Package ‘datelife’

June 20, 2023

Title Scientific Data on Time of Lineage Divergence for Your Taxa

Version 0.6.8

Maintainer Luna L. Sanchez Reyes <sanchez.reyes.luna@gmail.com>

Description Methods and workflows to get chronograms (i.e., phylogenetic trees with branch lengths proportional to time), using open, peer-reviewed, state-of-the-art scientific data on time of lineage divergence.

This package constitutes the main underlying code of the DateLife web service at <https://www.datelife.org>. To obtain a single summary chronogram from a group of relevant chronograms, we implement the Super Distance Matrix (SDM) method described in Criscuolo et al. (2006) <doi:10.1080/10635150600969872>.

To find the grove of chronograms with a sufficiently overlapping set of taxa for summarizing, we implement theorem 1.1. from Ané et al. (2009) <doi:10.1007/s00026-009-0017-x>.

A given phylogenetic tree can be dated using time of lineage divergence data as secondary calibrations (with caution, see Schenk (2016) <doi:10.1371/journal.pone.0148228>).


Depends R (>= 3.6.0)

biocViews Software

Imports ape, abind, bold, phangorn, phytools, ips, cluster, compare, geiger, stats, stringr, rotl, paleotree, knitr, phylogenies, phylobase, taxize, treebase, utils, httr, plyr, phylocomr, BiocManager, data.table, curl

Suggests testthat, knit, rmarkdown, usethis, devtools, covr, msa, Biostrings

LazyDataCompression xz

SystemRequirements PATHd8
URL: https://github.com/phylotastic/datelife,
http://phylotastic.org/datelife/
License GPL (>= 2)
LazyData true
RoxygenNote 7.2.3
Encoding UTF-8
VignetteBuilder knitr
NeedsCompilation no
Author Brian O'Meara [aut],
Jonathan Eastman [aut],
Tracy Heath [aut],
April Wright [aut],
Klaus Schliep [aut],
Scott Chamberlain [aut],
Peter Midford [aut],
Luke Harmon [aut],
Joseph Brown [aut],
Matt Pennell [aut],
Mike Alfaro [aut],
Luna L. Sanchez Reyes [aut, cre],
Emily Jane McTavish [ctb]
Repository CRAN
Date/Publication 2023-06-19 23:20:02 UTC

R topics documented:

.get_ott_lineage                         5  
birds_and_cats                           6  
birds_and_cats                           6  
birds_and_cats                           6  
build_grove_list                          6  
birds_and_cats                           6  
build_grove_matrix                       7  
build_grove_matrix                       7  
check_conflicting_calibrations           8  
check_ott_input                          8  
choose_cluster                           9  
classification_paths_from_taxonomy       10  
clean_ott_chronogram                     13  
clean_taxon_info_children                14  
clean_trns                               15  
cluster_patristic.matrix                 16  
cluster_patristic.matrix                 16  
congruify_and_check                      16  
congruify_and_mrca_multiPhylo            17  
congruify_and_mrca_phylo                 18  
congruify_and_mrca_phylo                 18  
congruify_and_mrca_phylo                 18  
contributor_cache                        18  
dateline_authors_tabulate                19  
dateline_authors_tabulate                19  
dateline_result_median                   20  
dateline_result_median                   20  
dateline_result_median                   20  
dateline_result_median_matrix            21  
dateline_result_median_matrix            21  

topics documented:

- datelife_result_MRCA .................................................. 21
- datelife_result_sdm_matrix .......................................... 22
- datelife_result_sdm_phylo ........................................... 22
- datelife_result_study_index .......................................... 23
- datelife_result_variance_matrix .................................... 24
- datelife_search ........................................................... 25
- datelife_use ............................................................. 28
- datelife_use_datelifequery .......................................... 29
- date_with_pbdb .......................................................... 30
- extract_calibrations_dateliferesult ............................... 31
- extract_calibrations_phylo .......................................... 32
- extract_ott_ids .......................................................... 32
- felid_gdr_phylo_all .................................................... 33
- felid_sdm ............................................................... 34
- filter_for_grove ........................................................ 34
- force_ultrametric ....................................................... 35
- get_all_calibrations .................................................. 36
- get_all_descendant_species ......................................... 36
- get_best_grove ........................................................ 37
- get_biggest_multiphylo ............................................... 38
- get_bold_data .......................................................... 38
- get_calibrations_datelifequery .................................... 39
- get_calibrations_vector .............................................. 40
- get_dated_otol_induced_subtree .................................... 41
- get_datelife_result ................................................... 42
- get_datelife_result_datelifequery .................................. 43
- get_fossil_range ...................................................... 44
- get_goodmatrices ..................................................... 45
- get mrbayes_node_constraints ...................................... 45
- get_opentree_chronograms .......................................... 47
- get_opentree_species ................................................ 47
- get_otol_synthetic_tree ............................................. 48
- get_ott_children ...................................................... 49
- get_ott_clade .......................................................... 50
- get_ott_lineage ....................................................... 51
- get_subset_array_dispatch ......................................... 51
- get_tnrs_names ........................................................ 52
- get_valid_children .................................................... 54
- input_process .......................................................... 55
- is_datelife_query ...................................................... 55
- is_datelife_result_empty ........................................... 56
- is_good_chronogram .................................................. 57
- is_n_overlap ........................................................... 57
- make_all_associations ............................................... 58
- make_bladj_tree ....................................................... 58
- make_bold_otol_tree ................................................ 59
- make_contributor_cache ............................................. 60
topics documented:

- make_datelife_query ................................................. 60
- make_datelife_query2 .............................................. 61
- make mrbayes_runfile ............................................. 63
- make mrbayes_tree .................................................. 64
- make otol associations ............................................ 65
- make overlap_table .................................................. 66
- make sdm ............................................................. 66
- make_treebase_assocations ....................................... 67
- map nodes_ott ....................................................... 68
- match all calibrations .............................................. 69
- matrices to table ................................................. 69
- matrix to table ..................................................... 70
- message multiphylo ................................................ 70
- missing_taxa_check ............................................... 71
- mrca_calibrations .................................................. 71
- opentree_chronograms ............................................. 72
- patristic_matrix_array_congruify ............................... 73
- patristic_matrix_array_phylo_congruify ......................... 74
- patristic_matrix_array_split ..................................... 74
- patristic_matrix_array_subset ................................... 75
- patristic_matrix_array_subset_both ............................. 75
- patristic_matrix_list_to_array .................................. 76
- patristic_matrix_MRCA ............................................. 77
- patristic_matrix_name_order_test ................................. 77
- patristic_matrix_name_reorder ................................... 78
- patristic_matrix_pad ............................................... 78
- patristic_matrix_taxa_all_matching ............................. 79
- patristic_matrix_to_newick ....................................... 79
- patristic_matrix_to_phylo ........................................ 80
- patristic_matrix_unpad ............................................. 80
- phylo_check ........................................................ 81
- phylo_congruify .................................................... 82
- phylo_generate_uncertainty ...................................... 82
- phylo_get_node_numbers .......................................... 84
- phylo_get_subset_array .......................................... 85
- phylo_get_subset_array_congruify ............................... 85
- phylo_has_brlen .................................................... 86
- phylo_prune_missing_taxa ........................................ 86
- phylo_subset_both ................................................ 87
- phylo_tiplabel_space_to_underscore ............................. 88
- phylo_tiplabel_underscore_to_space ............................. 88
- phylo_to_patristic_matrix ........................................ 89
- pick_grove .......................................................... 89
- plant_bold_otol_tree .............................................. 90
- problems ............................................................ 91
- recover_mrcaott .................................................... 91
- relevant_curators_tabulate ...................................... 92
.get_ott_lineage

Get the lineage of a set of taxa.

.get_ott_lineage uses rotl::taxonomy_taxon_info() with include_lineage = TRUE.

Description

Get the lineage of a set of taxa. .get_ott_lineage uses rotl::taxonomy_taxon_info() with include_lineage = TRUE.
Usage
.build_grove_list(datelife_result, n = 2)

Arguments

datelife_result
        A multiPhyllo object with trees resulting from a datelife search of some
        birds and cats species

Details
        Generated with: taxa <- c("Rhea americana", "Pterocnemia pennata", "Struthio camelus", "Gallus",
        "Felis") birds_and_cats <- datelife_search(input = taxa, summary_format = "phylo_all", get_spp_from_taxon
        = TRUE) usethis::use_data(birds_and_cats)

Description
        This function implements theorem 1.1 of Ané et al. (2009) doi:10.1007/s000260090017x to find a
        grove for a given group of chronograms.

Usage
        build_grove_list(datelife_result, n = 2)
Arguments

- `datelife_result`:
  A `datelifeResult` object.

- `n`:
  The degree of taxon name overlap among input chronograms. Defaults to \( n = 2 \), i.e., at least two overlapping taxon names.

Value

A list of vectors; each list element is a grove.

References

check_conflicting_calibrations

*Check for conflicting calibrations.*

**Description**

check_conflicting_calibrations checks if calibrations are younger or older relative to descendants and ancestors, respectively.

**Usage**

`check_conflicting_calibrations(phy, calibration_distribution)`

**Arguments**

- `phy` A phylo object.
- `calibration_distribution` A list of node age distributions, named with `phy`'s node numbers.

**Details**

It removes conflicting calibrations if needed, but BLADJ works as long as it has an age for the root.

check_ott_input

*Check input for usage in other datelife functions*

**Description**

check_ott_input is currently used in functions `get_ott_clade()`, `get_ott_children()`, and `get_otol_synthetic_tree()`.

**Usage**

`check_ott_input(input = NULL, ott_ids = NULL, ...)`

**Arguments**

- `input` Optional. A character vector of names or a `datelifeQuery` object.
- `ott_ids` If not NULL, it takes this argument and ignores input. A numeric vector of `ott` ids obtained with `rotl::taxonomy_taxon_info()` or `rotl::tnrs_match_names()` or `tnrs_match()`.
- `...` Arguments passed on to `make_datelife_query`
- `use_tnrs` Whether to use Open Tree of Life’s Taxonomic Name Resolution Service (TNRS) to process input taxon names. Default to TRUE, it corrects misspellings and taxonomic name variations with `tnrs_match()`, a wrapper of `rotl::tnrs_match_names()`.
choose_cluster

get_spp_from_taxon Whether to search ages for all species belonging to a given taxon or not. Default to FALSE. If TRUE, it must have same length as input. If input is a newick string with some clades it will be converted to a phylo object, and the order of get_spp_from_taxon will match phy$tip.label.

reference_taxonomy A character vector specifying the reference taxonomy to use for TNRS. Options are "ott", "ncbi", "gbif" or "irmng". The function defaults to "ott".

Details

By default, it uses the ott_id argument if it is not NULL.

Value

A named numeric vector of valid Open Tree Taxonomy (OTT) ids.

choose_cluster

Choose an ultrametric phylo object from cluster_patristicmatrix() obtained with a particular clustering method, or the next best tree. If there are no ultrametric trees, it does not force them to be ultrametric.

Description

Choose an ultrametric phylo object from cluster_patristicmatrix() obtained with a particular clustering method, or the next best tree. If there are no ultrametric trees, it does not force them to be ultrametric.

Usage

choose_cluster(phycluster, clustering_method = "nj")

Arguments

phycluster An output from cluster_patristicmatrix()
clustering_method A character vector indicating the method to construct the tree. Options are:

nj Neighbor-Joining method applied with ape::nj().
upgma Unweighted Pair Group Method with Arithmetic Mean method applied with phangorn::upgma().
bionj An improved version of the Neighbor-Joining method applied with ape::bionj().
triangle Triangles method applied with ape::triangMtd().
mvr Minimum Variance Reduction method applied with ape::mvr().

Value

A phylo object or NA.
classification_paths_from_taxonomy

*Gets classification paths for a vector of taxa*

**Description**

This uses the taxize package’s wrapper of the Global Names Resolver to get taxonomic paths for the vector of taxa you pass in. Sources is a vector of source labels in order (though it works best if everything uses the same taxonomy, so we recommend doing just one source). You can see options by doing taxize::gnr_datasources(). Our default is Catalogue of Life

**Usage**

```r
classification_paths_from_taxonomy(taxa, sources = "Catalogue of Life")
```

**Arguments**

- **taxa**: Vector of taxon names
- **sources**: Vector of names of preferred sources; see taxize::gnr_datasources(). Currently supports 100 taxonomic resources, see details.

**Details**

Taxonomies supported by taxize::gnr_datasources()

1. Catalogue of Life
2. Wikispecies
3. ITIS
4. NCBI
5. Index Fungorum
6. GRIN Taxonomy for Plants
7. Union 4
8. The Interim Register of Marine and Nonmarine Genera
9. World Register of Marine Species
10. Freebase
11. GBIF Backbone Taxonomy
12. EOL
13. Passiflora vernacular names
14. Inventory of Fish Species in the Wami River Basin
15. Pheasant Diversity and Conservation in the Mt. Gaoligongshan Region
16. Finding Species
17. Birds of Lindi Forests Plantation
18. Nemertea
19. Kihansi Gorge Amphibian Species Checklist
20. Mushroom Observer
21. TaxonConcept
22. Amphibia and Reptilia of Yunnan
23. Common names of Chilean Plants
24. Invasive Species of Belgium
25. ZooKeys
26. COA Wildlife Conservation List
27. AskNature
29. Native Orchids from Gaoligongshan Mountains, China
30. Illinois Wildflowers
31. Coleorrhyncha Species File
32. /home/dimus/files/dwca/zoological names.zip
33. Peces de la zona hidrogeográfica de la Amazonia, Colombia (Spreadsheet)
34. Eastern Mediterranean Syllidae
35. Gaoligong Shan Medicinal Plants Checklist
36. birds_of_tanzania
37. AmphibiaWeb
38. tanzania_plant_sepecimens
39. Papahanaumokuakea Marine National Monument
40. Taiwanese IUCN species list
41. BioPedia
42. AnAge
43. Embioptera Species File
44. Global Invasive Species Database
45. Sendoya S., Fernández F. AAT de hormigas (Hymenoptera: Formicidae) del Neotrópico 1.0 2004 (Spreadsheet)
46. Flora of Gaoligong Mountains
47. ARKive
48. True Fruit Flies (Diptera, Tephritidae) of the Afrotropical Region
49. 3i - Typhlocybinae Database
50. CATE Sphingidae
51. ZooBank
52. Diatoms
53. AntWeb
<table>
<thead>
<tr>
<th></th>
<th>Classification Paths From Taxonomy</th>
</tr>
</thead>
<tbody>
<tr>
<td>54.</td>
<td>Endemic species in Taiwan</td>
</tr>
<tr>
<td>55.</td>
<td>Dermaptera Species File</td>
</tr>
<tr>
<td>56.</td>
<td>Mantodea Species File</td>
</tr>
<tr>
<td>57.</td>
<td>Birds of the World: Recommended English Names</td>
</tr>
<tr>
<td>58.</td>
<td>New Zealand Animalia</td>
</tr>
<tr>
<td>59.</td>
<td>Blattodea Species File</td>
</tr>
<tr>
<td>60.</td>
<td>Plecoptera Species File</td>
</tr>
<tr>
<td>61.</td>
<td><code>/home/dimus/files/dwca/clemens.zip</code></td>
</tr>
<tr>
<td>62.</td>
<td>Coreoidea Species File</td>
</tr>
<tr>
<td>63.</td>
<td>Freshwater Animal Diversity Assessment - Normalized export</td>
</tr>
<tr>
<td>64.</td>
<td>Catalogue of Vascular Plant Species of Central and Northeastern Brazil</td>
</tr>
<tr>
<td>65.</td>
<td>Wikipedia in EOL</td>
</tr>
<tr>
<td>66.</td>
<td>Database of Vascular Plants of Canada (VASCAN)</td>
</tr>
<tr>
<td>67.</td>
<td>Phasmida Species File</td>
</tr>
<tr>
<td>68.</td>
<td>OBIS</td>
</tr>
<tr>
<td>69.</td>
<td>USDA NRCS PLANTS Database</td>
</tr>
<tr>
<td>70.</td>
<td>Catalog of Fishes</td>
</tr>
<tr>
<td>71.</td>
<td>Aphid Species File</td>
</tr>
<tr>
<td>72.</td>
<td>The National Checklist of Taiwan</td>
</tr>
<tr>
<td>73.</td>
<td>Psocodea Species File</td>
</tr>
<tr>
<td>74.</td>
<td>FishBase</td>
</tr>
<tr>
<td>75.</td>
<td>3i - Typhlocybinae Database</td>
</tr>
<tr>
<td>76.</td>
<td>Belgian Species List</td>
</tr>
<tr>
<td>77.</td>
<td>EUNIS</td>
</tr>
<tr>
<td>78.</td>
<td>CU*STAR</td>
</tr>
<tr>
<td>79.</td>
<td>Orthoptera Species File</td>
</tr>
<tr>
<td>80.</td>
<td>Bishop Museum</td>
</tr>
<tr>
<td>81.</td>
<td>IUCN Red List of Threatened Species</td>
</tr>
<tr>
<td>82.</td>
<td>BioLib.cz</td>
</tr>
<tr>
<td>83.</td>
<td>Tropicos - Missouri Botanical Garden</td>
</tr>
<tr>
<td>84.</td>
<td>nlbif</td>
</tr>
<tr>
<td>85.</td>
<td>The International Plant Names Index</td>
</tr>
<tr>
<td>86.</td>
<td>Index to Organism Names</td>
</tr>
<tr>
<td>87.</td>
<td>uBio NameBank</td>
</tr>
<tr>
<td>88.</td>
<td>Arctos</td>
</tr>
<tr>
<td>90.</td>
<td>The Paleobiology Database</td>
</tr>
</tbody>
</table>
91. The Reptile Database
92. The Mammal Species of The World
93. BirdLife International
94. Checklist da Flora de Portugal (Continental, Açores e Madeira)
95. FishBase Cache
96. Silva
97. Open Tree of Life Reference Taxonomy
98. iNaturalist
99. The Interim Register of Marine and Nonmarine Genera
100. Gymno

Value

A list with resolved taxa (a tibble, from taxize::gnr_resolve) and a vector of taxa not resolved

clean_ott_chronogram

Clean up some issues with Open Tree of Life chronograms. For now it 1) checks unmapped taxa and maps them with tnrs_match.phylo, 2) roots the chronogram if unrooted

Description

Clean up some issues with Open Tree of Life chronograms. For now it 1) checks unmapped taxa and maps them with tnrs_match.phylo, 2) roots the chronogram if unrooted

Usage

clean_ott_chronogram(phy)

Arguments

phy

A phylo object.

Details

There is no limit to the number of names that can be queried and matched. The output will preserve all elements from original input phylo object and will add

```r
phy$mapped
```

A character vector indicating the state of mapping of phy$tip.labels:

- **original**: Tnrs matching was not attempted. Original labeling is preserved.
- **ott**: Matching was manually made by a curator in Open Tree of Life.
- **tnrs**: Tnrs matching was attempted and successful with no approximate matching. Original label is replaced by the matched name.
**approximated**  Tnrs matching was attempted and successful but with approximate matching. Original labeling is preserved.

**unmatched**  Tnrs matching was attempted and unsuccessful. Original labeling is preserved.

**phy$original.tip.label**  A character vector preserving all original labels.

**phy$ott_ids**  A numeric vector with ott id numbers of matched tips. Unmatched and original tips will be NaN.

if tips are duplicated, tnr will only be run once (avoiding increases in function running time) but the result will be applied to all duplicated tip labels

**Value**

An object of class data frame or phylo, with the added class match_names.

NULL

NULL

---

**clean_taxon_info_children**

Identify, extract and clean taxonomic children names from a taxonomy_taxon_info() output.

**Description**

clean_taxon_info_children eliminates all taxa that will give problems when trying to retrieve an induced subtree from Open Tree of Life.

**Usage**

```r
clean_taxon_info_children(
  taxon_info,
  invalid = c("barren", "extinct", "uncultured", "major_rank_conflict", "incertae_sedis", 
              "unplaced", "conflict", "environmental", "not.otu", "hidden", "hybrid")
)
```

**Arguments**

- **taxon_info**: An output of `rotl::taxonomy_taxon_info()`.
- **invalid**: A character vector of "flags", i.e., characteristics that are used by Open Tree of Life Taxonomy to detect invalid taxon names.

**Value**

A list with valid children unique OTT names, OTT ids and taxonomic ranks.
clean_tnrs

Eliminates unmatched (NAs) and invalid taxa from a `rotl::tnrs_match_names()` or `tnrs_match()` output. Useful to get OTT ids to retrieve an induced synthetic Open Tree of Life. Needed because using `include_suppressed = FALSE` in `rotl::tnrs_match_names()` does not drop all invalid taxa.

**Description**

Eliminates unmatched (NAs) and invalid taxa from a `rotl::tnrs_match_names()` or `tnrs_match()` output. Useful to get OTT ids to retrieve an induced synthetic Open Tree of Life. Needed because using `include_suppressed = FALSE` in `rotl::tnrs_match_names()` does not drop all invalid taxa.

**Usage**

```r
clean_tnrs(
  tnrs,
  invalid = c("barren", "extinct", "uncultured", "major_rank_conflict", "incertae",
             "unplaced", "conflict", "environmental", "not_otu"),
  remove_nonmatches = FALSE
)
```

**Arguments**

- `tnrs`  
  A data frame, usually an output from datelife::tnrs_match or rotl::tnrs_match_names functions, but see details.
- `invalid`  
  A character string with flags to be removed from final object.
- `remove_nonmatches`  
  Boolean, whether to remove unsuccessfully matched names or not.

**Details**

Input can be any data frame or named list that relates taxa stored in an element named "unique" to a validity category stored in "flags".

**Value**

A data frame or named list (depending on the input) with valid taxa only.
**cluster_patristicmatrix**

*Cluster a patristic matrix into a tree with various methods.*

**Description**

Cluster a patristic matrix into a tree with various methods.

**Usage**

```
cluster_patristicmatrix(patristic_matrix, variance_matrix = NULL)
```

**Arguments**

- `patristic_matrix` A patristic matrix
- `variance_matrix` A variance matrix from a datelifeResult object, usually an output from `datelife_result_variance_matrix()`. Only used if `clustering_method = "mvr"`.

**Details**

If clustering method fails, NA is returned.

**Value**

A list of trees obtained with clustering methods detailed in `patristic_matrix_to_phylo()`.

---

**congruify_and_check**

*Congruify and Check.*

**Description**

Congruify and Check.

**Usage**

```
congruify_and_check(
    reference,
    target,
    taxonomy = NULL,
    tol = 0.01,
    option = 2,
    scale = "pathd8",
    attempt_fix = TRUE
)
```
congruify_and_mrca_multiPhylo

Arguments

- **reference**: an ultrametric tree used to time-scale the target
- **target**: a phylogram that is sought to be ultrametricized based on the reference phylogeny
- **taxonomy**: a linkage table between tips of the phylogeny and clades represented in the tree; rownames of `taxonomy` should be tips found in the phylogeny
- **tol**: branching time in reference above which secondary constraints will be applied to target
- **option**: an integer (1 or 2; see details).
- **scale**: NA, "PATHd8" or "treePL" (if PATHd8 or "treePL" are available in the R PATH)
- **attempt_fix**: Default to TRUE. If congruification results in NA branch lengths, it will attempt to fix them.

Description

`congruify_and_mrca_multiPhylo` congruifies a target tree against all source chronograms in a multiPhylo object, and gets nodes of target tree that correspond to the most recent common ancestor (mrca) of taxon pairs in the congruified calibrations. It calls `congruify_and_mrca_phylo()` and `phytools::findMRCA()` to get mrca nodes.

Usage

`congruify_and_mrca_multiPhylo(phy, source_chronograms)`

Arguments

- **phy**: A phylo object.
- **source_chronograms**: A multiPhylo object, output of `datelife_search()`.

Value

a data.frame of node ages from `source_chronograms` and corresponding mrca nodes in target tree `phy`. `attributes(return)$phy` stores the congruified and mrca matched phylogeny.
congruify_and_mrca_phylo

Congruify nodes of a tree topology to nodes from a source chronogram, and find the mrca nodes

Description

congruify_and_mrca congruifies a target tree against a single source chronogram, and gets nodes of target tree that correspond to the most recent common ancestor (mrca) of taxon pairs from the congruified calibrations. It uses phytools::findMRCA() to get mrca nodes.

Usage

congruify_and_mrca_phylo(phy, source_chronogram, reference)

Arguments

phy A phylo object.
source_chronogram A phylo object, output of datelife_search().
reference A character string indicating the study reference that the source_chronogram comes from.

Value

a data.frame of node ages from source_chronograms and corresponding mrca nodes in target tree phy.

contributor_cache Information on contributors, authors, study ids and clades from studies with chronograms in Open Tree of Life (Open Tree)

Description

Information on contributors, authors, study ids and clades from studies with chronograms in Open Tree of Life (Open Tree)

Usage

contributor_cache
datelife_authors_tabulate

Format

A list of five data sets.

- **author.pretty**  A character vector with the author names from studies with chronograms that are in Open Tree.
- **author.results**  A dataframe with three variables: authors, study ids and clades.
- **curator.pretty**  A character vector with the names of curators of chronograms that are in Open Tree.
- **curator.results**  A data.frame with three variables: curators, study ids and clades.
- **missed_doi**  A character vector with study ids whose "doi" could not be retrieved.

Details

Generated with `make_contributor_cache()`.

Source

http://opentreeoflife.org

---

**datelife_authors_tabulate**

*Return the relevant authors for a set of studies.*

Description

Return the relevant authors for a set of studies.

Usage

datelife_authors_tabulate(results.index, cache = "opentree_chronograms")

Arguments

- **results.index**  A vector from `datelife_result_study_index()` with the indices of the relevant studies.
- **cache**  The cached chronogram database.

Value

A vector with counts of each author, with names equal to author names.
**datelife_result_median**

Get a median summary chronogram from a datelifeResult object.

### Description

Get a median summary chronogram from a datelifeResult object.

### Usage

```r
datelife_result_median(datelife_result, ...)
```

### Arguments

- `datelife_result`  
  A datelifeResult object, usually an output of `get_datelife_result()`.
- `...`  
  Arguments passed on to `summary_matrix_to_phylo`
- `summ_matrix`  
  Any summary patristic distance matrix, such as the ones obtained with `datelife_result_sdm_matrix()` or `datelife_result_median_matrix()`.
- `total_distance`  
  Whether the input `summ_matrix` stores total age distance (from tip to tip) or distance from node to tip. Default to TRUE, divides the matrix in half, if FALSE it will take it as is.
- `use`  
  A character vector indicating what type of age to use for summary tree. One of the following:
  - "mean" It will use the `mean()` of the node ages in `summ_matrix`.
  - "median" It uses the `stats::median()` age of node ages in `summ_matrix`.
  - "min" It will use the `min()` age from node ages in `summ_matrix`.
  - "max" Choose this if you wanna be conservative; it will use the `max()` age from node ages in `summ_matrix`.
  - "midpoint" It will use the mean of minimum age and maximum age.
- `target_tree`  
  A phylo object. Use this in case you want a specific backbone for the output tree.
- `datelife_query`  
  A datelifeQuery object, usually an output of `make_datelife_query()`.

### Value

A phylo object.
datelife_result_median_matrix

Compute a median matrix of a datelifeResult object.

Description
Compute a median matrix of a datelifeResult object.

Usage
datelife_result_median_matrix(datelife_result)

Arguments
datelife_result
A datelifeResult object, usually an output of get_datelife_result().

Value
A patristic distance summary matrix from a datelifeResult object.

datelife_result_MRCA

Get a numeric vector of MRCAs from a datelifeResult object. Used in summarize_datelife_result().

Description
Get a numeric vector of MRCAs from a datelifeResult object. Used in summarize_datelife_result().

Usage
datelife_result_MRCA(datelife_result, na.rm = TRUE)

Arguments
datelife_result
A datelifeResult object, usually an output of get_datelife_result().

na.rm
If TRUE, it drops rows containing NAs from the datelifeResult patristic matrix; if FALSE, it returns NA where there are missing entries.

Value
A named numeric vector of MRCA ages for each element given in datelife_result.
datelife_result_sdm_matrix

*Go from a datelifeResult object to a Super Distance Matrix (SDM) using weighting = "flat"*

Description

Go from a datelifeResult object to a Super Distance Matrix (SDM) using weighting = "flat"

Usage

datelife_result_sdm_matrix(datelife_result)

Arguments

datelife_result

A datelifeResult object, usually an output of get_datelife_result().

Value

A numeric matrix.

datelife_result_sdm_phylo

*Reconstruct a supertree from a datelifeResult object using the Super Distance Matrix (SDM) method.*

Description

Reconstruct a supertree from a datelifeResult object using the Super Distance Matrix (SDM) method.

Usage

datelife_result_sdm_phylo(datelife_result, weighting = "flat", ...)

Arguments

datelife_result

A datelifeResult object, usually an output of get_datelife_result().

weighting

A character vector indicating how much weight to give to each tree in input during the SDM analysis. Options are:

- weighting = "flat" All trees have equal weighting.
- weighting = "taxa" Weight is proportional to number of taxa.
- weighting = "inverse" Weight is proportional to 1 / number of taxa.
Defaults to weighting = "flat".

Arguments passed on to `summary_matrix_to_phylo`

`summ_matrix` Any summary patristic distance matrix, such as the ones obtained with `datelife_result_sdm_matrix()` or `datelife_result_median_matrix()`.

`total_distance` Whether the input `summ_matrix` stores total age distance (from tip to tip) or distance from node to tip. Default to TRUE, divides the matrix in half, if FALSE it will take it as is.

`use` A character vector indicating what type of age to use for summary tree. One of the following:
- "mean" It will use the `mean()` of the node ages in `summ_matrix`.
- "median" It uses the `stats::median()` age of node ages in `summ_matrix`.
- "min" It will use the `min()` age from node ages in `summ_matrix`.
- "max" Choose this if you wanna be conservative; it will use the `max()` age from node ages in `summ_matrix`.
- "midpoint" It will use the mean of minimum age and maximum age.

`target_tree` A `phylo` object. Use this in case you want a specific backbone for the output tree.

`datelife_query` A `datelifeQuery` object, usually an output of `make_datelife_query()`.

Details

Chronograms given as input in `datelife_result` are summarized with the Super Distance Matrix (SDM) method described in Criscuolo et al. (2006) doi:10.1080/10635150600969872, implemented with the function `ape::SDM()`. The resulting summary SDM is clustered with `summary_matrix_to_phylo()`.

Value

A supertree with branch lengths proportional to time, obtained by summarizing individual chronograms given as input in `datelife_result`. It is returned as an object of class `datelifeSDM`, which is a `phylo` object with an additional `$data` element storing the input chronograms as a `datelifeResult` object, and a `$citation` element containing citations of studies from input chronograms.

References

Usage

datelife_result_study_index(datelife_result, cache = "opentree_chronograms")

Arguments

datelife_result
  A datelifeResult object, usually an output of get_datelife_result().

Value

A vector of indices of studies that have relevant information.

datelife_result_variance_matrix

Compute a variance matrix of a datelifeResult object.

Description

Compute a variance matrix of a datelifeResult object.

Usage

datelife_result_variance_matrix(datelife_result)

Arguments

datelife_result
  A datelifeResult object, usually an output of get_datelife_result().

Value

A variance matrix from a datelifeResult object.
datelife_search

Description

datelife_search is the core DateLife function to find and get all openly available, peer-reviewed scientific information on time of lineage divergence for a set of input taxon names given as a character vector, a newick character string, a phylo or multiPhylo object or as a an already processed datelifeQuery object obtained with make_datelife_query().

Usage

datelife_search(
  input = c("Rhea americana", "Pterocnemia pennata", "Struthio camelus"),
  use_tnrs = FALSE,
  get_spp_from_taxon = FALSE,
  partial = TRUE,
  cache = "opentree_chronograms",
  summary_format = "phylo_all",
  na_rm = FALSE,
  summary_print = c("citations", "taxa"),
  taxon_summary = c("none", "summary", "matrix"),
  criterion = "taxa"
)

Arguments

input

One of the following:

A **character vector** With taxon names as a single comma separated starting or concatenated with `c()`.

A **phylogenetic tree with taxon names as tip labels** As a phylo or multiPhylo object, OR as a newick character string.

A **datelifeQuery object** An output from make_datelife_query()

use_tnrs

Whether to use Open Tree of Life’s Taxonomic Name Resolution Service (TNRS) to process input taxon names. Default to TRUE, it corrects misspellings and taxonomic name variations with `tnrs_match()`, a wrapper of `rotl::tnrs_match_names()`.

get_spp_from_taxon

Whether to search ages for all species belonging to a given taxon or not. Default to FALSE. If TRUE, it must have same length as input. If input is a newick string with some clades it will be converted to a phylo object, and the order of get_spp_from_taxon will match phy$tip.label.

partial

Whether to return or exclude partially matching source chronograms, i.e, those that match some and not all of taxa given in datelife_query. Options are TRUE or FALSE. Defaults to TRUE: return all matching source chronograms.
cache  A character vector of length one, with the name of the data object to cache. Default to "opentree_chronograms", a data object storing Open Tree of Life’s database chronograms and other associated information.

summary_format  A character vector of length one, indicating the output format for results of the DateLife search. Available output formats are:

"citations"  A character vector of references where chronograms with some or all of the target taxa are published (source chronograms).

"mrca"  A named numeric vector of most recent common ancestor (mrca) ages of target taxa defined in input, obtained from the source chronograms. Names of mrca vector are equal to citations.

"newick_all"  A named character vector of newick strings corresponding to target chronograms derived from source chronograms. Names of newick_all vector are equal to citations.

"newick_sdm"  Only if multiple source chronograms are available. A character vector with a single newick string corresponding to a target chronogram obtained with SDM supertree method (Criscuolo et al. 2006).

"newick_median"  Only if multiple source chronograms are available. A character vector with a single newick string corresponding to a target chronogram from the median of all source chronograms.

"phylo_sdm"  Only if multiple source chronograms are available. A phylo object with a single target chronogram obtained with SDM supertree method (Criscuolo et al. 2006).

"phylo_median"  Only if multiple source chronograms are available. A phylo object with a single target chronogram obtained from source chronograms with median method.

"phylo_all"  A named list of phylo objects corresponding to each target chronogram obtained from available source chronograms. Names of phylo_all list correspond to citations.

"phylo_biggest"  The chronogram with the most taxa. In the case of a tie, the chronogram with clade age closest to the median age of the equally large trees is returned.

"html"  A character vector with an html string that can be saved and then opened in any web browser. It contains a 4 column table with data on target taxa: mrca, number of taxa, citations of source chronogram and newick target chronogram.

"data_frame"  A 4 column data.frame with data on target taxa: mrca, number of taxa, citations of source chronograms and newick string.

na_rm  If TRUE, it drops rows containing NAs from the datelifeResult patristic matrix; if FALSE, it returns NA where there are missing entries.

summary_print  A character vector specifying the type of summary information to be printed to screen. Options are:

"citations"  Prints references of chronograms where target taxa are found.

"taxa"  Prints a summary of the number of chronograms where each target taxon is found.

"none"  Nothing is printed to screen.
datelife_search

Defaults to c("citations", "taxa"), which displays both.

taxon_summary

A character vector specifying if data on target taxa missing in source chronograms should be added to the output as a "summary" or as a presence/absence "matrix". Default to "none", no information on taxon_summary added to the output.

criterion

Defaults to criterion = "taxa". Used for chronogram summarizing, i.e., obtaining a single summary chronogram from a group of input chronograms. For summarizing approaches that return a single summary tree from a group of phylogenetic trees, it is necessary that the latter form a grove, roughly, a sufficiently overlapping set of taxa between trees, see Ané et al. (2009) doi:10.1007/s00026-0090017x. In rare cases, a group of trees can have multiple groves. This argument indicates whether to get the grove with the most trees (criterion = "trees") or the most taxa (criterion = "taxa").

Details

If only one taxon name is given as input, get_spp_from_taxon is always set to TRUE.

Value

The output is determined by the argument summary_format:

If summary_format = "citations" The function returns a character vector of references.
If summary_format = "mrca" The function returns a named numeric vector of most recent common ancestor (mrca) ages.
If summary_format = "newick_[all, sdm, or median]" The function returns output chronograms as newick strings.
If summary_format = "phylo_[all, sdm, median, or biggest]" The function returns output chronograms as phylo or multiPhylo objects.
If summary_format = "html" or "data_frame" The function returns a 4 column table with data on mrca ages, number of taxa, references, and output chronograms as newick strings.

Examples

## Not run:

# For this example, we will set a temp working directory, but you can set
# your working directory as needed:
# we will use the tempdir() function to get a temporary directory:
tempwd <- tempdir()

# Obtain median ages from a set of source chronograms in newick format:
ages <- datelife_search(c(  
  "Rhea americana", "Pterocnemia pennata", "Struthio camelus",  
  "Mus musculus"  
), summary_format = "newick_median")

# Save the tree in the temp working directory in newick format:
write(ages, file = file.path(tempwd, "some.bird.ages.txt"))
# Obtain median ages from a set of source chronograms in phylo format # Will produce same tree as above but in "phylo" format:
ages.again <- datelife_search(c("Rhea americana", "Pterocnemia pennata", "Struthio camelus",
                                "Mus musculus"), summary_format = "phylo_median")
plot(ages.again)
library(ape)
ape::axisPhylo()
mtext("Time (million years ago)", side = 1, line = 2, at = (max(get("last_plot.phylo",
envir = .PlotPhyloEnv
})$xx) * 0.5))

# Save "phylo" object in newick format
write.tree(ages.again, file = file.path(tempwd, "some.bird.tree.again.txt"))

# Obtain MRCA ages and target chronograms from all source chronograms
# Generate an html output readable in any web browser:
ages.html <- datelife_search(c("Rhea americana", "Pterocnemia pennata", "Struthio camelus",
                               "Mus musculus"), summary_format = "html")
write(ages.html, file = file.path(tempwd, "some.bird.trees.html"))
system(paste("open", file.path(tempwd, "some.bird.trees.html")))

## End(Not run) # end dontrun

---

datelife_use Generate one or multiple chronograms for a set of given taxon names.

**Description**

datelife_use gets secondary calibrations available for any pair of given taxon names, mined from the opentree_chronograms object, congruifies them, and uses them to date a given tree topology with the algorithm defined in dating_method. If no tree topology is provided, it will attempt to get one for the given taxon names from Open Tree of Life synthetic tree, using make_bold_otol_tree().

**Usage**

datelife_use(input = NULL, each = FALSE, dating_method = "bladj", ...)

**Arguments**

input One of the following:

A **character vector** With taxon names as a single comma separated starting or concatenated with c().

A **phylogenetic tree with taxon names as tip labels** As a phylo or multiPhylo object, OR as a newick character string.
A `datelifeQuery` object An output from `make_datelife_query()`. If each Boolean, default to FALSE: all calibrations are returned in the same data.frame. If TRUE, calibrations from each chronogram are returned in separate data frames.

dating_method Tree dating algorithm to use. Options are "bladj" or "pathd8" (Webb et al., 2008, doi:10.1093/bioinformatics/btn358; Britton et al., 2007, doi:10.1080/10635150701613783).

... Arguments passed on to `make_datelife_query`

use_tnrs Whether to use Open Tree of Life’s Taxonomic Name Resolution Service (TNRS) to process input taxon names. Default to TRUE, it corrects misspellings and taxonomic name variations with `tnrs_match()`, a wrapper of `rotl::tnrs_match_names()`.

get_spp_from_taxon Whether to search ages for all species belonging to a given taxon or not. Default to FALSE. If TRUE, it must have same length as input. If input is a newick string with some clades it will be converted to a `phylo` object, and the order of `get_spp_from_taxon` will match `phy$tip.label`.

reference_taxonomy A character vector specifying the reference taxonomy to use for TNRS. Options are "ott", "ncbi", "gbif" or "irmng". The function defaults to "ott".

Details

If input is a vector of taxon names, the function will attempt to reconstruct a BOLD tree with `make_bold_otol_tree()` to get a tree with branch lengths. If it fails, it will get an Open Tree of Life synthetic tree topology. The function then calls `use_calibrations()`.

Value

A `phylo` or `multiPhylo` object with branch lengths proportional to time.

More

The output object stores the used calibrations and `dating_method` as attributes(output)$datelife_calibrations and attributes(output)$dating_method.

---

`datelife_use_datelifequery`

*Generate one or multiple chronograms for a set of taxon names given as a datelifeQuery object.*

---

**Description**

datelife_use gets secondary calibrations available for any pair of given taxon names, mined from the `opentree_chronograms` object, congruifies them, and uses them to date a given tree topology with the algorithm defined in `dating_method`. If no tree topology is provided, it will attempt to get one for the given taxon names from Open Tree of Life synthetic tree, using `make_bold_otol_tree()`.
Usage

datelife_use_datelifequery(
    datelife_query = NULL,
    dating_method = "bladj",
    each = FALSE
)

Arguments

datelife_query A datelifeQuery object, usually an output of `make_datelife_query()`.
dating_method Tree dating algorithm to use. Options are "bladj" or "pathd8" (Webb et al., 2008, doi:10.1093/bioinformatics/btn358; Britton et al., 2007, doi:10.1080/10635150701613783).
each Boolean, default to FALSE: all calibrations are returned in the same data.frame. If TRUE, calibrations from each chronogram are returned in separate data frames.

Details

If phy has no branch lengths, dating_method is ignores, and the function applies secondary calibrations to date the tree with the BLADJ algorithm. See `make_bladj_tree()` and `use_calibrations_bladj()`.
If phy has branch lengths, the function can use the PATHd8 algorithm. See `use_calibrations_pathd8()`.

Value

A phylo or multiPhylo object with branch lengths proportional to time.

More

The output object stores the used calibrations and dating_method as attributes(output)$datelife_calibrations and attributes(output)$dating_method.

---

date_with_pbdb Date with Paleobiology Database and paleotree.

Description

This will take a topology, look up information about fossils for taxa on the tree, and use `paleotree::timePaleoPhy()` to compute branch lengths.

Usage

date_with_pbdb(phy, recent = FALSE, assume_recent_if_missing = TRUE)

Arguments

phy A phylo object.
recent If TRUE, forces the minimum age to be zero for any taxon
assume_recent_if_missing If TRUE, any taxon missing from PBDB is assumed to be recent.
Use congruification to extract secondary calibrations from a datelifeResult object.

This function extracts node ages for each taxon pair given in input$tip.labels. It applies the congruification method described in Eastman et al. (2013) doi:10.1111/2041-210X.12051, implemented with the function geiger::congruify.phylo(), to create a data.frame of taxon pair node ages that can be used as secondary calibrations.

usage

extract_calibrations_dateliferesult(input = NULL, each = FALSE)

Arguments

input A datelifeResult object.

each Boolean, default to FALSE: all calibrations are returned in the same data.frame. If TRUE, calibrations from each chronogram are returned in separate data frames.

Details

The function takes a datelifeResult object and calls summarize_datelife_result() with summary_format = "phylo_all" to convert a datelifeResult object to a phylo or multiPhylo object that is passed to extract_calibrations_phylo().

Value

An object of class calibrations, which is a data.frame (if each = FALSE) or a list of data.frames (if each = TRUE) of node ages for each pair of taxon names. You can access the input data from which the calibrations were extracted with attributes(output)$chronograms.
extract_calibrations_phylo

Use congruification to extract secondary calibrations from a phylo or multiPhylo object with branch lengths proportional to time.

Description

This function extracts node ages for each taxon pair given in input$tip.labels. It applies the congruification method described in Eastman et al. (2013) doi:10.1111/2041210X.12051, implemented with the function geiger::congruify.phylo(), to create a data.frame of taxon pair node ages that can be used as secondary calibrations.

Usage

extract_calibrations_phylo(input = NULL, each = FALSE)

Arguments

input A phylo or multiPhylo object with branch lengths proportional to time.
each Boolean, default to FALSE: all calibrations are returned in the same data.frame. If TRUE, calibrations from each chronogram are returned in separate data frames.

Value

An object of class calibrations, which is a data.frame (if each = FALSE) or a list of data.frames (if each = TRUE) of node ages for each pair of taxon names. You can access the input data from which the calibrations were extracted with attributes(output)$chronograms.

References


extract_ott_ids

Extract numeric OTT ids from a character vector that combines taxon names and OTT ids.

Description

Extract numeric OTT ids from a character vector that combines taxon names and OTT ids.

Usage

extract_ott_ids(x, na.rm = TRUE)

## Default S3 method:
extract_ott_ids(x, na.rm = TRUE)
Arguments

- **x**: A character vector of taxon names, or a phylo object with tree tip labels containing OTT ids.
- **na.rm**: A logical value indicating whether NA values should be stripped from the output.

Value

An object of class numeric containing OTT ids only.

NULL

Examples

```r
## Not run: # This is a flag for package development. You are welcome to run the example.

canis <- rotl::tnrs_match_names("canis")
canis_taxonomy <- rotl::taxonomy_subtree(canis$ott_id)
my_ott_ids <- extract_ott_ids(x = canis_taxonomy$tip_label)
# Get the problematic elements from input
canis_taxonomy$tip_label[attr(my_ott_ids, "na.action")]

## End(Not run) # end dontrun
```

Description

datelifeSummary of a datelifeResult object of all Felidae species.

Usage

felid_gdr_phylo_all

Format

A list of three elements, containing the summary of a datelifeResult object

- **phylo_all**: List of subset chronograms in phylo format
- **taxon_distribution**: A data frame with taxon presence across subset chronograms
- **absent_taxa**: A dataframe with names of taxon not found in any chronogram

Details

Generated with:

```r
felid_spp <- make_datelife_query(input = "felidae", get_spp_from_taxon = TRUE)
felid_gdr <- get_datelife_result(input = felid_spp, get_spp_from_taxon = TRUE)
felid_gdr_phylo_all <- summarize_datelife_result(datelife_result = felid_gdr, taxon_summary = "summary", summary_format = "phylo_all", datelife_query = felid_spp) usethis::use_data(felid_gdr_phylo_all)
```
Source

http://opentreeoflife.org

---

**felid_sdm**

*SDM tree of a datelifeResult object of all Felidae species.*

**Description**

SDM tree of a datelifeResult object of all Felidae species.

**Usage**

felid_sdm

**Format**

A list of two elements, containing the summary of a datelifeResult object

- **phy** An ultrametric phylo object with the SDM tree.
- **data** A datelifeResult object with data used to construct phy

**Details**

Generated with:

```r
felid_spp <- make_datelife_query(input = "felidae", get_spp_from_taxon = TRUE)
felid_gdr <- get_datelife_result(input = felid_spp, get_spp_from_taxon = TRUE) felid_sdm <- datelife_result_sdm_phylo(felid_gdr) usethis::use_data(felid_sdm)
```

Source

http://opentreeoflife.org

---

**filter_for_grove**

*Filter a datelifeResult object to find the largest grove.*

**Description**

Filter a datelifeResult object to find the largest grove.

**Usage**

filter_for_grove(datelife_result, criterion = "taxa", n = 2)
Arguments

- datelife_result
  - A datelifeResult object. Only needed for criterion = "taxa".
- criterion
  - Defaults to criterion = "taxa". Used for chronogram summarizing, i.e., obtaining a single summary chronogram from a group of input chronograms. For summarizing approaches that return a single summary tree from a group of phylogenetic trees, it is necessary that the latter form a grove, roughly, a sufficiently overlapping set of taxa between trees, see Ané et al. (2009) doi:10.1007/s00026-0090017x. In rare cases, a group of trees can have multiple groves. This argument indicates whether to get the grove with the most trees (criterion = "trees") or the most taxa (criterion = "taxa").
- n
  - The degree of taxon name overlap among input chronograms. Defaults to n = 2, i.e., at least two overlapping taxon names.

Value

- A datelifeResult object filtered to only include one grove of trees.

force_ultrametric

Force a non-ultrametric phylo object to be ultrametric with phytools::force.ultrametric().

Description

Force a non-ultrametric phylo object to be ultrametric with phytools::force.ultrametric().

Usage

force_ultrametric(phy)

Arguments

- phy
  - A phylo object.

Value

- A phylo object.
get_all_calibrations

Description

get_all_calibrations performs a `datelife_search()` and gets divergence times (i.e., secondary calibrations) from a chronogram database for each taxon name pair given as input.

Usage

```r
get_all_calibrations(input = NULL, each = FALSE)
```

Arguments

- `input` One of the following:
  - A character vector With taxon names as a single comma separated starting or concatenated with `c()`.
  - A phylogenetic tree with taxon names as tip labels As a phylo or multiPhylo object, OR as a newick character string.
  - A datelifeQuery object An output from `make_datelife_query()`.
- `each` Boolean, default to FALSE: all calibrations are returned in the same data.frame. If TRUE, calibrations from each chronogram are returned in separate data frames.

Value

An object of class calibrations, which is a data.frame (if each = FALSE) or a list of data.frames (if each = TRUE) of node ages for each pair of taxon names. You can access the input data from which the calibrations were extracted with attributes(output)$chronograms.

get_all_descendant_species

Description

This is less thorough than `get_open_tree_species()`, but much faster. It uses the fact that something has just two names (genus and species) to assume that something is a single species; if it has more than two names, it is assumed to be a subspecies so it goes up one level in the hierarchy. It will return the subspecies and the species.
get_best_grove

Usage

get_all_descendant_species(taxon_name, ott_id)

Arguments

taxon_name  A character vector providing an inclusive taxonomic name.
ott_id  A numeric vector providing an Open Tree Taxonomic id number for a taxonomic name. If provided, taxon_name is ignored. Used in the context of OTT to detect invalid taxon names.

Value

A list of unique OTT names and OTT ids of species within the provided taxon.

get_best_grove

Get grove from a datelifeResult object that can be converted to phylo from a median summary matrix

Description

Get grove from a datelifeResult object that can be converted to phylo from a median summary matrix

Usage

get_best_grove(datelife_result, criterion = "taxa", n = 2)

Arguments

datelife_result  A datelifeResult object. Only needed for criterion = "taxa".
criterion  Defaults to criterion = "taxa". Used for chronogram summarizing, i.e., obtaining a single summary chronogram from a group of input chronograms. For summarizing approaches that return a single summary tree from a group of phylogenetic trees, it is necessary that the latter form a grove, roughly, a sufficiently overlapping set of taxa between trees, see Ané et al. (2009) doi:10.1007/s00026-0090017x. In rare cases, a group of trees can have multiple groves. This argument indicates whether to get the grove with the most trees (criterion = "trees") or the most taxa (criterion = "taxa").
n  The degree of taxon name overlap among input chronograms. Defaults to n = 2, i.e., at least two overlapping taxon names.
Value

A list of two elements:

- **best_grove** A datelifeResult object filtered to only include one grove of trees that can be summarized with median or sdm.
- **overlap** The degree of taxon names overlap among trees in the best grove.

---

**get_biggest_multiphylo**

*Get the tree with the most tips from a multiPhylo object: the biggest tree.*

---

Description

Get the tree with the most tips from a multiPhylo object: the biggest tree.

Usage

```r
get_biggest_multiphylo(trees)
```

Arguments

- `trees` A list of trees as multiPhylo or as a generic list object.

Value

The largest tree from those given in `trees`, as a phylo object with an additional `$citation` element containing the reference of the original publication.

---

**get_bold_data**

*Get genetic data from the Barcode of Life Database (BOLD) for a set of taxon names.*

---

Description

`get_bold_data` uses taxon names from a tree topology, a character vector of names or a datelifeQuery object, to search for genetic markers in the Barcode of Life Database (BOLD).

Usage

```r
get_bold_data(
    input = c("Rhea americana", "Struthio camelus", "Gallus gallus"),
    marker = "COI",
    ...
)
```
Arguments

input  One of the following:

A character vector  With taxon names as a single comma separated starting or concatenated with \texttt{c()}.

A phylogenetic tree with taxon names as tip labels  As a \texttt{phylo} or \texttt{multiPhylo} object, OR as a newick character string.

A datelifeQuery object  An output from \texttt{make_datelife_query()}.

marker  A character vector indicating the gene from BOLD system to be used for branch length estimation. It searches "COI" marker by default.

...  Arguments passed on to \texttt{get_otol_synthetic_tree}

otol_version  Version of Open Tree of Life to use

resolve  Defaults to \texttt{TRUE}. Whether to resolve the tree at random or not.

ott_ids  If not \texttt{NULL}, it takes this argument and ignores input. A numeric vector of ott ids obtained with \texttt{rotl::taxonomy_taxon_info()} or \texttt{rotl::tnrs_match_names()} or \texttt{tnrs_match()}.

Details

If input is a \texttt{phylo} object or a newick string, it is used as backbone topology. If input is a character vector of taxon names, an induced synthetic OpenTree subtree is used as backbone.

Value

A \texttt{phylo} object. If there are enough BOLD sequences available for the input taxon names, the function returns a tree with branch lengths proportional to relative substitution rate. If not enough BOLD sequences are available for the input taxon names, the function returns the topology given as input, or a synthetic Open Tree of Life for the taxon names given in input, obtained with \texttt{get_otol_synthetic_tree()}.

Description

The function searches DateLife’s local database of phylogenetic trees with branch lengths proportional to time (chronograms) with \texttt{datelife_search()}, and extracts available node ages for each pair of given taxon names with \texttt{extract_calibrations_phylo()}.

Usage

\texttt{get_calibrations_datelifequery(datelife_query = NULL, each = FALSE)}
get_calibrations_vector

Search and extract secondary calibrations for a given character vector of taxon names

Arguments

datelife_query A datelifeQuery object.
each Boolean, default to FALSE: all calibrations are returned in the same data.frame.
If TRUE, calibrations from each chronogram are returned in separate data frames.

Details

The function calls datelife_search() with summary_format = "phylo_all" to get all chronograms in the database containing at least two taxa in input, and generates a phylo or multiPhylo object object that will be passed to extract_calibrations_phylo().

Value

An object of class calibrations, which is a data.frame (if each = FALSE) or a list of data.frames (if each = TRUE) of node ages for each pair of taxon names. You can access the input data from which the calibrations were extracted with attributes(output)$chronograms.

Usage

get_calibrations_vector(input = NULL, each = FALSE)

Arguments

input A character vector of taxon names.
each Boolean, default to FALSE: all calibrations are returned in the same data.frame.
If TRUE, calibrations from each chronogram are returned in separate data frames.

Details

The function calls datelife_search() with summary_format = "phylo_all" to get all chronograms in the database containing at least two taxa in input, and generates a phylo or multiPhylo object object that will be passed to extract_calibrations_phylo().
An object of class calibrations, which is a data.frame (if each = FALSE) or a list of data.frames (if each = TRUE) of node ages for each pair of taxon names. You can access the input data from which the calibrations were extracted with attributes(output)$chronograms.

Description
Get a dated OpenTree induced synthetic subtree from a set of given taxon names, from blackrim’s FePhyFoFum service.

Usage
get_dated_otol_induced_subtree(input = NULL, ott_ids = NULL, ...)

Arguments
- input: Optional. A character vector of names or a datelifeQuery object.
- ott_ids: If not NULL, it takes this argument and ignores input. A numeric vector of ott ids obtained with rotl::taxonomy_taxon_info() or rotl::tnrs_match_names() or tnrs_match().
- ...: Arguments passed on to check_ott_input

Details
OpenTree dated tree from Stephen Smith’s OpenTree scaling service at https://github.com/FePhyFoFum/gophy
if you want to make an LTT plot of a dated OpenTree tree you’ll need to get rid of singleton nodes with ape::collapse.singles() and also probably do phytools::force.ultrametric().

Value
A phylo object with edge length proportional to time in Myrs. It will return NA if any ott_id is invalid.


get_datelife_result  Get a patristic matrix of time of lineage divergence data for a given set of taxon names

Description

get_datelife_result takes as input a vector of taxon names, a newick string, a phylo object, or adatelifeQuery object. It searches the chronogram database specified in cache for chronograms matching two or more given taxon names. For each matching chronogram, it extracts time of lineage divergence data and stores it as a patristic matrix. It then lists all resulting patristic matrices. Each list element is named with the study citation of the source chronogram.

Usage

get_datelife_result(
  input = NULL,
  partial = TRUE,
  cache = "opentree_chronograms",
  update_opentree_chronograms = FALSE,
  ...
)

Arguments

input One of the following:

  A character vector With taxon names as a single comma separated starting or concatenated with c().

  A phylogenetic tree with taxon names as tip labels As a phylo or multiPhylo object, OR as a newick character string.

  A datelifeQuery object An output from make_datelife_query().

partial Whether to return or exclude partially matching source chronograms, i.e, those that match some and not all of taxa given in datelife_query. Options are TRUE or FALSE. Defaults to TRUE: return all matching source chronograms.

cache A character vector of length one, with the name of the data object to cache. Default to "opentree_chronograms", a data object storing Open Tree of Life's database chronograms and other associated information.

update_opentree_chronograms Whether to update the chronogram database or not. Defaults to FALSE.

... Arguments passed on to make_datelife_query

use_tnrs Whether to use Open Tree of Life's Taxonomic Name Resolution Service (TNRS) to process input taxon names. Default to TRUE, it corrects misspellings and taxonomic name variations with tnrs_match(), a wrapper of rotl::tnrs_match_names().
get_spp_from_taxon  Whether to search ages for all species belonging to a given taxon or not. Default to FALSE. If TRUE, it must have same length as input. If input is a newick string with some clades it will be converted to a phylo object, and the order of get_spp_from_taxon will match phy$tip.label.

reference_taxonomy  A character vector specifying the reference taxonomy to use for TNRS. Options are "ott", "ncbi", "gbif" or "irmng". The function defaults to "ott".

Value
A datelifeResult object – a named list of patristic matrices.

get_datelife_result_datelifequery

Get a list of patristic matrices from a given datelifeQuery object

Description
Get a list of patristic matrices from a given datelifeQuery object

Usage
get_datelife_result_datelifequery(
  datelife_query = NULL,
  partial = TRUE,
  cache = "opentree_chronograms",
  update_opentree_chronograms = FALSE,
  ...
)

Arguments

datelife_query  A datelifeQuery object, usually an output of make_datelife_query().
partial         Whether to return or exclude partially matching source chronograms, i.e, those that match some and not all of taxa given in datelife_query. Options are TRUE or FALSE. Defaults to TRUE: return all matching source chronograms.
cache           A character vector of length one, with the name of the data object to cache. Default to "opentree_chronograms", a data object storing Open Tree of Life’s database chronograms and other associated information.
update_opentree_chronograms

...              Arguments passed on to make_datelife_query

input            Taxon names as one of the following:

  A character vector of taxon names  With taxon names as a single comma separated starting or concatenated with c().
A phylogenetic tree with taxon names as tip labels As a phylo or multiPhylo object, OR as a newick character string.

use_tnrs Whether to use Open Tree of Life’s Taxonomic Name Resolution Service (TNRS) to process input taxon names. Default to TRUE, it corrects misspellings and taxonomic name variations with tnrs_match(), a wrapper of rotl::tnrs_match_names().

get_spp_from_taxon Whether to search ages for all species belonging to a given taxon or not. Default to FALSE. If TRUE, it must have same length as input. If input is a newick string with some clades it will be converted to a phylo object, and the order of get_spp_from_taxon will match phy$tip.label.

reference_taxonomy A character vector specifying the reference taxonomy to use for TNRS. Options are "ott", "ncbi", "gbif" or "irmng". The function defaults to "ott".

Details

If there is just one taxon name in input$cleaned_names, the function will run make_datelife_query() setting get_spp_from_taxon = TRUE. The datelifeQuery used as input can be accessed with attributes(datelifeResult)$query.

Value

A datelifeResult object – a named list of patristic matrices.

get_fossil_range

Get the ages for a taxon from PBDB

Description

This uses the Paleobiology Database’s API to gather information on the ages for all specimens of a taxon. It will also look for all descendants of the taxon. It fixes name misspellings if possible.

Usage

get_fossil_range(taxon, recent = FALSE, assume_recent_if_missing = TRUE)

Arguments

taxon The scientific name of the taxon you want the range of occurrences of
recent If TRUE, forces the minimum age to be zero
assume_recent_if_missing
        If TRUE, any taxon missing from pbdb is assumed to be recent

Value

a data.frame of max_ma and min_ma for the specimens
**get_goodmatrices**

Get indices of good matrices to apply Super Distance Matrix (SDM) method with `make_sdm()`.

**Description**

Get indices of good matrices to apply Super Distance Matrix (SDM) method with `make_sdm()`.

**Usage**

```r
get_goodmatrices(unpadded.matrices)
```

**Arguments**

- `unpadded.matrices`:
  A list of patristic matrices, a datelifeResult object.

**Value**

A numeric vector of good matrix indices in unpadded.matrices.

---

**get_mrbayes_node_constraints**

Makes a block of node constraints and node calibrations for a MrBayes run file from a list of taxa and ages, or from a dated tree.

**Description**

Makes a block of node constraints and node calibrations for a MrBayes run file from a list of taxa and ages, or from a dated tree.

**Usage**

```r
get_mrbayes_node_constraints(
  constraint = NULL,
  taxa = NULL,
  missing_taxa = NULL,
  ncalibration = NULL,
  age_distribution = "fixed",
  root_calibration = FALSE,
  mrbayes_constraints_file = NULL,
  clockratepr = "prset clockratepr = fixed(1);"
)
```
### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>constraint</td>
<td>The constraint tree: a phylo object or a newick character string, with or without branch lengths.</td>
</tr>
<tr>
<td>taxa</td>
<td>A character vector with taxon names to be maintained in tree</td>
</tr>
<tr>
<td>missing_taxa</td>
<td>A tree, a data frame or a vector enlisting all missing taxa you want to include.</td>
</tr>
<tr>
<td>ncalibration</td>
<td>The node calibrations: a phylo object with branch lengths proportional to time; in this case all nodes from ncalibration will be used as calibration points. Alternatively, a list with two elements: the first is a character vector with node names from phy to calibrate; the second is a numeric vector with the corresponding ages to use as calibrations.</td>
</tr>
<tr>
<td>age_distribution</td>
<td>A character string specifying the type of calibration. Only &quot;fixed&quot; and &quot;uniform&quot; are implemented for now.</td>
</tr>
<tr>
<td>root_calibration</td>
<td>Used to set a calibration at the root or not. Default to FALSE. Only relevant if ncalibration is specified.</td>
</tr>
<tr>
<td>mrbayes_constraints_file</td>
<td>NULL or a character vector indicating the name of mrbayes constraint and/or calibration block file.</td>
</tr>
<tr>
<td>clockrateprior</td>
<td>A character vector indicating the clockrateprior to be used.</td>
</tr>
</tbody>
</table>

### Value

A set of MrBayes constraints and/or calibration commands printed in console as character strings or as a text file specified in mrbayes_constraints_file.
get_opentree_chronograms

Get all chronograms from Open Tree of Life database

Description

Get all chronograms from Open Tree of Life database

Usage

get_opentree_chronograms(max_tree_count = "all")

get_otol_chronograms(max_tree_count = "all")

Arguments

max_tree_count Default to "all", it gets all available chronograms. For testing purposes, a numeric value indicating the max number of trees to be cached.

Value

A list of 4 elements:

authors A list of lists of author names of the original studies that published chronograms currently stored in the Open Tree of Life database.

curators A list of lists of curator names that uploaded chronograms to the Open Tree of Life database.

studies A list of study identifiers from original studies that published chronograms currently stored in the Open Tree of Life database.

trees A multiPhylo object storing the chronograms from Open Tree of Life database.

update A character vector indicating the time when the database object was last updated.

version A character vector indicating the datelife package version when the object was last updated.

get_opentree_species

Get all species belonging to a taxon from the Open Tree of Life Taxonomy (OTT)

Description

Get all species belonging to a taxon from the Open Tree of Life Taxonomy (OTT)

Usage

get_opentree_species(taxon_name, ott_id, synth_tree_only = TRUE)
get_otol_synthetic_tree

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>taxon_name</td>
<td>A character vector providing an inclusive taxonomic name.</td>
</tr>
<tr>
<td>ott_id</td>
<td>A numeric vector providing an Open Tree Taxonomic id number for a taxonomic name. If provided, taxon_name is ignored. Used in the context of OTT to detect invalid taxon names.</td>
</tr>
<tr>
<td>synth_tree_only</td>
<td>Whether to include species that are in the synthetic Open Tree of Life only or not. Default to TRUE.</td>
</tr>
</tbody>
</table>

Value

A list of unique OTT names and OTT ids of species within the provided taxon.

get_otol_synthetic_tree

Get an Open Tree of Life synthetic subtree of a set of given taxon names.

Description

Get an Open Tree of Life synthetic subtree of a set of given taxon names.

Usage

get_otol_synthetic_tree(
  input = NULL,
  ott_ids = NULL,
  otol_version = "v3",
  resolve = FALSE,
  ...
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>Optional. A character vector of names or a datelifeQuery object.</td>
</tr>
<tr>
<td>ott_ids</td>
<td>If not NULL, it takes this argument and ignores input. A numeric vector of ott ids obtained with rotl::taxonomy_taxon_info() or rotl::tnrs_match_names() or tnrs_match().</td>
</tr>
<tr>
<td>otol_version</td>
<td>Version of Open Tree of Life to use</td>
</tr>
<tr>
<td>resolve</td>
<td>Defaults to TRUE. Whether to resolve the tree at random or not.</td>
</tr>
<tr>
<td>...</td>
<td>Arguments passed on to check_ott_input</td>
</tr>
</tbody>
</table>

Value

A phylo object
get_ott_children

Use this instead of `rotl::tol_subtree()` when taxa are not in synthesis tree and you still need to get all species or an induced OpenTree subtree

Description

Use this instead of `rotl::tol_subtree()` when taxa are not in synthesis tree and you still need to get all species or an induced OpenTree subtree

Usage

```r
get_ott_children(input = NULL, ott_ids = NULL, ott_rank = "species", ...)
```

Arguments

- `input`: Optional. A character vector of names or a `datelifeQuery` object.
- `ott_ids`: If not NULL, it takes this argument and ignores input. A numeric vector of `ott_ids` obtained with `rotl::taxonomy_taxon_info()` or `rotl::tnrs_match_names()` or `tnrs_match()`.
- `ott_rank`: A character vector with the ranks you wanna get lineage children from.
- `...`: Other arguments to pass to `get_valid_children()`.

Value

A `data.frame` object.

Examples

```r
# An example with the dog genus:
# It is currently not possible to get an OpenTree subtree of a taxon that is
# missing from the OpenTree synthetic tree.
# The dog genus is not monophyletic in the OpenTree synthetic tree, so in
# practice, it has no node to extract a subtree from.
tnrs <- tnrs_match("Canis")

## Not run: # This is a flag for package development. You are welcome to run the example.
rotl::tol_subtree(tnrs$ott_id[1])
#> Error: HTTP failure: 400
#> [/v3/tree_of_life/subtree] Error: node_id was not found (broken taxon).

## End(Not run) # end dontrun

ids <- tnrs$ott_id[1]
names(ids) <- tnrs$unique_name
children <- get_ott_children(ott_ids = ids) # or
```
children <- get_ott_children(input = "Canis")
if (!is.na(children)) {
  str(children)
  ids <- children$Canis$ott_id
  names(ids) <- rownames(children$Canis)
  tree_children <- datelife::get_otol_synthetic_tree(ott_ids = ids)
  plot(tree_children, cex = 0.3)
}

# An example with flowering plants:

## Not run: # This is a flag for package development. You are welcome to run the example.

oo <- get_ott_children(input = "magnoliophyta", ott_rank = "order")
# Get the number of orders of flowering plants that we have
sum(oo$Magnoliophyta$rank == "order")

## End(Not run) # end dontrun

---

**get_ott_clade**  
*Get the Open Tree of Life Taxonomic identifiers (OTT ids) and name of one or several given taxonomic ranks from one or more input taxa.*

### Description

Get the Open Tree of Life Taxonomic identifiers (OTT ids) and name of one or several given taxonomic ranks from one or more input taxa.

### Usage

```
get_ott_clade(input = NULL, ott_ids = NULL, ott_rank = "family")
```

### Arguments

- **input**: Optional. A character vector of names or a datelifeQuery object.
- **ott_ids**: If not NULL, it takes this argument and ignores input. A numeric vector of ott ids obtained with `rotl::taxonomy_taxon_info()` or `rotl::tnrs_match_names()` or `tnrs_match()`.
- **ott_rank**: A character vector with the ranks you wanna get lineage children from.

### Value

A list of named numeric vectors with OTT ids from input and all requested ranks.
get_ott_lineage

Get the Open Tree of Life Taxonomic identifier (OTT id) and name of all lineages from one or more input taxa.

**Description**

Get the Open Tree of Life Taxonomic identifier (OTT id) and name of all lineages from one or more input taxa.

**Usage**

```r
get_ott_lineage(input = NULL, ott_ids = NULL)
```

**Arguments**

- **input**: Optional. A character vector of names or a datelifeQuery object.
- **ott_ids**: If not NULL, it takes this argument and ignores input. A numeric vector of ott ids obtained with `rotl::taxonomy_taxon_info()` or `rotl::tnrs_match_names()` or `tnrs_match()`.

**Value**

A list of named numeric vectors of ott ids from input and all the clades it belongs to.

**Examples**

```r
## Not run: # This is a flag for package development. You are welcome to run the example.
taxa <- c("Homo", "Bacillus anthracis", "Apis", "Salvia")
lin <- get_ott_lineage(taxa)
lin
# Look up an unknown OTT id:
get_ott_lineage(ott_id = 454749)

## End(Not run) # end dontrun
```

---

get_subset_array_dispatch

*Figure out which subset function to use.*

**Description**

`get_subset_array_dispatch` is used inside `get_datelife_result()`
Usage

get_subset_array_dispatch(
  study_element,
  taxa,
  phy = NULL,
  phy4 = NULL,
  dating_method = "PATHd8"
)

Arguments

study_element  The thing being passed in: an array or a phylo object to serve as reference for congruification.
taxa           Vector of taxon names to get a subset for.
phy             A user tree to congruify as phylo object (ape).
phy4            A user tree to congruify in phylo4 format (phylobase).
dating_method  The method used for tree dating.

Value

A patristic matrix with ages for the target taxa.

Description

Get a taxon summary of a datelifeResult object.

Usage

get_taxon_summary(datelife_result = NULL, datelife_query = NULL)

Arguments

datelife_result  A datelifeResult object, usually an output of get_datelife_result().
datelife_query   A datelifeQuery object, usually an output of make_datelife_query().
Value

A datelifeTaxonSummary object, which is a list of 4 elements:

$matrix  Data as a presence/absence matrix of taxon names across chronograms.

$summary  A data.frame with taxon names as row.names() and two columns, one with the number of chronograms that contain a taxon name and the other one with the total number of chronograms that have at least 2 taxon names.

$summary2  A data.frame with chronogram citations as row.names() and two columns, one with the number of taxon names found in each chronogram and the other one with the total number of taxon names.

$absent_taxa  A character vector of taxon names that are not found in the chronogram database.

get_tnrs_names  Process a character vector of taxon names with TNRS

Description

make_datelife_query2 always uses TNRS (Taxonomic Name Resolution Service to process input taxon names, to correct misspellings and taxonomic name variations with tnrs_match(), a wrapper of rotl::tnrs_match_names()).

Usage

get_tnrs_names(
  input = c("Rhea americana", "Pterocnemia pennata", "Struthio camelus"),
  reference_taxonomy = "ott",
  ...
)

Arguments

input  Taxon names as a character vector of taxon names. Two or more names can be provided as a single comma separated string or concatenated with c().

reference_taxonomy  A character vector specifying the reference taxonomy to use for TNRS. Options are "ott", "ncbi", "gbif" or "irmng". The function defaults to "ott".

...  Arguments passed on to rotl::tnrs_match_names

context_name name of the taxonomic context to be searched (length-one character vector or NULL). Must match (case sensitive) one of the values returned by tnrs_contexts. Default to "All life".

do_approximate_matching A logical indicating whether or not to perform approximate string (a.k.a. "fuzzy") matching. Using FALSE will greatly improve speed. Default, however, is TRUE.

ids  A vector of ids to use for identifying names. These will be assigned to each name in the names array. If ids is provided, then ids and names must be identical in length.
Ordinarily, some quasi-taxa, such as incertae sedis buckets and other non-OTUs, are suppressed from TNRS results. If this parameter is true, these quasi-taxa are allowed as possible TNRS results.

Value

A `datelifeTNRS` object, which is a list of three elements:

- `$cleaned_names` A character vector of names provided as input.
- `$tnrs_names` A character vector of taxon names processed with TNRS.
- `$ott_ids` A numeric vector of Open Tree of Life Taxonomy (OTT) ids.

---

**get_valid_children**

Extract valid children from given taxonomic name(s) or Open Tree of Life Taxonomic identifiers (OTT ids) from a taxonomic source.

### Description

Extract valid children from given taxonomic name(s) or Open Tree of Life Taxonomic identifiers (OTT ids) from a taxonomic source.

### Usage

```r
get_valid_children(input = NULL, ott_ids = NULL, reference_taxonomy = "ncbi")
```

### Arguments

- **input** Optional. A character vector of names or a `datelifeQuery` object.
- **ott_ids** If not NULL, it takes this argument and ignores input. A numeric vector of ott ids obtained with `rotl::taxonomy_taxon_info()` or `rotl::tnrs_match_names()` or `tnrs_match()`.
- **reference_taxonomy** A character vector with the desired taxonomic sources. Options are "ncbi", "gbif" or "irmng". Any other value will retrieve data from all taxonomic sources. The function defaults to "ncbi".

### Details

GBIF and other taxonomies contain deprecated taxa that are not marked as such in the Open Tree of Life Taxonomy. We are relying mainly in the NCBI taxonomy for now.

### Value

A named list containing valid taxonomic children of given taxonomic name(s).
Examples

# genus Dictyophyllidites with ott id = 6003921 has only extinct children
# in cases like this the same name will be returned

tti <- rotl::taxonomy_taxon_info(6003921, include_children = TRUE)
gvc <- get_valid_children(ott_ids = 6003921)

# More examples:

gvc <- get_valid_children(ott_ids = 769681) # Psilotopsida

gvc <- get_valid_children(ott_ids = 56601) # Marchantiophyta

input_process(input_process)

Process a phylo object or a character string to determine if it’s correct newick

Description

Process a phylo object or a character string to determine if it’s correct newick

Usage

input_process(input)

Arguments

input

Taxon names as one of the following:

A character vector of taxon names With taxon names as a single comma separated or concatenated with \( \text{c}() \).

A phylogenetic tree with taxon names as tip labels As a phylo or multiPhylo object, OR as a newick character string.

Value

A phylo object or NA if input is not a tree.

is_datelife_query

Check if input is a datelifeQuery object

Description

is_datelife_query checks for two things to be TRUE or FALSE. First, that input is of class datelifeQuery. Second, that input is a list that contains at least two elements of a datelifeQuery object:

cleaned_names A character vector of taxon names.

phy Either NA or a phylo object.
Usage

is_datelife_query(input)

Arguments

input An object to be checked as an object with essential properties of a `datelife-Query` object.

Details

If the object has the correct format but it has a class different than `datelifeQuery`, the class is not modified.

Value

Is determined by the second condition.

---

is_datelife_result_empty

*Check if we obtained an empty search with the given taxon name(s).*

---

Description

Check if we obtained an empty search with the given taxon name(s).

Usage

is_datelife_result_empty(datelife_result, use_tnrs = FALSE)

Arguments

datelife_result A datelifeResult object, usually an output of `get_datelife_result()`.

use_tnrs Whether to use Open Tree of Life’s Taxonomic Name Resolution Service (TNRS) to process input taxon names. Default to TRUE, it corrects misspellings and taxonomic name variations with `tnrs_match()`, a wrapper of `rotl::tnrs_match_names()`.

Value

Boolean. If TRUE, no chronograms were found for the given taxon name(s). If FALSE, the chronogram search was successful.
is_good_chronogram

Check if a tree is a valid chronogram.

Description
Check if a tree is a valid chronogram.

Usage
is_good_chronogram(phy)

Arguments
phy A phylo object.

Value
TRUE if it is a valid tree.

is_n_overlap
Function for computing n-overlap for two vectors of names (ie., phy1$tip.label, phy2$tip.label) and seeing if they have n overlap

Description
This function implements definition 2.8 for n-overlap from Ané et al. (2009) doi:10.1007/s00026-0090017x.

Usage
is_n_overlap(names_1, names_2, n = 2)

Arguments
names_1 First vector of names
names_2 Second vector of names
n Degree of overlap required

Value
Boolean for whether the degree of overlap was met or not.

References
**make_all_associations**  
*Find all authors and where they have deposited their trees*

**Description**  
Find all authors and where they have deposited their trees

**Usage**  
`make_all_associations(outputfile = "depositorcache.RData")`

**Arguments**
- **outputfile**  
  Path including file name. NULL to prevent saving.

**Value**  
A data.frame of "person" and "urls".

---

**make_bladj_tree**  
*Use the BLADJ algorithm to get a chronogram from a tree topology for which you have age data for some of its nodes.*

**Description**  
The function takes a tree topology and uses the BLADJ algorithm implemented with `phylocomr::ph_bladj()` to assign node ages and branch lengths, given a set of fixed node ages and respective node names.

**Usage**  
`make_bladj_tree(tree = NULL, nodenames = NULL, nodeages = NULL)`

**Arguments**
- **tree**  
  A tree either as a newick character string or as a phylo object.
- **nodenames**  
  A character vector with names of nodes in tree with known ages
- **nodeages**  
  A numeric vector with the actual ages of named nodes

**Details**  
Input tree can be dated or not, $\text{edge.length}$ is ignored. Ages given in nodeages are fixed on their corresponding nodes given in nodenames.

**Value**  
A phylo object.
**make_bold_otol_tree**

Use genetic data from the Barcode of Life Database (BOLD) to reconstruct branch lengths on a tree.

---

**Description**

`make_bold_otol_tree` takes taxon names from a tree topology or a vector of names to search for genetic markers in the Barcode of Life Database (BOLD), create an alignment, and reconstruct branch lengths on a tree topology with Maximum Likelihood.

**Usage**

```r
make_bold_otol_tree(
  input = c("Rhea americana", "Struthio camelus", "Gallus gallus"),
  marker = "COI",
  otol_version = "v3",
  chronogram = TRUE,
  doML = FALSE,
  aligner = "muscle",
  ...
)
```

**Arguments**

- **input**
  - One of the following:
    - A **character vector** with taxon names as a single comma separated starting or concatenated with `c()`.
    - A **phylogenetic tree with taxon names as tip labels** as a phylo or multiPhylo object, OR as a newick character string.
    - A **datelifeQuery object** an output from `make_datelife_query()`.

- **marker**
  - A character vector indicating the gene from BOLD system to be used for branch length estimation.

- **otol_version**
  - Version of Open Tree of Life to use.

- **chronogram**
  - Default to TRUE, branch lengths returned are estimated with `ape::chronoMPL()`.
  - If FALSE, branch lengths returned are estimated with `phangorn::acctran()` and represent relative substitution rates.

- **doML**
  - Default to FALSE. If TRUE, it does a ML branch length optimization with `phangorn::optim.pml()`.

- **aligner**
  - A character vector indicating whether to use MAFFT or MUSCLE to align BOLD sequences. It is not case sensitive. Default to MUSCLE, supported using the msa package from Bioconductor, which needs to be installed using `BiocManager::install()`.

- **...**
  - Arguments passed on to `get_otol_synthetic_tree`

- **resolve**
  - Defaults to TRUE. Whether to resolve the tree at random or not.

- **ott_ids**
  - If not NULL, it takes this argument and ignores input. A numeric vector of ott ids obtained with `rotl::taxonomy_taxon_info()` or `rotl::tnrs_match_names()` or `tnrs_match()`.
make_datelife_query

Details

If input is a phylo object or a newick string, it is used as backbone topology. If input is a character vector of taxon names, an induced synthetic OpenTree subtree is used as backbone.

Value

A phylo object. If there are enough BOLD sequences available for the input taxon names, the function returns a tree with branch lengths proportional to relative substitution rate. If not enough BOLD sequences are available for the input taxon names, the function returns the topology given as input, or a synthetic Open Tree of Life for the taxon names given in input, obtained with get_otol_synthetic_tree().

make_contributor_cache

Create a cache from Open Tree of Life

Description

Create a cache from Open Tree of Life

Usage

make_contributor_cache(outputfile = "contributorcache.RData")

Arguments

outputfile	Path including file name

Value

List containing author and curator results

make_datelife_query

Go from taxon names to a datelifeQuery object

Description

Go from taxon names to a datelifeQuery object

Usage

make_datelife_query(
    input = c("Rhea americana", "Pterocnemia pennata", "Struthio camelus"),
    use_tnrs = TRUE,
    get_spp_from_taxon = FALSE,
    reference_taxonomy = "ott"
)
Arguments

- **input**
  
  Taxon names as one of the following:
  
  A **character vector of taxon names**  With taxon names as a single comma separated starting or concatenated with c().
  
  A **phylogenetic tree with taxon names as tip labels**  As a phylo or multiPhylo object, OR as a newick character string.

- **use_tnrs**
  
  Whether to use Open Tree of Life’s Taxonomic Name Resolution Service (TNRS) to process input taxon names. Default to TRUE, it corrects misspellings and taxonomic name variations with tnr$match(), a wrapper of rotl::tnr$match_names().

- **get_spp_from_taxon**
  
  Whether to search ages for all species belonging to a given taxon or not. Default to FALSE. If TRUE, it must have same length as input. If input is a newick string with some clades it will be converted to a phylo object, and the order of get_spp_from_taxon will match phy$tip.label.

- **reference_taxonomy**
  
  A character vector specifying the reference taxonomy to use for TNRS. Options are "ott", "ncbi", "gbif" or "irmng". The function defaults to "ott".

Details

It processes phylo objects and newick character string inputs with input_process(). If input is a multiPhylo object, only the first phylo element will be used. Similarly, if an input newick character string has multiple trees, only the first one will be used.

Value

A datelifeQuery object, which is a list of three elements:

- **$phy**  A phylo object or NA, if input is not a tree.
- **$cleaned_names**  A character vector of cleaned taxon names.
- **$ott_ids**  A numeric vector of OTT ids if use_tnrs = TRUE, or NULL if use_tnrs = FALSE.

Description

Go from taxon names to a datelifeQuery object

Usage

```r
make_datelife_query2(
  input = c("Rhea americana", "Pterocnemia pennata", "Struthio camelus"),
  get_spp_from_taxon = FALSE,
  reference_taxonomy = "ott",
  ...
)
```
Arguments

input Taxon names as one of the following:

A character vector of taxon names With taxon names as a single comma separated starting or concatenated with c().

A phylogenetic tree with taxon names as tip labels As a phylo or multiPhylo object, OR as a newick character string.

get_spp_from_taxon Whether to search ages for all species belonging to a given taxon or not. Default to FALSE. If TRUE, it must have same length as input. If input is a newick string with some clades it will be converted to a phylo object, and the order of get_spp_from_taxon will match phy$tip.label.

reference_taxonomy A character vector specifying the reference taxonomy to use for TNRS. Options are "ott", "ncbi", "gbif" or "irmng". The function defaults to "ott".

Details

It processes phylo objects and newick character string inputs with input_process(). If input is a multiPhylo object, only the first phylo element will be used. Similarly, if an input newick character string has multiple trees, only the first one will be used.

Value

A datelineQuery object, which is a list of four elements:

$input_names A character vector of input taxon names.
$tnrs_names A character vector of taxon names processed with TNRS.
$sott_ids A numeric vector of OTT ids.
$phy A phylo object or NA, if input is not a tree.
**make_mrbayes_runfile**  
*Make a mrBayes run block file with a constraint topology and a set of node calibrations and missing taxa*

**Description**

Make a mrBayes run block file with a constraint topology and a set of node calibrations and missing taxa

**Usage**

```r
make_mrbayes_runfile(
  constraint = NULL,
  taxa = NULL,
  ncalibration = NULL,
  missing_taxa = NULL,
  age_distribution = "fixed",
  root_calibration = FALSE,
  mrbayes_output_file = "mrbayes_run.nexus"
)
```

**Arguments**

- **constraint**  
The constraint tree: a phylo object or a newick character string, with or without branch lengths.

- **taxa**  
A character vector with taxon names to be maintained in tree

- **ncalibration**  
The node calibrations: a phylo object with branch lengths proportional to time; in this case all nodes from ncalibration will be used as calibration points. Alternatively, a list with two elements: the first is a character vector with node names from phy to calibrate; the second is a numeric vector with the corresponding ages to use as calibrations.

- **missing_taxa**  
A tree, a data frame or a vector enlisting all missing taxa you want to include.

  A **tree** Either as a phylo object or as a newick character string. It contains all taxa that you want at the end, both missing and non missing. This tree will be used as a hard constraint.

  A **data.frame** It contains two columns named "taxon" and "clade". The first one contains a character vector of missing taxon names. The second one contains a character or numeric vector of nodes from a constraint tree to which each taxon will be assigned.

  A **character vector** It contains the names of the missing taxa. They will be added at random to the constraint tree.

- **age_distribution**  
A character string specifying the type of calibration. Only "fixed" and "uniform" are implemented for now.

  **fixed** The age given in ncalibration will be used as fixed age.
The age given in ncalibration will be used as mean age. The standard deviation can be provided. # still need to add this option. By default, a 95 CI sd is used.

**uniform** The age given in ncalibration will be used as mean age. Where min_age = 0.9 * mean age, and max_age = 1.1 * mean age.

**root_calibration**
Used to set a calibration at the root or not. Default to FALSE. Only relevant if ncalibration is specified.

**mrbayes_output_file**
A character vector specifying the name of mrBayes run file and outputs (can specify directory too).

**Value**
A MrBayes block run file in nexus format.

---

**Description**
Take a constraint tree and use mrBayes to get node ages and branch lengths given a set of node calibrations without any data.

**Usage**

```r
globus 
```make_mrbayes_tree

```r
make_mrbayes_tree(
    constraint = NULL,
    taxa = NULL,
    ncalibration = NULL,
    missing_taxa = NULL,
    age_distribution = "fixed",
    root_calibration = FALSE,
    mrbayes_output_file = "mrbayes_run.nexus"
)
```

**Arguments**

**constraint** The constraint tree: a phylo object or a newick character string, with or without branch lengths.

**taxa** A character vector with taxon names to be maintained in tree

**ncalibration** The node calibrations: a phylo object with branch lengths proportional to time; in this case all nodes from ncalibration will be used as calibration points. Alternatively, a list with two elements: the first is a character vector with node names from phy to calibrate; the second is a numeric vector with the corresponding ages to use as calibrations.
The `make_otol_associations` function is used to associate Open Tree of Life authors with studies.

**Description**

Associate Open Tree of Life authors with studies.

**Usage**

```r
make_otol_associations()
```

**Value**

A `data.frame` with author last name, author first and other names, and comma delimited URLs for OToL studies.
make_overlap_table  Create an overlap table

Description

Create an overlap table

Usage

make_overlap_table(results_table)

Arguments

results_table  An "author.results" or "curator.results" data.frame

Value

A data.frame with information on curators and what clades they've worked on

make_sdm  Make a Super Distance Matrix (SDM) from a list of good matrices obtained with get_goodmatrices()

Description

Make a Super Distance Matrix (SDM) from a list of good matrices obtained with get_goodmatrices()

Usage

make_sdm(unpadded.matrices, weighting = "flat")

Arguments

unpadded.matrices  A list of patristic matrices, a datelifeResult object.

weighting  A character vector indicating how much weight to give to each tree in input during the SDM analysis. Options are:

weighting = "flat"  All trees have equal weighting.
weighting = "taxa"  Weight is proportional to number of taxa.
weighting = "inverse"  Weight is proportional to 1 / number of taxa.

Defaults to weighting = "flat".

Value

A matrix.
**make_treebase_associations**

*Associate TreeBase authors with studies*

**Description**

Associate TreeBase authors with studies

**Usage**

```r
make_treebase_associations()
```

**Value**

data.frame with author last name, author first and other names, and comma delimited URLs for TreeBase studies

---

**make_treebase_cache**  
*Create a cache from TreeBase*

**Description**

Create a cache from TreeBase

**Usage**

```r
make_treebase_cache(outputfile = "treebasecache.RData")
```

**Arguments**

- **outputfile**  
  Path including file name

**Value**

List containing author and curator results
map_nodes_ott  

*Add Open Tree of Life Taxonomy to tree nodes.*

**Description**

Add Open Tree of Life Taxonomy to tree nodes.

**Usage**

```r
codes map_nodes_ott(tree)
```

**Arguments**

- `tree`  
  A tree either as a newick character string or as a `phylo` object.

**Value**

A `phylo` object with "nodelabels".

**Examples**

```r
## Not run: # This is a flag for package development. You are welcome to run the example.

# Load the Open Tree chronograms database cached in datelife:
utils:::data(opentree_chronograms)

# Get the small chronograms (i.e., chronograms with less than ten tips) to generate a pretty plot:
small <- opentree_chronograms$trees[unlist(sapply(opentree_chronograms$trees, ape::Ntip)) < 10]

# Now, map the Open Tree taxonomy to the nodes of the first tree
phy <- map_nodes_ott(tree = small[[1]])
# and plot it:
# plot_phylo_all(phy)
library(ape)
plot(phy)
nodelabels(phy$node.label)

## End(Not run) #end dontrun
```
match_all_calibrations

Match calibrations to nodes of a given tree

Description

match_all_calibrations searches a given tree for the most recent common ancestor (mrca) of all taxon name pairs in a datelifeCalibration. It uses phytools::findMRCA().

Usage

match_all_calibrations(phy, calibrations)

Arguments

phy        A phylo object.
calibrations  A calibrations object, an output of extract_calibrations_phylo().

Details

The function takes pairs of taxon names in a secondary calibrations data frame, and looks for them in the vector of tip labels of the tree. If both are present, then it gets the node that represents the most recent common ancestor (mrca) for that pair of taxa in the tree. Nodes of input phy can be named or not.

Value

A list of two elements:

phy  A phylo object with nodes renamed with tree_add_nodelabels().
matched_calibrations  A matchedCalibrations object, which is the input calibrations object with two additional columns storing results from the mrca search with phytools::findMRCA(): $mrca_node_number and $mrca_node_name.

matrices_to_table  Go from a list of patristic distance matrix to a table of node ages

Description

Go from a list of patristic distance matrix to a table of node ages

Usage

matrices_to_table(matrices)
message_multiphylo

Arguments

matrices A names list of patristic distance matrices. Names correspond to the study reference.

Value

A single data.frame of "taxonA", "taxonB", and "age".

matrix_to_table Go from a patristic distance matrix to a node ages table

Description

Go from a patristic distance matrix to a node ages table

Usage

matrix_to_table(matrix, reference)

Arguments

matrix A patristic distance matrix.
reference A character vector with the study reference from where the ages come from.

Value

A data.frame of "taxonA", "taxonB", and "age".

message_multiphylo Message for a multiPhylo input

Description

Message for a multiPhylo input

Usage

message_multiphylo()

Value

A relevant message as a character string.
missing_taxa_check  Checks that missing_taxa argument is ok to be used by make_mrbayes_runfile inside tree_add_dates functions.

Description
Checks that missing_taxa argument is ok to be used by make_mrbayes_runfile inside tree_add_dates functions.

Usage
missing_taxa_check(missing_taxa = NULL, dated_tree = NULL)

Arguments

- **missing_taxa**: A tree, a data frame or a vector enlisting all missing taxa you want to include.
  - **A tree**: Either as a phylo object or as a newick character string. It contains all taxa that you want at the end, both missing and non missing. This tree will be used as a hard constraint.
  - **A data.frame**: It contains two columns named "taxon" and "clade". The first one contains a character vector of missing taxon names. The second one contains a character or numeric vector of nodes from a constraint tree to which each taxon will be assigned.
  - **A character vector**: It contains the names of the missing taxa. They will be added at random to the constraint tree.

- **dated_tree**: a tree (newick or phylo) with branch lengths proportional to absolute time

Value
A phylo object, a newick character string or a dataframe with taxonomic assignations

mrca_calibrations  Identify nodes of a tree topology that are most recent common ancestor (mrca) of taxon pairs from a calibrations object

Description
mrca_calibrations get nodes of a tree topology given in phy that correspond to the most recent common ancestor (mrca) of taxon pairs given in calibrations. It uses phytools::findMRCA() to get mrca nodes.

Usage
mrca_calibrations(phy, calibrations)
opentree_chronograms

Arguments

- **phy**: A phylo object.
- **calibrations**: A calibrations object, an output of `extract_calibrations_phylo()`.

Details

The function takes pairs of taxon names in a calibrations data frame, and looks for them in the vector of tip labels of the tree. If both are present, then it gets the node that represents the most recent common ancestor (mrca) for that pair of taxa in the tree. Nodes of input phy can be named or not. They will be renamed.

Value

A list of two elements:

- **matched_phy**: A phylo object with nodes renamed to match results of the mrca search. Nodes are renamed using `tree_add_nodelabels()`.
- **matched_calibrations**: A matchedCalibrations object, which is the input calibrations object with two additional columns storing results from the mrca search with `phytools::findMRCA()`: `$mrca_node_number` and `$mrca_node_name`.

opentree_chronograms  Chronogram database

Description

Now storing >200 chronograms from Open Tree of Life

Usage

opentree_chronograms

Format

A list of four elements, containing data from Open Tree of Life chronograms

- **authors**: A list of lists of author names of the original studies that published chronograms in the Open Tree of Life database.
- **curators**: A list of lists of curator names that uploaded chronograms to the Open Tree of Life database.
- **studies**: A list of study identifiers.
- **trees**: A multiPhylo object storing the chronograms from Open Tree of Life database.
- **update**: A character vector indicating the time when the database object was last updated.
- **version**: A character vector indicating the datelife `utils::packageVersion()` when the database was last updated.
Details

Generated with 

```r
opentree_chronograms <- get_opentree_chronograms() opentree_chronograms$update <- Sys.time() opentree_chronograms$version <- '2022.01.28' usethis::use_data(opentree_chronograms, overwrite = T, compress = "xz") and updated with update_datelife_cache()
```

Source

http://opentreeoflife.org

Description

`patristic_matrix_array_congruify` is used for `patristic_matrix_array_subset_both` and `patristic_matrix_array_congruify`.

Usage

```r
patristic_matrix_array_congruify(
  patristic_matrix_array,
  taxa,
  phy = NULL,
  dating_method = "PATHd8"
)
```

Arguments

- `patristic_matrix_array`: A patristic matrix array, rownames and colnames must be taxa.
- `taxa`: Vector of taxon names to get a subset for.
- `phy`: A user tree to congruify as phylo object (ape).
- `dating_method`: The method used for tree dating.

Value

A patristic matrix with ages for the target taxa.
patristic_matrix_array_phylo_congruify

Congruify a patristic matrix array from a given phylo object.

Description

Congruify a patristic matrix array from a given phylo object.

Usage

```r
patristic_matrix_array_phylo_congruify(
  patristic_matrix,
  target_tree,
  dating_method = "PATHd8",
  attempt_fix = TRUE
)
```

Arguments

- `patristic_matrix`: A patristic matrix, rownames and colnames must be taxa.
- `target_tree`: A phylo object. Use this in case you want a specific backbone for the output tree.
- `dating_method`: The method used for tree dating.
- `attempt_fix`: Default to TRUE. If congruification results in NA branch lengths, it will attempt to fix them.

Value

A matrix.

patristic_matrix_array_split

Split a patristic matrix array Used inside: patristic_matrix_array_congruify

Description

Split a patristic matrix array Used inside: patristic_matrix_array_congruify

Usage

```r
patristic_matrix_array_split(patristic_matrix_array)
```
Arguments

patristic_matrix_array
A patristic matrix array, rownames and colnames must be taxa.

Value
A patristic matrix 3d array.

Description
Subset a patristic matrix array

Usage
patristic_matrix_array_subset(patristic_matrix_array, taxa, phy4 = NULL)

Arguments

patristic_matrix_array
A patristic matrix array, rownames and colnames must be taxa.
taxa Vector of taxon names to get a subset for.
phy4 A user tree to congruify in phylo4 format (phylobase).

Value
A list with a patristic matrix array and a $problem if any.

Description
Are all desired taxa in the patristic matrix array?

Usage
patristic_matrix_array_subset_both(patristic_matrix_array, taxa, phy = NULL, phy4 = NULL, dating_method = "PATHd8")
Arguments

patristic_matrix_array
A patristic matrix array, rownames and colnames must be taxa.
taxa
Vector of taxon names to get a subset for.
phy
A user tree to congruify as phylo object (ape).
phy4
A user tree to congruify in phylo4 format (phylobase).
dating_method
The method used for tree dating.

Value

A patristic matrix with ages for the target taxa.

patristic_matrix_list_to_array
Convert list of patristic matrices to a 3D array.

Description

patristic_matrix_list_to_array used inside summarize_datelife_result(), patristic_matrix_array_congrui...

Usage

patristic_matrix_list_to_array(patristic_matrix_list, pad = TRUE)

Arguments

patristic_matrix_list
List of patristic matrices
pad
If TRUE, pad missing entries

Value

A 3d array of patristic matrices
**patristic_matrix_MRCA**

`Get time of MRCA from patristic matrix. Used in datelife_result_MRCA().`

**Description**

Get time of MRCA from patristic matrix. Used in `datelife_result_MRCA()`.

**Usage**

```r
patristic_matrix_MRCA(patristic_matrix, na.rm = TRUE)
```

**Arguments**

- `patristic_matrix`
  A patristic matrix (aka a `datelifeResult` object of length 1)
- `na_rm`
  If TRUE, it drops rows containing NAs from the `datelifeResult` patristic matrix; if FALSE, it returns NA where there are missing entries.

**Value**

The depth of the MRCA as a numeric vector.

---

**patristic_matrix_name_order_test**

`Test the name order of a patristic matrix so that row and column labels are in alphabetical order.`

**Description**

`patristic_matrix_name_order_test` is only used in `patristic_matrix_list_to_array()`.

**Usage**

```r
patristic_matrix_name_order_test(
  patristic_matrix,
  standard.rownames,
  standard.colnames
)
```

**Arguments**

- `patristic_matrix`
  A patristic matrix, rownames and colnames must be taxa.
- `standard.rownames`
  A character vector of row names.
- `standard.colnames`
  A character vector of column names.
patristic_matrix_pad

Value

Boolean.

patristic_matrix_name_reorder

Reorder a matrix so that row and column labels are in alphabetical order.

Description

patristic_matrix_name_reorder is only used in: patristic_matrix_pad().

Usage

patristic_matrix_name_reorder(patristic_matrix)

Arguments

patristic_matrix

A patristic matrix, rownames and colnames must be taxa.

Value

A patristic matrix with row and column names for taxa in alphabetical order.

patristic_matrix_pad

Fill in empty cells in a patristic matrix for missing taxa.

Description

Used in: patristic_matrix_list_to_array().

Usage

patristic_matrix_pad(patristic_matrix, all_taxa)

Arguments

patristic_matrix

A patristic matrix, rownames and colnames must be taxa.

all_taxa

A vector of names of all taxa you want, including ones not in the patristic matrix.

Value

A patristic matrix, with NA for entries between taxa where at least one was not in the original patristic matrix.
**patristic_matrix_taxa_all_matching**

*Are all desired taxa in the patristic matrix?*

---

**Description**

`patristic_matrix_taxa_all_matching` is used inside: `results_list_process()`.

**Usage**

`patristic_matrix_taxa_all_matching(patristic_matrix, taxa)`

**Arguments**

- `patristic_matrix`
  
  A patristic matrix, rownames and colnames must be taxa.

- `taxa`
  
  Vector of taxon names to get a subset for.

**Value**

A Boolean.

---

**patristic_matrix_to_newick**

*Convert patristic matrix to a newick string. Used inside: summarize_datelife_result.*

---

**Description**

Convert patristic matrix to a newick string. Used inside: `summarize_datelife_result`.

**Usage**

`patristic_matrix_to_newick(patristic_matrix)`

**Arguments**

- `patristic_matrix`
  
  A patristic matrix

**Value**

A newick string
patristic_matrix_to_phylo

Convert a patristic matrix to a phylo object.

Description

Function `patristic_matrix_to_phylo` is used inside `summarize_datelife_result()`.

Usage

```r
patristic_matrix_to_phylo(
  patristic_matrix,
  clustering_method = "nj",
  fix_negative_brlen = TRUE,
  fixing_method = 0,
  ultrametric = TRUE,
  variance_matrix = NULL
)
```

Arguments

- `patristic_matrix`: A patristic matrix
- `clustering_method`: A character vector indicating the method to construct the tree. Options are:
  - `nj`: Neighbor-Joining method applied with `ape::nj()`.
  - `upgma`: Unweighted Pair Group Method with Arithmetic Mean method applied with `phangorn::upgma()`.
  - `bionj`: An improved version of the Neighbor-Joining method applied with `ape::bionj()`.
  - `triangle`: Triangles method applied with `ape::triangMtd()`.
  - `mvr`: Minimum Variance Reduction method applied with `ape::mvr()`.
- `fix_negative_brlen`: Boolean indicating whether to fix negative branch lengths in resulting tree or not. Default to `TRUE`.
- `fixing_method`: A character vector specifying the method to fix branch lengths: "bladj", "mr-bayes" or a number to be assigned to all branches meeting `fixing_criterion`
- `ultrametric`: Boolean indicating whether to force ultrametric or not.
- `variance_matrix`: A variance matrix from a datelifeResult object, usually an output from `datelife_result_variance_matrix()`. Only used if `clustering_method = "mvr"`.

Details

We might add the option to insert a function as `clustering_method` in the future. Before, we had hard-coded the function to try Neighbor-Joining (NJ) first; if it errors, it will try UPGMA. Now, it uses NJ for a "phylo_all" summary, and we are using our own algorithm to get a tree from a summary matrix.
**patristic_matrix_unpad**

Function to remove missing taxa from a datelifeResult object.

**Description**

Used in `datelife_result_sdm_phylo()`.

**Usage**

```r
patristic_matrix_unpad(patristic_matrix)
```

**Arguments**

- `patristic_matrix`:
  A patristic matrix with row and column names for taxa

**Value**

- patristic_matrix for all_taxa

---

**phylo_check**

Checks if phy is a phylo object and/or a chronogram.

**Description**

Checks if phy is a phylo object and/or a chronogram.

**Usage**

```r
phylo_check(phy = NULL, brlen = FALSE, dated = FALSE)
```

**Arguments**

- `phy`:
  A phylo object.
- `brlen`:
  Boolean. If TRUE it checks if phylo object has branch lengths.
- `dated`:
  Boolean. If TRUE it checks if phylo object is ultrametric.

**Value**

- Nothing
phylo_congruify

*Congruify a reference tree and a target tree given as phylo objects.*

**Description**

Congruify a reference tree and a target tree given as phylo objects.

**Usage**

```r
phylo_congruify(
  reference_tree,
  target_tree,
  dating_method = "PATHd8",
  attempt_fix = TRUE
)
```

**Arguments**

- `reference_tree`: A phylo object.
- `target_tree`: A phylo object. Use this in case you want a specific backbone for the output tree.
- `dating_method`: The method used for tree dating.
- `attempt_fix`: Default to TRUE. If congruification results in NA branch lengths, it will attempt to fix them.

**Value**

A matrix.

---

phylo_generate_uncertainty

*Generate uncertainty in branch lengths using a lognormal.*

**Description**

Generate uncertainty in branch lengths using a lognormal.
**phylo_generate_uncertainty**

**Usage**

```r
phylo_generate_uncertainty(
    phy,
    size = 100,
    uncertainty_method = "other",
    age_distribution = "uniform",
    age_sd = NULL,
    age_var = 0.1,
    age_scale = 0,
    alpha = 0.025,
    rescale = TRUE
)
```

**Arguments**

- **phy** A phylo object.
- **size** A numeric vector indicating the number of samples to be generated.
- **uncertainty_method** A character vector specifying the method to generate uncertainty. mrbayes is default.
- **age_distribution** A character string specifying the type of calibration. Only "fixed" and "uniform" are implemented for now.
  - **fixed** The age given in `ncalibration` will be used as fixed age.
  - **lognormal** The age given in `ncalibration` will be used as mean age. The standard deviation can be provided. # still need to add this option. By default, a 95 CI sd is used.
  - **uniform** The age given in `ncalibration` will be used as mean age. Where min_age = 0.9 * mean age, and max_age = 1.1 * mean age.
- **age_sd** The standard deviation around the age to use for generating the uncertainty. If not a numeric value, var will be used to calculate it.
- **age_var** The variance to calculate age_sd and generate uncertainty.
- **age_scale** How to scale sd by the depth of the node. If 0, same sd for all. If not, older nodes have more uncertainty.
- **alpha** The significance level on uncertainty to generate. By default 0.025
- **rescale** Boolean. If true, observed age will be rescaled each round.

**Details**

If you want to change the size of sampled trees you do not need to run mrbayes again. Just use `sample_trees("mrbayes_trees_file_directory", size = new_size)` and you will get a multiPhylo object with a new tree sample.

**Value**

A phylo or multiPhylo object with the same topology as phy but different branch lengths.
Examples

## Not run:
# Generate uncertainty over feline species SDM chronogram.
# Load the data:

data(felid_sdm)

# By default, generates a sample of 100 trees with var = 0.1:

unc <- phylo_generate_uncertainty(felid_sdm$phy)
length(unc)

# Make an LTT plot:

max_age <- max(sapply(unc, ape::branching.times))
ape::ltt.plot(phy = unc[[1]], xlim = c(-max_age, 0), col = "#cce5ff50")
for (i in 2:100) {
  ape::ltt.lines(phy = unc[[i]], col = "#cce5ff50")
}
ape::ltt.lines(felid_sdm$phy, col = "red")
title(c("fake uncertainty", "in Felidae SDM chronogram"))

## End(Not run) # end dontrun

---

phylo_get_node_numbers

### phylo_get_node_numbers

*Gets node numbers from any phylogeny*

**Description**

Gets node numbers from any phylogeny

**Usage**

phylo_get_node_numbers(phy)

**Arguments**

- **phy**
  A phylo object.

**Value**

A numeric vector with node numbers
**phylo_get_subset_array**

*Get a subset array from a phylo object*

**Description**
Get a subset array from a phylo object

**Usage**

```r
phylo_get_subset_array(
  reference_tree,
  taxa,
  phy4 = NULL,
  dating_method = "PATHd8"
)
```

**Arguments**

- `reference_tree` A phylo object.
- `taxa` Vector of taxon names to get a subset for.
- `phy4` A user tree to congruify in phylo4 format (phylobase).
- `dating_method` The method used for tree dating.

**Value**
A list with a patristic matrix array and a `$problem` if any.

---

**phylo_get_subset_array_congruify**

*Get a congruified subset array from a phylo object*

**Description**
Get a congruified subset array from a phylo object

**Usage**

```r
phylo_get_subset_array_congruify(
  reference_tree,
  taxa,
  phy = NULL,
  dating_method = "PATHd8"
)
```
phylo_prune_missing_taxa

Arguments
reference_tree A phylo object.
taxa Vector of taxon names to get a subset for.
phy A user tree to congruify as phylo object (ape).
dating_method The method used for tree dating.

Value
A list with a patristic matrix array and a $problem if any.

phylo_has_brlen

Check if a tree has branch lengths

Description
Check if a tree has branch lengths

Usage
phylo_has_brlen(phy)

Arguments
phy A phylo object.

Value
A TRUE or FALSE

phylo_prune_missing_taxa

Prune missing taxa from a phylo object Used inside phylo_get_subset_array and phylo_get_subset_array_congruify.

Description
Prune missing taxa from a phylo object Used inside phylo_get_subset_array and phylo_get_subset_array_congruify.

Usage
phylo_prune_missing_taxa(phy, taxa)
**phylo_subset_both**

**Arguments**

- **phy**  
  A user tree to congruify as phylo object (ape).

- **taxa**  
  Vector of taxon names to get a subset for.

**Value**

A phylo object.

---

**phylo_subset_both**  
*Subset a reference and a target tree given as phylo objects.*

**Description**

Subset a reference and a target tree given as phylo objects.

**Usage**

```r
phylo_subset_both(
  reference_tree,
  taxa,
  phy = NULL,
  phy4 = NULL,
  dating_method = "PATHd8"
)
```

**Arguments**

- **reference_tree**  
  A phylo object.

- **taxa**  
  Vector of taxon names to get a subset for.

- **phy**  
  A user tree to congruify as phylo object (ape).

- **phy4**  
  A user tree to congruify in phylo4 format (phylobase).

- **dating_method**  
  The method used for tree dating.

**Value**

A list with a patristic matrix array and a $problem if any.
### phylo_tiplabel_space_to_underscore

*Convert spaces to underscores in trees.*

**Description**

`phylo_tiplabel_space_to_underscore` is used in: `make mrbayes runfile()`, `tree get singleton outgroup()`, `congruify and check()`, `patristic matrix array phylo congruify()`.

**Usage**

```r
phylo_tiplabel_space_to_underscore(phy)
```

**Arguments**

- `phy` A phylo object.

**Value**

A phylo object.

### phylo_tiplabel_underscore_to_space

*Convert underscores to spaces in trees.*

**Description**

`phylo_tiplabel_underscore_to_space` is used inside `patristic matrix array phylo congruify()`, `congruify and check()`.

**Usage**

```r
phylo_tiplabel_underscore_to_space(phy)
```

**Arguments**

- `phy` A phylo object.

**Value**

A phylo object.
**phylo_to_patristic_matrix**

Get a patristic matrix from a phylo object.

---

**Description**

Get a patristic matrix from a phylo object.

**Usage**

```r
phylo_to_patristic_matrix(phy, test = TRUE, tol = 0.01, option = 2)
```

**Arguments**

- `phy`: A phylo object.
- `test`: Default to `TRUE`. Whether to test if `phy` has branch lengths and is ultrametric or not.
- `tol`: Branching time in reference above which secondary constraints will be applied to target.
- `option`: An integer (1 or 2; see details).

**Value**

A patristic matrix.

---

**pick_grove**

Pick a grove in the case of multiple groves in a set of trees.

---

**Description**

Pick a grove in the case of multiple groves in a set of trees.

**Usage**

```r
pick_grove(grove_list, criterion = "taxa", datelife_result)
```

**Arguments**

- `grove_list`: A list of vectors of tree indices. Each element is a grove.
- `criterion`: Defaults to `criterion = "taxa"`. Used for chronogram summarizing, i.e., obtaining a single summary chronogram from a group of input chronograms. For summarizing approaches that return a single summary tree from a group of phylogenetic trees, it is necessary that the latter form a grove, roughly, a sufficiently overlapping set of taxa between trees, see Ané et al. (2009) doi:10.1007/s00026-0090017x. In rare cases, a group of trees can have multiple groves. This argument indicates whether to get the grove with the most trees (`criterion = "trees"`) or the most taxa (`criterion = "taxa"`).
datelife_result

A datelifeResult object. Only needed for criterion = "taxa".

Value

A numeric vector of the elements of the picked grove.

plant_bold_otol_tree  Some plants chronogram

Description

Some plants chronogram

Usage

plant_bold_otol_tree

Format

A phylo object with 6 tips and 5 internal nodes

edge  Integer vector with edge (branch) numbers
tip.label  Character vector with species names of plants
Nnode  Integer vector with the number of nodes
node.label  Character vector with node names
dge.length  Numeric vector with edge (branch) lengths

Details

Generated with make_bold_otol_tree(input = "((Zea mays, Oryza sativa), (Arabidopsis thaliana, (Glycine max, Medicago sativa), Solanum lycopersicum) Penta petalae);") usethis::use_data(plant_bold_otol_tree)

Author(s)

Luna L. Sanchez-Reyes <lsanche7@utk.edu>
Brian O’Meara <bomeara@utk.edu>

Source

http://opentreeoflife.org
http://www.boldsystems.org
problems

Problematic chronograms from Open Tree of Life.

Description
Problematic chronograms from Open Tree of Life.

Usage
problems

Format
A list of trees with unmapped taxa

Details
Before we developed tools to clean and map tip labels for our cached trees we found some trees that were stored with unmapped tip labels we extracted them and saved them to be used for testing functions. Generated with problems <- opentree_chronograms$trees[sapply(sapply(opentree_chronograms$trees, 
"[", "tip.label"), function(x) any(grepl("not.mapped", x)))] usethis::use_data(problems) opentree_chronograms object from commit https://github.com/phylotastic/datelife/tree/be894448f6fc437241cd0916fab4e84ac3e09c6

Source
http://opentreeoflife.org

recover_mrcaott
Get an mrcaott tag from an OpenTree induced synthetic tree and get its name and ott id

Description
Get an mrcaott tag from an OpenTree induced synthetic tree and get its name and ott id

Usage
recover_mrcaott(tag)

Arguments
tag A character vector with the mrca tag

Value
A numeric vector with ott id from original taxon named with the corresponding ott name
relevant_curators_tabulate

Return the relevant curators for a set of studies.

Description

Return the relevant curators for a set of studies.

Usage

relevant_curators_tabulate(results.index, cache = "opentree_chronograms")

Arguments

- **results.index**: A vector from `datelife_result_study_index()` with the indices of the relevant studies.
- **cache**: The cached chronogram database.

Value

A vector with counts of each curator, with names equal to curator names.

results_list_process

Take results_list and process it.

Description

results_list_process is used inside: `get_datelife_result()`

Usage

results_list_process(results_list, taxa = NULL, partial = FALSE)

Arguments

- **results_list**: A list returned from using `get_subset_array_dispatch()` on opentree_chronograms$trees
- **taxa**: Vector of taxon names to get a subset for.
- **partial**: If TRUE, return matrices that have only partial matches.

Value

A list with the patristic.matrices that are not NA.
run

Core function to generate results

Description

Core function to generate results

Usage

run(
    input = c("Rhea americana", "Pterocnemia pennata", "Struthio camelus"),
    format = "citations",
    partial = "yes",
    plot.width = 600,
    plot.height = 600,
    use_tnrs = "no",
    opentree_chronograms = NULL
)

Arguments

input A newick string or vector of taxa
format The output format
partial How to deal with trees that have a subset of taxa in the query
plot.width Width in pixels for output plot
plot.height Height in pixels for output plot
use_tnrs Whether to use OpenTree’s TNRS for the input
opentree_chronograms The list of lists containing the input trees and other info

Value

results in the desired format

run_mrbayes

Runs MrBayes from R

Description

Runs MrBayes from R

Usage

run_mrbayes(mrbayes_output_file = NULL)
Arguments

mrbayes_output_file
A character vector specifying the name of MrBayes run file and outputs (can specify directory too).

Value

A phylo object with the consensus tree. MrBayes output files are stored in the working directory.

Description

Sample trees from a file containing multiple trees. Usually from a bayesian analysis output trees file.

Usage

sample_trees(trees_file, trees_object = NULL, burnin = 0.25, size = 100)

Arguments

trees_file A character vector indicating the name and directory of file with trees to sample.
trees_object An R object containing a list of trees already read into R from a tree file from a bayesian analysis output.
burnin A numeric vector indicating the burnin fraction. It should be a number between 0 and 1. Default to 0.25
size A numeric vector indicating the number of samples to be generated.

Value

A multiPhylo object with a random sample of trees.
some_ants_datelife_result

datelifeResult object of some ants

Description
datelifeResult object of some ants

Usage
    some_ants_datelife_result

Format
    A list of one element, containing a named patristic matrix

Details
    Generated with: some_ants_input <- "(Aulacopone_relicta,(Myrmecia_gulosa,(Aneuretus_simoni,Dolichoderus_mariae)),((Ectatomma_ruidum,Huberia_brounii),Formica_rufa)),Apomyrma_stygia),Martialis_heureka)Formicidae;"
some_ants_datelife_query <- make_datelife_query(input = some_ants_input) some_ants_datelife_result <- get_datelife_result(input = some_ants_datelife_query) usethis::use_data(some_ants_datelife_result)

Source
    http://opentreeoflife.org

subset2_search
    A list with datelifeQuery and datelifeResult objects from a search of taxon names from subset2_taxa

Description
    A list with datelifeQuery and datelifeResult objects from a search of taxon names from subset2_taxa

Usage
    subset2_search

Format
    A list with two named elements. datelifeResult object with 24 patristic matrices

    datelife_query A datelifeQuery object using names_subset_2 as input.
    datelife_result A datelifeResult object resulting from a search of names in datelifeQuery
summarize_congruifiedCalibrations

Details

Generated with: datelife_query <- make_datelife_query(subset2_taxa) datelife_result <- get_datelife_result(datelife_query) subset2_search <- list(query = datelife_query, result = datelife_result) usethis::use_data(subset2_search, overwrite = TRUE)

| subset2_taxa | Long list of >2.7k virus, bacteria, plant and animal taxon names |

Description

Long list of >2.7k virus, bacteria, plant and animal taxon names

Usage

subset2_taxa

Format

A character vector of length 2778

Details

Generated with: subset2_taxa <- rphylotastic::url_get_scientific_names("https://github.com/phylotastic/rphylotastic/blob/master/tests/subset2.txt") usethis::use_data(subset2_taxa)

Source

https://github.com/phylotastic/rphylotastic/tree/master/tests/testthat

summarize_congruifiedCalibrations

Get summary statistics of ages in a congruifiedCalibrations object.

Description

Function summarize_congruifiedCalibrations returns a table of summary statistics for each node in congruified_calibrations argument.

Usage

summarize_congruifiedCalibrations(congruified_calibrations, age_column)
**summarize_datelife_result**

*Summarize a datelifeResult object.*

**Arguments**

congruified_calibrations

A congruifiedCalibrations object, output of `congruify_and_mrca_multiPhylo()`.

age_column

A character string indicating the name of the column to be summarized.

**Value**

A data.frame of summarized ages.

**Description**

Get different types of summaries from a datelifeResult object, an output from `get_datelife_result()`. This allows rapid processing of data. If you need a list of chronograms from your datelifeResult object, this is the function you are looking for.

**Usage**

```r
summarize_datelife_result(
  datelife_result = NULL,
  datelife_query = NULL,
  summary_format = "phylo_all",
  na_rm = TRUE,
  summary_print = c("citations", "taxa"),
  taxon_summary = c("none", "summary", "matrix"),
  criterion = "taxa"
)
```

**Arguments**

datelife_result

A datelifeResult object, usually an output of `get_datelife_result()`.

datelife_query

A datelifeQuery object, usually an output of `make_datelife_query()`.

summary_format

A character vector of length one, indicating the output format for results of the DateLife search. Available output formats are:

"citations" A character vector of references where chronograms with some or all of the target taxa are published (source chronograms).

"mrca" A named numeric vector of most recent common ancestor (mrca) ages of target taxa defined in input, obtained from the source chronograms. Names of mrca vector are equal to citations.

"newick_all" A named character vector of newick strings corresponding to target chronograms derived from source chronograms. Names of newick_all vector are equal to citations.
"newick_sdm" Only if multiple source chronograms are available. A character vector with a single newick string corresponding to a target chronogram obtained with SDM supertree method (Criscuolo et al. 2006).

"newick_median" Only if multiple source chronograms are available. A character vector with a single newick string corresponding to a target chronogram from the median of all source chronograms.

"phylo_sdm" Only if multiple source chronograms are available. A phylo object with a single target chronogram obtained with SDM supertree method (Criscuolo et al. 2006).

"phylo_median" Only if multiple source chronograms are available. A phylo object with a single target chronogram obtained from source chronograms with median method.

"phylo_all" A named list of phylo objects corresponding to each target chronogram obtained from available source chronograms. Names of phylo_all list correspond to citations.

"phylo_biggest" The chronogram with the most taxa. In the case of a tie, the chronogram with clade age closest to the median age of the equally large trees is returned.

"html" A character vector with an html string that can be saved and then opened in any web browser. It contains a 4 column table with data on target taxa: mrca, number of taxa, citations of source chronogram and newick target chronogram.

"data_frame" A 4 column data.frame with data on target taxa: mrca, number of taxa, citations of source chronograms and newick string.

na_rm If TRUE, it drops rows containing NAs from the datelifeResult patristic matrix; if FALSE, it returns NA where there are missing entries.

summary_print A character vector specifying the type of summary information to be printed to screen. Options are:

"citations" Prints references of chronograms where target taxa are found.

"taxa" Prints a summary of the number of chronograms where each target taxon is found.

"none" Nothing is printed to screen. Defaults to c("citations", "taxa"), which displays both.

taxon_summary A character vector specifying if data on target taxa missing in source chronograms should be added to the output as a "summary" or as a presence/absence "matrix". Default to "none", no information on taxon_summary added to the output.

criterion Defaults to criterion = "taxa". Used for chronogram summarizing, i.e., obtaining a single summary chronogram from a group of input chronograms. For summarizing approaches that return a single summary tree from a group of phylogenetic trees, it is necessary that the latter form a groove, roughly, a sufficiently overlapping set of taxa between trees, see Ané et al. (2009) doi:10.1007/s00026-0090017x. In rare cases, a group of trees can have multiple groves. This argument indicates whether to get the groove with the most trees (criterion = "trees") or the most taxa (criterion = "taxa").
**Value**

The output is determined by the argument `summary_format`:

- **If** `summary_format = "citations"` The function returns a character vector of references.
- **If** `summary_format = "mrca"` The function returns a named numeric vector of most recent common ancestor (mrca) ages.
- **If** `summary_format = "newick_[all, sdm, or median]"` The function returns output chronograms as newick strings.
- **If** `summary_format = "phylo_[all, sdm, median, or biggest]"` The function returns output chronograms as phylo or multiPhylo objects.
- **If** `summary_format = "html"` or "data_frame" The function returns a 4 column table with data on mrca ages, number of taxa, references, and output chronograms as newick strings.

**References**


---

**summarize_fossil_range**

*Summarize taxon age from PBDB to just a single min and max age*

**Description**

This uses the Paleobiology Database’s API to gather information on the ages for all specimens of a taxon. It will also look for all descendants of the taxon. It fixes name misspellings if possible. It is basically a wrapper for `get_fossil_range`.

**Usage**

```r
summarize_fossil_range(taxon, recent = FALSE, assume_recent_if_missing = TRUE)
```

**Arguments**

- **taxon** The scientific name of the taxon you want the range of occurrences of
- **recent** If TRUE, forces the minimum age to be zero
- **assume_recent_if_missing** If TRUE, any taxon missing from pbdb is assumed to be recent

**Value**

a single row data.frame of max_ma and min_ma for the specimens, with rowname equal to taxon input
summarize_summary_matrix

*Gets all ages per taxon pair from a distance matrix Internal function used in summary_matrix_to_phylo_all().*

**Description**

Gets all ages per taxon pair from a distance matrix Internal function used in summary_matrix_to_phylo_all().

**Usage**

```r
summarize_summary_matrix(summ_matrix)
```

**Arguments**

- **summ_matrix**
  
  Any summary patristic distance matrix, such as the ones obtained with `datelife_result_sdm_matrix()` or `datelife_result_median_matrix()`.

**Value**

A data.frame of pairwise ages, with row number equal to the combinatorial of column names (or row names), estimated as \( ncol(summ_matrix)^2 - \sum(1:(ncol(summ_matrix)-1)) \).

---

summary.datelifeResult

*Summarize a datelifeResult object.*

**Description**

Summarize a datelifeResult object.

**Usage**

```r
## S3 method for class 'datelifeResult'
summary(object, datelife_query, na_rm = TRUE, ...)
```

**Arguments**

- **object**
  
  An object of class datelifeResult, usually an output of `get_datelife_result()`.

- **datelife_query**
  
  A datelifeQuery object, usually an output of `make_datelife_query()`.

- **na_rm**
  
  Default to TRUE, whether to include partial matches or not.

- **...**
  
  Further arguments passed to or from other methods.
Value

A named list of 11 elements:

"citations" A character vector of references where chronograms with some or all of the target taxa are published (source chronograms).

"mrca" A named numeric vector of most recent common ancestor (mrca) ages of target taxa defined in input, obtained from the source chronograms. Names of mrca vector are equal to citations.

"newick_all" A named character vector of newick strings corresponding to target chronograms derived from source chronograms. Names of newick_all vector are equal to citations.

"newick_sdm" Only if multiple source chronograms are available. A character vector with a single newick string corresponding to a target chronogram obtained with SDM supertree method (Criscuolo et al. 2006).

"newick_median" Only if multiple source chronograms are available. A character vector with a single newick string corresponding to a target chronogram from the median of all source chronograms.

"phylo_sdm" Only if multiple source chronograms are available. A phylo object with a single target chronogram obtained with SDM supertree method (Criscuolo et al. 2006).

"phylo_median" Only if multiple source chronograms are available. A phylo object with a single target chronogram obtained from source chronograms with median method.

"phylo_all" A named list of phylo objects corresponding to each target chronogram obtained from available source chronograms. Names of phylo_all list correspond to citations.

"phylo_biggest" The chronogram with the most taxa. In the case of a tie, the chronogram with clade age closest to the median age of the equally large trees is returned.

"html" A character vector with an html string that can be saved and then opened in any web browser. It contains a 4 column table with data on target taxa: mrca, number of taxa, citations of source chronogram and newick target chronogram.

"data_frame" A 4 column data.frame with data on target taxa: mrca, number of taxa, citations of source chronograms and newick string.

---

**summary.matchedCalibrations**

*Summarize a matchedCalibrations object*

**summary.matchedCalibrations** gets the node age distribution from a matchedCalibrations object.

---

Description

Summarize a matchedCalibrations object **summary.matchedCalibrations** gets the node age distribution from a matchedCalibrations object.

Usage

```r
## S3 method for class 'matchedCalibrations'
summary(object, ...)
```
summary_matrix_to_phylo

Go from a summary matrix to an ultrametric phylo object.

Arguments

summ_matrix

Any summary patristic distance matrix, such as the ones obtained with `datelife_result_sdm_matrix()` or `datelife_result_median_matrix()`.

datelife_query

A datelifeQuery object, usually an output of `make_datelife_query()`.

target_tree

A phylo object. Use this in case you want a specific backbone for the output tree.

Details

Columns `in_phy$mrca_node_name` and `in_phy$reference` are factors.

Value

A `summaryMatchedCalibrations` object, which is a list of two `matchedCalibrations` objects:

- **not_in_phy** A data.frame subset of input `matchedCalibrations` object containing taxon name pairs that were not present in the given tree. NULL if all input taxon names are found in the given tree.
- **in_phy** A data.frame subset of input `matchedCalibrations` object containing all taxon name pairs that were present in the given tree.

Usage

```r
summary_matrix_to_phylo(
  summ_matrix,
  datelife_query = NULL,
  target_tree = NULL,
  total_distance = TRUE,
  use = "mean",
  ...
)
```
summary_matrix_to_phylo_all

Arguments passed on to summary_matrix_to_phylo_all

Details

It can take a regular patristic distance matrix, but there are simpler methods for that implemented in patristic_matrix_to_phylo().

Value

An ultrametric phylo object.
Arguments

summ_matrix Any summary patristic distance matrix, such as the ones obtained with `datelife_result_sdm_matrix()` or `datelife_result_median_matrix()`.
datelife_query A datelifeQuery object, usually an output of `make_datelife_query()`.
target_tree A phylo object. Use this in case you want a specific backbone for the output tree.
total_distance Whether the input `summ_matrix` stores total age distance (from tip to tip) or distance from node to tip. Default to `TRUE`, divides the matrix in half, if `FALSE` it will take it as is.
...
... Arguments passed on to `get_otol_synthetic_tree` otol_version Version of Open Tree of Life to use resolve Defaults to `TRUE`. Whether to resolve the tree at random or not.
input Optional. A character vector of names or a datelifeQuery object.
ott_ids If not NULL, it takes this argument and ignores input. A numeric vector of ott ids obtained with `rotl::taxonomy_taxon_info()` or `rotl::tnrs_match_names()` or `tnrs_match()`.

Details

With this function users can choose the minimum, mean or maximum ages from the summary matrix as calibration points to get a single summary chronogram. Users get all three summary chronograms in a `multiPhylo` object.

Value

A `multiPhylo` object of length 5. It contains min, mean, median, midpoint, and max summary chronograms.

summary_patristic_matrix_array

Summarize patristic matrix array (by default, median). Used inside: `summarize_datelife_result`.

Description

Summarize patristic matrix array (by default, median). Used inside: `summarize_datelife_result`.

Usage

`summary_patristic_matrix_array(patristic_matrix_array, fn = stats::median)`

Arguments

patristic_matrix_array

3D array of patristic matrices

fn The function to use to summarize
**threebirds_dr**

**Value**

A 2d array with the median (or max, or mean, etc) of the input array

| threebirds_dr | datelifeResult object of three birds "Rhea americana", "Pterocnemia pennata", and "Struthio camelus" |

**Description**

datelifeResult object of three birds "Rhea americana", "Pterocnemia pennata", and "Struthio camelus"

**Usage**

threebirds_dr

**Format**

A list of 9 named patristic matrix

**Details**

Generated with: threebirds_dr <- get_datelife_result(input=c("Rhea americana", "Pterocnemia pennata", "Struthio camelus"), partial = TRUE, use_tnrs = FALSE, approximate_match = TRUE, cache = "opentree_chronograms") use_data(threebirds_dr)

**Source**

http://opentreeoflife.org

| tnrs_match | Taxon name resolution service (tnrs) applied to a vector of names by batches |

**Description**

Taxon name resolution service (tnrs) applied to a vector of names by batches

**Usage**

tnrs_match(input, reference_taxonomy, tip, ...)

## Default S3 method:

```
tnrs_match(input, reference_taxonomy = "ott", ...)  
```

## S3 method for class 'phylo'

```
tnrs_match(input, reference_taxonomy = "ott", tip = NULL, ...)  
```
Arguments

input A character vector of taxon names, or a phylo object with tip names, to be matched to taxonomy.

reference_taxonomy A character vector specifying the reference taxonomy to use for TNRS. Options are "ott", "ncbi", "gbif" or "irmng". The function defaults to "ott".

tip A vector of mode numeric or character specifying the tips to match. If left empty all tips will be matched.

... Arguments passed on to `rotl::tnrs_match_names`

context_name name of the taxonomic context to be searched (length-one character vector or NULL). Must match (case sensitive) one of the values returned by `tnrs_contexts`. Default to "All life".

do_approximate_matching A logical indicating whether or not to perform approximate string (a.k.a. “fuzzy”) matching. Using FALSE will greatly improve speed. Default, however, is TRUE.

ids A vector of ids to use for identifying names. These will be assigned to each name in the names array. If ids is provided, then ids and names must be identical in length.

include_suppressed Ordinarily, some quasi-taxa, such as incertae sedis buckets and other non-OTUs, are suppressed from TNRS results. If this parameter is true, these quasi-taxa are allowed as possible TNRS results.

Details

There is no limit to the number of names that can be queried and matched. The output will preserve all elements from original input phylo object and will add

`phy$_mapped` A character vector indicating the state of mapping of `phy$tip.labels`:

- `original` Tnrs matching was not attempted. Original labeling is preserved.
- `ott` Tnrs matching was manually made by a curator in Open Tree of Life.
- `tnrs` Tnrs matching was attempted and successful with no approximate matching. Original label is replaced by the matched name.
- `approximated` Tnrs matching was attempted and successful but with approximate matching. Original labeling is preserved.
- `unmatched` Tnrs matching was attempted and unsuccessful. Original labeling is preserved.

`phy$original.tip.label` A character vector preserving all original labels.

`phy$ott_ids` A numeric vector with ott id numbers of matched tips. Unmatched and original tips will be NaN.

If tips are duplicated, tnr will only be run once (avoiding increases in function running time) but the result will be applied to all duplicated tip labels.

Value

An object of class data frame or phylo, with the added class `match_names`.

NULL

NULL
**Examples**

```r
tnrs_match(input = c("Mus"))
tnrs_match(input = c("Mus", "Mus musculus"))
tnrs_match(input = c("Mus", "Echinus", "Hommo", "Mus"))
```

**Description**

Information on contributors, authors, study ids and clades from studies with chronograms in Open tree of Life

**Usage**

```r
treebase_cache
```

**Format**

A list of five data sets

- **tb.author.pretty** A dataframe with two elements: author names and number of studies in TreeBase authored by each
- **tb.author.results** A dataframe with two elements: author names and study identifiers

**Details**

Generated with `make_treebase_cache()`

**Source**

TreeBASE database, no longer available online [https://en.wikipedia.org/wiki/TreeBASE](https://en.wikipedia.org/wiki/TreeBASE)

**tree_add_dates**

Add missing taxa to a dated tree and fabricate node ages for these missing taxa.

**Description**

This function adds missing taxa to a chronogram given in `dated_tree`. It is still work in progress.
Usage

tree_add_dates(
  dated_tree = NULL,
  missing_taxa = NULL,
  dating_method = "mrbayes",
  adding_criterion = "random",
  mrbayes_output_file = "mrbayes_tree_add_dates.nexus"
)

Arguments

dated_tree  a tree (newick or phylo) with branch lengths proportional to absolute time
missing_taxa A tree, a data frame or a vector enlisting all missing taxa you want to include.
  A tree  Either as a phylo object or as a newick character string. It contains all taxa that you want at the end, both missing and non missing. This tree will be used as a hard constraint.
  A data.frame It contains two columns named "taxon" and "clade". The first one contains a character vector of missing taxon names. The second one contains a character or numeric vector of nodes from a constraint tree to which each taxon will be assigned.
  A character vector  It contains the names of the missing taxa. They will be added at random to the constraint tree.

dating_method  The method used for tree dating, options are "mrbayes" and "bladj".
adding_criterion Only used when dating_method = "mrbayes". A character vector to specify how missing_taxa should be added to dated_tree. Choose one of:
  adding_method = "random"  missing_taxa will be added at random to dated_tree.
  adding_method = "taxonomy"  taxa will be added to dated_tree following a dataframe with taxonomic assignations given in missing_taxa argument. If no dataframe is given, OpenTree’s reference taxonomy will be used.
  adding_method = "tree"  taxa will be added to dated_tree following a tree given in missing_taxa argument. If no tree is given, OpenTree’s synthetic tree will be used.

mrbayes_output_file  A character vector specifying the name of mrbayes run file and outputs (can specify directory too).

Value

A phylo object.
**tree_add_nodelabels**  
*Adds labels to nodes with no assigned label*

**Description**  
Adds labels to nodes with no assigned label

**Usage**

```r
tree_add_nodelabels(tree = NULL, node_prefix = "n", node_index = "node_number")
```

**Arguments**

- **tree**: A tree either as a newick character string or as a `phylo` object.
- **node_prefix**: Character vector. If length 1, it will be used to name all nodes with no labels, followed by a number which can be the node_number or consecutive, as specified in node_index.
- **node_index**: Character vector. Choose between "from_1" and "node_number" as numeric index for node labels. It will use consecutive numbers from 1 to total node number in the first case and phylo node numbers in the second case (i.e, from Ntip + 1).

**Value**

A `phylo` object

---

**tree_add_outgroup**  
*Function to add an outgroup to any phylogeny, in phylo or newick format*

**Description**

Function to add an outgroup to any phylogeny, in phylo or newick format

**Usage**

```r
tree_add_outgroup(tree = NULL, outgroup = "outgroup")
```

**Arguments**

- **tree**: A tree either as a newick character string or as a `phylo` object.
- **outgroup**: A character vector with the name of the outgroup. If it has length>1, only first element will be used.

**Value**

A `phylo` object with no root edge.
**tree_check**

Checks if a tree is a phylo class object otherwise it uses input_process. Additionally it can check if tree is a chronogram with phylo_check.

**Usage**

```r
tree_check(tree = NULL, ...)
```

**Arguments**

- `tree` A tree either as a newick character string or as a phylo object.
- `...` Arguments passed on to `phylo_check`.
  - `brlen` Boolean. If TRUE it checks if phylo object has branch lengths.
  - `dated` Boolean. If TRUE it checks if phylo object is ultrametric.

**Value**

If tree is correctly formatted, it returns a phylo object.

**tree_fix_brlen**

Take a tree with branch lengths and fix negative or zero length branches.

**Description**

Take a tree with branch lengths and fix negative or zero length branches.

**Usage**

```r
tree_fix_brlen(
    tree = NULL,
    fixing_criterion = "negative",
    fixing_method = 0,
    ultrametric = TRUE
)
```
Arguments

- **tree**: A tree either as a newick character string or as a *phylo* object.
- **fixing_criterion**: A character vector specifying the type of branch length to be fixed: "negative" or "zero" (the number 0 is also allowed).
- **fixing_method**: A character vector specifying the method to fix branch lengths: "bladj", "mr-bayes" or a number to be assigned to all branches meeting **fixing_criterion**.
- **ultrametric**: Boolean indicating whether to force ultrametric or not.

Value

A *phylo* object with no negative or zero branch lengths.

Description

This uses the taxize package’s wrapper of the Global Names Resolver to get taxonomic paths for the vector of taxa you pass in. Sources is a vector of source labels in order (though it works best if everything uses the same taxonomy, so we recommend doing just one source). You can see options by doing taxize::gnr_datasources(). Our default is Catalogue of Life. The output is a *phylo* object (typically with many singleton nodes if collapse_singles is FALSE: nodes with only one descendant (like "Homo" having "Homo sapiens" as its only descendant) but these singletons typically have node.labels

Usage

```r
tree_from_taxonomy(
  taxa,           sources = "Catalogue of Life",
  collapse_singles = TRUE
)
```

Arguments

- **taxa**: Vector of taxon names
- **sources**: Vector of names of preferred sources; see taxize::gnr_datasources(). Currently supports 100 taxonomic resources, see details.
- **collapse_singles**: If true, collapses singleton nodes

Value

A list containing a *phylo* object with resolved names and a vector with unresolved names
Examples

```r
## Not run: # This is a flag for package development. You are welcome to run the example.

taxa <- c(
  "Homo sapiens", "Ursus arctos", "Pan paniscus", "Tyrannosaurus rex",
  "Ginkgo biloba", "Vulcan", "Klingon"
)
results <- tree_from_taxonomy(taxa)
print(results$unresolved) # The taxa that do not match
ape::plot.phylo(results$phy) # may generate warnings due to problems with singletons
ape::plot.phylo(ape::collapse.singles(results$phy), show.node.label = TRUE)
# got rid of singles, but this also removes a lot of the node.labels

## End(Not run) # end dontrun
```

---

tree_get_node_data

Get node numbers, node names, descendant tip numbers and labels of nodes from any tree, and node ages from dated trees.

Description

Get node numbers, node names, descendant tip numbers and labels of nodes from any tree, and node ages from dated trees.

Usage

```r
tree_get_node_data(
  tree = NULL,
  nodes = NULL,
  node_data = c("node_number", "node_label", "node_age", "descendant_tips_number",
                "descendant_tips_label")
)
```

Arguments

- **tree**: A tree either as a newick character string or as a phylo object.
- **nodes**: Numeric vector with node numbers from which you want to obtain data. Default to NULL: obtain data for all nodes in the tree.
- **node_data**: A character vector containing one or all from: "node_number", "node_label", "node_age", "descendant_tips_number", "descendant_tips_label"

Value

A list
Description
Identify the presence of a single lineage outgroup in a phylogeny

Usage

tree_get_singleton_outgroup(tree = NULL)

Arguments

tree A tree either as a newick character string or as a phylo object.

Value
A character vector with the name of the single lineage outgroup. Returns NA if there is none.

tree_node_tips To get tip numbers descending from any given node of a tree

Description
To get tip numbers descending from any given node of a tree

Usage

tree_node_tips(tree = NULL, node = NULL, curr = NULL)

Arguments

tree a phylogenetic tree as an object of class "phylo".
node an integer specifying a node number in the tree.
curr the set of previously stored node numbers - used in recursive function calls.

Value
A numeric vector with tip numbers descending from a node
update_all_cached  

Update all data files as data objects for the package

Description
This includes opentree chronograms, contributors, treebase and curators. For speed, datelife caches chronograms and other information. Running this (within the checked out version of datelife) will refresh these. Then git commit and git push them back.

Usage
update_all_cached()

Value
None

update_datelife_cache  

Create an updated OpenTree chronograms database object

Description
The function calls `get_opentree_chronograms()` to update the OpenTree chronograms database cached in datelife. It has the option to write the updated object as an .Rdata file, that will be independent of the `opentree_chronograms` data object that you can load with `data("opentree_chronograms", package = "datelife")`.

Usage
update_datelife_cache(
  write = TRUE,  
  updated_name = "opentree_chronograms_updated",  
  file_path = file.path(tempdir()),  
  ...  
)

Arguments
write  
Defaults to TRUE, it saves an .Rdata file named indicated by argument name, containing available chronograms from Open Tree of Life. Saves to path indicated by argument path.

updated_name  
Used if write = TRUE. Defaults to "opentree_chronograms_updated". A character vector of length one indicating the name to assign to both the updated OpenTree chronogram database object and the "Rdata" file. For example, if name = "my_database", the function will assign the updated chronogram database to an object named my_database and will write it to a file named "my_database.Rdata" in the path indicated by argument file_path.
use_all_calibrations

file_path  Used if write = TRUE. A character vector of length 1 indicating the path to write the updated database ".Rdata" file to, excluding file name. Defaults to temporary directory obtained with base::tempdir() and formatted with base::file.path().

...  Arguments passed on to get_opentree_chronograms

max_tree_count  Default to "all", it gets all available chronograms. For testing purposes, a numeric value indicating the max number of trees to be cached.

Value

A list of 4 elements:

- **authors**  A list of lists of author names of the original studies that published chronograms currently stored in the Open Tree of Life database.
- **curators**  A list of lists of curator names that uploaded chronograms to the Open Tree of Life database.
- **studies**  A list of study identifiers from original studies that published chronograms currently stored in the Open Tree of Life database.
- **trees**  A multiPhylo object storing the chronograms from Open Tree of Life database.
- **update**  A character vector indicating the time when the database object was last updated.
- **version**  A character vector indicating the datelife package version when the object was last updated.

Description

use_all_calibrations generates one or multiple chronograms (i.e., phylogenetic trees with branch lengths proportional to time) by dating a tree topology given in phy, and secondary calibrations given in calibrations, using the algorithm specified in the argument dating_method.

Usage

use_all_calibrations(
    phy = NULL,
    calibrations = NULL,
    each = FALSE,
    dating_method = "bladj",
    ...
)
**Arguments**

- **phy**
  A phylo object to use as tree topology.

- **calibrations**
  A calibrations object, an output of `get_all_calibrations()`.

- **each**
  Boolean, default to FALSE: all calibrations are returned in the same data.frame. If TRUE, calibrations from each chronogram are returned in separate data frames.

- **dating_method**
  Tree dating algorithm to use. Options are "bladj" or "pathd8" (Webb et al., 2008, doi:10.1093/bioinformatics/btn358; Britton et al., 2007, doi:10.1080/10635150701613783).

- **...**
  Arguments passed on to `use_calibrations`

- **type**
  The type of age to use as calibration. Options are "median", "mean", "min", or "max".

**Details**

If phy has no branch lengths, dating_method is ignores, and the function applies secondary calibrations to date the tree with the BLADJ algorithm. See `make_bladj_tree()` and `use_calibrations_bladj()`.

If phy has branch lengths, the function can use the PATHd8 algorithm. See `use_calibrations_pathd8()`.

**Value**

A phylo or multiPhylo object with branch lengths proportional to time.

**More**

The output object stores the used calibrations and dating_method as attributes(output)$datelife_calibrations and attributes(output)$dating_method.

**References**


---

**Date a given tree topology using a combined set of given calibrations**

**Description**

`use_calibrations` combines all given calibrations and uses them as constraints to perform a dating analysis on a given tree topology, using BLADJ if it has no branch lengths, or PATHd8 if the given tree topology has initial branch lengths.
use_calibrations_bladj

Usage

use_calibrations(
    phy = NULL,
    calibrations = NULL,
    dating_method = "bladj",
    type = "median",
    ...
)

Arguments

phy
A phylo object to use as tree topology.

calibrations
A calibrations object, an output of get_all_calibrations()

dating_method
Tree dating algorithm to use. Options are "bladj" or "pathd8" (Webb et al., 2008, doi:10.1093/bioinformatics/btn358; Britton et al., 2007, doi:10.1080/10635150701613783).

type
The type of age to use as calibration. Options are "median", "mean", "min", or "max".

... Arguments passed on to use_calibrations_pathd8

expand How much to expand by each step to get consistent calibrations. Should be between 0 and 1.
giveup How many expansions to try before giving up

Details

If phy has no branch lengths, dating_method is ignores, and the function applies secondary calibrations to date the tree with the BLADJ algorithm. See make_bladj_tree() and use_calibrations_bladj(). If phy has branch lengths, the function can use the PATHd8 algorithm. See use_calibrations_pathd8().

Value

A phylo object with branch lengths proportional to time.

More

The output object stores the used calibrations and dating_method as attributes(output)$datelife_calibrations and attributes(output)$dating_method.

use_calibrations_bladj

Use calibrations to date a topology with the BLADJ algorithm.

Description

The function use_calibrations_bladj prepares the input for BLADJ and calls make_bladj_tree().
Usage

use_calibrations_bladj(phy = NULL, calibrations, type = "median", root_age)

Arguments

- **phy**: A phylo object with or without branch lengths.
- **calibrations**: A data.frame of secondary calibrations for any pair of taxon names in phy, usually obtained with `get_all_calibrations()`.
- **type**: The type of age to use as calibration. Options are "median", "mean", "min", or "max".
- **root_age**: Numeric specifying the age of the root. Only used if there are no ages for the root node in calibrations argument. If missing, NULL, or not numeric, the value of the oldest calibration plus one unit of the mean differences across calibrations, will be used as root calibration. If there is one single age point provided as calibrations, the root age will be set to 10% more than the age of the single calibration.

Details

The BLADJ algorithm is part of the Phylocom software, presented in Webb et al. (2008) doi:10.1093/bioinformatics/btn358.

Value

A chronogram: a phylo object with branch lengths proportional to time.

References

use_calibrations_each

Arguments

calibrations A data.frame of secondary calibrations for any pair of taxon names in phy, usually obtained with get_all_calibrations().

type The type of age to use as calibration. Options are "median", "mean", "min", or "max".

root_age Numeric specifying an age for the root, provided by the user. Only used if there are no time calibrations for the root node in the chronograms database. If NULL or not numeric, the maximum calibration age plus one unit of the sd (calculated with stats::sd()) of all node ages available for the tree will be used as root calibration. If there is only one calibration available for the whole tree, the root node age will be proportional to 1.1 of the age of that calibration.

Details

The BLADJ algorithm is part of the Phylocom software, presented in Webb et al. (2008) doi:10.1093/bioinformatics/btn358.

Value

A phylo object with branch lengths proportional to time.

References


Description

use_calibrations_each wraps use_calibrations to take each set of given calibrations and use it independently as constraints for BLADJ or PATHd8 to date a given tree topology.

Usage

use_calibrations_each(phy = NULL, calibrations = NULL, ...)

Arguments

phy A phylo object to use as tree topology.

calibrations A calibrations object, an output of get_all_calibrations().

... Arguments passed on to use_calibrations
use_calibrations_pathd8

Tree dating algorithm to use. Options are "bladj" or "pathd8" (Webb et al., 2008, doi:10.1093/bioinformatics/btn358; Britton et al., 2007, doi:10.1080/10635150701613783).

type The type of age to use as calibration. Options are "median", "mean", "min", or "max".

Details

If phy has no branch lengths, dating_method is ignored, and the function applies secondary calibrations to date the tree with the BLADJ algorithm. See make_bladj_tree() and use_calibrations_bladj(). If phy has branch lengths, the function can use the PATHd8 algorithm. See use_calibrations_pathd8().

Value

A multiPhylo object of trees with branch lengths proportional to time.

More

The output object stores the used calibrations and dating_method as attributes(output)$datelife_calibrations and attributes(output)$dating_method.

Description

use_calibrations_pathd8 uses secondary calibrations to date a tree with initial branch lengths using PATHd8.

Usage

use_calibrations_pathd8(
  phy = NULL,
  calibrations = NULL,
  expand = 0.1,
  giveup = 100
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>phy</td>
<td>A phylo object with branch lengths.</td>
</tr>
<tr>
<td>calibrations</td>
<td>A data.frame of secondary calibrations for any pair of taxon names in phy, usually obtained with get_all_calibrations().</td>
</tr>
<tr>
<td>expand</td>
<td>How much to expand by each step to get consistent calibrations. Should be between 0 and 1.</td>
</tr>
<tr>
<td>giveup</td>
<td>How many expansions to try before giving up</td>
</tr>
</tbody>
</table>
use_calibrations_treePL

Details
This function implements the PATHd8 algorithm described in Britton et al. (2007) doi:10.1080/10635150701613783, with geiger::PATHd8.phylo(). The function first attempts to use the given calibrations as fixed ages. If that fails (often due to conflict between calibrations), it will expand the range of the minimum age and maximum age and try again. And repeat. If expand = 0, it uses the summarized calibrations. In some cases, it returns edge lengths in relative time (with maximum tree depth = 1) instead of absolute time, as given by calibrations. In this case, the function returns NA. This is an issue from PATHd8.

Value
A phylo object with branch lengths proportional to time.

References

use_calibrations_treePL

Date a tree with initial branch lengths with treePL.

Description
Date a tree with initial branch lengths with treePL.

Usage
use_calibrations_treePL(phy, calibrations)

Arguments
phy A phylo object with or without branch lengths.
calibrations A data.frame of secondary calibrations for any pair of taxon names in phy, usually obtained with get_all_calibrations().

Details
This function uses treePL as described in Smith, S. A., & O’Meara, B. C. (2012). doi:10.1093/bioinformatics/bts492, with the function treePL.phylo. It attempts to use the calibrations as fixed ages. If that fails (often due to conflict between calibrations), it will expand the range of the minimum age and maximum age and try again. And repeat. If expand = 0, it uses the summarized calibrations. In some cases, it returns edge lengths in relative time (with maximum tree depth = 1) instead of absolute time, as given by calibrations. In this case, the function returns NA. This is an issue from PATHd8.
Value

A phylo object

References

Index

* animal
  subset2_taxa, 96
* ants
  some_ants_datelife_result, 95
  threebirds_dr, 105
* author
  contributor_cache, 18
  treebase_cache, 107
* bacteria
  subset2_taxa, 96
* bold
  plant_bold_otol_tree, 90
* chronogram
  contributor_cache, 18
  felid_gdr_phylo_all, 33
  felid_sdm, 34
  opentree_chronograms, 72
  plant_bold_otol_tree, 90
  problems, 91
  some_ants_datelife_result, 95
  threebirds_dr, 105
  treebase_cache, 107
* count
  treebase_cache, 107
* datelifeResult
  birds_and_cats, 6
  subset2_search, 95
* datelife
  some_ants_datelife_result, 95
  threebirds_dr, 105
* dates
  opentree_chronograms, 72
* felidae
  felid_gdr_phylo_all, 33
  felid_sdm, 34
* id
  treebase_cache, 107
* million
  opentree_chronograms, 72
* myrs
  opentree_chronograms, 72
* names
  subset2_search, 95
  subset2_taxa, 96
* opentree
  opentree_chronograms, 72
* otol
  contributor_cache, 18
  felid_gdr_phylo_all, 33
  felid_sdm, 34
  plant_bold_otol_tree, 90
  problems, 91
  some_ants_datelife_result, 95
  threebirds_dr, 105
* phylogeny
  opentree_chronograms, 72
* plant
  plant_bold_otol_tree, 90
  subset2_taxa, 96
* sdm
  felid_sdm, 34
* studies
  contributor_cache, 18
* study
  contributor_cache, 18
  treebase_cache, 107
* subset2
  subset2_search, 95
* subset
  felid_gdr_phylo_all, 33
  felid_sdm, 34
  some_ants_datelife_result, 95
  subset2_taxa, 96
  threebirds_dr, 105
* supertree
  felid_sdm, 34
* taxon
  subset2_search, 95
subset2_taxa, 96
* time
    opentree_chronograms, 72
* tnrs
    problems, 91
* treebase
    treebase_cache, 107
* tree
    contributor_cache, 18
    felid_gdr_phylo_all, 33
    felid_sdm, 34
    plant_bold_otol_tree, 90
    problems, 91
    some_ants_datelife_result, 95
    threebirds_dr, 105
    treebase_cache, 107
* unmapped
    problems, 91
* virus
    subset2_taxa, 96
* years
    opentree_chronograms, 72
    .get_ott_lineage, 5
ape::bionj(), 9, 80
ape::chronoMPL(), 59
ape::collapse.singles(), 41
ape::mvr(), 9, 80
ape::nj(), 9, 80
ape::SDM(), 23
ape::triangMtd(), 9, 80

base::file.path(), 115
base::tempdir(), 115
BiocManager::install(), 59
birds_and_cats, 6
build_grove_list, 6
build_grove_matrix, 7
c(), 25, 28, 36, 39, 42, 43, 53, 55, 59, 61, 62
check_conflicting_calibrations, 8
check_ott_input, 8, 41, 48
choose_cluster, 9
classification_paths_from_taxonomy, 10
clean_ott_chronogram, 13
clean_taxon_info_children, 14
clean_tnrs, 15
cluster_patristicmatrix, 16
cluster_patristicmatrix(), 9
congruify_and_check, 16
congruify_and_check(), 88
congruify_and_mrca_multiPhylo, 17
congruify_and_mrca_multiPhylo(), 97
congruify_and_mrca_phylo, 18
congruify_and_mrca_phylo(), 17
contributor_cache, 18
date_with_pbdb, 30
datelife(datelife_search), 25
datelife_authors_tabulate, 19
datelife_calibrations
    (get_all_calibrations), 36
datelife_result_median, 20
datelife_result_median_matrix, 21
datelife_result_median_matrix(), 20, 23,
    100, 102, 104
datelife_result_MRCA, 21
datelife_result_MRCA(), 77
datelife_result_sdm_matrix, 22
datelife_result_sdm_matrix(), 20, 23,
    100, 102, 104
datelife_result_sdm_phylo, 22
datelife_result_sdm_phylo(), 81
datelife_result_study_index, 23
datelife_result_study_index(), 19, 92
datelife_result_variance_matrix, 24
datelife_result_variance_matrix(), 16,
    80
datelife_search, 25
datelife_search(), 17, 18, 36, 39, 40
datelife_use, 28
datelife_use_datelifequery, 29

extract_calibrations_dateliferesult,
    31
extract_calibrations_phylo, 32
extract_calibrations_phylo(), 31, 39, 40,
    69, 72
extract_ott_ids, 32
felid_gdr_phylo_all, 33
felid_sdm, 34
filter_for_grove, 34
force_ultrametric, 35
geiger::congruify.phylo(), 31, 32
geiger::PATHd8.phylo(), 121
get_all_calibrations, 36
INDEX

get_all_calibrations(), 116–121
get_all_descendant_species, 36
get_best_grove, 37
get_biggest_multiphylo, 38
get_bold_data, 38
get_calibrations_datelifequery, 39
get_calibrations_vector, 40
get_dated_otol_induced_subtree, 41
get_datelife_result, 42
get_datelife_result(), 20–22, 24, 51, 52, 56, 92, 97, 100
get_datelife_result_datelifequery, 43
get_fossil_range, 44
get_goodmatrices, 45
get_goodmatrices(), 66
get mrbayes_node_constraints, 45
get_opentree_chronograms, 47
get_opentree_chronograms(), 114
get_opentree_species, 47
get_otol_chronograms(get_opentree_chronograms), 47
get_otol_synthetic_tree, 39, 48, 59, 104
get_otol_synthetic_tree(), 39, 39, 60
get_ott_children, 49
get_ott_children(), 8
get_ott_clade, 50
get_ott_clade(), 8
get_ott_lineage, 51
get_subset_array_dispatch, 51
get_subset_array_dispatch(), 75, 92
get_taxon_summary, 52
get_tnrs_names, 53
get_valid_children, 54
get_valid_children(), 49

input_process, 55
input_process(), 61, 62
is_datelife_query, 55
is_datelife_result_empty, 56
is_good_chronogram, 57
is_n_overlap, 57

make_all_associations, 58
make_bladj_tree, 58
make_bladj_tree(), 30, 116–118, 120
make_bold_otol_tree, 59
make_bold_otol_tree(), 28, 29
make_contributor_cache, 60
make_contributor_cache(), 19
make_datelife_query, 8, 29, 42, 43, 60
make_datelife_query(), 20, 23, 25, 29, 30, 36, 39, 42–44, 52, 59, 97, 100, 102, 104
make_datelife_query2, 61
make mrbayes_runfile, 63
make mrbayes_runfile(), 88
make mrbayes_tree, 64
make otol_associations, 65
make_overlap_table, 66
make_sdm, 66
make_sdm(), 45
make treebase_associations, 67
make treebase_cache, 67
map_nodes_ott, 68
match_all_calibrations, 69
match_all_calibrations(), 102
matrices_to_table, 69
matrix_to_table, 70
max(), 20, 23, 103
mean(), 20, 23, 103
message_multiphylo, 70
min(), 20, 23, 103
missing_taxa_check, 71
mrca_calibrations, 71

opentree_chronograms, 28, 29, 72
paleotree:: timePaleoPhy(), 30
patristic_matrix_array_congruify, 73
patristic_matrix_array_congruify(), 76
patristic_matrix_array_phylo_congruify, 74
patristic_matrix_array_phylo_congruify(), 88
patristic_matrix_array_split, 74
patristic_matrix_array_subset, 75
patristic_matrix_array_subset_both, 75
patristic_matrix_list_to_array, 76
patristic_matrix_list_to_array(), 77, 78
patristic_matrix_MRCA, 77
patristic_matrix_name_order_test, 77
patristic_matrix_name_reorder, 78
patristic_matrix_pad, 78
patristic_matrix_pad(), 78
patristic_matrix_taxa_all_matching, 79
patristic_matrix_to_newick, 79
patristic_matrix_to_phylo, 80
patristic_matrix_to_phylo(), 16, 103
patristic_matrix_unpad, 81
phangorn::acctran(), 59
phangorn::optim.pml(), 59
phangorn::upgma(), 9, 80
phylo_check, 81, 110
phylo_congruity, 82
phylo_generate_uncertainty, 82
phylo_get_node_numbers, 84
phylo_get_subset_array, 85
phylo_get_subset_array_congruity, 85
phylo_has_brlen, 86
phylo_prune_missing_taxa, 86
phylo_subset_both, 87
phylo_tiplabel_space_to_underscore, 88
phylo_tiplabel_underscore_to_space, 88
phylo_to_patristic_matrix, 89
phylocomr::ph_bladj(), 58
phytools::findMRCA(), 17, 18, 69, 71, 72
phytools::force.ultrametric(), 35, 41
pick_grove, 89
plant_bold_otol_tree, 90
problems, 91
recover_mrcaott, 91
relevant_curators_tabulate, 92
results_list_process, 92
results_list_process(), 79
rotl::taxonomy_taxon_info(), 5, 8, 14, 39, 41, 48–51, 54, 59, 104
rotl::tnrs_match_names, 53, 62, 106
rotl::tnrs_match_names(), 8, 15, 25, 29, 39, 41, 42, 44, 48–51, 53, 54, 56, 59, 61, 104
rotl::tol_subtree(), 49
row.names(), 53
run, 93
run_mrBayes, 93
sample_trees, 94
some_ants_datelife_result, 95
stats::median(), 20, 23, 103
stats::sd(), 119
subset2_search, 95
subset2_taxa, 96
summarize_congruifiedCalibrations, 96
summarize_datelife_result, 97
summarize_datelife_result(), 21, 23, 31, 76, 80
summarize_fossil_range, 99
summarize_summary_matrix, 100
summary.datelifeResult, 100
summary.matchedCalibrations, 101
summary_matrix_to_phylo, 20, 23, 102
summary_matrix_to_phylo(), 23
summary_matrix_to_phylo_all, 103, 103
summary_patristic_matrix_array, 104
taxonomy_taxon_info(), 14
treebirds_dr, 105
tnrs_contexts, 53, 62, 106
tnrs_match, 105
tnrs_match(), 8, 15, 25, 29, 39, 41, 42, 44, 48–51, 53, 54, 56, 59, 61, 104
tree_add_dates, 107
tree_add_nodelabels, 109
tree_add_nodelabels(), 69, 72
tree_add_outgroup, 109
tree_check, 110
tree_fix_brlen, 110
tree_from_datelife, 111
tree_get_node_data, 112
tree_get_singleton_outgroup, 113
tree_get_singleton_outgroup(), 88
tree_node_tips, 113
treebase_cache, 107
update_all_cached, 114
update_datelife_cache, 114
use_all_calibrations, 115
use_calibrations, 116, 116, 119
use_calibrations(), 29
use_calibrations_bladj, 117
use_calibrations_bladj(), 30, 116, 117, 120
use_calibrations_bladj.matchedCalibrations, 118
use_calibrations.each, 119
use_calibrations_pathd8, 117, 120
use_calibrations_pathd8(), 30, 116, 117, 120
use_calibrations_treePL, 121
utils::packageVersion(), 72