Package: tidypq (via r-universe)

June 6, 2026

Title Tidyverse-Style Verbs for Phyloseq Objects

Version 0.1.0

Description Provides tidyverse-style verbs for manipulating phyloseq objects at four scales: samples, taxa, occurrences, and tree. Functions follow a consistent naming convention ({verb}_{scale}_pq) and support data masking for intuitive filtering, selection, and mutation operations.

License MIT + file LICENSE

URL https://github.com/adrientaudiere/tidypq

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

Depends R (>= 4.1.0), phyloseq, MiscMetabar

Imports ape, dplyr (>= 1.0.0), rlang (>= 1.0.0), tibble, tidyselect

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

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

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

Date/Publication 2026-03-13 13:36:55 UTC

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

RemoteRef HEAD

RemoteSha aa85f6fb8a4ffe760bd5bd4e16349a35d6d623fb

Contents

`arrange_samples_pq` *Arrange samples by column values*

Description

Reorder samples based on sample_data columns. Supports the . pronoun to refer to the phyloseq object for sorting by computed values.

Usage

```
arrange_samples_pq(physeq, ..., clean_phyloseq_object = TRUE)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object.
`...`	Variables to sort by. Use `desc()` for descending order. Use . to refer to the phyloseq object.
`clean_phyloseq_object`	if TRUE (default), the resulting phyloseq object is cleaned using `clean_pq()` to remove empty taxa/samples.

Value

A phyloseq object with reordered samples.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Arrange by a single column
arrange_samples_pq(data_fungi, Height)

# Arrange by sequencing depth (descending)
arrange_samples_pq(data_fungi, dplyr::desc(sample_sums(.)))
```

---

arrange_taxa_pq *Arrange taxa by column values*

Description

Reorder taxa based on tax_table columns or computed values. Supports the . pronoun to refer to the phyloseq object for sorting by abundance.

Usage

```
arrange_taxa_pq(physeq, ..., clean_phyloseq_object = TRUE)
```

Arguments

physeq	(phyloseq, required) A phyloseq object.
...	Variables to sort by. Use desc() for descending order. Use . to refer to the phyloseq object.
clean_phyloseq_object	if TRUE (default), the resulting phyloseq object is cleaned using clean_pq() to remove empty taxa/samples.

Value

A phyloseq object with reordered taxa.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Arrange by taxonomy
arrange_taxa_pq(data_fungi, Phylum, Class)@tax_table[, "Phylum"]

# Arrange by total abundance (descending)
arrange_taxa_pq(data_fungi, dplyr::desc(taxa_sums(.)))

# order of columns matters
dfm_arr <- arrange_taxa_pq(data_fungi_mini, Class, Genus)@tax_table[, c("Class", "Genus")]
arrange_taxa_pq(data_fungi_mini, Genus, Class)@tax_table[, c("Class", "Genus")]
```

---

chimera_removal_dada2 *Remove chimeric sequences using dada2*

Description

Applies `dada2::removeBimeraDenovo()` to identify and remove chimeric sequences from a phyloseq object based on sequence abundance patterns.

Usage

```
chimera_removal_dada2(physeq, method = "consensus", return_a_list = FALSE, ...)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object with a refseq slot containing DNA sequences.
`method`	(character, default: "consensus") Method for chimera detection. Passed to `dada2::removeBimeraDenovo` Options: "consensus", "pooled", "per-sample".
`return_a_list`	(logical, default: FALSE) If TRUE, returns a list with the filtered phyloseq, kept taxa names, and chimeric taxa names.
`...`	Additional arguments passed to `dada2::removeBimeraDenovo()`.

Details

This function extracts sequences and their abundances from the phyloseq object, applies dada2's de novo chimera detection algorithm, and returns a pruned phyloseq object containing only non-chimeric sequences.

The dada2 method uses sequence abundance information to identify chimeras, assuming that chimeric sequences are less abundant than their parent sequences.

Value

If `return_a_list = FALSE` (default), returns a phyloseq object with chimeric sequences removed. If `return_a_list = TRUE`, returns a list with:

physeq The filtered phyloseq object

kept_taxa Character vector of retained taxa names

chimeric_taxa Character vector of removed chimeric taxa names

See Also

`MiscMetabar::chimera_removal_vs()` for vsearch-based chimera removal, `create_chimera_pq()` for creating test data with synthetic chimeras.

Examples

```
## Not run:
library(MiscMetabar)
data(data_fungi)

# Basic usage
data_nochim <- chimera_removal_dada2(data_fungi)

# Get detailed output
result <- chimera_removal_dada2(data_fungi, return_a_list = TRUE)
cat("Removed", length(result$chimeric_taxa), "chimeric ASVs\n")

# Use pooled method
data_nochim <- chimera_removal_dada2(data_fungi, method = "pooled")

## End(Not run)
```

---

create_chimera_pq *Create a phyloseq object with synthetic chimeric sequences*

Description

This function creates synthetic chimeric sequences by combining parts of existing sequences from a phyloseq object. Useful for benchmarking chimera detection methods like `MiscMetabar::chimera_removal_vs()` or `chimera_removal_dada2()`.

Usage

```
create_chimera_pq(
  physeq,
  n_chimeras = 5,
  prop_mean = 0.5,
  prop_sd = 0.15,
  prop_min = 0.1,
```

```
  seed = 123,
  median_abundance_multiplier = 0.1,
  ensure_distinct_parents = TRUE,
  min_parent_distance = 0.1
)
```

Arguments

physeq	(phyloseq, required) A phyloseq object with a refseq slot containing DNA sequences.
n_chimeras	(integer, default: 5) Number of chimeric sequences to create.
prop_mean	(numeric, default: 0.5) Mean of the normal distribution used to sample the proportion of the first parent sequence. A value of 0.5 means chimeras will be centered around 50/50 splits.
prop_sd	(numeric, default: 0.15) Standard deviation of the normal distribution used to sample proportions. Higher values create more variable chimera breakpoints.
prop_min	(numeric, default: 0.1) Minimum proportion threshold. Proportions below this value (or above 1 - prop_min) are resampled to ensure each parent contributes meaningfully to the chimera.
seed	(integer, default: 123) Random seed for reproducibility.
median_abundance_multiplier	(numeric, default: 0.1) Multiplier to set the abundance of chimeric sequences relative to the median abundance of existing sequences. A value of 0.1 means chimeras will have approximately 10% of the median abundance.
ensure_distinct_parents	(logical, default: TRUE) If TRUE, ensures that parent2 is sufficiently different from parent1 based on min_parent_distance. Chimeras created from very similar parent sequences may be undetectable.
min_parent_distance	(numeric, default: 0.1) Minimum sequence distance (proportion of differing positions) between parent1 and parent2. Only used when ensure_distinct_parents = TRUE. Values typically range from 0.05 (5% divergence) to 0.3 (30% divergence).

Value

A list containing:

physeq The new phyloseq object with added chimeric sequences

chimera_names Character vector of chimera taxa names

parent_info Data frame with details about each chimera: chimera name, parent1, parent2, parent_distance, prop_parent1, breakpoint, seq_length

params List of parameters used (prop_mean, prop_sd, prop_min, ensure_distinct_parents, min_parent_distance)

Author(s)

Adrien Taudiere

See Also

MiscMetabar::chimera_removal_vs(), chimera_removal_dada2()

Examples

```
## Not run:
library(MiscMetabar)
data(data_fungi)

# Default: centered around 50% with some variation
result <- create_chimera_pq(data_fungi, n_chimeras = 40)
data_fungi_test <- result$physeq
known_chimeras <- result$chimera_names

# View the parent information and proportions
print(result$parent_info)

# More variable proportions (wider distribution)
result2 <- create_chimera_pq(data_fungi, n_chimeras = 40,
                             prop_mean = 0.5, prop_sd = 0.25)

# Biased toward more of parent1 (e.g., 70/30 splits on average)
result3 <- create_chimera_pq(data_fungi, n_chimeras = 40,
                             prop_mean = 0.7, prop_sd = 0.1)

# Benchmark chimera detection methods
if (MiscMetabar::is_vsearch_installed()) {
  nochim_vs <- MiscMetabar::chimera_removal_vs(data_fungi_test)
  detected_vs <- known_chimeras[!known_chimeras %in% phyloseq::taxa_names(nochim_vs)]
  cat("vsearch detected:", length(detected_vs), "/",
      length(known_chimeras), "chimeras\n")
}

# Visualize the distribution of proportions
hist(result$parent_info$prop_parent1,
     main = "Distribution of parent1 proportions",
     xlab = "Proportion from parent1", xlim = c(0, 1))

# Ensure parents are at least 15% different (more detectable chimeras)
result4 <- create_chimera_pq(data_fungi, n_chimeras = 40,
                             min_parent_distance = 0.15)

# Disable parent distance filtering (allows similar parents)
result5 <- create_chimera_pq(data_fungi, n_chimeras = 40,
                             ensure_distinct_parents = FALSE)

## End(Not run)
```

`decontam_sam_control` *Decontaminate based on negative/blank control samples*

Description

Remove potential contaminants by setting OTU values to 0 when they are at or below the level observed in negative/blank control samples for that particular OTU. If multiple control are available, for each taxon a threshold is computed from the control samples using a summary function (default: `max`). Occurrences in non-control samples that are at or below this threshold are set to 0.

Usage

```
decontam_sam_control(
  physeq,
  control_condition,
  fun = max,
  global_threshold = FALSE,
  remove_controls = FALSE,
  clean_phyloseq_object = TRUE,
  verbose = TRUE
)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object.
`control_condition`	An expression evaluated on sample_data that returns TRUE for control samples. Use `.` to refer to the phyloseq object (e.g., `sample_type == "negative"`, `is_control == TRUE`).
`fun`	(function, default `max`) A function to summarize the control sample values for each taxon (or globally if `global_threshold = TRUE`). Common choices: `max` (most conservative, default), `mean`, `median`, or a custom function. The function should take a numeric vector and return a single value.
`global_threshold`	(logical, default FALSE) If TRUE, compute a single global threshold from all control occurrences instead of per-taxon thresholds. This applies `fun` to all values in the control samples.
`remove_controls`	(logical, default FALSE) Whether to remove the control samples from the output phyloseq object after decontamination.
`clean_phyloseq_object`	(logical, default TRUE) Whether to clean the resulting phyloseq object using `clean_pq()` to remove empty taxa/samples.
`verbose`	(logical, default TRUE) Whether to print additional information.

Value

A phyloseq object with decontaminated OTU values.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Add a mock control column for demonstration using the 3 samples with lowest
# total abundance as controls
pq <- mutate_samdata_pq(data_fungi, is_control = sample_sums(.) < sort(sample_sums(.))[3])

# Decontaminate using max of controls as threshold (per-taxon)
decontam_sam_control(pq, is_control)

# Use a global threshold (single value for all taxa)
decontam_sam_control(pq, is_control, global_threshold = TRUE)

# Use mean instead of max (less conservative)
decontam_sam_control(pq, is_control, fun = mean)

# Keep control samples in output
decontam_sam_control(pq, is_control, remove_controls = FALSE)

# Use a custom function (e.g., 2x the max)
decontam_sam_control(pq, is_control, fun = \(x) 2 * max(x))
```

---

decontam_taxa_control *Decontaminate based on control taxa*

Description

Remove potential contaminants by using known control taxa (e.g., spike-ins, synthetic sequences) to estimate background contamination levels. For each sample, a threshold is computed from the control taxa using a summary function (default: `max`). Occurrences of non-control taxa that are at or below this threshold are set to 0.

Usage

```
decontam_taxa_control(
  physeq,
  control_condition,
  fun = max,
  global_threshold = FALSE,
  remove_control_taxa = TRUE,
  clean_phyloseq_object = TRUE,
  verbose = TRUE
)
```

Arguments

physeq	(phyloseq, required) A phyloseq object.
control_condition	An expression evaluated on tax_table that returns TRUE for control taxa. Use . to refer to the phyloseq object. Examples: Genus == "Tintelnotia", Family == "Mitochondria", taxa_names(.) %in% c("ASV1", "ASV2").
fun	(function, default max) A function to summarize the control taxa values for each sample (or globally if global_threshold = TRUE). Common choices: max (most conservative, default), mean, median.
global_threshold	(logical, default FALSE) If TRUE, compute a single global threshold from all control taxa occurrences instead of per-sample thresholds.
remove_control_taxa	(logical, default TRUE) Whether to remove the control taxa from the output phyloseq object after decontamination.
clean_phyloseq_object	(logical, default TRUE) Whether to clean the resulting phyloseq object using clean_pq() to remove empty taxa/samples.
verbose	(logical, default TRUE) Whether to print additional information.

Value

A phyloseq object with decontaminated OTU values.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Using a condition on tax_table (e.g., select by Genus)
decontam_taxa_control(data_fungi, Genus == "Tintelnotia")

# Using taxa names directly
control_taxa <- phyloseq::taxa_names(data_fungi)[1:2]
decontam_taxa_control(data_fungi, taxa_names(.) %in% control_taxa)

# Use a global threshold
decontam_taxa_control(data_fungi, Genus == "Tintelnotia", global_threshold = TRUE)

# Keep control taxa in output
decontam_taxa_control(data_fungi, Genus == "Tintelnotia", remove_control_taxa = FALSE)
```

`filter_occurrences_pq` *Filter occurrences in the OTU table*

Description

Set OTU table values to 0 based on a condition. This is useful for removing singletons, low-abundance values, or other filtering operations at the cell level.

The condition is evaluated vectorized across the entire OTU matrix with access to special variables (all as matrices matching OTU dimensions):

- `.` = cell values (the OTU matrix)
- `sample_total` = sum of each sample (repeated per column)
- `taxon_total` = sum of each taxon (repeated per row)
- `sample_mean` = mean of each sample (repeated per column)
- `taxon_mean` = mean of each taxon (repeated per row)

Values that do not satisfy the condition are set to 0.

Usage

```
filter_occurrences_pq(physeq, condition, clean_phyloseq_object = TRUE)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object.
`condition`	An expression evaluated on the OTU matrix. Values where the condition is FALSE (or NA) are set to 0. Use `.` to refer to cell values.
`clean_phyloseq_object`	if TRUE (default), the resulting phyloseq object is cleaned using `clean_pq()` to remove empty taxa/samples.

Value

A phyloseq object with filtered OTU values.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Remove singletons (abundance = 1)
filter_occurrences_pq(data_fungi, . > 1)

# Keep only values above 0.01% of sample total
filter_occurrences_pq(data_fungi, . / sample_total > 0.0001)
```

```
# Keep only values above taxon mean
filter_occurrences_pq(data_fungi, . > taxon_mean)
```

`filter_samples_pq` *Filter samples in a phyloseq object*

Description

Filter samples using data masking on sample_data. Supports the `.` pronoun to refer to the phyloseq object for use with functions like `sample_sums()`.

Usage

```
filter_samples_pq(physeq, ..., clean_phyloseq_object = TRUE)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object.
`...`	Expressions that return a logical value, evaluated in the context of sample_data. Multiple conditions are combined with `&`. Use `.` to refer to the phyloseq object (e.g., `sample_sums(.) > 1000`).
`clean_phyloseq_object`	if TRUE (default), the resulting phyloseq object is cleaned using `clean_pq()` to remove empty taxa/samples.

Value

A phyloseq object with filtered samples.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Filter by sample metadata
filter_samples_pq(data_fungi, Height == "Low")

# Filter by sequencing depth
filter_samples_pq(data_fungi, sample_sums(.) > 1000)

# Combine multiple conditions
filter_samples_pq(data_fungi, Height == "Low", sample_sums(.) > 5000)

# Keep samples above median abundance
filter_samples_pq(data_fungi, sample_sums(.) > median(sample_sums(.)))
```

```
# Keep samples above half of the average abundance
filter_samples_pq(data_fungi, sample_sums(.) > sum(sample_sums(.))/phyloseq::nsamples(.)/2)
```

`filter_taxa_pq` *Filter taxa in a phyloseq object*

Description

Filter taxa using data masking on tax_table. Supports the `.` pronoun to refer to the phyloseq object for use with functions like `taxa_sums()`.

Usage

```
filter_taxa_pq(physeq, ..., clean_phyloseq_object = TRUE)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object.
`...`	Expressions that return a logical value, evaluated in the context of tax_table. Multiple conditions are combined with `&`. Use `.` to refer to the phyloseq object (e.g., `taxa_sums(.) > 100`).
`clean_phyloseq_object`	if TRUE (default), the resulting phyloseq object is cleaned using `clean_pq()` to remove empty taxa/samples.

Value

A phyloseq object with filtered taxa.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Filter by taxonomy
filter_taxa_pq(data_fungi, Phylum == "Basidiomycota")

# Filter by total abundance
filter_taxa_pq(data_fungi, taxa_sums(.) > 100)

# Combine multiple conditions
filter_taxa_pq(data_fungi, Phylum == "Basidiomycota", taxa_sums(.) > 100)

# Keep taxa above median abundance
filter_taxa_pq(data_fungi, taxa_sums(.) > median(taxa_sums(.)))
```

`filter_tree_pq` *Filter phyloseq by tree topology*

Description

Filter a phyloseq object to include only taxa that are present in the phylogenetic tree, or prune the tree to match the taxa in the phyloseq. Can also filter based on tree properties like tip labels matching a pattern.

Usage

```
filter_tree_pq(
  physeq,
  taxa = NULL,
  pattern = NULL,
  invert = FALSE,
  clean_phyloseq_object = TRUE
)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object with a phy_tree slot.
`taxa`	(character, optional) Character vector of taxa names to keep. If NULL, keeps all taxa that are present in both the OTU table and tree.
`pattern`	(character, optional) A regular expression pattern to match against tip labels. Only tips matching the pattern are kept.
`invert`	(logical, default FALSE) If TRUE and pattern is provided, keep tips that do NOT match the pattern.
`clean_phyloseq_object`	if TRUE (default), the resulting phyloseq object is cleaned using `clean_pq()` to remove empty taxa/samples.

Value

A phyloseq object with filtered tree and matching taxa.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Synchronize tree with OTU table (remove taxa not in tree)
# filter_tree_pq(physeq_with_tree)

# Keep only specific taxa in tree
```

```
# filter_tree_pq(physeq_with_tree, taxa = c("ASV1", "ASV2", "ASV3"))

# Filter by tip label pattern
# filter_tree_pq(physeq_with_tree, pattern = "^ASV")
```

`mutate_occurrences_pq` *Transform OTU table values*

Description

Apply a transformation to all values in the OTU table. This is useful for computing relative abundances, log transformations, or other value-level operations.

The expression is evaluated vectorized across the entire OTU matrix with access to special variables (all as matrices matching OTU dimensions):

- `.` = cell values (the OTU matrix)
- `sample_total` = sum of each sample (repeated per column)
- `taxon_total` = sum of each taxon (repeated per row)
- `sample_mean` = mean of each sample (repeated per column)
- `taxon_mean` = mean of each taxon (repeated per row)
- `sample_median` = median of each sample (repeated per column)
- `taxon_median` = median of each taxon (repeated per row)

Usage

```
mutate_occurrences_pq(physeq, expr)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object.
`expr`	An expression evaluated on the OTU matrix. The result replaces the original values. Use `.` to refer to cell values.

Value

A phyloseq object with transformed OTU values.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Convert to relative abundance (proportion of sample total)
mutate_occurrences_pq(data_fungi, . / sample_total)

# Log transformation (adding pseudocount)
mutate_occurrences_pq(data_fungi, log1p(.))

# Center by taxon mean
mutate_occurrences_pq(data_fungi, . - taxon_mean)
```

---

mutate_samdata_pq *Add or modify columns in sample_data*

Description

Create new columns or modify existing ones in sample_data using data masking. Supports the . pronoun to refer to the phyloseq object.

This function only modifies the sample_data slot (columns/metadata). It cannot add or remove samples. The number of samples and sample names are preserved.

Usage

```
mutate_samdata_pq(physeq, ...)
```

Arguments

physeq	(phyloseq, required) A phyloseq object.
...	Name-value pairs. The name gives the name of the column in the output. The value must be a vector of length 1 (recycled) or exactly the same length as the number of samples. Use . to refer to the phyloseq object.

Value

A phyloseq object with modified sample_data (same samples, modified or new columns).

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Add a new column based on sequencing depth
mutate_samdata_pq(data_fungi, log_depth = log(sample_sums(.)))

# Modify an existing column
mutate_samdata_pq(data_fungi, Height = toupper(Height))
```

`mutate_taxa_pq` *Add or modify columns in tax_table*

Description

Create new columns or modify existing ones in tax_table using data masking. Supports the . pronoun to refer to the phyloseq object.

This function only modifies the tax_table slot (columns/taxonomic ranks). It cannot add or remove taxa. The number of taxa and taxa names are preserved.

Usage

```
mutate_taxa_pq(physeq, ...)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object.
`...`	Name-value pairs. The name gives the name of the column in the output. The value must be a vector of length 1 (recycled) or exactly the same length as the number of taxa. Use . to refer to the phyloseq object.

Value

A phyloseq object with modified tax_table (same taxa, modified or new columns).

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Replace NA values in a column
mutate_taxa_pq(data_fungi, Genus = ifelse(is.na(Genus), "Unknown", Genus))

# Add a new column based on abundance
mutate_taxa_pq(data_fungi, total_abundance = taxa_sums(.))
```

`plot_sample_depth_pq` *Plot sample depth differences to detect outliers*

Description

Creates a diagnostic plot showing the log10 differences between consecutive sorted sample sums. This helps identify samples with unusually low sequencing depth by detecting large "jumps" in the distribution.

Usage

```
plot_sample_depth_pq(
  physeq,
  lower_quantile = 0.1,
  threshold_quantile = 0.05,
  show_threshold = TRUE
)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object.
`lower_quantile`	(numeric, default: 0.1) Lower quantile threshold to exclude smallest differences when computing statistics.
`threshold_quantile`	(numeric, default: 0.05) Quantile used to define the threshold for detecting large jumps.
`show_threshold`	(logical, default: TRUE) Whether to color points based on the computed threshold and show the threshold rank in the title.

Details

The function sorts samples by their total read count (sample sums), then computes the difference between each consecutive pair. Large differences indicate potential outlier samples with unusually low depth.

The threshold is computed as the `threshold_quantile` of differences, excluding the smallest `lower_quantile` of differences to avoid noise. Samples with rank >= the first sample exceeding this threshold are considered to have sufficient depth.

Value

A ggplot object showing:

- Points representing log10(difference) vs rank
- Solid horizontal line at the mean log10(difference)
- Dashed horizontal lines at the 5th and 95th percentiles
- If `show_threshold = TRUE`, points colored by whether they pass the threshold

Examples

```
library(MiscMetabar)
data(data_fungi)

# Basic plot
plot_sample_depth_pq(data_fungi)

# Adjust thresholds
plot_sample_depth_pq(data_fungi, lower_quantile = 0.2, threshold_quantile = 0.1)

# Without threshold coloring
plot_sample_depth_pq(data_fungi, show_threshold = FALSE)
```

---

rename_samples_pq *Rename columns in sample_data*

Description

Rename columns in sample_data using tidyselect semantics.

Usage

```
rename_samples_pq(physeq, ...)
```

Arguments

physeq	(phyloseq, required) A phyloseq object.
...	Name-value pairs where the name is the new name and the value is the old name. Use new_name = old_name syntax.

Value

A phyloseq object with renamed sample_data columns.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Rename a single column
rename_samples_pq(data_fungi, sample_height = Height)

# Rename multiple columns
rename_samples_pq(data_fungi, sample_height = Height, sample_time = Time)
```

rename_taxa_pq *Rename columns in tax_table*

Description

Rename columns (taxonomic ranks) in tax_table using tidyselect semantics.

Usage

```
rename_taxa_pq(physeq, ...)
```

Arguments

physeq	(phyloseq, required) A phyloseq object.
...	Name-value pairs where the name is the new name and the value is the old name. Use new_name = old_name syntax.

Value

A phyloseq object with renamed tax_table columns.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Rename a single rank
rename_taxa_pq(data_fungi, tax_kingdom = Kingdom)

# Rename multiple ranks
rename_taxa_pq(data_fungi, tax_kingdom = Kingdom, tax_phylum = Phylum)
```

select_samdata_pq *Select columns from sample_data in a phyloseq object*

Description

Select sample_data columns using tidyselect semantics.

Usage

```
select_samdata_pq(physeq, ...)
```

Arguments

physeq	(phyloseq, 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.

Value

A phyloseq object with selected sample_data columns.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Select specific columns
select_samdata_pq(data_fungi, Height, Time)

# Select a range of columns
select_samdata_pq(data_fungi, Height:Time)

# Exclude columns
select_samdata_pq(data_fungi, !Sample_id)
```

---

select_taxa_pq *Select columns from tax_table in a phyloseq object*

Description

Select tax_table columns (taxonomic ranks) using tidyselect semantics.

Usage

```
select_taxa_pq(physeq, ...)
```

Arguments

physeq	(phyloseq, 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 Kingdom:Genus can be used to select a range of taxonomic ranks.

Value

A phyloseq object with selected tax_table columns.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Select specific ranks
select_taxa_pq(data_fungi, Kingdom, Phylum, Class)

# Select a range of ranks
select_taxa_pq(data_fungi, Kingdom:Genus)

# Exclude ranks
select_taxa_pq(data_fungi, !Species)
```

`slice_samples_pq` *Subset samples by position*

Description

Select samples by their integer positions.

Usage

```
slice_samples_pq(physeq, ..., clean_phyloseq_object = TRUE)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object.
`...`	Integer row indices. Positive values select samples, negative values drop samples.
`clean_phyloseq_object`	if TRUE (default), the resulting phyloseq object is cleaned using `clean_pq()` to remove empty taxa/samples.

Value

A phyloseq object with selected samples.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Select first 5 samples
slice_samples_pq(data_fungi, 1:5)

# Remove first 2 samples
slice_samples_pq(data_fungi, -(1:2))
```

`slice_taxa_pq` *Subset taxa by position*

Description

Select taxa by their integer positions.

Usage

```
slice_taxa_pq(physeq, ..., clean_phyloseq_object = TRUE)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object.
`...`	Integer row indices. Positive values select taxa, negative values drop taxa.
`clean_phyloseq_object`	if TRUE (default), the resulting phyloseq object is cleaned using `clean_pq()` to remove empty taxa/samples.

Value

A phyloseq object with selected taxa.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Select first 10 taxa
slice_taxa_pq(data_fungi, 1:10)

# Remove first 5 taxa
slice_taxa_pq(data_fungi, -(1:5))
```

`taxa_prevalence` *Calculate taxa prevalence*

Description

Calculate the prevalence (number of samples in which a taxon is present) for each taxon in a phyloseq object.

Usage

```
taxa_prevalence(physeq, threshold = 0)
```

Arguments

`physeq`	(phyloseq, required) A phyloseq object.
`threshold`	(numeric, default 0) Minimum abundance to consider a taxon as present in a sample. Values > threshold are counted as present.

Value

A named numeric vector (class integer) with prevalence for each taxon.

Author(s)

Adrien Taudière

Examples

```
library(MiscMetabar)
# Get prevalence for each taxon
prev <- taxa_prevalence(data_fungi)
head(prev)

# Get prevalence with minimum abundance of 10
prev10 <- taxa_prevalence(data_fungi, threshold = 10)
head(prev10)

# Use in filter_taxa_pq
# Keep taxa present in at least 5 samples
filter_taxa_pq(data_fungi, taxa_prevalence(.) >= 5)
```

Index