Package ‘fossilbrush’

July 22, 2025

Title Automated Cleaning of Fossil Occurrence Data

Version 1.0.6

Description
Functions to automate the detection and resolution of taxonomic and stratigraphic errors in fossil occurrence datasets. Functions were developed using data from the Paleobiology Database.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Depends R (>= 2.10)

Imports igraph, curl, data.table, pbapply, stringdist, stringr,
Matrix, methods

Suggests rmarkdown, knitr

VignetteBuilder knitr

URL https://cran.r-project.org/package=fossilbrush

BugReports https://cran.r-project.org/package=fossilbrush

NeedsCompilation no

Author Joe Flannery-Sutherland [aut, cre] (ORCID:
<https://orcid.org/0000-0001-8232-6773>),
Nussaïbah Raja-Schoob [aut, ctb],
Ádam Kocsis [aut, ctb],
Wolfgang Kiessling [aut]

Maintainer Joe Flannery-Sutherland <josephflannerysutherland@gmail.com>

Repository CRAN

Date/Publication 2025-06-05 11:10:06 UTC

Contents

add_itp . . . 2
add_kingdoms . . . 3
age_ranges . . . 4
assess_duplicates . . . 5
brachios . . . 6
check_taxonomy . . . 7
chrono_scale . . . 10
clean_name . . . 11
densify . . . 12
discrete_ranks . . . 14
find_duplicates . . . 15
find_peaks . . . 15
flag_ranges . . . 16
format_check . . . 18
geog_lookup . . . 19
get_pbdb . . . 19
GTS_2020 . . . 21
GTS_2020_changelog . . . 22
intersect_ranges . . . 22
pacmacro_ranges . . . 23
pbdb_fields . . . 24
pbdb_kingdoms . . . 25
plot_dprofile . . . 25
plot_taxa . . . 26
quantile_coef_density_BMS . . . 27
resolve_duplicates . . . 28
revise_ranges . . . 29
sepkoski . . . 31
sep_code . . . 31
spell_check . . . 32
tgraph . . . 33
threshold_peaks . . . 34
threshold_ranges . . . 35
update_graph . . . 36

Index **38**

`add_itp` *add_itp*

Description

Function to add detected peaks using the output of

Usage

```
add_itp(x, taxon, legend.pos = "topright", exit = TRUE)
```

Arguments

`x` The list output of @seealso threshold_ranges

`taxon` A character vector of length one, specifying one of the taxon names in x to be plotted

`legend.pos` One of topleft, bottomleft, topright or bottomright, or a vector of length two, giving the xy coordinates of the legend. A convenience parameter so that the plot detail can remain unobscured.

`exit` Restore base plotting parameters on function exit (default as a requirement for CRAN). Can be set to false to allow other elements to be aded to a plot

Value

None, the detected peaks are added to an existing density plot

See Also

threshold_ranges. This function should be used to add information to an existing plot from @seealso densify, ensuring that the same taxon name is being used

Examples

```
# load dataset#
data("brachios")
# subsample brachios to make for a short example runtime
set.seed(1)
brachios <- brachios[sample(1:nrow(brachios), 1000),]
# densify ranges
dens <- densify(brachios)
# interpeak thresholding
itp <- threshold_ranges(brachios, win = 8, thresh = 10,
                        rank = "genus", srt = "max_ma", end = "min_ma")
# append the stratigraphically thresholded taxon names to the dataset
# plot the taxon, now identifying the peaks
plot_dprofile(dens, "Atrypa", exit = FALSE)
add_itp(itp, "Atrypa")
```

`add_kingdoms` *add_kingdoms*

Description

Convenience function to add in a kingdom column to a PBDB dataset. This relies on the dataset having a column of phylum-level assignments for occurrences. The kingdom column is a useful addition for filtering very large taxonomically diverse datasets, and adds in an additional level of data which can inform taxonomic cleaning routines like those called by @seealso check_taxonomy

Usage

```
add_kingdoms(x, phylum = "phylum", insert.left = TRUE)
```

Arguments

`x`	A dataframe containing, minimally, phylum-level assignments of the data
`phylum`	A character of length 1 specifing the column in x with the phylum level assignments
`insert.left`	A convenience argument which will make sure that the kingdom column will be inserted in dataframe left immediately to the left of the phylum column

Value

The dataframe x, with the kingdom column inserted

Examples

```
# load dataset
data("brachios")
# add kingdoms to dataset
brachios <- add_kingdoms(brachios)
```

`age_ranges` *roxygen documentation*

Description

age_ranges

Usage

```
age_ranges(
  data,
  taxonomy = "genus",
  srt = "max_ma",
  end = "min_ma",
  mode = "max"
)
```

Arguments

`data`	A three column dataframe comprising one or more character columns of taxonomic names, a numeric column of FADs and a numeric column of LADs
`taxonomy`	A character vector corresponding to one or more of the taxonomic name columns in data
`srt`	A character vector of length one specifying the FAD column in data

`end` A character vector of length one specifying the LAD column in data

`mode` A character vector of length one specifying the type of range table to return: one of max, min or bounds. If not specified by the user, the function behaviour will default to max

Details

Function to derive a range table of taxon names from a stratigraphic occurrence dataset. The default behaviour is to return a total range table - the oldest FAD and youngest LAD for each taxon (max), but the function can also return the minimum range - youngest FAD and oldest LAD (min), or the uncertainty bounds on each FAD and LAD - the two oldest FADs and two youngest LADs (bounds). The names for which ranges are derived are specified by the taxononmy argument, but multiple elements can be given here, allowing taxonomic range for higher clades to also be returned.

Value

A dataframe containing at least four columns: taxon name, FAD, LAD and the taxonomic rank. If taxonomy is of length one, taxonomic rank will be a vector of identical names. If mode = "bounds", there will be two pairs of age columns, denoting the upper and lower bounds on the FAD and LAD for each taxon name

Examples

```
# load dataset
data("brachios")
# derive age ranges
rng <- age_ranges(brachios)
```

`assess_duplicates` *assess_duplicates*

Description

Function to assess and resolve elements with multiple higher classifications in a tgraph object. Assessment is performed based on the topology of the graph they form. Linear paths (i.e. two totally separate paths diverging from the a shared node), rings (divergent paths which only reunite at the highest rank in the tgraph) or more than two divergent paths are treated as distinct. If not any of these cases, the distance between the focal element and the reunion of the divergent paths, along with their subtopologies are assessed and a consensus or preferred path based on the frequency of each path in the tgraph or their completeness returned, or the element judged as having multiple distinct classifications

Usage

```
assess_duplicates(
  x,
  node,
  mode = c("frequency", "completeness"),
  jump = 3,
  plot = FALSE
)
```

Arguments

x A tgraph object

node character vector of elements with multiple higher classifications in x, or a tvertseq object with those same elements as focal

mode The rule to be used in selecting between multiple higher classifications. It is possible for the most complete pathway to also be the most frequent

jump The maximum number of levels between the point of divergence and the point of reunion (if present) for a given path, below which the divergence will be taken as conflicting

plot A logical speciying if the divergent paths should be plotted

Value

A list with as many items as elements with multiple classifications, each recording the assessment for a given element

brachios *brachios*

Description

An example dataset of Palaeozoic brachiopods downloaded from the Paleobiology Database.

Usage

```
brachios
```

Format

An object of class `data.frame` with 151473 rows and 10 columns.

check_taxonomy *check_taxonomy*

Description

Wrapper functions to implement a multi-step cleaning routine for hierarchically structured taxonomic data. The first part of the routine calls @seealso format_check to perform a few presumptive checks on all columns, scanning for non-letter characters and checking the number of words in each string. By default, @seealso clean_name is called to ensure correct formatting as this improves downstream checking. The second part of the routine calls @seealso spell_check to flag spelling discrepancies between names within a given taxonomic group. If chosen, the function can automatically impose the more frequent spelling. The third part of the routine calls @seealso discrete_ranks to flag name re-use at different taxonomic levels. Some of these cases may arise when a name has been unfortunately, (although permissibly) used to refer to groups at different taxonomic levels, or where a higher classification may have been inserted as a placeholder for a missing lower classification. The fourth part of the routine calls @seealso find_duplicates to flag variable higher classifications for a given taxon, including cases where a higher classification is missing for one instance of a taxon, but present for the others. If chosen, @seealso resolve_duplicates is called to ensure a consistent classification is imposed. For cases where a name has been re-used at the same rank for genuinely different taxa (not permissible, unlike name re-use at different ranks) suffixes are added as capital letters, e.g. TaxonA, TaxonB. If any of the automatic cleaning routines are employed (again the default behaviour as clean_name is TRUE by default), the function will return are a cleaned version of the dataset. If the use of suffixes from @seealso resolve_duplicates is not desirable, the function behaviour can be altered so that any suffixes are dropped before returning.

Usage

```
check_taxonomy(
  x,
  ranks = c("phylum", "class", "order", "family", "genus"),
  species = FALSE,
  species_sep = NULL,
 routine = c("format_check", "spell_check", "discrete_ranks", "find_duplicates"),
  report = TRUE,
  verbose = TRUE,
  clean_name = FALSE,
  clean_spell = FALSE,
  thresh = NULL,
  resolve_duplicates = FALSE,
  append = TRUE,
  term_set = NULL,
  collapse_set = NULL,
  jw = 0.1,
  str = 1,
  str2 = NULL,
  alternative = "jaccard",
  q = 1,
  pref_set = NULL,
  suff_set = NULL,
  exclude_set = NULL,
  jump = 3,
  plot = FALSE
)
```

Arguments

`x` A dataframe with hierarchically organised taxonomic information. If x only comprises the taxonomic information, @param ranks does not need to be specified, but the columns must be in order of decreasing taxonomic rank

`ranks` The column names of the taxonomic data fields in x. These must be provided in order of decreasing taxonomic rank

`species` A logical indicating if x contains a species column. As the data must be supplied in hierarchical order, this column will naturally be the last column in x and species-specific spell checks will be performed on this column. NOTE that for the function to work, the species name must be the full species name rather than just the specific epithet, e.g., 'Tyto_alba' rather than just 'alba'.

`species_sep` A character vector of length one specifying the genus name and specific epithet in the species column

- Flagging routine arguments *

`routine` A character vector determining the flagging and cleaning routines to employ. Valid values are format_check (check for non letter characters and the number of words in names), spell_check (flag potential spelling errors), discrete_rank (check that taxonomic names are unique to their rank), duplicate_tax (flag conflicting higher classifications of a given taxon)

`report` A logical of length one determining if the flagging outputs of each cleaning routine should be returned to the user for inspection. This is different to @param verbose, which controls whether flagging should additionally be reported to the user on the console

`verbose` A logical determining if function progress and flagged errors should be reported to the console

- Cleaning routine arguments *

`clean_name` If TRUE, the function will return cleaned versions of the columns in x using the routines in @seealso clean_name. These routines can be altered using the 'term_set' and 'collapse_set' arguments.

`clean_spell` If TRUE, the function will return a cleaned version of the supplied taxonomic dataframe, using the supplied threshold for the similarity method given by method2, to automatically update any names in pairs of flagged synonyms to the more frequent spelling. This is not recommended, however, so the argument is FALSE by default and the threshold left as NULL

`thresh` The threshold for the similarity method given by method2, below which flagged pairs of names will be considered synonyms and resolved automatically. See @seealso spell_check for details on method2

`resolve_duplicates`
If TRUE, the function will return a cleaned version of the supplied taxonomic dataframe, using @seealso resolve_duplicates to resolve conflicts in the way documented by the function. Both spell_clean and tax_clean can both be TRUE to return a dataset cleaned by both methods

`append` If TRUE, any suffixes used during cleaning will be retained in the cleaned version of the data. This is preferable as it ensures that all taxonomic names are rank-discrete and uniquely classified

- Routine specific arguments *

`term_set` A character vector of terms (to be used at all ranks) or a list of rank-specific terms which will be supplied, element-wise as the @param collapse argument called by @seealso clean_name. If a list, this

`collapse_set` A character vector of character strings (to be used at all ranks) or a list of rank-specific strings which will be supplied, element-wise as the @param collapse argument called by @seealso clean_name. If a list, this should be given in descending rank order

`jw` Called by @seealso spell_check

`str` Called by @seealso spell_check

`str2` Called by @seealso spell_check

`alternative` Called by @seealso spell_check

`q` Called by @seealso spell_check

`pref_set` A character vector of prefixes (which will be used at all ranks) or a list of rank-specific prefixes, which will be supplied, element-wise as the @param pref argument called by @seealso spell_check. If a list, this should be given in descending rank order.

`suff_set` A character vector of suffixes (which will be used at all ranks) or a list of rank-specific suffixes, which will be supplied, element-wise as the @param suff argument called by @seealso spell_check. If a list, this should be given in descending rank order.

`exclude_set` A character vector of terms to exclude (which will be used at all ranks) or a list of rank-specific exclusion terms, which will be supplied, element-wise as the @param exclude argument called by @seealso spell_check. If a list, this should be given in descending rank order.

`jump` Called by @seealso resolve_duplicates

`plot` Called by @seealso resolve_duplicates

Details

- Data supply arguments *

Value

A list with elements corresponding to the outputs of the chosen flagging routines (four by default: $formatting, $synonyms, $ranks, $duplicates), plus a cleaned verison of the data ($data) if any of clean_name, clean_spell or resolve_duplicates are TRUE. See @seealso format_check, @seealso spell_clean,

See Also

discrete_ranks and @seealso find_duplicates for details of the structure of the flagging outputs

Examples

```
# load dataset
data("brachios")
# subsample brachios to make for a short example runtime
set.seed(1)
brachios <- brachios[sample(1:nrow(brachios), 1000),]
# define the taxonomic ranks used in the dataset (re-used elsewhere)
b_ranks <- c("phylum", "class", "order", "family", "genus")
# define a list of suffixes to be used at each taxonomic level when scanning for synonyms
b_suff = list(NULL, NULL, NULL, NULL, c("ina", "ella", "etta"))
# scan for errors
brachios <- check_taxonomy(brachios, suff_set = b_suff, ranks = b_ranks)
```

chrono_scale *chrono_scale*

Description

Convenience function to apply user-specified chronostratigraphy to fossil datasets. The function relies on a lookup table generated based on the named intervals in the PBDB in early 2021. First and last interval names in the supplied dataset are matched against this lookup table, by default using ’get("GTS_2020)", to get GTS_2020 numeric ages. If the dataset contains intervals which are not present in the lookup table, they will not be matched and the user will be warned. To get around this possibility, the user can also supply the original numeric ages which will be used as default ages if an interval cannot be matched, to ensure that the returned vectors of numeric ages do not contain NAs.

Usage

```
chrono_scale(
  x,
  tscale = "GTS_2020",
  srt = "early_interval",
  end = "late_interval",
  max_ma = NULL,
  min_ma = NULL,
  verbose = TRUE
)
```

Arguments

x A data.frame containing, minimally two columns corresponding respectively to the first and last intervals of the data. Values should only be present in the second column where the minimum age interval for a row is different to the maximum age interval. Otherwise the values should be NA and the ages returned will be based on the interval specified in the first column, in line with PBDB formatting.

`tscale`	A character string specifying one of the inbuilt chronostratigraphic timescales (currently GTS 2020 only) or a data.frame supplied by the user. If the latter, this must contain columns named 'Interval', 'FAD', 'LAD', specifying the interval names to be matched and their lower and upper age in Ma
`srt`	A character of length 1 specifing the column name of the first interval field in x
`end`	A character of length 1 specifing the column name of the last interval field in x
`max_ma`	If not NULL, a character of length 1 specifing the column name of the original numeric maximum age field in x, to be used as fall back values if interval names cannot all be matched
`min_ma`	If not NULL, a character of length 1 specifing the column name of the original numeric minimum age field in x, to be used as fall back values if interval names cannot all be matched
`verbose`	A logical indicating if warning messages should be displayed or otherwise

Value

The dataframe, x, with two additional columns containing the revised first and last numeric ages of the data, with column names GTS_FAD and GTS_LAD respectively

Examples

```
# example dataset
data("brachios")
# add GTS_2020 dates
brachios <- chrono_scale(brachios, srt = "early_interval", end = "late_interval",
                         max_ma = "max_ma", min_ma = "min_ma")
```

`clean_name` *roxygen documentation*

Description

clean_name

Usage

```
clean_name(x, terms = NULL, collapse = NULL, verbose = FALSE)
```

Arguments

`x` a vector of names to clean. This will be coerced to class character internally

`terms` a character vector of terms to remove from elements of x. Terms are only removed as whole words, rather than if they also happen to occur as strings within elements of x

`collapse` a character vector of strings which should collapsed (i.e. replaced by "", rather than the default " "). If one of the collapse terms is a special regex character, it will need to be escaped, e.g. "\-"

`verbose` A logical of length 1 determining if function progress should be reported to the console

Details

Function which bundles a series of cleaning routines into a single process. First any words in brackets are removed, followed by a series of user-defined terms if given. Next Roman and Arabic numerical are removed, then abbreviations up to five letters (abbreviations are matched by the following dot e.g ABFS.). By default, characters for removal are replaced by a white space to prevent accidental collapse of strings. However, there may be specific cases where a collapse is required and so terms given in collapse are dealt with next. After collapsing, rogue all rogue punctation is removed, then isolated lowercase letters, then isolated groups of capitals up to 5 characters long. Finally, white spaces greater than 1 are removed, along with trailing white space, any remaining strings longer than 2 words subsetted to the first word, the first letter of each string capitalised and zero length strings converted to NA

Value

a character vector the same length as x. Elements which were reduced to zero characters during cleaning are returned as NA

Examples

```
# load dataset
data("brachios")
# clean genus names
gen_clean <- clean_name(brachios$genus)
```

`densify` *densify*

Description

Function to create a matrix of occurrence record densities through geological time from an occurrence dataset. Each column represents a taxon. Each row represents a user defined window of time, with the first row starting at the oldest FAD in the dataset and spanning to the youngest LAD stepwise by the user defined window (default of 1 Ma). Occurrence records are densified by generating a vector of time points from occurrence FAD to occurrence LAD (default step of 0.1 Ma), then tallied in two ways. The first way is a simple histogram count of points-per-window, with the same number of histogram bins as time steps between the overall taxon FAD and LAD. The second way is a kernel density estimate, using a Gaussian kernel with a equally spaced estimatopms equal to the number of timesteps between the overall taxon FAD and LAD

Usage

```
densify(
  x,
  rank = "genus",
  srt = "max_ma",
  end = "min_ma",
  step = 1,
  density = 0.1,
  method = c("histogram", "kernel"),
  ...,
  verbose = TRUE
)
```

Arguments

x An occurrence dataset

rank The column name in x containing the taxon names for which densified columns will be generated

srt A column name in x denoting the occurrence FADs

end A column name in x denoting the occurrence LADs

step A positive integer specifying the time window size (i.e. the duration represented by each row in the output matrix)

density A positive numeric specifying the step size for densifying records. This should ideally be smaller than step

method The method for quantifying occurrence density. By default both histogram and kernel density will be used

... additional arguments passed to @seealso density

verbose A logical determining if function progress should be reported

Value

A list of two sparse matrices, the first containing the histogram counts, the second the kernel density estimates

Examples

```
# load dataset
data("brachios")
# subsample brachios to make for a short example runtime
set.seed(1)
brachios <- brachios[sample(1:nrow(brachios), 1000),]
```

```
# densify ranges
dens <- densify(brachios)
```

discrete_ranks *roxygen documentation*

Description

discrete_ranks

Usage

```
discrete_ranks(x, ranks = NULL)
```

Arguments

x	A dataframe containing hierarchically structured information, for example a table of genus names and their higher taxonomic classifications
ranks	If not NULL, a vector of column names of x, given in rank order. This is useful if x contains columns which are not rank relevant or if columns are not in hierarchical order. If not supplied, the column order in x is used directly and is assumed to be in rank order

Details

Function for checking whether names in one column of a hierarchically organised dataframe reoccur at other levels. Two checks are performed. The first checks for names in adjacent column, assuming that accidental reuse of names at other levels are most likely to occur at an adjacent rank. The second compares across all columns.

Value

A list of two lists. The first list contains names which reoccur at adjacent ranks. The second list contains names that reoccur at any rank

Examples

```
# load dataset
data("brachios")
# define ranks
b_ranks <- c("phylum", "class", "order", "family", "genus")
# run function
flag <- discrete_ranks(brachios, ranks = b_ranks)
```

`find_duplicates` *find_duplicates*

Description

Function to detect and report elements with multiple higher assigments in a hierarchically structured dataframe

Usage

```
find_duplicates(x, ranks = NULL)
```

Arguments

`x`	A hierarchically organised dataframe
`ranks`	The ranks in the dataframe in which to check for elements with multiple higher classifications. The top rank is ignored by default

Value

A dataframe of elements with multiple higher classifications and their ranks

Examples

```
# load dataset
data("brachios")
b_ranks <- c("phylum", "class", "order", "family", "genus")
# run function
flag <- find_duplicates(brachios, ranks = b_ranks)
```

`find_peaks` *find_peaks*

Description

Function to scan, column-wise, a matrix of per-taxon observation density time series. This can be applied to either the histogram or the kernel density output of `densify`, but the latter is recommended. Peaks are detected as local maxima, then smoothed within a local window and tested to distinguish if they are noise or significant. Strict threshold is that the peak is greater than the mean + sd of the window

Usage

```
find_peaks(x, win = 5, verbose = TRUE)
```

Arguments

`x`	A matrix as outputed by `densify`
`win`	A positive integer specifying the window length on either side of a peak (i.e. win 5 will give a total window of 11 - -5 indices + peak index + 5 indices)
`verbose`	A logical determining if function progress should be reported

Value

A list of four, the first three positions containing lists of the peak indices for each taxon, under raw, mean + sd and mean detection regimes. The fourth item is a dataframe of counts of peaks per taxon, 1 row per taxon, 1 column per detection regime

Examples

```
# load dataset
data("brachios")
# subsample brachios to make for a short example runtime
set.seed(1)
brachios <- brachios[sample(1:nrow(brachios), 1000),]
# get density matrix
dens <- densify(brachios)
# run function, using kernel density matrix
pk <- find_peaks(dens$kdensity)
```

---

flag_ranges *flag_ranges*

Description

Function to compare stratigraphic ranges in x to a set of reference ranges from y. A list of two elements is returned. The first is a dataframe summarising the overall error status, specific error counts FAD and LAD differences, and the 95% density distributions of the FAD and LAD errors for each unique taxon in the column of x denoted by the first element of xcols. If a taxon in x is not present in y, it is assigned the status 000 and its other entries in the returned dataframe will be NA. The second element of the returned list is the error code for every individual element of the column of x denoted by the first element of xcols - this will have the same number of rows as x. If x is a range table rather than an occurrence dataset, then the two list elements will have the same number of rows. Ranges for comparison may be supplied directly in y, or y may be another occurrence dataset, in which case

Usage

```
flag_ranges(
  x = NULL,
  y = NULL,
  xcols = c("genus", "max_ma", "min_ma"),
  ycols = NULL,
  flag.diff = 5,
  verbose = TRUE
)
```

Arguments

`x` Stratigraphic range data for taxa as a whole or for individual fossil occurrences

`y` The same as in x. This is the dataset to which ranges will be compared

`xcols` A character vector of length three specifying, in the following order, the taxonomic name, stratigraphic base (FAD) and stratigraphic top (LAD) columns in x.

`ycols` An optional character vector of length three for the same column types as in xcols, but for dataset y. This is useful if the column names differ between the datasets

`flag.diff` A vector of thresholds, given in millions of years which will be used to flag discrepancies between occurrence FADs and LADs with respect to the reference range. This is a convenience parameter so that occurrences with large discrepancies can be quickly identified. Multiple thresholds can be supplied

`verbose` A logical of length one determining if the flagging progress should be reported to the console

Value

A list of two data.frames, the first recording overall error statistics, the second recording error types for each element of x. In the second data.frame, FAD or LAD differences in excess of the supplied threshold(s) are marked with 1, otherwise 0

See Also

age_ranges is called internally to generate the range table for comparison.

Examples

```
# load the example datasets
data(brachios)
data(sepkoski)
# subsample brachios to make for a short example runtime
set.seed(1)
brachios <- brachios[sample(1:nrow(brachios), 1000),]
# update brachios to GTS2020 to match Sepkoski
brachios <- chrono_scale(brachios, srt = "early_interval", end = "late_interval",
                          max_ma = "max_ma", min_ma = "min_ma", verbose = FALSE)
brachios$max_ma <- brachios$newFAD
brachios$min_ma <- brachios$newLAD
# drop occurrences with older LADs than FADs
brachios <- brachios[brachios$max_ma > brachios$min_ma,]
# trim the Sepkoski Compendium to the relevant entries
sepkoski <- sepkoski[which(sepkoski$PHYLUM == "Brachiopoda"),]
# run flag ranges
flg <- flag_ranges(x = brachios, y = sepkoski, ycols = c("GENUS", "RANGE_BASE", "RANGE_TOP"))
```

format_check *format_check*

Description

Function to perform a series of basic formatting checks geared towards taxonomic name data. The function very simply checks for non letter characters in the taxonomic names, that species-level names contain two words, and genus-level and above names contain one word.

Usage

```
format_check(x, ranks, species = FALSE, species_sep = " ", verbose = TRUE)
```

Arguments

`x` A dataframe with hierarchically organised, taxonomic information. If x only comprises the taxonomic information,

`ranks` does not need to be specified, but the columns must be in order of decreasing taxonomic rank @param ranks The column names of the taxonomic data fields in x. These must be provided in order of decreasing taxonomic rank

`species` A logical indicating if x contains a species column. As the data must be supplied in hierarchical order, this column will naturally be the last column in x and species-specific spell checks will be performed on this column.

`species_sep` A character vector of length one specifying the genus name and specific epithet in the species column, if present

`verbose` A logical determining if any flagged errors should be reported to the console

Value

A list of two lists. The first list flags the row indexes of columns whose elements contains non-letter characters. The second list flags the row indexes of columns whose elements do not contain the correct numbers of words

Examples

```
# load dataset
data("brachios")
# define ranks
b_ranks <- c("phylum", "class", "order", "family", "genus")
# run function
flag <- format_check(brachios, ranks = b_ranks)
```

geog_lookup *geog_lookup*

Description

lookup table called by 'get_pbdb'

Usage

```
geog_lookup
```

Format

An object of class `data.frame` with 511 rows and 2 columns.

get_pbdb *get_pbdb*

Description

Function for downloading Paleobiology Database (PBDB) data (saved to disk and/or imported into R) or generating PBDB API-compatible URLs. If downloading data over timespans greater than 100 Ma, the download is performed in 100 Ma chunks to better track the download progress.

Usage

```
get_pbdb(
  taxon = NULL,
  interval = NULL,
  mode = "occurrence",
  res = "all",
  fields = c("ident", "coords", "class"),
  ex_taxon = NULL,
  area = NULL,
  ex_area = NULL,
  invert_area = FALSE,
  litho = NULL,
  invert_litho = FALSE,
  env = NULL,
  ex_env = NULL,
  invert_env = NULL,
  pres = NULL,
  idqual = NULL,
  return_url = FALSE,
  return_data = TRUE,
  save_as = NULL,
  tscale = "ICS_2013",
  wait = Inf
)
```

Arguments

taxon A character vector of taxon names. Prepending a taxon name with ^ will exclude it from the PBDB search. Alternatively @param ex_taxon can be used to do this

interval A numeric vector of length two with positive ages in Ma, or a character vector containing one or two ICS chronostratigraphic interval names

mode A character vector of length one specifying the type of data to return: one of occurrence, collection, taxa, specimen, measurement, strata, diversity, opinion or reference

res A character vector of length one specifying the taxonomic resolution of the dataset: one of all, family, genus species, lump_genus or lump_subgen. The latter two lump multiple occurrences of genera or subgenera within collections into a single representative occurrence

fields A character vector of PBDB vocabulary for additional data fields to download: see https://paleobiodb.org/data1.2/occs/list_doc.html

ex_taxon A character vector of taxon names to exclude from the PBDB search

area If not NULL, then a numeric vector of length four specifying, in order, the min lng, max lng, min lat and max lat of the area from which occurrences will be returned, in decimal degrees (equator = 0 lat, prime meridian = 0 lng). Alternatively, a character vector of regions from which occurrences will be returned: any valid country name or ISO2 code. Continent names and codes are also supported as follows: ATA Antarctica, AFR Africa, ASI Asia, AUS Australia, EUR Europe, IOC Indian Ocean, NOA North America, OCE Oceania,SOA South America

ex_area If not NULL, then a character vector of valid country names or ISO2 codes, as in @param area (), from which occurrences will be excluded from a PBDB search

invert_area If TRUE, then regions specified in area will be excluded from a PBDB search, except for the regions specified in ex_area

litho If not NULL, a character vector of PBDB vocabulary corresponding to which lithologies a PBDB search should return

invert_litho If TRUE, a character vector of PBDB vocabulary corresponding to which lithologies a PBDB search should exclude

env If not NULL, a character vector of PBDB vocabulary corresponding to which environments a PBDB search should return

ex_env If not NULL, a character vector of PBDB vocabulary corresponding to which environments a PBDB search should exclude

invert_env If TRUE, then environments specified in env will be excluded from a PBDB search, except for the environments specified in ex_env

pres A character vector of length one specifying the preservation mode of the occurrences to return: one of regular, form, ichno, or 'form,ichno' `idqual` A character vector of length one specifying the taxonomic certainty of the occurrences to return: one of certain, genus_certain, uncertain, new"

`return_url` If TRUE, the function will return a correctly formatted url suitable for use with curl or similar API functions, comprising the search parameters set by the user

`return_data` If TRUE (default), the downloaded csv will automatically be read into R (this must be assigned to an object)

`save_as` If not NULL, the file name to which the downloaded data will be saved on the disk as a .csv

`tscale` A character vector of length one determining what chronostratigraphic timescale will be applied to the data. "ICS_2013" will retain the PBDB ICS 2013 standard. "GTS_2020" will update all early and late interval ages to the GTS_2020 standard, using a lookup table supplied with the function. Alternatively, the pathway to a custom .csv file with columns Interval, FAD and LAD where Interval are the names of the early and late intervals in the PBDB, and FAD and LAD are the numeric lower and upper boundaries of those intervals

`wait` The maximum wait time for the download in milliseconds, as used by curl. This is set to no wait time by default

Value

either a PBDB API compatible URL or a PBDB dataset

Examples

```
# download Triassic dinosaurs (wait time set to meet CRAN example requirement)
tdinos <- fossilbrush:::get_pbdb(taxon = "Dinosauria", interval = "Triassic", wait = 499)
```

`GTS_2020` *GTS_2020*

Description

lookup table called by 'get_pbdb'

Usage

```
GTS_2020
```

Format

An object of class `data.frame` with 1534 rows and 10 columns.

`GTS_2020_changelog` *GTS_2020_changelog*

Description

changelog of periodic updates made to the GTS2020 table originally published in this package. The purpose of this changelog is to allow the user to assess how up-to-date the resource is and made any changes themselves if needed. A data.frame with date-wise rows of edits

Usage

```
GTS_2020_changelog
```

Format

An object of class `data.frame` with 1 rows and 2 columns.

`intersect_ranges` *intersect_ranges*

Description

Function to find the maximum intersection between a set of numeric ranges, in this case first and last appearence datums on taxonomic ranges.

Usage

```
intersect_ranges(x, srt = NULL, end = NULL, verbose = TRUE)
```

Arguments

`x`	A numeric data.frame or matrix of ranges. If just two columns are supplied, the first column is assumed to be the srt column
`srt`	If x contains more than two columns, srt is the name of the range base column - the FAD
`end`	If x contains more than two columns, end is the name of the range top column - the LAD
`verbose`	A logical indicating whether the function should report progress to the console

Value

A matrix with three columns, indicating the intersection (FAD and LAD) and the number of ranges that intersection encompasses

Examples

```
# plot an example
df <- cbind(c(1.5, 3, 2.1, 1), c(6, 5, 3.7, 10.1))
plot(1:11, ylim = c(0, 5), col = NA)
segments(x0 = c(1.5, 3, 2.1, 1), y0 = 1:4, x1 = c(6, 5, 3.7, 10.1), y1 = 1:4)
abline(v = 3, col  = "red", lty = 2)
abline(v = 3.7, col  = "red", lty = 2)
# intersect function
intersect_ranges(df)
```

---

pacmacro_ranges *pacmacro_ranges*

Description

Function to apply a modification of Pacman trimming to macrofossil data. The function generates a densified occurrence record using the same methods as `densify` then trim the upper and lower ranges by a user-defined percentage. The full and trimmed ranges are then compared against each other to test if the FAD and the LAD for a taxon form a long tail in its distribution. Multiple tail thresholds can be supplied, but all test to see if the sum of the FAD and LAD which exceeds the trimmed range constitute the threshold proportion of the total range for than taxon, e.g. does the FAD and the LAD outside of the trimmed range comprise a quarter (`tail.flag = 0.25`) of the taxon range?

Usage

```
pacmacro_ranges(
  x,
  rank = "genus",
  srt = "max_ma",
  end = "min_ma",
  step = 1,
  density = 0.1,
  top = 5,
  bottom = 5,
  tail.flag = 0.35,
  method = c("histogram", "kernel")
)
```

Arguments

`x` A stratigraphic occurrence dataset

`rank` The column name in x containing the taxon names for which trimmed ranges will be calculated

`srt` A column name in x denoting the occurrence FADs

`end` A column name in x denoting the occurrence LADs `step` A positive integer specifying the time window size (i.e. the duration represented by each row in the output matrix)

`density` A positive numeric specifying the step size for densifying records. This should ideally be smaller than step

`top` The percentage by which the top of the range will be trimmed

`bottom` The percentage by which the bottom of the range will be trimmed

`tail.flag` a numeric vector of proportions in the range 0 > x > 1 which will be used to test for long tails

`method` The method for quantifying occurrence density. By default both histogram and kernel density will be used

Value

If the user specifies a specific method (e.g. method = "kernel"), the returned value will be a data.frame containing the taxa as row names, the original taxon ranges (FAD, LAD), their ranges as trimmed by the specified value (default FAD95, LAD95), and the tail status (0 = none, 1 = tail) at the user-specified tail proportions. If method is not specified, the result will be a list of 2 data.frames, one for each method

Source

Pacman procedure modified from https://rdrr.io/github/plannapus/CONOP9companion/src/R/pacman.R.

References

Lazarus et al (2012) Paleobiology

Examples

```
# load dataset
data("brachios")
# subsample brachios to make for a short example runtime
set.seed(1)
brachios <- brachios[sample(1:nrow(brachios), 1000),]
# run pacmacro
pacm <- pacmacro_ranges(brachios, tail.flag = c(0.3, 0.35, 0.4),
                        rank = "genus", srt = "max_ma", end = "min_ma")
```

pbdb_fields *geog_lookup*

Description

lookup table called by 'get_pbdb'

Usage

```
pbdb_fields
```

Format

An object of class `list` of length 31.

`pbdb_kingdoms` *pbdb_kingdoms*

Description

lookup table called by 'get_pbdb'

Usage

```
pbdb_kingdoms
```

Format

An object of class `list` of length 3.

`plot_dprofile` *plot_dprofile*

Description

Function to plot density profiles of occurrences through time using the output of @seealso densify.

Usage

```
plot_dprofile(x, taxon, exit = TRUE)
```

Arguments

`x`	The list output of @seealso densify
`taxon`	A character vector of length one, specifying one of the taxon names in x to be plotted
`exit`	Restore base plotting parameters on function exit (default as a requirement for CRAN). Can be set to false to allow other elements to be aded to a plot

Value

NULL, the plotted density profile

Examples

```
# load dataset
data("brachios")
# subsample brachios to make for a short example runtime
set.seed(1)
brachios <- brachios[sample(1:nrow(brachios), 1000),]
# densify ranges
dens <- densify(brachios)
plot_dprofile(dens, "Atrypa")
```

plot_taxa *plot_taxa*

Description

Function to plot the parent or child relationships of an element in a hierarchically organised dataframe. Multiple taxa can be plotted simultaneously

Usage

```
plot_taxa(
  x,
  taxon,
  trank,
  ranks,
  mode = c("parent", "child", "all"),
  step = NULL
)
```

Arguments

x	a dataframe containing hierarchically organised data in columns
taxon	A character vector of element names whose relationships will be plotted (these must be of the same rank)
trank	A character vector of length one corresponding to the column name in x in which taxa is located
ranks	A character vector corresponding to the column names in x, given in hierarchical order
mode	The direction of the relationships to be plotted
step	A positive integer specifinyg the neighbourhood of the relationships to plot. Specifying a number greater than the number of ranks will not cause a failure, and will instead plot all relationships in the direction specified in mode

Value

A plot of the relationships of the specified elements

Examples

```
# load dataset
data("brachios")
# define ranks in dataset
b_ranks <- c("phylum", "class", "order", "family", "genus")
# plot taxon
plot_taxa(brachios, "Atrypa", trank = "genus", ranks = b_ranks, mode = "parent")
```

quantile_coef_density_BMS

quantile_coef_density_BMS

Description

Static rip of the quantile.coef.density function and relevant internals from the BMS package as the package is archived.

Usage

```
quantile_coef_density_BMS(
  x,
  probs = seq(0.25, 0.75, 0.25),
  names = TRUE,
  normalize = TRUE,
  ...
)
```

Arguments

`x`	a object of class pred.density, coef.density, density, or a list of densities
`probs`	numeric vector of probabilities with values in range 0 - 1. Elements very close to the boundaries return Inf or -Inf
`names`	logical; if TRUE, the result has a names attribute, resp. a rownames and colnames attributes. Set to FALSE for speedup with many probs
`normalize`	logical if TRUE then the values in x$y are multiplied with a factor such that their integral is equal to one
`...`	further arguments passed to or from other methods.

Value

If x is of class density (or a list with exactly one element), a vector with quantiles. If x is a list of densities with more than one element (e.g. as resulting from pred.density or coef.density), then the output is a matrix of quantiles, with each matrix row corresponding to the respective density.

Source

static rip from BMS package

resolve_duplicates *resolve_duplicates*

Description

Function for identifying and resolving alternative higher assignments in a hierarchically structured dataframe. Columns are checked from the lowest to the highest rank for elements with multiple higher assignments. These assignments are then assessed topologically to determine if they represent inadvertent use of the same name at a given rank for genuinely different entities, or whether the higher classifications are conflicting. In the case of the former, unique character suffixes are applied to each differently classified case (up to 26 currently supported), effectively splitting up the alternatively classified element. In the case of the latter, the alternative classifications are assessed and are either combined, or the more frequently used or the more complete classification scheme is taken (the more frequent pathway can also be the most complete).

Usage

```
resolve_duplicates(x, ranks = NULL, jump = 4, plot = FALSE, verbose = TRUE)
```

Arguments

x	A dataframe containing hierarchically structured information, for example a table of genus names and their higher taxonomic classifications
ranks	If not NULL, a vector of column names of x, given in rank order. This is useful if x contains columns which are not rank relevant or if columns are not in hierarchical order. If not supplied, the column order in x is used directly and is assumed to be in rank order
jump	The maximum number of levels between the point of divergence and the point of reunion (if present) for a given path, below which the divergence will be taken as conflicting
plot	A logical speciying if the divergent paths should be plotted
verbose	A logical of length one which determines if the function should report the detection and resolution of elements with multiple higher classifications (if any)

Value

The dataframe x, with any alternative higher classifications resolved, giving the classification a strict tree structure

Examples

```
# load dataset
data("brachios")
# define ranks
b_ranks <- c("phylum", "class", "order", "family", "genus")
# run function
res <- resolve_duplicates(brachios, ranks = b_ranks)
```

revise_ranges *revise_ranges*

Description

Function to generate a consensus age for assemblages of fossil data in x, given a table of taxonomic ranges. The need for error-checking is informed by the error codes for the individual fossil occurrences within each collection - if there is no error, then the consensus age is unchanged. If errors are present, then a consensus age for a threshold proportion of taxa is searched for using the overlap of the ranges for those taxa, as given in range table y. Taxa whose occurrences lie outside this consensus age are flagged as potential taxonomic errors. If the threshold consensus partially overlaps with the assemblage age, this overlap is returned to present overzealous alteration of the age - otherwise the complete consensus age is returned. If a consensus age cannot be found, the original assemblage age is returned, and each occurrence in the collection flagged as potential taxonomic errors.

Usage

```
revise_ranges(
  x,
  y,
  assemblage = "collection_no",
  srt = "max_ma",
  end = "min_ma",
  taxon = "genus",
  err = NULL,
  do.flag = FALSE,
  prop = 0.75,
  allow.zero = TRUE,
  verbose = TRUE
)
```

Arguments

`x`	Fossil occurrence data grouped into spatiotemporally distinct assemblages
`y`	A stratigraphic range dataset from which consensus assemblage ages will be derived
`assemblage`	The column name of the assemblage groups in x
`srt`	The column name of stratigraphic bases for each element in both x and y - i.e. x and y must have this same name for that column
`end`	The column name of stratigraphic tops for each element in both x and y - i.e. x and y must have this same name for that column
`taxon`	The column name denoting the taxon names in both x and y - i.e. x and y must have this same name for that column

`err` The column name flagging age errors for occurrences in x. This allows 100$ valid assemblages to be skipped. Age errors can be derived using @seealso flag_ranges. All error codes must be one of: "000" - unchecked, "R1R" - valid, "0R0" - both FAD and LAD exceeded, "00R" - totally older than range, "R00" - totally younger than range, "01R" - FAD exceeded, "1R0" - LAD exceeded. If not supplied, all assemblages will be checked, even if they are already valid a priori.

`do.flag` Rather than supplying error codes, should flag_ranges be called internally to generate error codes for supply to the rest of revise_ranges? As with err, this is useful to prefilter individual occurrences, allowing assemblages contain all valid, all unchecked or a mixture of such error codes to be skipped. This can massively speed up processing time for large datasets.

`prop` A numeric, between 0 and 1, denoting the threshold percentage of taxa in the assemblage for which a consensus age must be found

`allow.zero` A logical determining if, in the case of a collection LAD being equal to the consensus age FAD (i.e. a pointwise overlap), that pointwise age will be taken as the revised age. The resultant collection age will have no uncertainty as a result, which may be unrealistic. The default behaviour is FALSE, in which case pointwise overlaps will be ignored and the revised age taken instead

`verbose` A logical determining if the progress of the redating procedure should be reported

Value

A list of two dataframes, the first recording the results of the consensus redating procedure for each assemblage in x, the second recording any flags (if any) for each occurrence in x

Examples

```
# load datasets
data("brachios")
data("sepkoski")
# subsample brachios to make for a short example runtime
set.seed(1)
brachios <- brachios[sample(1:nrow(brachios), 1000),]
# rename columns in Sepkoski to match brachios
colnames(sepkoski)[4:6] <- c("genus", "max_ma", "min_ma")
# flag and resolve against the Sepkoski Compendium, collection-wise
revrng <- revise_ranges(x = brachios, y = sepkoski, do.flag = TRUE, verbose = TRUE,
                        taxon = "genus", assemblage = "collection_no",
                        srt = "max_ma", end = "min_ma")
# append the revised occurrence ages and error codes to the dataset
brachios$newfad <- revrng$occurrence$FAD
brachios$newlad <- revrng$occurrence$LAD
brachios$errcode <- revrng$occurence$status
```

`sepkoski` *sepkoski*

Description

An example dataset. A port of the Sepkoski Compendium from the chronosphere package, with a few corrections and GTS2020 dating applied

Usage

```
sepkoski
```

Format

An object of class `data.frame` with 35700 rows and 6 columns.

`sep_code` *sep_code*

Description

Lookup table of chronostratigraphic stage abbreviations used in the Sepkoski Compendium, with interval boundaries updated to the GTS2020 standard

Usage

```
data(sep_code)
```

Format

An object of class `data.frame` with 306 rows and 8 columns.

Source

chronosphere (fetch), Sepkoski 2002

spell_check *spell_check*

Description

Function for checking for potential synonyms with alternate spellings. Synonyms are checked for within group using using a Jaro Winkler string distance matrix. Potential synonyms are selected using the jw threshold. These can then be further filtered by the number of shared letters at the beginning and end of the a synonym pair, and by prefixes or suffixes which may give erroneously high similarities.

Usage

```
spell_check(
  x,
  terms = NULL,
  groups = NULL,
  jw = 0.1,
  str = 1,
  str2 = NULL,
  alternative = "jaccard",
  q = 1,
  pref = NULL,
  suff = NULL,
  exclude = NULL,
  verbose = TRUE
)
```

Arguments

x	a dataframe containing a column with terms, and a further column denoting the groups within which terms will be checked against one another. If supplying a dataframe with just these columns, terms should be column 1
terms	a character vector of length 1, specifying the terms column in x. This is required if x contains more than two columns. Alternatively, if x is not provided, terms can be a character vector. If groups are not specified, all elements of terms will be treated as part of the same group
groups	a character vector of length 1, specifying the groups column in x. This is required if x contains more than two columns. Alternatively, if terms is supplied as a character vector, groups can also be supplied in the same way to denote their groups
jw	a numeric greater than 0 and less than 1. This is the distance threshold below which potential synonyms will be considered
str	A positive integer specifying the number of matching characters at the beginning of synonym pairs. By default 1, i.e. the first letters must match

`str2` If not NULL, a positive integer specifying the number of matching characters at the end of synonym pairs

`alternative` A character string of length one corresponding to one of the methods used by @seealso afind. One of "osa", "lv", "dl", "hamming", "lcs", "qgram", "cosine", "running_cosine", "jaccard", or "soundex".

`q` q-gram size. Only used when alternative is "qgram", "cosine" or "Jaccard".

`pref` If not NULL, a character vector of prefixes which may result in erroneously low JW distances. Synonyms will only be considered if both terms share the same prefix

`suff` If not NULL, a character vector of suffices which may result in erroneously low JW distances. Synonyms will only be considered if both terms share the same suffix

`exclude` If not NULL, a character vector of group names which should be skipped - useful for groups which are known to contain potentially similar terms

`verbose` A logical determining if function progress be reported using the pbapply progress bar

Value

a dataframe of synonyms (cols 1 and 2), the group in which they occur, the frequencies of each synonym in the dataset and finally the q-gram difference between the synonyms

Examples

```
# load dataset
data("brachios")
# define suffixes
b_suff <- c("ina", "ella", "etta")
# run function
spl <- spell_check(brachios, terms = "genus", groups = "family", suff = b_suff)
```

`tgraph` *tgraph*

Description

Function to create a tgraph representation of a hierarchically organised dataframe. This is the focal object of the t* functions - the complete set of hierarchical relationships between a set of elements

Usage

```
tgraph(x, ranks = NULL, verbose = TRUE)
```

Arguments

x — A dataframe containing a set of hierarchical relationships. The leftmost column contains the elements which will form the highest rank, followed rightwards by successive ranks

ranks — If not NULL, a vector of column names of x, given in rank order. This is useful if x contains columns which are not rank relevant or if columns are not in hierarchical order. If not supplied, the column order in x is used directly and is assumed to be in rank order

verbose — A logical indicating whether the progress of tgraph construction should be reported to the console

Value

a tgraph object

threshold_peaks *threshold_peaks*

Description

Function to detect if two peaks in a density spectrum can be considered separate based on a user supplied threshold. Creates a sequence of divisions from the troughs immediately preceding any significant peaks, then bins occurrences for a given taxon name by those divisions.

Usage

```
threshold_peaks(
  x,
  y,
  ycols = c("genus", "max_ma", "min_ma"),
  thresh = 15,
  verbose = TRUE
)
```

Arguments

x — A list of significant peaks as returned by `find_peaks`

y — An occurrence dataset with taxon names corresponding to the list names of x

ycols — A character vector denoting, in order, the taxon, FAD and LAD columns in y

thresh — The threshold distance between peaks above which they will be considered distinct - given in Ma

verbose — A logical determining if function progress should be reported

threshold_ranges *threshold_ranges*

Description

Function to detect if two peaks in a density spectrum can be considered separate based on a user supplied threshold. Creates a sequence of divisions from the troughs immediately preceding any significant peaks, then bins occurrences for a given taxon name by those divisions.

Usage

```
threshold_ranges(
  x,
  rank = "genus",
  srt = "max_ma",
  end = "min_ma",
  method = "kernel",
  step = 1,
  density = 0.1,
  use_sd = TRUE,
  win = 5,
  thresh = 5,
  ...,
  report = TRUE,
  verbose = TRUE
)
```

Arguments

`x` An occurrence dataset containing taxon names, maximum ages and minimum ages

`rank` The column name in x containing the taxon names

`srt` A column name in x denoting the occurrence maximum ages

`end` A column name in x denoting the occurrence minumum ages

`method` The method for quantifying occurrence density: one histogram or kernel. Kernel is the recommended default. As called be @seealso densify

`step` A positive integer specifying the time window size for density calculation. As called by @seealso densify

`density` A positive numeric specifying the step size for densifying records. This should ideally be smaller than step. As called by @seealso densify

`use_sd` A logical determining whether to use peaks detected as significant using the mean + standard deviation of its neighbourhood. If FALSE, then the peaks need only be greater than the neighbourhood mean to be significant. Thus, use_sd is more conservative, but less prone to noise. As called by @seealso find_peaks

win	A positive integer specifying the neighborhood window length on either side of a peak durign significance testing (i.e. win 5 will give a total window of 11: -5 indices + peak index + 5 indices). As called by @seealso find_peaks
thresh	The threshold distance between peaks above which they will be considered distinct - given in Ma
...	additional arguments passed to @seealso density
report	A logical determining if the analytical outputs of the function be returned to the user, as well as the revised taxon names, TRUE by default
verbose	A logical determining if function progress should be reported

Value

If report = TRUE (the default), a list of five elements. $data gives the thresholded (and potentially subdivided) taxon names. $matrix is the taxon-wise matrix of occurrence densities. $peaks is a list containing three lists of peaks (all peaks, significant by mean + sd, significant by sd only) for each taxon and a dataframe of peak counts between the three treatments. $comparison

Examples

```
# load dataset
data("brachios")
# subsample brachios to make for a short example runtime
set.seed(1)
brachios <- brachios[sample(1:nrow(brachios), 1000),]
# interpeak thresholding
itp <- threshold_ranges(brachios, win = 8, thresh = 10,
                        rank = "genus", srt = "max_ma", end = "min_ma")
```

update_graph *update_graph*

Description

Function to update the structure of a graph, given a set of modification as returned by assess_duplicates

Usage

```
update_graph(x, del = NULL, add = NULL, changes = NULL)
```

Arguments

x	a tgraph object to modify
del	A vector of element names or numbers to delete
add	An edge sequence of edges to add to the graph
changes	Alternatively, the output of assess_duplicates, containing proposed deletions and additions

Value

An updated tgraph object

Index

∗ **datasets**
- brachios, 6
- geog_lookup, 19
- GTS_2020, 21
- GTS_2020_changelog, 22
- pbdb_fields, 24
- pbdb_kingdoms, 25
- sep_code, 31
- sepkoski, 31

add_itp, 2
add_kingdoms, 3
age_ranges, 4
assess_duplicates, 5

brachios, 6

check_taxonomy, 7
chrono_scale, 10
clean_name, 11

densify, 12
discrete_ranks, 14

find_duplicates, 15
find_peaks, 15
flag_ranges, 16
format_check, 18

geog_lookup, 19
get_pbdb, 19
GTS_2020, 21
GTS_2020_changelog, 22

intersect_ranges, 22

pacmacro_ranges, 23
pbdb_fields, 24
pbdb_kingdoms, 25
plot_dprofile, 25
plot_taxa, 26

quantile_coef_density_BMS, 27

resolve_duplicates, 28
revise_ranges, 29

sep_code, 31
sepkoski, 31
spell_check, 32

tgraph, 33
threshold_peaks, 34
threshold_ranges, 35

update_graph, 36