

Package: tidypq (via r-universe)

October 16, 2026

Title Tidiverse-Style Verbs for 'phyloseq' Objects

Version 0.3.0.9000

Description Tidiverse-style verbs for manipulating 'phyloseq' objects at four scales: samples, taxa, occurrences, and tree. Functions follow a consistent naming convention and support data masking for intuitive filtering, selection, and mutation operations. Designed for metabarcoding analyses, it integrates with 'MiscMetabar' and the 'pqverse' ecosystem.

License AGPL-3

URL <https://adrientaudiere.github.io/tidypq/>,
<https://github.com/adrientaudiere/tidypq>

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

Depends R (>= 4.1.0), phyloseq, MiscMetabar

Imports ape, cli, dplyr (>= 1.0.0), ggplot2, lifecycle, methods, rlang
(>= 1.0.0), tibble, tidyr, tidyselect, utils

Suggests Biostrings, dada2, knitr, patchwork, rmarkdown, testthat (>= 3.0.0), vegan

Remotes adrientaudiere/MiscMetabar

Config/testthat/edition 3

Encoding UTF-8

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

RdMacros lifecycle

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake 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 2026-09-16 15:44:01 UTC

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

RemoteRef HEAD

RemoteSha 9fe55631b13744b4409264d01343cc073cb4e856

Contents

arrange_samples_pq	2
arrange_taxa_pq	3
autoplot.contam_tbl	4
create_chimera_pq	5
decontam_control_samples_pq	7
decontam_control_taxa_pq	9
filter_chimera_pq	10
filter_contam_pq	11
filter_occurrences_pq	12
filter_samples_pq	13
filter_taxa_pq	14
filter_tree_pq	15
identify_contam_blocklist_pq	16
identify_contam_chimera_pq	18
identify_contam_corr_pq	19
identify_contam_negcontrol_pq	21
identify_contam_primer_pq	22
known_contaminant_genera	23
merge_markers_pq	24
mutate_occurrences_pq	26
mutate_samdata_pq	28
mutate_taxa_pq	29
plot_uncross2_scores	30
pq_to_tidy	31
print.contam_tbl	33
rbind.contam_tbl	33
rename_samples_pq	34
rename_taxa_pq	35
select_samdata_pq	35
select_taxa_pq	36
slice_samples_pq	37
slice_taxa_pq	38
tax_table_to_df	39
taxa_prevalence	40
uncross2_pq	41
Index	44

arrange_samples_pq	<i>Arrange samples by column values</i>
--------------------	---

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	<i>Arrange taxa by column values</i>
-----------------	--------------------------------------

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, Class, Genus)@tax_table[, c("Class", "Genus")]
arrange_taxa_pq(data_fungi, Genus, Class)@tax_table[, c("Class", "Genus")]
```

autoplot.contam_tbl	<i>Diagnostic plot for a contam_tbl</i>
---------------------	---

Description

A lightweight overview of which detectors flagged how many taxa. For publication-grade contamination figures, richer plotting lives in the ggplotpq package and consumes the same contam_tbl.

Usage

```
## S3 method for class 'contam_tbl'
autoplot(object, ...)

## S3 method for class 'contam_tbl'
plot(x, ...)
```

Arguments

object, x	(contam_tbl, required) A detector output.
...	Currently ignored.

Value

A ggplot object (autoplot) or x invisibly (plot).

Author(s)

Adrien Taudière

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 `identify_contam_chimera_pq()`.

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,
  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.
min_parent_distance	(numeric, default: 0.1) Minimum sequence distance (proportion of differing positions) between parent1 and parent2. If 0, chimeras can be created from very similar parents, which may be harder to detect. In some cases, with min_parent_distance = 0, you may end up with chimeras that are identical to one of the parents.

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, min_parent_distance)

Author(s)

Adrien Taudiere

See Also

[MiscMetabar::chimera_removal_vs\(\)](#), [identify_contam_chimera_pq\(\)](#)

Examples

```
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,
                             min_parent_distance = 0
)
```

decontam_control_samples_pq

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_control_samples_pq(
  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_control_samples_pq(pq, is_control)

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

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

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

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

decontam_control_taxa_pq

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_control_taxa_pq(
  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 == "Tintelnolia", 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_control_taxa_pq(data_fungi, Genus == "Tintelnotia")

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

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

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

filter_chimera_pq	<i>Remove chimeric taxa in one step</i>
-------------------	---

Description

A convenience wrapper that runs `identify_contam_chimera_pq()` and then `filter_contam_pq()`, returning the chimera-free phyloseq object directly. Chimera removal is a near-ubiquitous cleaning step, so it gets a dedicated verb; the other contaminant methods are composed explicitly via their `identify_contam*_pq()` detector and `filter_contam_pq()`.

Usage

```
filter_chimera_pq(
  physeq,
  method = c("dada2", "vsearch_ref"),
  clean_phyloseq_object = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

physeq	(phyloseq, required) A phyloseq object with a populated @refseq slot containing DNA sequences.
method	(character, default "dada2") Detection engine, passed to <code>identify_contam_chimera_pq()</code> : "dada2" (de novo) or "vsearch_ref" (reference-based UCHIME, requires database).
clean_phyloseq_object	(logical, default TRUE) Whether to clean the result with <code>MiscMetabar::clean_pq()</code> .

verbose (logical, default TRUE) Whether to report detection and removal.
 ... Additional arguments passed to `identify_contam_chimera_pq()` (e.g. `dada2_method`, `database`, `vsearchpath`, `vsearch_args`).

Value

A phyloseq object with chimeric taxa removed (unchanged when none are detected).

Author(s)

Adrien Taudière

See Also

`identify_contam_chimera_pq()` to inspect chimeras before removal, `filter_contam_pq()` for the general removal verb.

Examples

```
library(MiscMetabar)
data(data_fungi)

data_nochim <- filter_chimera_pq(data_fungi, method = "dada2")
```

filter_contam_pq	<i>Remove contaminant taxa flagged by a detector</i>
------------------	--

Description

The single removal verb for contamination handling. It consumes any `contam_tbl` produced by an `identify_contam*_pq()` detector (or several, combined with `rbind()`) and prunes the flagged taxa from physeq by name. When the `contam_tbl` is empty, physeq is returned unchanged.

Usage

```
filter_contam_pq(physeq, contam, clean_phyloseq_object = TRUE, verbose = TRUE)
```

Arguments

physeq (phyloseq, required) A phyloseq object.
 contam (contam_tbl, required) Output of an `identify_contam*_pq()` detector. Removal is keyed on the taxon column, so taxa are matched by name rather than position.
 clean_phyloseq_object (logical, default TRUE) Whether to clean the resulting object with `MiscMetabar::clean_pq()` to drop emptied taxa/samples.
 verbose (logical, default TRUE) Whether to report how many taxa were removed.

Value

A phyloseq object with the flagged taxa removed.

Author(s)

Adrien Taudière

See Also

`identify_contam_corr_pq()` and the other `identify_contam*_pq()` detectors that produce a `contam_tbl`.

Examples

```
library(MiscMetabar)

flagged <- identify_contam_corr_pq(data_fungi, verbose = FALSE)
data_clean <- filter_contam_pq(data_fungi, flagged)
```

`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 (required) 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	<i>Filter samples in a phyloseq object</i>
-------------------	--

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	<i>Filter taxa in a phyloseq object</i>
----------------	---

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	<i>Filter phyloseq by tree topology</i>
----------------	---

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)
library(ape)
data(data_fungi)
pg <- data_fungi
pg@phy_tree <- ape::rtree(phyloseq::ntaxa(data_fungi),
                        tip.label = phyloseq::taxa_names(data_fungi))
filter_tree_pq(pg, taxa = phyloseq::taxa_names(data_fungi)[1:10])
```

identify_contam_blocklist_pq

Detect taxa whose genus belongs to a known-contaminant blocklist

Description

Flags taxa whose genus matches a blocklist of known reagent and laboratory contaminants. A taxonomy-only method requiring no negative controls, best used as a complementary, curation-oriented filter – review flagged taxa before removal, as several blocklist genera also occur as genuine environmental taxa.

This is a pure detector: it flags taxa and returns a [contam_tbl](#) without modifying physeq. Use [filter_contam_pq\(\)](#) to remove the flagged taxa.

Usage

```
identify_contam_blocklist_pq(
  physeq,
  genus_blocklist = known_contaminant_genera(),
  tax_rank = "Genus",
  extra_genera = NULL,
```

```
ignore_case = TRUE,  
verbose = TRUE  
)
```

Arguments

physeq	(phyloseq, required) A phyloseq object with a tax_table.
genus_blocklist	(character, default <code>known_contaminant_genera()</code>) Vector of genus names to flag.
tax_rank	(character, default "Genus") Name of the tax_table column holding genus-level assignments.
extra_genera	(character, default NULL) Additional genus names to append to genus_blocklist.
ignore_case	(logical, default TRUE) If TRUE, matching is case-insensitive.
verbose	(logical, default TRUE) Print a one-line detection summary.

Value

A `contam_tbl` with one row per flagged taxon, method = "blocklist", and the evidence columns genus, total_reads, and prevalence. Empty when no genus matches the blocklist.

Author(s)

Adrien Taudière

See Also

`known_contaminant_genera()` for the default blocklist, `filter_contam_pq()` to remove flagged taxa.

Examples

```
library(MiscMetabar)  
data(data_fungi)  
  
# data_fungi is fungal, so the default bacterial blocklist matches nothing;  
# extend it with a genus present in the data for illustration.  
identify_contam_blocklist_pq(data_fungi, extra_genera = "Mortierella")
```

identify_contam_chimera_pq

Detect chimeric taxa with dada2 or vsearch

Description

Flags chimeric taxa using either `dada2::removeBimeraDenovo()` de novo detection (method = "dada2") or UCHIME reference-based detection via **vsearch** (method = "vsearch_ref").

This is a pure detector: it flags taxa and returns a `contam_tbl` without modifying physeq. Use `filter_contam_pq()` to remove the flagged taxa.

Usage

```
identify_contam_chimera_pq(
  physeq,
  method = c("dada2", "vsearch_ref"),
  dada2_method = "consensus",
  database = NULL,
  vsearchpath = "vsearch",
  vsearch_args = "--fasta_width 0",
  keep_temporary_files = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

physeq	(phyloseq, required) A phyloseq object with a populated @refseq slot containing DNA sequences.
method	(character, default "dada2") Detection engine: "dada2" for de novo abundance-based detection, or "vsearch_ref" for reference-based UCHIME detection (requires database).
dada2_method	(character, default "consensus") Passed as method to <code>dada2::removeBimeraDenovo()</code> when method = "dada2". One of "consensus", "pooled", "per-sample".
database	(character, default NULL) Path to the reference FASTA database used by vsearch. Required when method = "vsearch_ref".
vsearchpath	(character, default "vsearch") Path to the vsearch executable.
vsearch_args	(character, default "--fasta_width 0") Extra arguments passed verbatim to vsearch.
keep_temporary_files	(logical, default FALSE) Keep the temporary FASTA files written to tempdir() (vsearch only).
verbose	(logical, default TRUE) Print a one-line detection summary.
...	Additional arguments passed to <code>dada2::removeBimeraDenovo()</code> .

Value

A `contam_tbl` with one row per chimeric taxon, `method = "chimera"`, and the evidence columns `chimera_method` ("dada2" or "vsearch_ref") and `n_reads`. Empty when no chimera is detected.

Author(s)

Adrien Taudière

See Also

`filter_contam_pq()` to remove flagged taxa, `create_chimera_pq()` to generate synthetic chimeras for benchmarking.

Examples

```
library(MiscMetabar)
data(data_fungi)

flagged <- identify_contam_chimera_pq(data_fungi, method = "dada2")
flagged
```

```
identify_contam_corr_pq
```

Detect contaminants by correlating taxon reads with total sample depth

Description

Flags potential contaminant taxa from the correlation between each taxon's relative abundance and total sample depth. Contaminants, introduced at a roughly constant level during processing, show a **negative** correlation (their proportion is inflated in low-biomass samples). Inspired by the GRIMER tool.

This is a pure detector: it flags taxa and returns a `contam_tbl` without modifying `physeq`. Use `filter_contam_pq()` to remove the flagged taxa and `plot.contam_tbl()` for a quick diagnostic.

Usage

```
identify_contam_corr_pq(
  physeq,
  min_total_reads = 1000,
  min_prevalence = 2,
  contam_threshold = -0.5,
  verbose = TRUE
)
```

Arguments

phyloseq (phyloseq, required) A phyloseq object.
min_total_reads (numeric, default 1000) Samples with fewer total reads are excluded to avoid spurious correlations from low-depth outliers.
min_prevalence (integer, default 2) Taxa present in fewer than this many samples are excluded from the analysis.
contam_threshold (numeric, default -0.5) Taxa whose relative- abundance correlation (**cor_rel**) is below this value are flagged. Use values closer to 0 to flag more aggressively.
verbose (logical, default TRUE) Print a one-line detection summary.

Value

A [contam_tbl](#) with one row per flagged taxon, method = "corr", and the evidence columns **total_reads**, **prevalence**, **cor_abs**, **cor_rel**, **slope**, and **p_value**. Empty when nothing is flagged.

Author(s)

Adrien Taudière

See Also

[filter_contam_pq\(\)](#) to remove flagged taxa, [contam_tbl](#) for the return type.

Examples

```

library(MiscMetabar)

# Inject a constant-reads fake contaminant so the example flags something
otu <- as(phyloseq::otu_table(data_fungi), "matrix")
fake <- matrix(rep(50, nrow(otu)), ncol = 1,
  dimnames = list(rownames(otu), "fake_contaminant"))
tax <- rbind(
  as.matrix(phyloseq::tax_table(data_fungi)),
  fake_contaminant = rep("Contaminant", ncol(phyloseq::tax_table(data_fungi)))
)
data_fungi_c <- phyloseq::phyloseq(
  phyloseq::otu_table(cbind(otu, fake), taxa_are_rows = FALSE),
  phyloseq::tax_table(tax),
  phyloseq::sample_data(data_fungi)
)

flagged <- identify_contam_corr_pq(data_fungi_c)
flagged

```

identify_contam_negcontrol_pq

Detect contaminants from negative-control occurrence patterns

Description

Examines taxa detected in negative-control samples and assigns each a contamination sub-type from read abundance and occurrence patterns:

- **artifact:** very low total reads AND present in very few samples – likely sequencing noise or cross-talk.
- **lab_contaminant:** predominantly found in negative controls relative to real samples – likely introduced during library preparation.
- **sample_contaminant:** detected in controls but also widespread in real samples – may represent genuine biological signal.

This is a pure detector: it flags taxa and returns a `contam_tbl` without modifying `physeq`. Because `sample_contaminant` may be genuine signal, only `artifact` and `lab_contaminant` are flagged by default (see `flag_categories`). Use `filter_contam_pq()` to remove the flagged taxa.

Usage

```
identify_contam_negcontrol_pq(
  physeq,
  neg_control,
  min_reads_artifact = 10,
  max_samples_artifact = 2,
  max_ratio_lab_contam = 0.2,
  min_neg_samples_lab = 2,
  flag_categories = c("artifact", "lab_contaminant"),
  verbose = TRUE
)
```

Arguments

<code>physeq</code>	(phyloseq, required) A phyloseq object.
<code>neg_control</code>	An expression evaluated on <code>sample_data</code> returning TRUE for negative-control samples (e.g. <code>is_control == TRUE</code>). Use <code>.</code> to refer to the phyloseq object.
<code>min_reads_artifact</code>	(numeric, default 10) Maximum total reads for a taxon to be considered an artifact.
<code>max_samples_artifact</code>	(integer, default 2) Maximum total number of samples in which an artifact taxon may appear.
<code>max_ratio_lab_contam</code>	(numeric, default 0.2) Maximum ratio of non-control to control sample occurrences for the lab-contaminant category.

`min_neg_samples_lab`
 (integer, default 2) Minimum number of control samples a lab-contaminant taxon must appear in.

`flag_categories`
 (character, default `c("artifact", "lab_contaminant")`) Which sub-types to include in the returned `contam_tbl`. Add `"sample_contaminant"` to flag those as well.

`verbose`
 (logical, default TRUE) Print a one-line detection summary.

Value

A `contam_tbl` with one row per flagged taxon, `method = "negcontrol"`, and the evidence columns `subtype` (artifact, lab_contaminant, or sample_contaminant), `total_reads`, `reads_in_neg`, `reads_in_samples`, `n_neg_samples`, `n_non_neg_samples`, and `ratio_non_neg_to_neg`. Empty when no eligible taxon is found.

Author(s)

Adrien Taudière

See Also

`filter_contam_pq()` to remove flagged taxa.

Examples

```
library(MiscMetabar)
data(data_fungi)

pq <- mutate_samdata_pq(
  data_fungi,
  is_control = sample_sums(.) < sort(sample_sums(.))[4]
)
flagged <- identify_contam_negcontrol_pq(pq, is_control)
flagged
```

identify_contam_primer_pq

Detect taxa whose representative sequence contains a primer

Description

For each taxon, checks whether the representative sequence in the `@refseq` slot matches any primer in primers (forward or reverse complement) using IUPAC-aware pattern matching. Taxa embedding a primer are likely chimeric or otherwise artefactual.

This is a pure detector: it flags taxa and returns a `contam_tbl` without modifying physeq. Use `filter_contam_pq()` to remove the flagged taxa.

Usage

```
identify_contam_primer_pq(physeq, primers, verbose = TRUE)
```

Arguments

physeq	(phyloseq, required) A phyloseq object with a populated @refseq slot containing DNA sequences.
primers	(named character vector, required) Primer sequences to search for. IUPAC ambiguity codes are supported (e.g. M, R, Y, N). Names are reported in the matched_primers evidence column.
verbose	(logical, default TRUE) Print a one-line detection summary.

Value

A `contam_tbl` with one row per flagged taxon, `method = "primer"`, and the evidence columns `matched_primers` (comma-separated primer names) and `n_reads`. Empty when no taxon embeds a primer.

Author(s)

Adrien Taudière

See Also

`filter_contam_pq()` to remove flagged taxa, `MiscMetabar::cutadapt_remove_primers()` for cutadapt-based primer removal of fastq files.

Examples

```
library(MiscMetabar)
# Requires a phyloseq object with a populated @refseq slot.
primers <- c(
  fwd = "CCCTACGGGGTGCASCAG",
  rev = "GGACTACVSGGTATCTAAT"
)
identify_contam_primer_pq(data_fungi, primers)
```

known_contaminant_genera

Known reagent- and laboratory-contaminant genera

Description

A curated vector of bacterial genera repeatedly reported as reagent, kit and laboratory contaminants in low-biomass sequencing studies. The list is compiled from Salter et al. (2014), the micRoclean workflow and the GRIMER `contaminants.yml` / `human-related.yml` resources. It is intended as a starting blocklist for `identify_contam_blocklist_pq()`; users should adapt it to their own substrate and primers.

Usage

```
known_contaminant_genera()
```

Value

A character vector of genus names.

Author(s)

Adrien Taudière

References

Salter SJ, Cox MJ, Turek EM, et al. (2014) Reagent and laboratory contamination can critically impact sequence-based microbiome analyses. BMC Biology 12:87. doi:[10.1186/s129150140087z](https://doi.org/10.1186/s129150140087z)

Examples

```
head(known_contaminant_genera())
length(known_contaminant_genera())
```

merge_markers_pq

Merge phyloseq objects from different markers into a single object

Description

Combine several phyloseq objects that describe the **same set of samples** sequenced with **different markers** (e.g. 16S and ITS, or ITS1 and ITS2) into a single phyloseq object. The result is assembled slot by slot with full control (it does *not* rely on `phyloseq::merge_phyloseq()`, which merges phyloseq *components* rather than whole objects):

- **Samples** are combined with a *full join* of the `sam_data` slots: the output keeps the union of all sample names and the union of all metadata columns. A sample shared by two markers keeps a single row; for a shared column, the first non-missing value across the input objects is used.
- **Taxa are never merged.** Taxa names are disambiguated by prefixing them with the marker name (see `disambiguate_taxa`), so a taxon named "ASV_1" in two markers becomes e.g. "16S_ASV_1" and "ITS_ASV_1" and stays as two distinct taxa. The output `otu_table` is therefore the block-diagonal union of the inputs.
- A **new tax_table column** (named by `marker_column`) records the marker of origin for every taxon.
- An **occurrence** of a taxon in a sample that was not sequenced with that taxon's marker is set to 0 (the taxon was simply not observable there).
- `refseq` slots are concatenated when *every* input object has one; otherwise `refseq` is dropped with a warning. Phylogenetic trees cannot be meaningfully merged across markers and are dropped with a warning.

Usage

```
merge_markers_pq(
  listpq,
  marker_names = names(listpq),
  marker_column = "marker",
  sep = "_",
  disambiguate_taxa = TRUE
)
```

Arguments

<code>listpq</code>	(named list of phyloseq, required) A list of at least two phyloseq objects, one per marker. If the list is named, the names are used as marker names (see <code>marker_names</code>).
<code>marker_names</code>	(character, default <code>names(listpq)</code>) Marker name for each object, in the same order as <code>listpq</code> . Must be unique and have one entry per object. Required when <code>listpq</code> has no names.
<code>marker_column</code>	(character of length 1, default "marker") Name of the new <code>tax_table</code> column that stores the marker of origin.
<code>sep</code>	(character of length 1, default "_") Separator inserted between the marker name and the original taxon name when disambiguating.
<code>disambiguate_taxa</code>	(logical, default TRUE) If TRUE, every taxon name is prefixed with its marker name to guarantee that no two taxa from different objects can ever be merged. If FALSE, the original taxa names are kept and the function errors when names collide across objects.

Value

A single phyloseq object stacking all input markers (union of samples, disambiguated union of taxa, marker column in `tax_table`).

Do NOT post-cluster the result

The taxa carried by different markers are intentionally kept apart. You must **not** run any post-clustering step (e.g. `MiscMetabar::postcluster_pq()`) on the merged object, because that would collapse taxa coming from different markers into a single taxon, mixing incomparable sequences (a 16S OTU and an ITS OTU are not the same organism just because their sequences happen to be similar). Keep the markers separate for any per-marker analysis and treat the merged object as a *stacked* view for cross-marker exploration.

The only legitimate cross-marker merging is at the **taxonomy** level and must be done explicitly on the `tax_table` (e.g. aggregating ITS1 and ITS2 taxa that resolve to the same species with `MiscMetabar::merge_taxa_vec()` or `phyloseq::tax_glom()`), never through sequence-based clustering.

Author(s)

Adrien Taudière

Examples

```
library(phyloseq)

# Two tiny fake markers describing the same three samples -----
make_marker <- function(taxa, ranks) {
  otu <- matrix(
    sample(0:20, length(taxa) * 3, replace = TRUE),
    nrow = length(taxa),
    dimnames = list(taxa, c("samp1", "samp2", "samp3"))
  )
  tax <- matrix(
    paste(rep(ranks, each = length(taxa)), seq_along(taxa), sep = "_"),
    nrow = length(taxa),
    dimnames = list(taxa, ranks)
  )
  sam <- data.frame(
    row.names = c("samp1", "samp2", "samp3"),
    site = c("A", "A", "B"),
    depth = c(10, 20, 30)
  )
  phyloseq(
    otu_table(otu, taxa_are_rows = TRUE),
    tax_table(tax),
    sample_data(sam)
  )
}

pq_16S <- make_marker(c("ASV_1", "ASV_2"), c("Kingdom", "Phylum"))
# Note the SAME taxon name "ASV_1" reused in the ITS object:
pq_ITS <- make_marker(c("ASV_1", "ASV_3"), c("Kingdom", "Phylum", "Genus"))

merged <- merge_markers_pq(list("marker_16S" = pq_16S, "marker_ITS" = pq_ITS))

# Taxa are disambiguated and never merged: 4 taxa, not 3.
taxa_names(merged)
# The marker of origin is recorded in a new tax_table column.
tax_table(merged)[, "marker"]

# Providing marker names explicitly for an unnamed list:
merge_markers_pq(list(pq_16S, pq_ITS), marker_names = c("16S", "ITS"))

# WARNING: never post-cluster `merged` (e.g. MiscMetabar::postcluster_pq()).
# That would wrongly merge a 16S taxon with an ITS taxon.
```

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

<code>physeq</code>	(phyloseq, required) A phyloseq object.
<code>expr</code>	An expression evaluated on the OTU matrix. The result replaces the original values. Use <code>.</code> 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	<i>Add or modify columns in sample_data</i>
-------------------	---

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.

Unlike `dplyr::mutate()`, columns created in the same call cannot reference each other (e.g., `mutate_samdata_pq(pq, a = 1, b = a + 1)` will not work because `a` is not yet available when `b` is evaluated).

Usage

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

Arguments

<code>physeq</code>	(<code>phyloseq</code> , required) A <code>phyloseq</code> object.
<code>...</code>	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 <code>.</code> to refer to the <code>phyloseq</code> 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.

Unlike `dplyr::mutate()`, columns created in the same call cannot reference each other (e.g., `mutate_taxa_pq(pq, a = 1, b = a + 1)` will not work because `a` is not yet available when `b` is evaluated).

Usage

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

Arguments

<code>physeq</code>	(<code>phyloseq</code> , required) A <code>phyloseq</code> object.
<code>...</code>	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 <code>.</code> to refer to the <code>phyloseq</code> 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_uncross2_scores *Plot UNCROSS2 cross-talk scores against OTU abundance*

Description

Diagnostic scatter plot for the output of `uncross2_pq()` run with `return_scores = TRUE`. Each point is one non-zero occurrence (an OTU in a sample), placed by the OTU's total abundance across the object (x) and the read count of that occurrence (y), and coloured by its UNCROSS2 score. Occurrences scoring at or below `tmin` (kept as genuine signal) are drawn in grey; those at or above `tmin` (flagged as cross-talk) are coloured on the score scale. The grey 1:1 line marks where an occurrence equals the OTU total: genuine "source" samples sit near it, cross-talk sits far below. Both axes are on a log10 scale.

Usage

```
plot_uncross2_scores(scores_list, grey_non_significant = TRUE)
```

Arguments

`scores_list` (list, required) The list returned by `uncross2_pq()` with `return_scores = TRUE`. Must contain `old_physeq` (the input object), the scores matrix, and the `tmin` threshold.

`grey_non_significant`
 (logical, default TRUE) Draw occurrences scoring at or below `tmin` in grey rather than on the colour scale.

Value

A `ggplot2::ggplot` object.

Author(s)

Adrien Taudière

See Also

`uncross2_pq()`, which produces the scored list.

Examples

```
library(MiscMetabar)
data(data_fungi)

res <- uncross2_pq(data_fungi, return_scores = TRUE)
plot_uncross2_scores(res)
```

pq_to_tidy

*Convert a phyloseq object to a tidy tibble***Description**

pq_to_tidy() is the canonical phyloseq-to-tidy data-preparation module for the pqverse. It owns the full pipeline: cleaning, flattening, optional sample aggregation, optional abundance transformation, and fact/bifactor resolution. Every ggplotpq plot function calls pq_to_tidy() instead of reinventing its own data-prep pipeline.

The function builds the tidy tibble from scratch using tidyr::pivot_longer() (it does not wrap phyloseq::psmelt()). Taxonomic rank grouping happens in tibble space (no tax_glom()). Sample aggregation happens in tibble space (no merge_samples() round-trip).

Alternative transform approaches: users may also transform the phyloseq object before calling pq_to_tidy() via MiscMetabar::transform_pq(), or transform the tibble after via tidyverse mutate() (e.g. dplyr::mutate(abundance = abundance / sum(abundance))).

Usage

```
pq_to_tidy(
  physeq,
  fact = NULL,
  bifactor = NULL,
  merge_sample_by = NULL,
  transform = NULL,
  ranks = NULL,
  filter_zero = TRUE,
  verbose = TRUE
)
```

Arguments

physeq	(phyloseq, required) A phyloseq object. Must have sample_data.
fact	(character or NULL, default NULL) Name of a sample_data column used as the primary grouping factor. NULL defaults to sample_id.
bifactor	(character or NULL, default NULL) Name of a sample_data column used as the secondary grouping factor. Must have exactly 2 levels. NULL defaults to sample_id.
merge_sample_by	(character or NULL, default NULL) Name of a sample_data column used to aggregate (merge) samples. Abundances are summed per group; other columns take their first value. sample_id becomes the group label.
transform	(string, function, or NULL, default NULL) Abundance transformation applied per sample_id after aggregation. A string is resolved via match.fun() (e.g. "log1p", "sqrt"). A function is applied directly. NULL leaves raw counts.

ranks	(character or NULL, default NULL) Taxonomic ranks to include as columns. NULL includes all ranks from <code>tax_table</code> . A character vector (e.g. <code>c("Phylum", "Genus")</code>) selects specific ranks. Ignored if no <code>tax_table</code> .
filter_zero	(logical, default TRUE) Drop rows where abundance == 0.
verbose	(logical, default TRUE) Print diagnostic messages.

Value

An ungrouped tibble with columns:

- `sample_id` (character) — sample identifier or merge group label.
- `taxon_id` (character) — OTU/ASV identifier.
- `abundance` (numeric) — counts or transformed values.
- `abundance_raw` (numeric) — pre-transform counts, present only when transform is applied.
- `fact` (character) — resolved primary grouping column.
- `bifactor` (character) — resolved secondary grouping column.
- Rank columns (character) — one per requested rank, NA replaced by "Unknown". Present only if `tax_table` exists.
- All other `sample_data` columns (original names and types).

Author(s)

Adrien Taudière

See Also

`phyloseq::psmelt()`, `MiscMetabar::psmelt_samples_pq()`, `MiscMetabar::clean_pq()`, `MiscMetabar::transform_pq()`

Examples

```
library(MiscMetabar)
# Basic tidy conversion
tidy_df <- pq_to_tidy(data_fungi_mini)
head(tidy_df)

# With fact and bifactor
tidy_df <- pq_to_tidy(data_fungi_mini, fact = "Height", bifactor = "Time")

# With sample aggregation
tidy_df <- pq_to_tidy(data_fungi_mini, merge_sample_by = "Height")

# With log1p transform
tidy_df <- pq_to_tidy(data_fungi_mini, transform = "log1p")

# With selected ranks only
tidy_df <- pq_to_tidy(data_fungi_mini, ranks = c("Phylum", "Genus"))
```

print.contam_tbl	<i>Print a contam_tbl</i>
------------------	---------------------------

Description

Print a contam_tbl

Usage

```
## S3 method for class 'contam_tbl'
print(x, ...)
```

Arguments

x	(contam_tbl, required) The object to print.
...	Passed to the tibble print method.

Value

x, invisibly.

Author(s)

Adrien Taudière

rbind.contam_tbl	<i>Combine several contam_tbl objects</i>
------------------	---

Description

Row-binds detector outputs into a single contam_tbl. Evidence columns missing from one detector are filled with NA.

Usage

```
## S3 method for class 'contam_tbl'
rbind(..., deparse.level = 1)
```

Arguments

...	(contam_tbl) One or more contam_tbl objects.
deparse.level	Ignored; present for rbind() compatibility.

Value

A contam_tbl holding the union of the flagged taxa.

Author(s)

Adrien Taudière

rename_samples_pq	<i>Rename columns in sample_data</i>
-------------------	--------------------------------------

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	<i>Rename columns in tax_table</i>
----------------	------------------------------------

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_domain = Domain)

# Rename multiple ranks
rename_taxa_pq(data_fungi, tax_phylum = Phylum, tax_class = Class)
```

select_samdata_pq	<i>Select columns from sample_data in a phyloseq object</i>
-------------------	---

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	<i>Select columns from tax_table in a phyloseq object</i>
----------------	---

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 Phylum: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, Phylum, Class)

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

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

slice_samples_pq	<i>Subset samples by position</i>
------------------	-----------------------------------

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	<i>Subset taxa by position</i>
---------------	--------------------------------

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))
```

tax_table_to_df	<i>Convert a tax_table into a data.frame with numeric and list columns</i>
-----------------	--

Description

The `tax_table` slot of a `phyloseq` object is stored as a **character** matrix, so numeric information (e.g. a bootstrap support, a trait value) is silently coerced to text and list-valued cells cannot be represented at all. This helper lifts those limitations by returning a tibble in which columns that look numeric (or logical) are converted to their natural type, and selected columns can be split into genuine list columns.

It does **not** modify the `phyloseq` object; it produces a stand-alone tibble for downstream (tidyverse) manipulation.

Usage

```
tax_table_to_df(
  physeq,
  convert = TRUE,
  split = NULL,
  sep = "/",
  taxa_names_col = "taxon"
)
```

Arguments

<code>physeq</code>	(<code>phyloseq</code> or <code>taxonomyTable</code> , required) A <code>phyloseq</code> object with a <code>tax_table</code> slot, or a <code>tax_table</code> object directly.
<code>convert</code>	(logical, default <code>TRUE</code>) If <code>TRUE</code> , each column is passed through <code>utils::type.convert()</code> (with <code>as.is = TRUE</code>) so that columns holding only numbers become numeric/integer and <code>"TRUE"/"FALSE"</code> columns become logical. Columns with any non-numeric value are kept as character.
<code>split</code>	(character, optional) Names of columns to turn into list columns by splitting each cell on <code>sep</code> . Useful when a rank cell packs several values (e.g. multiple GBIF matches).
<code>sep</code>	(character, default <code>"/"</code>) Separator used to split the columns named in <code>split</code> . Passed to <code>base::strsplit()</code> with <code>fixed = TRUE</code> .
<code>taxa_names_col</code>	(character or <code>NULL</code> , default <code>"taxon"</code>) Name of the leading column holding the taxa names. If <code>NULL</code> , taxa names are kept as row names instead of a column.

Value

A `tibble::tibble` with one row per taxon and one column per taxonomic rank, plus (unless `taxa_names_col` is `NULL`) a leading column of taxa names.

Author(s)

Adrien Taudière

See Also

`phyloseq::tax_table()`

Examples

```
data_fungi_mini <-
  tidypq::mutate_taxa_pq(data_fungi_mini,
    Mol_Abundance = taxa_sums(.))

# Numeric-looking ranks are converted to numeric
df <- tax_table_to_df(data_fungi_mini)
df |>
  group_by(Class) |>
  summarize(mean=mean(Mol_Abundance))

## Not run:
# Split a packed column into a list column
tax_table_to_df(data_fungi_mini, split = "Genus", sep = "_")

## End(Not run)
```

taxa_prevalence	<i>Calculate taxa prevalence</i>
-----------------	----------------------------------

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)
```

uncross2_pq

Remove cross-talk (tag-jumps) using the UNCROSS2 score

Description

Detect and remove per-cell cross-talk (also called tag-jumps) in an OTU table using Edgar's UNCROSS2 score (Edgar 2018, [doi:10.1101/400762](https://doi.org/10.1101/400762)). Unlike tail-trimming approaches (e.g. `metabar::tagjumpslayer()`), the score scales the expected leak by each OTU's **total** abundance: an abundant genuine OTU is expected to leak more reads into wells where it is absent than a rare one.

For every OTU (row) an expected per-sample cross-talk count is computed as $z = f * N / n$, where N is the OTU's total abundance across the whole object and n is the **total number of samples** (Edgar's spreading model: cross-talk leaks into all wells, mostly the zero/low ones). Each cell then receives the UNCROSS2 score

$$s = \frac{2}{1 + \exp(k \cdot x/z)}$$

where x is the observed count in that cell. The score tends to 1 as $x \rightarrow 0$ (looks like cross-talk) and to 0 as $x \gg z$ (genuine signal). Cells with $s \geq t_{\min}$ are treated as cross-talk. The default $t_{\min} = 0.1$ matches Edgar's "weak" (yellow) cut-off; use 0.4 for "strong" only.

Usage

```
uncross2_pq(
  physeq,
  f = 0.01,
  tmin = 0.1,
  k = 1,
  method = c("cut", "subtract"),
  return_scores = FALSE,
  clean_phyloseq_object = TRUE,
  verbose = TRUE
)
```

Arguments

phyloseq	(phyloseq, required) A phyloseq object with integer counts.
f	(numeric or "auto", default 0.01) Expected cross-talk rate: the fraction of an OTU's reads assumed to be misassigned across the plate. Set f = "auto" to estimate it de novo with <code>estimate_xtalk_rate_pq()</code> ; this errors if the estimate is undetermined.
tmin	(numeric, default 0.1) Minimum UNCROSS2 score for a cell to be flagged as cross-talk. 0.1 = weak, 0.4 = strong (Edgar 2018).
k	(numeric, default 1) Steepness of the logistic score, applied as a slope on the exponent ($\exp(k * x / z)$). k = 1 reproduces UNCROSS2; k > 1 makes the transition sharper, k < 1 gentler.
method	(character, default "cut") How flagged cross-talk is handled. "cut": set every flagged cell to 0 (UNCROSS2 behaviour). "subtract": subtract the expected count z from all cells of each OTU and floor negatives at 0 (background subtraction; ignores tmin).
return_scores	(logical, default FALSE) If TRUE, return a list with the input phyloseq object, the cleaned phyloseq object and the score / flag matrices instead of just the phyloseq object (see Value).
clean_phyloseq_object	(logical, default TRUE) Whether to clean the resulting object with <code>MiscMetabar::clean_pq()</code> to drop emptied taxa/samples.
verbose	(logical, default TRUE) Whether to print a one-line summary.

Value

By default a phyloseq object with flagged cross-talk removed. If return_scores = TRUE, a list with:

- old_physeq: the input phyloseq object (before cleaning).
- new_physeq: the cleaned phyloseq object.
- scores: taxa-by-samples matrix of UNCROSS2 scores.
- tagjump: logical taxa-by-samples matrix of flagged cells (always relative to tmin, even when method = "subtract"). Matrices are oriented taxa-as-rows regardless of the input orientation.

Relationship to Edgar's usearch binary

This function faithfully implements the *published* UNCROSS2 score (eq. 3), verified cell-for-cell against an independent reference. It differs from the usearch v11 otutab_xtalk binary in two ways that users should know:

- Rate.** f is fixed (default 0.01) unless f = "auto", whereas the binary *estimates* it de novo. Use `estimate_xtalk_rate_pq()` (or f = "auto") for a de novo estimate.
- Aggressiveness.** The binary's -otutab filtering is far more aggressive than eq. (3): empirically it zeroes essentially every count below the per-OTU mean (N_i/n), keeping only the few "big" source samples, and removed ~40x more cells than eq. (3) at the same rate on data. That behaviour is undocumented and beyond the paper, so this function does **not** reproduce the binary's filtered table; it is the gentler, paper-faithful filter.

UNCROSS2 is defined for **integer count** tables. The usearch binary rejects non-integer input (Invalid count); this function will process relative abundances but the score's interpretation no longer holds, so prefer raw counts.

Author(s)

Adrien Taudière

References

Edgar, R.C. (2018) UNCROSS2: identification of cross-talk in 16S rRNA OTU tables. *bioRxiv* 400762. doi:10.1101/400762

See Also

[estimate_xtalk_rate_pq\(\)](#) for the de novo rate; [decontam_control_samples_pq\(\)](#) for blank-based decontamination.

Examples

```
library(MiscMetabar)
data(data_fungi)

# Remove weak cross-talk (default)
unc_data_fungi <- uncross2_pq(data_fungi)

# Estimate the rate de novo instead of assuming 1%
uncross2_pq(data_fungi, f = "auto")

# Only strong cross-talk, sharper transition
uncross2_pq(data_fungi, tmin = 0.4, k = 2)

# Inspect scores without committing to a threshold
res <- uncross2_pq(data_fungi, return_scores = TRUE)
summary(as.vector(res$scores))

summary_plot_pq(res$old_physeq) / summary_plot_pq(res$new_physeq)
```

Index

arrange_samples_pq, [2](#)
arrange_taxa_pq, [3](#)
autoplot.contam_tbl, [4](#)

base::strsplit(), [39](#)

contam_tbl, [16–23](#)
create_chimera_pq, [5](#)
create_chimera_pq(), [19](#)

decontam_control_samples_pq, [7](#)
decontam_control_samples_pq(), [43](#)
decontam_control_taxa_pq, [9](#)

estimate_xtalk_rate_pq(), [42, 43](#)

filter_chimera_pq, [10](#)
filter_contam_pq, [11](#)
filter_contam_pq(), [10, 11, 16–23](#)
filter_occurrences_pq, [12](#)
filter_samples_pq, [13](#)
filter_taxa_pq, [14](#)
filter_tree_pq, [15](#)

ggplot2::ggplot, [30](#)

identify_contam_blocklist_pq, [16](#)
identify_contam_blocklist_pq(), [23](#)
identify_contam_chimera_pq, [18](#)
identify_contam_chimera_pq(), [5, 6, 10, 11](#)
identify_contam_corr_pq, [19](#)
identify_contam_corr_pq(), [12](#)
identify_contam_negcontrol_pq, [21](#)
identify_contam_primer_pq, [22](#)

known_contaminant_genera, [23](#)
known_contaminant_genera(), [17](#)

merge_markers_pq, [24](#)
MiscMetabar::chimera_removal_vs(), [5, 6](#)

MiscMetabar::clean_pq(), [32, 42](#)
MiscMetabar::cutadapt_remove_primers(),
[23](#)
MiscMetabar::psmelt_samples_pq(), [32](#)
MiscMetabar::transform_pq(), [32](#)
mutate_occurrences_pq, [26](#)
mutate_samdata_pq, [28](#)
mutate_taxa_pq, [29](#)

phyloseq::psmelt(), [32](#)
phyloseq::tax_table(), [40](#)
plot.contam_tbl (autoplot.contam_tbl), [4](#)
plot.contam_tbl(), [19](#)
plot_uncross2_scores, [30](#)
pq_to_tidy, [31](#)
print.contam_tbl, [33](#)

rbind(), [11](#)
rbind.contam_tbl, [33](#)
rename_samples_pq, [34](#)
rename_taxa_pq, [35](#)

select_samdata_pq, [35](#)
select_taxa_pq, [36](#)
slice_samples_pq, [37](#)
slice_taxa_pq, [38](#)

tax_table_to_df, [39](#)
taxa_prevalence, [40](#)
tibble::tibble, [39](#)

uncross2_pq, [41](#)
uncross2_pq(), [30](#)
utils::type.convert(), [39](#)