Package ‘refdb’

September 22, 2022

Type Package

Title A DNA Reference Library Manager

Version 0.1.1

Maintainer Francois Keck <francois.keck@gmail.com>


License GPL-3

URL https://fkeck.github.io/refdb/

BugReports https://github.com/fkeck/refdb/issues

Encoding UTF-8

Depends R (>= 3.1.0)

Imports tibble, readr, dplyr, stringr, tidyr, rentrez, bold, taxize, xml2, bioseq, ape, igraph, ggplot2, ggraph, yaml, rlang, markdown, leaflet

RoxygenNote 7.2.1

Suggests testthat (>= 3.0.0), covr, DT, knitr, forcats

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Francois Keck [aut, cre, cph] (<https://orcid.org/0000-0002-3323-4167>)

Repository CRAN

Date/Publication 2022-09-22 17:00:05 UTC
R topics documented:

- check_fields
- check_fields
db
- filter_scores
- get_ncbi_taxonomy
- igraph_from_taxo
- make_ncbi_table
- ncbi_taxo_rank
- process_geno_ncbi
- redb_check_seq_conflict
- redb_check_seq_homogeneity
- redb_check_seq_tax_conflict
- redb_check_seq_typed
- redb_clean_seq_crop_primers
- redb_clean_seq_remove_gaps
- redb_clean_seq_remove_sideN
- redb_clean_tax_harmonize_nomenclature
- redb_clean_tax_NA
- redb_clean_tax_remove_blank
- redb_clean_tax_remove_extra
- redb_clean_tax_remove_subsp
- redb_clean_tax_remove_uncertainty
- redb_export_dada2
- redb_export_idtaxa
- redb_export_mothur
- redb_fill_tax_downstream
- redb_fill_tax_upstream
- redb_filter_ref_length
- redb_filter_ref_scope
- redb_filter_seq_ambiguous
- redb_filter_seq_duplicates
- redb_filter_seq_homopolymers
- redb_filter_seq_length
- redb_filter_seq_primer
- redb_filter_seq_stopcodon
- redb_filter_tax_na
- redb_filter_tax_precision
- redb_get_fields
- redb_import_BOLD
- redb_import_NCBI
- redb_merge
- redb_plot_map
- redb_plot_seqlen_hist
- redb_plot_tax_barplot
- redb_plot_tax_tree
- redb_plot_tax_treemap
- redb_report
- redb_sample_tax
**check_fields**

Internal check for fields

**Usage**

```
check_fields(x, what = c("source", "id", "taxonomy", "sequence", "marker"))
```

**Arguments**

- `x` a reference database (tibble object).
- `what` a vector of fields to be checked.

**Value**

Invisible or error.

---

**fields_dbs**

Functions to set fields for various databases

**Description**

Functions to set fields for various databases

**Usage**

```
refdb_set_fields_NCBI(x)
refdb_set_fields_BOLD(x)
refdb_set_fields_PR2(x)
refdb_set_fields_diatbarcode(x)
```
get_ncbi_taxonomy

Arguments

x a reference database.

Value

The function returns x with updated attributes.

Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
refdb_set_fields_BOLD(lib)
```

filter_scores Scores for filtering operations

Description

Scores for filtering operations

Usage

`.filter_seq_length(x, gaps)`

Arguments

x a reference database
gaps should gaps be included.

Value

a numeric vector

get_ncbi_taxonomy Get NCBI taxonomy

Description

Download and parse NCBI taxonomy records

Usage

`get_ncbi_taxonomy(id, verbose = TRUE)`
igraph_from_taxo

Arguments
id
A vector of id for records in the NCBI Taxonomy database.
verbose
print information in the console.

Value
A tibble with each row corresponding to an id and each column to a taxonomic level.

igraph_from_taxo  Create a graph from a taxonomic table

Description
Create a graph representation from a taxonomic classification included in a reference database. For this function to work, taxonomic fields must be set.

Usage
igraph_from_taxo(x, cols = NULL)

Arguments
x
a reference database (tibble).
cols
an optional vector of column names to use a subset of columns.

Value
An igraph object representing taxonomic relationships.

make_ncbi_table  Parse NCBI XML and make a table

Description
Parse NCBI XML and make a table

Usage
make_ncbi_table(x)

Arguments
x
A XML nodeset.

Value
A tibble.
### ncbi_taxo_rank

**Taxonomic ranks of the NCBI Taxonomy database**

**Description**

Taxonomic ranks of the NCBI Taxonomy database

**Usage**

```r
ncbi_taxo_rank()
```

**Value**

a vector of ordered ranks

---

### process_geo_ncbi

**Process coordinate column returned by NCBI**

**Description**

Process coordinate column returned by NCBI

**Usage**

```r
process_geo_ncbi(x, col = "lat_lon")
```

**Arguments**

- `x` : NCBI dataframe.
- `col` : column name containing geographical coordinates.

**Value**

NCBI dataframe.
**refdb_check_seq_conflict**

*Check for conflicts in sequences*

**Description**

Check for conflicts in sequences

**Usage**

`refdb_check_seq_conflict(x, na_omit = TRUE)`

**Arguments**

- `x`  
a reference database.
- `na_omit`  
if `FALSE` conflicts involving NA taxonomic names are also reported.

**Value**

A list of two-columns tibbles reporting duplicated sequences with different taxonomy.

**Examples**

```r
lib <- read.csv(system.file("extdata", "ephem.csv", package = "refdb"))
lib <- refdb_set_fields(lib,
                         taxonomy = c(family = "family_name",
                         genus = "genus_name",
                         species = "species_name"),
                         sequence = "DNA_seq",
                         marker = "marker")
refdb_check_seq_conflict(lib)
```

**refdb_check_seq_homogeneity**

*Check for genetic homogeneity of taxa*

**Description**

This function assesses the genetic similarity among sequences within each taxa. It takes user-defined thresholds (one threshold per taxonomic level) to warn about sequences which are singularly different (based on median distance) from the others. Sequences in the reference database must be aligned.

**Usage**

`refdb_check_seq_homogeneity(x, levels, min_n_seq = 3)`
Arguments

x     a reference database (sequences must be aligned).
levels  a named vector of genetic similarity thresholds. Names must correspond to
taxonomic levels (taxonomic fields) and values must be included in the interval
[0, 1]. For example to assess homogeneity at 5 percents (within species) and 10
percents (within genus): levels = c(species = 0.05, genus = 0.1)
min_n_seq  the minimum number of sequences for a taxon to be tested.

Details

For every tested taxonomic levels, the algorithm checks all sequences in every taxa (for which the
total number of sequence is > min_n_seq) In each taxon, the pairwise distance matrix among all the
sequences belonging to this taxon is computed. A sequence is tagged as suspicious and returned by
the function if its median genetic distance from the other sequences is higher than the threshold set
by the user (levels argument).

Value

A dataframe reporting suspicious sequences whose median distance to other sequences of the same
taxon is greater than the specified threshold. The first column "level_threshold_homogeneity" indi-
cates the lowest taxonomic level for which the threshold has been exceeded and the second column
"value_threshold_homogeneity" gives the computed median distance.

Examples

lib <- read.csv(system.file("extdata", "homogeneity.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_check_seq_homogeneity(lib, levels = c(species = 0.05, genus = 0.1))

refdb_check_tax_conflict
Check for conflicts in taxonomy

Description

Check for conflicts in taxonomy

Usage

refdb_check_tax_conflict(x)

Arguments

x     a reference database.
refdb_check_tax_typo

Value

A list of two-columns tibbles reporting for each taxonomic level the taxa with identical names but different upstream taxonomy.

Examples

```r
lib <- read.csv(system.file("extdata", "ephem.csv", package = "refdb"))
lib <- refdb_set_fields(lib,
  taxonomy = c(family = "family_name",
               genus = "genus_name",
               species = "species_name"),
  sequence = "DNA_seq",
  marker = "marker")
refdb_check_tax_conflict(lib)
```

---

refdb_check_tax_typo  
Check for typos in taxonomic names

Description

This function uses the generalized Levenshtein (edit) distance to identify possible issue with taxonomic names.

Usage

```r
refdb_check_tax_typo(x, tol = 1)
```

Arguments

- `x`  
a reference database.
- `tol`  
the edit distance below which two taxonomic names are reported.

Value

A list of two-columns tibbles reporting for each taxonomic level the pairs of taxonomic names sharing the same upstream taxonomy and for which the generalized Levenshtein (edit) distance is below the `tol` value.

Examples

```r
lib <- read.csv(system.file("extdata", "ephem.csv", package = "refdb"))
lib <- refdb_set_fields(lib,
  taxonomy = c(family = "family_name",
               genus = "genus_name",
               species = "species_name"),
  sequence = "DNA_seq",
  marker = "marker")
refdb_check_tax_typo(lib)
```
refdb_clean_seq_crop_primers

*Crop genetic sequences with a set of primers*

**Description**

Crop genetic sequences with a set of primers

**Usage**

```r
refdb_clean_seq_crop_primers(
  x,
  primer_forward,
  primer_reverse,
  max_error_in = 0.1,
  max_error_out = 0.1,
  include_primers = TRUE
)
```

**Arguments**

- **x**
  - a reference database with a defined sequence field.
- **primer_forward**
  - primer forward.
- **primer_reverse**
  - primer reverse.
- **max_error_in**, **max_error_out**
  - maximum error for a match (frequency based on primer length).
- **include_primers**
  - a logical indicating whether the detected primers are included in the cropped sequences.

**Value**

A reference database.

**Examples**

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_clean_seq_crop_primers(lib, "AGT", "TTTA")
```
**refdb_clean_seq_remove_gaps**

*Remove gaps from genetic sequences*

**Description**

Remove gaps from genetic sequences

**Usage**

```r
refdb_clean_seq_remove_gaps(x)
```

**Arguments**

- `x` 
  a reference database with a defined sequence field.

**Value**

A reference database.

**Examples**

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_clean_seq_remove_gaps(lib)
```

---

**refdb_clean_seq_remove_sideN**

*Remove repeated side N from genetic sequences*

**Description**

Remove repeated side N from genetic sequences

**Usage**

```r
refdb_clean_seq_remove_sideN(x, side = "both")
```

**Arguments**

- `x` 
  a reference database with a defined sequence field.
- `side` 
  which side to clean. Can be one of "left", "right" or "both" (default).

**Value**

A reference database.
Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_clean_tax_remove_sideN(lib)
```

```r
refdb_clean_tax_harmonize_nomenclature

Harmonize taxonomic name nomenclature
```

Description

Harmonize taxonomic name nomenclature

Usage

```r
refdb_clean_tax_harmonize_nomenclature(x, cols = NULL)
```

Arguments

- `x`: a reference database.
- `cols`: an optional vector of column names. If `NULL` (default), the function is applied to the columns associated with the taxonomy and organism fields.

Value

A reference database.

Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_clean_tax_harmonize_nomenclature(lib)
```

```r
refdb_clean_tax_NA

Convert missing taxonomic names to NA
```

Description

Convert missing taxonomic names to NA

Usage

```r
refdb_clean_tax_NA(x, cols = NULL, hybrid = TRUE, uncertain = FALSE)
```
refdb_clean_tax_remove_blank

Arguments

x a reference database.
cols an optional vector of column names. If NULL (default), the function is applied to the columns associated with the taxonomy and organism fields.
hybrid hybrids are converted to NA (default TRUE).
uncertain taxa with qualifiers of uncertainty (cf., aff., etc.) are converted to NA (default FALSE).

Value

A reference database.

Examples

lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_clean_tax_remove_blank(lib)

Description

Remove blank characters from taxonomic names

Usage

refdb_clean_tax_remove_blank(x, cols = NULL)

Arguments

x a reference database.
cols an optional vector of column names. If NULL (default), the function is applied to the columns associated with the taxonomy and organism fields.

Value

A reference database.

Examples

lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_clean_tax_remove_blank(lib)
---

**refdb_clean_tax_remove_extra**

*Remove extra words from taxonomic names*

**Description**

Remove extra words from taxonomic names

**Usage**

```r
refdb_clean_tax_remove_extra(x, cols = NULL)
```

**Arguments**

- `x`: a reference database.
- `cols`: an optional vector of column names. If `NULL` (default), the function is applied to the columns associated with the *taxon*omy and *organ*ism fields.

**Details**

As the function can match words like "g.", "s." or "x", which can have a signification in some nomenclatures, it is recommended to execute `refdb_clean_tax_harmonize_nomenclature` first.

**Value**

A reference database.

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_clean_tax_remove_extra(lib)
```

---

**refdb_clean_tax_remove_subsp**

*Remove subspecific information from taxonomic names*

**Description**

Remove subspecific information from taxonomic names

**Usage**

```r
refdb_clean_tax_remove_subsp(x, cols = NULL)
```

**Arguments**

- `x`: a reference database.
- `cols`: an optional vector of column names. If `NULL` (default), the function is applied to the columns associated with the *taxon*omy and *organ*ism fields.
Value

A reference database.

Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_clean_tax_remove_subsp(lib)
```

Description

Remove terms indicating uncertainty in taxonomic names

Usage

```r
refdb_clean_tax_remove_uncertainty(x, cols = NULL)
```

Arguments

- `x`: a reference database.
- `cols`: an optional vector of column names. If NULL (default), the function is applied to the columns associated with the taxonomy and organism fields.

Value

A reference database.

Warning

Marks of taxonomic uncertainty provided by specialists are not without value. The consequences of their deletion must be well understood by the user before using this function.

Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_clean_tax_remove_uncertainty(lib)
```
refdb_export_dada2  
Export reference database for DADA2

Description
Write reference database in formats which can be used with the functions of the package `dada2`.

Usage
```r
refdb_export_dada2(x, file, mode = "taxonomy")
```

Arguments
- `x`  
a reference database.
- `file`  
a path to the file to be written.
- `mode`  
character string to determine the type of file to produce. Use "taxonomy" to produce a file for function `assignTaxonomy` or "species" to produce a file for function `assignSpecies`.

Value
No return value, called for side effects.

Examples
```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_export_dada2(lib, tempfile())
```

refdb_export_idtaxa  
Export reference database for DECIPHER (IDTAXA)

Description
Write a reference database in file formats which can be used to train the IDTAXA classifier implemented in DECIPHER.

Usage
```r
refdb_export_idtaxa(x, file, taxid = FALSE)
```
Arguments

- **x**: a reference database.
- **file**: a file path without extension. This will be used to create a .fasta file and two .txt files.
- **taxid**: should the taxid file be generated (can be very slow with large databases)

Details

The function generates three files:
- A fasta file containing the sequences with their IDs. This file must be imported as a DNAStringSet to be used with DECIPHER, using eg:
  ```r
  Biostrings::readDNAStringSet("ex_seqs.fasta")
  ```
- A text file containing the sequence taxonomic assignment. This file must be imported as a character vector to be used with DECIPHER, using eg:
  ```r
  readr::read_lines("ex_taxo.txt")
  ```
- A text file ("taxid") containing the taxonomic ranks associated with each taxon. This is an asterisk delimited file which must be imported as a dataframe (see LearnTaxa), using eg:
  ```r
  readr::read_delim("ex_ranks.txt", col_names = c("Index", "Name", "Parent", "Level", "Rank"), delim = ",", quote = "")
  ```

The taxid file can be very slow to write for large datasets. Therefore it is not generated by default.

Value

No return value, called for side effects.

Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_export_idtaxa(lib, tempfile())
```

Description

Export reference database for Mothur

Usage

`refdb_export_mothur(x, file)`

Arguments

- **x**: a reference database.
- **file**: a file path. This will be used to create a .fasta file and a .txt file.
refdb_fill_tax_downstream

Value

No return value, called for side effects.

Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_export_mothur(lib, tempfile())
```

Description

Replace NA values in taxonomic classification using upstream ranks.

Usage

```r
refdb_fill_tax_downstream(x, qualifier = "indet.")
```

Arguments

- `x`: a reference database.
- `qualifier`: a string to add the new labels. Default ensure that `refdb_clean_tax_NA` will correctly identify the label as NA.

Value

A reference database.

See Also

`refdb_fill_tax_upstream` to replace NA values using downstream data.

Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_fill_tax_downstream(lib)
```
**refdb_fill_tax_upstream**

*Fill missing data in taxonomy*

**Description**

Replace NA values in taxonomic classification using downstream ranks.

**Usage**

```r
refdb_fill_tax_upstream(x, qualifier = "undef.")
```

**Arguments**

- `x`: a reference database.
- `qualifier`: a string to add the new labels. Default ensure that `refdb_clean_tax_NA` will correctly identify the label as NA.

**Value**

A reference database.

**See Also**

`refdb_fill_tax_downstream` to replace terminal NA values using upstream data.

**Examples**

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_fill_tax_upstream(lib)
```

---

**refdb_filter_ref_scope**

*Filter records by taxonomic scope of studies*

**Description**

Filter records by taxonomic scope of studies.

**Usage**

```r
refdb_filter_ref_scope(x, max_tax)
```
Arguments

- **x**: a reference database (tibble).
- **max_tax**: the maximum (widest) taxonomic focus of the study.

Details

A reference field (one or more columns) must be set to use this function. If reference is not available (NA) for a record, the record is not dropped.

Value

A tibble (filtered reference database).

Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
lib$refs <- rep("REF_1", nrow(lib))
lib <- refdb_set_fields(lib, reference = "refs")
refdb_filter_ref_scope(lib, max_tax = "family_name")
```

---

**refdb_filter_seq_ambiguous**

*Filter sequences based on their number of ambiguous character.*

Description

Filter sequences based on their number of ambiguous character.

Usage

```
refdb_filter_seq_ambiguous(x, max_ambig = 3L, char = "N")
```

Arguments

- **x**: a reference database.
- **max_ambig**: maximum number of ambiguous character.
- **char**: characters interpreted as ambiguous (vector).

Value

A tibble (filtered reference database).
refdb_filter_seq_duplicates

Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_filter_seq_ambiguous(lib)
```

Description

Filter duplicated sequences. This is based both on sequences and taxonomy. NA values are assumed to be comparable.

Usage

```r
refdb_filter_seq_duplicates(x)
```

Arguments

- `x`: a reference database.

Value

A tibble (filtered reference database).

Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_filter_seq_duplicates(lib)
```

refdb_filter_seq_homopolymers

Filter sequences based on their number of repeated character.

Description

Filter sequences based on their number of repeated character.

Usage

```r
refdb_filter_seq_homopolymers(x, max_len = 16L)
```
Arguments

- `x`: a reference database.
- `max_len`: maximum number of repeated character (homopolymer).

Value

A tibble (filtered reference database).

Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_filter_seq_homopolymers(lib)
```

```r
refdb_filter_seq_length

Filter sequences based on their number of character.

Description

Filter sequences based on their number of character.

Usage

`refdb_filter_seq_length(x, min_len = NULL, max_len = NULL, gaps = FALSE)`

Arguments

- `x`: a reference database.
- `min_len`, `max_len`: minimum and maximum sequence lengths. Use NULL (default) to ignore.
- `gaps` if TRUE gaps are accounted.

Value

A tibble (filtered reference database).

Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_filter_seq_length(lib, 50L)
```
**refdb_filter_seq_primer**

Filter sequences based on the presence of primers.

**Description**

Filter sequences based on the presence of primers.

**Usage**

```r
refdb_filter_seq_primer(
  x,
  primer_forward = NULL,
  primer_reverse = NULL,
  max_error_forward = 0.1,
  max_error_reverse = 0.1
)
```

**Arguments**

- `x` a reference database.
- `primer_forward` forward primer.
- `primer_reverse` reverse primer.
- `max_error_forward`, `max_error_reverse` maximum error for match (frequency base on primer length).

**Value**

A tibble (filtered reference database).

**Examples**

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_filter_seq_primer(lib, "ACTA")
```
refdb_filter_seq_stopcodon

Filter sequences based on their number of stop codons.

Description

Filter sequences based on their number of stop codons.

Usage

refdb_filter_seq_stopcodon(x, max_stop = 0, code, codon_frame = NA)

Arguments

x

a reference database.

max_stop

maximum number of stop codons.

code

an integer indicating the genetic code to use for translation (see genetic-codes).

codon_frame

an integer giving the nucleotide position where to start translation. If NA (the default), the three different frames are tested and the frame producing the lowest number of stop codons will be used.

Value

A tibble (filtered reference database).

Examples

lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_filter_seq_stopcodon(lib, code = 5)

refdb_filter_tax_na

Filter records NA taxa

Description

Remove records where taxa is NA if it is not the only representant of the upper clade. Note that the function maybe slow on large datasets. //EXPERIMENTAL//

Usage

refdb_filter_tax_na(x)
refdb_filter_tax_precision

Arguments

x a reference database. (column name of the reference database).

Value

A tibble (filtered reference database).

Examples

lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_filter_tax_precision(lib)

Description

Filter records based on their taxonomic precision.

Usage

refdb_filter_tax_precision(x, min_tax)

Arguments

x a reference database.

min_tax minimum taxonomic level (column name of the reference database).

Value

A tibble (filtered reference database).

Examples

lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_filter_tax_precision(lib, min_tax = "family_name")
refdb_get_fields | Get fields of a reference database

Description
Get fields of a reference database

Usage
refdb_get_fields(x, silent = FALSE)

Arguments

- `x` a reference database.
- `silent` if TRUE silently and invisibly returns fields.

Value
The list of fields is returned invisibly.

Examples
```r
lib <- read.csv(system.file("extdata", "ephem.csv", package = "refdb"))
refdb_get_fields(lib)
```

refdb_import_BOLD | Download and import BOLD records

Description
This function allows to search and download data from the the BOLD database. Additionally it allows to convert the BOLD taxonomic classification into the NCBI classification using the NCBI Taxonomy database.

Usage
```r
refdb_import_BOLD(
  taxon = NULL,
  ids = NULL,
  bin = NULL,
  container = NULL,
  institutions = NULL,
  researchers = NULL,
  geo = NULL,
  ncbi_taxo = TRUE,
```

refdb_import_BOLD

full = FALSE,
verbose = TRUE
)

Arguments

taxon, ids, bin, container, institutions, researchers, geo
terms used to search the BOLD database. See bold_seqspec from which they are inherited.

ncbi_taxo if TRUE (the default) the BOLD taxonomy is replaced by the NCBI taxonomy using refdb_set_ncbitax.

full if TRUE the complete data are returned for every records. If FALSE (the default) only a subset of the most important columns is returned.

verbose print information in the console.

Details

This function uses several functions of the bold package to interface with the BOLD API. The NCBI Taxonomy database is queried using the rentrez package.

Value

A tibble.

Large requests

As detailed in the manual of bold_seqspec, some large requests (e.g. high ranked taxa) can lead to errors. In that case one strategy can be to sequentially download data for lower rank taxa. See https://docs.ropensci.org/bold/#large-data.

See Also

link[bold]bold_stats and bold_seqspec from bold which are used internally.

Examples

## Not run:
goera_bold <- refdb_import_BOLD(taxon = "Goera pilosa", ncbi_taxo = FALSE)

## End(Not run)
refdb_import_NCBI  
Download and import NCBI Nucleotide records

Description
This function allows to search and download data from the NCBI Nucleotide database. Additionally, it uses the NCBI Taxonomy database to get the sequence taxonomic classification.

Usage
```r
refdb_import_NCBI(
  query,
  full = FALSE,
  max_seq_length = 10000,
  seq_bin = 200,
  verbose = TRUE
)
```

Arguments
- `query` a character string with the query.
- `full` a logical. If FALSE (the default), only a subset of the most important fields is included in the result.
- `max_seq_length` a numeric giving the maximum length of sequences to retrieve. Useful to exclude complete genomes.
- `seq_bin` number of sequences to download at once.
- `verbose` print information in the console.

Details
This function uses several functions of the `rentrez` package to interface with the NCBI’s EUtils API.*

Value
A tibble.

Errors
```
Error in curl::curl_fetch_memory(url, handle = handle) : transfer closed with outstanding read data remaining
This error seems to appear with long sequences. You can try to decrease max_seq_length to exclude them.
```
refdb_merge

Examples

```
## Not run:
silo_ncbi <- refdb_import_NCBI("Silo COI")

## End(Not run)
```

---

refdb_merge,Merge reference databases

Description

Merge several reference database by common fields.

Usage

```
refdb_merge(..., keep = "fields_all")
```

Arguments

- `...`: reference databases (tibbles).
- `keep`: determines which columns to keep. Can be "fields_all" (default), "fields_shared" or "all" (see Details).

Details

Columns are merged only if they are associated to the same field.

The `keep` argument determines which columns are returned as follow. "fields_all" (the default) returns all the fields existing in all the reference databases. "fields_shared" returns only the fields shared by all the reference databases. "all" returns all the columns of all the databases. Columns which are not associated to a field are not merged and are prefixed with the name of the object they originated from.

Value

- a merged reference database (tibble).

Examples

```
lib_1 <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib_1 <- refdb_set_fields_BOLD(lib_1)
lib_2 <- lib_1
refdb_merge(lib_1, lib_2)
```
refdb_plot_map

Plot an interactive map

Description
This function generates an interactive map showing the location of the records of a reference database. Note that only records with latitude and longitude data will be displayed.

Usage
refdb_plot_map(x)

Arguments
x
a reference database.

Value
An interactive map object from the leaflet package.

Examples
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
lib <- refdb_set_fields(lib, latitude = "lat", longitude = "lon")
refdb_plot_map(lib)

refdb_plot_seqlen_hist

Plot an histogram of sequence lengths

Description
Plot an histogram of sequence lengths

Usage
refdb_plot_seqlen_hist(x, remove_gaps = TRUE)

Arguments
x
a reference database
remove_gaps
a logical (default TRUE) to control whether gaps (-) should be removed prior computing sequence lengths.
Value

A ggplot object. This means the plot can be further customized using ggplot2 compatible functions.

Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_plot_tax_barplot(lib)
```

---

refdb_plot_tax_barplot

Barplots of the number of records for the most represented taxa

Description

Generate a multipanel plot where, for each taxonomic level, a barplot represent the number of records available in the reference database for the most represented taxa.

Usage

```r
refdb_plot_tax_barplot(x, show_n = 10)
```

Arguments

- `x` a reference database.
- `show_n` an integer value indicating the number of taxa to show in each panel.

Value

A ggplot object.

Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
lib <- refdb_set_fields(lib, latitude = "lat", longitude = "lon")
refdb_plot_tax_barplot(lib)
```
refdb_plot_tax_tree  Reference database taxonomy tree

Description

Represent the hierarchical structure of the taxonomic information of a reference database as a tree.

Usage

refdb_plot_tax_tree(
  x,
  leaf_col = NULL,
  color_col = NULL,
  freq_labels = 0,
  expand_plot = 0.5
)

Arguments

x a reference database.

leaf_col a column name referring to the taxonomic level for the leaves of the tree. If not provided (NULL) the function tries to find a relevant level.

color_col a column name referring to the taxonomic level for the color of the leaves (must be higher or equal to the level of leaf_col). If not provided (NULL) the function tries to find a relevant level.

freq_labels a numeric value to adjust the number of printed labels (minimum frequency). Default is zero which means all non-NA labels are printed.

expand_plot a value to expand the limits of the plot. Useful if the labels are too long.

Details

The underlying graph is computed using the non-exported function igraph_from_taxo.

Value

A ggplot2 (ggraph) object. This means the plot can be further customized using ggplot2 compatible functions.

Examples

lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_plot_tax_tree(lib)
Description

Represent the hierarchical structure of the taxonomic information of a reference database as a set of nested rectangles (treemap).

Usage

```
refdb_plot_tax_treemap(x, cols = NULL, freq_labels = c(0.01, 0.003))
```

Arguments

- **x**: a reference database.
- **cols**: a vector of column names referring to taxonomic levels to include in the treemap. If not provided (NULL) the function tries to find a relevant subset of columns.
- **freq_labels**: a numeric vector of length two to adjust the number of printed labels (see Details).

Only the columns provided in the `cols` argument are represented in the treemap. Large labels are printed for the highest rank, while light text labels are printed for the lowest rank. Intermediate ranks are drawn but their names are not shown. The number of labels printed are determined by `freq_labels`. The first value gives the threshold for the highest rank (large labels) and the second value gives the threshold for the lowest rank (light text labels).

The underlying graph is computed using the non-exported function `igraph_from_taxo`.

Value

A `ggplot2` (`ggraph`) object. This means the plot can be further customized using `ggplot2` compatible functions.

Examples

```r
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_plot_tax_treemap(lib)
```
refdb_report

Compile a report with different checks

Description
This function produce an HTML report to investigate potential issues in a reference database.

Usage
refdb_report(x, file = NULL, view = TRUE)

Arguments
x
a reference database.

file
the file (path) to write the report. If NULL the report is written in the user temp directory.

view
A logical. If TRUE (default), the file is instantly opened in the web browser.

Value
The function invisibly returns the file where the report was written.

Examples

lib <- read.csv(system.file("extdata", "ephem.csv", package = "refdb"))
lib <- refdb_set_fields(lib,
taxonomy = c(family = "family_name",
genus = "genus_name",
.species = "species_name"),
.sequence = "DNA_seq",
.marker = "marker")
tmp <- tempfile()

refdb_report(lib, tmp, view = FALSE)

refdb_sample_tax

Sample records within taxa

Description
This function can be useful to keep a maximum of records per taxa. This function require dplyr dev version to work because of slice_sample. Will be exported once available.
refdb_set_fields

Usage

```
refdb_sample_tax(x, n_max = 10, cols = NULL)
```

Arguments

- `x`: a reference database.
- `n_max`: maximum number of records to keep for each taxa.
- `cols`: an optional vector of column names. If NULL (default), the function is applied to the columns associated with the taxonomy field.

Value

A reference database.

refdb_set_fields

Associate columns to fields

Description

Associate columns to fields so they are recognized and appropriately treated by refdb functions.

Usage

```
refdb_set_fields(
  x,
  source = NA,
  id = NA,
  organism = NA,
  taxonomy = NA,
  sequence = NA,
  marker = NA,
  latitude = NA,
  longitude = NA,
  reference = NA,
  config_yaml = NULL
)
```

Arguments

- `x`: a reference database (tibble).
- `source`: name of the column which contains the data source.
- `id`: name of the column which contains the record IDs.
- `organism`: name of the column which contains the names of the organisms.
- `taxonomy`: a vector of column names.
- `sequence`: name of the column which contains the sequences.
marker  name of the column which contains marker names.
latitude name of the column which contains latitudes (WGS 84)
longitude name of the column which contains longitudes (WGS 84).
reference a vector of column names.
config_yam a file path to a YAML file

Details

Taxonomy reordering. NA to ignore, NULL to delete. Fields set using config_yam always overwrite those set by arguments.

Value

The function returns x with updated attributes.

Examples

```r
lib <- read.csv(system.file("extdata", "ephem.csv", package = "refdb"))
lib <- refdb_set_fields(lib,
taxonomy = c(family = "family_name",
genus = "genus_name",
species = "species_name"),
sequence = "DNA_seq",
marker = "marker")
```

--

**refdb_set_ncbitax**

Replace the current taxonomy using the NCBI Taxonomy database

Description

Replace the current taxonomy using the NCBI Taxonomy database

Usage

```r
refdb_set_ncbitax(
  x,
  min_level = "species",
  force_species_name = TRUE,
  verbose = TRUE
)
```
refdb_write_fields

Arguments

x a reference database (tibble) with one or several columns giving the taxonomy of each record and explicitly indicated in the field taxonomy. See refdb_set_fields.

min_level minimum taxonomic level at which taxonomy should be replaced. Default is the finest level ("species").

force_species_name if TRUE, species not found in NCBI Taxonomy will keep their original names instead of NAs.

verbose print information in the console.

Value

The reference database with the NCBI taxonomy for the genus level and higher ranks. (the original taxonomy above the genus level is removed).

Examples

```r
## Not run:
lib <- read.csv(system.file("extdata", "baetidae_bold.csv", package = "refdb"))
lib <- refdb_set_fields_BOLD(lib)
refdb_set_ncbitax(lib)
## End(Not run)
```

---

refdb_write_fields | Write fields to a file

Description

This function can be used to save fields defined using e.g. refdb_set_fields to a file. Data are saved in YAML and can be read again using the config_yaml argument of refdb_set_fields.

Usage

refdb_write_fields(x, file)

Arguments

x a reference database with some fields to be saved.

file a path to the file to write.

Value

No return value, called for its side effects.
Examples

```r
lib <- read.csv(system.file("extdata", "ephem.csv", package = "refdb"))
tmp <- tempfile()
refdb_write_fields(lib, tmp)

valid_taxo_rank
```

---

**valid_taxo_rank**  
*Ranks considered as valid by refdb*

---

**Description**

Ranks considered as valid by refdb

**Usage**

`valid_taxo_rank()`

**Value**

A vector of ordered ranks.

**References**

This is a simplified version of the list `rank_ref` available in `taxize`.

**Examples**

```r
valid_taxo_rank()
```

---

**xml_extract**  
*Extract XML elements*

---

**Description**

Combine `xml_find_first` and `xml_text` to extract elements.

**Usage**

`xml_extract(x, xpath)`

**Arguments**

- `x`  
  A document, node, or node set.

- `xpath`  
  A string containing a xpath expression.

**Value**

A character vector, the same length as `x`. 
Index

.filter_seq_length (filter_scores), 4
bold_seqspec, 27
check_fields, 3
fields_dbs, 3
filter_scores, 4
genetic-codes, 24
get_ncbi_taxonomy, 4
igraph_from_taxo, 5
make_ncbi_table, 5
ncbi_taxo_rank, 6
process_geo_ncbi, 6

refdb_check_seq_conflict, 7
refdb_check_seq_homogeneity, 7
refdb_check_seq_tax_conflict, 8
refdb_check_tax_typo, 9
refdb_clean_seq_crop_primers, 10
refdb_clean_seq_remove_gaps, 11
refdb_clean_seq_remove_sideN, 11
refdb_clean_tax_harmonize_nomenclature, 12, 14
refdb_clean_tax_NA, 12
refdb_clean_tax_remove_blank, 13
refdb_clean_tax_remove_extra, 14
refdb_clean_tax_remove_subsp, 14
refdb_clean_tax_remove_uncertainty, 15
refdb_export_dada2, 16
refdb_export_idtaxa, 16
refdb_export_mothur, 17
refdb_fill_tax_downstream, 18
refdb_fill_tax_upstream, 19
refdb_filter_ref_scope, 19
refdb_filter_seq_ambiguous, 20
refdb_filter_seq_duplicates, 21
refdb_filter_seq_homopolymers, 21
refdb_filter_seq_length, 22
refdb_filter_seq_primer, 23
refdb_filter_seq_stopcodon, 24
refdb_filter_tax_na, 24
refdb_filter_tax_precision, 25
refdb_get_fields, 26
refdb_import_BOLD, 26
refdb_import_NCBI, 28
refdb_merge, 29
refdb_plot_map, 30
refdb_plot_seqlen_hist, 30
refdb_plot_tax_barplot, 31
refdb_plot_tax_tree, 32
refdb_plot_tax_treemap, 33
refdb_report, 34
refdb_sample_tax, 34
refdb_set_fields, 35, 37
refdb_set_fields_BOLD (fields_dbs), 3
refdb_set_fields_diatbarcode
  (fields_dbs), 3
refdb_set_fields_NCBI (fields_dbs), 3
refdb_set_fields_PR2 (fields_dbs), 3
refdb_set_ncbitax, 27, 36
refdb_write_fields, 37
valid_taxo_rank, 38
xml_extract, 38