot(lpq_res, type = "volcano")
gg_aldex_plot(lpq_res, type = "volcano.var")


gingival_pq <-
  MicrobiomeBenchmarkData::getBenchmarkData("HMP_2012_16S_gingival_V35_subset",
    dryrun = FALSE
  )[[1]] |>
  mia::convertToPhyloseq()

aldex_res <- aldex_pq(gingival_pq,
  bifactor = "body_subsite",
  modalities = c("supragingival_plaque", "subgingival_plaque")
)

gg_aldex_plot(aldex_res, type = "volcano")
```

---

gg_bubbles_pq *Circle-packed bubble plot of a phyloseq object using ggplot2*

Description

Creates a static circle-packed bubble plot of taxa abundances from a phyloseq object using ggplot2. Circles can be packed in a circular layout (tight, default) or a square layout. Optionally facets the plot by a sample data variable, producing one bubble chart per level.

When a list_phyloseq is passed as `physeq`, it is first merged into a single phyloseq object using `merge_lpq()` (each original phyloseq becomes one sample) and the plot is automatically faceted by `source_name`.

When `diff_contour = TRUE` together with `facet_by` (or a list_phyloseq input), all pairwise comparisons between facet levels are shown side by side using **patchwork**. For each pair (A vs B), taxa unique to A are highlighted with A's color and taxa unique to B with B's color. Shared taxa receive a transparent contour. This makes it easy to spot which taxa are exclusive to each group in every pairwise comparison.

Usage

```
gg_bubbles_pq(
  physeq,
  rank_label = "Taxa",
  rank_color = "Family",
  rank_contour = NULL,
  layout = c("circle", "square"),
  facet_by = NULL,
  log1ptransform = FALSE,
  min_nb_seq = 0,
  label_size = 2,
  label_color = "grey10",
  show_labels = TRUE,
  border_color = "white",
  border_width = 0.5,
  alpha = 0.8,
  npoints = 50,
  ncol_facet = NULL,
  return_dataframe = FALSE,
  diff_contour = FALSE,
  diff_contour_colors = c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3"),
  diff_border_width = 1.5,
  show_title = TRUE,
  match_by = c("refseq", "names")
)
```

Arguments

`physeq`	(required) A `phyloseq-class` object obtained using the `phyloseq` package.
`rank_label`	(character, default "Taxa") The name of the column in the `@tax_table` slot to label the circles. If set to "Taxa", the taxa names are used.
`rank_color`	(character, default "Family") The name of the column in the `@tax_table` slot to color the circles.

`rank_contour` (character, default NULL) The name of a column in the `@tax_table` slot to color the circle borders (contours). When NULL, the fixed `border_color` is used for all borders. Ignored when `diff_contour = TRUE`.

`layout` (character, default "circle") The packing layout. `"circle"` produces a tight circular packing. `"square"` constrains circles inside a square boundary, with large circles placed centrally.

`facet_by` (character, default NULL) A column name from `@sam_data` to facet the plot. When set, one bubble chart is produced per level of the variable, with taxa abundances computed within each level. When `physeq` is a list_phyloseq, this is automatically set to `"source_name"`.

`log1ptransform` (logical, default FALSE) If TRUE, the number of sequences is log1p transformed before computing circle sizes.

`min_nb_seq` (integer, default 0) Minimum number of sequences to filter out taxa with low abundance.

`label_size` (numeric, default 2) Font size for the labels inside circles.

`label_color` (character, default "grey10") Color for the label text.

`show_labels` (logical, default TRUE) If TRUE, labels are displayed inside circles. Only circles large enough to fit text are labeled.

`border_color` (character, default "white") Color for circle borders.

`border_width` (numeric, default 0.5) Width of circle borders.

`alpha` (numeric, default 0.8) Transparency of circle fill.

`npoints` (integer, default 50) Number of vertices used to approximate each circle polygon. Higher values produce smoother circles.

`ncol_facet` (integer, default NULL) Number of columns for facet layout. Passed to `ggplot2::facet_wrap()`.

`return_dataframe` (logical, default FALSE) If TRUE, the plot is not returned, but the resulting dataframe to plot is returned. Ignored when `diff_contour = TRUE`.

`diff_contour` (logical, default FALSE) If TRUE and `facet_by` is set (or `physeq` is a list_phyloseq), produces pairwise comparison panels for all pairs of facet levels using **patchwork**. For each pair, taxa unique to each side are highlighted with a distinct contour color from `diff_contour_colors`. Shared taxa get a transparent contour. When TRUE, `rank_contour` is ignored.

`diff_contour_colors` (character vector, default `c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3")`) Border colors for taxa unique to each facet level in `diff_contour` mode. Recycled if shorter than the number of facet levels. Each level gets a distinct color so unique taxa from different groups are visually distinguishable.

`diff_border_width` (numeric, default 1.5) Border width used in `diff_contour` mode.

`show_title` (logical, default TRUE) If TRUE, adds an informative title describing what the fill color, contour color, circle size, and labels represent.

`match_by` (character, default `"refseq"`) How to match taxa when `physeq` is a list_phyloseq. Passed to `merge_lpq()`. One of `"refseq"` (match by reference sequences) or `"names"` (match by taxa names).

Value

A ggplot2 object, a patchwork object (when `diff_contour = TRUE`), or a data.frame if `return_dataframe = TRUE`.

Author(s)

Adrien Taudière

Examples

```
gg_bubbles_pq(physeq = data_fungi_mini, rank_color = "Class")
gg_bubbles_pq(
  physeq = data_fungi_mini, rank_color = "Class",
  rank_contour = "Order"
)
gg_bubbles_pq(
  physeq = data_fungi_mini, rank_color = "Class",
  layout = "square"
)

# Faceted by sample variable
gg_bubbles_pq(
  physeq = data_fungi, rank_color = "Order",
  facet_by = "Height", min_nb_seq = 100
) + no_legend()

# Pairwise diff_contour on a faceted phyloseq
gg_bubbles_pq(
  physeq = data_fungi, rank_color = "Order",
  facet_by = "Height", min_nb_seq = 100,
  diff_contour = TRUE, show_labels = FALSE
) & no_legend()

# list_phyloseq: automatically merged and faceted
mini2 <- subset_taxa_pq(data_fungi_mini, taxa_sums(data_fungi_mini) < 10000)
lpq <- list_phyloseq(
  list(full = data_fungi, mini = data_fungi_mini, mini2 = mini2),
)
gg_bubbles_pq(lpq, rank_color = "Class")

# list_phyloseq with diff_contour: pairwise panels
gg_bubbles_pq(lpq, rank_color = "Class", diff_contour = TRUE,
  show_labels = FALSE, diff_border_width = 1) & no_legend()
```

`gg_hill_lpq` *Scatter plots of Hill diversity across pairs of phyloseq objects*

Description

For each pair of phyloseq objects in a list_phyloseq, computes Hill numbers (richness, Shannon exponential, inverse Simpson) per sample and draws scatter plots comparing the diversity of the **same samples** across objects. This is a visual diagnostic for REPRODUCIBILITY, ROBUSTNESS, and REPLICABILITY comparisons.

Each point represents one sample. A 1:1 reference line and optional regression line help assess how similar (or different) the diversity estimates are between two pipelines, runs, or primers.

Usage

```
gg_hill_lpq(
  x,
  hill_scales = c(0, 1, 2),
  pairs = NULL,
  add_1to1 = TRUE,
  add_smooth = TRUE,
  cor_method = "pearson",
  point_color = "black",
  point_alpha = 0.6,
  point_size = 2,
  smooth_color = "steelblue",
  verbose = TRUE
)
```

Arguments

x	(required) A list_phyloseq object.
hill_scales	(numeric vector, default c(0, 1, 2)) Hill number orders to compute: 0 = richness, 1 = Shannon exponential, 2 = inverse Simpson.
pairs	(list of integer pairs or NULL, default NULL) Which pairs of phyloseq objects to compare. Each element must be a length-2 integer vector of indices. If NULL, all pairwise combinations are used.
add_1to1	(logical, default TRUE) If TRUE, adds a dashed 1:1 identity line to each panel.
add_smooth	(logical, default TRUE) If TRUE, adds a linear regression line with confidence interval via ggplot2::geom_smooth().
cor_method	(character, default "pearson") Correlation method for the annotation label. One of "pearson", "spearman", "kendall".
point_color	(character, default "black") Color for scatter points.
point_alpha	(numeric, default 0.6) Transparency for scatter points.
point_size	(numeric, default 2) Size for scatter points.
smooth_color	(character, default "steelblue") Color for the regression line and confidence ribbon.
verbose	(logical, default TRUE) If TRUE, print a message when common samples are used instead of all samples.

Details

The function works on **common samples** across all phyloseq objects. When the list_phyloseq has `same_samples = TRUE`, all samples are used. When samples differ (e.g., NESTED_ROBUSTNESS), only the intersection is kept.

Comparison type context:

REPRODUCIBILITY High correlation expected for all Hill orders.

ROBUSTNESS Moderate to high correlation is desirable; deviations indicate pipeline sensitivity.

REPLICABILITY Correlation may vary across Hill orders depending on primer or technology differences.

Value

A `ggplot2::ggplot()` object with panels arranged in a grid: rows = pairs of phyloseq objects, columns = Hill number orders.

See Also

`estim_cor_pq()`, `estim_diff_lpq()`

Examples

```
lpq <- list_phyloseq(
  list(run1 = data_fungi, run2 = data_fungi_mini),
  same_bioinfo_pipeline = FALSE
)

gg_hill_lpq(lpq)
gg_hill_lpq(lpq, hill_scales = c(0, 1))
gg_hill_lpq(lpq, add_smooth = FALSE, add_1to1 = TRUE)
```

---

gg_maaslin3_plot *Plot MaAsLin3 results*

Description

Creates ggplot2 visualizations of MaAsLin3 differential abundance results. Supports summary plots (default), volcano plots, and forest plots.

Usage

```
gg_maaslin3_plot(
  res,
  type = c("summary", "volcano", "forest"),
  model = c("abundance", "prevalence", "both"),
  metadata_filter = NULL,
```

```
  signif_threshold = 0.1,
  use_qval = TRUE,
  top_n = 25,
  show_labels = FALSE,
  point_size = 2,
  colors = NULL,
  show_annotations = TRUE,
  reference_label = NULL,
  coef_plot_vars = NULL,
  heatmap_vars = NULL,
  normalization = "TSS",
  transform = "LOG"
)
```

Arguments

`res`	(list, required) The full result object from `maaslin3_pq()`. For types other than "summary", can also be directly the `$results` data frame.
`type`	(character, default "summary") Type of plot. One of: • `"summary"`: Default MaAsLin3 summary plot with coefficients and heatmap (uses `maaslin3::maaslin_plot_results()` internally) • `"volcano"`: Effect size (coef) vs -log10(p-value) • `"forest"`: Effect sizes with confidence intervals
`model`	(character, default "abundance") Which model results to plot. One of `"abundance"`, `"prevalence"`, or `"both"`. Used for volcano and forest plots.
`metadata_filter`	(character, default NULL) Filter results to a specific metadata variable. If NULL, uses the first non-intercept variable. Used for volcano and forest plots.
`signif_threshold`	(numeric, default 0.1) Significance threshold for q-value (FDR-corrected). For summary plot, this is passed to `max_significance`. For other plots, used to color significant features.
`use_qval`	(logical, default TRUE) If TRUE, use q-values (FDR-corrected) for significance. If FALSE, use raw p-values. Used for volcano and forest.
`top_n`	(integer, default 25) For summary plot, number of top features to display (`summary_plot_first_n`). For forest plot, show only the top N features by absolute effect size.
`show_labels`	(logical, default FALSE) For volcano plots, show feature labels for significant points.
`point_size`	(numeric, default 2) Size of points in volcano plot.
`colors`	(character vector, default NULL) Colors for significant/ non-significant points. If NULL, uses c("grey60", "firebrick3").
`show_annotations`	(logical, default TRUE) If TRUE, adds annotations indicating which group has higher values on each side of the plot. For simple formulas like `~ Height`, shows "More in High" vs "More in Low".

`reference_label` (character, default NULL) Label for the reference group. If NULL, attempts to extract from the metadata column name.

`coef_plot_vars` (character vector, default NULL) For summary plot only. Variables to include in coefficient plot. If NULL, uses all non-intercept variables.

`heatmap_vars` (character vector, default NULL) For summary plot only. Variables to include in heatmap. If NULL, determined automatically.

`normalization` (character, default "TSS") Normalization method used. Only needed for summary plot. One of "TSS", "CLR", or "NONE".

`transform` (character, default "LOG") Transform method used. Only needed for summary plot. One of "LOG", "PLOG", or "NONE".

Details

#TODO VERY experimental

Summary plot (default): Uses `maaslin3::maaslin_plot_results()` to create the standard MaAsLin3 visualization with sorted per-feature coefficients plotted with standard errors, and a heatmap for additional variables. This requires the full maaslin3 result object (not just the results data frame).

Volcano plot: Shows the relationship between effect size (coefficient) and statistical significance. Points above the horizontal dashed line are significant at the specified threshold. Points are colored by significance.

Forest plot: Shows effect sizes with 95 percent confidence intervals for the top features. Features are ordered by effect size. Significant associations are highlighted.

The `coef` in MaAsLin3 abundance models represents log2 fold change: a one-unit change in the variable corresponds to a 2^coef fold change in relative abundance.

Value

A ggplot2 object.

Author(s)

Adrien Taudière

See Also

`maaslin3_pq()`, `gg_aldex_plot()`

Examples

```
## Not run:
# Run MaAsLin3
res <- maaslin3_pq(
  data_fungi,
  formula = "~ Height",
  reference = list(Height = "Low")
)
```

```
# Summary plot (default) - uses maaslin3's native visualization
gg_maaslin3_plot(res)

# Volcano plot
gg_maaslin3_plot(res, type = "volcano")

# Forest plot of top 15 features
gg_maaslin3_plot(res, type = "forest", top_n = 15)

# Plot prevalence model results
gg_maaslin3_plot(res, type = "volcano", model = "prevalence")

# Customize significance threshold
gg_maaslin3_plot(res, type = "volcano", signif_threshold = 0.05, show_labels = TRUE)

 # Complete Example with HMP2 Dataset

 library(maaslin3)
 # ============================================================
 # Convert maaslin3 default HMP2 dataset to phyloseq
 # ============================================================

 # Read the HMP2 default dataset from maaslin3 package
 taxa_table_name <- system.file("extdata", "HMP2_taxonomy.tsv", package = "maaslin3")
 taxa_table <- read.csv(taxa_table_name, sep = "\t", row.names = 1)

 metadata_name <- system.file("extdata", "HMP2_metadata.tsv", package = "maaslin3")
 metadata <- read.csv(metadata_name, sep = "\t", row.names = 1)

 # Set factor levels
 metadata$antibiotics <- factor(metadata$antibiotics, levels = c("No", "Yes"))

 # Create phyloseq object (HMP2 data has samples as rows, taxa as columns)
 otu <- otu_table(as.matrix(taxa_table), taxa_are_rows = FALSE)
 sam <- sample_data(metadata)
 species_names <- colnames(taxa_table)
 tax_df <- data.frame(
   Species = species_names,
   Genus = sapply(strsplit(species_names, "_"), \(x) x[1]),
   row.names = species_names
 )
 tax <- tax_table(as.matrix(tax_df))
 physeq_hmp2 <- phyloseq(otu, sam, tax)

 # ============================================================
 # Run MaAsLin3 analysis
 # ============================================================

 res <- maaslin3_pq(
   physeq_hmp2,
   formula = "~ antibiotics",
   reference = list(antibiotics = "No"),
   output = "output/maaslin3_hmp2",
```

```
  correction_for_sample_size = FALSE
)

# ============================================================
# Compare plots
# ============================================================

# Summary plot (NEW DEFAULT) - uses maaslin3's native visualization
gg_maaslin3_plot(res)

# Alternative plot types
gg_maaslin3_plot(res, type = "volcano")
gg_maaslin3_plot(res, type = "forest", top_n = 15)

## End(Not run)
```

`glmulti_lpq` *Automated model selection for Hill diversity on each phyloseq in a list_phyloseq*

Description

Performs automated model selection and multimodel inference using `MiscMetabar::glmutli_pq()` on each phyloseq object in a list_phyloseq. Returns a summary table with the results from all phyloseq objects.

Usage

```
glmulti_lpq(
  x,
  formula,
  fitfunction = "lm",
  hill_scales = c(0, 1, 2),
  aic_step = 2,
  confsetsize = 100,
  plotty = FALSE,
  level = 1,
  method = "h",
  crit = "aicc",
  verbose = TRUE,
  ...
)
```

Arguments

x	(required) A list_phyloseq object.

formula	(character, required) A model formula for glmulti. Variables must be present in the sample_data slot of **all** phyloseq objects. Hill numbers (Hill_0, Hill_1, Hill_2) and Abundance are automatically available.
fitfunction	(character, default "lm") The model fitting function to use. Options include "lm" for linear models or "glm" for generalized linear models.
hill_scales	(numeric vector, default c(0, 1, 2)) The q values for Hill number computation. Defaults to Hill numbers 0 (richness), 1 (Shannon exponential), and 2 (inverse Simpson).
aic_step	(numeric, default 2) The AIC score threshold for model selection. Models within this threshold from the best model are included.
confsetsize	(integer, default 100) The number of models to return in the confidence set.
plotty	(logical, default FALSE) If TRUE, display IC profile during glmulti search.
level	(integer, default 1) Model complexity level. 1 for main effects only, 2 for pairwise interactions.
method	(character, default "h") The search method for glmulti. Options: "h" (exhaustive), "g" (genetic algorithm), "l" (branch-and-bound), "d" (summary only).
crit	(character, default "aicc") Information criterion for model selection. Options include "aic", "aicc" (small-sample corrected AIC), "bic".
verbose	(logical, default TRUE) If TRUE, print progress messages.
...	Additional arguments passed to MiscMetabar::glmutli_pq().

Details

This function requires that the list_phyloseq type is NOT SEPARATE_ANALYSIS, as the formula must contain variables that are common across all phyloseq objects.

The function wraps MiscMetabar::glmutli_pq(), which itself wraps the **glmulti** package for automated model selection. For each phyloseq object, Hill diversity indices are computed and used as response variables in the model selection process.

Value

A tibble with the combined results from all phyloseq objects, containing the following columns:

name Name of the phyloseq object

variable The variable name from the model

estimates The model coefficient estimate

unconditional_interval Confidence interval from model averaging

nb_model Number of models containing this variable

importance Relative importance of the variable (sum of Akaike weights)

alpha Significance level

See Also

MiscMetabar::glmutli_pq(), MiscMetabar::hill_pq()

Examples

```
lpq <- list_phyloseq(
  list(
    fungi = data_fungi,
    fungi_clust = postcluster_pq(data_fungi)
  ),
  same_bioinfo_pipeline = FALSE
)

results <- glmulti_lpq(lpq, formula = "Hill_0 ~ Height + Time")
results

# With interactions
results_int <- glmulti_lpq(lpq, formula = "Hill_1 ~ Height * Time", level = 2)
```

---

list_phyloseq *S7 class for comparing phyloseq objects*

Description

A class to store and compare multiple phyloseq objects. It contains:

- A list of phyloseq objects
- A summary table with computed characteristics for each phyloseq
- A list of comparison characteristics between phyloseq objects

An S7 class to store and compare multiple phyloseq objects.

Usage

```
list_phyloseq(
  physeq_list,
  names = NULL,
  same_primer_seq_tech = TRUE,
  same_bioinfo_pipeline = TRUE
)
```

Arguments

physeq_list	(required) A named list of phyloseq objects.
names	(character vector, default NULL) Optional names for the phyloseq objects. If NULL and the list is unnamed, names are generated automatically.
same_primer_seq_tech	(logical, default TRUE) Whether the same primer and sequencing technology was used across all phyloseq objects. Set to FALSE when comparing different primers (e.g., ITS1 vs ITS2) or technologies (e.g., Illumina vs PacBio).

same_bioinfo_pipeline
: (logical, default TRUE) Whether the same bioinformatics pipeline was used across all phyloseq objects. Set to FALSE when comparing different clustering methods, taxonomic databases, or analysis parameters.

Details

The combination of same_primer_seq_tech and same_bioinfo_pipeline along with detected sample overlap determines the comparison type:

same_samples	same_primer_seq_tech	same_bioinfo_pipeline	Type
TRUE	TRUE	TRUE	REPRODUCIBILITY
TRUE	TRUE	FALSE	ROBUSTNESS
TRUE	FALSE	-	REPLICABILITY
nested	-	-	NESTED_ROBUSTNESS
FALSE	-	-	EXPLORATION or SEPARATE_ANALYSIS

Slots

phyloseq_list A named list of phyloseq objects

summary_table A tibble summarizing each phyloseq object

comparison A list of characteristics comparing the phyloseq objects

Types of Comparison

The list_phyloseq class determines the type of comparison based on:

- Detected characteristics (sample overlap, nested samples, shared modalities)
- User-provided parameters (same_primer_seq_tech and same_bioinfo_pipeline)

There are six main types of comparisons:

REPRODUCIBILITY Same pipeline (same_bioinfo_pipeline = TRUE), same primer/technology (same_primer_seq_tech = TRUE), same samples. Used to test **reproducibility** of results when running the exact same analysis multiple times.

ROBUSTNESS Different pipeline (same_bioinfo_pipeline = FALSE, e.g., different clustering method, different assignment database) but same primer/technology (same_primer_seq_tech = TRUE), same samples. Used to test **robustness** of conclusions to methodological choices.

NESTED_ROBUSTNESS One phyloseq object is derived from another, with samples being a subset (e.g., rarefied version created with rarefy_even_depth()). Used to test **robustness** to data processing choices like rarefaction, filtering, or subsetting. Comparisons should focus on the common (nested) samples.

REPLICABILITY Different primer and/or technology (same_primer_seq_tech = FALSE, e.g., ITS1 vs ITS2, Illumina vs PacBio), same samples. Used to test **replicability** across taxonomic groups or sequencing technologies.

EXPLORATION Different samples but with shared modalities. Useful to **explore** differences among groups of samples. Note: consider merging samples into one phyloseq object for some analyses instead.

SEPARATE_ANALYSIS Different samples with no shared modalities. **Separate analysis** of each phyloseq object is recommended as direct comparison may not be meaningful.

Examples

```
# REPRODUCIBILITY: Same samples, same pipeline (default)
lpq_repro <- list_phyloseq(list(run1 = data_fungi, run2 = data_fungi))

# ROBUSTNESS: Same samples, different pipeline
lpq_robust <- list_phyloseq(
  list(method_A = data_fungi, method_B = data_fungi),
  same_bioinfo_pipeline = FALSE
)

# REPLICABILITY: Same samples, different primer/technology
lpq_replic <- list_phyloseq(
  list(ITS1 = data_fungi, ITS2 = data_fungi),
  same_primer_seq_tech = FALSE
)
```

`maaslin3_pq` *Run MaAsLin3 differential abundance analysis on a phyloseq object*

Description

A wrapper around `maaslin3::maaslin3()` for phyloseq objects. Optionally includes the number of reads (library size) as a covariate in the model to account for differences in sequencing depth.

Usage

```
maaslin3_pq(
  physeq,
  formula,
  reference = NULL,
  correction_for_sample_size = TRUE,
  output = "res_maaslin3",
  ...
)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object.
`formula`	(character, required) A formula string for the model (e.g., `"~ Height"` or `"~ Height + Time"`).
`reference`	(named list, default NULL) Reference levels for categorical variables with more than 2 levels. Format: `list(varname = "ref_level")`. For example: `list(Height = "Low")` sets "Low" as the reference for Height. Variables not in this list will use their first level as reference.

correction_for_sample_size
: (logical, default TRUE) If TRUE, adds nb_seq (library size) to the formula to control for sequencing depth.

output
: (character, default "res_maaslin3") Output directory for MaAsLin3 results.

...
: Additional arguments passed to maaslin3::maaslin3().

Details

#' #TODO VERY experimental

MaAsLin3 requires categorical variables with more than 2 levels to have a defined reference level. This function handles this by:

1. Converting character variables to factors
2. Setting reference levels via the reference parameter
3. Using relevel() to set the specified reference as the first level

The correction_for_sample_size option adds library size as a covariate, which can help account for compositional effects and sequencing depth differences between samples.

Value

The result object from maaslin3::maaslin3(), containing:

- results: Data frame with differential abundance results
- results_ordered: Results ordered by significance
- Other MaAsLin3 output components

Author(s)

Adrien Taudière

References

Nickols WA, et al. (2024). MaAsLin 3: Refining and extending generalized multivariable linear models for meta-omic association discovery. bioRxiv.

See Also

MiscMetabar::aldex_pq(), ancombc_lpq(), gg_maaslin3_plot()

Examples

```
# Basic usage with a binary factor
data_fungi_high_low <- subset_samples(
data_fungi, Height %in% c("High", "Low")
)
res <- maaslin3_pq(data_fungi_high_low, formula = "~ Height")
res$results

# Specify reference level for multi-level factor
```

```
res <- maaslin3_pq(
  data_fungi,
  formula = "~ Height",
  reference = list(Height = "Low")
)

# Multiple variables with references
res <- maaslin3_pq(
  data_fungi,
  formula = "~ Height + Site",
  reference = list(Height = "Low", Site = "A")
)

# Without library size correction
res <- maaslin3_pq(
  data_fungi,
  formula = "~ Height",
  correction_for_sample_size = FALSE
)

# Plot results
gg_maaslin3_plot(res, type = "volcano")

# Full example with GlobalPatterns dataset
 data("GlobalPatterns")

 # Subset to two very different environments: Feces vs Soil
 gp_subset <- subset_samples(GlobalPatterns, SampleType %in% c("Feces", "Soil"))

 # Agglomerate at Phylum level for clearer results
 gp_phylum <- tax_glom(gp_subset, taxrank = "Phylum", NArm = FALSE)

 # Run MaAsLin3 with Soil as reference
 res <- maaslin3_pq(
   gp_phylum,
   formula = "~ SampleType",
   reference = list(SampleType = "Soil"),
   output = "output/maaslin3_example"  ,
  correction_for_sample_size = FALSE
 )

 # Plot results
 gg_maaslin3_plot(res, type = "volcano", signif_threshold = 0.1)
 gg_maaslin3_plot(res, type = "forest", top_n = 15)
```

---

merge_lpq *Merge a list_phyloseq into a single phyloseq object*

Description

Merges all phyloseq objects from a list_phyloseq into a single phyloseq object where each original phyloseq becomes one sample. Abundances are summed across samples within each phyloseq object.

Taxa are matched across phyloseq objects either by reference sequences (`refseq` slot, default) or by taxa names. Matching by refseq is preferred because taxa names are often inconsistent across independently built phyloseq objects (e.g., ASV_1 in one object is not the same taxon as ASV_1 in another).

Usage

```
merge_lpq(
  x,
  match_by = c("refseq", "names"),
  tax_priority = 1L,
  verbose = TRUE
)
```

Arguments

`x`	(list_phyloseq, required) A list_phyloseq object.
`match_by`	(character, default `"refseq"`) How to match taxa across phyloseq objects. One of: • `"refseq"`: match by DNA sequence in the `refseq` slot (recommended). All phyloseq objects must have a `refseq` slot. • `"names"`: match by taxa names. Use only when taxa names are consistent across objects (e.g., same pipeline, same database).
`tax_priority`	(character or integer, default 1L) Which phyloseq object's taxonomy to use when taxa are matched. Either a name from the list or an integer index. When a taxon appears in multiple objects, the taxonomy from the priority object is used; if absent there, the first available taxonomy is used.
`verbose`	(logical, default TRUE) Print information about the merge.

Value

A phyloseq object with:

`otu_table` One column per original phyloseq object (summed across its samples), one row per unique taxon.

`sample_data` One row per original phyloseq, with a column `source_name` containing the list_phyloseq names.

`tax_table` Taxonomy from the priority object (or first available).

`refseq` Present when `match_by = "refseq"`.

Author(s)

Adrien Taudière

See Also

list_phyloseq, simple_venn_pq()

Examples

```
lpq <- list_phyloseq(list(
  fungi = data_fungi_mini,
  fungi2 = data_fungi_mini
))

# Merge by refseq (default)
merged <- merge_lpq(lpq)
merged

# Merge by taxa names
merged_names <- merge_lpq(lpq, match_by = "names")
```

---

midasim_pq *Simulate microbiome data with differential abundance using MIDASim*

Description

Uses the MIDASim package to generate realistic simulated microbiome count data with known differential abundance effects. The function learns the structure (taxon correlations, sparsity patterns, library size distribution) from a template phyloseq object and generates new data with specified DA taxa.

Usage

```
midasim_pq(
  physeq,
  fact,
  condition,
  n_samples = NULL,
  prop_case = 0.5,
  n_da_taxa = 10,
  effect_size = 1,
  da_taxa_idx = NULL,
  min_prevalence = 0.1,
  mode = c("nonparametric", "parametric"),
  seed = NULL,
  verbose = TRUE
)
```

Arguments

physeq	(phyloseq, required) A phyloseq object to use as template. MIDASim will learn taxon-taxon correlations and abundance distributions from this data.
fact	(character, required) Column name in sample_data defining the binary factor for DA simulation.
condition	(character, required) The level of fact where taxa should be differentially abundant (case group).
n_samples	(integer, default NULL) Number of samples to simulate. If NULL, uses the same number as the template.
prop_case	(numeric, default 0.5) Proportion of simulated samples that belong to the case group (where DA taxa are elevated).
n_da_taxa	(integer, default 10) Number of taxa to make differentially abundant.
effect_size	(numeric, default 1) Log-fold change for DA taxa in case samples. Positive values increase abundance; negative decrease. Typical values: 0.5-2 for moderate effects.
da_taxa_idx	(integer vector, default NULL) Specific taxon indices to make DA. If NULL, randomly selects n_da_taxa from prevalent taxa.
min_prevalence	(numeric, default 0.1) Minimum prevalence for a taxon to be eligible for DA selection.
mode	(character, default "nonparametric") MIDASim mode. One of "nonparametric" (more realistic) or "parametric" (more controllable).
seed	(integer, default NULL) Random seed for reproducibility.
verbose	(logical, default FALSE) Print progress messages.

Details

#TODO : NOT WORKING with aldex2

The simulation workflow follows MIDASim's three-step process:

1. **Setup**: `MIDASim.setup()` learns the template's taxon abundance distributions, sparsity patterns, and taxon-taxon correlations.
2. **Modify**: `MIDASim.modify()` introduces DA effects by creating sample-specific relative abundances. For case samples (where `fact == condition`), selected taxa are multiplied by `exp(effect_size)`, then renormalized to sum to 1.
3. **Simulate**: `MIDASim()` generates realistic count data preserving the template's correlation structure.

This approach is superior to simple count multiplication because:

- Taxon-taxon correlations are preserved
- Sparsity patterns are realistic
- Library sizes follow realistic distributions
- The DA signal is embedded in the data generation process

Value

A phyloseq object with simulated counts. The object includes:

- Simulated OTU table with DA effects
- Tax table from template (taxa are preserved)
- New sample_data with binary factor column
- Attributes: `da_taxa_idx` (indices), `da_taxa_names` (names), `effect_size`, `condition`

Author(s)

Adrien Taudière

References

He M, et al. (2024). MIDASim: a fast and simple simulator for realistic microbiome data. Microbiome. doi:10.1186/s40168-024-01822-z

See Also

`multiply_counts_pq()`, `permute_da_pq()`

Examples

```
# Requires MIDASim package
# install.packages("MIDASim")

# Simulate data with 10 DA taxa having log-fold change of 2
sim_pq <- midasim_pq(
  data_fungi_mini,
  fact = "Height",
  condition = "High",
  n_da_taxa = 10,
  effect_size = 2,
  seed = 42
)

# Test with ALDEx2
sim_pq@sam_data$Height <-
  as.character(sim_pq@sam_data$Height)
res <- MiscMetabar::aldex_pq(
  sim_pq,
  bifactor = "Height",
  modalities = c("Low", "High")
)
gg_aldex_plot(res, type = "volcano")

# Check which taxa were set as DA
attr(sim_pq, "da_taxa_names")

res_height <- ancombc_pq(
  sim_pq,
```

```
  fact = "Height",
  levels_fact = c("Low", "High"),
  verbose = TRUE, tax_level = NULL
)

ggplot(
  res_height$res,
  aes(
    y = reorder(taxon, lfc_HeightHigh),
    x = lfc_HeightHigh,
    color = diff_HeightHigh
  )
) +
  geom_vline(xintercept = 0) +
  geom_segment(aes(
    xend = 0, y = reorder(taxon, lfc_HeightHigh),
    yend = reorder(taxon, lfc_HeightHigh)
  ), color = "darkgrey") +
  geom_point()
```

`multipatt_lpq` *Indicator species analysis on each phyloseq object in a list_phyloseq*

Description

Performs indicator species analysis using `indicspecies::multipatt()` on each phyloseq object in a list_phyloseq and returns a combined result table of significant indicator taxa.

Usage

```
multipatt_lpq(
  x,
  fact,
  p_adjust_method = "BH",
  pval = 0.05,
  control = permute::how(nperm = 999),
  verbose = TRUE,
  ...
)
```

Arguments

`x`	(required) A list_phyloseq object.
`fact`	(character, required) The name of a column in `sample_data` to use as the grouping factor. Must be present in **all** phyloseq objects.
`p_adjust_method`	(character, default "BH") The p-value adjustment method. See `stats::p.adjust()` for available methods.

pval	(numeric, default 0.05) The significance threshold for adjusted p-values.
control	(list, default permute::how(nperm = 999)) Permutation control settings for the permutation test.
verbose	(logical, default TRUE) If TRUE, print progress messages.
...	Additional arguments passed to indicspecies::multipatt().

Details

This function requires that the list_phyloseq type is NOT SEPARATE_ANALYSIS, as the factor must be common across all phyloseq objects.

Unlike MiscMetabar::multipatt_pq() which returns a plot, this function returns the underlying data as a tibble, making it easier to compare results across phyloseq objects.

When no common taxa exist across the phyloseq objects, taxa names are suffixed with the phyloseq object name to make them distinguishable.

Value

A tibble with the combined significant indicator taxa from all phyloseq objects. Contains columns from multipatt()$sign output plus taxon (taxon name), p.adj (adjusted p-value), and name (identifying the source phyloseq object). Only taxa with p.adj < pval are included.

See Also

indicspecies::multipatt(), MiscMetabar::multipatt_pq(), ancombc_lpq(), aldex_lpq()

Examples

```
## Not run:
lpq <- list_phyloseq(
  list(
    fungi = data_fungi,
    fungi_clust = postcluster_pq(data_fungi)
  ),
  same_bioinfo_pipeline = FALSE
)

results <- multipatt_lpq(lpq, fact = "Height")
results

## End(Not run)
```

`multiply_counts_pq` *Multiply OTU counts conditionally based on sample metadata*

Description

Multiplies OTU table counts by different values depending on the level of a factor column in `sample_data`. Samples whose factor value does not match any condition (including `NA` samples) are left unchanged unless explicitly targeted.

Usage

```
multiply_counts_pq(
  physeq,
  fact,
  conditions,
  multipliers,
  prop_taxa = 0.5,
  seed = NULL,
  compensate = FALSE,
  min_prevalence = 0,
  round = TRUE,
  verbose = TRUE
)
```

Arguments

`physeq`	(required) A `phyloseq-class` object obtained using the `phyloseq` package.
`fact`	(character, required) The name of a column in `sample_data(physeq)`.
`conditions`	(character vector, required) Values of `fact` to match. Use `NA` to target samples with missing values.
`multipliers`	(numeric vector, required) Multiplying factors corresponding to each element of `conditions`. Must be the same length as `conditions`.
`prop_taxa`	(numeric, default 0.5) Proportion of taxa (between 0 and 1) to which the multiplication is applied. Taxa are randomly sampled. Set to 1 to apply to all taxa.
`seed`	(integer, default NULL) Random seed for reproducible taxa sampling when `prop_taxa` < 1.
`compensate`	(logical, default FALSE) If TRUE, scale down the non-selected taxa so that total library size per sample is preserved. This creates a pure compositional shift: selected taxa gain relative abundance while the rest lose it. This mode is essential for testing differential abundance methods (ALDEx2, ANCOM-BC) which normalize for library size.
`min_prevalence`	(numeric, default 0) Minimum prevalence (proportion of matched samples with non-zero counts) for a taxon to be eligible for selection. Setting this > 0 (e.g. 0.5) ensures that only taxa actually present in the target samples are inflated, producing a stronger and more realistic DA signal.

round (logical, default TRUE) If TRUE, round the resulting counts to integers (since OTU tables typically contain integer counts).

verbose (logical, default TRUE) If TRUE, print a message with the number of modified taxa.

Details

Each sample is checked against conditions in order. When the sample's value in fact matches a condition, its counts are multiplied by the corresponding multiplier. Samples that do not match any condition are left unchanged.

When prop_taxa < 1, only a random subset of taxa is affected by the multiplication. This is useful to simulate differential abundance where only some taxa respond to a condition.

Simulating differential abundance for DA methods: DA methods like ALDEx2 and ANCOM-BC work on compositions (relative abundances). Simply multiplying counts changes library size, which these methods normalize away. Use compensate = TRUE to create a real compositional shift: selected taxa are inflated, then non- selected taxa are scaled down so total counts per sample stay constant. Combine with min_prevalence > 0 and small prop_taxa (e.g. 0.05) for a realistic DA signal on few, prevalent taxa.

Compensation limits: When the multiplier is very high and selected taxa already have high counts, the non-selected taxa may not have enough counts to compensate. In such cases, non-selected taxa are zeroed and selected taxa are scaled down to preserve library size. A warning is issued when this occurs. To avoid this, use moderate multipliers (2-3) or select taxa with lower initial abundance.

To target NA values in the factor column, include NA in conditions (e.g., conditions = c("High", NA)).

Value

A phyloseq object with modified OTU counts. The selected taxa are stored as an attribute accessible via attr(result, "taxa_modified").

Author(s)

Adrien Taudière

Examples

```
# Multiply counts by 1.2 for "High" samples in half of taxa
d <- multiply_counts_pq(data_fungi,
  fact = "Height",
  conditions = "High", multipliers = 2
)

# Multiple conditions with different multipliers
d2 <- multiply_counts_pq(data_fungi,
  fact = "Height",
  conditions = c("High", "Low"), multipliers = c(1.5, 0.8)
)

# Target NA samples
```

```
d3 <- multiply_counts_pq(data_fungi,
  fact = "Height",
  conditions = c("High", NA), multipliers = c(1.2, 0.5)
)

sum(data_fungi@otu_table)
sum(d@otu_table)
sum(d2@otu_table)
sum(d3@otu_table)

# Apply only to 30% of taxa with a reproducible seed set
d4 <- multiply_counts_pq(data_fungi,
  fact = "Height",
  conditions = "High", multipliers = 2, prop_taxa = 0.3, seed = 42
)

# Simulate DA for compositional methods (ALDEx2, ANCOM-BC):
# Use compensate=TRUE to create a real compositional shift
# Use min_prevalence to select only prevalent taxa
# Use small prop_taxa to affect few taxa strongly
d_da <- multiply_counts_pq(
  data_fungi,
  fact = "Height",
  conditions = "High",
  multipliers = 5,
  prop_taxa = 0.05,
  min_prevalence = 0.5,
  compensate = TRUE,
  seed = 123
)

# Library sizes are preserved
sum(sample_sums(data_fungi))
sum(sample_sums(d_da))

# Test with maaslin3
res <- maaslin3_pq(d_da,
  formula = "~Height"
)

gg_maaslin3_plot(res, type = "volcano")


# Set to 0 the sequences numbers of 80% of taxa in "High" samples
 data_fungi_mini_with_less_otu_in_High =  multiply_counts_pq(data_fungi_mini,
      fact = "Height", prop=0.8,
      conditions = "High",
      multipliers = 0)

ggbetween_pq(data_fungi_mini_with_less_otu_in_High,
  "Height",
  one_plot=T)
```

`n_levels_lpq` *Count unique taxonomic levels across phyloseq objects*

Description

Creates a summary table showing the number of unique taxonomic values (levels) for each taxonomic rank across all phyloseq objects in a list_phyloseq. This is useful for comparing taxonomic resolution and diversity across different datasets or classification methods.

Usage

```
n_levels_lpq(x, taxonomic_ranks, na.rm = TRUE)
```

Arguments

`x`	(required) A list_phyloseq object.
`taxonomic_ranks`	(character vector, required) Names of taxonomic ranks to count. Must be present in the tax_table of ALL phyloseq objects in the list.
`na.rm`	(logical, default TRUE) If TRUE, NA values are excluded when counting unique levels.

Value

A data frame with:

Rows One row per phyloseq object (named by the phyloseq name)

Columns One column per taxonomic rank, containing the count of unique values for that rank in that phyloseq object

Author(s)

Adrien Taudière

See Also

`upset_lpq()`

Examples

```
lpq <- list_phyloseq(list(fungi = data_fungi, fungi_mini = data_fungi_mini))

n_levels_lpq(lpq, c("Phylum", "Class", "Order", "Family", "Genus"))
```

`permute_da_pq` *Simulate differential abundance by redistributing OTU counts*

Description

Creates a fake differential abundance signal by redistributing counts within matched samples: selected "DA" taxa receive counts taken from non-selected taxa. Library sizes (row sums) are preserved, but taxa totals (column sums) are allowed to change to create a detectable DA signal.

This approach is suitable for testing DA methods (ALDEx2, ANCOM-BC) because it creates a true compositional shift while maintaining the library size normalization these methods rely on.

Usage

```
permute_da_pq(
  physeq,
  fact,
  conditions,
  effect_size = 3,
  prop_taxa = 0.05,
  min_prevalence = 0.1,
  seed = NULL,
  verbose = FALSE
)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object.
`fact`	(character, required) Column name in `sample_data` for grouping.
`conditions`	(character vector, required) Levels of `fact` where DA should be simulated.
`effect_size`	(numeric, default 3) The fold-change in relative abundance for selected taxa in matched samples. Values > 1 increase abundance.
`prop_taxa`	(numeric, default 0.05) Proportion of taxa to be selected as DA taxa.
`min_prevalence`	(numeric, default 0.1) Minimum prevalence in matched samples for a taxon to be eligible.
`seed`	(integer, default NULL) Random seed for reproducibility.
`verbose`	(logical, default FALSE) Print progress info.

Details

The algorithm works as follows:

1. Select a subset of prevalent taxa as "DA" taxa
2. For matched samples only:
    - Multiply selected taxa counts by `effect_size`

- Scale down non-selected taxa to preserve library size

3. Round to integers

Unlike multiply_counts_pq(), this function guarantees that library sizes are exactly preserved (except for rounding), creating a pure compositional shift that DA methods can detect.

Value

A phyloseq object with modified OTU counts creating a DA signal. The selected DA taxa are stored in attr(result, "da_taxa").

Author(s)

Adrien Taudière

See Also

multiply_counts_pq()

Examples

```
## Not run:
d_da <- permute_da_pq(
  data_fungi,
  fact = "Height",
  conditions = "High",
  effect_size = 3,
  prop_taxa = 0.05,
  seed = 123
)

# Library sizes are preserved
all.equal(sample_sums(data_fungi), sample_sums(d_da))

# Test with ALDEx2
res <- MiscMetabar::aldex_pq(d_da,
  bifactor = "Height",
  modalities = c("Low", "High")
)
gg_aldex_plot(res, type = "volcano")

# Check which taxa were made DA
attr(d_da, "da_taxa")

## End(Not run)
```

`rainplot_taxo_na` *Rainplot of the nb taxa assigned (not NA)*

Description

Useful to compare taxonomy from two different source/db/algo

Usage

```
rainplot_taxo_na(
  physeq,
  ranks = NULL,
  min_nb_seq = 0,
  merge_sample_by = NULL,
  sample_colored = FALSE,
  sample_linked = FALSE,
  ...
)
```

Arguments

`physeq`	(required) A `phyloseq-class` object obtained using the `phyloseq` package.
`ranks`	(character or integer vector, default NULL) The ranks to include in the rainplot. If left to NULL, all ranks are used. Each rank can be defined either by integer for the index or by its full names (exactly matching the colnames of the `tax_table` slot).
`min_nb_seq`	(integer, default 0) Minimum number of sequences to filter out taxa with low abundance.
`merge_sample_by`	(character, default NULL) A vector to determine which samples to merge using `MiscMetabar::merge_samples2()` function. Need to be in `physeq@sam_data`.
`sample_colored`	(logical, default FALSE) If TRUE, points are colored by samples.
`sample_linked`	(logical, default FALSE) If TRUE, points are linked by samples.
`...`	Additional arguments passed to `ggrain::geom_rain()`.

Value

A ggplot2 object

Author(s)

Adrien Taudière

Examples

```
rainplot_taxo_na(subset_taxa_pq(Glom_otu, taxa_sums(Glom_otu) > 5000),
  ranks = c("Family", "Family__eukaryome_Glomero")
)
## Not run:
rainplot_taxo_na(Glom_otu)

Glom_otu@sam_data$tmt_type <- paste0(Glom_otu@sam_data$Tmt, "_", Glom_otu@sam_data$Type)
rainplot_taxo_na(
  Glom_otu,
  merge_sample_by = "tmt_type",
  sample_colored = TRUE,
  sample_linked = TRUE
)
rainplot_taxo_na(Glom_otu, ranks = c(4, 12), rain.side = "f1x1")
rainplot_taxo_na(
  Glom_otu,
  ranks = c(6, 14),
  rain.side = "f1x1",
  sample_linked = TRUE
) +
  theme(legend.position = "none")

## End(Not run)
```

---

remove_phyloseq *Remove a phyloseq object from a list_phyloseq*

Description

Remove a phyloseq object from a list_phyloseq

Usage

```
remove_phyloseq(x, name)
```

Arguments

x	(required) A list_phyloseq object.
name	(character or integer) Name or index of the phyloseq object to remove.

Value

A new list_phyloseq object without the removed phyloseq

rename_ranks_pq *Rename names of ranks in the tax_table slot of a phyloseq object*

Description

The users can use the function using a couple of vector old_names/new_names or using a pattern to replace by replacement

Usage

```
rename_ranks_pq(
  physeq,
  old_names = NULL,
  new_names = NULL,
  pattern = NULL,
  replacement = NULL,
  fixed = FALSE,
  perl = FALSE,
  useBytes = FALSE
)
```

Arguments

physeq	(required) A phyloseq object.
old_names	(character vector, default NULL) Names of the columns to replace.
new_names	(character vector, default NULL) New names for the columns.
pattern	(character, default NULL) Pattern to replace by the replacement argument.
replacement	(character, default NULL) Replacement string for the pattern.
fixed	(logical, default FALSE) See ?grep.
perl	(logical, default FALSE) See ?grep.
useBytes	(logical, default FALSE) See ?grep.

Value

An object of class phyloseq

Author(s)

Adrien Taudière

Examples

```
rename_ranks_pq(data_fungi, c("Confidence.Ranking", "Phylum"), c("Conf.Rank.Guild", "Phyla"))@tax_table |>
  colnames()
rename_ranks_pq(data_fungi, pattern = ".", replacement = "_", fixed = TRUE)@tax_table |>
  colnames()
```

`resolve_taxo_conflict` *Resolve taxonomic conflict in the tax_table of a phyloseq object*

Description

Resolve taxonomic conflict in the tax_table of a phyloseq object

Usage

```
resolve_taxo_conflict(
  physeq,
  pattern_tax_ranks = NULL,
 method = c("consensus", "rel_majority", "abs_majority", "preference", "unanimity"),
  strict = FALSE,
  second_method = c("consensus", "rel_majority", "abs_majority", "preference",
    "unanimity"),
  nb_agree_threshold = 1,
  keep_tax_ranks = TRUE,
  new_names = NULL,
  preference_pattern = NULL,
  collapse_string = "/",
  replace_collapsed_rank_by_NA = FALSE
)
```

Arguments

`physeq`	(required) A phyloseq object.
`pattern_tax_ranks`	(character vector, default NULL) A vector of patterns to aggregate taxonomic ranks. For example "^Genus" stands for all taxonomic ranks (columns in tax_table slot) starting with Genus.
`method`	(character, default "consensus") One of "consensus", "rel_majority", "abs_majority", "preference" or "unanimity". See details in the documentation of the function `MiscMetabar::resolve_vector_ranks()`.
`strict`	(logical, default FALSE) If TRUE, NA are considered as informative in resolving conflict (i.e. NA are taken into account in vote). See details for more information.
`second_method`	(character, default "consensus") One of "consensus", "rel_majority", "abs_majority", or "unanimity". Only used if method = "preference". See details.
`nb_agree_threshold`	(integer, default 1) The minimum number of times a value must arise to be selected using vote. If 2, only taxonomic values present at least 2 times in the vector are kept.
`keep_tax_ranks`	(logical, default TRUE) If TRUE, keep the old taxonomic ranks in the result.

new_names (character vector, default NULL) A vector of new names for the taxonomic columns.

preference_pattern
(character, default NULL) A pattern to match the only column used as preferred one if method = "preference".

collapse_string
(character, default "/") The character to collapse taxonomic names when multiple assignment is done.

replace_collapsed_rank_by_NA
(logical, default FALSE) If TRUE, all multiple assignments (all taxonomic rank including the 'collapse_string' parameter) are replaced by NA.

Value

A phyloseq object

Author(s)

Adrien Taudière

Examples

```
data_fungi_mini_new <- assign_sintax(data_fungi_mini,
  ref_fasta = system.file("extdata", "mini_UNITE_fungi.fasta.gz",
    package = "MiscMetabar"
  ),
  behavior = "add_to_phyloseq"
)

resolve_taxo_conflict(data_fungi_mini_new, pattern_tax_ranks = c("^Genus\\."), method = "consensus", new_names =
resolve_taxo_conflict(data_fungi_mini_new, pattern_tax_ranks = c("^Genus\\."), method = "consensus", new_names =

resolve_taxo_conflict(data_fungi_mini_new, pattern_tax_ranks = c("^Genus\\."), method = "preference", preference_
resolve_taxo_conflict(data_fungi_mini_new, pattern_tax_ranks = c("^Genus\\."), method = "abs_majority")@tax_tabl
resolve_taxo_conflict(data_fungi_mini_new, pattern_tax_ranks = c("^Genus\\."), method = "rel_majority")@tax_tabl
resolve_taxo_conflict(data_fungi_mini_new, pattern_tax_ranks = c("^Genus\\."), method = "unanimity")@tax_table

resolve_taxo_conflict(data_fungi_mini_new, pattern_tax_ranks = c("^Genus\\.", "^Family\\.", "^Species\\."), meth
```

---

select_ranks_pq *Select taxonomic ranks in a phyloseq object*

Description

Useful in tidyverse-like pipeline.

Usage

```
select_ranks_pq(physeq, ...)
```

Arguments

physeq	(required) A phyloseq object.
...	One or more unquoted expressions separated by commas. Variable names can be used as if they were positions in the data frame, so expressions like x:y can be used to select a range of variables. See ?dplyr::select.

Value

A phyloseq object

Author(s)

Adrien Taudière

Examples

```
select_ranks_pq(data_fungi, Order, Family)@tax_table |>
  dim()
select_ranks_pq(data_fungi, !Order)@tax_table |>
  colnames()
#
```

---

shared_mod_lpq *Display shared sample_data modalities*

Description

Create a table showing which sample_data variable modalities are shared across phyloseq objects in a list_phyloseq.

Create a table showing which sample_data variable modalities are shared across phyloseq objects in a list_phyloseq.

Usage

```
shared_mod_lpq(x, max_modalities = NULL)

shared_mod_lpq(x, max_modalities = NULL)
```

Arguments

x	(required) A list_phyloseq object.
max_modalities	(integer, default 10) Maximum number of modalities to display per variable.

Details

Value

A tibble or NULL if no shared modalities exist

A tibble or NULL if no shared modalities exist

Examples

```
lpq <- list_phyloseq(list(data1 = data_fungi, data2 = data_fungi_mini))

shared_mod_lpq(lpq)
shared_mod_lpq(lpq, 10)
lpq <- list_phyloseq(list(data1 = data_fungi, data2 = data_fungi_mini))

shared_mod_lpq(lpq)
shared_mod_lpq(lpq, 10)
```

---

simple_venn_pq *Venn diagram of shared taxa across sample groups*

Description

Draws a Venn diagram showing shared and unique taxa (or higher-rank groups) across 2 to 4 levels of a sample variable, using only ggplot2 (no external Venn diagram package needed).

When `taxonomic_rank` is a character vector of length > 1 (the default), all ranks are displayed in a single combined figure using the patchwork package (must be installed). Set `combine = FALSE` to get a named list of individual plots instead.

For 2 and 3 groups, circles are used. For 4 groups, ellipses are used to ensure all intersection regions are representable.

Usage

```
simple_venn_pq(
  physeq,
  fact = NULL,
  min_nb_seq = 0,
  taxonomic_rank = c("Class", "Order", "Family", "Genus", "Species"),
  na_remove = TRUE,
  count_type = c("rank", "taxa", "sequences", "rank_taxa"),
  add_nb_samples = TRUE,
  fill_alpha = 0.3,
  border_size = 0.8,
  text_size = 4,
  scale_text = FALSE,
  hide_zero = TRUE,
  label_size = 4.5,
  colors = NULL,
  show_na_count = FALSE,
```

```
  count_taxa = TRUE,
  match_by = c("refseq", "names"),
  combine = TRUE,
  verbose = TRUE,
  .lpq_n_samples = NULL
)
```

Arguments

physeq	(phyloseq or list_phyloseq, required) A phyloseq object, or a list_phyloseq object. When a list_phyloseq is provided, it is first merged into a single phyloseq using `merge_lpq()` (each original phyloseq becomes one sample) and the `fact` parameter is automatically set to `"source_name"`.
fact	(character, required when `physeq` is a phyloseq) Name of a variable in `sample_data(physeq)` defining the groups (2-4 levels). Ignored when `physeq` is a list_phyloseq.
min_nb_seq	(integer, default 0) Minimum total read count for a taxon to be considered present in a group. A taxon must have strictly more than `min_nb_seq` reads in a group to be included.
taxonomic_rank	(character or NULL) Taxonomic rank(s) at which to aggregate (via `phyloseq::tax_glom()`) before computing the Venn diagram. Defaults to all standard ranks (Kingdom through Species). Use `NULL` to skip aggregation and work at ASV/OTU level.
na_remove	(logical, default TRUE) Remove samples with NA in `fact` and, when aggregating, taxa with NA at `taxonomic_rank`.
count_type	(character, default `"rank"`) What to count in each Venn region. One of: • `"rank"`: number of unique taxonomic levels (e.g. number of shared Classes). This is the default. • `"taxa"`: number of ASVs/OTUs assigned to the shared taxonomic levels. • `"sequences"`: total number of reads for ASVs/OTUs assigned to the shared taxonomic levels. • `"rank_taxa"`: shows both rank and taxa counts as `"nb_rank (nb_taxa)"`. Ignored when `taxonomic_rank` is `NULL` (ASV-level), where `"rank"` and `"taxa"` are equivalent.
add_nb_samples	(logical, default TRUE) Append sample count to group labels.
fill_alpha	(numeric, default 0.3) Fill transparency for shapes.
border_size	(numeric, default 0.8) Border line width.
text_size	(numeric, default 4) Base size of count labels inside regions.
scale_text	(logical, default FALSE) If `TRUE`, scale the size of count labels proportionally to the count value. The `text_size` parameter then acts as the base (minimum) size.
hide_zero	(logical, default TRUE) If `TRUE`, hide count labels that are zero (or `"0 (0)"` when `count_type = "rank_taxa"`).
label_size	(numeric, default 4.5) Size of group name labels.
colors	(character or NULL) Vector of colors, one per group. Defaults to a 4-color qualitative palette.

show_na_count	(logical, default FALSE) If TRUE, display the number of taxa with NA at the chosen taxonomic_rank in the bottom-left corner of the plot. When count_type = "taxa", the sum of all Venn region counts plus the NA count equals ntaxa(physeq). Ignored when taxonomic_rank is NULL.
count_taxa	(logical, default TRUE) If TRUE, append a "Taxa" panel to the Venn diagram showing shared and unique individual taxa (ASVs/OTUs) alongside the aggregated taxonomic ranks. A temporary Taxa column is added to the tax_table with each taxon's name as its value. Ignored when taxonomic_rank is NULL.
match_by	(character, default "refseq") Passed to merge_lpq() when physeq is a list_phyloseq. One of "refseq" or "names".
combine	(logical, default TRUE) When taxonomic_rank has length > 1, combine plots into a single patchwork figure. Set to FALSE to return a named list of individual ggplot objects. Requires the patchwork package.
verbose	(logical, default TRUE) Print a message when no taxa meet the criteria.

Value

A ggplot2 object (single rank), a patchwork object (multiple ranks with combine = TRUE), or a named list of ggplot2 objects (multiple ranks with combine = FALSE).

Author(s)

Adrien Taudiere

Examples

```
# Default: all ranks combined in one figure
simple_venn_pq(data_fungi_mini, "Height")

# At genus level only
simple_venn_pq(data_fungi_mini, "Height", taxonomic_rank = "Genus")

# Multiple ranks as a list
plots <- simple_venn_pq(
  data_fungi_mini, "Height",
  taxonomic_rank = c("Family", "Genus"),
  combine = FALSE
)
plots[["Family"]]

# Count ASVs instead of rank levels
simple_venn_pq(data_fungi_mini, "Height",
  taxonomic_rank = "Genus", count_type = "taxa"
)

# Scale text by count value
simple_venn_pq(data_fungi_mini, "Height",
  taxonomic_rank = "Genus", scale_text = TRUE
)
```

```
# From a list_phyloseq object
lpq <- list_phyloseq(list(
  fungi = data_fungi_mini,
  fungi2 = data_fungi
))
simple_venn_pq(lpq, taxonomic_rank = "Genus", count_taxa)
```

`taxo2tree` *Convert taxonomy dataframe to phylogenetic tree*

Description

Creates a phylo object from a taxonomy table with hierarchical taxonomic ranks as columns.

Usage

```
taxo2tree(
  physeq,
  ranks = c("Domain", "Phylum", "Class", "Order", "Family", "Genus", "Species"),
  internal_node_singletons = TRUE,
  use_taxa_names = TRUE
)
```

Arguments

`physeq`	(required) A `phyloseq-class` object
`ranks`	Character vector specifying the column names to use as taxonomic ranks, ordered from highest to lowest. By default: c("Domain", "Phylum", "Class", "Order", "Family", "Genus", "Species").
`internal_node_singletons`	Logical, if TRUE, create internal nodes for singleton. If FALSE, internal nodes with only one descendant are discarded.
`use_taxa_names`	Logical, if TRUE (default), use the taxa names (rownames, e.g., ASV_1, ASV_2) as terminal leaves. If FALSE, collapse identical taxonomy paths and use the lowest rank value as tip labels. This is useful for creating cleaner trees that show only taxonomic structure without individual ASV/OTU names.

Details

Value

A phylo object (ape package) representing the taxonomic tree.

Author(s)

Adrien Taudière

Examples

```
data_fungi_mini@phy_tree <-
  phyloseq::phy_tree(taxo2tree(data_fungi_mini,
    ranks = c(
      "Domain", "Phylum", "Class", "Order",
      "Family", "Genus", "Genus_species"
    )
  ))

library(ggtree)

ggtree(data_fungi_mini@phy_tree) +
  geom_nodelab(size = 2, nudge_x = -0.2, nudge_y = 0.6) +
  geom_tiplab()

psm <- psmelt(data_fungi_mini) |>
  group_by(OTU) |>
  summarize(
    Mol_Abund = sum(Abundance),
    Frequency = sum(Abundance > 0),
    Class = tidyr::replace_na(unique(Class), "Unknown")
  )


ggtree(data_fungi_mini@phy_tree) %<+% psm +
  geom_tiplab(offset = .6, hjust = .5) +
  geom_tippoint(aes(
    shape = Class,
    color = log10(1 + Mol_Abund),
    size = Frequency
  )) +
  geom_nodelab(nudge_x = -0.2, nudge_y = 0.6) +
  scale_color_viridis_c()


# Without internal node singletons and only until the Genus level
tree_wo_singletons <-
  phyloseq::phy_tree(taxo2tree(data_fungi_mini,
    ranks = c(
      "Domain", "Phylum", "Class", "Order",
      "Family", "Genus"
    ),
    internal_node_singletons = FALSE
  ))

ggtree::ggtree(tree_wo_singletons) +
  ggtree::geom_nodelab(size = 2, nudge_x = -0.2, nudge_y = 0.6) +
  ggtree::geom_tiplab()

# Without taxa names (collapse identical paths)
tree_no_taxa <- taxo2tree(data_fungi_mini,
  ranks = c("Domain", "Phylum", "Class", "Order", "Family", "Genus"),
```

```
  use_taxa_names = FALSE
)

ggtree::ggtree(tree_no_taxa) +
  ggtree::geom_nodelab(size = 2, nudge_x = -0.2, nudge_y = 0.6) +
  ggtree::geom_tiplab()
```

`taxtab_replace_pattern_by_NA` *Replace taxonomic value with a given pattern by NA*

Description

Use `base::gsub()` for substitution.

Usage

```
taxtab_replace_pattern_by_NA(
  physeq,
  patterns = c(".*_incertae_sedis", "unclassified.*"),
  taxonomic_ranks = NULL,
  progress_bar = FALSE,
  ...
)
```

Arguments

`physeq`	(required) A `phyloseq-class` object obtained using the `phyloseq` package.
`patterns`	(character vector, default c("*_incertae_sedis", "unclassified.*")) A vector of patterns to select taxonomic values to convert to NA.
`taxonomic_ranks`	(character vector, default NULL) A vector of taxonomic ranks where the substitution should occur. If left to NULL, all taxonomic ranks are modified.
`progress_bar`	(logical, default FALSE) If TRUE, print progress during the calculation.
`...`	Additional arguments passed to `base::gsub()`.

Value

A phyloseq object

Author(s)

Adrien Taudière

Examples

```
data_fungi@tax_table["ASV85", "Family"]
data_fungi2 <- taxtab_replace_pattern_by_NA(data_fungi, "fam_Incertae_sedis", taxonomic_ranks = "Family")
data_fungi2@tax_table["ASV85", "Family"]

# By default patterns ".*_incertae_sedis" and "unclassified.*" are replaced by NA
data_fungi3 <-
  taxtab_replace_pattern_by_NA(data_fungi, ignore.case = TRUE, progress_bar = TRUE)
data_fungi3@tax_table["ASV85", "Family"]
```

---

tc_bar *Barchart of ratio to compare 2 taxonomic ranks*

Description

Useful to compare taxonomy from two different source/db/algo side-by-side

Usage

```
tc_bar(
  physeq,
  rank_1,
  rank_2,
  color_rank,
  point_size = 0.3,
  point_alpha = 0.3,
  merge_sample_by = NULL,
  log10trans = TRUE
)
```

Arguments

physeq	(required) A phyloseq-class object obtained using the phyloseq package.
rank_1	(character or integer) Define the first taxonomic rank as the number or the name of the column in tax_table slot.
rank_2	(character or integer) Define the second taxonomic rank as the number or the name of the column in tax_table slot.
color_rank	(character or integer) Define the taxonomic rank for color as the number or the name of the column in tax_table slot.
point_size	(numeric, default 0.3) Size of points.
point_alpha	(numeric, default 0.3) Transparency of points.
merge_sample_by	(character, default NULL) A vector to determine which samples to merge using MiscMetabar::merge_samples2() function. Need to be in physeq@sam_data.
log10trans	(logical, default TRUE) If TRUE, the abundance is log10 transformed.

Value

A ggplot2 object

Author(s)

Adrien Taudière

Examples

```
tc_bar(subset_taxa_pq(Glom_otu, taxa_sums(Glom_otu) > 5000), rank_1 = 5, rank_2 = 13, color_rank = 3)
## Not run:
tc_bar(Glom_otu, rank_1 = 5, rank_2 = 13, color_rank = 3)
tc_bar(as_binary_otu_table(Glom_otu), rank_1 = 5, rank_2 = 13, color_rank = 3, log10trans = FALSE)
tc_bar(Glom_otu,
  rank_1 = "Genus",
  rank_2 = "Genus__eukaryome_Glomero",
  color_rank = "Family"
)

## End(Not run)
```

---

tc_circle *Circle of correspondence between two taxonomic levels*

Description

Useful to compare taxonomy from two different source/db/algo side-by-side

Usage

```
tc_circle(
  physeq,
  rank_1 = NULL,
  rank_2 = NULL,
  suffix_1 = "_1",
  suffix_2 = "_2"
)
```

Arguments

physeq	(required) A phyloseq-class object obtained using the phyloseq package.
rank_1	(character or integer) Define the first taxonomic rank as the number or the name of the column in tax_table slot.
rank_2	(character or integer) Define the second taxonomic rank as the number or the name of the column in tax_table slot.
suffix_1	(character, default "_1") A suffix to add to rank_1 values.
suffix_2	(character, default "_2") A suffix to add to rank_2 values.

Value

A ggplot2 object

Author(s)

Adrien Taudière

Examples

```
tc_circle(
  Glom_otu,
  "Genus__eukaryome_Glomero",
  "Genus",
  suffix_1 = "_Euk",
  suffix_2 = "_Marjaam"
)

tc_circle(
  Glom_otu,
  "Family__eukaryome_Glomero",
  "Family",
  suffix_1 = "_Euk",
  suffix_2 = "_Marjaam"
)
```

---

tc_congruence_metrics *Compute congruence metrics between two taxonomic assignments*

Description

Computes metrics quantifying the congruence between two taxonomic assignments for the same set of taxa (ASVs/OTUs). This is useful for comparing different taxonomic databases, assignment methods, or reference versions.

Usage

```
tc_congruence_metrics(physeq_1, physeq_2 = NULL, ranks_1, ranks_2 = NULL)
```

Arguments

physeq_1	(required) A phyloseq-class object with the first taxonomic assignment.
physeq_2	(phyloseq, default NULL) A phyloseq-class object with the second taxonomic assignment. If NULL, uses physeq_1 (useful for comparing different rank columns from the same object).
ranks_1	(character vector, required) Taxonomic rank names to use for the first assignment. Must match column names in physeq_1's tax_table.
ranks_2	(character vector, default NULL) Taxonomic rank names to use for the second assignment. Must match column names in physeq_2's tax_table. If NULL, uses the same ranks as ranks_1 vector.

Value

A list with the following components:

summary A data frame with counts and percentages for each category, including `leaf_match_congruence` which counts taxa where the deepest classification matches regardless of the path

only_in_1 Character vector of taxa names present only in physeq_1

only_in_2 Character vector of taxa names present only in physeq_2

classified_only_1 Character vector of common taxa classified only in physeq_1 (all NA in physeq_2 for the specified ranks)

classified_only_2 Character vector of common taxa classified only in physeq_2 (all NA in physeq_1 for the specified ranks)

unclassified_both Character vector of common taxa with all NA in both physeq objects

total_congruent Character vector of taxa with identical taxonomic paths (same values at all ranks, same NA positions)

partial_1_deeper Character vector of taxa where physeq_1 classifies to a deeper rank but agrees on shared ranks

partial_2_deeper Character vector of taxa where physeq_2 classifies to a deeper rank but agrees on shared ranks

incongruent_leaves Character vector of taxa with disagreement at the deepest (leaf) level where both have assignments

incongruent_nodes Character vector of taxa with disagreement at higher (internal node) levels, even if leaves match

details A data frame with per-taxon details including: taxon name, depth_1, depth_2, leaf_1, leaf_2, leaf_match (TRUE if leaves are equal), and category

Author(s)

Adrien Taudière

Examples

```
# Compare two taxonomic assignments from the same phyloseq object
metrics <- tc_congruence_metrics(
  subset_taxa(Glom_otu, Phyla == "Fungi" | Phylum__eukaryome_Glomero == "Fungi"),
  ranks_1 = c("Phyla", "Class", "Order", "Family"),
  ranks_2 = c(
    "Phylum__eukaryome_Glomero", "Class__eukaryome_Glomero",
    "Order__eukaryome_Glomero", "Family__eukaryome_Glomero"
  )
)

# View summary
metrics$summary

# Get taxa with leaf-level incongruence
metrics$incongruent_leaves
```

tc_df_pq *Contingency table of two taxonomic ranks*

Description

Creates a cross-tabulation (contingency table) comparing two taxonomic ranks from a phyloseq object. Useful for comparing taxonomic assignments from different databases, algorithms, or taxonomic levels.

Usage

```
tc_df_pq(physeq, rank_1 = "Family", rank_2 = "Class", ...)
```

Arguments

physeq	(required) A phyloseq-class object obtained using the phyloseq package.
rank_1	(character, default "Family") The name of the first taxonomic rank (column in tax_table slot) for the cross-tabulation rows.
rank_2	(character, default "Class") The name of the second taxonomic rank (column in tax_table slot) for the cross-tabulation columns.
...	Additional arguments passed to gtsummary::tbl_cross().

Value

A gtsummary tbl_cross object displaying the cross-tabulation of the two taxonomic ranks.

Author(s)

Adrien Taudière

Examples

```
tc_df_pq(data_fungi_mini)
tc_df_pq(data_fungi_mini, rank_1 = "Order", rank_2 = "Family")

## Not run:
# Compare taxonomic assignments from different methods
ref_fasta <- system.file("extdata", "mini_UNITE_fungi.fasta.gz",
  package = "MiscMetabar", mustWork = TRUE
)
data_fungi_mini2 <- data_fungi_mini |>
  add_new_taxonomy_pq(ref_fasta, suffix = "_sintax", method = "sintax") |>
  add_new_taxonomy_pq(ref_fasta, suffix = "_lca", method = "lca")

tc_df_pq(data_fungi_mini2, rank_1 = "Class_lca", rank_2 = "Class_sintax")

## End(Not run)
```

`tc_linked_trees` *Plot two taxonomy trees with linked correspondences*

Description

Creates a visualization of two taxonomy trees facing each other, with matching taxa aligned horizontally. Optionally draws lines connecting matching leaf taxa. Useful for comparing taxonomic assignments from different databases or classification methods.

Usage

```
tc_linked_trees(
  physeq_1,
  physeq_2 = NULL,
  ranks_1,
  ranks_2,
  tree_distance = 1,
  show_tip_links = TRUE,
  link_by_taxa = FALSE,
  link_alpha = 0.5,
  link_color = "grey50",
  show_labels = TRUE,
  label_size = 2,
  internal_node_singletons = FALSE,
  use_taxa_names = FALSE
)
```

Arguments

`physeq_1`	(required) A `phyloseq-class` object for the left tree.
`physeq_2`	(phyloseq, default NULL) A `phyloseq-class` object for the right tree. If NULL, uses physeq_1 (useful for comparing different rank columns from the same object).
`ranks_1`	(character vector, required) Taxonomic rank names to use for the first tree. Must match column names in physeq_1's tax_table.
`ranks_2`	(character vector, required) Taxonomic rank names to use for the second tree. Must match column names in physeq_2's tax_table.
`tree_distance`	(numeric, default 1) Distance between the two trees.
`show_tip_links`	(logical, default TRUE) Whether to draw lines between matching leaf taxa (tips). Internal nodes are never linked.
`link_by_taxa`	(logical, default FALSE) If TRUE, links are drawn based on taxa (ASV/OTU) correspondence: for each taxon in physeq_1, a link is drawn to its corresponding tip in tree_2 based on its taxonomy. Line width is proportional to the number of taxa sharing each link. Requires that physeq_1 and physeq_2 have the same taxa names (rownames).

link_alpha	(numeric, default 0.5) Transparency of the correspondence lines.
link_color	(character, default "grey50") Color of the correspondence lines. Use NULL to color by label.
show_labels	(logical, default TRUE) Whether to show node/tip labels.
label_size	(numeric, default 2) Base size for labels. Labels are scaled by depth: shallower nodes (closer to root) get larger text (up to 1.5x), deeper nodes get smaller text (down to 0.6x). All labels are positioned above their branch to avoid overlap.
internal_node_singletons	(logical, default FALSE) Whether to create internal nodes for singletons. Passed to taxo2tree.
use_taxa_names	(logical, default FALSE) Whether to use taxa names (e.g., ASV_1, ASV_2) as terminal leaves. If FALSE (default), collapses identical taxonomy paths and uses the lowest rank value as tip labels. Passed to taxo2tree.

Value

A ggplot2 object that can be further customized.

Author(s)

Adrien Taudière

Examples

```
# Compare two taxonomic assignments from the same phyloseq object
# using different rank columns (physeq_2 defaults to physeq_1)
tc_linked_trees(
  Glom_otu,
  ranks_1 = c("Kingdom", "Class", "Order", "Family"),
  ranks_2 = c(
    "Kingdom__eukaryome_Glomero", "Class__eukaryome_Glomero",
    "Order__eukaryome_Glomero", "Family__eukaryome_Glomero"
  )
)

## Not run:
# Link by taxa with line width proportional to ASV count
tc_linked_trees(
  Glom_otu,
  ranks_1 = c("Kingdom", "Class", "Order", "Family"),
  ranks_2 = c(
    "Kingdom__eukaryome_Glomero", "Class__eukaryome_Glomero",
    "Order__eukaryome_Glomero", "Family__eukaryome_Glomero"
  ),
  link_by_taxa = TRUE
)

# Without tip links
tc_linked_trees(
  Glom_otu,
```

```
  ranks_1 = c("Kingdom", "Class", "Order", "Family"),
  ranks_2 = c(
    "Kingdom__eukaryome_Glomero", "Class__eukaryome_Glomero",
    "Order__eukaryome_Glomero", "Family__eukaryome_Glomero"
  ),
  show_tip_links = FALSE
)

# Color links by label
tc_linked_trees(
  Glom_otu,
  ranks_1 = c("Kingdom", "Class", "Order", "Family"),
  ranks_2 = c(
    "Kingdom__eukaryome_Glomero", "Class__eukaryome_Glomero",
    "Order__eukaryome_Glomero", "Family__eukaryome_Glomero"
  ),
  link_color = NULL
) +
  theme(legend.position = "none")

# Using one phyloseq object for both trees
# and different rank levels with link_by_taxa
tc_linked_trees(
  Glom_otu,
  ranks_1 = c("Kingdom", "Class", "Order", "Family"),
  ranks_2 = c(
    "Kingdom__eukaryome_Glomero", "Class__eukaryome_Glomero",
    "Order__eukaryome_Glomero", "Family__eukaryome_Glomero"
  ), link_color = NULL,
  link_by_taxa = TRUE,
) +
  theme(legend.position = "none")

## End(Not run)
```

`tc_metrics_mock` *Compute accuracy metrics of multiple taxonomic assignations method using mock for multi-rank and multi assignation methods*

Description

Compute numerous metrics comparing the computed taxonomic assignation to a true assignation.

Note that to compute all metrics, one need to insert fake taxa (by shuffling sequences and/or by adding external sequences). The user must fake taxa using functions `add_external_seq_pq()`, `add_shuffle_seq_pq()`) before taxonomic assignation.

Usage

```
tc_metrics_mock(
```

```
  physeq,
  ranks_df,
  true_values_df,
  fake_taxa = TRUE,
  fake_pattern = c("^fake_", "^external_")
)
```

Arguments

physeq	(required) A phyloseq-class object obtained using the phyloseq package.
ranks_df	(required). A dataframe with at least one column (one database or one method) and a number of row equal to the column in true_values_df
true_values_df	(required). A dataframe with the true taxonomic assignation. Note that the column names (names of taxonomic ranks) of the true_values_df defined the names present in the tax_level column of the resulting dataframe.
fake_taxa	(logical, default TRUE) If TRUE, the fake_pattern vector is used to identify fake taxa, i.e. taxa who are not in the reference database (see add_external_seq_pq()) or taxa with fake sequences (see add_shuffle_seq_pq()).
fake_pattern	(character vector, default c("^fake_", "^external_")) A vector of patterns used to identify the fake taxa using a regex search in their name.

Value

A long-format dataframe with 4 columns: (i) the name of the method_db (ii) the name of the tax_level (taxonomic rank), (iii) the metrics (see tc_metrics_mock_vec() for more details) and (iv) the values.

Author(s)

Adrien Taudière

See Also

tc_metrics_mock_vec()

---

tc_metrics_mock_vec *Compute accuracy metrics of taxonomic assignation using a mock (known) community for one rank*

Description

Compute numerous metrics comparing the computed taxonomic assignation to a true assignation.

Note that to compute all metrics, one need to insert fake taxa (by shuffling sequences and/or by adding external sequences). The user must fake taxa using functions add_external_seq_pq(), add_shuffle_seq_pq()) before taxonomic assignation.

Usage

```
tc_metrics_mock_vec(
  physeq,
  taxonomic_rank,
  true_values,
  fake_taxa = TRUE,
  fake_pattern = c("^fake_", "^external_"),
  verbose = TRUE
)
```

Arguments

physeq	(required) A phyloseq-class object obtained using the phyloseq package.
taxonomic_rank	(required) Name (or number) of a taxonomic rank to count.
true_values	(required) A vector with the true taxonomic assignation
fake_taxa	(logical, default TRUE) If TRUE, the fake_pattern vector is used to identify fake taxa, i.e. taxa who are not in the reference database (see add_external_seq_pq()) or taxa with fake sequences (see add_shuffle_seq_pq()).
fake_pattern	(character vector, default c("^fake_", "^external_")) A vector of patterns used to identify the fake taxa using a regex search in their name.
verbose	(logical, default TRUE) If TRUE, print informative messages.

Value

A list of metrics (see the confusion matrix article on wikipedia):

- TP (number of *true positive*)
- FP (number of *false positive*)
- FN (number of *false negative*)
- FDR (*false discovery rate*) = FP / (FP + TP)
- TPR (*true positive rate*, also named *recall* or *sensitivity*)
- PPV (*positive predictive value*, also named *precision*) = TP / (TP + FP)
- F1_score (*F1 score*) = 2 * TP / (2 * TP + FP + FN)

If fake taxa are present and fake_taxa is true, other metrics are computed:

- TN (number of *true negative*)
- ACC (*Accuracy*) = (TP + TN) / (TP + TN + FP + FN)
- MCC (*Matthews correlation coefficient*) = (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (FP + TN) * (TN + FN))

Author(s)

Adrien Taudière

See Also

tc_metrics_mock(), add_external_seq_pq(), add_shuffle_seq_pq())

`tc_points_matrix` *Matrix of point to compare two taxonomic ranks*

Description

Useful to compare taxonomy from two different source/db/algo side-by-side

Usage

```
tc_points_matrix(
  physeq,
  rank_1,
  rank_2,
  color_1 = "#dc863b",
  color_2 = "#2e7891",
  stat_across_sample = "sum",
  merge_sample_by = NULL
)
```

Arguments

`physeq`	(required) A `phyloseq-class` object obtained using the `phyloseq` package.
`rank_1`	(character or integer) Define the first taxonomic rank as the number or the name of the column in tax_table slot.
`rank_2`	(character or integer) Define the second taxonomic rank as the number or the name of the column in tax_table slot.
`color_1`	(character, default "#dc863b") Color for rank_1 values.
`color_2`	(character, default "#2e7891") Color for rank_2 values.
`stat_across_sample`	(character, default "sum") Either "mean" or "sum". Set how the abundance is computed across samples.
`merge_sample_by`	(character, default NULL) A vector to determine which samples to merge using `MiscMetabar::merge_samples2()` function. Need to be in `physeq@sam_data`.

Value

A ggplot2 object

Author(s)

Adrien Taudière

Examples

```
tc_points_matrix(
  subset_taxa_pq(Glom_otu, taxa_sums(Glom_otu) > 5000),
  "Order", "Order__eukaryome_Glomero"
)
## Not run:
tc_points_matrix(Glom_otu, 6, 14)
tc_points_matrix(Glom_otu, 4, 12)
tc_points_matrix(Glom_otu, 4, 12, stat_across_sample = "mean")

Glom_otu@sam_data$unique_value <- rep("samp", nsamples(Glom_otu))
tc_points_matrix(as_binary_otu_table(Glom_otu), 5, 13,
  stat_across_sample = "sum", merge_sample_by = "unique_value"
)
tc_points_matrix(as_binary_otu_table(Glom_otu), 5, 13,
  stat_across_sample = "mean"
)
tc_points_matrix(Glom_otu, 5, 13,
  stat_across_sample = "mean"
)

## End(Not run)
```

---

tc_sankey *Sankey diagram to compare two taxonomic ranks*

Description

Creates a Sankey (alluvial) diagram to visualize the correspondence between two taxonomic ranks. Useful for comparing taxonomy assignments from different databases or algorithms.

Usage

```
tc_sankey(physeq, rank_1, rank_2, fill_by = "rank_1")
```

Arguments

physeq	(required) A phyloseq-class object obtained using the phyloseq package.
rank_1	(character or integer, required) Define the first taxonomic rank as the number or the name of the column in tax_table slot.
rank_2	(character or integer, required) Define the second taxonomic rank as the number or the name of the column in tax_table slot.
fill_by	(character, default "rank_1") Which rank to use for fill color. Either "rank_1" or "rank_2".

Value

A ggplot2 object that can be further customized.

Author(s)

Adrien Taudière

Examples

```
tc_sankey(
  Glom_otu,
  "Class__eukaryome_Glomero",
  "Class"
)

# Fill by rank_2 instead
tc_sankey(
  Glom_otu,
  "Class__eukaryome_Glomero",
  "Class",
  fill_by = "rank_2"
)

## Not run:
# Add labels to the strata
tc_sankey(Glom_otu, "Class__eukaryome_Glomero", "Class") +
  geom_label(
    stat = "stratum",
    aes(label = after_stat(stratum)),
    na.rm = TRUE
  ) +
  theme(legend.position = "none")

## End(Not run)
```

---

update_list_phyloseq *Update the summary table and comparison characteristics*

Description

Recomputes the summary_table and comparison slots. Useful after modifying the phyloseq objects in place or to change the comparison parameters.

Usage

```
update_list_phyloseq(
  x,
  same_primer_seq_tech = NULL,
  same_bioinfo_pipeline = NULL
)
```

Arguments

x	(required) A list_phyloseq object.
same_primer_seq_tech	(logical, default NULL) Whether the same primer and sequencing technology was used. If NULL, preserves the original value.
same_bioinfo_pipeline	(logical, default NULL) Whether the same bioinformatics pipeline was used. If NULL, preserves the original value.

Details

Value

An updated list_phyloseq object

---

upset_lpq *UpSet or Venn plot of shared taxonomic values across phyloseq objects*

Description

Create an UpSet plot (or Venn diagram) showing the shared taxonomic values at a specified rank across all phyloseq objects in a list_phyloseq.

Usage

```
upset_lpq(x, tax_rank = "Genus", plot_type = "auto", remove_na = TRUE, ...)
```

Arguments

x	(required) A list_phyloseq object.
tax_rank	(character, required) The name of the taxonomic rank column present in the `@tax_table` slot of each phyloseq object. For example, "Genus", "Family", or "Species".
plot_type	(character, default "auto") Type of plot to generate. One of "auto", "upset", or "venn". If "auto", uses Venn diagram for 4 or fewer phyloseq objects, UpSet plot otherwise.
remove_na	(logical, default TRUE) If TRUE, remove NA values from the taxonomic rank before computing intersections.
...	Additional arguments passed to `ComplexUpset::upset()` or `ggVennDiagram::ggVennDiagram()`.

Details

This function extracts the unique values for the specified taxonomic rank from each phyloseq object and creates a visualization showing the intersections between them. UpSet plots are generally better for visualizing complex intersections with more than 4 sets, while Venn diagrams work well for 2-4 sets.

Value

A ggplot2 object (both UpSet and Venn diagrams)

See Also

upset_lpq()

Examples

```
data("enterotype", package = "phyloseq")
lpq <- list_phyloseq(list(
  fung = data_fungi,
  fung_mini = data_fungi_mini,
  fung_rarefy = rarefy_even_depth(data_fungi),
  enterotype = enterotype
))
upset_lpq(lpq, plot_type = "upset")
lpq2 <- list_phyloseq(list(
  fung = data_fungi,
  fung_mini = data_fungi_mini
))
upset_lpq(lpq2, tax_rank = "Family")
```

Index

add_external_seq_pq, 3
add_external_seq_pq(), *75–77*
add_phyloseq, 4
add_shuffle_seq_pq, 4
add_shuffle_seq_pq(), *75–77*
adonis_lpq, 5
adonis_lpq(), *20*, *22*
ALDEx2::aldex(), *27*
ALDEx2::aldex.plot(), *27*
aldex_lpq, 7
aldex_lpq(), *9*, *26, 27*, *49*
ancombc_lpq, 8
ancombc_lpq(), *7*, *42*, *49*
apply_to_lpq, 10

base::gsub(), *67*
bubbles_pq, 11

compare_refseq, 13
ComplexUpset::upset(), *81*

div_pq, 15
dplyr::across(), *16*

estim_cor_lpq, 16
estim_cor_lpq(), *19, 20*
estim_cor_pq, 18
estim_cor_pq(), *16, 17*, *22*, *33*
estim_diff_lpq, 19
estim_diff_lpq(), *17*, *22*, *33*
estim_diff_pq, 20
estim_diff_pq(), *18–20*

factor_formatter, 22
filter_common_lpq, 23
filter_common_lpq(), *10*
formattable_lpq, 24
formattable_lpq_full, 25

gg_aldex_plot, 26
gg_aldex_plot(), *35*
gg_bubbles_pq, 28
gg_hill_lpq, 31
gg_maaslin3_plot, 33
gg_maaslin3_plot(), *42*
ggplot2::facet_wrap(), *30*
ggplot2::geom_smooth(), *32*
ggplot2::ggplot(), *33*
ggrain::geom_rain(), *56*
ggVennDiagram::ggVennDiagram(), *81*
glmulti_lpq, 37
gtsummary::tbl_cross(), *72*

indicspecies::multipatt(), *48, 49*

list_phyloseq, *13, 14*, *29, 30*, *32, 33*, 39, *44, 45*, *63*

maaslin3_pq, 41
maaslin3_pq(), *35*
merge_lpq, 43
merge_lpq(), *29, 30*, *63, 64*
midasim_pq, 45
MiscMetabar::adonis_pq(), *5, 6*
MiscMetabar::aldex_pq(), *7*, *42*
MiscMetabar::ancombc_pq(), *8, 9*
MiscMetabar::clean_pq(), *23*
MiscMetabar::glmutli_pq(), *37, 38*
MiscMetabar::hill_pq(), *38*
MiscMetabar::merge_samples2(), *56*, *68*, *78*
MiscMetabar::multipatt_pq(), *49*
MiscMetabar::resolve_vector_ranks(), *59*
multipatt_lpq, 48
multipatt_lpq(), *7*, *9*
multiply_counts_pq, 50
multiply_counts_pq(), *47*, *55*

n_levels_lpq, 53

permute_da_pq, 54

permute_da_pq(), *47*
phyloseq::distance(), *5*
phyloseq::tax_glom(), *63*

rainplot_taxo_na, 56
remove_phyloseq, 57
rename_ranks_pq, 58
resolve_taxo_conflict, 59

select_ranks_pq, 60
shared_mod_lpq, 61
simple_venn_pq, 62
simple_venn_pq(), *45*
stats::p.adjust(), *48*

taxo2tree, 65, *74*
taxtab_replace_pattern_by_NA, 67
tc_bar, 68
tc_circle, 69
tc_congruence_metrics, 70
tc_df_pq, 72
tc_linked_trees, 73
tc_metrics_mock, 75
tc_metrics_mock(), *3, 4*, *77*
tc_metrics_mock_vec, 76
tc_metrics_mock_vec(), *3, 4*, *76*
tc_points_matrix, 78
tc_sankey, 79

update_list_phyloseq, 80
update_list_phyloseq(), *10*
upset_lpq, 81
upset_lpq(), *53*, *82*

vegan::adonis2(), *5, 6*
vegan::diversity(), *15, 16*
vegan::renyi(), *15, 16*