Package: taxinfo (via r-universe)

June 22, 2026

Type Package

Title Augment 'Phyloseq' Objects with Taxonomy-Based Information

Version 0.1.2

Description Augments 'phyloseq' objects with taxonomy-based information retrieved from multiple external data sources. Provides functions to query 'GBIF' (Global Biodiversity Information Facility), 'Wikipedia', 'GLOBI' (Global Biotic Interactions), 'OpenAlex', and other databases to verify taxonomic names, retrieve occurrence data, access species interaction networks, and fetch scientific literature metadata. Designed to work with the 'phyloseq' data structure common in metabarcoding analyses. Most functions support both 'phyloseq' objects and plain taxonomic name vectors as input.

License AGPL-3

URL https://adrientaudiere.github.io/taxinfo/, https://adrientaudiere.github.io/taxinfo

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

Depends MiscMetabar, R (>= 4.1.0)

Imports cli, dplyr, ggplot2, httr, jsonlite, lifecycle, phyloseq, purrr, rgbif, rglobi, sf, stringr, taxize, tibble, tidyr

Suggests CoordinateCleaner, DECIPHER, digest, duckdb, elevatr, forcats, gbif.range, ggforce, ggpmisc, ggrepel, ggtext, ggraph, ggstatsplot, ggVennDiagram, ggtree, htmltools, igraph, knitr, leaflet, leafpop, maps, mapview, openalexR, patchwork, rentrez, rmarkdown, rnaturalearth, rvest, systemfonts, terra, testthat (>= 3.0.0), vcr (>= 1.0.0), wikitaxa

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libglpk-dev make libbz2-dev libicu-dev libjpeg-dev liblzma-dev libpng-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev xz-utils zlib1g-dev

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

Date/Publication 2026-06-22 16:10:42 UTC

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

RemoteRef HEAD

RemoteSha c66182e1010563a2547c0827a280987f9012173f

Contents

`taxinfo-package` `taxinfo` *package*

Description

`taxinfo` package

`check_package` *Check package availability and propose installation instructions*

Description

 <img src="https://img.shields.io/badg
maturing-blue" alt="lifecycle-maturing">

This function checks if a package is available using requireNamespace. If the package is not available, it provides helpful installation instructions.

Usage

```
check_package(
  package,
  repo = "CRAN",
  github_repo = NULL,
  stop_on_error = TRUE,
  quietly = TRUE
)
```

Arguments

`package`	(required) Character string. Name of the package to check.
`repo`	Character string. Repository source for installation suggestion. Options: "CRAN" (default), "Bioconductor", "GitHub".
`github_repo`	Character string. GitHub repository in format "username/repository". It overrides repo if provided. Required if repo is "GitHub".

stop_on_error	Logical. If TRUE (default), stops execution when package is missing. If FALSE, returns FALSE and shows message.
quietly	Logical. If TRUE, suppresses the requireNamespace loading messages. Default is TRUE.

Value

Logical. TRUE if package is available, FALSE if not available.

Examples

```
## Not run:
# Check CRAN package
check_package("dplyr")

# Check Bioconductor package
check_package("Biostrings", repo = "Bioconductor")

# Check GitHub package
check_package("MiscMetabar",
  repo = "GitHub",
  github_repo = "adrientaudiere/MiscMetabar"
)

# Stop execution if package is missing
check_package("ggplot2", stop_on_error = TRUE)

## End(Not run)
```

---

cluster_sbc *Create Species-Bound Clusters using SWARM algorithm*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

This function creates Species-Bound Clusters (SBC) from a phyloseq object containing taxa (ASV/OTU) sequences and taxonomy based on a proposition by Riley *et al.* 2025 (<https://doi.org/10.1186/s12915-025-02284-x>).

SBC (Species bound cluster) are defined as "clusters that include all and only ESVs assigned to one species, the sequence similarity threshold can vary between these clusters" (Riley *et al.* 2025 <https://doi.org/10.1186/s12915-025-02284-x>).

It uses the SWARM algorithm to cluster taxa within each taxnames (e.g. Genus species) based on sequence similarity, allowing for variable d values to optimize clustering.

Run swarm with d=1 to d=max_d, then for each taxnames (e.g. Species binomial name), find the lowest d that clusters all taxa assigned to this taxnames into one cluster. If a taxnames is represented by only one taxa, it is not clustered. Taxnames containig "NA" are considered as unassigned. By

default, unassigned ASVs are clustered into other cluster without counting for their own taxnames. Set include_unassigned = FALSE to force cluster to included all taxa with a given taxnames but none of the unassigned ones.

If the maximum d is reached, keep the clustering at this d and print a warning. If strict_sbc = TRUE, only taxa corresponding to strict SBC will be clustered and return in the phyloseq object, in that cases, taxa whose taxnames is clustered into multiple clusters or whose cluster contains multiple taxnames will have NA as cluster_ID and will be removed from phyloseq object.

The function returns a data.frame with the cluster assignments and the optimal d value for each taxnames, as well as a modified phyloseq object with the cluster information added to the taxonomy table.

Usage

```
cluster_sbc(
  physeq,
  taxonomic_rank = c("Genus", "Species"),
  max_d = 20,
  include_unassigned = TRUE,
  allow_multiple_taxa = FALSE,
  regroup_cluster = TRUE,
  tax_adjust = 1L,
  verbose = TRUE
)
```

Arguments

physeq	A phyloseq object containing ASV/OTU sequences and taxonomy
taxonomic_rank	Character. Name of the taxonomy column(s) containing taxonomic assignments to build SBC. Can be a vector of two columns (e.g. c("Genus", "Species"), the default).
max_d	Integer. Maximum d value to test for SWARM (default: 20)
include_unassigned	Logical. Whether to cluster unassigned taxa separately (default: TRUE)
allow_multiple_taxa	Logical. If TRUE, allow clusters to contain multiple taxnames (default: FALSE)
regroup_cluster	Logical. If TRUE, regroup taxa in the phyloseq object based on their cluster_ID using [merge_taxa_vec()] (default: TRUE)
tax_adjust	Character vector. See ?[MiscMetabar::merge_taxa_vec()] 0: no adjustment; 1: phyloseq-compatible adjustment; 2: conservative adjustment
verbose	Logical. Print progress messages (default: TRUE)

Value

A list containing: - clusters: data.frame with taxa_id, taxnames, cluster_ID, optimal_d - summary: data.frame with summary statistics - n_taxa: total number of taxa - n_unassigned: number of unassigned taxa - n_taxa: number of unique taxnames - n_already_SBC: number of taxnames already

represented by a single taxa - n_taxa_to_cluster: number of taxnames with multiple taxa to cluster - n_SBC: number of SBC clusters created - d_per_taxnames: data.frame with taxnames, n_taxa, optimal_d, n_clusters, other_taxnames, unassigned_taxa - physeq_with_info: modified phyloseq object with cluster info added to tax_table - cluster_ID: The id of the SBC cluster - cluster_d: The optimal d value used to create the SBC cluster - other_taxnames_in_cluster (logical) - unassigned_taxa_in_cluster (logical) - physeq_SBC: modified phyloseq object with cluster info added to tax_table

Author(s)

Adrien Taudiere

See Also

[MiscMetabar::swarm_clustering()], [MiscMetabar::postcluster_pq()]

Examples

```
res <- cluster_sbc(data_fungi_mini)

track_wkflow(list(data_fungi_mini, res$physeq_SBC))

ggplot(
  res$d_per_taxnames,
  aes(x = reorder(taxnames, n_taxa), y = n_taxa, fill = optimal_d)
) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = paste0(other_taxnames)),
    hjust = -0.1, size = 3,
    fontface = "italic"
  ) +
  coord_flip() +
  scale_fill_viridis_c(option = "plasma") +
  labs(
    title = "Species-Bound Clusters with Optimal d Values",
    subtitle = "Labels depict taxonomic names clustered into SBC",
    x = "Species",
    y = "Number of Taxa",
    fill = "Optimal d"
  ) +
  theme(axis.text.y = element_text(size = 10, face = "italic"))
```

---

extract_spores_mycodb *Extract spore size from mycoDB for a single species*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Usage

```
extract_spores_mycodb(species_name, verbose = TRUE)
```

Arguments

species_name	Character. Species name, e.g. "Amanita muscaria"
verbose	(logical, default TRUE) If TRUE, prompt some messages.

Value

A character string with the spore size, e.g. "8-10 x 6-8 \u00b5m". If the species is not found in mycoDB, returns "Not in mycoDB", if the species is found but no spore size info is available, returns "No spore size info in mycoDB".

Author(s)

Adrien Taudiere

See Also

[tax_spores_size_pq()]

Examples

```
extract_spores_mycodb("Amanita muscaria")
extract_spores_mycodb("Boletus edulis")
extract_spores_mycodb("Xylobolus subpileatus")
extract_spores_mycodb("Nonexistent species")
extract_spores_mycodb("Amanita")
```

---

fungal_traits_guilds *Add FungalTraits and FUNGuild information to a phyloseq object*

Description

<img src="https://img.shields.io/badg
maturing-blue" alt="lifecycle-maturing">

A convenience wrapper that adds guild and trait information from both the FungalTraits database and the FUNGuild database to the ‘tax_table‘ slot of a phyloseq object. Optionally creates consensus columns that summarise agreement between the two databases.

If ‘currentCanonicalSimple‘ is not already present in the ‘tax_table‘, [gna_verifier_pq()] is called internally to clean and verify the taxonomic names before querying the databases.

Usage

```
fungal_traits_guilds(
  physeq,
  fungal_traits_file = system.file("extdata", "fungal_traits.csv", package = "taxinfo"),
  ft_taxonomic_rank = "genusEpithet",
  ft_csv_rank = "GENUS",
  ft_sep = "\t",
  ft_col_prefix = "ft_",
  fg_tax_levels = c("Kingdom", "Phylum", "Class", "Order", "Family", "Genus", "Species"),
  fg_col_prefix = "fg_",
  ft_csv_cols_select = c("GENUS", "COMMENT.on.genus", "primary_lifestyle",
    "Secondary_lifestyle", "Comment_on_lifestyle_template",
   "Endophytic_interaction_capability_template", "Plant_pathogenic_capacity_template",
   "Decay_substrate_template", "Decay_type_template", "Aquatic_habitat_template",
   "Animal_biotrophic_capacity_template", "Specific_hosts", "Growth_form_template",
    "Fruitbody_type_template", "Hymenium_type_template",
   "Ectomycorrhiza_exploration_type_template", "Ectomycorrhiza_lineage_template",
    "primary_photobiont",
     "secondary_photobiont"),
  db_url = "http://www.stbates.org/funguild_db_2.php",
  add_consensus = TRUE,
  consensus_col_prefix = "cons_",
  add_to_phyloseq = TRUE,
  gna_data_sources = c(1, 12),
  verbose = TRUE
)
```

Arguments

physeq	A phyloseq object.
fungal_traits_file	(Character) Path to the FungalTraits CSV file. Defaults to the simplified version bundled with the package.
ft_taxonomic_rank	(Character, default ‘"genusEpithet"‘) Column in ‘tax_table‘ used to match against the FungalTraits genus column.
ft_csv_rank	(Character, default ‘"GENUS"‘) Column in the FungalTraits CSV file that contains genus names.
ft_sep	(Character, default ‘";"‘) Field separator of the FungalTraits CSV file. See [utils::read.csv()].
ft_col_prefix	(Character, default ‘"ft_"‘) Prefix applied to all columns imported from FungalTraits.
fg_tax_levels	(Character vector) Names of the tax_table columns that represent the 7 standard taxonomic ranks fed to FUNGuild.
fg_col_prefix	(Character, default ‘"fg_"‘) Prefix applied to all columns imported from FUNGuild.

`ft_csv_cols_select`
: A character vector of the column names to select from the FungalTraits CSV file.

`db_url`
: (Character) URL of the FUNGuild database. See [MiscMetabar::get_funguild_db()].

`add_consensus`
: (Logical, default 'TRUE') If 'TRUE', add consensus columns comparing trophic modes assigned by the two databases.

`consensus_col_prefix`
: (Character, default '"cons_"') Prefix applied to consensus columns.

`add_to_phyloseq`
: (Logical, default 'TRUE') If 'TRUE', return an updated phyloseq object. If 'FALSE', return a tibble of the tax_table.

`gna_data_sources`
: Integer or character vector passed to [gna_verifier_pq()] when taxonomic names need to be verified. See [taxize::gna_verifier()].

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

Value

Either an updated phyloseq object (when 'add_to_phyloseq = TRUE') or a tibble of the augmented tax_table.

Author(s)

Adrien Taudiere

See Also

[tax_info_pq()], [gna_verifier_pq()], [MiscMetabar::add_funguild_info()], [MiscMetabar::funguild_assign()]

Examples

```
## Not run:
# physeq object with already-verified names
res_guild <- data_fungi |>
 gna_verifier_pq(data_sources = 210) |>
   fungal_traits_guilds()

table(res_guild@tax_table[, "cons_trophicMode"], useNA = "always")
table(res_guild@tax_table[, "cons_trophicMode_agreement"], useNA = "always")

# physeq object WITHOUT verified names: gna_verifier_pq is called internally
res_guild_2 <- fungal_traits_guilds(data_fungi, gna_data_sources = 210)
table(res_guild_2@tax_table[, "ft_primary_lifestyle"])
table(res_guild_2@tax_table[, "fg_trophicMode"])
table(res_guild_2@tax_table[, "cons_trophicMode"])

# Return a tibble instead of a phyloseq
data_fungi_cleanNames <- gna_verifier_pq(data_fungi, data_sources = 210)
tib <- fungal_traits_guilds(data_fungi_cleanNames, add_to_phyloseq = FALSE)
```

```
res_guild_2 |> psmelt() |>
 filter(Abundance > 0) |>
 ggplot(aes(x = Height, y = Abundance, fill = cons_trophicMode)) +
 geom_col() +
 theme_bw() +
 labs(x = "Height", y = "Molecular abundance", fill = "Consensus trophic mode") +
 theme(axis.text.x = element_text(angle = 45, hjust = 1))

 tax_bar_pq(res_guild_2, "Height", "cons_trophicMode", add_ribbon=TRUE)

## End(Not run)
```

---

gna_verifier_pq *Verify (and fix) scientific names (Genus species) of a phyloseq object.*

Description

 <img src="https://img.shields.io/badg
maturing-blue" alt="lifecycle-maturing">

A wrapper of [taxize::gna_verifier()] apply to phyloseq object

Usage

```
gna_verifier_pq(
  physeq = NULL,
  taxnames = NULL,
  taxonomic_rank = c("Genus", "Species"),
  data_sources = c(1, 12),
  all_matches = FALSE,
  capitalize = FALSE,
  species_group = FALSE,
  fuzzy_uninomial = FALSE,
  verbose = TRUE,
  add_to_phyloseq = NULL,
  col_prefix = NULL,
  genus_species_canonical_col = TRUE,
  year_col = TRUE,
  authorship_col = TRUE,
  discard_NA = TRUE,
  problematic_chars = "[?\\\\#|&]",
  clean_problematic_chars = FALSE,
  force_recompute = FALSE
)
```

Arguments

physeq	(optional) A phyloseq object. Either ‘physeq‘ or ‘taxnames‘ must be provided, but not both.

`taxnames` (optional) A character vector of taxonomic names.

`taxonomic_rank` (Character) The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. the default c("Genus", "Species")).

`data_sources` A character or integer vector. See [taxize::gna_verifier()] documentation. For example, 1=Catalogue of Life, 3=ITIS, 5=Index Fungarum, 11=GBIF backbone and 210=TaxRef.

`all_matches` (Logical) See [taxize::gna_verifier()] documentation.

`capitalize` (Logical) See [taxize::gna_verifier()] documentation.

`species_group` (Logical) See [taxize::gna_verifier()] documentation.

`fuzzy_uninomial`
(Logical) See [taxize::gna_verifier()] documentation.

`verbose` (logical, default TRUE) If TRUE, prompt some messages.

`add_to_phyloseq`
(logical, default TRUE when physeq is provided, FALSE when taxnames is provided)

- If FALSE, return the result of the [taxize::gna_verifier()] function + a column taxa_names_in_phyloseq depicting the name of the taxa from the phyloseq object.

- If TRUE return a phyloseq object with amended slot '@taxtable'. Cannot be TRUE if 'taxnames' is provided. At least three new columns are added: - **taxa_name**: The character string sent to gna_verifier (e.g. 'Antrodiella brasiliensis') - **currentName**: The current accepted name (resolve the synonym) with autorities at the end of the binominal name (e.g. 'Trametopsis brasiliensis (Ryvarden & de Meijer) Gomez-Mont. & Robledo)'. - **currentCanonicalSimple**: The current accepted name without autorities (e.g. 'Trametopsis brasiliensis', 'Russula').

Other columns can be added depending on the parameters: 'genus_species_canonical_col' (adds "genusEpithet", "specificEpithet", and "genusSpeciesEpithet"), 'year_col', 'authorship'.

`col_prefix` A character string to be added as a prefix to the new columns names added to the tax_table slot of the phyloseq object (default: NULL).

`genus_species_canonical_col`
(logical, default TRUE) If TRUE three new columns are added along with "currentCanonicalSimple": "genusEpithet", "specificEpithet" and "genusSpeciesEpithet". "genusSpeciesEpithet" is identical to "currentCanonicalSimple" but is NA when "specificEpithet" is NA or empty (i.e. genus-only names are excluded).

`year_col` (logical, default TRUE) If TRUE a new column "namePublishedInYear" is added with the year of publication.

`authorship_col` (logical, default TRUE) If TRUE three new columns are added: "authorship", "bracketauthorship" and "scientificNameAuthorship".

`discard_NA` (logical, default 'TRUE'). Passed to [taxonomic_rank_to_taxnames()].

`problematic_chars`
A regex pattern (character string) to detect characters that are problematic for the GNA Verifier API URL. The API pastes names pipe-separated into a GET URL

path, so characters like '¿ (query-string delimiter), '\' (escape), '|' (pipe separator), '#' (fragment), or '&' (parameter separator) corrupt the URL and can cause a length-mismatch crash in [taxize::gna_verifier()]. Names containing these characters are reported and, if 'clean_problematic_chars' is 'TRUE', handled before verification. Set to 'NULL' to disable detection. Default: '"[?\\#|&]"'.

`clean_problematic_chars`
: (logical, default 'FALSE') If 'TRUE', cells in the 'taxonomic_rank' columns that match 'problematic_chars' are replaced with 'NA' (when 'physeq' is provided) and matching names are filtered out (when 'taxnames' is provided) before verification. If 'FALSE' (the default), a warning is issued listing the problematic names but they are sent as-is – this will likely cause an error in [taxize::gna_verifier()]. Set to 'TRUE' to handle them automatically, or clean the data upstream (e.g. with [MiscMetabar::simplify_taxo()]).

`force_recompute`
: (logical, default 'FALSE') If 'TRUE', remove any existing columns in the 'tax_table' that would be re-added by this call (i.e. columns matching 'col_prefix' when 'col_prefix' is set, or columns in 'new_cols' when 'col_prefix' is 'NULL') before performing the verification. This is useful when re-running 'gna_verifier_pq()' on a phyloseq that already contains result columns from a previous call. If 'FALSE', existing columns are left in place, which can cause duplicate-column errors in 'tax_table()' on re-runs.

Details

This function is mainly a wrapper of the work of others. Please cite 'taxize' package.

Value

Either a tibble (if add_to_phyloseq = FALSE) or a new phyloseq object with new columns (see param add_to_phyloseq) in the tax_table slot.

Author(s)

Adrien Taudiere

See Also

[taxize::gna_verifier()]

Examples

```
## Not run:
df <- gna_verifier_pq(data_fungi, data_sources = 210, add_to_phyloseq = FALSE)

data_fungi_mini_cleanNames <- gna_verifier_pq(data_fungi_mini, data_sources = 210)

data_fungi_cleanNames <- gna_verifier_pq(data_fungi, data_sources = 210)

sum(!is.na(data_fungi_cleanNames@tax_table[, "currentName"]))
sum(data_fungi_cleanNames@tax_table[, "currentCanonicalSimple"] !=
```

```
  data_fungi_cleanNames@tax_table[, "taxa_name"], na.rm = TRUE)
# 1010 taxa (71% of total) are identified using a currentName including 434
# corrected values (correction using synonym disambiguation)


tr <- rotl_pq(data_fungi_cleanNames,
  taxonomic_rank = "currentCanonicalSimple",
  context_name = "Basidiomycetes"
)

p <- ggtree::ggtree(tr, layout = "roundrect") +
  ggtree::geom_nodelab(hjust = 1, vjust = -1.2, size = 2) +
  ggtree::geom_tiplab(size = 2)

p + xlim(0, max(p$data$x) + 1)


psmelt(data_fungi_mini_cleanNames) |>
  filter(Abundance > 0) |>
  mutate(namePublishedInYear = as.numeric(namePublishedInYear)) |>
  pull(namePublishedInYear) |>
  hist(breaks = 100)


# Does the fungal species discovered more recently tend to be found at
# greater heights in the tree?
psmelt(data_fungi_mini_cleanNames) |>
  filter(Abundance > 0) |>
  group_by(Height) |>
  mutate(namePublishedInYear = as.numeric(namePublishedInYear)) |>
  ggstatsplot::ggbetweenstats("Height", "namePublishedInYear")

## End(Not run)
```

`idest_colors` *IdEst colors for ggplot theme_idest*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Usage

```
idest_colors(
  palette_name = "all_color_idest",
  n,
  type = c("discrete", "continuous"),
  direction = c(1, -1),
  override_order = FALSE
)
```

Arguments

palette_name	The name of the palette to use. The available palette are c("all_color_idest", "ligth_color_idest", "dark_color_idest", "Picabia", "Picasso", "Levine2", "Rattner", "Sidhu", "Hokusai2", "Hokusai3"). See [idest_pal] for more details.
n	Number of colors to return.
type	Type of palette. Either "discrete" or "continuous".
direction	Direction of the palette. 1 for standard, -1 for reversed.
override_order	Logical, whether to override the order of the palette.

Value

A vector of colors.

Author(s)

Adrien Taudiere

---

idest_pal *IdEst color palettes*

Description

Palettes of color for IdEst including also some palettes from MoMAColors <https://github.com/BlakeRMills/MoMAColors/bl

The available palette are c("all_color_idest", "ligth_color_idest", "dark_color_idest", "Picabia", "Picasso", "Levine2", "Rattner", "Sidhu", "Hokusai2", "Hokusai3")

Usage

```
idest_pal
```

Author(s)

Adrien Taudiere

`intra_taxnames_dist` *Compute intra-taxanames distances for each taxa names*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

This function computes intra-taxanames distances for each taxonomic names (e.g. Genus species) in a phyloseq object containing ASV/OTU sequences and taxonomy.

The distances are computed using the DECIPHER package, which aligns the sequences (‘DECIPHER::AlignSeqs()‘) and calculates a distance matrix (‘DECIPHER::DistanceMatrix()‘).

Usage

```
intra_taxnames_dist(
  physeq,
  taxonomic_rank = c("Genus", "Species"),
  verbose = TRUE,
  verbose_DECIPHER = FALSE,
  discard_NA = TRUE,
  ...
)
```

Arguments

`physeq`	A phyloseq object containing ASV/OTU sequences and refseq
`taxonomic_rank`	Character. Name of the taxonomy column(s) containing taxonomic assignments to compute intra-taxa distances. Can be a vector of two columns (e.g. c("Genus", "Species"), the default).
`verbose`	Logical. Print progress messages (default: TRUE)
`verbose_DECIPHER`	Logical. If TRUE, print messages from DECIPHER functions (default: FALSE)
`discard_NA`	(logical, default ‘TRUE‘). Passed to [taxonomic_rank_to_taxnames()].
`...`	Additional arguments to pass to ‘DECIPHER::AlignSeqs()‘

Value

A data.frame with columns: - taxnames: taxonomic names - n_taxa: number of taxa assigned to this taxnames - mean_dist: mean intra-taxanames distance - min_dist: minimum intra-taxanames distance - max_dist: maximum intra-taxanames distance

Author(s)

Adrien Taudiere

See Also

[DECIPHER::AlignSeqs()], [DECIPHER::DistanceMatrix()]

Examples

```
intra_taxn_dist <- intra_taxnames_dist(data_fungi_mini)
plot(intra_taxn_dist$mean_dist, intra_taxn_dist$n_taxa)
plot(intra_taxn_dist$min_dist, intra_taxn_dist$n_taxa)
plot(intra_taxn_dist$max_dist, intra_taxn_dist$n_taxa)
```

---

label_italic_species *Format taxon labels with species names in italic*

Description

Returns plotmath expressions that render binomial or trinomial species names (labels containing a space) in italic, leaving single-word labels (genus, family, etc.) unchanged. Intended as a ‘labels‘ formatter for discrete ggplot2 scales, or as a standalone helper.

Uses plotmath ‘italic()‘ expressions instead of markdown, so no ‘ggtext::element_markdown()‘ theme element is required. This makes the function compatible with any ggplot2 theme, including complete themes such as [theme_idest()].

Usage

```
label_italic_species(x)
```

Arguments

x	A character vector of taxon labels.

Value

A list of plotmath expressions (for species names) and plain character strings (for non-species labels), suitable for use as ‘labels‘ in [ggplot2::scale_x_discrete()] or [ggplot2::scale_y_discrete()].

Author(s)

Adrien Taudiere

See Also

[scale_x_italic_species()], [scale_y_italic_species()]

Examples

```
label_italic_species(c("Russula nigricans", "Amanita", "Boletus edulis"))
```

plot_tax_gbif_pq *Plot the taxa occurrence using gbif.range package*

Description

<img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

A wrapper of [gbif.range::get_gbif()] function to plot the range of taxa using ggplot2. The function can take either a phyloseq object or a vector of taxonomic names. If a phyloseq object is provided, the taxonomic names are extracted from the specified taxonomic rank. The occurrences are plotted either as points or using hexagonal binning. The function can also filter the occurrences by country.

Usage

```
plot_tax_gbif_pq(
  physeq = NULL,
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  interactive_plot = FALSE,
  zcol = c("year", "taxonomicStatus"),
  hexagons = FALSE,
  bins = 100,
  verbose = TRUE,
  countries = NULL,
 info_names = c("country", "country code", "acceptedScientificName", "ScientificName"),
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE,
  ...
)
```

Arguments

physeq	(optional) A phyloseq object
taxnames	(optional) A character vector of taxonomic names.
taxonomic_rank	(Character, default "currentCanonicalSimple") The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. c("Genus", "Species")).
interactive_plot	(logical, default FALSE) If TRUE, an interactive map is created using the package mapview.
zcol	(character vector, default c("year", "taxonomicStatus")) Only used if interactive_plot is TRUE. The column(s) of the occurrences to use for coloring the points in the interactive map. See ?mapview::mapview() for more details.
hexagons	(logical, default FALSE) Only used if interactive_plot is FALSE. If TRUE, use hexagonal binning to plot the occurrences. If FALSE, plot the occurrences as points.

bins	(Number of bins for hexagonal binning, default 100) Only used if hexagons is TRUE and interactive_plot is FALSE.
verbose	(logical, default TRUE) If TRUE, prompt some messages.
countries	A character vector of country names to filter the occurrences. If NULL (default), all countries are used (no filter).
info_names	(Character vector) The information to retrieve from GBIF for each occurrence. See [gbif.range::get_gbif()] for more details.
discard_genus_alone	(logical, default 'TRUE' when 'taxonomic_rank == "currentCanonicalSimple"'). Passed to [taxonomic_rank_to_taxnames()].
discard_NA	(logical, default 'TRUE'). Passed to [taxonomic_rank_to_taxnames()].
...	Additional arguments to pass to [gbif.range::get_gbif()].

Value

A list of ggplot2 objects, one for each taxon.

Author(s)

Adrien Taudiere

See Also

[gbif.range::get_gbif()], [range_bioreg_pq()], [tax_check_occur_pq()], [tax_check_ecoregion()]

Examples

```
## Not run:
data_fungi_mini_cleanNames <- gna_verifier_pq(data_fungi_mini, data_sources = 210)

p <- plot_tax_gbif_pq(
  subset_taxa_pq(
    data_fungi_mini_cleanNames,
    taxa_sums(data_fungi_mini) > 20000
  ),
  hexagons = TRUE,
  verbose = TRUE, bins = 50, occ_samp = 100, grain = 10000
)

p <- plot_tax_gbif_pq(taxnames = c("Xylobolus subpileatus", "Stereum subpileatus"))

p <- plot_tax_gbif_pq(taxnames = c("Stereum ostrea", "Mycena renati"))
requireNamespace("patchwork")
p[[1]] / p[[2]] & no_legend()


p <- plot_tax_gbif_pq(
  taxnames = c("Xylobolus subpileatus", "Stereum  subpileatus"),
  hexagons = TRUE, verbose = FALSE
)
```

```
p <- plot_tax_gbif_pq(
  taxnames = c("Xylobolus subpileatus", "Stereum subpileatus"),
  hexagons = TRUE, verbose = FALSE, countries = c("france", "spain")
)

p[[1]] + coord_fixed(ylim = c(30, 50), xlim = c(-5, 25)) + no_legend()

p <- plot_tax_gbif_pq(
  taxnames = c(
    "Ossicaulis lachnopus",
    "Antrodiella brasiliensis",
    "Stereum ostrea",
    "Xylobolus subpileatus"
  ),
  hexagons = TRUE,
  verbose = F, bins = 50, occ_samp = 100, grain = 10000
)

requireNamespace("patchwork")
(p[[1]] + p[[2]]) /
  (p[[3]] + p[[4]]) & no_legend()

## End(Not run)
```

`points_to_ecoregions` *Map GPS points to WWF/TNC terrestrial ecoregions*

Description

<img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Assigns each GPS point (pair of longitude/latitude in decimal degrees, WGS84) to the WWF/TNC terrestrial ecoregion, biome and realm that contains it.

Usage

```
points_to_ecoregions(longitudes, latitudes, ecoregions = NULL)
```

Arguments

`longitudes`	(numeric vector). Longitudes of the points to locate, in decimal degrees in '[-180, 180]'.
`latitudes`	(numeric vector). Latitudes of the points to locate, in decimal degrees in '[-90, 90]'. Must have the same length as 'longitudes'.
`ecoregions`	(optional 'sf' object, default 'NULL'). Ecoregion polygon layer to use. If 'NULL', the shipped WWF/TNC layer is loaded via [load_ecoregions()] (result is cached, so passing 'NULL' is usually the right choice).

Value

A tibble with one row per input point and the columns ‘point_id‘ (integer), ‘longitude‘, ‘latitude‘, ‘ECO_NAME‘, ‘biome‘, ‘realm‘. Points falling outside any ecoregion (oceans, poles...) have ‘NA‘ in the three ecoregion columns.

Author(s)

Adrien Taudiere

See Also

[tax_check_ecoregion()], [tax_ecoregion_occur()]

Examples

```
## Not run:
points_to_ecoregions(
  longitudes = c(2.3522, 4.2, -70),
  latitudes  = c(48.8566, 33, -33)
)

## End(Not run)
```

`range_bioreg_pq` *Get and plot the range of taxa within a bioregion using gbif.range package*

Description

<img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

A wrapper of [gbif.range::get_gbif()] and [gbif.range::get_range()] functions to get and plot the range of taxa using ggplot2. The function takes a phyloseq object as input and extracts the taxonomic names from the specified taxonomic rank. The occurrences are plotted as points on a map, along with the range of the taxon within a specified bioregion. The bioregion used is "eco_terra", which corresponds to terrestrial ecoregions defined by the World Wildlife Fund (WWF).

Usage

```
range_bioreg_pq(
  physeq = NULL,
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  occ_samp = 5000,
  verbose = TRUE,
  verbose_gbif_range = FALSE,
  make_plot = FALSE,
```

```
  crop_plot = TRUE,
  remove_legend = TRUE,
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE,
  ...
)

plot_range_bioreg_pq(...)
```

Arguments

physeq	(optional) A phyloseq object. Either ‘physeq‘ or ‘taxnames‘ must be provided, but not both.
taxnames	(optional) A character vector of taxonomic names.
taxonomic_rank	(Character, default "currentCanonicalSimple") The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. c("Genus", "Species")).
occ_samp	(Numeric, default 5000) Number of occurrences to sample from GBIF. See [gbif.range::get_gbif()] for more details.
verbose	(logical, default TRUE) If TRUE, prompt some messages.
verbose_gbif_range	(logical, default TRUE) If TRUE, prompt some messages from gbif.range functions.
make_plot	(logical, default TRUE) If TRUE, return a list of ggplot objects. Else return a list of range outputs from [gbif.range::get_range()].
crop_plot	(logical, default TRUE) If TRUE, crop the plot to the extent of the bioregion.
remove_legend	(logical, default TRUE) If TRUE, remove the legend from the plot.
discard_genus_alone	(logical, default ‘TRUE‘ when ‘taxonomic_rank == "currentCanonicalSimple"‘). Passed to [taxonomic_rank_to_taxnames()].
discard_NA	(logical, default ‘TRUE‘). Passed to [taxonomic_rank_to_taxnames()].
...	Additional arguments to pass to [gbif.range::get_gbif()].

Details

[plot_range_bioreg_pq()] is a wrapper of just a shortcut for ‘range_bioreg_pq(..., make_plot = TRUE)‘.

Value

If make_plot = TRUE (default), a list of ggplot objects, one for each taxon. If make_plot = FALSE, a list of range outputs from [gbif.range::get_range()].

Author(s)

Adrien Taudiere

See Also

[gbif.range::get_gbif()], [plot_tax_gbif_pq()], [gbif.range::get_range()]

Examples

```
## Not run:

data_fungi_mini_cleanNames <- gna_verifier_pq(data_fungi_mini, data_source = 210)

res_range <- range_bioreg_pq(subset_taxa(
  data_fungi_mini_cleanNames,
  currentCanonicalSimple %in% c("Xylodon flaviporus", "Basidiodendron eyrei")
), occ_samp = 100)

p <- plot_range_bioreg_pq(subset_taxa(
  data_fungi_mini_cleanNames,
  currentCanonicalSimple %in% c("Xylodon flaviporus", "Basidiodendron eyrei")
), occ_samp = 100)

requireNamespace("patchwork")
p[[1]] / p[[2]]

p <- plot_range_bioreg_pq(subset_taxa_pq(
  data_fungi_mini_cleanNames,
  taxa_sums(data_fungi_mini) > 20000
), occ_samp = 500)

p[[1]] / p[[2]]

## End(Not run)
```

`scale_color_idest_c` *IdEst continuous color scales for ggplot2*

Description

<img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Usage

```
scale_color_idest_c(palette_name = "all_color_idest", direction = 1, ...)
```

Arguments

`palette_name`	The name of the palette to use. The available palette are c("all_color_idest", "ligth_color_idest", "dark_color_idest", "Picabia", "Picasso", "Levine2", "Rattner", "Sidhu", "Hokusai2", "Hokusai3"). See [idest_pal] for more details.
`direction`	Direction of the palette. 1 for standard, -1 for reversed.
`...`	Additional arguments passed to [ggplot2::scale_color_gradientn()].

Value

A ggplot2 scale object.

Author(s)

Adrien Taudiere

`scale_color_idest_d` *IdEst discrete color scales for ggplot2*

Description

<img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Usage

```
scale_color_idest_d(
  palette_name = "all_color_idest",
  direction = 1,
  override_order = FALSE,
  ...
)
```

Arguments

`palette_name`	The name of the palette to use. The available palette are c("all_color_idest", "ligth_color_idest", "dark_color_idest", "Picabia", "Picasso", "Levine2", "Rattner", "Sidhu", "Hokusai2", "Hokusai3"). See [idest_pal] for more details.
`direction`	Direction of the palette. 1 for standard, -1 for reversed.
`override_order`	Logical (default FALSE), whether to override the order of the palette.
`...`	Additional arguments passed to [ggplot2::scale_color_gradientn()].

Value

A ggplot2 scale object.

Author(s)

Adrien Taudiere

---

scale_fill_idest_c *IdEst continuous fill scales for ggplot2 <img src="https://img.shields.io/badge/lifecycle-experimental-orange" alt="lifecycle-experimental">*

Description

IdEst continuous fill scales for ggplot2 <img src="https://img.shields.io/badge/lifecycle-experimental-orange" alt="lifecycle-experimental">

Usage

```
scale_fill_idest_c(palette_name = "all_color_idest", direction = 1, ...)
```

Arguments

palette_name	The name of the palette to use. The available palette are c("all_color_idest", "ligth_color_idest", "dark_color_idest", "Picabia", "Picasso", "Levine2", "Rattner", "Sidhu", "Hokusai2", "Hokusai3"). See [idest_pal] for more details.
direction	Direction of the palette. 1 for standard, -1 for reversed.
...	Additional arguments passed to [ggplot2::scale_color_gradientn()].

Value

A ggplot2 scale object.

Author(s)

Adrien Taudiere

---

scale_fill_idest_d *IdEst discrete fill scales for ggplot2 <img src="https://img.shields.io/badge/lifecycle-experimental-orange" alt="lifecycle-experimental">*

Description

IdEst discrete fill scales for ggplot2 <img src="https://img.shields.io/badge/lifecycle-experimental-orange" alt="lifecycle-experimental">

Usage

```
scale_fill_idest_d(
  palette_name = "all_color_idest",
  direction = 1,
  override_order = FALSE,
  ...
)
```

Arguments

palette_name	The name of the palette to use. The available palette are c("all_color_idest", "ligth_color_idest", "dark_color_idest", "Picabia", "Picasso", "Levine2", "Rattner", "Sidhu", "Hokusai2", "Hokusai3"). See [idest_pal] for more details.
direction	Direction of the palette. 1 for standard, -1 for reversed.
override_order	Logical (default FALSE), whether to override the order of the palette.
...	Additional arguments passed to [ggplot2::scale_color_gradientn()].

Value

A ggplot2 scale object.

Author(s)

Adrien Taudiere

---

scale_x_italic_species

Discrete x-axis scale with species names in italic

Description

A drop-in replacement for [ggplot2::scale_x_discrete()] that automatically renders binomial species names (labels containing a space) in italic using plotmath expressions, while leaving single-word labels unchanged. Works with any theme, including complete themes like [theme_idest()].

Usage

```
scale_x_italic_species(...)
```

Arguments

...	Arguments passed to [ggplot2::scale_x_discrete()].

Value

A ‘scale_x_discrete‘ ggplot2 scale object.

Author(s)

Adrien Taudiere

See Also

[scale_y_italic_species()], [label_italic_species()]

Examples

```
library(ggplot2)
df <- data.frame(
  sp = c("Russula nigricans", "Amanita", "Boletus edulis"),
  n  = c(10, 5, 8)
)
ggplot(df, aes(x = sp, y = n)) +
  geom_col() +
  theme_minimal() +
  scale_x_italic_species()
```

---

scale_y_italic_species

Discrete y-axis scale with species names in italic

Description

A drop-in replacement for [ggplot2::scale_y_discrete()] that automatically renders binomial species names (labels containing a space) in italic using plotmath expressions, while leaving single-word labels unchanged. Works with any theme, including complete themes like [theme_idest()].

Usage

```
scale_y_italic_species(...)
```

Arguments

...	Arguments passed to [ggplot2::scale_y_discrete()].

Value

A ‘scale_y_discrete‘ ggplot2 scale object.

Author(s)

Adrien Taudiere

See Also

[scale_x_italic_species()], [label_italic_species()]

Examples

```
library(ggplot2)
df <- data.frame(
  sp = c("Russula nigricans", "Amanita", "Boletus edulis"),
  n  = c(10, 5, 8)
)
ggplot(df, aes(y = sp, x = n)) +
  geom_col() +
  theme_minimal() +
  scale_y_italic_species()
```

`select_taxa_pq` *Select taxa in a phyloseq object based on names in a given column of the tax_table*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Usage

```
select_taxa_pq(
  physeq,
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  verbose = TRUE,
  clean_pq = FALSE,
  ...
)
```

Arguments

`physeq`	A phyloseq object
`taxnames`	(optional) A character vector of taxonomic names.
`taxonomic_rank`	(Character, default "currentCanonicalSimple") The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. c("Genus", "Species")).
`verbose`	(logical, default TRUE) If TRUE, prompt some messages.
`clean_pq`	(logical, default FALSE) If TRUE, clean the phyloseq object after subsetting (i.e. remove empty taxa and samples). If FALSE, only empty taxa are removed to take all samples.
`...`	Additional arguments to pass to [subset_taxa_pq()].

Value

A new phyloseq object containing only the selected taxa.

Author(s)

Adrien Taudiere

See Also

[MiscMetabar::subset_taxa_pq()]

Examples

```
## Not run:
data_fungi_mini_cleanNames <- gna_verifier_pq(data_fungi_mini, data_sources = 210)

select_taxa_pq(data_fungi_mini_cleanNames,
  taxonomic_rank = "currentCanonicalSimple",
  taxnames = c("Xylodon flaviporus", "Basidiodendron eyrei"),
  verbose = FALSE,
  clean_pq = FALSE
)

## End(Not run)
select_taxa_pq(data_fungi,
  taxonomic_rank = c("Genus", "Species"),
  taxnames = c("Xylodon flaviporus"), verbose = FALSE, clean_pq = FALSE
)

select_taxa_pq(data_fungi, taxonomic_rank = "Trait", taxnames = c("Soft Rot")) |>
  summary_plot_pq()
```

---

tax_check_ecoregion *Check whether GPS points fall in ecoregions occupied by a set of taxa*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

For each name in ‘taxnames‘ (or for each taxon of a ‘physeq‘ object), checks whether a set of test GPS points lie within a WWF/TNC terrestrial ecoregion that is present in the taxon’s GBIF range. The function is a thin comparison wrapper around [tax_ecoregion_occur()] (for the taxa) and [points_to_ecoregions()] (for the test points).

Usage

```
tax_check_ecoregion(
  physeq = NULL,
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  longitudes,
  latitudes,
```

```
  n_occur = 1000,
  min_nb_occur = 0,
  min_proportion = 0,
  clean_coord = FALSE,
  verbose = TRUE,
  time_to_sleep = 0.3,
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE
)
```

Arguments

physeq	(optional) A phyloseq object. Either ‘physeq‘ or ‘taxnames‘ must be provided, but not both.
taxnames	(optional) A character vector of taxonomic names.
taxonomic_rank	(character, default ‘"currentCanonicalSimple"‘). The column(s) of ‘physeq@tax_table‘ to paste together as taxon names.
longitudes	(numeric vector) Longitudes of the points to test.
latitudes	(numeric vector) Latitudes of the points to test. Must have the same length as ‘longitudes‘.
n_occur	(numeric, default ‘1000‘). Maximum number of occurrences to keep per taxon. With ‘method = "search"‘ this is a server-side limit; with the download methods it is applied as a local sample after import (a warning is issued when a taxon exceeded ‘n_occur‘).
min_nb_occur	(numeric, default ‘0‘). Keep only (taxon, ecoregion) pairs with at least this many occurrences.
min_proportion	(numeric, default ‘0‘). Keep only (taxon, ecoregion) pairs whose share of the taxon’s total occurrences is ‘>= min_proportion‘ (a number in ‘[0, 1]‘). Combined with ‘min_nb_occur‘ via AND.
clean_coord	(logical, default ‘FALSE‘). If ‘TRUE‘, run [CoordinateCleaner::clean_coordinates()] on the result (requires the ‘CoordinateCleaner‘ package).
verbose	(logical, default ‘TRUE‘). If ‘TRUE‘, print progress messages.
time_to_sleep	(numeric, default ‘0.3‘). Seconds to pause between [rgbif::occ_search()] calls to avoid GBIF rate-limiting. Only used when ‘method = "search"‘.
discard_genus_alone	(logical, default ‘TRUE‘ when ‘taxonomic_rank == "currentCanonicalSimple"‘). Passed to [taxonomic_rank_to_taxnames()].
discard_NA	(logical, default ‘TRUE‘). Passed to [taxonomic_rank_to_taxnames()].

Details

The previous positional signature ‘tax_check_ecoregion(taxa_name, lon, lat)‘ is no longer supported: the first argument is now ‘physeq‘. Use ‘tax_check_ecoregion(taxnames = "Sp.", longitudes = lon, latitudes = lat)‘ for single-species calls.

Value

A list with four elements: - ‘taxon_ecoregions‘: the long tibble produced by [tax_ecoregion_occur()]. - ‘points_ecoregion‘: the tibble produced by [points_to_ecoregions()]. - ‘is_in_ecoregion‘: a logical matrix with rownames = taxon names and colnames = ‘"point_<i>"‘, shape ‘n_taxa x n_points‘. ‘TRUE‘ means the ecoregion of the point is among the taxon’s ecoregions that pass ‘min_nb_occur‘ / ‘min_proportion‘. - ‘ecoregion‘: a named list (one named integer vector per taxon) kept for backward compatibility with earlier versions; prefer ‘taxon_ecoregions‘.

Author(s)

Adrien Taudiere

See Also

[tax_ecoregion_occur()], [tax_ecoregion_occur_pq()], [points_to_ecoregions()], [tax_occur_check()]

Examples

```
## Not run:
requireNamespace("rgbif")
res <- tax_check_ecoregion(
  taxnames = "Xylobolus subpileatus",
  longitudes = c(2.3522, 4.2),
  latitudes  = c(48.8566, 33),
  n_occur = 200
)
res$is_in_ecoregion

## End(Not run)
```

`tax_crosscheck_pq` *Cross-check taxonomic names using GBIF backbone and GNA Verifier*

Description

<img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Compares name-verification results from two independent sources:

- **GNA Verifier** (via [taxize::gna_verifier()]) with ‘data_sources = 11‘ (GBIF Backbone Taxonomy)
- **rgbif backbone** (via [rgbif::name_backbone_checklist()])

Because the two services use different matching algorithms and update schedules, discrepancies highlight taxa that may need manual review. A Venn-style summary shows the overlap in matched canonical names.

Usage

```
tax_crosscheck_pq(
  physeq = NULL,
  taxnames = NULL,
  taxonomic_rank = c("Genus", "Species"),
  data_sources = 11,
  plot = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

physeq	(optional) A phyloseq object. Either ‘physeq‘ or ‘taxnames‘ must be provided, but not both.
taxnames	(optional) A character vector of taxonomic names.
taxonomic_rank	Character vector. The column(s) in the ‘@tax_table‘ slot used to construct taxon names when ‘physeq‘ is provided. Default ‘c("Genus", "Species")‘.
data_sources	Integer or character vector passed to [taxize::gna_verifier()]. Default ‘11‘ (GBIF Backbone Taxonomy). Use ‘c(1, 11)‘ to also include Catalogue of Life, for example.
plot	(logical, default ‘TRUE‘). If ‘TRUE‘ and **ggVennDiagram** is installed, a Venn diagram of the two sets of matched canonical names is included in the returned list.
verbose	(logical, default ‘TRUE‘). Print progress messages.
...	Additional arguments passed to [gna_verifier_pq()].

Value

A list with the following elements:

- gna_results: tibble returned by [gna_verifier_pq()] (with ‘add_to_phyloseq = FALSE‘).
- backbone_results: tibble returned by [rgbif::name_backbone_checklist()].
- comparison: data.frame with one row per submitted taxon, columns for the canonical name from each source, and a status column ("match", "mismatch", "gna_only", "backbone_only", or "both_na").
- summary: named numeric vector with counts of each status category.
- venn_plot: (optional) a **ggVennDiagram** object comparing the two sets of matched canonical names.

Author(s)

Adrien Taudière

See Also

[gna_verifier_pq()], [rgbif::name_backbone_checklist()]

Examples

```
## Not run:
# Cross-check a phyloseq object
res <- tax_crosscheck_pq(data_fungi)
res$summary
res$comparison |> filter(status == "mismatch")

res$venn_plot

res_taxref <- tax_crosscheck_pq(data_fungi, data_sources = 12)

# Cross-check a vector of names
res2 <- tax_crosscheck_pq(taxnames = c(
  "Trametopsis brasiliensis",
  "Fake species Waller 2022",
  "Russula"
))
res2$summary

## End(Not run)
```

`tax_ecoregion_occur` *Count GBIF occurrences of taxa in each WWF/TNC terrestrial ecoregion*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

For each name in ‘taxnames‘, retrieves GBIF occurrence coordinates ([tax_gbif_occur_coords()]), maps them to WWF/TNC terrestrial ecoregions in a single spatial join and returns a long tibble with the number and the proportion of occurrences per (taxon, ecoregion). Use [tax_ecoregion_occur_pq()] for the phyloseq wrapper, and [tax_check_ecoregion()] to compare the profile to specific GPS points.

Usage

```
tax_ecoregion_occur(
  taxnames,
  n_occur = 1000,
  method = "search",
  min_nb_occur = 0,
  min_proportion = 0,
  clean_coord = FALSE,
  verbose = TRUE,
  time_to_sleep = 0.3
)
```

Arguments

taxnames	(character vector) Scientific names of the taxa to query.
n_occur	(numeric, default ‘1000‘). Maximum number of occurrences to keep per taxon. With ‘method = "search"‘ this is a server-side limit; with the download methods it is applied as a local sample after import (a warning is issued when a taxon exceeded ‘n_occur‘).
method	(character, default ‘"search"‘). How GBIF occurrences are fetched, passed to [tax_gbif_occur_coords()]. Ecoregion profiling defaults to the credential-free, per-taxon-capped ‘"search"‘ path; set ‘"download"‘ (or ‘"download_sql"‘) to use the Download API (**requires GBIF credentials**).
min_nb_occur	(numeric, default ‘0‘). Keep only (taxon, ecoregion) pairs with at least this many occurrences.
min_proportion	(numeric, default ‘0‘). Keep only (taxon, ecoregion) pairs whose share of the taxon’s total occurrences is ‘>= min_proportion‘ (a number in ‘[0, 1]‘). Combined with ‘min_nb_occur‘ via AND.
clean_coord	(logical, default ‘FALSE‘). If ‘TRUE‘, run [CoordinateCleaner::clean_coordinates()] on the result (requires the ‘CoordinateCleaner‘ package).
verbose	(logical, default ‘TRUE‘). If ‘TRUE‘, print progress messages.
time_to_sleep	(numeric, default ‘0.3‘). Seconds to pause between [rgbif::occ_search()] calls to avoid GBIF rate-limiting. Only used when ‘method = "search"‘.

Value

A tibble with columns ‘taxon_name‘, ‘ECO_NAME‘, ‘biome‘, ‘realm‘, ‘n_occur‘, ‘prop_occur‘. Taxa with zero retrievable occurrences appear once with ‘NA‘ in the ecoregion columns and ‘n_occur = 0L‘, so downstream joins do not silently drop them.

Author(s)

Adrien Taudiere

See Also

[tax_gbif_occur_coords()], [tax_check_ecoregion()], [tax_ecoregion_occur_pq()]

Examples

```
## Not run:
tax_ecoregion_occur(
  c("Xylobolus subpileatus", "Amanita muscaria"),
  n_occur = 200
)

## End(Not run)
```

`tax_ecoregion_occur_pq` *Count GBIF occurrences per ecoregion for the taxa of a phyloseq object*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Phyloseq wrapper around [tax_ecoregion_occur()]. Extracts taxon names from ‘physeq‘ using the column(s) named in ‘taxonomic_rank‘ (default ‘"currentCanonicalSimple"‘, the output of [gna_verifier_pq()]; use ‘"genusSpeciesEpithet"‘ to match the column produced by ‘gna_verifier_pq(..., genus_species_canonical_col = TRUE)‘), then queries GBIF and maps occurrences to WWF/TNC terrestrial ecoregions.

Usage

```
tax_ecoregion_occur_pq(
  physeq = NULL,
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  add_to_phyloseq = NULL,
  col_prefix = NULL,
  n_occur = 1000,
  min_nb_occur = 0,
  min_proportion = 0,
  clean_coord = FALSE,
  verbose = TRUE,
  time_to_sleep = 0.3,
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE
)
```

Arguments

`physeq`	(optional) A phyloseq object. Either ‘physeq‘ or ‘taxnames‘ must be provided, but not both.
`taxnames`	(optional) A character vector of taxonomic names.
`taxonomic_rank`	(character, default ‘"currentCanonicalSimple"‘). The column(s) of ‘physeq@tax_table‘ to paste together as taxon names.
`add_to_phyloseq`	(logical, default ‘TRUE‘ when ‘physeq‘ is provided, ‘FALSE‘ otherwise). If ‘TRUE‘, add three columns (‘<col_prefix>ecoregion_top‘, ‘<col_prefix>ecoregion_n‘, ‘<col_prefix>ecoregion_list‘) to ‘physeq@tax_table‘ and return the updated phyloseq object. If ‘FALSE‘, return the long tibble from [tax_ecoregion_occur()].

`col_prefix` (character, default ‘NULL‘). Prefix for the new tax_table columns. Defaults to ‘"ecoregion_"‘ if ‘NULL‘ (yielding ‘ecoregion_top‘ / ‘ecoregion_n‘ / ‘ecoregion_list‘).

`n_occur` (numeric, default ‘1000‘). Maximum number of occurrences to keep per taxon. With ‘method = "search"‘ this is a server-side limit; with the download methods it is applied as a local sample after import (a warning is issued when a taxon exceeded ‘n_occur‘).

`min_nb_occur` (numeric, default ‘0‘). Keep only (taxon, ecoregion) pairs with at least this many occurrences.

`min_proportion` (numeric, default ‘0‘). Keep only (taxon, ecoregion) pairs whose share of the taxon’s total occurrences is ‘>= min_proportion‘ (a number in ‘[0, 1]‘). Combined with ‘min_nb_occur‘ via AND.

`clean_coord` (logical, default ‘FALSE‘). If ‘TRUE‘, run [CoordinateCleaner::clean_coordinates()] on the result (requires the ‘CoordinateCleaner‘ package).

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

`time_to_sleep` (numeric, default ‘0.3‘). Seconds to pause between [rgbif::occ_search()] calls to avoid GBIF rate-limiting. Only used when ‘method = "search"‘.

`discard_genus_alone` (logical, default ‘TRUE‘ when ‘taxonomic_rank == "currentCanonicalSimple"‘). Passed to [taxonomic_rank_to_taxnames()].

`discard_NA` (logical, default ‘TRUE‘). Passed to [taxonomic_rank_to_taxnames()].

Value

Either a phyloseq object with three new tax_table columns (if ‘add_to_phyloseq = TRUE‘) or the long tibble produced by [tax_ecoregion_occur()] (otherwise). In the latter case, ‘attr(result, "tax_summary")‘ holds the one-row-per-taxon summary used to build the phyloseq columns.

Author(s)

Adrien Taudiere

See Also

[tax_ecoregion_occur()], [tax_check_ecoregion()], [taxonomic_rank_to_taxnames()]

Examples

```
## Not run:
data_fungi_mini_clean <- gna_verifier_pq(data_fungi_mini)
tax_ecoregion_occur_pq(
  data_fungi_mini_clean,
  taxonomic_rank = "genusSpeciesEpithet",
  n_occur = 100
)

## End(Not run)
```

tax_gbif_alt *Get altitude range statistics for each taxa from GBIF*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Retrieve altitude/elevation statistics (minimum, maximum, 5 mean and standard deviation) for taxa from GBIF occurrence data.

Two methods are available: - **"gbif"** (default): Uses GBIF's Download API (‘occ_download()‘) to retrieve occurrence records with non-null elevation values. This is the recommended approach by GBIF for research purposes. **Requires GBIF credentials.** - **"elevatr"**: Computes elevation from GPS coordinates retrieved from GBIF using AWS Terrain Tiles via the ‘elevatr‘ package. This provides more complete coverage for occurrences that lack elevation data but requires the ‘elevatr‘ and ‘rnaturalearth‘ packages.

Usage

```
tax_gbif_alt(
  physeq = NULL,
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  add_to_phyloseq = NULL,
  col_prefix = NULL,
  method = c("gbif", "elevatr"),
  elev_zoom = 5,
  n_coor_alt = NULL,
  verbose = TRUE,
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE
)
```

Arguments

physeq	(optional) A phyloseq object. Either ‘physeq‘ or ‘taxnames‘ must be provided, but not both.
taxnames	(optional) A character vector of taxonomic names.
taxonomic_rank	(Character, default "currentCanonicalSimple") The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. c("Genus", "Species")).
add_to_phyloseq	(logical, default TRUE when physeq is provided, FALSE when taxnames is provided) If TRUE, add new column(s) in the tax_table of the phyloseq object. Automatically set to TRUE when a phyloseq object is provided and FALSE when taxnames is provided. Cannot be TRUE if ‘taxnames‘ is provided.

`col_prefix` A character string to be added as a prefix to the new columns names added to the tax_table slot of the phyloseq object (default: NULL).

`method` (character, default "gbif") Method to retrieve elevation data: - "gbif": Use GBIF's Download API with ‘pred_notnull("elevation")‘ to retrieve only records with elevation data. This is the recommended approach by GBIF for research. **Requires GBIF credentials** (see Details). - "elevatr": Compute elevation from GPS coordinates using AWS Terrain Tiles. Requires the ‘elevatr‘ and ‘rnaturalearth‘ packages.

`elev_zoom` (numeric, default 5) Zoom level for AWS Terrain Tiles. Only used when ‘method = "elevatr"‘. Higher values give finer resolution but are slower. Range: 1-14. See [elevatr::get_elev_point()] for details.

`n_coor_alt` (int, default NULL) Number of occurrences to samples. If left to NULL, all occurrences are used to computed the altitute. It allow quicker computation when using method "elevatr" on taxa with a large number of occurrences.

`verbose` (logical, default TRUE) If TRUE, prompt some messages.

`discard_genus_alone` (logical, default ‘TRUE‘ when ‘taxonomic_rank == "currentCanonicalSimple"‘). Passed to [taxonomic_rank_to_taxnames()].

`discard_NA` (logical, default ‘TRUE‘). Passed to [taxonomic_rank_to_taxnames()].

Details

Method "gbif" (default)

This method uses GBIF's Download API via ‘rgbif::occ_download()‘ with the following predicates: - ‘pred_in("taxonKey", gbif_taxon_keys)‘ - Filter by taxon keys - ‘pred("hasCoordinate", TRUE)‘ - Only records with coordinates - ‘pred("hasGeospatialIssue", FALSE)‘ - Exclude records with geospatial issues - ‘pred_notnull("elevation")‘ - Only records with elevation data

This is the recommended approach by GBIF for research purposes as it provides citable downloads with DOIs.

GBIF credentials are required. You must: 1. Register at <https://www.gbif.org/user/register> 2. Store credentials in your ‘.Renviron‘ file: - ‘GBIF_USER=your_username‘ - ‘GBIF_PWD=your_password‘ - ‘GBIF_EMAIL=your_email‘ 3. See <https://docs.ropensci.org/rgbif/reference/occ_download.html> for more details.

Note: Downloads are asynchronous and may take some time to complete.

Method "elevatr"

This method retrieves GPS coordinates from GBIF occurrence records and computes elevation using AWS Terrain Tiles via the ‘elevatr‘ package. This provides more complete coverage than relying on GBIF's elevation field.

Ocean points are detected using land boundaries from ‘rnaturalearth‘ and are reported in the ‘altitude_n_ocean‘ column. A warning is issued if ocean points are detected for a taxon.

Please cite ‘rgbif‘ package. When using method "elevatr", also cite ‘elevatr‘ and ‘rnaturalearth‘ packages.

Value

Either a tibble (if add_to_phyloseq = FALSE) or a new phyloseq object, if add_to_phyloseq = TRUE, with new column(s) in the tax_table. The returned data includes: altitude_min, altitude_max, altitude_q05, altitude_q50, altitude_q95, altitude_mean, altitude_sd, altitude_n_records, and canonicalName. When ‘method = "elevatr"‘, also includes altitude_n_ocean (number of points detected in ocean).

Author(s)

Adrien Taudiere

See Also

[rgbif::occ_download()], [elevatr::get_elev_point()], [tax_gbif_occur_pq()], [plot_tax_gbif_pq()]

Examples

```
## Not run:
data_fungi_mini_cleanNames <-
  gna_verifier_pq(data_fungi_mini)

# Get altitude range statistics using GBIF Download API (default)
# Note: Requires GBIF credentials (GBIF_USER, GBIF_PWD, GBIF_EMAIL)
# Register at https://www.gbif.org/user/register
data_fungi_mini_alt <- tax_gbif_alt(data_fungi_mini_cleanNames,
  add_to_phyloseq = FALSE
)

# Using taxnames vector (returns a tibble)
altitude_gbif <- tax_gbif_alt(
  taxnames = c("Amanita muscaria", "Boletus edulis")
)

# Use elevatr method to compute elevation from GPS coordinates
# (provides more coverage, no GBIF credentials needed)
altitude_elevatr <- tax_gbif_alt(
  taxnames = c("Amanita muscaria"),
  method = "elevatr",
  n_coor_alt = 100,
  verbose = FALSE
)

# Add altitude data to phyloseq object
data_fungi_mini_with_alt <- tax_gbif_alt(data_fungi_mini_cleanNames)

data_fungi_mini_with_alt@tax_table |>
  as.data.frame() |>
  tibble() |>
  filter(as.numeric(altitude_n_records) > 100) |>
  distinct(taxa_name, .keep_all = TRUE) |>
  ggplot(aes(y = as.numeric(altitude_mean), x = taxa_name, fill = Guild)) +
```

```
  geom_col() +
  coord_flip() +
  geom_errorbar(
    aes(ymin = as.numeric(altitude_q05), ymax = as.numeric(altitude_q95)),
    width = 0.2
  ) +
  geom_label(aes(label = paste0("n=", altitude_n_records)), size = 2) +
  labs(
    title = "Mean altitude with 5%-95% quantiles (only taxa with >100 records)",
    subtitle = "Labels depict the number of gbif records with altitude data, \n
    color depict ecological Guild",
    x = "Taxa names",
    fill = "Guild"
  ) +
  theme(legend.position = "bottom")

## End(Not run)
```

`tax_gbif_occur_coords` *Get GBIF occurrence coordinates for a vector of taxa*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Retrieves georeferenced GBIF occurrences for each name in ‘taxnames‘ and returns them as a long tibble. Taxa are resolved to GBIF usage keys once via [rgbif::name_backbone_checklist()] (filtering on ‘matchType with one of three methods (see ‘method‘). Rows with missing coordinates are dropped.

Usage

```
tax_gbif_occur_coords(
  taxnames,
  n_occur = 1000,
  method = c("download", "download_sql", "search"),
  country = NULL,
  year_gte = NULL,
  year_lte = NULL,
  geometry = NULL,
  clean_coord = FALSE,
  verbose = TRUE,
  time_to_sleep = 0.3
)
```

Arguments

`taxnames`	(character vector) Scientific names of the taxa to query.
`n_occur`	(numeric, default ‘1000‘). Maximum number of occurrences to keep per taxon. With ‘method = "search"‘ this is a server-side limit; with the download methods it is applied as a local sample after import (a warning is issued when a taxon exceeded ‘n_occur‘).
`method`	(character, default ‘"download"‘). How occurrences are fetched: - ‘"download"‘: a single [rgbif::occ_download()] request for all taxa at once (no 100,000-record cap, mints a citable DOI). **Requires GBIF credentials** (see [check_gbif_credentials()]). - ‘"download_sql"‘: [rgbif::occ_download_sql()] with server-side column selection and ‘WHERE‘ filtering (gated preview, must be enabled for your account). **Requires GBIF credentials.** Because GBIF SQL ‘taxonkey‘ is not hierarchical, this method matches ‘taxonkey‘/‘specieskey‘ directly and may under-return records for names matched at a higher rank (‘HIGHERRANK‘); use ‘"download"‘ if you need full hierarchical coverage. - ‘"search"‘: the legacy per-taxon [rgbif::occ_search()] loop (fast, capped at 100,000 records, no credentials).
`country`	(character, default ‘NULL‘). Optional ISO2 country code used as a server-side filter for the download methods (e.g. ‘"FR"‘).
`year_gte, year_lte`	(numeric, default ‘NULL‘). Optional inclusive year bounds used as server-side filters for the download methods.
`geometry`	(character, default ‘NULL‘). Optional WKT polygon used as a server-side spatial filter for ‘method = "download"‘ (via [rgbif::pred_within()]). Not supported with ‘method = "download_sql"‘.
`clean_coord`	(logical, default ‘FALSE‘). If ‘TRUE‘, run [CoordinateCleaner::clean_coordinates()] on the result (requires the ‘CoordinateCleaner‘ package).
`verbose`	(logical, default ‘TRUE‘). If ‘TRUE‘, print progress messages.
`time_to_sleep`	(numeric, default ‘0.3‘). Seconds to pause between [rgbif::occ_search()] calls to avoid GBIF rate-limiting. Only used when ‘method = "search"‘.

Value

A tibble with columns ‘taxon_name‘, ‘usageKey‘, ‘decimalLongitude‘, ‘decimalLatitude‘, ‘countryCode‘, ‘year‘, ‘gbifID‘. Taxa with zero valid occurrences are listed in ‘attr(result, "missing_taxa")‘.

Author(s)

Adrien Taudiere

See Also

[tax_ecoregion_occur()], [rgbif::occ_download()], [rgbif::occ_download_sql()], [rgbif::occ_search()]

Examples

```
## Not run:
# Default: GBIF Download API (requires GBIF_USER, GBIF_PWD, GBIF_EMAIL)
tax_gbif_occur_coords(
  c("Xylobolus subpileatus", "Amanita muscaria"),
  n_occur = 200
)

# Narrow the download server-side to reduce transfer
tax_gbif_occur_coords(
  c("Amanita muscaria"),
  country = "FR",
  year_gte = 2000
)

# Legacy fast path (no credentials, capped at 100,000 records)
tax_gbif_occur_coords(
  c("Xylobolus subpileatus"),
  method = "search",
  n_occur = 200
)

## End(Not run)
```

---

tax_gbif_occur_pq *Get number of occurrences for each taxa of a phyloseq object*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

A wrapper of [rgbif::occ_search()] function to get the number of occurences. Optionally, the number of occurrences can be obtained by years or by country.

Usage

```
tax_gbif_occur_pq(
  physeq = NULL,
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  add_to_phyloseq = NULL,
  col_prefix = NULL,
  by_country = FALSE,
  by_years = FALSE,
  verbose = TRUE,
  time_to_sleep = 0.3,
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE
)
```

Arguments

physeq	(optional) A phyloseq object. Either ‘physeq‘ or ‘taxnames‘ must be provided, but not both.
taxnames	(optional) A character vector of taxonomic names.
taxonomic_rank	(Character, default "currentCanonicalSimple") The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. c("Genus", "Species")).
add_to_phyloseq	(logical, default TRUE when physeq is provided, FALSE when taxnames is provided) If TRUE, add new column(s) in the tax_table of the phyloseq object. Automatically set to TRUE when a phyloseq object is provided and FALSE when taxnames is provided. Cannot be TRUE if ‘taxnames‘ is provided.
col_prefix	A character string to be added as a prefix to the new columns names added to the tax_table slot of the phyloseq object (default: NULL).
by_country	(logical, default FALSE) If TRUE, the number of occurences is computed by country
by_years	(logical, default FALSE) If TRUE, the number of occurences is computed by years
verbose	(logical, default TRUE) If TRUE, prompt some messages.
time_to_sleep	(numeric, default 0.3) Time to sleep between two calls to rgbif::occ_search(). Useful to avoid to be blocked by GBIF. Try to increase this value if you are blocked by the error "To download GBIF occurrence data in bulk, please request..."
discard_genus_alone	(logical, default ‘TRUE‘ when ‘taxonomic_rank == "currentCanonicalSimple"‘). Passed to [taxonomic_rank_to_taxnames()].
discard_NA	(logical, default ‘TRUE‘). Passed to [taxonomic_rank_to_taxnames()].

Details

This function is mainly a wrapper of the work of others. Please cite ‘rgbif‘ package.

Value

Either a tibble (if add_to_phyloseq = FALSE) or a new phyloseq object, if add_to_phyloseq = TRUE, with new column(s) in the tax_table.

Author(s)

Adrien Taudiere

See Also

[rgbif::occ_search()], [plot_tax_gbif_pq()], [tax_occurr_pq()]

Examples

```
## Not run:
data_fungi_mini_cleanNames <-
  gna_verifier_pq(data_fungi_mini)

data_fungi_mini_cleanNames <- tax_gbif_occur_pq(data_fungi_mini_cleanNames, by_country = TRUE)

# Get data without adding to phyloseq
tax_gbif_occur_pq(data_fungi_mini_cleanNames, add_to_phyloseq = FALSE)
tax_gbif_occur_pq(data_fungi_mini_cleanNames, by_years = TRUE, add_to_phyloseq = FALSE)

# Using taxnames vector (returns a tibble)
tax_gbif_occur_pq(taxnames = c("Amanita muscaria", "Boletus edulis"))
ggplot(
  data_fungi_mini_cleanNames@tax_table,
  aes(y = log10(as.numeric(Global_occurences)), x = currentCanonicalSimple)
) +
  geom_col() +
  geom_col(aes(y = -log10(as.numeric(FR))), fill = "blue") +
  coord_flip() +
  xlab("Number of occurences (log10 scale) at global (grey) scale and in France (blue)")

## End(Not run)
```

`tax_get_wk_info_pq` *Retrieve information about taxa from wikipedia*

Description

<img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Usage

```
tax_get_wk_info_pq(
  physeq = NULL,
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  add_to_phyloseq = NULL,
  col_prefix = NULL,
  verbose = TRUE,
  languages_pages = NULL,
  time_to_sleep = 0.3,
  summarize_function_length = "mean",
  summarize_function_views = "sum",
  n_days = 30,
  start_date = NULL,
  end_date = NULL,
```

```
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE
)
```

Arguments

physeq	(optional) A phyloseq object. Either ‘physeq‘ or ‘taxnames‘ must be provided, but not both.
taxnames	(optional) A character vector of taxonomic names.
taxonomic_rank	(Character, default = "currentCanonicalSimple") The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. the c("Genus", "Species")).
add_to_phyloseq	(logical, default TRUE when physeq is provided, FALSE when taxnames is provided) If TRUE, a new phyloseq object is returned with new columns in the tax_table. Automatically set to TRUE when a phyloseq object is provided and FALSE when taxnames is provided. Cannot be TRUE if ‘taxnames‘ is provided.
col_prefix	A character string to be added as a prefix to the new columns names added to the tax_table slot of the phyloseq object (default: NULL).
verbose	(logical, default TRUE) If TRUE, prompt some messages.
languages_pages	(Character vector or NULL, default NULL) If not NULL, only the languages present in this vector will be queried. The language codes are the two- or three-letter codes defined by ISO 639-1. For example, c("en", "fr", "de") will query only the English, French and German Wikipedia pages. If NULL (default), all languages will be queried. See https://en.wikipedia.org/wiki/List_of_Wikipedias for the list of language codes. Note that some taxa may not have pages in the specified languages. In this case, the function will return NA for these taxa.
time_to_sleep	(numeric, default 0.3) Time to sleep between two calls to wikipedia API.
summarize_function_length	A function to summarize the page length across languages. Default is "mean".
summarize_function_views	A function to summarize the page views across languages. Default is "sum".
n_days	(numeric, default 30) Number of days to consider for the page views.
start_date	The start date for the page views. If NULL (default), the start date is set to ’n_days’ before the end date. Passed to [tax_get_wk_pages_info()].
end_date	The end date for the page views. If NULL (default), the end date is set to yesterday’s date. Passed to [tax_get_wk_pages_info()].
discard_genus_alone	(logical, default ‘TRUE‘ when ‘taxonomic_rank == "currentCanonicalSimple"‘). Passed to [taxonomic_rank_to_taxnames()].
discard_NA	(logical, default ‘TRUE‘). Passed to [taxonomic_rank_to_taxnames()].

Details

This is a very brut/raw approach of the notion of cultural keystone species (see Mattalia et al. 2025, https://doi.org/10.1002/pan3.10653 for a review of the concept). Taxa with only genus name are discarded.

Value

Either a tibble (if add_to_phyloseq = FALSE) or a new phyloseq object, if add_to_phyloseq = TRUE, with new column(s) in the tax_table. The tibble contains the following columns: - 'lang': Number of languages in which the taxon has a wikipedia page - 'page_length': Mean length of the wikipedia pages (in characters) - 'page_views': Total number of page views over the last 'n_days' days - 'taxon_id': Wikidata taxon identifier (e.g. "Q10723171" for Stereum ostrea) - 'taxa_name': Taxonomic name used to query wikipedia

See Also

[tax_get_wk_lang()], [tax_get_wk_pages_info()], [tax_photo_pq()]

Examples

```
## Not run:
data_fungi_mini_cleanNames <- gna_verifier_pq(data_fungi_mini, data_source = 210)

wk_info <- tax_get_wk_info_pq(subset_taxa_pq(
  data_fungi_mini_cleanNames,
  taxa_sums(data_fungi_mini_cleanNames@otu_table) > 20000
))

data_fungi_mini_cleanNames_wk_info <-
  tax_get_wk_info_pq(data_fungi_mini_cleanNames)

subset_taxa(data_fungi_mini_cleanNames_wk_info, !is.na(page_views)) |>
  tax_table() |>
  as.data.frame() |>
  distinct(currentCanonicalSimple, .keep_all = TRUE) |>
  ggplot(
    aes(
      x = log10(as.numeric(page_views) + 1),
      y = forcats::fct_reorder(currentCanonicalSimple, as.numeric(page_views)),
      col = Order
    )
  ) +
  geom_segment(aes(
    x = 0, xend = log10(as.numeric(page_views) + 1),
    y = currentCanonicalSimple, yend = currentCanonicalSimple
  ), linewidth = 0.4) +
  geom_point(aes(size = as.numeric(page_length)), shape = 15) +
  geom_text(aes(label = lang), size = 2, color = "black") +
 xlab("Page views log-10 transformed. Number denoted the number of language in #' wikipedia.
      Shape size is proportional to mean page lenght in wikipedia.") +
  ylab("")
```

```
## End(Not run)
```

`tax_get_wk_lang` *Retrieve the wikipedia pages for a given Wikidata taxon identifier*

Description

<img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Filter only wikipedia page link to a language with a two- or three-letter code defined by ISO 639-1 or ISO 639-3 (e.g. "en" for English, "fr" for French, "de" for German). We also add a list of more-than-three-letter codes for some languages: c("zh-yue", "nds-nl", "ru-sib", "bat-smg", "fiu-vro", "roa-rup", "map-bms", "cbk-zam", "roa-tara", "tokipona", "be-tarask", "zh-min-nan", "zh-classical"))

Usage

```
tax_get_wk_lang(taxon_id, languages_pages = NULL)
```

Arguments

`taxon_id`	(Character string, required) The Wikidata taxon identifier (e.g. "Q10723171" for Xylobolus subpileatus)
`languages_pages`	(Character vector) If not NULL, only the languages present in this vector will be queried.

Value

A tibble with three columns: "title", "site" and "lang". NA values are returned if wikipedia api return a response different from 200 or if the taxon_id is set to NA or "". If no wikipedia page is found in the all languages, a tibble with 0 is returned.

See Also

[tax_get_wk_pages_info()], [tax_get_wk_info_pq()], [tax_photo_pq()]

Examples

```
tax_get_wk_lang("Q10723171")
tax_get_wk_lang("Q10723171") |>
  nrow()

tax_get_wk_lang("Q10723171")
```

`tax_get_wk_pages_info` *Retrieve information about wikipedia pages for a given taxon id*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Input can be either a taxon_id (Wikidata taxon identifier) or a tibble as returned by [tax_get_wk_lang()].

Usage

```
tax_get_wk_pages_info(
  taxon_id = NULL,
  tib_list = NULL,
  languages_pages = NULL,
  time_to_sleep = 0.3,
  summarize_function_length = "mean",
  summarize_function_views = "sum",
  n_days = 30,
  start_date = NULL,
  end_date = NULL,
  verbose = FALSE
)
```

Arguments

`taxon_id`	(Character string, required) The Wikidata taxon identifier (e.g. "Q10723171" for Xylobolus subpileatus)
`tib_list`	A tibble as returned by [tax_get_wk_lang()] with columns "title", "site" and "lang".
`languages_pages`	(Character vector) If not NULL, only the languages present in this vector will be queried. The language codes are the two- or three-letter codes defined by ISO 639-1. For example, c("en", "fr", "de") will query only the English, French and German Wikipedia pages.
`time_to_sleep`	(numeric, default 0.3) Time to sleep between two calls to wikipedia API.
`summarize_function_length`	A function to summarize the page length across languages. Default is "mean". Other options can be "sum", "median", "max", "min", etc.
`summarize_function_views`	A function to summarize the page views across languages. Default is "sum". Other options can be "mean", "median", "max", "min", etc.
`n_days`	(numeric, default 30) Number of days to consider for the page views.
`start_date`	The start date for the page views. If NULL (default), the start date is set to 'n_days' before the end date.

end_date	The end date for the page views. If NULL (default), the end date is set to yesterday's date.
verbose	(logical, default TRUE) If TRUE, prompt some messages.

Value

A list with two elements: - 'page_length': Mean length of the wikipedia pages (in characters) - 'page_views': Total number of page views over the last 'n_days' days

Author(s)

Adrien Taudiere

See Also

[tax_get_wk_lang()], [tax_get_wk_info_pq()], [tax_photo_pq()]

Examples

```
## Not run:
tax_get_wk_pages_info("Q10723171")
tax_get_wk_pages_info("Q10723171", languages_pages = c("fr", "en"))
tax_get_wk_pages_info("Q10723171", languages_pages = c("fr"))

pages_Q10723171 <- tax_get_wk_lang("Q10723171")
tax_get_wk_pages_info(tib_list = pages_Q10723171)
tax_get_wk_pages_info(
  tib_list = pages_Q10723171,
  summarize_function_length = "sum"
)
tax_get_wk_pages_info(
  tib_list = pages_Q10723171,
  summarize_function_length = "sum",
  n_days = 365
)

tax_get_wk_pages_info(
  tib_list = pages_Q10723171,
  start_date = "2023-01-01",
  end_date = "2023-12-31"
)

## End(Not run)
```

tax_globi_pq *Get biotic interactions for taxa present in a phyloseq object using rglobi*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

A wrapper of [rglobi::get_interactions_by_taxa()] function to get biotic interactions for each taxa of a phyloseq object

Usage

```
tax_globi_pq(
  physeq = NULL,
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  discard_synonym = TRUE,
  add_to_phyloseq = NULL,
  col_prefix = NULL,
  interaction_types = NULL,
  valid_taxo_target_taxon = TRUE,
  add_target_canonical = TRUE,
  data_sources = c(1, 12),
  verbose = FALSE,
  strict_interaction_types = TRUE,
  max_interactions = 1000,
  batch_size_gna_verifier = 50,
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE
)
```

Arguments

physeq	(optional) A phyloseq object. Either ‘physeq‘ or ‘taxnames‘ must be provided, but not both.
taxnames	(optional) A character vector of taxonomic names.
taxonomic_rank	(Character, default "currentCanonicalSimple") The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. c("Genus", "Species")).
discard_synonym	(logical, default TRUE) If TRUE, discard interactions where the source_taxon_name is a synonym of the taxon name used to query
add_to_phyloseq	(logical, default TRUE when physeq is provided, FALSE when taxnames is provided) If TRUE, return a new phyloseq object with new columns in the tax_table

slot. If FALSE, return a tibble with the interactions found for each taxon. Automatically set to TRUE when a phyloseq object is provided and FALSE when taxnames is provided. Cannot be TRUE if ‘taxnames‘ is provided.

`col_prefix` A character string to be added as a prefix to the new columns names added to the tax_table slot of the phyloseq object (default: NULL).

`interaction_types`
A character vector of interaction types to query. See [rglobi::get_interaction_types()]. If NULL (default), all interaction types are queried.

`valid_taxo_target_taxon`
(logical, default TRUE) If TRUE, verify the scientific names of the target_taxon_name using [taxize::gna_verifier()] function and keep only valid names.

`add_target_canonical`
(logical, default TRUE) If TRUE, add a column ‘target_taxon_Canonical‘ with the current accepted name (resolve the synonymie) of the target_taxon_name using [taxize::gna_verifier()] function.

`data_sources` A character or integer vector with numbers corresponding to data sources. See the Global Names Architecture documentation for a list of available options.

`verbose` (logical, default FALSE) If TRUE, prompt some messages.

`strict_interaction_types`
(logical, default TRUE) If TRUE, keep only interactions exactly matching the interaction_types provided. If FALSE, keep all interactions returned by rglobi for the queried taxon. For exemple, rglobi for interaction_types = "hasHost" will also return interactions with interaction_type = "pathogenOf" and "parasiteOf" if strict_interaction_types is set to FALSE.

`max_interactions`
(numeric, default 1000) The maximum number of interactions to query for each taxon.

`batch_size_gna_verifier`
(numeric, default 100) The number of names to verify at once with’ [taxize::gna_verifier()] function. Its a hack because gna_verifier seems to fail when too many names are sent at once including strange ones such as what is obtain whith rglobi. Only used if ‘valid_taxo_target_taxon‘ is set to TRUE.

`discard_genus_alone`
(logical, default ‘TRUE‘ when ‘taxonomic_rank == "currentCanonicalSimple"‘). Passed to [taxonomic_rank_to_taxnames()].

`discard_NA` (logical, default ‘TRUE‘). Passed to [taxonomic_rank_to_taxnames()].

Details

This function is mainly a wrapper of the work of others. Please cite ‘rglobi‘ and ‘taxize‘ packages.

Value

Either a tibble (if add_to_phyloseq = FALSE) or a new phyloseq object, if add_to_phyloseq = TRUE, with new column(s) in the tax_table.

Author(s)

Adrien Taudiere

Examples

```
## Not run:
data_fungi_mini_cleanNames <- gna_verifier_pq(data_fungi_mini,
  data_sources = 210
)

data_fungi_mini_cleanNames <- tax_globi_pq(data_fungi_mini_cleanNames,
  interaction_types = c("hasHost")
)

res_globi <- tax_globi_pq(data_fungi_mini,
  taxonomic_rank = c("Genus", "Species"),
  interaction_types = list("parasiteOf", "hasHost"),
  verbose = TRUE,
  max_interactions = 10
)

## End(Not run)
```

---

tax_info_pq *Get information from a custom csv file using taxonomic names present in a phyloseq object*

Description

 <img src="https://img.shields.io/badg
maturing-blue" alt="lifecycle-maturing">

A function to add information from a custom csv file (e.g. FungalTraits, Taxref, ...) to the tax_table slot of a phyloseq object by joining taxonomic names from phyloseq object (column 'taxonomic_rank') with a column of the csv file ('csv_taxonomic_rank') containing the correspondant taxonomic names. Be carefull that the taxonomic names in the csv file must match exactly the taxonomic names in the phyloseq object. For example, if the taxonomic names in the phyloseq object are in the form "Genus species" the taxonomic names in the csv file must be in the same form (not "Genus_species" or "Genus Species Author"...).

Note that the csv file need to be in a wide-format, i.e. one line for each distinct value in the 'csv_taxonomic_rank' columns. You may want to transform your data.frame using [tidyr::tidyr::pivot_wider()] fonctions prior to write it in a new file.

Usage

```
tax_info_pq(
  physeq = NULL,
```

```
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  file_name = NULL,
  csv_taxonomic_rank = NULL,
  add_to_phyloseq = NULL,
  col_prefix = NULL,
  use_duck_db = FALSE,
  csv_cols_select = NULL,
  sep = ",",
  dec = ".",
  verbose = TRUE,
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE
)
```

Arguments

physeq	(optional) A phyloseq object. Either ‘physeq‘ or ‘taxnames‘ must be provided, but not both.
taxnames	(optional) A character vector of taxonomic names.
taxonomic_rank	(Character, default "currentCanonicalSimple") The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. c("Genus", "Species")).
file_name	(required) A file path to your csv file.
csv_taxonomic_rank	(required) The name of the column in your csv file containing the taxonomic names. Must match the taxonomic_rank of the phyloseq.
add_to_phyloseq	(logical, default TRUE when physeq is provided, FALSE when taxnames is provided) If TRUE, add new column(s) in the tax_table of the phyloseq object. Automatically set to TRUE when a phyloseq object is provided and FALSE when taxnames is provided. Cannot be TRUE if ‘taxnames‘ is provided.
col_prefix	A character string to be added as a prefix to the new columns names added to the tax_table slot of the phyloseq object.
use_duck_db	(logical, default FALSE) If TRUE, use duckdb to handle the join between the csv file and the tax_table of the phyloseq object. Useful for large csv files.
csv_cols_select	A character vector of the column names to select in the csv file.
sep	the field separator character. See [utils::read.csv()].
dec	the field separator character. See [utils::read.csv()].
verbose	(logical, default TRUE) If TRUE, prompt some messages.
discard_genus_alone	(logical, default ‘TRUE‘ when ‘taxonomic_rank == "currentCanonicalSimple"‘). Passed to [taxonomic_rank_to_taxnames()].
discard_NA	(logical, default ‘TRUE‘). Passed to [taxonomic_rank_to_taxnames()].

Value

Either a tibble (if add_to_phyloseq = FALSE) or a new phyloseq object, if add_to_phyloseq = TRUE, with new column(s) in the tax_table.

Author(s)

Adrien Taudiere

Examples

```
## Not run:
data_fungi_cleanNames <- gna_verifier_pq(data_fungi,
  data_sources = 210
)

# FUNGAL TRAITS example
# --------------------
fungal_traits <- system.file("extdata", "fun_trait_mini.csv",
  package = "taxinfo") # minidataset for testing
# fungal_traits <- system.file("extdata", "fungal_traits.csv", package = "taxinfo")
fg_traits <- tax_info_pq(data_fungi_cleanNames,
  taxonomic_rank = "genusEpithet",
  file_name = fungal_traits,
  csv_taxonomic_rank = "GENUS",
  col_prefix = "ft_",
  sep = "\t",
  add_to_phyloseq = FALSE
)

table(fg_traits$ft_primary_lifestyle, fg_traits$Guild) |>
  as.data.frame() |>
  filter(Freq > 0) |>
  arrange(desc(Freq)) |>
  head()

# TAXREF example
# --------------------
TAXREFv18_fungi <- system.file("extdata", "TAXREFv18_fungi.csv", package = "taxinfo")

res_with_R <- tax_info_pq(data_fungi_cleanNames,
  file_name = TAXREFv18_fungi,
  csv_taxonomic_rank = "NOM_VALIDE_SIMPLE",
  col_prefix = "taxref_"
)
res_with_duckDB <- tax_info_pq(
  data_fungi_cleanNames,
  file_name = TAXREFv18_fungi,
  csv_taxonomic_rank = "NOM_VALIDE_SIMPLE",
  use_duck_db = TRUE,
  add_to_phyloseq = FALSE,
  col_prefix = "taxref_",
  csv_cols_select = c(
```

```
    "RANG", "HABITAT", "FR", "GF", "MAR", "GUA", "SM", "SB",
    "SPM", "MAY", "EPA", "REU", "SA", "TA", "TAAF", "PF", "NC", "WF", "CLI", "URL"
  )
)

data_fungi_cleanNames_2 <- tax_info_pq(
  data_fungi_cleanNames,
  file_name = TAXREFv18_fungi,
  csv_taxonomic_rank = "NOM_VALIDE_SIMPLE",
  use_duck_db = TRUE,
  col_prefix = "taxref_",
  csv_cols_select = c("RANG", "HABITAT", "FR", "URL", "CD_REF")
)
table(data_fungi_cleanNames_2@tax_table[, "taxref_FR"])
table(data_fungi_cleanNames_2@tax_table[, "taxref_HABITAT"])

# TAXREF example (with status)
# --------------------

taxref_status <- system.file("extdata", "bdc_18_01_wider_mini.csv", package = "taxinfo")
data_fungi_cleanNames_3 <- tax_info_pq(data_fungi_cleanNames_2,
  taxonomic_rank = "taxref_CD_REF",
  file_name = taxref_status,
  csv_taxonomic_rank = "CD_REF",
  col_prefix = "st_",
  use_duck_db = TRUE
)

data_fungi_cleanNames_3@tax_table[, "st_BCD_LRR"] |>
  table(useNA = "always")
data_fungi_cleanNames_3@tax_table[, "st_BCD_ZDET"] |>
  table(useNA = "always")
data_fungi_cleanNames_3@tax_table[, "st_BCD_TAXREF_STATUT_BIOGEO"] |>
  table(useNA = "always")

#' # EPPO (Pest species) example (https://gd.eppo.int/)
# --------------------
# You can visit https://gd.eppo.int/ to download database for other countries
# than France
EPPO_FR <- system.file("extdata", "EPPO_regulated_FR.csv", package = "taxinfo")

res_with_EPPO_FR <- tax_info_pq(data_fungi_cleanNames,
  file_name = EPPO_FR,
  csv_taxonomic_rank = "organism_prefname",
  col_prefix = "EPPO_"
)

res_with_EPPO_FR@tax_table |>
  as.data.frame() |>
  filter(!is.na(EPPO_qlistlabel))

## End(Not run)
```

tax_iucn_code_pq *Get iucn conservation status through gbif*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Usage

```
tax_iucn_code_pq(
  physeq = NULL,
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  add_to_phyloseq = NULL,
  col_prefix = NULL,
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE
)
```

Arguments

physeq	(optional) A phyloseq object. Either ‘physeq‘ or ‘taxnames‘ must be provided, but not both.
taxnames	(optional) A character vector of taxonomic names.
taxonomic_rank	(Character, default "currentCanonicalSimple") The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. c("Genus", "Species")).
add_to_phyloseq	(logical, default TRUE when physeq is provided, FALSE when taxnames is provided) If TRUE, add a new column (iucn_code) in the tax_table of the phyloseq object. Automatically set to TRUE when a phyloseq object is provided and FALSE when taxnames is provided. Cannot be TRUE if ‘taxnames‘ is provided.
col_prefix	A character string to be added as a prefix to the new columns names added to the tax_table slot of the phyloseq object (default: NULL).
discard_genus_alone	(logical, default ‘TRUE‘ when ‘taxonomic_rank == "currentCanonicalSimple"‘). Passed to [taxonomic_rank_to_taxnames()].
discard_NA	(logical, default ‘TRUE‘). Passed to [taxonomic_rank_to_taxnames()].

Details

This function is mainly a wrapper of the work of others. Please cite ‘rgbif‘ package.

Value

Either a tibble (if add_to_phyloseq = FALSE) or a new phyloseq object, if add_to_phyloseq = TRUE, with 1 new column (iucn_code) in the tax_table.

Author(s)

Adrien Taudiere

See Also

[tax_info_pq()], [rgbif::name_usage()]

Examples

```
## Not run:

data_fungi_mini_cleanNames <-
  gna_verifier_pq(data_fungi_mini) |>
  tax_iucn_code_pq()

table(data_fungi_mini_cleanNames@tax_table[, "iucn_code"])

# Using taxnames vector (returns a tibble)
tax_iucn_code_pq(taxnames = c("Amanita muscaria", "Boletus edulis"))

## End(Not run)
```

---

tax_oa_pq *Get scientific works about taxa present in a phyloseq object*

Description

<img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

A wrapper of [openalexR::oa_fetch()] function to get the number of scientific works (and a list of doi if count_only is set to FALSE) for each taxa of a phyloseq object. Each taxa name is searched in the title and abstract of the works present in Open Alex database.

Usage

```
tax_oa_pq(
  physeq = NULL,
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  count_only = FALSE,
  return_raw_oa = FALSE,
  add_to_phyloseq = NULL,
  col_prefix = NULL,
```

```
  type_works = c("article", "review", "book-chapter", "book", "letter"),
  verbose = TRUE,
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE,
  ...
)
```

Arguments

physeq	(optional) A phyloseq object. Either ‘physeq‘ or ‘taxnames‘ must be provided, but not both.
taxnames	(optional) A character vector of taxonomic names.
taxonomic_rank	(Character, default "currentCanonicalSimple") The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. c("Genus", "Species")).
count_only	(Logical, default FALSE) If TRUE, only the number of works on a given taxa is return, leading to a faster call to ‘openalexR::oa_fetch()‘. Note that if count_only is set to TRUE all works (including e.g. preprint and dataset) are count, leading to higher number of works than if count_only is set to FALSE (see parameter ‘type_works‘).
return_raw_oa	(Logical, default FALSE) If TRUE, return the raw list of publications from Open Alex for each taxa as a list of data.frame. Can be useful to filter works for example by topic or by number of citations (see section examples). If TRUE, add_to_phyloseq is set to FALSE automatically.
add_to_phyloseq	(logical, default TRUE when physeq is provided, FALSE when taxnames is provided and FALSE if return_raw_oa is set to TRUE). If TRUE, return a new phyloseq object with new columns in the tax_table slot. Automatically set to TRUE when a phyloseq object is provided and FALSE when taxnames is provided. Cannot be TRUE if ‘taxnames‘ is provided.
col_prefix	A character string to be added as a prefix to the new columns names added to the tax_table slot of the phyloseq object (default: NULL).
type_works	(A list of type to select) See Open Alex [documentation](https://docs.openalex.org/api-entities/works/work-object#type). Only used if count_only is set to FALSE Default is c("article", "review", "book-chapter", "book", "letter").
verbose	(logical, default TRUE) If TRUE, prompt some messages.
discard_genus_alone	(logical, default ‘TRUE‘ when ‘taxonomic_rank == "currentCanonicalSimple"‘). Passed to [taxonomic_rank_to_taxnames()].
discard_NA	(logical, default ‘TRUE‘). Passed to [taxonomic_rank_to_taxnames()].
...	Other params to passed on [openalexR::oa_fetch()]

Details

This function is mainly a wrapper of the work of others. Please cite ‘openalexR‘ package.

Value

Either a tibble (if add_to_phyloseq = FALSE) or a new phyloseq object, if add_to_phyloseq = TRUE, with 1 (‘n_doi‘) or 4 (‘n_doi‘, ‘list_doi‘, ‘n_citation‘ and ‘list_keywords‘ if ‘count_only‘ is FALSE) new column(s) in the tax_table.

- n_doi: number of publications citing this taxa in title or abstract - list_doi: list of DOIs separate by ";" - n_citation: total number of citations for all publications citing this taxa - list_keywords: list of keywords from all publications citing this taxa

Author(s)

Adrien Taudiere

Examples

```
## Not run:
data_fungi_mini_cleanNames <- gna_verifier_pq(data_fungi_mini) |>
  tax_oa_pq()

ggplot(
  subset_taxa(data_fungi_mini_cleanNames, !is.na(n_doi))@tax_table,
  aes(
    x = log10(as.numeric(n_doi)),
    y = forcats::fct_reorder(currentCanonicalSimple, as.numeric(n_doi))
  )
) +
  geom_point(aes(col = Order)) +
  xlab("Number of Scientific Papers (log10 scale)")

tax_oa_pq(data_fungi_mini_cleanNames, type_works = "dataset")


list_pub_raw <- tax_oa_pq(data_fungi_mini_cleanNames,
  col_prefix = "oa_",
  return_raw_oa = TRUE
)

list_pub_Health_science <- lapply(list_pub_raw, function(xx) {
  if (length(xx) == 0) {
    return(NULL)
  } else {
    filter(xx, map_lgl(topics, function(tibble_item) {
      if (is.null(tibble_item) || nrow(tibble_item) == 0) {
        return(FALSE)
      } else {
        any(grepl("Health science",
          tibble_item$display_name[tibble_item$type == "domain"],
          ignore.case = TRUE
        ))
      }
    }))
  }
```

```
})


list_pub_Ecology <- lapply(list_pub_raw, function(xx) {
  if (length(xx) == 0) {
    return(NULL)
  } else {
    filter(xx, map_lgl(topics, function(tibble_item) {
      if (is.null(tibble_item) || nrow(tibble_item) == 0) {
        return(FALSE)
      } else {
        any(grepl("Ecology",
          tibble_item$display_name[tibble_item$type == "subfield"],
          ignore.case = TRUE
        ))
      }
    }))
  }
})

list_pub_at_least_ten_citations <-
  lapply(list_pub_raw, function(xx) {
    if (length(xx) == 0) {
      return(NULL)
    } else {
      filter(xx, cited_by_count > 10)
    }
  })

## End(Not run)
```

---

tax_occur_check *Taxa occurrences check within a radius using GBIF data*

Description

<img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Performs a species occurrence check within a fixed radius around a GPS point using GBIF occurrence data.

Usage

```
tax_occur_check(
  taxa_name,
  longitude,
  latitude,
  radius_km = 50,
  method = c("download", "search"),
```

```
  circle_form = TRUE,
  clean_coord = TRUE,
  info_names = c("decimalLongitude", "decimalLatitude", "country", "year",
    "scientificName", "recordedBy", "gbifRegion"),
  return_all_occ = FALSE,
  verbose = TRUE,
  clean_coord_verbose = FALSE,
  n_occur = 1000,
  ...
)
```

Arguments

taxa_name	Character. Scientific name of the species to check.
longitude	Numeric. Longitude of the test point in decimal degrees.
latitude	Numeric. Latitude of the test point in decimal degrees.
radius_km	Numeric. Search radius in kilometers (default: 50).
method	(character, default ‘"download"‘). How occurrences are fetched: - ‘"download"‘: a single [rgbif::occ_download()] request constrained to the search bounding box (mints a citable DOI). **Requires GBIF credentials** (see [check_gbif_credentials()]). - ‘"search"‘: the legacy [rgbif::occ_search()] call (fast, capped at ‘n_occur‘ records, no credentials).
circle_form	(Logical, default: TRUE). Whether to use a circular search area. If FALSE, a square bounding box is used.
clean_coord	(Logical, default: TRUE). Whether to clean coordinates using CoordinateCleaner
info_names	Character vector. Columns to select from GBIF data (default:c("decimalLongitude", "decimalLatitude", "country", "year", "scientificName", "recordedBy", "gbifRegion")). Note that "scientificName", "decimalLongitude" and "decimalLatitude" are required. With ‘method = "download"‘, ‘"country"‘ is mapped to ‘"countryCode"‘ and download-only absent columns (e.g. ‘"gbifRegion"‘) are silently dropped.
return_all_occ	(Logical, default: FALSE). If TRUE, return all occurrences found within the radius in a data frame called "occ_data" in the resulting list.
verbose	(Logical, default: TRUE). Whether to print progress messages.
clean_coord_verbose	(Logical, default: FALSE). Whether to print messages from CoordinateCleaner.
n_occur	Numeric (default: 1000). Maximum number of occurrences to retrieve from GBIF. A server-side limit with ‘method = "search"‘; applied as a local sample after import with ‘method = "download"‘.
...	Additional parameters passed to [rgbif::occ_search()] (only used when ‘method = "search"‘).

Value

A list containing: - count_in_radius: Number of occurrences found within the radius - closest_distance_km: Distance to the closest occurrence in kilometers - mean_distance_km: Mean distance to all occur-

rences in kilometers - total_count_in_world: Total number of occurrences with coordinates worldwide - search_radius: The search radius used (in kilometers) - closest_point_lat: Latitude of the closest occurrence - closest_point_lon: Longitude of the closest occurrence - sample_point_lat: Latitude of the tested point - sample_point_lon: Longitude of the tested point - occ_data (optional, if 'return_all_occ' is TRUE): Data frame of all occurrences found within the radius

Author(s)

Adrien Taudiere

See Also

[tax_occur_check_pq()], [tax_occur_multi_check_pq()], [rgbif::occ_download()]

Examples

```
## Not run:
# Check for Oak species near Paris
long <- 2.3522
lat <- 48.8566

Q_rob_in_Paris <- tax_occur_check("Quercus robur", long, lat, radius_km=10)
Q_rob_in_Paris

tax_occur_check("Trametopsis brasiliensis", long, lat, radius_km=100)


# Visualize occurrences around Paris for Fagus sylvatica
res_occ <- tax_occur_check("Fagus sylvatica", long, lat, radius_km=20,
  return_all_occ = TRUE
)

occ_data_sf <- sf::st_as_sf(res_occ$occ_data,
  coords = c("decimalLongitude", "decimalLatitude"),
  crs = 4326
)

if (requireNamespace("leaflet")) {
  library(leaflet)
}
if (requireNamespace("leafpop")) {
  library(leafpop)
}
leaflet() |>
  addTiles() |>
  setView(lat, long, zoom = 12) |>
  fitBounds(
    lat1 = as.vector(sf::st_bbox(occ_data_sf))[2],
    lng1 = as.vector(sf::st_bbox(occ_data_sf))[1],
    lat2 = as.vector(sf::st_bbox(occ_data_sf))[4],
    lng2 = as.vector(sf::st_bbox(occ_data_sf))[3]
  ) |>
```

```
  leaflet::addCircles(data = occ_data_sf, color = "blue", stroke = 1, opacity = 0.8) |>
  leaflet::addCircleMarkers(lat, long, color = "orange", radius = 2, opacity = 1)

## End(Not run)
```

`tax_occur_check_pq` *Check for taxa occurrences within a radius around samples using GBIF data*

Description

<img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

This function performs a species range check for taxa contained in a phyloseq object. The result can optionally be added to the phyloseq object's tax_table as new columns.

Usage

```
tax_occur_check_pq(
  physeq = NULL,
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  longitude = NULL,
  latitude = NULL,
  radius_km = 50,
  n_occur = 1000,
  method = c("download", "search"),
  circle_form = TRUE,
  clean_coord = TRUE,
  clean_coord_verbose = FALSE,
  add_to_phyloseq = NULL,
  col_prefix = NULL,
  verbose = TRUE,
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE,
  ...
)
```

Arguments

`physeq`	(optional) phyloseq object. Either 'physeq' or 'taxnames' must be provided, but not both. The phyloseq object containing the taxa to check.
`taxnames`	(optional) A character vector of taxonomic names.
`taxonomic_rank`	Character. The taxonomic rank to use for the check. Default is "currentCanonicalSimple" which corresponds to the cleaned scientific names in the phyloseq object if [gna_verifier_pq()] was used with default parameter.

`longitude` Numeric. Longitude of the test point in decimal degrees.

`latitude` Numeric. Latitude of the test point in decimal degrees.

`radius_km` Numeric. Search radius in kilometers (default: 50).

`n_occur` Numeric. Maximum number of occurrences to retrieve from GBIF for each taxon (default: 1000).

`method` (character, default ‘"download"‘). How occurrences are fetched. ‘"download"‘ issues a single [rgbif::occ_download()] for all taxa around the point (**requires GBIF credentials**); ‘"search"‘ uses a per-taxon [rgbif::occ_search()] loop. See [tax_occur_check()].

`circle_form` (Logical, default: TRUE). Whether to use a circular search area. If FALSE, a square bounding box is used.

`clean_coord` (Logical, default: TRUE). Whether to clean coordinates using ‘CoordinateCleaner‘.

`clean_coord_verbose` (Logical, default: FALSE). Whether to print messages from ‘CoordinateCleaner‘.

`add_to_phyloseq` (Logical, default TRUE when physeq is provided, FALSE when taxnames is provided). Whether to add the results as new columns in the phyloseq object's tax_table. If TRUE, the results will be appended to the tax_table with appropriate column names. Automatically set to TRUE when a phyloseq object is provided and FALSE when taxnames is provided. Cannot be TRUE if ‘taxnames‘ is provided.

`col_prefix` A character string to be added as a prefix to the new columns names added to the tax_table slot of the phyloseq object (default: NULL).

`verbose` (Logical, default: TRUE). Whether to print progress messages.

`discard_genus_alone` (logical, default ‘TRUE‘ when ‘taxonomic_rank == "currentCanonicalSimple"‘). Passed to [taxonomic_rank_to_taxnames()].

`discard_NA` (logical, default ‘TRUE‘). Passed to [taxonomic_rank_to_taxnames()].

`...` Additional parameters passed to [tax_occur_check()].

Value

Either a data frame (if add_to_phyloseq = FALSE) or a new phyloseq object (if add_to_phyloseq = TRUE).

Author(s)

Adrien Taudiere

See Also

[tax_occur_check()], [tax_occur_multi_check_pq()]

Examples

```
## Not run:

data_fungi_mini_cleanNames <- gna_verifier_pq(data_fungi_mini)

check_res <- tax_occur_check_pq(data_fungi_mini_cleanNames,
  longitude = 2.3,
  latitude = 48,
  radius_km = 100,
  n_occur = 50,
  add_to_phyloseq = FALSE
)

check_res |>
  mutate(taxa_name = forcats::fct_reorder(taxa_name, count_in_radius)) |>
  ggplot(aes(x = count_in_radius, y = taxa_name, fill = total_count_in_world)) +
  geom_col()

data_fungi_mini_cleanNames_range_verif <-
  tax_occur_check_pq(data_fungi_mini_cleanNames,
    longitude = 2.3,
    latitude = 48,
    radius_km = 50,
    n_occur = 10
  )

df <- data_fungi_mini_cleanNames_range_verif@tax_table[, "count_in_radius"] |>
  table(useNA = "always") |>
  data.frame()

colnames(df) <- c("count_in_radius", "n_taxa")
df

# Subset taxa with at least one occurrence in the radius
cond_count_sup_0 <-
  data_fungi_mini_cleanNames_range_verif@tax_table[, "count_in_radius"] |>
    as.numeric() > 0
cond_count_sup_0[is.na(cond_count_sup_0)] <- FALSE
names(cond_count_sup_0) <- taxa_names(data_fungi_mini_cleanNames_range_verif)

subset_taxa_pq(data_fungi_mini_cleanNames_range_verif, cond_count_sup_0) |>
  summary_plot_pq()

## End(Not run)
```

`tax_occur_multi_check_pq` *Check for taxa occurrences within a radius around multiple samples using GBIF data*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

This function performs a species range check for taxa contained in a phyloseq object, for multiple samples based on their geographic coordinates (longitude and latitude).

Usage

```
tax_occur_multi_check_pq(
  physeq = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  min_occur = 0,
  verbose = TRUE,
  lon_column = NULL,
  longitudes = NULL,
  lat_column = NULL,
  latitudes = NULL,
  radius_km = 50,
  n_occur = 1000,
  method = c("download", "search"),
  circle_form = TRUE,
  clean_coord = TRUE,
  clean_coord_verbose = FALSE,
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE,
  ...
)
```

Arguments

physeq	(required) A phyloseq object.
taxonomic_rank	The taxonomic rank to use for the check. Default is "currentCanonicalSimple" which corresponds to the cleaned scientific names in the phyloseq object if [gna_verifier_pq()] was used with default parameter.
min_occur	Minimum number of occurrences in the radius to keep the taxon (default: 0).
verbose	(Logical, default: TRUE). Whether to print progress messages.
lon_column	Column name in sample_data containing longitudes.
longitudes	Vector of longitudes corresponding to samples in the phyloseq object. If provided, it overrides lon_column.
lat_column	Column name in sample_data containing latitudes.
latitudes	Vector of latitudes corresponding to samples in the phyloseq object. If provided, it overrides lat_column.
radius_km	Numeric. Search radius in kilometers (default: 50). See ?[tax_occur_check_pq()].
n_occur	Numeric (default: 1000). Maximum number of occurrences to retrieve from GBIF for each taxon.

method	(character, default '"download"'). How occurrences are fetched. '"download"' issues a **single** [rgbif::occ_download()] covering all taxa over the bounding box of every GPS point (**requires GBIF credentials**); '"search"' uses a per-taxon [rgbif::occ_search()] loop. See [tax_occur_check()].
circle_form	(Logical, default: TRUE). Whether to use a circular search area. If FALSE, a square bounding box is used.
clean_coord	(Logical, default: TRUE). Whether to clean coordinates using 'CoordinateCleaner'.
clean_coord_verbose	(Logical, default: FALSE). Whether to print messages from 'CoordinateCleaner'.
discard_genus_alone	(logical, default 'TRUE' when 'taxonomic_rank == "currentCanonicalSimple"'). Passed to [taxonomic_rank_to_taxnames()].
discard_NA	(logical, default 'TRUE'). Passed to [taxonomic_rank_to_taxnames()].
...	Additional parameters (currently unused; reserved for forward compatibility).

Value

A list containing: - A tibble resulting from the concatenation of result of function [tax_occur_check()] for each GPS position. - A matrix of samples x taxa with the number of occurrences in the radius for each case of the matrix. - A new phyloseq object with taxa filtered based on min_occur. Be careful, the filtering may be very stringent.

Author(s)

Adrien Taudiere

See Also

[tax_occur_check()], [tax_occur_multi_pq()]

Examples

```
## Not run:
data_fungi_mini_cleanNames <-
  gna_verifier_pq(data_fungi_mini,
    data_sources = 210
  )
res_occur_check <-
  tax_occur_multi_check_pq(subset_samples(data_fungi_mini_cleanNames, Diameter == 52),
    longitudes = c(8.31, 8.31, 8.64, -1.19, 7.03),
    latitudes = c(47.38, 47.38, 45.83, 43.65, 43.93)
  )

## End(Not run)
```

tax_photos_pq *Find photos of taxa from GBIF or Wikitaxa*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Usage

```
tax_photos_pq(
  physeq = NULL,
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  source = "gbif",
  folder_name = "photos_physeq",
  add_to_phyloseq = NULL,
  col_prefix = NULL,
  gallery = FALSE,
  overwrite_folder = FALSE,
  col_name_url = "photo_url",
  verbose = TRUE,
  caption_valign = "bottom",
  caption_font_size = 12,
  simple_caption = FALSE,
  img_height = "150px",
  img_width = "200px",
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE,
  ...
)
```

Arguments

physeq	(optional) A phyloseq object. Either ‘physeq‘ or ‘taxnames‘ must be provided, but not both.
taxnames	(optional) A character vector of taxonomic names.
taxonomic_rank	(Character, default = "currentCanonicalSimple") The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. the c("Genus", "Species")).
source	(Character) either "gbif" or "wikitaxa".
folder_name	(default "photos_physeq") Name of the folder where photos will be downloaded. Only used if both add_to_phyloseq and gallery are FALSE.

add_to_phyloseq
: (logical, default TRUE when physeq is provided, FALSE when taxnames is provided) If TRUE, a new phyloseq object is returned with a new column containing the URL (entitled with the parameter col_name_url) in the tax_table. Automatically set to TRUE when a phyloseq object is provided and FALSE when taxnames is provided. Cannot be TRUE if ‘taxnames‘ is provided.

col_prefix
: A character string to be added as a prefix to the new columns names added to the tax_table slot of the phyloseq object (default: NULL).

gallery
: (logical, default FALSE) If TRUE, a html gallery is created using [htmltools::browsable()].

overwrite_folder
: (logical, default FALSE) If TRUE, the folder specified in the parameter folder_name will be deleted if it already exists.

col_name_url
: (default "photo_url") Name of the new column in the tax_table

verbose
: (logical, default TRUE) If TRUE, prompt some messages.

caption_valign
: (character, default "bottom") Vertical alignment of the caption in the gallery. Either ‘"bottom"‘ or ‘"top"‘.

caption_font_size
: (int) Size of the caption font in the gallery.

simple_caption
: (logical, default FALSE) If TRUE, the caption of the gallery photo will be only the taxonomic name. If FALSE, the caption include information from the phyloseq object (number of sequences, taxa and samples).

img_height
: (character, default "150px") Height of images in the gallery.

img_width
: (character, default "200px") Width of images in the gallery.

discard_genus_alone
: (logical, default ‘TRUE‘ when ‘taxonomic_rank == "currentCanonicalSimple"‘). Passed to [taxonomic_rank_to_taxnames()].

discard_NA
: (logical, default ‘TRUE‘). Passed to [taxonomic_rank_to_taxnames()].

...
: Unused, kept for backward compatibility.

Details

There is three behavior. See the returns section. Gbif source is quicker than wikitaxa source. Note that for the moment the function only return one photo per species.

Value

There is three behavior.(i) If add_to_phyloseq = TRUE, a new phyloseq object is returned with a new column (called with the parameter col_name_url) in the tax_table containing the URL; the gallery is printed as a side-effect if ‘gallery = TRUE‘. (ii) If add_to_phyloseq = FALSE and gallery = TRUE, the HTML gallery is returned. (iii) If both gallery = FALSE and add_to_phyloseq = FALSE, photos are downloaded in a folder (folder_name parameter) and the list of url are returned in the form of a tibble.

Author(s)

Adrien Taudiere

Examples

```
## Not run:
data_fungi_mini_cleanNames <- gna_verifier_pq(data_fungi_mini)

tax_photos_pq(data_fungi_mini_cleanNames,
  gallery = TRUE,
  img_height = "40px",
  img_width = "80px",
  source = "wikitaxa"
)

tax_photos_pq(
  taxnames = c("Xylodon flaviporus", "Basidiodendron eyrei"),
  gallery = TRUE
)

data_fungi_mini_cleanNames_photos <-
  tax_photos_pq(data_fungi_mini_cleanNames)

# Which photo(s) depicted more than one OTU
data_fungi_mini_cleanNames_photos@tax_table[, "photo_url"] |>
  table() |>
  (\(tab) tab[as.numeric(tab) > 1])()

## End(Not run)
```

`tax_retroblast_pq` *Verify taxonomic assignment using BLAST against NCBI nucleotide database*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

The idea is to take the binomial taxonomic name assigned to each ASV/OTU at the Genus_species level, search for sequences in NCBI nucleotide database corresponding to this taxon name (with some additional filters including the marker name), retrieve the sequences in fasta format, and then perform a BLAST search of retrieved sequences against the ASV/OTU sequences.

We can therefore test for each ASV/OTU if the best BLAST hit corresponds to the same taxon name as the one assigned to the ASV/OTU. Moreover, we can also detect some cases where a better taxonomic assignment can be proposed based on the BLAST results limited to species name already present in the phyloseq object.

Note that this function need a physeq object and cannot works with a list of taxonomic names (taxnames is not a parameter of the function).

Usage

```
tax_retroblast_pq(
  physeq,
  taxonomic_rank = "currentCanonicalSimple",
  marker = NULL,
  id_cut = 99,
  retmax = 500,
  add_to_phyloseq = TRUE,
  verbose = TRUE,
  start_date = NULL,
  end_date = NULL,
  min_length = 300,
  max_length = 4000,
  refseq_only = FALSE,
  sup_params = "NOT uncultured[Title] NOT clone[Title]",
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE,
  ...
)
```

Arguments

physeq	(required) A phyloseq object
taxonomic_rank	(required, default = "currentCanonicalSimple") The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. c("Genus", "Species")).
marker	(required) A character vector of marker names to be used in the search term. For example, c("ITS", "internal transcribed spacer") for fungal ITS sequences. Note that the marker names should be present in the title of the sequences in NCBI nucleotide database.
id_cut	(default: 99) minimum as a good match. A 100 value means that only perfect matches are considered as good matches.
retmax	(default: 500) maximum number of sequences to retrieve from NCBI nucleotide database for each taxon name.
add_to_phyloseq	(logical, default TRUE) If TRUE, a new phyloseq object is returned with new columns in the tax_table.
verbose	(logical, default TRUE) If TRUE, prompt some messages.
start_date	The start date for the search. If NULL (default), the search is not limited by date. The date must be in the format "YYYY-MM-DD".
end_date	() The end date for the search. If NULL (default), the search is not limited by date. If start_date is not NULL and end_date is NULL, the end_date is set to today's date. The date must be in the format "YYYY-MM-DD".
min_length	(int) Minimum sequence length to consider in the search.
max_length	(int) Maximum sequence length to consider in the search.

`refseq_only` (logical, default FALSE) If TRUE, only sequences from the RefSeq database are retrieved. RefSeq is a curated non-redundant database of sequences from NCBI. If FALSE, all sequences from NCBI nucleotide database are retrieved. Note that using refseq_only = TRUE is experimental and may lead to no sequence retrieved for some taxon names.

`sup_params` (char) Additional parameters to be added to the search term. By default set to ("NOT uncultured[Title] NOT clone[Title]") to exclude uncultured and clone sequences.

`discard_genus_alone` (logical, default 'TRUE' when 'taxonomic_rank == "currentCanonicalSimple"'). Passed to [taxonomic_rank_to_taxnames()].

`discard_NA` (logical, default 'TRUE'). Passed to [taxonomic_rank_to_taxnames()].

`...` Additional parameters to be passed to [MiscMetabar::blast_to_phyloseq()] including: 'nproc', 'e_value_cut' and 'args_blastn'

Value

Either a list (if add_to_phyloseq = FALSE) or a new phyloseq object, if add_to_phyloseq = TRUE, with new columns based on the 'tib_retroblast' tibble describe below:

The list is composed of two elements: 1. 'tib_retroblast': A tibble with one row for each taxa of the phyloseq object: - 'blast_queried': (logical) queried names for sequences - 'blast_result': (logical) Number of queried names with at least one blast result - 'good_assign': (logical) Number of good assignation (best blast hit with as the one assigned to the ASV/OTU) - 'alt_assign': Number of alternative assignation proposed (best blast hit with the phyloseq object) - 'taxa_name': Taxonomic name used to query NCBI nucleotide database

2. 'entrez_search': A list of the rentrez::entrez_search results for each taxon name

Author(s)

Adrien Taudiere

See Also

[MiscMetabar::blast_to_phyloseq()], [rentrez::entrez_search()]

Examples

```
## Not run:
data_fungi_mini_cleanNames <-
  gna_verifier_pq(data_fungi_mini,
    data_source = 210)

res_retro <- tax_retroblast_pq(data_fungi_mini_cleanNames,
  marker = c("ITS", "internal transcribed spacer"),
  retmax = 10,
   id_cut = 99,
   add_to_phyloseq = FALSE
)
```

```
res_retro$tib_retroblast |>
  summarise(
    prop_good_assign = sum(good_assign) / sum(blast_result),
    n_alt_assign = sum(!is.na(alt_assign))
  )

table(res_retro$tib_retroblast$alt_assign)

res_retro_100 <- tax_retroblast_pq(data_fungi_mini_cleanNames,
  marker = c("ITS", "internal transcribed spacer"),
  retmax = 100, id_cut = 100
)

# nb of queried names for sequences (id=100%)
res_retro_100$tib_retroblast$blast_queried |> sum()
# nb of queried names with at least one blast result (id=100%)
res_retro_100$tib_retroblast$blast_result |> sum()
# nb of good assignation (id=100%)
res_retro_100$tib_retroblast$good_assign |> sum()
# nb of alternative assignation proposed (id=100%)
res_retro_100$tib_retroblast$alt_assign |>
  is.na() |>
  sapply(isFALSE) |>
  sum()


## End(Not run)
```

`tax_spores_size_pq` *Extract spore size from mycoDB*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Extract spore size information from mycoDB (https://www.mycodb.fr/).

Usage

```
tax_spores_size_pq(
  physeq = NULL,
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  verbose = TRUE,
  time_to_sleep = 0.5,
  add_to_phyloseq = NULL,
  col_prefix = NULL,
  discard_genus_alone = identical(taxonomic_rank, "currentCanonicalSimple"),
  discard_NA = TRUE
)
```

Arguments

physeq	(optional) A phyloseq object. Either ‘physeq‘ or ‘taxnames‘ must be provided, but not both.
taxnames	(optional) A character vector of taxonomic names.
taxonomic_rank	(Character, default "currentCanonicalSimple") The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. c("Genus", "Species")).
verbose	(logical, default TRUE) If TRUE, prompt some messages.
time_to_sleep	(numeric, default 0.5) Time to sleep between two queries to mycoDB, in seconds.
add_to_phyloseq	(logical, default NULL) If TRUE, add the spore size information to the phyloseq object. If FALSE, return a data.frame. If NULL (default), add to phyloseq if ‘physeq‘ is provided, else return a data.frame.
col_prefix	(character, default NULL) If not NULL, prefix to add to the new columns added to the phyloseq object.
discard_genus_alone	(logical, default ‘TRUE‘ when ‘taxonomic_rank == "currentCanonicalSimple"‘). Passed to [taxonomic_rank_to_taxnames()].
discard_NA	(logical, default ‘TRUE‘). Passed to [taxonomic_rank_to_taxnames()].

Value

If ‘add_to_phyloseq‘ is TRUE, returns a phyloseq object with new columns in the tax_table slot: ‘spore_size‘, ‘spore_length‘, ‘spore_width‘. If ‘add_to_phyloseq‘ is FALSE, returns a data.frame with columns ‘taxa_name‘, ‘spore_size‘, ‘spore_length‘, ‘spore_width‘.

Author(s)

Adrien Taudiere

See Also

[extract_spores_mycodb()]

Examples

```
## Not run:
data_fungi_mini_cleanNames <- data_fungi_mini |>
  gna_verifier_pq()
data_fungi_mini_spore_size <- tax_spores_size_pq(data_fungi_mini_cleanNames)

psmelt(data_fungi_mini_spore_size) |>
  group_by(taxa_name) |>
  summarise(
    spore_length = as.numeric(unique(spore_length_mean)),
    spore_width = as.numeric(unique(spore_width_mean)),
    Abundance = sum(Abundance),
```

```
  Occurence = sum(Abundance > 0, na.rm = TRUE)
) |>
ggplot(aes(x = spore_length, y = spore_width, size = Abundance, col = Occurence)) +
geom_point(alpha = 0.7) +
ggrepel::geom_text_repel(aes(label = taxa_name),
  vjust = -0.5,
  size = 3,
  fontface = "italic",
  min.segment.length = 0.2,
  force = 4
) +
labs(
  title = "Spore sizes extracted from mycoDB",
  x = "Spore length (\u00b5m)",
  y = "Spore width (\u00b5m)",
  col = "Number of samples",
  size = "Number of sequences"
) +
theme_idest()

# Example with ellipses
psmelt(data_fungi_mini_spore_size) |>
  filter(!is.na(taxa_name) & !taxa_name == "") |>
  filter(!is.na(Time)) |>
  filter(Abundance > 0) |>
  mutate(taxa_name = as.factor(taxa_name)) |>
  group_by(taxa_name, Time) |>
  summarise(
    spore_length = 0.2 * as.numeric(unique(spore_length_mean)),
    spore_width = as.numeric(unique(spore_width_mean)),
    Abundance = sum(Abundance),
    Occurence = sum(Abundance > 0, na.rm = TRUE),
    Order = unique(Order)
  ) |>
  arrange(desc(Abundance)) |>
  mutate(
    taxa_name_num = as.numeric(taxa_name)
  ) |>
  filter(!is.na(spore_length)) |>
  ggplot(aes(
    x0 = log(Abundance), y0 = taxa_name_num / 5,
    a = spore_length / 2, b = spore_length / 2 / 5, fill = Order
  )) +
  coord_fixed() +
  ggforce::geom_ellipse(aes(angle = 0), alpha = 0.3) +
  ggrepel::geom_text_repel(aes(
    x = log(Abundance), y = taxa_name_num / 5,
    label = taxa_name, color = Order
  ), size = 2) +
  theme_idest() +
  theme(axis.text.y = element_blank()) +
  labs(x = "Number of sequences (log scale)", y = "Taxa") +
  facet_wrap(~Time, ncol = 2)
```

```
# Test for difference in mean spore length between sample's factor
psmelt(data_fungi_mini_spore_size) |>
  filter(!is.na(taxa_name) & !taxa_name == "") |>
  filter(!is.na(spore_length_mean)) |>
  filter(!is.na(Time)) |>
  filter(Abundance > 0) |>
  mutate(taxa_name = as.factor(taxa_name)) |>
  group_by(taxa_name, Time) |>
  summarise(
    spore_length = unique(as.numeric(spore_length_mean)),
    spore_width = unique(as.numeric(spore_width_mean)),
    Order = unique(Order)
  ) |>
  ggstatsplot::ggbetweenstats(Time, spore_length)

## End(Not run)
```

`taxa_summary_text` *Text summary for a taxonomic rank*

Description

<img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Create a text to summarize the number of samples, taxa, sequences and occurrences of selected taxa in a phyloseq object for a given value in the column of a tax_table

Usage

```
taxa_summary_text(
  physeq,
  taxnames = NULL,
  taxonomic_rank = "currentCanonicalSimple",
  verbose = TRUE,
  min_nb_seq = 0,
  ...
)
```

Arguments

`physeq`	A phyloseq object
`taxnames`	(optional) A character vector of taxonomic names.
`taxonomic_rank`	(Character, default "currentCanonicalSimple") The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. c("Genus", "Species")).

verbose	(logical, default TRUE) If TRUE, prompt some messages.
min_nb_seq	minimum number of sequences by OTUs by samples to take into count this OTUs in this sample. For example, if min_nb_seq=2,each value of 2 or less in the OTU table will not count in the venn diagram
...	Additional arguments to pass to [subset_taxa_pq()].

Value

A character string summarizing the number of samples, taxa, sequences and occurrences of the selected taxa.

Author(s)

Adrien Taudiere

Examples

```
data_fungi_cleanNames <- gna_verifier_pq(data_fungi_mini, data_sources = 210)

taxa_summary_text(data_fungi_cleanNames, taxnames = "Xylodon flaviporus")

taxa_summary_text(data_fungi_cleanNames,
  taxnames = "Xylodon flaviporus",
  min_nb_seq = 100, verbose = FALSE
)
taxa_summary_text(data_fungi_cleanNames,
  taxonomic_rank = "Trait",
  taxnames = c("Soft Rot"), verbose = FALSE
)
```

---

taxonomic_rank_to_taxnames

Extract taxonomic names from a phyloseq object

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

Mainly a internal function for function [gna_verifier_pq()], [tax_oa_pq()], [gbif_occur_pq()], [tax_iucn_code_pq()], [tax_globi_pq()], [plot_tax_gbif_pq()], ...

Usage

```
taxonomic_rank_to_taxnames(
  physeq,
  taxonomic_rank = c("Genus", "Species"),
  discard_genus_alone = FALSE,
  discard_NA = TRUE,
  distinct_names = TRUE
)
```

Arguments

physeq	A phyloseq object
taxonomic_rank	(Character) The column(s) present in the @tax_table slot of the phyloseq object. Can be a vector of two columns (e.g. the default c("Genus", "Species")).
discard_genus_alone	(logical default FALSE). If TRUE genus without information at the species level are discarded.
discard_NA	(logical default TRUE). If TRUE, taxa with NA in the taxonomic_rank are discarded.
distinct_names	(logical default TRUE). If TRUE, return only unique taxonomic names.

Value

A vector of unique taxonomic names

Author(s)

Adrien Taudiere

Examples

```
taxonomic_rank_to_taxnames(data_fungi_mini)
taxonomic_rank_to_taxnames(data_fungi_mini, discard_genus_alone = TRUE)
taxonomic_rank_to_taxnames(data_fungi_mini, discard_NA = TRUE)
taxonomic_rank_to_taxnames(data_fungi_mini,
  discard_NA = TRUE, discard_genus_alone = TRUE
)

## Not run:
taxonomic_rank_to_taxnames(gna_verifier_pq(data_fungi_mini), taxonomic_rank="currentCanonicalSimple")
taxonomic_rank_to_taxnames(gna_verifier_pq(data_fungi_mini), taxonomic_rank="genusEpithet")

## End(Not run)
taxonomic_rank_to_taxnames(data_fungi_mini, taxonomic_rank="Class")
taxonomic_rank_to_taxnames(data_fungi_mini, taxonomic_rank="Class",
 distinct_names = FALSE,
 discard_NA = TRUE
)
```

theme_idest *ggplot theme for IdEst*

Description

 <img src="https://img.shields.io/badg
experimental-orange" alt="lifecycle-experimental">

This theme is used by Adrien Taudiere [IdEst](https://adrientaudiere.com/). Based on [hrbrthemes](https://github.com/hrbrms
‘hrbrthemes::theme_ipsum()‘ by boB Rudis.

Usage

```
theme_idest(
  sans_family = if (.Platform$OS.type == "windows") {
     "Roboto Condensed"
 } else {

       "Roboto Condensed Light"
 },
  serif_family = "Linux Libertine G",
  mono_family = "Fira Code",
  base_size = 11.5,
  plot_title_family = serif_family,
  plot_title_size = 18,
  plot_title_face = "bold",
  plot_title_margin = 10,
  subtitle_family = serif_family,
  subtitle_size = 13,
  subtitle_face = "plain",
  subtitle_margin = 15,
  subtitle_color = "grey30",
  strip_text_family = mono_family,
  strip_text_size = 13,
  strip_text_face = "plain",
  strip_back_grey = FALSE,
  caption_family = sans_family,
  caption_size = 9,
  caption_face = "plain",
  caption_margin = 10,
  axis_text_size = base_size * 0.8,
  axis_text_family = sans_family,
  axis_title_family = mono_family,
  axis_title_size = 12,
  axis_title_face = "plain",
  axis_title_just = "c",
  plot_margin = margin(12, 12, 12, 12),
```

```
  panel_spacing = grid::unit(1.2, "lines"),
  grid_col = "#cccccc",
  grid = TRUE,
  axis_col = "#cccccc",
  axis = FALSE,
  ticks = FALSE
)
```

Arguments

sans_family	Font family for sans serif text (default is "Roboto Condensed" on Windows and "Roboto Condensed Light" on other OS).
serif_family	Font family for serif text (default is "Linux Libertine G").
mono_family	Font family for monospaced text (default is "Fira Code").
base_size	Base font size (default is 11.5).
plot_title_family	Font family for title (default is serif_family).
plot_title_size	Font size for title (default is 18).
plot_title_face	Font face for title (default is "bold").
plot_title_margin	Margin below title (default is 10).
subtitle_family	Font family for subtitle (default is serif_family).
subtitle_size	Font size for subtitle (default is 13).
subtitle_face	Font face for subtitle (default is "plain").
subtitle_margin	Margin below subtitle (default is 15).
subtitle_color	Font color for subtitle (default is "grey30").
strip_text_family	Font family for facet strip text (default is mono_family).
strip_text_size	Font size for facet strip text (default is 13).
strip_text_face	Font face for facet strip text (default is "plain").
strip_back_grey	Logical, whether to use grey background for facet strips (default is FALSE).
caption_family	Font family for caption (default is sans_family).
caption_size	Font size for caption (default is 9).
caption_face	Font face for caption (default is "plain").
caption_margin	Margin above caption (default is 10).
axis_text_size	Font size for axis text (default is 80% of base_size).

`axis_text_family`
: Font family for axis text (default is sans_family).

`axis_title_family`
: Font family for axis titles (default is mono_family).

`axis_title_size`
: Font size for axis titles (default is 12).

`axis_title_face`
: Font face for axis titles (default is "plain").

`axis_title_just`
: Justification for axis titles (default is "c" for center).

`plot_margin` Margin around the plot (default is margin(12, 12, 12, 12)).

`panel_spacing` Spacing between panels (default is unit(1.2, "lines")).

`grid_col` Color for grid lines (default is "#cccccc").

`grid` Logical or character, whether to show grid lines (default is TRUE).

`axis_col` Color for axis lines (default is "#cccccc").

`axis` Logical or character, whether to show axis lines (default is FALSE).

`ticks` Logical, whether to show axis ticks (default is FALSE).

Value

A ggplot2 theme object.

Author(s)

Adrien Taudiere

Index