Package 'metacoder'

May 23, 2017

Title Tools for Parsing, Manipulating, and Graphing Hierarchical Data

Version 0.1.3

Maintainer Zachary Foster <zacharyfoster1989@gmail.com>

Description A set of tools for parsing, manipulating, and graphing data classified by a hierarchy (e.g. a taxonomy).

Depends R (>= 3.0.2)

License GPL-2 | GPL-3

LazyData true

URL https://grunwaldlab.github.io/metacoder_documentation/

BugReports https://github.com/grunwaldlab/metacoder/issues

Imports stringr, ggplot2, igraph, scales, grid, taxize, seqinr, reshape2, zoo, traits, RColorBrewer, RCurl, ape, reshape, stats, grDevices, utils, lazyeval, dplyr, magrittr, readr

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

RoxygenNote 6.0.1

Date 2017-05-22

Encoding UTF-8

NeedsCompilation no

Author Zachary Foster [aut, cre], Niklaus Grunwald [ths]

Repository CRAN

Date/Publication 2017-05-23 05:36:48 UTC

R topics documented:

`arrange_obs` *Sort columns of* `taxmap` *objects*

Description

Sort columns of `obs_data` in `taxmap` objects. Any column name that appears in `obs_data(.data)` can be used as if it was a vector on its own. See arrange for more details.

Usage

```
arrange_obs(.data, ...)
```

Arguments

`.data`	`taxmap`
`...`	One or more column names to sort on.

Value

An object of type `taxmap`

See Also

Other dplyr-like functions: `arrange_taxa`, `filter_obs`, `filter_taxa`, `mutate_obs`, `mutate_taxa`, `sample_frac_obs`, `sample_frac_taxa`, `sample_n_obs`, `sample_n_taxa`, `select_obs`, `select_taxa`, `transmute_obs`, `transmute_taxa`

Examples

```
# Sort observations by sequence name alphabetically
arrange_obs(unite_ex_data_3, seq_name)
# Reverse order of sort
arrange_obs(unite_ex_data_3, desc(seq_name))
```

arrange_taxa *Sort columns of* taxmap *objects*

Description

Sort columns of `taxon_data` in `taxmap` objects. Any column name that appears in `taxon_data(.data)` can be used as if it was a vector on its own. See arrange for more details.

Usage

```
arrange_taxa(.data, ...)
```

Arguments

`.data`	`taxmap`
`...`	One or more column names to sort on.

Value

An object of type `taxmap`

See Also

Other dplyr-like functions: `arrange_obs`, `filter_obs`, `filter_taxa`, `mutate_obs`, `mutate_taxa`, `sample_frac_obs`, `sample_frac_taxa`, `sample_n_obs`, `sample_n_taxa`, `select_obs`, `select_taxa`, `transmute_obs`, `transmute_taxa`

Examples

```
# Sort by taxon name alphabetically
arrange_taxa(unite_ex_data_3, name)
# Reverse order of sort
arrange_taxa(unite_ex_data_3, desc(name))
```

bryophytes_ex_data *Example dataset of bryophytes*

Description

A dataset containing information from 171 bryophytes family names scraped from http://www.theplantlist.org/1.1/browse/B/:

Usage

```
bryophytes_ex_data
```

Format

An object of type taxmap

Examples

```
## Not run:

library(XML)
taxon_names <- XML::htmlTreeParse("http://www.theplantlist.org/1.1/browse/B/") %>%
xmlRoot() %>%
  getNodeSet("//ul[@id='nametree']/li/a/i") %>%
  sapply(xmlValue)

bryophytes_ex_data <- extract_taxonomy(taxon_names, key = "name", database = "itis")

## End(Not run)
```

---

contaminants *Example dataset of contamination*

Description

A dataset containing information from 97 NCBI accession numbers representing possible contamination:

Usage

```
contaminants
```

Format

An object of type taxmap

Examples

```
## Not run:

ids <- c("JQ086376.1", "AM946981.2", "JQ182735.1", "CP001396.1", "J02459.1",
         "AC150248.3", "X64334.1", "CP001509.3", "CP006698.1", "AC198536.1",
         "JF340119.2", "KF771025.1", "CP007136.1", "CP007133.1", "U39286.1",
         "CP006584.1", "EU421722.1", "U03462.1", "U03459.1", "AC198467.1",
         "V00638.1", "CP007394.1", "CP007392.1", "HG941718.1", "HG813083.1",
         "HG813082.1", "CP007391.1", "HG813084.1", "CP002516.1", "KF561236.1",
         "JX509734.1", "AP010953.1", "U39285.1", "M15423.1", "X98613.1",
         "CP006784.1", "CP007393.1", "CU928163.2", "AP009240.1", "CP007025.1",
         "CP006027.1", "CP003301.1", "CP003289.1", "CP000946.1", "CP002167.1",
         "HG428755.1", "JQ086370.1", "CP001846.1", "CP001925.1", "X99439.1",
         "AP010958.1", "CP001368.1", "AE014075.1", "CP002212.1", "CP003034.1",
```

```
          "CP000243.1", "AY940193.1", "CP004009.1", "JQ182732.1", "U02453.1",
          "AY927771.1", "BA000007.2", "CP003109.1", "CP007390.1", "U02426.1",
          "U02425.1", "CP006262.1", "HG738867.1", "U00096.3", "FN554766.1",
          "CP001855.1", "L19898.1", "AE005174.2", "FJ188381.1", "AK157373.1",
          "JQ182733.1", "U39284.1", "U37692.1", "AF129072.1", "FM180568.1",
          "CP001969.1", "HE616528.1", "CP002729.1", "JF974339.1", "AB248924.1",
          "AB248923.1", "CP002291.1", "X98409.1", "CU928161.2", "CP003297.1",
          "FJ797950.1", "CP000038.1", "U82598.1", "CP002211.1", "JQ806764.1",
          "U03463.1", "CP001665.1")
contaminants <- extract_taxonomy(ids, key = "obs_id", database = "ncbi")

## End(Not run)
```

`diverging_palette` *The default diverging color palette*

Description

Returns the default color palette for diverging data

Usage

```
diverging_palette()
```

Value

`character` of hex color codes

Examples

```
diverging_palette()
```

`extract_taxonomy` *Extract taxonomy information from sequence headers*

Description

Extracts the taxonomy from metadata (e.g. sequence headers) or parsed sequence data. The location and identity of important information in the input is specified using a regular expression with capture groups and an corresponding key. An object of type `taxmap` is returned containing the specifed information. Taxa are translated into unique codes if they are not already encoded this way.

Usage

```
extract_taxonomy(input, ...)

## Default S3 method:
extract_taxonomy(input, key = c("class", "taxon_id", "name",
  "taxon_info", "obs_id", "obs_info"), regex = "(.*)", class_key = c("name",
  "taxon_id", "taxon_info"), class_regex = "(.*)", class_sep = NULL,
  class_rev = FALSE, database = c("none", "ncbi", "itis", "eol", "col",
  "tropicos", "nbn"), allow_na = TRUE, vigilance = c("warning", "error",
  "message", "none"), return_match = FALSE, return_input = FALSE,
  redundant_names = FALSE, batch_size = 100, verbosity = c("low", "none",
  "high"), ...)

## S3 method for class 'DNAbin'
extract_taxonomy(input, ...)

## S3 method for class 'list'
extract_taxonomy(input, ...)
```

Arguments

input	A vector from which to extract taxonomy information or an object of class ape{DNAbin}.
...	Not used.
key	(character) The identity of the capturing groups defined using regex. The length of key must be equal to the number of capturing groups specified in regex. Any names added to the terms will be used as column names in the output. Only "taxon_info" and "obs_info" can be used multiple times. Each term must be one of those decribed below: taxon_id A unique numeric id for a taxon for a particular database (e.g. ncbi accession number). Requires an internet connection. name The name of a taxon. Not necessarily unique, but are interpretable by a particular database. Requires an internet connection. taxon_info Arbitrary taxon info you want included in the output. Can be used more than once. class A list of taxa information that constitutes the full taxonomic classification from broad to specific (see class_rev) for a particular database. Individual taxa are separated by the class_sep argument and the information is parsed by the class_regex and class_key arguments. obs_id An unique observation (e.g. sequence) identifier for a particular database. Requires an internet connection. obs_info Arbitrary observation info you want included in the output. Can be used more than once.
regex	(character; length == 1) A regular expression with capturing groups indicating the locations of relevant information. The identity of the information must be specified using the key argument.

class_key (character of length 1) The identity of the capturing groups defined using class_iregex. The length of class_key must be equal to the number of capturing groups specified in class_regex. Any names added to the terms will be used as column names in the output. At least "taxon_id" or "name" must be specified. Only "taxon_info" can be used multiple times. Each term must be one of those decribed below:

- taxon_id A unique numeric id for a taxon for a particular database (e.g. ncbi accession number). Requires an internet connection.
- name The name of a taxon. Not necessarily unique, but are interpretable by a particular database. Requires an internet connection.
- taxon_info Arbitrary taxon info you want included in the output. Can be used more than once.

class_regex (character of length 1) A regular expression with capturing groups indicating the locations of data for each taxon in the class term in the key argument. The identity of the information must be specified using the class_key argument. The class_sep option can be used to split the classification into data for each taxon before matching. If class_sep is NULL, each match of class_regex defines a taxon in the classification.

class_sep (character of length 1) Used with the class term in the key argument. The character(s) used to separate individual taxa within a classification. After the string defined by the class capture group in regex is split by class_sep, its capture groups are extracted by class_regex and defined by class_key. If NULL, every match of class_regex is used instead with first splitting by class_sep.

class_rev (logical of length 1) Used with the class term in the key argument. If TRUE, the order of taxon data in a classfication is reversed to be specific to broad.

database (character of length 1) The name of the database that patterns given in parser will apply to. Valid databases include "ncbi", "itis", "eol", "col", "tropicos", "nbn", and "none". "none" will cause no database to be quired; use this if you want to not use the internet. NOTE: Only "ncbi" has been tested so far.

allow_na (logical of length 1) If TRUE, any missing data will be represented as NAs in the output. This preserves the correspondance between the input and output values. Missing data can be generated if the regex does not match the input or online queries fail.

vigilance (character of length 1) Controls the reporting of possible problems, such as missing data and failed online queries (see allow_na). The following values are possible:

- "none" No warnings or errors are generated if the function can complete.
- "message" A message is generated when atypical events occur.
- "warning" Warnings are generated when atypical events occur.
- "error" Errors are generated when atypical events occur, stopping the completion of the function.

return_match (logical of length 1) If TRUE, include the part of the input matched by regex in the output object.

return_input (logical of length 1) If TRUE, include the input in the output object.

redundant_names
: (logical of length 1) If TRUE, remove any occurrence of the a supertaxon's name at the start of the taxon name. This is useful for removing the redundant genus information in species binomials.

batch_size
: (numeric of length 1) The number of IDs to look up at once. This only effects querys using "obs_id". If there is an error looking up an ID, reducing this to 1 can prevent it from ruining the whole batch, but it will take longer.

verbosity
: (character of length 1) Controls the printing of progress updates. The following values are possible:

  "none" No progress reports are printed

  "low" Minimal progress reports of a fixed length are printed.

  "high" Lots of information is printed depending on the amount of the input.

Value

Returns an object of type taxmap

Examples

```
## Not run:
# Extract embedded classifications from UNITE FASTA file offline
file_path <- system.file("extdata", "unite_general_release.fasta", package = "metacoder")
sequences <- ape::read.FASTA(file_path)
x <- extract_taxonomy(sequences,
                      regex = "^(.*)\\|(.*)\\|(.*)\\|.*\\|(.*)$",
                      key = c(seq_name = "obs_info", seq_id = "obs_info",
                              other_id = "obs_info", "class"),
                      class_regex = "^(.*)__(.*)$",
                      class_key = c(unite_rank = "taxon_info", "name"),
                      class_sep = ";")
# Look up taxonomic data online using sequence ID
# This might take a while. The speed is dependent on NCBI's servers.
file_path <- system.file("extdata", "ncbi_basidiomycetes.fasta", package = "metacoder")
sequences <- ape::read.FASTA(file_path)
y <- extract_taxonomy(sequences,
                      regex = "^.*\\|(.*)\\|.*\\|(.*)\\|(.*)$",
                      key = c(gi_no = "obs_info", "obs_id", desc = "obs_info"),
                      database = "ncbi")

## End(Not run)
```

---

filter_obs *Filter observations with a list of conditions*

Description

Filter observations in a taxmap object with a list of conditions. Any column name that appears in obs_data(.data) can be used as if it was a vector on its own. See filter for inspiration and more information.

Usage

```
filter_obs(.data, ..., unobserved = TRUE)
```

Arguments

.data	taxmap
...	One or more filtering conditions. This can be one of two things: integer One or more indexes of obs_data logical A TRUE/FALSE vector of length equal to the number of rows in obs_data Any column name that appears in obs_data(.data) can be used as if it was a vector on its own.
unobserved	(logical of length 1) If TRUE, preserve taxa even if all of their observations are filtered out. If FALSE, remove taxa for which all observations were filtered out. Note that only taxa that are unobserved due to this filtering will be removed; there might be other taxa without observations to begin with that will not be removed.

Value

An object of type taxmap

See Also

Other dplyr-like functions: arrange_obs, arrange_taxa, filter_taxa, mutate_obs, mutate_taxa, sample_frac_obs, sample_frac_taxa, sample_n_obs, sample_n_taxa, select_obs, select_taxa, transmute_obs, transmute_taxa

Examples

```
# Filter by sequence name, but preserve all taxa
filter_obs(unite_ex_data_3, grepl("Lachnum", seq_name))
# Filter by sequence name and only keep taxa with sequences that pass the filter
filter_obs(unite_ex_data_3, grepl("Lachnum", seq_name), unobserved = FALSE)
```

---

filter_taxa *Filter taxa with a list of conditions*

Description

Filter taxa in a taxmap object with a list of conditions. Any column name that appears in taxon_data(.data) can be used as if it was a vector on its own. See filter for inspiration and more information.

Usage

```
filter_taxa(.data, ..., subtaxa = FALSE, supertaxa = FALSE,
  taxonless = FALSE, reassign_obs = TRUE, reassign_taxa = TRUE,
  invert = FALSE)
```

Arguments

.data	taxmap
...	One or more filtering conditions. This can be one of three things: character One or more taxon_ids integer One or more indexes of taxon_data logical A TRUE/FALSE vector of length equal to the number of rows in taxon_data Any column name that appears in taxon_data(.data) can be used as if it was a vector on its own.
subtaxa	(logical of length 1) If TRUE, include subtaxa of taxa passing the filter.
supertaxa	(logical of length 1) If TRUE, include supertaxa of taxa passing the filter.
taxonless	(logical of length 1) If TRUE, include observations even if the taxon they are assigned to is filtered out. observation assigned to removed taxa will be assigned to NA. See the reassign option below for further complications.
reassign_obs	(logical of length 1) If TRUE, observations assigned to removed taxa will be reassigned to the closest supertaxon that passed the filter. If there are no supertaxa of such an observation that passed the filter, they will be filtered out if taxonless is TRUE.
reassign_taxa	(logical of length 1) If TRUE, subtaxa of removed taxa will be reassigned to the closest supertaxon that passed the filter.
invert	(logical of length 1) If TRUE, do NOT include the selection. This is different than just replacing a == with a != because this option negates the selection after taking into account the subtaxa and supertaxa options. This is useful for removing a taxon and all its subtaxa for example.

Value

An object of type taxmap

See Also

Other dplyr-like functions: arrange_obs, arrange_taxa, filter_obs, mutate_obs, mutate_taxa, sample_frac_obs, sample_frac_taxa, sample_n_obs, sample_n_taxa, select_obs, select_taxa, transmute_obs, transmute_taxa

Examples

```
# Remove singleton taxa, but reassign singletons to supertaxa that pass filter
filter_taxa(unite_ex_data_3, n_obs > 1)
# Remove singleton taxa and associated seqeuence data
filter_taxa(unite_ex_data_3, n_obs > 1, taxonless = FALSE, reassign_obs = FALSE)
```

```
# Subset to a single taxon and its subtaxa
filter_taxa(unite_ex_data_3, name == "Basidiomycota", subtaxa = TRUE)
# Remove a taxon and its subtaxa
filter_taxa(unite_ex_data_3, name == "Basidiomycota", subtaxa = TRUE, invert = TRUE)
# Remove taxa, reassigning supertaxa and subtaxa
filter_taxa(unite_ex_data_3, unite_rank != "p")
```

---

genbank_ex_data *Fungal ITS Genbank refseq*

Description

A dataset containing information for 299 sequences obtained from NCBI using the following query:

Usage

```
genbank_ex_data
```

Format

An object of type taxmap

Details

```
(18s[All Fields] AND 28s[All Fields]) AND "basidiomycetes"[porgn] AND (refseq[filter] AND ("700"[SLEN
```

Source

http://www.ncbi.nlm.nih.gov/nuccore

Examples

```
## Not run:

file_path <- system.file("extdata", "ncbi_basidiomycetes.fasta", package = "metacoder")
sequences <- ape::read.FASTA(file_path)
genbank_ex_data <- extract_taxonomy(sequences,
                                    regex = "^.*\\|(.*)\\|.*\\|(.*)\\|(.*)$",
                                  key = c(gi_no = "obs_info", "obs_id", desc = "obs_info"),
                                    database = "ncbi")

## End(Not run)
```

hierarchies *Get classification of taxa*

Description

Get classification strings of taxa in an object of type taxmap. Each classification is constructed by concatenating the taxon ids of the given taxon and its supertaxa.

Usage

```
hierarchies(obj, subset = 1:nrow(obj$taxon_data), sep = ";")
```

Arguments

obj	(taxmap)
subset	(character) The taxon_idss or indexes in taxon_data to get classifications for.
sep	(character of length 1) The character(s) to place between taxon IDs

Value

character of length equal to subset

See Also

Other taxon_funcs: n_obs_1, n_obs, n_subtaxa_1, n_subtaxa, n_supertaxa

its1_ex_data *Example of ITS1 fungal data*

Description

A dataset containing information from 170 Chytridiomycota ITS sequences from the ITS1 reference database.

Usage

```
its1_ex_data
```

Format

An object of type taxmap

Examples

```
## Not run:

file_path <- system.file("extdata", "its1_chytridiomycota_hmm.fasta", package = "metacoder")
sequences <- ape::read.FASTA(file_path)
its1_ex_data <- extract_taxonomy(sequences,
                                 regex = "^(.*)\\|(.*)\\|tax_id:(.*)\\|(.*)$",
                                 key = c("obs_id", taxon_name = "taxon_info",
                                         "taxon_id", description = "obs_info"),
                                 database = "ncbi")

## End(Not run)
```

`layout_functions` *Layout functions*

Description

Functions used to determine graph layout. Calling the function with no parameters returns available function names. Calling the function with only the name of a function returns that function. Supplying a name and a `graph` object to run the layout function on the graph.

Usage

```
layout_functions(name = NULL, graph = NULL, intitial_coords = NULL,
  effort = 1, ...)
```

Arguments

`name`	(`character` of length 1 OR NULL) name of algorithm. Leave `NULL` to see all options.
`graph`	(`igraph`) The graph to generate the layout for.
`intitial_coords`	(`matrix`) Initial node layout to base new layout off of.
`effort`	(`numeric` of length 1) The amount of effort to put into layouts. Typically determines the the number of iterations.
`...`	(other arguments) Passed to igraph layout function used.

Value

The name available functions, a layout functions, or a two-column matrix depending on how arguments are provided.

Examples

```
# List available function names:
layout_functions()

# Execute layout function on graph:
layout_functions("davidson-harel", igraph::make_ring(5))
```

`metacoder` *Metacoder*

Description

A package for planning and analysis of amplicon metagenomics research projects.

Details

The goal of the `metacoder` package is to provide a set of tools for:

- Standardized parsing of taxonomic information from diverse resources.
- A set of functions to manipulate taxonomic data modeled after dplyr.
- Visualization of statistics distributed over taxonomic classifications and phylogenies.
- Evaluating potential metabarcoding primers for taxonomic specificity.
- Evaluating potential metabarcoding loci for taxonomic discrimination (in development).

To accomplish these goals, 'metacoder' leverages resources from other R packages, interfaces with external programs, and provides novel functions where needed to allow for entire analyses within R.

To learn how to use the package and what it can do view the package vignettes by typing: `browseVignettes("metacoder")`

Documentation

The full documentation can be found online at http://grunwaldlab.github.io/metacoder_documentation.

There is also a short vignette included for offline use that can be accessed by the following code:

```
browseVignettes(package = "metacoder")
```

Most important functions

Parsing taxonomy information:

- `extract_taxonomy`

Dplyr-style manipulations of taxonomic data:

- `arrange_obs`

- arrange_taxa
- filter_obs
- filter_taxa
- mutate_obs
- mutate_taxa
- transmute_obs
- transmute_taxa
- sample_n_obs
- sample_n_taxa
- sample_frac_obs
- sample_frac_taxa
- select_obs
- select_taxa

Taxonomically balanced sub-sampling:

- taxonomic_sample

Plotting taxonomic distribution of data:

- heat_tree

In silico PCR:

- primersearch

Author(s)

Zachary Foster

`mutate_obs` *Add columns to* `taxmap` *objects*

Description

Add columns to the `obs_data` in `taxmap` objects. Any column name that appears in `obs_data(.data)` can be used as if it was a vector on its own. See `mutate` for inspiration and more information.

Usage

```
mutate_obs(.data, ...)
```

Arguments

.data	taxmap
...	One or more column names to add to the new object. Newly created columns can be referenced in the same function call.

Value

An object of type taxmap

See Also

Other dplyr-like functions: arrange_obs, arrange_taxa, filter_obs, filter_taxa, mutate_taxa, sample_frac_obs, sample_frac_taxa, sample_n_obs, sample_n_taxa, select_obs, select_taxa, transmute_obs, transmute_taxa

Examples

```
# Add one or more observation columns
mutate_obs(unite_ex_data_3, x = 1, y = x+2)
```

---

mutate_taxa *Add columns to taxmap objects*

Description

Add columns to the taxon_data in taxmap objects. Any column name that appears in taxon_data(.data) can be used as if it was a vector on its own. See mutate for inspiration and more information.

Usage

```
mutate_taxa(.data, ...)
```

Arguments

.data	taxmap
...	One or more column names to add to the new object. Newly created columns can be referenced in the same function call.

Value

An object of type taxmap

See Also

Other dplyr-like functions: arrange_obs, arrange_taxa, filter_obs, filter_taxa, mutate_obs, sample_frac_obs, sample_frac_taxa, sample_n_obs, sample_n_taxa, select_obs, select_taxa, transmute_obs, transmute_taxa

Examples

```
# Add one or more taxon columns
mutate_taxa(unite_ex_data_3, x = 1, y = x+2)
```

---

n_obs *Count observations in* taxmap

Description

Count observations for each taxon in taxmap. This includes observations for the specific taxon and its subtaxa.

Usage

```
n_obs(obj, subset = 1:nrow(obj$taxon_data))
```

Arguments

obj	(taxmap)
subset	(character) The taxon_ids or indexes in taxon_data to get counts for.

Value

numeric

See Also

Other taxon_funcs: hierarchies, n_obs_1, n_subtaxa_1, n_subtaxa, n_supertaxa

---

n_obs_1 *Count observation assigned in* taxmap

Description

Count observations assigned to a specific taxon in taxmap. This does not include observations assigned to subtaxa.

Usage

```
n_obs_1(obj, subset = 1:nrow(obj$taxon_data))
```

Arguments

obj	(taxmap)
subset	(character) The taxon_ids or indexes in taxon_data to get counts for.

Value

numeric

See Also

Other taxon_funcs: hierarchies, n_obs, n_subtaxa_1, n_subtaxa, n_supertaxa

---

n_subtaxa *Get number of subtaxa*

Description

Get number of subtaxa for each taxon in an object of type taxmap

Usage

```
n_subtaxa(obj, subset = 1:nrow(obj$taxon_data))
```

Arguments

obj	(taxmap)
subset	(character) The taxon_ids or indexes in taxon_data.

Value

numeric

See Also

Other taxon_funcs: hierarchies, n_obs_1, n_obs, n_subtaxa_1, n_supertaxa

---

n_subtaxa_1 *Get number of subtaxa*

Description

Get number of subtaxa for each taxon in an object of type taxmap, not including subtaxa of subtaxa etc. This does not include subtaxa assigned to subtaxa.

Usage

```
n_subtaxa_1(obj, subset = 1:nrow(obj$taxon_data))
```

Arguments

obj	(taxmap)
subset	(character) The taxon_ids or indexes in taxon_data.

Value

numeric

See Also

Other taxon_funcs: hierarchies, n_obs_1, n_obs, n_subtaxa, n_supertaxa

---

n_supertaxa *Get number of supertaxa*

Description

Get number of supertaxa for each taxon in an object of type taxmap

Usage

```
n_supertaxa(obj, subset = 1:nrow(obj$taxon_data))
```

Arguments

obj	(taxmap)
subset	(character) The taxon_ids or indexes in taxon_data.

Value

numeric

See Also

Other taxon_funcs: hierarchies, n_obs_1, n_obs, n_subtaxa_1, n_subtaxa

obs *Get observations associated with taxa*

Description

Given one or more taxa IDs and a taxmap object, return the observation indexes (e.g. sequence information) associated with each taxon.

Usage

```
obs(obj, subset = NULL, recursive = TRUE, simplify = FALSE)
```

Arguments

obj	(taxmap) The taxmap object containing taxon information to be queried.
subset	(character) taxon_ids or indexes of taxon_data for which supertaxa will be returned. Default: All taxa in obj will be used.
recursive	(logical) If FALSE, only return the observation assigned to the specified input taxa, not subtaxa. If TRUE, return all the observations of every subtaxa, etc.
simplify	(logical) If TRUE, then combine all the results into a single vector of unique observation indexes.

Value

If simplify = FALSE, then a list of vectors of observation indexes are returned corresponding to the target argument. If simplify = TRUE, then the observation indexes for all target taxa are returned in a single vector.

See Also

Other taxmap taxonomy functions: roots, subtaxa, supertaxa

obs_data *Return observation data from taxmap*

Description

Return a table of data associated with taxa of and object of type taxmap.

Usage

```
obs_data(obj, row_subset = NULL, col_subset = NULL,
  calculated_cols = TRUE, sort_by = NULL, decreasing = FALSE,
  drop = FALSE)
```

Arguments

obj	(taxmap)
row_subset	(character) The obs_ids of a subset of obj. Default: All rows.
col_subset	(character) The names of columns, either user defined or generated using obs_funcs. Default: All columns.
calculated_cols	(logical of length 1) If TRUE, return calculated columns using functions in taxmap$obs_funcs. These values are calculated each time obs_data is called since their values can change if the data is subset.
sort_by	(character of length 1) The name of a column in obj$obs_data or a function name in obj$obs_funcs. This column will be used to sort the output rows. If NULL, no sorting will be done.
decreasing	(logical of length 1) If TRUE, sort_by order is decreasing.
drop	(logical of length 1) If TRUE, if subset is a single column name, then a vector is returned instead of a data.frame

Value

A data.frame or vector with rows corresponding to taxa in input

---

obs_data_colnames *Get column names of obs_data*

Description

Get column names of obs_data without calculating columns

Usage

```
obs_data_colnames(obj)
```

Arguments

obj	a taxmap object

Value

character

parse_hmp_qiime *Parse HMP QIIME results*

Description

NOTE: Not extensively tested Parses the results of the Human Microbiome Project QIIME analysis of the 16S metagenomic data. This is mostly a wrapper for `extract_taxonomy`.

Usage

```
parse_hmp_qiime(otu_file, mapping_file, min_abundance = 1, max_otus = -1)
```

Arguments

`otu_file`	(`character` of length 1) A file path or URL to the OTU table.
`mapping_file`	(`character` of length 1) A file path or URL to the sample mapping file.
`min_abundance`	(`numeric` of length 1) Do not return rows less abundance. This can make the output object much smaller.
`max_otus`	(`numeric` of length 1) Only parse some number of OTUs. Useful for making small datasets for testing.

Value

A `taxmap` object

parse_mothur_summary *Parse mothur classification summary file*

Description

Parse mothur classification summary file

Usage

```
parse_mothur_summary(file_path, unclassified = FALSE)
```

Arguments

`file_path`	(`character` of length 1) The file path to the input file.
`unclassified`	(`logical` of length 1) If `FALSE`, remove any unclassified rows.

Value

`taxmap`

`parse_taxonomy_table` *Parse taxonomic data in a tsv/csv file*

Description

Parse taxonomic data in a tsv/csv file

Usage

```
parse_taxonomy_table(input, taxon_col, other_col_type = "obs_info",
  header = TRUE, sep = "\t", max_lines = NULL, comment_prefix = "#",
  ...)
```

Arguments

`input`	(`character` of length 1) The file path to the input file or a `data.frame`.
`taxon_col`	(named `integer` of length 1) The index of the column with taxonomic information, named by the type of information. A negative index is interpreted as the number of columns from the last. The name of the column can have to following values: `taxon_id` A unique numeric id for a taxon for a particular `database` (e.g. ncbi accession number). Requires an internet connection. `name` The name of a taxon. Not necessarily unique, but are interpretable by a particular `database`. Requires an internet connection. `class` A list of taxa information that constitutes the full taxonomic classification from broad to specific (see `class_rev`) for a particular `database`. Individual taxa are separated by the `class_sep` argument and the information is parsed by the `class_regex` and `class_key` arguments.
`other_col_type`	(`character`) The type of the other columns no specified by `taxon_col`. Can be `"taxon_info"` or `"obs_info"`.
`header`	(`logical` of length 1) If `TRUE`, the first row of the file is the column names.
`sep`	(`character` of length 1) The character(s) that separate each column in each row. Can be a regular expression.
`max_lines`	(`integer` of length 1) The maximum number of lines to read from the file.
`comment_prefix`	(`character`) One or more characters that appear at the start of a line indicating that the line is a comment and not part of the data.
`...`	Passed to `extract_taxonomy`.

Value

`taxmap`

pr2_ex_data *Example of PR2 SSU data*

Description

A dataset containing information from 249 Stramenopile sequences from the PR2 reference database.

Usage

```
pr2_ex_data
```

Format

An object of type taxmap

Examples

```
## Not run:

file_path <- system.file("extdata", "pr2_stramenopiles_gb203.fasta", package = "metacoder")
sequences <- ape::read.FASTA(file_path)
pr2_ex_data <- extract_taxonomy(sequences,
                                regex = "^(.*\\..*?)\\|(.*)$",
                                key = c("obs_id", "class"),
                                class_sep = "\\|")

## End(Not run)
```

primersearch *Use EMBOSS primersearch for in silico PCR*

Description

A pair of primers are aligned against a set of sequences. The location of the best hits, quality of match, and predicted amplicons are returned. Requires the EMBOSS tool kit (http://emboss.sourceforge.net/) to be installed.

Usage

```
primersearch(input, forward, reverse, mismatch = 5, ...)

## S3 method for class 'character'
primersearch(input, forward, reverse, mismatch = 5, ...)

## S3 method for class 'taxmap'
primersearch(input, forward, reverse, mismatch = 5,
  sequence_col = "sequence", result_cols = NULL, ...)
```

Arguments

input	(character)
forward	(character of length 1) The forward primer sequence
reverse	(character of length 1) The reverse primer sequence
mismatch	An integer vector of length 1. The percentage of mismatches allowed.
...	Unused.
sequence_col	(character of length 1) The name of the column in obs_data that has the input sequences.
result_cols	(character) The names of columns to include in the output. By default, all output columns are included.

Value

An object of type taxmap

Installing EMBOSS

The command-line tool "primersearch" from the EMBOSS tool kit is needed to use this function. How you install EMBOSS will depend on your operating system:

Linux:

Open up a terminal and type:

```
sudo apt-get install emboss
```

Mac OSX:

The easiest way to install EMBOSS on OSX is to use homebrew. After installing homebrew, open up a terminal and type:

```
brew install homebrew/science/emboss
```

NOTE: This has not been tested by us yet.

Windows:

There is an installer for Windows here:

ftp://emboss.open-bio.org/pub/EMBOSS/windows/mEMBOSS-6.5.0.0-setup.exe

NOTE: This has not been tested by us yet.

Examples

```
## Not run:
result <- primersearch(rdp_ex_data,
                       forward = c("U519F" = "CAGYMGCCRCGGKAAHACC"),
                       reverse = c("Arch806R" = "GGACTACNSGGGTMTCTAAT"),
                       mismatch = 10)

heat_tree(result,
          node_size = n_obs,
          node_label = name,
          node_color = prop_amplified,
```

```
            node_color_range = c("red", "yellow", "green"),
            node_color_trans = "linear",
            node_color_interval = c(0, 1),
            layout = "fruchterman-reingold")

## End(Not run)
```

`print.taxmap` *Print a* `taxmap` *object*

Description

Print a `taxmap` object

Usage

```
## S3 method for class 'taxmap'
print(x, max_rows = 7, ...)
```

Arguments

`x`	object to print
`max_rows`	(`integer` of length 1) The maximum number of rows to print in tables.
`...`	Not used

`qualitative_palette` *The default qualitative color palette*

Description

Returns the default color palette for qualitative data

Usage

```
qualitative_palette()
```

Value

`character` of hex color codes

Examples

```
qualitative_palette()
```

quantative_palette *The default quantative color palette*

Description

Returns the default color palette for quantative data.

Usage

```
quantative_palette()
```

Value

`character` of hex color codes

Examples

```
quantative_palette()
```

rdp_ex_data *Example of RDP Archea data*

Description

A dataset containing information from 400 Archaea 16S sequences from the RDP reference database.

Usage

```
rdp_ex_data
```

Format

An object of type `taxmap`

Source

http://rdp.cme.msu.edu/

Examples

```
## Not run:

file_path <- system.file("extdata", "rdp_current_Archaea_unaligned.fa", package = "metacoder")
sequences <- ape::read.FASTA(file_path)
rdp_ex_data <- extract_taxonomy(sequences,
                                regex = "^(.*?) (.*)\\tLineage=(.*)",
                               key = c(id = "obs_info", description = "obs_info", "class"),
                                class_regex = "(.+?);(.*?);",
                                class_key = c("name", "taxon_info"))

## End(Not run)
```

`read_fasta` *Read a FASTA file*

Description

Read a FASTA file

Usage

```
read_fasta(file_path, subset = NULL)
```

Arguments

`file_path`	(`character` of length 1) The path to a file to read.
`subset`	(`numeric`) Indexes of entries to return. If not `NULL`, the file will first be indexed without loading the whole file into RAM.

Value

names `character`

`remove_redundant_names` *Remove the redundant taxon names*

Description

Remove the names of parent taxa in the beginning of their children's names in a `taxmap` object. This is useful for removing genus names in binomials.

Usage

```
remove_redundant_names(obj, name_col, all_supertaxa = TRUE)
```

Arguments

obj	a taxmap object
name_col	(character of length 1) The name of a column in obj$taxon_data
all_supertaxa	(logical of length 1) If TRUE, check all supertaxa for redundant names instead of just the one immediate supertaxa.

Value

character

---

roots *Get root taxa*

Description

Return the root taxa for a taxmap object. Can also be used to get the roots of a subset of taxa.

Usage

```
roots(obj, subset = NULL, index = FALSE)
```

Arguments

obj	(taxmap) The taxmap object containing taxon information to be queried.
subset	(character) Taxon IDs for which supertaxa will be returned. Default: All taxon in obj will be used.
index	(logical) If TRUE, return the indexes of roots in taxon_data instead of taxon_ids

Value

character

See Also

Other taxmap taxonomy functions: obs, subtaxa, supertaxa

`sample_frac_obs` *Sample a proportion of observations from* `taxmap`

Description

Randomly sample some proportion of observations from a `taxmap` object. Weights can be specified for observations or the taxa they are taxmap by. See sample_frac for the inspiration for this function.

Usage

```
sample_frac_obs(.data, size = 1, replace = FALSE, taxon_weight = NULL,
  obs_weight = NULL, use_supertaxa = TRUE, collapse_func = mean, ...)
```

Arguments

`.data`	(`taxmap`) The object to sample from.
`size`	(`numeric` of length 1) The proportion of observations to sample.
`replace`	(`logical` of length 1) If `TRUE`, sample with replacement.
`taxon_weight`	(`numeric`) Non-negative sampling weights of each taxon. If `use_supertaxa` is `TRUE`, the weights for each taxon in an observation's classification are supplied to `collapse_func` to get the observation weight. The expression given is evaluated in the context of `taxon_data`. In other words, any column name that appears in `taxon_data(.data)` can be used as if it was a vector on its own. If `obs_weight` is also specified, the two weights are multiplied (after `taxon_weight` for each observation is calculated).
`obs_weight`	(`numeric`) Sampling weights of each observation. The expression given is evaluated in the context of `obs_data`. In other words, any column name that appears in `obs_data(.data)` can be used as if it was a vector on its own. If `taxon_weight` is also specified, the two weights are multiplied (after `taxon_weight` for each observation is calculated).
`use_supertaxa`	(`logical` of length 1) Affects how the `taxon_weight` is used. If `TRUE`, the weights for each taxon in an observation's classification are multiplied to get the observation weight. Otherwise, just the taxonomic level the observation is assign to it considered.
`collapse_func`	(`function` of length 1) If `taxon_weight` option is used and `supertaxa` is `TRUE`, the weights for each taxon in an observation's classification are supplied to `collapse_func` to get the observation weight. This function should take numeric vector and return a single number.
`...`	Additional options are passed to `filter_obs`.

Value

An object of type `taxmap`

See Also

Other dplyr-like functions: arrange_obs, arrange_taxa, filter_obs, filter_taxa, mutate_obs, mutate_taxa, sample_frac_taxa, sample_n_obs, sample_n_taxa, select_obs, select_taxa, transmute_obs, transmute_taxa

Examples

```
# Subsample without replacement, keeping all taxa
sample_frac_obs(unite_ex_data_3, 0.1)
# Subsample without replacement and remove unsampled taxa
sample_frac_obs(unite_ex_data_3, 0.1, unobserved = FALSE)
# Subsample with taxon weight
sample_frac_obs(unite_ex_data_3, 0.1, unobserved = FALSE, taxon_weight = 1 / n_obs)
# Sample with replacement
sample_frac_obs(unite_ex_data_3, 10, replace = TRUE)
```

---

sample_frac_taxa *Sample a proportion of taxa from* taxmap

Description

Randomly sample some proportion of taxa from a taxmap object. Weights can be specified for taxa or the observations assigned to them. See sample_frac for the inspiration for this function.

Usage

```
sample_frac_taxa(.data, size = 1, taxon_weight = NULL, obs_weight = NULL,
  use_subtaxa = TRUE, collapse_func = mean, ...)
```

Arguments

.data	(taxmap) The object to sample from.
size	(numeric of length 1) The proportion of taxa to sample.
taxon_weight	(numeric) Non-negative sampling weights of each taxon. The expression given is evaluated in the context of taxon_data. In other words, any column name that appears in taxon_data(.data) can be used as if it was a vector on its own. If obs_weight is also specified, the two weights are multiplied (after obs_weight for each taxon is calculated).
obs_weight	(numeric) Sampling weights of each observation. The weights for each observation assigned to a given taxon are supplied to collapse_func to get the taxon weight. If use_subtaxa is TRUE then the observations assigned to every subtaxa are also used. The expression given is evaluated in the context of obs_data. In other words, any column name that appears in obs_data(.data) can be used as if it was a vector on its own. If taxon_weight is also specified, the two weights are multiplied (after obs_weight for each observation is calculated).

use_subtaxa (logical of length 1) Affects how the obs_weight option is used. If TRUE, the weights for each taxon in an observation's classification are multiplied to get the observation weight. Otherwise, just the taxonomic level the observation is assign to it considered.

collapse_func (function of length 1) If taxon_weight is used and supertaxa is TRUE, the weights for each taxon in an observation's classification are supplied to collapse_func to get the observation weight. This function should take numeric vector and return a single number.

... Additional options are passed to filter_taxa.

Value

An object of type taxmap

See Also

Other dplyr-like functions: arrange_obs, arrange_taxa, filter_obs, filter_taxa, mutate_obs, mutate_taxa, sample_frac_obs, sample_n_obs, sample_n_taxa, select_obs, select_taxa, transmute_obs, transmute_taxa

Examples

```
# subsample taxa, preserving shared supertaxa
sample_frac_taxa(unite_ex_data_3, 0.1, supertaxa = TRUE)
# subsample taxa using weights, preserving subtaxa
sample_frac_taxa(unite_ex_data_3, 0.01, subtaxa = TRUE,
                 taxon_weight = ifelse(unite_rank == "g" & n_subtaxa > 3, 1, 0))
```

---

sample_n_obs *Sample n observations from* taxmap

Description

Randomly sample some number of observations from a taxmap object. Weights can be specified for observations or the taxa they are taxmap by. See sample_n for the inspiration for this function.

Usage

```
sample_n_obs(.data, size, replace = FALSE, taxon_weight = NULL,
  obs_weight = NULL, use_supertaxa = TRUE, collapse_func = mean, ...)
```

Arguments

.data	(taxmap) The object to sample from.
size	(numeric of length 1) The number of observations to sample.
replace	(logical of length 1) If TRUE, sample with replacement.
taxon_weight	(numeric) Non-negative sampling weights of each taxon. If use_supertaxa is TRUE, the weights for each taxon in an observation's classification are supplied to collapse_func to get the observation weight. The expression given is evaluated in the context of taxon_data. In other words, any column name that appears in taxon_data(.data) can be used as if it was a vector on its own. If obs_weight is also specified, the two weights are multiplied (after taxon_weight for each observation is calculated).
obs_weight	(numeric) Sampling weights of each observation. The expression given is evaluated in the context of obs_data. In other words, any column name that appears in obs_data(.data) can be used as if it was a vector on its own. If taxon_weight is also specified, the two weights are multiplied (after taxon_weight for each observation is calculated).
use_supertaxa	(logical of length 1) Affects how the taxon_weight is used. If TRUE, the weights for each taxon in an observation's classification are multiplied to get the observation weight. Otherwise, just the taxonomic level the observation is assign to it considered.
collapse_func	(function of length 1) If taxon_weight option is used and supertaxa is TRUE, the weights for each taxon in an observation's classification are supplied to collapse_func to get the observation weight. This function should take numeric vector and return a single number.
...	Additional options are passed to filter_obs.

Value

An object of type taxmap

See Also

Other dplyr-like functions: arrange_obs, arrange_taxa, filter_obs, filter_taxa, mutate_obs, mutate_taxa, sample_frac_obs, sample_frac_taxa, sample_n_taxa, select_obs, select_taxa, transmute_obs, transmute_taxa

Examples

```
# Subsample without replacement, keeping all taxa
sample_n_obs(unite_ex_data_3, 100)
# Subsample without replacement and remove unsampled taxa
sample_n_obs(unite_ex_data_3, 100, unobserved = FALSE)
# Subsample with taxon weight
sample_n_obs(unite_ex_data_3, 100, unobserved = FALSE, taxon_weight = 1 / n_obs)
# Sample with replacement
sample_n_obs(unite_ex_data_3, 10000, replace = TRUE)
```

`sample_n_taxa` *Sample n taxa from* `taxmap`

Description

Randomly sample some number of taxa from a `taxmap` object. Weights can be specified for taxa or the observations assigned to them. See sample_n for the inspiration for this function.

Usage

```
sample_n_taxa(.data, size, taxon_weight = NULL, obs_weight = NULL,
  use_subtaxa = TRUE, collapse_func = mean, ...)
```

Arguments

`.data`	(`taxmap`) The object to sample from.
`size`	(`numeric` of length 1) The number of taxa to sample.
`taxon_weight`	(`numeric`) Non-negative sampling weights of each taxon. The expression given is evaluated in the context of `taxon_data`. In other words, any column name that appears in `taxon_data(.data)` can be used as if it was a vector on its own. If `obs_weight` is also specified, the two weights are multiplied (after `obs_weight` for each taxon is calculated).
`obs_weight`	(`numeric`) Sampling weights of each observation. The weights for each observation assigned to a given taxon are supplied to `collapse_func` to get the taxon weight. If `use_subtaxa` is `TRUE` then the observations assigned to every subtaxa are also used. The expression given is evaluated in the context of `obs_data`. In other words, any column name that appears in `obs_data(.data)` can be used as if it was a vector on its own. If `taxon_weight` is also specified, the two weights are multiplied (after `obs_weight` for each observation is calculated).
`use_subtaxa`	(`logical` of length 1) Affects how the `obs_weight` option is used. If `TRUE`, the weights for each taxon in an observation's classification are multiplied to get the observation weight. Otherwise, just the taxonomic level the observation is assign to it considered.
`collapse_func`	(`function` of length 1) If `taxon_weight` is used and `supertaxa` is `TRUE`, the weights for each taxon in an observation's classification are supplied to `collapse_func` to get the observation weight. This function should take numeric vector and return a single number.
`...`	Additional options are passed to `filter_taxa`.

Value

An object of type `taxmap`

See Also

Other dplyr-like functions: arrange_obs, arrange_taxa, filter_obs, filter_taxa, mutate_obs, mutate_taxa, sample_frac_obs, sample_frac_taxa, sample_n_obs, select_obs, select_taxa, transmute_obs, transmute_taxa

Examples

```
# subsample taxa, preserving shared supertaxa
sample_n_taxa(unite_ex_data_3, 100, supertaxa = TRUE)
# subsample taxa using weights, preserving subtaxa
sample_n_taxa(unite_ex_data_3, 10, subtaxa = TRUE,
              taxon_weight = ifelse(unite_rank == "g" & n_subtaxa > 3, 1, 0))
```

---

select_obs *Subset columns in a taxmap object*

Description

Subsets obs_data columns in a taxmap object. Takes and returns a taxmap object. Any column name that appears in obs_data(.data) can be used as if it was a vector on its own. See select for more information.

Usage

```
select_obs(.data, ...)
```

Arguments

.data	taxmap
...	One or more column names to return in the new object. This can be one of three things: expression with unquoted column name The name of a column in taxon_data typed as if it was a varaible on its own. numeric Indexes of columns in taxon_data To match column names with a character vector, use matches("my_col_name"). To match a logical vector, convert it to a column index using which.

Value

An object of type taxmap

See Also

Other dplyr-like functions: arrange_obs, arrange_taxa, filter_obs, filter_taxa, mutate_obs, mutate_taxa, sample_frac_obs, sample_frac_taxa, sample_n_obs, sample_n_taxa, select_taxa, transmute_obs, transmute_taxa

Examples

```
# subset observation columns
select_obs(unite_ex_data_3, other_id, seq_id)
```

`select_taxa` *Subset columns in a* `taxmap` *object*

Description

Subsets `taxon_data` columns in a `taxmap` object. Takes and returns a `taxmap` object. Any column name that appears in `taxon_data(.data)` can be used as if it was a vector on its own. See `select` for more information.

Usage

```
select_taxa(.data, ...)
```

Arguments

`.data`	`taxmap`
`...`	One or more column names to return in the new object. This can be one of three things: `expression with unquoted column name` The name of a column in `taxon_data` typed as if it was a varaible on its own. `numeric` Indexes of columns in `taxon_data` To match column names with a character vector, use `matches("my_col_name")`. To match a logical vector, convert it to a column index using `which`.

Value

An object of type `taxmap`

See Also

Other dplyr-like functions: `arrange_obs`, `arrange_taxa`, `filter_obs`, `filter_taxa`, `mutate_obs`, `mutate_taxa`, `sample_frac_obs`, `sample_frac_taxa`, `sample_n_obs`, `sample_n_taxa`, `select_obs`, `transmute_obs`, `transmute_taxa`

Examples

```
# subset taxon columns
select_taxa(unite_ex_data_3, name)
```

silva_ex_data *Example dataset from SILVA*

Description

https://www.arb-silva.de/

Usage

```
silva_ex_data
```

Format

An object of type taxmap

Examples

```
## Not run:

file_path <- system.file("extdata", "silva_nr99.fasta", package = "metacoder")
sequences <- ape::read.FASTA(file_path)
silva_ex_data <- extract_taxonomy(sequences,
                                  regex = "^(.*?) (.*)$",
                                  key = c(id = "obs_info", "class"),
                                  class_sep = ";")

## End(Not run)
```

subtaxa *Get all subtaxa of a taxon*

Description

Return the taxon IDs or taxon_data indexes of all subtaxa in an object of type taxmap

Usage

```
subtaxa(obj, subset = NULL, recursive = TRUE, simplify = FALSE,
  include_input = FALSE, index = FALSE)
```

Arguments

obj	(taxmap) The taxmap object containing taxon information to be queried.
subset	(character) taxon_ids or indexes of taxon_data for which supertaxa will be returned. Default: All taxa in obj will be used.
recursive	(logical) If FALSE, only return the subtaxa one level below the target taxa. If TRUE, return all the subtaxa of every subtaxa, etc.
simplify	(logical) If TRUE, then combine all the results into a single vector of unique values.
include_input	(logical) If TRUE, the input taxa are included in the output
index	(logical) If TRUE, return the indexes of supertaxa in taxon_data instead of taxon_ids

Value

If simplify = FALSE, then a list of vectors are returned corresponding to the target argument. If simplify = TRUE, then the unique values are returned in a single vector.

See Also

Other taxmap taxonomy functions: obs, roots, supertaxa

Examples

```
## Not run:
subtaxa(contaminants, subset = 1:10)
## End(Not run)
```

---

supertaxa *Get all supertaxa of a taxon*

Description

Return the taxon IDs or taxon_data indexes of all supertaxa (i.e. all taxa the target taxa are a part of) in an object of type taxmap.

Usage

```
supertaxa(obj, subset = NULL, recursive = TRUE, simplify = FALSE,
  include_input = FALSE, index = FALSE, na = FALSE)
```

Arguments

obj	(taxmap) The taxmap object containing taxon information to be queried.
subset	(character) taxon_ids or indexes of taxon_data for which supertaxa will be returned. Default: All taxa in obj will be used.
recursive	(logical) If FALSE, only return the supertaxa one level above the target taxa. If TRUE, return all the supertaxa of every supertaxa, etc.
simplify	(logical) If TRUE, then combine all the results into a single vector of unique values.
include_input	(logical) If TRUE, the input taxa are included in the output
index	(logical) If TRUE, return the indexes of supertaxa in taxon_data instead of taxon_ids
na	(logical) If TRUE, return NA where information is not available.

Value

If simplify = FALSE, then a list of vectors are returned corresponding to the subset argument. If simplify = TRUE, then unique values are returned in a single vector.

See Also

Other taxmap taxonomy functions: obs, roots, subtaxa

Examples

```
## Not run:
supertaxa(contaminants, subset = 1:10)
## End(Not run)
```

---

taxmap *Create an instance of* taxmap

Description

Create an instance of taxmap containing observations taxmap by a taxonomy.

Usage

```
taxmap(taxon_ids, supertaxon_ids, taxa = taxon_ids,
  obs_taxon_ids = numeric(0), taxon_data = NULL, obs_data = NULL,
  taxon_funcs = list(n_obs = n_obs, n_obs_1 = n_obs_1, n_supertaxa =
  n_supertaxa, n_subtaxa = n_subtaxa, n_subtaxa_1 = n_subtaxa_1, hierarchies =
  hierarchies), obs_funcs = list())
```

Arguments

taxon_ids	(character) These are unique identifiers for taxa. They will be coerced into characters.
supertaxon_ids	(character OR (numeric)) Supertaxa of taxa. If a character vector, then these should be in the same format as taxon_ids. If a numeric vector, then it is interpreted as the indexes of taxon_ids. Taxa without parents should be NA.
taxa	(character) Objects representing taxa. Currently, these can be anything, but this might change in the future.
obs_taxon_ids	(character OR (numeric)) Taxon assignments of observations. Supertaxa of taxa. If a character vector, then these should be in the same format as taxon_ids. If a numeric vector, then it is interpreted as the indexes of taxon_ids.
taxon_data	(data.frame) A table with rows pretaining to taxa
obs_data	A (data.frame) A table with rows pretaining to obs_taxa
taxon_funcs	(list of named functions) These the values produced by these functions will be accessible as a column in taxon_data. The first parameter of each function should be a single taxmap object.
obs_funcs	(list of named functions) These the values produced by these functions will be accessible as a column in obs_data. The first parameter of each function should be a single taxmap object.

Value

An object of type taxmap

---

taxonomic_sample *Recursivly sample a set of taxonomic assignments*

Description

Recursivly sample a set of observations with taxonomic assignments and an associated taxonomy.

Usage

```
taxonomic_sample(taxmap_data, max_counts = c(), min_counts = c(),
  max_children = c(), min_children = c(), obs_filters = list(),
  subtaxa_filters = list(), stop_conditions = list(), ...)
```

Arguments

taxmap_data	(An object of type taxmap)
max_counts	(numeric) A named vector that defines that maximum number of observations in for each level specified. The names of the vector specifies that level each number applies to. If more than the maximum number of observations exist for a given taxon, it is randomly subsampled to this number.

`min_counts` (`numeric`) A named vector that defines that minimum number of observations in for each level specified. The names of the vector specifies that level each number applies to.

`max_children` (`numeric`) A named vector that defines that maximum number of subtaxa per taxon for each level specified. The names of the vector specifies that level each number applies to. If more than the maximum number of subtaxa exist for a given taxon, they are randomly subsampled to this number of subtaxa.

`min_children` (`numeric`) A named vector that defines that minimum number of subtaxa in for each level specified. The names of the vector specifies that level each number applies to.

`obs_filters` (`list` of `function(observations, id)`) A list of functions that take a data structure containing the information of multiple observations and a taxon id. Returns a object of the same type with some of the observations potentially removed.

`subtaxa_filters` (`list` of `function(observations, id)`) A list of functions that take a data structure containing the information of multiple subtaxa IDs and the current taxon id. Returns a object of the same type with some of the subtaxa potentially removed. If a function returns `NULL`, then no observations for the current taxon are returned.

`stop_conditions` (`list` of `function(id)`) A list of functions that take the current taxon id. If any of the functions return `TRUE`, the observations for the current taxon are returned rather than looking for observations of subtaxa, stopping the recursion.

`...` Additional parameters are passed to all of the function options.

Value

Returns an object of type `taxmap`

Examples

```
## Not run:
#Plot data before subsampling
heat_tree(unite_ex_data_3,
          node_size = n_obs,
          node_color = n_obs,
          node_label = n_obs)

# Subsampling
subsampled <- taxonomic_sample(unite_ex_data_3,
                               max_counts = c("4" = 20, "7" = 5),
                               min_counts = c("7" = 3))

# Remove unobserved taxa and plot
heat_tree(subset(subsampled, n_obs > 0, unobserved = FALSE),
          node_size = n_obs,
          node_color = n_obs,
```

```
            node_label = n_obs)

## End(Not run)
```

---

taxon_data *Return taxon data from* taxmap

Description

Return a table of data associated with taxa of and object of type taxmap.

Usage

```
taxon_data(obj, row_subset = NULL, col_subset = NULL,
  calculated_cols = TRUE, sort_by = NULL, decreasing = FALSE,
  drop = FALSE)
```

Arguments

obj	(taxmap)
row_subset	(character) The taxon_ids of a subset of obj. Default: All rows.
col_subset	(character) The names of columns, either user defined or generated using taxon_funcs. Default: All columns.
calculated_cols	(logical of length 1) If TRUE, return calculated columns using functions in taxmap$taxon_funcs. These values are calculated each time taxon_data is called since their values can change if the data is subset.
sort_by	(character of length 1) The name of a column in obj$taxon_data or a function name in obj$taxon_funcs. This column will be used to sort the output rows. If NULL, no sorting will be done.
decreasing	(logical of length 1) If TRUE, sort_by order is decreasing.
drop	(logical of length 1) If TRUE, if subset is a single column name, then a vector is returned instead of a data.frame

Value

A data.frame or vector with rows corresponding to taxa in input

taxon_data_colnames *Get column names of taxon_data*

Description

Get column names of taxon_data without calculating columns

Usage

```
taxon_data_colnames(obj)
```

Arguments

obj	a taxmap object

Value

character

transmute_obs *Replace columns in taxmap objects*

Description

Replace columns of obs_data in taxmap objects. Any column name that appears in obs_data(.data) can be used as if it was a vector on its own. See transmute for inspiration and more information.

Usage

```
transmute_obs(.data, ...)
```

Arguments

.data	taxmap
...	One or more column names to add to the new object. Newly created columns can be referenced in the same function call.

Value

An object of type taxmap

See Also

Other dplyr-like functions: arrange_obs, arrange_taxa, filter_obs, filter_taxa, mutate_obs, mutate_taxa, sample_frac_obs, sample_frac_taxa, sample_n_obs, sample_n_taxa, select_obs, select_taxa, transmute_taxa

Examples

```
# Replace all observation columns with new columns
transmute_obs(unite_ex_data_3, x = 1, y = x+2)
```

`transmute_taxa` *Replace columns in* `taxmap` *objects*

Description

Replace columns of `taxon_data` in `taxmap` objects. Any column name that appears in `taxon_data(.data)` can be used as if it was a vector on its own. See `transmute` for inspiration and more information.

Usage

```
transmute_taxa(.data, ...)
```

Arguments

`.data`	`taxmap`
`...`	One or more column names to add to the new object. Newly created columns can be referenced in the same function call.

Value

An object of type `taxmap`

See Also

Other dplyr-like functions: `arrange_obs`, `arrange_taxa`, `filter_obs`, `filter_taxa`, `mutate_obs`, `mutate_taxa`, `sample_frac_obs`, `sample_frac_taxa`, `sample_n_obs`, `sample_n_taxa`, `select_obs`, `select_taxa`, `transmute_obs`

Examples

```
# Replace all taxon columns with new columns
transmute_taxa(unite_ex_data_3, x = 1, y = x+2)
```

unite_ex_data_1 *Example of UNITE fungal ITS data*

Description

A dataset containing information from 449 sequences from the UNITE reference database.

Usage

```
unite_ex_data_1
```

Format

An object of type taxmap

Source

https://unite.ut.ee/

Examples

```
## Not run:

file_path <- system.file("extdata", "unite_general_release.fasta", package = "metacoder")
sequences <- ape::read.FASTA(file_path)
unite_ex_data_1 <- extract_taxonomy(sequences[!grepl(pattern = "\\|UDB", names(sequences))],
                                    regex = "^(.*)\\|(.*)\\|(.*)\\|.*\\|(.*)$",
                                    key = c(name = "obs_info", "obs_id",
                                        other_id = "obs_info", tax_string = "obs_info"),
                                    database = "ncbi")

## End(Not run)
```

unite_ex_data_2 *Example of UNITE fungal ITS data*

Description

A dataset containing information from 500 sequences from the UNITE reference database.

Usage

```
unite_ex_data_2
```

Format

An object of type taxmap

Source

https://unite.ut.ee/

Examples

```
## Not run:

file_path <- system.file("extdata", "unite_general_release.fasta", package = "metacoder")
sequences <- ape::read.FASTA(file_path)
unite_ex_data_2 <- extract_taxonomy(sequences,
                                    regex = "^(.*)\\|(.*)\\|(.*)\\|.*\\|(.*)$",
                                    key = c(seq_name = "obs_info", seq_id = "obs_info",
                                            other_id = "obs_info", "class"),
                                    class_regex = "^(.*)__(.*)$",
                                    class_key = c(unite_rank = "taxon_info", "name"),
                                    class_sep = ";",
                                    database = "ncbi")

## End(Not run)
```

---

unite_ex_data_3 *Example of UNITE fungal ITS data*

Description

A dataset containing information from 500 sequences from the UNITE reference database.

Usage

```
unite_ex_data_3
```

Format

An object of type taxmap

Source

https://unite.ut.ee/

Examples

```
## Not run:

file_path <- system.file("extdata", "unite_general_release.fasta", package = "metacoder")
sequences <- ape::read.FASTA(file_path)
unite_ex_data_3 <- extract_taxonomy(sequences,
                                    regex = "^(.*)\\|(.*)\\|(.*)\\|.*\\|(.*)$",
                                    key = c(seq_name = "obs_info", seq_id = "obs_info",
                                            other_id = "obs_info", "class"),
                                    class_regex = "^(.*)__(.*)$",
                                    class_key = c(unite_rank = "taxon_info", "name"),
                                    class_sep = ";")

## End(Not run)
```

Index