Package: prepR4pcm (via r-universe)

June 25, 2026

Title Prepare Data and Trees for Phylogenetic Comparative Methods

Version 1.0.0

Description Reconcile species names across datasets and phylogenetic trees for comparative biology workflows. Identifies mismatches due to formatting differences, taxonomic synonymy, and spelling errors. Produces detailed reports documenting how each name was resolved, which taxonomic authority was used, and what remains unresolved. Supports exact matching, name normalisation, synonym resolution via local taxonomic databases, and fuzzy matching for likely typos. Detects taxonomic splits and lumps. For methodological context, see Nakagawa et al. (2026) <doi:10.32942/X2468Z>.

License MIT + file LICENSE

URL https://github.com/itchyshin/prepR4pcm, https://itchyshin.github.io/prepR4pcm/

BugReports https://github.com/itchyshin/prepR4pcm/issues

Date 2026-06-16

Language en-GB

Encoding UTF-8

Depends R (>= 4.1.0)

Imports ape, cli, rlang, tibble

Suggests caper, clootl, digest, dplyr, fishtree, httr2, knitr, MCMCglmm, phytools, piggyback, pkgdown, readr, rgnparser, rmarkdown, rotl, rtrees, spelling, stringr, taxadb, testthat (>= 3.0.0)

LazyData true

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Shinichi Nakagawa [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-7765-5182>), Santiago Ortega
[aut], Ayumi Mizuno [aut], Eduardo S.A. Santos [aut],
Malgorzata Lagisz [aut] (ORCID:
<https://orcid.org/0000-0002-3993-6127>), Bhavya Jain [aut],
Jimuel Jr Celeste [aut], Sergio Poo Hernandez [aut]

Maintainer Shinichi Nakagawa <itchyshin@gmail.com>

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

Date/Publication 2026-06-25 12:36:49 UTC

RemoteUrl https://github.com/cran/prepR4pcm

RemoteRef HEAD

RemoteSha b78fe29addf5d4cf917c17b421e31a2348ed37d8

Contents

`avonet_subset` *AVONET morphological trait data (subset)*

Description

A subset of ~920 bird species from the AVONET database (BirdLife taxonomy), covering 12 passerine families within the Corvoidea and allied clades. Contains morphological measurements and ecological traits.

Usage

```
avonet_subset
```

Format

A data frame with ~920 rows and 16 columns:

Species1 Scientific name (BirdLife taxonomy)

Family1 Family

Order1 Order

Beak.Length_Culmen Beak length from culmen (mm)

Beak.Length_Nares Beak length from nares (mm)

Beak.Width Beak width (mm)

Beak.Depth Beak depth (mm)

Tarsus.Length Tarsus length (mm)

Wing.Length Wing length (mm)

Mass Body mass (g)

Habitat Primary habitat code

Habitat.Density Habitat density code

Migration Migration status

Trophic.Level Trophic level

Trophic.Niche Trophic niche

Primary.Lifestyle Primary lifestyle

Source

Tobias et al. (2022) AVONET: morphological, ecological and geographical data for all birds. *Ecology Letters* 25:581–597. doi:10.1111/ele.13898

`crosswalk_birdlife_birdtree`

BirdLife-BirdTree taxonomy crosswalk

Description

A crosswalk mapping species names between BirdLife International taxonomy and the BirdTree (Jetz et al. 2012) taxonomy. This is useful as a pre-built override table for reconciling datasets that use BirdLife names against phylogenies that use BirdTree names. See `reconcile_crosswalk()` to convert this into an overrides table.

Usage

```
crosswalk_birdlife_birdtree
```

Format

A data frame with ~11,000 rows and 4 columns:

Species1 Species name in BirdLife taxonomy

Species3 Species name in BirdTree taxonomy

Match.type Type of match: `"1BL to 1BT"` (one-to-one), `"Many BL to 1BT"` (lump), `"1BL to many BT"` (split), `"Extinct"`, `"Newly described species"`, `"Invalid taxon"`

Match.notes Additional notes on the match

Source

The crosswalk is distributed as supporting information with the AVONET database release (Tobias et al. 2022). It maps two underlying taxonomies, both of which should be cited if you use the crosswalk in published work — see the references below.

References

Tobias, J.A. et al. (2022) AVONET: morphological, ecological and geographical data for all birds. *Ecology Letters* 25:581–597. doi:10.1111/ele.13898

Jetz, W., Thomas, G.H., Joy, J.B., Hartmann, K. & Mooers, A.O. (2012) The global diversity of birds in space and time. *Nature* 491:444–448. doi:10.1038/nature11631

delhey_subset *Plumage lightness data (subset)*

Description

A subset of ~650 passerine species from Delhey et al. (2019), with plumage lightness measurements and climate variables. Covers species from the same families as avonet_subset that have plumage data. Note that species names use underscores (e.g., `"Corvus_corax"`), making this useful for demonstrating name normalisation.

Usage

```
delhey_subset
```

Format

A data frame with columns:

TipLabel Species name with underscores (tree tip label format)

family Family name

annual_mean_temperature Annual mean temperature at range centroid

annual_precipitation Annual precipitation at range centroid

lightness_male Mean plumage lightness, males

lightness_female Mean plumage lightness, females

Source

Delhey et al. (2019) Reconciling ecogeographical rules: rainfall and temperature predict global colour variation in the largest bird radiation. *Ecology Letters* 22:726–736. doi:10.1111/ele.13233

mammal_amniote_example *Amniote-style mammal life-history sample*

Description

A ~5,000-species sample of mammal life-history records, prepared to mirror the structure of the Amniote Life-History Database. Used by the `db-assembly-workflow_mammals` vignette to demonstrate assembling trait data from multiple sources before reconciling against a phylogenetic tree.

Usage

```
mammal_amniote_example
```

Format

A tibble with ~4,953 rows and 5 columns:

`name` Length-1 character vector. Scientific name (genus species), space-separated. Some rows carry trinomials.

`female_body_mass_g` Numeric. Female adult body mass (g); `NA` when unknown.

`adult_body_mass_g` Numeric. Sex-pooled adult body mass (g); `NA` when unknown.

`litter_or_clutch_size_n` Numeric. Mean offspring per reproductive event; `NA` when unknown.

`litters_or_clutches_per_y` Numeric. Number of reproductive events per year; `NA` when unknown.

Source

Myhrvold et al. (2015) An amniote life-history database to perform comparative analyses with birds, mammals, and reptiles. *Ecology* 96:3109. doi:10.1890/150846R.1

`mammal_pantheria_example`

PanTHERIA-style mammal life-history sample

Description

A ~5,400-species sample of mammal life-history records, prepared to mirror the structure of the PanTHERIA database. Used by the `db-assembly-workflow_mammals` vignette.

Usage

```
mammal_pantheria_example
```

Format

A tibble with ~5,416 rows and 4 columns:

`MSW05_Binomial` Length-1 character vector. Scientific name under MSW3 (Mammal Species of the World 3) taxonomy.

`5-1_AdultBodyMass_g` Numeric. Adult body mass (g); `NA` when unknown.

`15-1_LitterSize` Numeric. Mean litter size; `NA` when unknown.

`16-1_LittersPerYear` Numeric. Litters per year; `NA` when unknown.

Source

Jones et al. (2009) PanTHERIA: a species-level database of life history, ecology, and geography of extant and recently extinct mammals. *Ecology* 90:2648. doi:10.1890/081494.1

mammal_tetrapodtraits_example *TetrapodTraits-style mammal sample*

Description

A ~5,900-species sample of mammal trait records, prepared to mirror the structure of the TetrapodTraits 1.0.0 database. Used by the `db-assembly-workflow_mammals` vignette.

Usage

```
mammal_tetrapodtraits_example
```

Format

A tibble with ~5,911 rows and 3 columns:

`Scientific.Name` Length-1 character vector. Scientific name (genus species), period-separated genus.species column name as in the source release.

`BodyMass_g` Numeric. Body mass (g); `NA` when unknown.

`LitterSize` Numeric. Mean litter size; `NA` when unknown.

Source

Moura et al. (2024) A phylogeny-informed characterisation of global tetrapod traits addresses data gaps and biases. *PLOS Biology* 22:e3002658. doi:10.1371/journal.pbio.3002658

mammal_tree_example *Mammal phylogenetic tree (example)*

Description

A 5,987-tip subset of the Upham, Esselstyn & Jetz (2019) VertLife mammal phylogeny, used by the `db-assembly-workflow_mammals` vignette to demonstrate reconciling species names from multiple trait sources against a tree. Tip labels use underscores (`Genus_species`); 76 tips carry an `X_` prefix, denoting Mesozoic stem-mammal fossils grafted onto the molecular backbone via the Upham et al. "backbone-and-patch" framework.

Usage

```
mammal_tree_example
```

Format

An object of class `phylo` (from the ape package), with 5,987 tips and 5,986 internal nodes.

Details

Source confirmed by Santiago Ortega, who contributed the data, on issue #11.

If you use this tree in published work, please cite Upham et al. (2019) directly. The bundled object is a *subset* used for examples only — for analysis-grade trees, download the full credible set from https://vertlife.org/phylosubsets/.

Source

Upham, N.S., Esselstyn, J.A. & Jetz, W. (2019) Inferring the mammal tree: Species-level sets of phylogenies for questions in ecology, evolution, and conservation. *PLOS Biology* 17(12):e3000494. doi:10.1371/journal.pbio.3000494. Full credible sets at https://vertlife.org/phylosubsets/.

References

Other published mammal phylogenies suitable for comparative analysis (alternatives to Upham et al. 2019):

Faurby, S. & Svenning, J.-C. (2015) A species-level phylogeny of all extant and late Quaternary extinct mammals using a novel heuristic-hierarchical Bayesian approach. *Molecular Phylogenetics and Evolution* 84:14–26. doi:10.1016/j.ympev.2014.11.001

Bininda-Emonds, O.R.P. et al. (2007) The delayed rise of present-day mammals. *Nature* 446:507–512. doi:10.1038/nature05634

---

nesttrait_subset *Nest trait data (subset)*

Description

A subset of ~920 bird species from the global nest trait database (v2), covering the same Corvoidea + allied families as avonet_subset. Contains nest site and structure information.

Usage

```
nesttrait_subset
```

Format

A data frame with columns:

Scientific_name Scientific name (HBW/BirdLife v5 taxonomy)

Order Order

Family Family

Common_name English common name

NestSite_ground Ground nesting (0/1)

NestSite_tree Tree nesting (0/1)

NestSite_nontree Non-tree elevated nesting (0/1)

NestSite_cliff_bank Cliff/bank nesting (0/1)

NestStr_scrape Scrape nest (0/1)

NestStr_platform Platform nest (0/1)

NestStr_cup Cup nest (0/1)

NestStr_dome Dome nest (0/1)

NestStr_primary_cavity Primary cavity nester (0/1)

NestStr_second_cavity Secondary cavity nester (0/1)

Source

Chia et al. (2023) A global database of bird nest traits. *Scientific Data* 10:923. doi:10.1038/s41597-023028371

`pr_cite_tree` *Format the citations for a tree result*

Description

Given a `pr_tree_result` produced by `pr_get_tree()` or `pr_date_tree()`, emit a formatted citation block listing the backend used, the underlying paper(s), and per-tree source citations when the result is a multi-tree posterior. Useful when writing the methods section of a paper or when adding a tree provenance footnote to a figure.

Usage

```
pr_cite_tree(result, format = c("text", "markdown", "bibtex"))
```

Arguments

`result`	A `pr_tree_result` from `pr_get_tree()` or `pr_date_tree()`.
`format`	A length-1 character vector. One of: `"text"` **(default)** Plain-text citation block, suitable for printing or copy-pasting. `"markdown"` GitHub-flavoured markdown with bullets and headings; suitable for issue threads, PR descriptions, and README sections. `"bibtex"` One BibTeX entry per source. Use this to paste into a manuscript bibliography. Note: the entries are hand-rolled from package metadata, not pulled from a canonical bibliographic source — always sanity-check before submission.

Value

A length-1 character vector containing the formatted citation block. The result is also printed (invisibly returned) so calling `pr_cite_tree(res)` on its own at the console shows the block.

See Also

pr_get_tree() / pr_date_tree() for producing the pr_tree_result that this function formats.

Examples

```
# Build a minimal `pr_tree_result` by hand so the three citation
# formats are visible without a network call. In real use this
# object is returned by `pr_get_tree()` or `pr_date_tree()`.
fake_res <- structure(
  list(
    source       = "fishtree",
    tree         = ape::read.tree(text = "(Salmo_salar,Esox_lucius);"),
    backend_meta = list(tree_provenance = list())
  ),
  class = "pr_tree_result"
)

cat(pr_cite_tree(fake_res, format = "text"))      # human-readable
cat(pr_cite_tree(fake_res, format = "markdown"))  # for a README
cat(pr_cite_tree(fake_res, format = "bibtex"))    # for a .bib file


  # Realistic use after actually retrieving a tree from a backend:
  if (requireNamespace("fishtree", quietly = TRUE)) {
    res <- pr_get_tree(c("Salmo salar", "Esox lucius"),
                       source = "fishtree")
    cat(pr_cite_tree(res, format = "markdown"))
  }
```

---

pr_date_tree *Time-calibrate a topology using the DateLife chronogram database*

Description

Wraps datelife::datelife_use() to add divergence-time calibrations to an existing phylo (or multiPhylo) using DateLife's database of pre-computed chronograms (Sanchez Reyes et al. 2024, *Systematic Biology* 73:470). Returns a result with the same shape as pr_get_tree() so downstream PCM workflows — including pigauto's posterior-tree imputation — can consume it without further glue code.

Usage

```
pr_date_tree(
  tree,
  n_dated = 1L,
  dating_method = "bladj",
  check_ultrametric = TRUE,
```

```
  ...
)
```

Arguments

tree	An ape::phylo (or multiPhylo) object: the topology (or topologies) to calibrate.
n_dated	A length-1 positive integer. How many calibrated trees to return per input topology. 1L (default) returns a single dated tree (DateLife's combined SDM-summary chronogram); > 1L triggers each = TRUE so DateLife returns one chronogram per source paper in its database (capped at n_dated). All resulting chronograms share the input topology — only the branch lengths vary across the returned set.
dating_method	A length-1 character vector. Forwarded to datelife::datelife_use(). One of "bladj" (default; fast, no calibration uncertainty) or "mrbayes" (Bayesian; slower, produces credible intervals).
check_ultrametric	Logical. After dating, check that the result is ultrametric and warn if not. Default TRUE. datelife::datelife_use() is supposed to produce ultrametric chronograms; this catches regressions.
...	Additional arguments forwarded to datelife::datelife_use().

Value

A list with class pr_tree_result and components:

tree The dated topology — a phylo when n_dated = 1 or a multiPhylo when n_dated > 1.

matched Tip labels of the input that DateLife was able to date.

unmatched Tip labels of the input absent from DateLife's database (returned with no calibration applied).

mapping A tibble with one row per input tip label, mirroring pr_get_tree()'s audit table: input_name, normalized_name, query_name, tree_name, in_tree, match_type, placement_status, and the four tnrs_* columns. placement_status and the tnrs_* columns are NA for DateLife dating, which applies no TNRS step.

source Always "datelife" (paired with pr_get_tree()'s dispatch).

backend_meta Includes dating_method, calibrations (per-node calibration table from DateLife), and the standard tree_provenance list (one entry per returned tree).

When to use this

Use pr_date_tree() when you already have a topology (e.g. from a published phylogeny or your own analysis) and want to attach divergence times. Use pr_get_tree() with source = "datelife" if you have only species names. Both end up calling the GitHub-only datelife package, but the starting point is different. Install datelife before calling this function — **prepR4pcm** does NOT pull it in via Suggests (its transitive dep tree can't be auto-resolved by pak on a clean CI image, so we keep it as an opt-in install): pak::pak("phylotastic/datelife").

What "n_dated > 1" actually returns

This is a common point of confusion. With `n_dated = 50`, `pr_date_tree()` does NOT change the input topology — it returns up to 50 chronograms that *all share* the input topology but differ in their branch lengths, because each variant is dated using a different source paper in DateLife's chronogram database (think: variant 1 uses Hedges et al. 2015, variant 2 uses Bininda-Emonds et al. 2007, etc.). So you get one topology and N versions of branch lengths, not N different topologies.

If you want **both** axes of variation (topology uncertainty + dating uncertainty), feed a `multiPhylo` of N topologies in. DateLife's `each = TRUE` mode is then applied per input tree, so the output reflects the cross-product of input topology and DateLife source. Example pipeline:

```
trees  <- pr_get_tree(species, source = "rtrees",
                      taxon = "mammal")  # ~100 topologies
dated  <- pr_date_tree(trees$tree, n_dated = 5)
```

By contrast, `pr_get_tree(species, source = "datelife", n_tree = 50)` returns up to 50 chronograms where each variant comes from a different DateLife source — i.e. a different topology AND different branch lengths per variant, because DateLife's source chronograms aren't constrained to share a topology.

References

Sanchez Reyes, L. L., McTavish, E. J., & O'Meara, B. (2024). DateLife: Leveraging databases and analytical tools to reveal the dated Tree of Life. *Systematic Biology*, 73(2), 470–485. doi:10.1093/sysbio/syae015

See Also

`pr_get_tree()` for retrieval (species –> tree); `pr_cite_tree()` for formatting the citations of the result; `reconcile_augment()` for filling tip-level gaps in an existing tree (a complementary operation to dating).

Examples

```
if (rlang::is_installed("datelife")) {
  # Example 1: one chronogram from a topology
  library(ape)
  tr  <- read.tree(text =
    "(Rhea_americana,(Pterocnemia_pennata,Struthio_camelus));")
  res <- pr_date_tree(tr)
  res$tree                       # phylo (chronogram)
  res$backend_meta$dating_method # "bladj"

  # Example 2: per-source chronograms for posterior-tree PCMs
  res <- pr_date_tree(tr, n_dated = 5)
  class(res$tree)                # "multiPhylo"
  length(res$backend_meta$tree_provenance)  # one entry per tree
}
```

pr_extract_tips *Extract tip labels from a phylogenetic tree*

Description

Return the tip labels of a tree as a character vector, whether the tree is already an `ape::phylo` object in memory or lives in a Newick or Nexus file on disk. Convenience wrapper around `tree$tip.label` that also handles file input and multi-tree files (returns the tips of the first tree).

Usage

```
pr_extract_tips(tree)
```

Arguments

`tree`	An `ape::phylo` object, or a length-1 character vector giving the path to a Newick (`.nwk`, `.tre`, `.tree`, `.newick`) or Nexus (`.nex`, `.nexus`) file. Format is auto-detected.

Value

A character vector of tip labels (one element per tip).

See Also

`pr_normalize_names()` for cleaning tip labels before joining against a data frame.

Other name utilities: `pr_normalize_names()`

Examples

```
data(tree_jetz)
head(pr_extract_tips(tree_jetz))
```

pr_get_tree *Retrieve a candidate phylogeny for a species list*

Description

Connects reconciled species names to an external phylogenetic resource and returns a pruned candidate tree plus a report of which species were matched and which were dropped. Intended as the bridge between the package's reconciliation cascade and any downstream comparative analysis: feed the result of `reconcile_data()` / `reconcile_tree()` (or any character vector of cleaned names) into `pr_get_tree()` and get back a `phylo` ready for `reconcile_apply()`.

Usage

```
pr_get_tree(
  x,
  source = c("rotl", "rtrees", "clootl", "fishtree", "datelife", "auto"),
  species_col = NULL,
  taxon = NULL,
  n_tree = 1L,
  cache = FALSE,
  tnrs = c("auto", "always", "never"),
  min_match = 0.8,
  check_ultrametric = TRUE,
  resolve_polytomies = FALSE,
  branch_lengths = NULL,
  ...
)
```

Arguments

x	One of:
	a `reconciliation` **object** returned by `reconcile_tree()` or `reconcile_data()`; species are taken from the reconciled `name_y` column with `NA`s and unresolved entries dropped.
	a character vector used directly after deduplication and NA removal.
	a data frame `species_col` must name a character column; its unique non-NA values are used.
source	A length-1 character vector. Which external backend to use. One of:
	`"rotl"` Open Tree of Life synthesis tree, via the CRAN package `rotl`. Universal taxonomic coverage; calls `tnrs_match_names()` to resolve names to OTT ids and then `tol_induced_subtree()`.
	`"rtrees"` Taxon-specific mega-trees (bird, mammal, fish, amphibian, reptile, plant, shark/ray, bee, butterfly) via the GitHub package `rtrees` (https://daijiang.github.io/rtrees/). Requires `taxon = "<group>"`. Calls `get_tree()`. Install with `pak::pak("daijiang/rtrees")` (GitHub-only). **Grafting behaviour:** when an input species is not in the chosen mega-tree, `rtrees::get_tree()` grafts it at the genus level (tip suffix `*`) or family level (`**`); if no co-family species is in the mega-tree, the species is dropped. The placement of every input species is reported per-row in `result$backend_meta$placement` (a tibble with columns `input_name`, `tree_name`, `placement_status` where `placement_status` is one of `"exact"`, `"genus_added"`, `"family_added"`, `"skipped"`, or `"unmatched"`). The grafting itself cannot be disabled at the wrapper level (rtrees 1.0.4 has no switch); to exclude grafted tips from a downstream analysis, filter the placement table on `placement_status == "exact"` and prune the tree to those tip labels. See `?rtrees::get_tree` for upstream control (`scenario` *where* a graft is placed, but not *whether*).
	`"clootl"` Bird-only phylogenies in current Clements taxonomy, via the GitHub package `clootl` (https://github.com/eliotmiller/clootl). Calls `extractTree()`.

Install with `pak::pak("eliotmiller/clootl")`.

`"fishtree"` Fish-only time-calibrated phylogeny (Rabosky et al. 2018), via the CRAN package `fishtree`. Calls `fishtree_phylogeny()` (single tree) or `fishtree_complete_phylogeny()` (multi-tree posterior; triggered by `n_tree > 1`). Requires exact name matches against the Fish Tree of Life taxonomy — pre-clean with `reconcile_data()` (with a `taxadb` authority) for best results.

`"datelife"` Universal database of pre-computed chronograms (Sanchez Reyes et al. 2024, *Syst. Biol.* 73:470), via the GitHub package `datelife` (https://github.com/phylotastic/datelife). Returns a single SDM-summary chronogram by default; with `n_tree > 1`, returns a multiPhylo of up to that many per-source candidate chronograms. **Install before use** with `pak::pak("phylotastic/dateli` — the package is GitHub-only (archived from CRAN in 2024 with a heavy transitive dep tree pak can't auto-resolve), so **prepR4pcm** does NOT pull it in via `Suggests`.

`"auto"` Fall-through dispatcher: try installed backends in priority order (rtrees if `taxon` provided, then rotl, fishtree, clootl, datelife), return the first result that resolves at least `min_match` of the species. Useful for first-pass exploration when you don't yet know which backend covers your taxa.

`species_col` A length-1 character vector. Required when `x` is a data frame; ignored otherwise.

`taxon` A length-1 character vector. Required when `source = "rtrees"`. One of `"bird"`, `"mammal"`, `"fish"`, `"amphibian"`, `"reptile"`, `"plant"`, `"shark_ray"`, `"bee"`, `"butterfly"` (see the `rtrees` package help for `get_tree`). Ignored for other backends.

`n_tree` A length-1 positive integer. How many trees to request from the backend. Default `1L` (single phylo for back-compat). Each backend negotiates this differently:

`"rotl"` Always returns 1 (the synthesis tree). A one-shot warning is emitted if `n_tree > 1`.

`"rtrees"` `n_tree` **is informational only here.** `rtrees::get_tree()` does not have an `n_tree` argument; the multi-tree count is fixed by which mega-tree was selected. Reference trees rtrees uses internally: birds = Jetz et al. 2012 (https://birdtree.org, 100 posterior trees); mammals = Upham et al. 2019 (VertLife, 100 by default; set `mammal_tree = "phylacine"` for the PHYLACINE set); amphibians + squamates = VertLife; fish = Rabosky et al. 2018 (also wrapped by `source = "fishtree"`); plants = V.PhyloMaker; bees = Bee Tree of Life. Override which mega-tree is used via `...` (e.g. `bee_tree = "bootstrap"` for 100 bee trees instead of the single ML tree). Requires `taxon`.

`"clootl"` `n_tree = 1` calls `clootl::extractTree()` and works out of the box with the v1.6 / 2025 taxonomy bundled in the `clootl` package. `n_tree > 1` calls `clootl::sampleTrees(count = n_tree)` (capped at 100 upstream) and **requires the AvesData repo to be set up once via** `clootl::get_avesdata_repo(".")` **first**; otherwise it errors with `AvesData repo not found`.

`"fishtree"` Single phylo via `fishtree_phylogeny()` when `n_tree = 1`; switches to `fishtree_complete_phylogeny()` returning a multiPhylo of stochastically polytomy-resolved trees when `n_tree > 1`.

"datelife" summary_format = "phylo_sdm" (single summary chronogram) when n_tree = 1; switches to summary_format = "phylo_all" (one chronogram per source, capped at n_tree) when n_tree > 1.

When the request returns a multiPhylo, the result's tree slot is multiPhylo; otherwise phylo.

cache — Logical. Cache the result on disk and reuse it on subsequent identical calls? Default FALSE. When TRUE, the request is keyed by (species, source, n_tree, taxon, tnrs, ...) and stored at pr_tree_cache_dir(). See pr_tree_cache_status() and pr_tree_cache_clear() for inspecting / wiping the cache.

tnrs — A length-1 character vector. Run a TNRS preflight (Open Tree of Life name resolution via rotl::tnrs_match_names) on the species list before calling the backend? One of:

"auto" **(default)** Run TNRS only for fishtree, where OTL-resolved names tend to improve the match rate. **Not run for** clootl **by default**: clootl uses the eBird / Clements taxonomy, so OTL-resolved names are often different from clootl's preferred names; the network call is also the dominant cost for large requests (~15 min for 10k species before this change). Pass tnrs = "always" if you want it for clootl anyway.

"always" Run TNRS regardless of backend.

"never" Skip TNRS even when the backend would benefit.

When rotl is not installed, TNRS is silently skipped with a one-shot warning.

min_match — A length-1 numeric in [0, 1]. Only used when source = "auto". The minimum fraction of input species a backend must resolve for the dispatcher to accept its result; if no backend meets the threshold, the best available is returned with a warning. Default 0.8.

check_ultrametric — Logical. After producing the tree, check that it's ultrametric (all tips equidistant from the root) and warn if not. Default TRUE. Only enforced for backends that normally return chronograms (rtrees, clootl, fishtree, datelife); rotl returns a topology without real branch lengths, so the check is skipped. To force ultrametricity on a non-ultrametric result, use phytools::force.ultrametric() or ape::chronos() directly — prepR4pcm does not modify the tree silently.

resolve_polytomies — Logical. After retrieval, resolve any polytomies via ape::multi2di() with random = TRUE? Default FALSE (back-compat; topology preserved). Useful for phylogenetic meta-analysis, where a strictly bifurcating tree is required for pr_phylo_cor() / ape::vcv() to produce a full-rank correlation matrix.

branch_lengths — A length-1 character vector or NULL. After retrieval (and after polytomy resolution if requested), assign branch lengths via the named method? Default NULL (no transformation; backend's branch lengths are kept as-is). Other values:

"grafen" Grafen's (1989) method via ape::compute.brlen() with method = "Grafen". The canonical choice for phylogenetic meta-analysis when the topology comes from rotl (whose edge lengths are unit-length placeholders). See Cinar et al. (2022) *Methods Ecol. Evol.* 13:383, who use this exact pattern.

"compute.brlen" Same as "grafen" — Grafen is ape::compute.brlen()'s default method. Provided as an alias for users who think in terms of the underlying function name.

"unit" Set every edge length to 1. The crudest option; useful only for sensitivity-analysis comparisons.

... Backend-specific arguments forwarded to the underlying call. See the help page of the underlying function in the relevant backend package (tol_induced_subtree in rotl, extractTree in clootl, get_tree in rtrees, fishtree_phylogeny / fishtree_complete_phylogeny in fishtree, datelife_search in datelife) for the full list.

Details

Each backend is provided by an external R package that we list in Suggests rather than Imports, so installing **prepR4pcm** does not pull them in automatically. The error message tells you what to install if you ask for a backend you don't have.

Name handling. Input names are run through pr_normalize_names() before the backend is queried — underscores become spaces, leading/trailing whitespace is trimmed, OTT-id suffixes (e.g. ott770315) and authority strings (e.g. (Linnaeus, 1758)) are stripped, and hybrid signs are standardised. The matched and unmatched slots in the result use the *original* input format (as you typed it), not the normalised form.

When TNRS substitutes a name (only when tnrs = "always", or for the fishtree backend under tnrs = "auto"), the replacement is recorded in result$backend_meta$tnrs_replacements as a named character vector (original = resolved). A one-shot cli warning lists the first few substitutions on the call itself.

TNRS also returns structured match metadata. pr_get_tree() records it per name in the mapping tibble: tnrs_number_matches, tnrs_is_synonym, tnrs_approximate_match, and tnrs_flags. When a name resolves to more than one taxon (tnrs_number_matches > 1, a homonym), a one-shot cli warning names the affected species, since the resolved name is then only one of several candidates.

Value

A list with class pr_tree_result and components:

tree A phylo (single) or multiPhylo (posterior) object from the chosen backend, pruned to the matched species.

matched Character vector of names from the user's **original input** (preserving the input format, including any underscores) that resolved to a tip in tree. The dispatcher enforces that matched names are a subset of unique(input) — TNRS substitution, normalisation, and backend-internal name juggling cannot leak intermediate names into this slot.

unmatched Character vector of names from the **original input** that did not resolve. Disjoint from matched; length(matched) + length(unmatched) == length(unique(input)) always holds. Inspect these and consider running them back through reconcile_suggest() / a manual override.

`mapping` A tibble with one row per unique input species. Core columns: `input_name`, `normalized_name`, `query_name`, `tree_name`, `in_tree`, `match_type`, and `placement_status`. This is the audit trail for name handling: `input_name` is what the user supplied, `normalized_name` is the result of `pr_normalize_names()`, `query_name` is the backend query after optional TNRS, `tree_name` is the actual returned tip label, and `match_type` is one of `"exact"`, `"normalized"`, `"tnrs"`, or `"unmatched"`. For `source = "rtrees"`, `placement_status` carries the grafting status from `backend_meta$placement`; otherwise it is `NA`. Four further columns record what `rotl`'s TNRS resolver reported for each name: `tnrs_number_matches`, `tnrs_is_synonym`, `tnrs_approximate_match`, and `tnrs_flags`. These are `NA` for backends or `tnrs` settings where TNRS did not run. `tnrs_number_matches > 1` flags a homonym, meaning the resolved name is only one of several candidate taxa.

`source` The backend that produced the tree.

`backend_meta` A named list of diagnostic information. Standard fields populated by the dispatcher:

`n_queried` Unique input species count.

`n_requested` The `n_tree` argument the user passed.

`n_returned` Number of trees in `tree` (1 for `phylo`).

`n_matched` Equal to `length(matched)`.

`tnrs_replacements` When TNRS ran (`tnrs = "always"`, or `tnrs = "auto"` for `fishtree`) and `rotl` is installed: a named character vector mapping original input to the TNRS-resolved name, for names that TNRS changed. `NULL` when no TNRS or no replacements occurred. A one-shot `cli` warning lists the first three substitutions on the call, so silent name correction is impossible.

`tip_set_consistent` Logical. For `multiPhylo` returns: `TRUE` if every tree shares the same tip set.

`dropped_per_tree` For `multiPhylo` returns where `tip_set_consistent = FALSE`: a list of character vectors, per tree, listing species missing from each tree relative to the union of all trees. `NULL` otherwise.

`tree_provenance` A list with one entry per returned tree (so `tree[[i]]` pairs with `backend_meta$tree_provenance[` when `tree` is a `multiPhylo`).

Backend-specific fields (e.g. `taxon`, `n_grafted`, `grafted_tips`, `placement` for `rtrees`; `backend`, `type`, `tnrs_table` for `fishtree` / `rotl`; `summary_format`, `source_citations`, `reference` for `datelife`) are merged in at the top level by the wrapper that called the backend. The `rtrees`-specific `placement` slot is a tibble with one row per unique input species and columns `input_name`, `tree_name`, `placement_status` (`"exact"`, `"genus_added"`, `"family_added"`, `"skipped"`, or `"unmatched"`).

References

Backend reference trees:

Jetz, W., Thomas, G. H., Joy, J. B., Hartmann, K., & Mooers, A. O. (2012). The global diversity of birds in space and time. *Nature* 491: 444–448. doi:10.1038/nature11631 (Used by `rtrees` for `taxon = "bird"` and by BirdTree.)

Rabosky, D. L., Chang, J., Title, P. O., Cowman, P. F., Sallan, L., Friedman, M., Kaschner, K., Garilao, C., Near, T. J., Coll, M., & Alfaro, M. E. (2018). An inverse latitudinal gradient in speciation rate for marine fishes. *Nature* 559: 392–395. doi:10.1038/s41586-018-0273-1 (Fish Tree of Life; used by `source = "fishtree"` and by `rtrees` for `taxon = "fish"`.)

Upham, N. S., Esselstyn, J. A., & Jetz, W. (2019). Inferring the mammal tree: Species-level sets of phylogenies for questions in ecology, evolution, and conservation. *PLOS Biology* 17(12): e3000494. doi:10.1371/journal.pbio.3000494 (VertLife mammal posterior; used by `rtrees` for `taxon = "mammal"` with `mammal_tree = "vertlife"`.)

Jin, Y. & Qian, H. (2019). V.PhyloMaker: an R package that can generate very large phylogenies for vascular plants. *Ecography* 42(8): 1353–1359. doi:10.1111/ecog.04434 (Vascular-plant mega-tree used by `rtrees` for `taxon = "plant"`; also the basis for the `source = "vphylomaker"` augmentation backend in `reconcile_augment()`.)

Sanchez Reyes, L. L., O'Meara, B. C., Brown, J. W., & McTavish, E. J. (2024). DateLife: Leveraging databases and analytical tools to reveal the dated Tree of Life. *Systematic Biology* 73(2): 470–485. doi:10.1093/sysbio/syae015 (Used by `source = "datelife"` and by `pr_date_tree()`.)

Methodology:

Chang, J., Rabosky, D. L., & Alfaro, M. E. (2019). Estimating diversification rates on incompletely sampled phylogenies: Theoretical concerns and practical solutions. *Systematic Biology* 69(3): 602–611. doi:10.1093/sysbio/syz081 (Stochastic polytomy resolution behind `fishtree_complete_phylogeny()` for `n_tree > 1`.)

Michonneau, F., Brown, J. W., & Winter, D. J. (2016). rotl: an R package to interact with the Open Tree of Life data. *Methods in Ecology and Evolution* 7(12): 1476–1481. doi:10.1111/2041-210X.12593 (TNRS preflight and `source = "rotl"`.)

See Also

`reconcile_tree()` / `reconcile_data()` for producing the reconciled species list that feeds this function; `reconcile_apply()` for combining the returned `phylo` with the data frame ready for analysis; `reconcile_augment()` for filling gaps in an existing tree (a tree-aware alternative to retrieving a fresh tree); `pr_date_tree()` for time-calibrating an existing topology; `pr_cite_tree()` for formatting citations for a tree result; `pr_tree_compare()` for comparing two or more retrieved trees; `pr_get_tree_status()` for checking which backends are installed and reachable; `pr_tree_cache_dir()` / `pr_tree_cache_status()` / `pr_tree_cache_clear()` for managing the on-disk cache. The companion package pigauto consumes a `multiPhylo` directly via `multi_impute_trees()` for posterior- tree PCMs — request a posterior sample with `n_tree > 1`.

Examples

```
if (interactive()) {
  # Example 1: birds via clootl (Clements taxonomy). Uses the
  # bundled AVONET subset (657 species placed in the Clements tree).
  data(avonet_subset)
  if (requireNamespace("clootl", quietly = TRUE)) {
    res <- pr_get_tree(avonet_subset, species_col = "Species1",
                       source = "clootl")
    ape::Ntip(res$tree)        # species placed in the tree
    head(res$unmatched)        # names clootl could not resolve
  }

  # Example 2: fish via fishtree (Rabosky et al. 2018, time-calibrated)
  if (requireNamespace("fishtree", quietly = TRUE)) {
    res <- pr_get_tree(c("Salmo salar", "Esox lucius", "Gadus morhua"),
                       source = "fishtree")
```

```
  res$tree
}

# Example 3: anything via rotl (universal, network)
if (requireNamespace("rotl", quietly = TRUE)) {
  res <- pr_get_tree(c("Homo sapiens", "Pan troglodytes",
                       "Mus musculus"),
                     source = "rotl")
  res$tree
}

# Example 4: posterior of fish trees (50 trees, for multi-tree PCMs)
if (requireNamespace("fishtree", quietly = TRUE)) {
  res <- pr_get_tree(c("Salmo salar", "Esox lucius"),
                     source = "fishtree", n_tree = 50)
  class(res$tree)            # "multiPhylo"
}
}
```

`pr_get_tree_status` *Report the install status of every* `pr_get_tree()` *backend*

Description

Walks every backend supported by `pr_get_tree()` and reports whether the underlying package is installed (and at what version), whether it requires network, and what to do if it's missing. Useful for first-time users figuring out which backends are available, and for CI sanity checks.

Usage

```
pr_get_tree_status(check_network = FALSE)
```

Arguments

`check_network`	Logical. Should the probe attempt a tiny network call to test that backends needing the network are actually reachable? Default `FALSE` (purely local check, no side effects). Set `TRUE` to also test reachability — adds 1-3 seconds and requires internet.

Value

A `data.frame` with one row per backend and columns:

`source` Backend name, as passed to `pr_get_tree()`.

`installed` Logical — is the package available?

`version` Character — installed version, or `NA`.

`needs_network` Logical — does the backend hit a remote server at runtime?

`reachable` Logical or `NA` — result of the network check (only populated when `check_network = TRUE`).

`install_hint` Character — the install command to run when `installed = FALSE`.

`source_repo` Character — "CRAN" or a GitHub repo for non-CRAN backends.

See Also

`pr_get_tree()` / `pr_date_tree()` for the consumers.

Examples

```
# Local-only probe (fast, no network)
pr_get_tree_status()


  # Also test reachability of remote backends
  pr_get_tree_status(check_network = TRUE)
```

---

pr_normalize_names *Normalise scientific names to a canonical form*

Description

Apply a sequence of deterministic text transformations so that scientific names which differ only in formatting compare equal. This is the same routine used by stage 2 of the matching cascade in `reconcile_data()` and `reconcile_tree()`. Use it directly when you want to clean a column of names without running a full reconciliation — for example, when building a crosswalk by hand.

Usage

```
pr_normalize_names(
  names,
  rank = c("species", "subspecies"),
  parser = c("internal", "gnparser")
)
```

Arguments

`names`	A character vector of scientific names (any length; each element is a single name). `NA` values are preserved as `NA`.
`rank`	A length-1 character vector. Taxonomic rank to normalise to:
	`"species"` **(default)** Strip infraspecific epithets so trinomials become binomials (`Parus major major` -> `Parus major`).
	`"subspecies"` Keep trinomials intact.
`parser`	A length-1 character vector. Which parsing engine to use:

`"internal"` **(default)** The package's own regex-based cascade described above. No external dependency.

`"gnparser"` Delegates parsing to `rgnparser::gn_parse_tidy()`, which wraps the gnparser Go binary (part of the Global Names Architecture). Handles hybrid signs, complex multi-author year strings, and trailing parentheticals (Open Tree homonym / rank flags) more robustly than the internal cascade. Requires both the **rgnparser** R package and the `gnparser` binary on the system PATH; the function errors helpfully if either is missing. Returns the same shape and `normalisation_log` attribute as the internal path, so the two are drop-in interchangeable.

Details

The transformations, applied in order, are:

1. Replace underscores and multiple whitespace with a single space (`Homo_sapiens` -> `Homo sapiens`).
2. Strip authority strings and year, including multi-author and parenthetical forms (`Corvus corax (Linnaeus, 1758)` -> `Corvus corax`).
3. Strip any other trailing parenthetical qualifier, such as the Open Tree of Life homonym / rank flags that `rotl` returns (`Prunella (genus in kingdom Archaeplastida)` -> `Prunella`).
4. Fold diacritics to ASCII (`Passer domesticus` stays as `Passer domesticus`; accented characters are simplified).
5. Standardise case: genus capitalised, epithet lowercase.
6. Strip infraspecific epithets if `rank = "species"`.
7. Trim whitespace and collapse leftover empty tokens.

Value

A character vector of normalised names, the same length as `names`, with an attribute `"normalisation_log"` — a tibble recording every non-trivial change, for auditing.

Note

On the spelling: the title and prose use British English *normalise*, consistent with the package's `Language: en-GB` declaration. The function identifier `pr_normalize_names()` keeps the American-English `z` because R-package function names conventionally use ASCII identifiers in the form most R users expect. The two spellings are equivalent and intentional.

See Also

`reconcile_data()` and `reconcile_tree()` for the full four-stage matching cascade; `pr_extract_tips()` for pulling tip labels out of a tree prior to normalising them.

Other name utilities: `pr_extract_tips()`

Examples

```
pr_normalize_names(c("Homo_sapiens",
                     "homo sapiens",
                     "Parus major major",
                     "Corvus corax (Linnaeus, 1758)"))

# Keep trinomials
pr_normalize_names("Parus major major", rank = "subspecies")
```

---

pr_phylo_cor *Phylogenetic correlation matrix from a tree*

Description

Convert a phylogeny into the correlation matrix used as a random- effect structure in phylogenetic meta-analysis (`metafor::rma.mv`) or phylogenetic mixed models (`MCMCglmm`, `brms`, etc.).

Usage

```
pr_phylo_cor(x, corr = TRUE, ...)
```

Arguments

`x`	A `phylo` object, a `multiPhylo`, or a `pr_tree_result` (the `$tree` slot is extracted). For `multiPhylo` input, returns a list of correlation matrices.
`corr`	Logical. Pass through to `ape::vcv()`. `TRUE` (default) returns a correlation matrix (diagonal = 1); `FALSE` returns the variance-covariance matrix.
`...`	Additional arguments forwarded to `ape::vcv()`.

Details

Wraps `ape::vcv()` with `corr = TRUE`. Designed to slot in after `pr_get_tree()` when the goal is meta-analysis, where typically:

1. Topology comes from Open Tree of Life (`source = "rotl"`) because the species span many higher taxa.
2. Polytomies are resolved at random (`resolve_polytomies = TRUE`).
3. Branch lengths are computed via Grafen's method (`branch_lengths = "grafen"`) because rotl's edge lengths are unit-length placeholders.
4. The correlation matrix is computed once and reused as `random = ~1 | species`'s `R = list(species = phy_cor)` in `metafor::rma.mv()` (or `random = ~species` with `cov.formula = ~ phylo` in `MCMCglmm`).

The correlation matrix has the property that, for a Brownian-motion model on a tree with branch lengths in time units, two species' off-diagonal entry equals the time from root to their MRCA divided by the time from root to tip. So an ultrametric tree always has diagonal = 1 (every tip is the same distance from the root).

For meta-analysis with rotl topology + Grafen's method, the resulting matrix is the standard Pagel's lambda = 1 phylogenetic correlation that `metafor::rma.mv()` accepts directly.

Value

A square symmetric matrix with row/column names equal to the tip labels. For `multiPhylo` input, a list of such matrices.

References

Paradis, E., & Schliep, K. (2019). ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R. *Bioinformatics*, 35(3), 526–528. doi:10.1093/bioinformatics/bty633

Cinar, O., Nakagawa, S., & Viechtbauer, W. (2022). Phylogenetic multilevel meta-analysis: a simulation study on the importance of modelling the phylogeny. *Methods in Ecology and Evolution*, 13(2), 383–395. doi:10.1111/2041210X.13760

See Also

`pr_get_tree()` (use with `branch_lengths = "grafen"` and `resolve_polytomies = TRUE` for the meta-analysis path); `ape::vcv()` for the underlying computation.

Examples

```
set.seed(1)
tr <- ape::rcoal(5)             # ultrametric, bifurcating
phy_cor <- pr_phylo_cor(tr)
dim(phy_cor)
all(diag(phy_cor) == 1)


  # End-to-end meta-analysis prep
  if (requireNamespace("rotl", quietly = TRUE)) {
    res <- try(pr_get_tree(c("Homo sapiens", "Pan troglodytes",
                             "Mus musculus", "Rattus norvegicus"),
                           source             = "rotl",
                           resolve_polytomies = TRUE,
                           branch_lengths     = "grafen"),
               silent = TRUE)
    if (!inherits(res, "try-error")) {
      phy_cor <- pr_phylo_cor(res)
      # phy_cor can now be supplied to downstream meta-analysis models.
    }
  }
```

pr_tree_cache_clear *Clear the local tree-retrieval cache*

Description

Removes all cached `pr_get_tree()` / `pr_date_tree()` results. By default asks for confirmation before deleting; pass `confirm = FALSE` to skip the prompt (useful in scripts).

Usage

```
pr_tree_cache_clear(confirm = TRUE, source = NULL)
```

Arguments

`confirm`	Logical. Ask interactively before deleting? Default `TRUE`. Ignored in non-interactive sessions (deletion proceeds).
`source`	A length-1 character vector or `NULL`. If non-`NULL`, only entries from that backend are cleared (e.g. `"datelife"` to wipe only datelife cache after a database refresh). If `NULL` (default), all entries are cleared.

Value

Invisibly, the number of files removed.

See Also

`pr_tree_cache_dir()` / `pr_tree_cache_status()`.

Examples

```
# Demo against a throwaway cache so the user's real cache is untouched
old_opt   <- getOption("prepR4pcm.cache_dir")
tmp_cache <- file.path(tempdir(), "prepR4pcm-cache-demo")
pr_tree_cache_dir(tmp_cache)

# Drop two dummy entries so there is something to clear:
dir.create(file.path(tmp_cache, "fishtree"), showWarnings = FALSE)
dir.create(file.path(tmp_cache, "rotl"),     showWarnings = FALSE)
saveRDS(NULL, file.path(tmp_cache, "fishtree", "abc.rds"))
saveRDS(NULL, file.path(tmp_cache, "rotl",     "def.rds"))

pr_tree_cache_status()                # 2 entries
pr_tree_cache_clear(confirm = FALSE)  # removes both
pr_tree_cache_status()                # empty

# Restore the previous cache directory
options(prepR4pcm.cache_dir = old_opt)
```

`pr_tree_cache_dir` *Get or set the local tree-retrieval cache directory*

Description

Returns the path to the cache directory used by `pr_get_tree()` and `pr_date_tree()` when called with `cache = TRUE`. Pass a path to override the default.

Usage

```
pr_tree_cache_dir(path = NULL)
```

Arguments

`path`	A length-1 character vector or `NULL`. If non-`NULL`, sets the cache directory to `path` (creating it if it doesn't exist). If `NULL` (default), returns the currently configured directory.

Details

The default cache directory is `tools::R_user_dir()` with type `"cache"` and the package name `"prepR4pcm"`, which on Linux is typically `~/.cache/R/prepR4pcm/`, on macOS `~/Library/Caches/org.R-project.R/R/` and on Windows something under `%LOCALAPPDATA%\R\cache\R\prepR4pcm\`.

To use a cache directory you control, pass its path explicitly with `pr_tree_cache_dir(path)`.

Value

A length-1 character vector — the absolute path of the cache directory.

See Also

`pr_tree_cache_status()` / `pr_tree_cache_clear()`; `pr_get_tree()` for the consumer.

Examples

```
# Default location
pr_tree_cache_dir()

old_cache <- getOption("prepR4pcm.cache_dir", NULL)
tmp_cache <- tempfile("prepR4pcm-cache-")
pr_tree_cache_dir(tmp_cache)
options(prepR4pcm.cache_dir = old_cache)
unlink(tmp_cache, recursive = TRUE)
```

pr_tree_cache_status *Show the contents of the local tree-retrieval cache*

Description

Lists every cache entry by `source`, with file size and modification timestamp. Useful for figuring out where the disk space went or for confirming a fresh run hit the cache.

Usage

```
pr_tree_cache_status()
```

Value

A `data.frame` (sorted by most recent first) with columns `source`, `hash`, `size_kb`, `modified`. Returns an empty data frame with the same columns when the cache is empty.

See Also

`pr_tree_cache_dir()` / `pr_tree_cache_clear()`.

Examples

```
pr_tree_cache_status()
```

pr_tree_compare *Compare two or more phylogenetic trees*

Description

Computes a small set of standard metrics for comparing trees that come from different backends (or different runs of the same backend). Designed for the common case of "I retrieved a tree from rotl and another from fishtree — do they agree?"

Usage

```
pr_tree_compare(..., prune_to_common = TRUE)
```

Arguments

`...`	Two or more `phylo` objects, or two or more `pr_tree_result` objects (the `tree` slot is extracted), or `multiPhylo` objects (the first tree is used). Trees can be passed as positional arguments or as a named list.
`prune_to_common`	Logical. Restrict each tree to the shared tip set before computing topology metrics? Default `TRUE` — without this, RF distance is undefined when tip sets differ.

Details

RF distance is computed via `ape::dist.topo()` with the default method. Branch-length correlation matches edges by their tip-set bipartition: for each edge in tree A, the corresponding edge in tree B (if any) is the one that splits the same set of tips. The Pearson correlation is taken over the matched edge-length pairs; edges whose bipartition is absent in the other tree are dropped. This is a proper bipartition-matched correlation as introduced in Kuhner & Felsenstein (1994) for tree comparison.

Value

A list with class `pr_tree_compare` and components:

`n_trees` Number of input trees.

`tip_sets` Named list of character vectors, one per tree.

`shared_tips` Tips present in every input tree.

`unique_to` Named list, one per tree, of tips present in that tree but not in every other tree.

`n_shared` Length-1 integer.

`pairwise_jaccard` Square matrix; `(i, j)` is the Jaccard index of `tip_sets[[i]]` vs `tip_sets[[j]]`.

`pairwise_rf` Square matrix of Robinson-Foulds distances between pairs of trees pruned to `shared_tips`. `NA` when the pair has < 4 shared tips.

`pairwise_branch_cor` Square matrix of Pearson correlations between matching edge lengths in each pair, or `NA` when one or both trees have no branch lengths.

References

Kuhner, M. K., & Felsenstein, J. (1994). A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Molecular Biology and Evolution* 11(3): 459–468. doi:10.1093/oxfordjournals.molbev.a040126

Robinson, D. F., & Foulds, L. R. (1981). Comparison of phylogenetic trees. *Mathematical Biosciences* 53(1–2): 131–147. doi:10.1016/00255564(81)900432

See Also

`pr_get_tree()` for retrieval; `reconcile_apply()` for combining a chosen tree with a dataset.

Examples

```
# Two trees with identical tip sets
set.seed(1)
t1 <- ape::rtree(10)
t2 <- ape::rtree(10, tip.label = t1$tip.label)
cmp <- pr_tree_compare(t1, t2)
cmp$n_shared
cmp$pairwise_rf

# Two trees with overlapping but not identical tips
t3 <- ape::rtree(8, tip.label = t1$tip.label[1:8])
cmp <- pr_tree_compare(t1, t3)
```

```
cmp$pairwise_jaccard
```

`reconcile_apply` *Apply a reconciliation to produce an aligned data-tree pair*

Description

Turn a reconciliation object into an analysis-ready data frame and pruned phylogenetic tree whose species labels agree. This is the step that feeds directly into `caper::pgls()`, `MCMCglmm::MCMCglmm()`, `phytools::fastAnc()`, or any other PCM that expects matching names in data and tree.

Usage

```
reconcile_apply(
  reconciliation,
  data = NULL,
  tree = NULL,
  species_col = NULL,
  drop_unresolved = FALSE
)
```

Arguments

`reconciliation`	A reconciliation object returned by `reconcile_tree()`, `reconcile_data()`, or a related matcher.
`data`	A data frame to align. If `NULL`, only the tree is processed and the returned `data` slot is `NULL`.
`tree`	An `ape::phylo` object (or path to a Newick / Nexus file) to align. If `NULL`, only the data frame is processed and the returned `tree` slot is `NULL`. (Passing both `data = NULL` and `tree = NULL` is allowed but produces an empty result; the normal use is to pass at least one of them.)
`species_col`	A length-1 character vector. Column in `data` containing species names. Auto-detected from a small set of common heuristics (e.g. `species`, `Species1`, `scientific_name`) when `NULL`; the heuristics list is not exhaustive — pass the column name explicitly if your data uses a non-standard label.
`drop_unresolved`	Logical. Drops unmatched rows and tips when `TRUE`. Defaults to `FALSE` (keep everything and just warn). Set to `TRUE` when preparing data for an analysis that cannot tolerate mismatches.

Details

Rows in `data` whose species have no match in the tree (and tips in `tree` whose species have no match in the data) are handled according to `drop_unresolved`. Matched data rows are kept as-is. Matched tree tips are renamed to the source-x (data-side) name when the tree-side label differs, so downstream PCM software can look up tips by the species names in your data frame.

Value

A list with two elements:

`data` The aligned data frame (or `NULL` if `data` was not supplied).

`tree` The aligned `phylo` object (or `NULL` if `tree` was not supplied).

See Also

`reconcile_tree()` to build the reconciliation; `reconcile_merge()` when you want a single merged data frame instead of aligned data + tree; `reconcile_export()` to write everything to disk.

Other reconciliation functions: `reconcile_augment()`, `reconcile_crosswalk()`, `reconcile_data()`, `reconcile_diff()`, `reconcile_export()`, `reconcile_mapping()`, `reconcile_merge()`, `reconcile_multi()`, `reconcile_override()`, `reconcile_override_batch()`, `reconcile_plot()`, `reconcile_report()`, `reconcile_review()`, `reconcile_splits_lumps()`, `reconcile_suggest()`, `reconcile_summary()`, `reconcile_to_trees()`, `reconcile_tree()`, `reconcile_trees()`

Examples

```
data(avonet_subset)
data(tree_jetz)
rec <- reconcile_tree(avonet_subset, tree_jetz,
                      x_species = "Species1", authority = NULL)

aligned <- reconcile_apply(rec,
                           data = avonet_subset,
                           tree = tree_jetz,
                           species_col = "Species1",
                           drop_unresolved = TRUE)
nrow(aligned$data)
ape::Ntip(aligned$tree)

# aligned$data and aligned$tree are ready for downstream PCM tools
```

---

reconcile_augment *Graft missing species onto a phylogenetic tree (genus-level placement)*

Description

When a reconciliation identifies species that are present in your data but missing from the tree, `reconcile_augment()` attaches each missing species as sister to a congener — i.e., a species in the same genus already present in the tree. The result is a tree that contains every species in your dataset, at the cost of making a strong assumption about where the new tips sit.

Usage

```
reconcile_augment(
  reconciliation,
  tree,
  where = c("genus", "near"),
  branch_length = c("congener_median", "half_terminal", "zero"),
  seed = NULL,
  quiet = FALSE,
  source = c("internal", "rtrees", "vphylomaker", "uphylomaker"),
  taxon = NULL,
  check_ultrametric = TRUE,
  ...
)
```

Arguments

`reconciliation`	A reconciliation object, typically from `reconcile_tree()`.
`tree`	An `ape::phylo` object. Must be the same tree used to build `reconciliation` (or a tree with the same tip set). For `source = "rtrees"`, this is passed to `rtrees` as the user-supplied backbone (`tree_by_user = TRUE`).
`where`	A length-1 character vector. Where to attach each new tip (only used when `source = "internal"`; ignored otherwise): `"genus"` **(default)** Attach as sister to a single congener chosen at random from the genus. Recommended when the genus has only one or two representatives in the tree, or when you want variation across runs for sensitivity analyses. `"near"` Attach at the most recent common ancestor (MRCA) of all congeners in the tree. Better when the genus is well-represented, because the new tip is not arbitrarily tied to one sister taxon.
`branch_length`	A length-1 character vector. How to set the terminal branch length of each newly added tip (only used when `source = "internal"`; ignored otherwise — `rtrees` sets its own branch lengths): `"congener_median"` **(default)** Median terminal branch length of the species' congeners. Uses the average "how long since this group diverged" for the genus. Recommended for time-calibrated trees because it preserves approximate branch-length scale. `"half_terminal"` Half the sister tip's terminal branch. A conservative alternative that places the new tip as a recent split from its sister. Useful when the genus is sparsely sampled and the median is unreliable. `"zero"` Zero-length branch, producing a polytomy with the sister taxon (or MRCA). Use for exploratory sensitivity checks where you want to see the effect of adding species without assuming any divergence time. When the input tree is ultrametric, each grafted tip's terminal edge is adjusted after placement so the augmented tree stays ultrametric — a requirement of phylogenetic comparative methods. `branch_length` then governs the initial graft only; `"zero"` is exempt, since it asks for a polytomy by construction.

seed — A length-1 integer or NULL. When non-NULL and source = "internal", a fixed seed for the random congener choice when where = "genus", making the call reproducible. When NULL (default), the session's current RNG state is used so results vary across runs — useful for sensitivity analyses that explore the variation introduced by the random choice. Set to a fixed integer in real analyses so results are reproducible. The seed is scoped to this call: the session RNG state is saved before and restored after, so subsequent random draws in your script are unaffected. Default NULL. (For source = "rtrees", set the seed in your script before calling reconcile_augment(); rtrees does not accept a seed argument.)

quiet — Logical. Suppress progress messages? Default FALSE.

source — A length-1 character vector. Which grafting backend to use. One of "internal" (default), "rtrees", or "vphylomaker". See "Choosing a source".

taxon — A length-1 character vector. Required when source = "rtrees". One of "bird", "mammal", "fish", "amphibian", "reptile", "plant", "shark_ray", "bee", "butterfly". Ignored for "internal" and "vphylomaker".

check_ultrametric — Logical. After grafting, check that the result is ultrametric and warn if not. Default TRUE. The "rtrees", "vphylomaker", and "uphylomaker" backends produce ultrametric trees by design; the "internal" backend does too when the input tree was ultrametric and branch_length is "congener_median" or "half_terminal", but not when branch_length = "zero" (which produces zero-length tip edges that break ultrametricity by construction).

... — Additional arguments forwarded to the chosen backend: rtrees::get_tree() for source = "rtrees" (e.g. scenario, n_tree); V.PhyloMaker2::phylo.maker() for source = "vphylomaker" (e.g. scenarios = "S3", nodes.type); U.PhyloMaker::phylo.maker() for source = "uphylomaker" (e.g. gen.list, scenario). Ignored when source = "internal".

Value

A list with:

tree The augmented phylo object (or multiPhylo when source = "rtrees" returns a posterior sample).

original The original (unmodified) phylo object, for easy comparison.

augmented A tibble documenting each added species: species, genus, placed_near (sister tip / MRCA node / rtrees placement note), branch_length, method, n_congeners. For source = "rtrees", branch_length and n_congeners are NA because the backend chooses them.

skipped A tibble of species that could not be placed, with the reason (e.g. "No congener in tree", "rtrees did not place this species").

meta Provenance metadata: source, placement strategy, branch length rule, counts; for source = "rtrees" includes a backend_meta sub-list with the taxon and the number of grafted tips.

When to use this

Tip-grafting is an *exploratory* convenience, not a substitute for a properly inferred phylogeny. Both source modes (see below) make strong placement assumptions that are often wrong in detail. Use it to keep exploratory PCMs running while you decide how to handle orphan species, and always:

1. Report exactly which species were augmented (see `$augmented` in the return value).
2. Run sensitivity analyses with and without the augmented tips.
3. Prefer a published imputed phylogeny (e.g. the PhyloMaker or TACT approaches) when grafting many species.

Choosing a source

`"internal"` **(default)** Genus-level placement using only your tree (no external dependencies). Each missing species is attached as sister to a congener (or at the congeneric MRCA). Fast and reproducible, but only works when the genus is already represented in the tree, and assumes the new tip diverged in roughly the same way as its congeners.

`"rtrees"` Delegates the grafting to the `rtrees` mega-tree machinery via `rtrees::get_tree(tree_by_user = TRUE)`. Uses your tree as the backbone and lets `rtrees` place each missing species using genus / family information from a taxon-specific reference tree. Requires `taxon` and the GitHub-only `rtrees` package (https://daijiang.github.io/rtrees/). Helpful when the genus is absent from your tree but present in `rtrees`' reference — which the internal mode would skip.

`"vphylomaker"` Plant-only alternative to `"rtrees"` via either of the GitHub packages **V.PhyloMaker2** (https://github.com/jinyizju/V.PhyloMaker2, preferred when installed; updated and enlarged version) or **V.PhyloMaker** (https://github.com/jinyizju/V.PhyloMaker, used as a fallback; original 2019 version). Calls `phylo.maker(sp.list, tree, scenarios = ...)` with your tree as the backbone. Use this when you want explicit control over the V.PhyloMaker placement scenario (`"S1"`, `"S2"`, or `"S3"` — see Jin & Qian 2019/2022); otherwise `"rtrees"` with `taxon = "plant"` is simpler.

`"uphylomaker"` Universal (plants + animals) variant of V.PhyloMaker, via the GitHub package **U.PhyloMaker** (https://github.com/jinyizju/U.PhyloMaker). Same `phylo.maker` convention but takes a `gen.list` (a genus-family lookup) so it can graft non-plant taxa as well as plants. Use this when your tree spans multiple kingdoms and you want the V.PhyloMaker placement strategy.

Use `pr_get_tree()` when you have only a species list and need a candidate tree from scratch (rotl, clootl, or rtrees). Use `reconcile_augment()` when you already have a tree and want to fill the gaps.

References

Paradis, E. & Schliep, K. (2019). ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R. *Bioinformatics* 35: 526–528. doi:10.1093/bioinformatics/bty633

Augmentation backends:

Jin, Y. & Qian, H. (2019). V.PhyloMaker: an R package that can generate very large phylogenies for vascular plants. *Ecography* 42(8): 1353–1359. doi:10.1111/ecog.04434 (`source = "vphylomaker"`, fallback path.)

Jin, Y. & Qian, H. (2022). V.PhyloMaker2: an updated and enlarged R package that can generate very large phylogenies for vascular plants. *Plant Diversity* 44(4): 335–339. doi:10.1016/j.pld.2022.05.005 (`source = "vphylomaker"`, preferred path.)

Jin, Y. & Qian, H. (2023). U.PhyloMaker: an R package that can generate large phylogenetic trees for plants and animals. *Plant Diversity* 45(3): 347–352. doi:10.1016/j.pld.2022.12.007 (`source = "uphylomaker"`.)

See Also

`reconcile_tree()` for the reconciliation step; `reconcile_apply()` for the non-augmenting alternative (prune data and tree to the intersection); `pr_get_tree()` for retrieving a candidate tree from external resources when you don't have a tree yet; `pr_date_tree()` for time-calibrating an existing topology; `pr_cite_tree()` for formatting tree provenance citations. The companion package pigauto consumes the resulting tree (or `multiPhylo`) directly via `multi_impute_trees()` for posterior-tree PCMs.

Other reconciliation functions: `reconcile_apply()`, `reconcile_crosswalk()`, `reconcile_data()`, `reconcile_diff()`, `reconcile_export()`, `reconcile_mapping()`, `reconcile_merge()`, `reconcile_multi()`, `reconcile_override()`, `reconcile_override_batch()`, `reconcile_plot()`, `reconcile_report()`, `reconcile_review()`, `reconcile_splits_lumps()`, `reconcile_suggest()`, `reconcile_summary()`, `reconcile_to_trees()`, `reconcile_tree()`, `reconcile_trees()`

Examples

```
# --- Example 1: genus-level placement with congener_median branch lengths ---
x <- data.frame(species = c("A a", "A missing", "B c", "C absent"))
tree <- ape::read.tree(text = "((A_a:1,A_b:1):1,B_c:2);")
result <- reconcile_tree(x, tree, x_species = "species",
                         authority = NULL, quiet = TRUE)

aug <- reconcile_augment(result, tree, seed = 42, quiet = TRUE)

# Compare original vs augmented tree
cat("Original tips:", ape::Ntip(tree), "\n")
cat("Augmented tips:", ape::Ntip(aug$tree), "\n")
cat("Added:", nrow(aug$augmented), "| Skipped:", nrow(aug$skipped), "\n")

# Inspect which species were added and where they were placed
head(aug$augmented[, c("species", "genus", "placed_near",
                       "branch_length", "n_congeners")])

# Species skipped (no congener in tree)
head(aug$skipped)

# --- Example 2: MRCA placement with zero-length branches ---
aug_near <- reconcile_augment(result, tree,
                              where = "near",
                              branch_length = "zero",
                              seed = 42, quiet = TRUE)

cat("\nMRCA placement (zero branches):\n")
cat("  Added:", nrow(aug_near$augmented), "\n")
```

```
# Compare: MRCA placement shows genus-level context
head(aug_near$augmented[, c("species", "placed_near", "method")])


  # --- Example 3: delegate grafting to rtrees ---
  # Useful when the genus is missing from your tree but present in
  # the rtrees taxon-specific reference tree.
  if (requireNamespace("rtrees", quietly = TRUE)) {
    aug_rt <- try(
      reconcile_augment(result, tree,
                        source = "rtrees",
                        taxon  = "bird",
                        quiet  = TRUE),
      silent = TRUE
    )
    if (!inherits(aug_rt, "try-error")) {
      nrow(aug_rt$augmented)              # how many were placed
      aug_rt$meta$backend_meta$n_grafted  # how many at higher rank
    }
  }
```

`reconcile_crosswalk` *Convert a published taxonomy crosswalk into an overrides table*

Description

Turn a curated species-name crosswalk (e.g. the BirdLife–BirdTree crosswalk bundled as crosswalk_birdlife_birdtree, or Clements updates released each year) into a data frame that can be passed straight to the `overrides` argument of `reconcile_tree()`, `reconcile_data()` and friends.

Usage

```
reconcile_crosswalk(
  crosswalk,
  from_col,
  to_col,
  match_type_col = NULL,
  notes_col = NULL,
  one_to_one_only = FALSE
)
```

Arguments

`crosswalk`	A data frame, or a file path. File format is inferred from the extension: `.csv` (comma-separated), `.tsv` (tab-separated), or `.txt` (tab-separated). For other delimited formats, read the file yourself with `read.delim()` or `read.table()` and pass the resulting data frame.

`from_col` A length-1 character vector. Column name for source names (e.g., `"Species1"` for BirdLife names).

`to_col` A length-1 character vector. Column name for target names (e.g., `"Species3"` for BirdTree names).

`match_type_col` A length-1 character vector or `NULL`. Name of an optional column in `crosswalk` that classifies each row's relationship between the two taxonomies — e.g. `"1BL to 1BT"` (one BirdLife species mapped to one BirdTree species; a clean one-to-one match), `"Many BL to 1BT"` (a lump: several BirdLife species mapped to a single BirdTree species), `"1BL to many BT"` (a split). When supplied, the contents of this column are appended to each override's `user_note` so the audit trail records the relationship; if you also pass `one_to_one_only = TRUE`, only the rows whose match type starts `"1... to 1..."` are kept. Pass `NULL` (default) when your crosswalk has no such classification column — every row is then kept and notes carry no provenance label.

`notes_col` A length-1 character vector or NULL. Column containing additional notes.

`one_to_one_only`
Logical. If `TRUE`, keeps only one-to-one matches (e.g., `"1BL to 1BT"`). Default `FALSE`.

Details

Using a crosswalk is preferable to automated synonym resolution when an authoritative mapping exists — it is reproducible, does not depend on **taxadb** being available, and you can point to the published source in the methods section of your paper.

Value

A data frame with columns `name_x`, `name_y`, and `user_note`, ready to be passed as the `overrides` argument.

See Also

`reconcile_override_batch()` for applying this table directly to an existing reconciliation; crosswalk_birdlife_birdtree for the bundled example.

Other reconciliation functions: `reconcile_apply()`, `reconcile_augment()`, `reconcile_data()`, `reconcile_diff()`, `reconcile_export()`, `reconcile_mapping()`, `reconcile_merge()`, `reconcile_multi()`, `reconcile_override()`, `reconcile_override_batch()`, `reconcile_plot()`, `reconcile_report()`, `reconcile_review()`, `reconcile_splits_lumps()`, `reconcile_suggest()`, `reconcile_summary()`, `reconcile_to_trees()`, `reconcile_tree()`, `reconcile_trees()`

Examples

```
data(crosswalk_birdlife_birdtree)
overrides <- reconcile_crosswalk(
  crosswalk_birdlife_birdtree,
  from_col = "Species1",
  to_col = "Species3",
  match_type_col = "Match.type"
)
```

```
head(overrides)
```

`reconcile_data` *Reconcile species names between two datasets*

Description

Match the species column of one data frame (`x`) to the species column of another (`y`), returning a reconciliation object that records how every name was resolved. Use this when combining trait datasets, range datasets, or any other species-level tables that may use slightly different taxonomies or spellings.

Usage

```
reconcile_data(
  x,
  y,
  x_species = NULL,
  y_species = NULL,
  authority = "col",
  rank = c("species", "subspecies"),
  overrides = NULL,
  db_version = NULL,
  fuzzy = FALSE,
  fuzzy_threshold = 0.9,
  flag_threshold = 0.95,
  resolve = c("flag", "first"),
  quiet = FALSE,
  x_label = NULL,
  y_label = NULL
)
```

Arguments

`x`	A data frame whose species will be matched *from*.
`y`	A data frame whose species will be matched *to* (typically the "reference" taxonomy or the dataset you want to merge with).
`x_species`	A length-1 character vector. Name of the column in `x` containing scientific names. Auto-detected (e.g. `species`, `Species1`, `scientific_name`) when `NULL`.
`y_species`	A length-1 character vector. Name of the column in `y` containing scientific names. Auto-detected when `NULL`.
`authority`	A length-1 character vector, or `NULL`. Taxonomic authority used for synonym resolution (stage 3 of the cascade). One of:

`"col"` **(default)** Catalogue of Life — broad, curated, frequently updated. A sensible default for most taxa.

`"itis"` Integrated Taxonomic Information System — strong for North American vertebrates and plants.

`"gbif"` Global Biodiversity Information Facility backbone. Wider coverage; includes more recent synonymy.

`"ncbi"` NCBI Taxonomy — best when working with sequence data.

`"ott"` Open Tree of Life synthetic taxonomy. Useful when your downstream phylogeny is from the Open Tree synthesis.

`"itis_test"` A small bundled subset of ITIS, cached locally with **taxadb** for testing. Intended for examples and unit tests; not for analysis.

`"gnverifier"` HTTP-backed verification against ~100 sources via the Global Names verifier; no local database download. See `vignette("getting-started")` for the trade-off (wider coverage, requires network and the **httr2** package).

`NULL` Skip the synonym stage entirely. Useful for quick checks or when **taxadb** is unavailable. Stages 1, 2 and 4 still run.

Five authority codes that earlier versions of the package advertised — `"iucn"`, `"tpl"`, `"fb"`, `"slb"`, `"wd"` — are no longer accepted. Empirical testing against **taxadb** v22.12 showed that `iucn` errors with a schema mismatch and the others are not **taxadb** providers at all. Passing one of those values now produces a helpful migration error.

`rank` A length-1 character vector. Controls how trinomials are handled during normalisation:

`"species"` **(default)** Strip infraspecific epithets so that `"Parus major major"` becomes `"Parus major"` before matching.

`"subspecies"` Keep trinomials intact. Use this when your analysis operates at subspecies level.

`overrides` Optional pre-built corrections. Either a data frame with at least columns `name_x` and `name_y` (plus an optional `user_note` column), or a file path to a CSV with the same columns. Any name listed here bypasses the cascade and is recorded as `match_type = "manual"`. Useful for applying published crosswalks (see `reconcile_crosswalk()`) or for locking down decisions made in a previous run.

`db_version` A length-1 character vector. **taxadb** database snapshot to use (e.g. `"22.12"`). `NULL` (default) uses the latest available.

`fuzzy` Logical. Enables the fuzzy-matching stage when `TRUE`. Default `FALSE`. Turn this on to catch likely typos (*Corvus brachyrhnchos* -> *Corvus brachyrhynchos*). When `FALSE`, stages 1–3 still run.

`fuzzy_threshold`

Numeric in [0, 1]. Minimum genus-weighted similarity score for a fuzzy match to be accepted. Default `0.9` (roughly "no more than ~10% of characters differ"). Lower values (e.g. `0.7`) are more permissive but produce more false positives; always review fuzzy matches with `reconcile_suggest()` or `reconcile_review()` before trusting them.

`flag_threshold` Numeric in [0, 1]. When `resolve = "flag"`, fuzzy matches with a score below this value are recorded as `match_type = "flagged"` rather than `"fuzzy"`, marking them for manual review. Default `0.95`. Must be `>= fuzzy_threshold` to have any effect.

`resolve` A length-1 character vector. What to do with borderline matches:

`"flag"` **(default)** Mark low-confidence fuzzy matches (score below `flag_threshold`) and names with indirect taxadb synonymy as `match_type = "flagged"` so you can audit them with `reconcile_review()` or `reconcile_suggest()`.

`"first"` Accept the highest-scoring candidate silently, without flagging. Faster but riskier; use only when you have already reviewed the ambiguities.

`quiet` Logical. Suppresses progress messages when `TRUE`. Default `FALSE`.

`x_label` A length-1 character vector or `NULL`. Human-readable label for source `x` stored in the reconciliation metadata and shown in `print()` / `format()`. Defaults to the expression passed as `x` (via `deparse(substitute())`). Set this explicitly when calling `reconcile_data()` inside another function so the label reflects the real data source rather than the local argument name.

`y_label` A length-1 character vector or `NULL`. Same as `x_label`, for source `y`.

Details

Names are passed through a four-stage matching cascade, and the first stage that returns a match is recorded in `match_type`:

1. **exact** — verbatim string equality.
2. **normalized** — after stripping underscores, authority strings (*"Corvus corax Linnaeus, 1758"*), diacritics, and case/whitespace differences.
3. **synonym** — lookup in a local taxonomic database via **taxadb** (Catalogue of Life, GBIF, ITIS, NCBI, ...). Skipped if `authority = NULL`.
4. **fuzzy** — character-level similarity (opt-in via `fuzzy = TRUE`). Uses a genus-weighted Levenshtein score (60% genus, 40% specific epithet) with a genus pre-filter so that only plausibly similar genera are compared.

Names that survive all four stages are labelled `unresolved`. Any entries supplied through `overrides` take precedence over the cascade.

After the call. A `reconciliation` object is the input to most other functions in the package. Common next steps:

- `reconcile_summary()` — human-readable breakdown of matches.
- `reconcile_plot()` — one-glance bar/pie of match composition.
- `reconcile_mapping()` — extract the full per-name tibble.
- `reconcile_suggest()` — near-miss candidates for unresolved names.
- `reconcile_merge()` — join the two datasets using the reconciliation as the species key.
- `reconcile_report()` — shareable HTML audit trail.

Value

A reconciliation object. The accompanying mapping tibble, match-type counts, provenance metadata, and applied / unused override slots are documented in reconciliation. See the "After the call" section above for the most common next steps.

References

Norman, K.E., Chamberlain, S. & Boettiger, C. (2020) taxadb: A high-performance local taxonomic database interface. *Methods in Ecology and Evolution* 11:1153–1159. doi:10.1111/2041-210X.13440

See Also

reconcile_tree() for matching against a phylogenetic tree; reconcile_to_trees() / reconcile_trees() / reconcile_multi() for multi-input workflows.

Other reconciliation functions: reconcile_apply(), reconcile_augment(), reconcile_crosswalk(), reconcile_diff(), reconcile_export(), reconcile_mapping(), reconcile_merge(), reconcile_multi(), reconcile_override(), reconcile_override_batch(), reconcile_plot(), reconcile_report(), reconcile_review(), reconcile_splits_lumps(), reconcile_suggest(), reconcile_summary(), reconcile_to_trees(), reconcile_tree(), reconcile_trees()

Examples

```
# Merge AVONET morphology with nest-site data. Both datasets use
# slightly different taxonomies; authority = NULL keeps the example
# offline (no taxadb download).
data(avonet_subset)
data(nesttrait_subset)

rec <- reconcile_data(avonet_subset, nesttrait_subset,
                      x_species = "Species1",
                      y_species = "Scientific_name",
                      authority = NULL)
rec                      # concise print method
reconcile_summary(rec)   # full breakdown

# Join the two datasets on the reconciled species key
merged <- reconcile_merge(rec, avonet_subset, nesttrait_subset,
                          species_col_x = "Species1",
                          species_col_y = "Scientific_name")
head(merged[, c("species_resolved", "Family1", "Common_name")])
```

`reconcile_diff` *Diff two reconciliations to see what changed*

Description

Compare a "before" and "after" reconciliation and list every species whose outcome differs: newly matched, newly unresolved, promoted to a higher-confidence match type, or linked to a different target. Useful for:

- checking the effect of adding a taxonomy crosswalk or a batch of manual overrides,
- comparing two taxonomic authorities (e.g. Catalogue of Life vs GBIF),
- auditing changes between runs before and after tightening the fuzzy threshold.

Usage

```
reconcile_diff(x, y, quiet = FALSE)
```

Arguments

x	A reconciliation object — the "before" state.
y	A reconciliation object — the "after" state. Must be reconciled against the same x data so that name_x values are comparable.
quiet	Logical. Suppresses the console summary when TRUE. Default FALSE.

Value

A list with the following components:

gained Tibble of species matched in y but unresolved in x.

lost Tibble of species matched in x but unresolved in y.

type_changed Tibble of species whose match_type differs between the two runs.

target_changed Tibble of species whose name_y differs.

unused_overrides_diff Tibble of overrides that are in the unused_overrides slot of one reconciliation but not the other; columns name_x, name_y, reason, side ("x" or "y").

summary A one-row tibble with counts: n_gained, n_lost, n_type_changed, n_target_changed, n_shared, n_unused_override_diff.

See Also

reconcile_crosswalk() for building an override table from a published taxonomy crosswalk; reconcile_override_batch() for applying many hand edits.

Other reconciliation functions: reconcile_apply(), reconcile_augment(), reconcile_crosswalk(), reconcile_data(), reconcile_export(), reconcile_mapping(), reconcile_merge(), reconcile_multi(), reconcile_override(), reconcile_override_batch(), reconcile_plot(), reconcile_report(), reconcile_review(), reconcile_splits_lumps(), reconcile_suggest(), reconcile_summary(), reconcile_to_trees(), reconcile_tree(), reconcile_trees()

Examples

```
x <- data.frame(species = c("A a", "A old", "B c"))
tree <- ape::read.tree(text = "((A_a:1,A_new:1):1,B_c:2);")

# Without manual overrides
r1 <- reconcile_tree(x, tree, x_species = "species",
                     authority = NULL, quiet = TRUE)

# With one manual override
overrides <- data.frame(name_x = "A old", name_y = "A new",
                        match_type = "manual")
r2 <- reconcile_tree(x, tree, x_species = "species",
                     authority = NULL, overrides = overrides,
                     quiet = TRUE)

d <- reconcile_diff(r1, r2, quiet = TRUE)
cat("Gained:", nrow(d$gained), "| Lost:", nrow(d$lost), "\n")
```

---

reconcile_export *Write an aligned dataset, tree, and mapping table to disk*

Description

Apply a reconciliation and save three files: the aligned CSV, the pruned tree, and the full mapping tibble. Intended for producing analysis-ready, archivable outputs — drop the three files into a Zenodo deposit or a project's `data-output/` folder alongside the reconciliation report and you have a fully documented provenance trail.

Usage

```
reconcile_export(
  reconciliation,
  data = NULL,
  tree = NULL,
  species_col = NULL,
  dir = tempfile("prepR4pcm-export-"),
  prefix = "reconciled",
  tree_format = c("nexus", "newick"),
  drop_unresolved = TRUE
)
```

Arguments

`reconciliation`	A reconciliation object returned by `reconcile_tree()`, `reconcile_data()`, or a related matcher.
`data`	A data frame to align. If `NULL`, only the tree and mapping are written.

`tree` An `ape::phylo` object or file path. If `NULL`, only the data and mapping are written.

`species_col` A length-1 character vector. Column name in `data` containing species names. Auto-detected when `NULL`.

`dir` A length-1 character vector. Path to the output directory that will receive the exported files (e.g. a project's `data-output/` folder, or a staging directory before a Zenodo deposit). Created if it does not exist. By default, a unique temporary directory is used so the function does not write to the current working directory unless you explicitly request it.

`prefix` A length-1 character vector. File name prefix. Default `"reconciled"`.

`tree_format` A length-1 character vector. Tree output format: `"nexus"` (default) or `"newick"`.

`drop_unresolved`
Logical. Drops unresolved species when `TRUE`. Default `TRUE`.

Value

A named list of file paths (invisibly): `$data` (CSV), `$tree` (Nexus or Newick), `$mapping` (CSV), and `$unused_overrides` (CSV; `NULL` when there are no rejected overrides on the reconciliation).

See Also

`reconcile_apply()` for in-memory alignment without writing to disk; `reconcile_report()` for a self-contained HTML audit trail.

Other reconciliation functions: `reconcile_apply()`, `reconcile_augment()`, `reconcile_crosswalk()`, `reconcile_data()`, `reconcile_diff()`, `reconcile_mapping()`, `reconcile_merge()`, `reconcile_multi()`, `reconcile_override()`, `reconcile_override_batch()`, `reconcile_plot()`, `reconcile_report()`, `reconcile_review()`, `reconcile_splits_lumps()`, `reconcile_suggest()`, `reconcile_summary()`, `reconcile_to_trees()`, `reconcile_tree()`, `reconcile_trees()`

Examples

```
data(avonet_subset)
data(tree_jetz)
result <- reconcile_tree(avonet_subset, tree_jetz,
                         x_species = "Species1", authority = NULL)
out_dir <- tempfile("export_")
files <- reconcile_export(result,
                          data = avonet_subset, tree = tree_jetz,
                          species_col = "Species1",
                          dir = out_dir, prefix = "avonet_jetz")
files$data     # path to CSV
files$tree     # path to Nexus tree
files$mapping  # path to mapping CSV
unlink(out_dir, recursive = TRUE)  # clean up
```

`reconcile_mapping` *Extract the per-name mapping table from a reconciliation*

Description

Returns the mapping tibble inside a reconciliation object. Use this when you want to filter matches programmatically (e.g. pull all unresolved species, all fuzzy matches above a given score, or join the mapping back to the original data frame).

Usage

```
reconcile_mapping(reconciliation, include_unused_overrides = FALSE)
```

Arguments

`reconciliation`	A reconciliation object returned by `reconcile_tree()`, `reconcile_data()`, `reconcile_trees()`, `reconcile_to_trees()`, or `reconcile_multi()`.
`include_unused_overrides`	Logical. Appends the rejected override rows to the returned tibble when `TRUE`, with `match_type = "override_unused"`, `match_score = NA`, and the `notes` column carrying the rejection reason (`name_x_not_in_data`, `name_y_not_in_target`, or `already_matched`). Default `FALSE` for backward compatibility — the bare mapping tibble has the same shape as before.

Value

A tibble with one row per unique name seen in either source and the following columns:

`name_x` Statement: this column holds the original name as it appeared in source `x` (your data). `NA` for rows that exist only in source `y` (e.g. tree tips not in your data).

`name_y` Statement: this column holds the original name as it appeared in source `y` (the reference dataset or tree). `NA` for rows that exist only in source `x`.

`name_resolved` The accepted/canonical name returned by the taxonomic authority, when synonym resolution was used. `NA` when `authority = NULL` or no synonym was found.

`match_type` One of `"exact"`, `"normalized"`, `"synonym"`, `"fuzzy"`, `"manual"` (set via `reconcile_override()`), `"flagged"` (low-confidence, needs review), `"unresolved"`, or — when `include_unused_overrides = TRUE` — `"override_unused"` (override row not applied because of missing names or prior matches).

`match_score` Numeric in [0, 1]. `1` for exact/normalized/synonym/manual matches; a genus-weighted Levenshtein score for fuzzy matches; `NA` for unresolved and for unused-override rows.

`match_source` Where the match came from: `"exact"`, `"normalisation"`, the taxadb authority code (e.g. `"col"`), `"fuzzy"`, or `"user_override"`.

`in_x` Logical. This column records whether the name was present in source `x`.

`in_y` Logical. This column records whether the name was present in source `y`.

`notes` Free-text notes, populated e.g. when a name is flagged for review or when an override carries a user comment. For `match_type = "override_unused"` rows this column carries the rejection reason.

See Also

reconcile_summary() for a printed breakdown; reconcile_suggest() for near-miss candidates for unresolved names; reconcile_apply() to turn the mapping into an aligned data-tree pair. The unused-override rows surfaced by include_unused_overrides = TRUE mirror the unused_overrides slot on the reconciliation object.

Other reconciliation functions: reconcile_apply(), reconcile_augment(), reconcile_crosswalk(), reconcile_data(), reconcile_diff(), reconcile_export(), reconcile_merge(), reconcile_multi(), reconcile_override(), reconcile_override_batch(), reconcile_plot(), reconcile_report(), reconcile_review(), reconcile_splits_lumps(), reconcile_suggest(), reconcile_summary(), reconcile_to_trees(), reconcile_tree(), reconcile_trees()

Examples

```
data(avonet_subset)
data(tree_jetz)
rec <- reconcile_tree(avonet_subset, tree_jetz,
                      x_species = "Species1", authority = NULL)
mapping <- reconcile_mapping(rec)

# How many species matched?
sum(mapping$in_x & mapping$in_y)

# Which species are in the data but missing from the tree?
head(mapping[mapping$in_x & !mapping$in_y, c("name_x", "match_type")])

# Append rejected overrides for audit
mapping_full <- reconcile_mapping(rec, include_unused_overrides = TRUE)
```

---

reconcile_merge *Merge two reconciled datasets*

Description

After reconciling two datasets with reconcile_data(), use this function to join them into a single analysis-ready data frame. The reconciliation mapping table provides the species-level join key, so names that differ between the two datasets (due to formatting, synonyms, or typos) are correctly linked.

Usage

```
reconcile_merge(
  reconciliation,
  data_x,
  data_y,
  species_col_x = NULL,
  species_col_y = NULL,
```

```
  how = c("inner", "left", "full"),
  suffix = c("_x", "_y"),
  drop_unresolved = FALSE
)
```

Arguments

`reconciliation`	A reconciliation object (typically from `reconcile_data()`).
`data_x`	The first data frame (source x in the reconciliation).
`data_y`	The second data frame (source y in the reconciliation).
`species_col_x`	A length-1 character vector. Species column in `data_x`. Auto-detected if `NULL`.
`species_col_y`	A length-1 character vector. Species column in `data_y`. Auto-detected if `NULL`.
`how`	A length-1 character vector. Join type: • `"inner"` (default): keep only species matched in both datasets. • `"left"`: keep all species from `data_x`. • `"full"`: keep all species from both datasets.
`suffix`	A length-2 character vector. Suffixes to disambiguate columns with the same name in both datasets. Default `c("_x", "_y")`.
`drop_unresolved`	Logical. If `TRUE`, rows where `species_resolved` is `NA` (i.e., species that could not be reconciled) are removed from the final result. Default `FALSE` (keep all rows, fill unmatched columns with `NA`). Only relevant for `how = "left"` or `how = "full"`; inner joins drop unmatched rows by definition.

Details

One row per species. `reconcile_merge()` works best when each dataset has exactly one row per species. If a species appears in multiple rows (e.g., sex-specific measurements, repeated populations), the merge produces all pairwise combinations for that species—the same behaviour as base `merge()`. To avoid unexpected row expansion, aggregate to one row per species before merging, or be aware that the output will contain more rows than either input.

Asymmetric datasets. When `data_y` contains many more species than `data_x` (common when merging against a large reference database), use `how = "inner"` or `how = "left"`. Inner joins keep only the species present in both datasets; left joins keep all `data_x` rows and fill `data_y` columns with `NA` for unmatched species. Use `how = "full"` only when you need to retain species unique to either side.

Recommended workflow for multi-row data. Reconcile using a species-level summary (one row per species), inspect the mapping with `reconcile_mapping()`, then join the mapping back to your full dataset using the species column as key.

Value

A data frame with a `species_resolved` column as the join key, plus all columns from both datasets (with suffixes added when column names collide).

See Also

reconcile_data() to build the reconciliation; reconcile_apply() when you want aligned data + tree instead of a single merged data frame.

Other reconciliation functions: reconcile_apply(), reconcile_augment(), reconcile_crosswalk(), reconcile_data(), reconcile_diff(), reconcile_export(), reconcile_mapping(), reconcile_multi(), reconcile_override(), reconcile_override_batch(), reconcile_plot(), reconcile_report(), reconcile_review(), reconcile_splits_lumps(), reconcile_suggest(), reconcile_summary(), reconcile_to_trees(), reconcile_tree(), reconcile_trees()

Examples

```
data(avonet_subset)
data(nesttrait_subset)

rec <- reconcile_data(avonet_subset, nesttrait_subset,
                      x_species = "Species1",
                      y_species = "Scientific_name",
                      authority = NULL, quiet = TRUE)

merged <- reconcile_merge(rec, avonet_subset, nesttrait_subset,
                          species_col_x = "Species1",
                          species_col_y = "Scientific_name")
cat(sprintf("Merged: %d rows, %d cols\n", nrow(merged), ncol(merged)))
head(merged[, c("species_resolved", "Family1", "Common_name")])
```

---

reconcile_multi *Reconcile several datasets against one phylogenetic tree*

Description

Match several trait or occurrence datasets against a single phylogenetic tree in one call. Species that appear in more than one dataset are reconciled once; the combined mapping records which dataset(s) each species belongs to, making it easy to identify the set of species with complete trait coverage.

Usage

```
reconcile_multi(
  datasets,
  tree,
  species_cols = NULL,
  authority = "col",
  rank = c("species", "subspecies"),
  overrides = NULL,
  db_version = NULL,
  fuzzy = FALSE,
```

```
  fuzzy_threshold = 0.9,
  resolve = c("flag", "first"),
  quiet = FALSE
)
```

Arguments

datasets	A named list of data frames. The names are used as dataset labels (e.g. morpho, nests, plumage) in the output.
tree	An ape::phylo object, or a path to a Newick/Nexus file.
species_cols	Character vector. Species column name in each dataset. If length 1, the same column name is used for every dataset. Auto-detected from each data frame if NULL.
authority	A length-1 character vector, or NULL. Taxonomic authority used for synonym resolution (stage 3 of the cascade). One of: "col" **(default)** Catalogue of Life — broad, curated, frequently updated. A sensible default for most taxa. "itis" Integrated Taxonomic Information System — strong for North American vertebrates and plants. "gbif" Global Biodiversity Information Facility backbone. Wider coverage; includes more recent synonymy. "ncbi" NCBI Taxonomy — best when working with sequence data. "ott" Open Tree of Life synthetic taxonomy. Useful when your downstream phylogeny is from the Open Tree synthesis. "itis_test" A small bundled subset of ITIS, cached locally with **taxadb** for testing. Intended for examples and unit tests; not for analysis. "gnverifier" HTTP-backed verification against ~100 sources via the Global Names verifier; no local database download. See vignette("getting-started") for the trade-off (wider coverage, requires network and the **httr2** package). NULL Skip the synonym stage entirely. Useful for quick checks or when **taxadb** is unavailable. Stages 1, 2 and 4 still run. Five authority codes that earlier versions of the package advertised — "iucn", "tpl", "fb", "slb", "wd" — are no longer accepted. Empirical testing against **taxadb** v22.12 showed that iucn errors with a schema mismatch and the others are not **taxadb** providers at all. Passing one of those values now produces a helpful migration error.
rank	A length-1 character vector. Controls how trinomials are handled during normalisation: "species" **(default)** Strip infraspecific epithets so that "Parus major major" becomes "Parus major" before matching. "subspecies" Keep trinomials intact. Use this when your analysis operates at subspecies level.
overrides	Optional pre-built corrections. Either a data frame with at least columns name_x and name_y (plus an optional user_note column), or a file path to a CSV with the same columns. Any name listed here bypasses the cascade and is recorded

as `match_type = "manual"`. Useful for applying published crosswalks (see `reconcile_crosswalk()`) or for locking down decisions made in a previous run.

`db_version` A length-1 character vector. **taxadb** database snapshot to use (e.g. `"22.12"`). `NULL` (default) uses the latest available.

`fuzzy` Logical. Enables the fuzzy-matching stage when `TRUE`. Default `FALSE`. Turn this on to catch likely typos (*Corvus brachyrhnchos* -> *Corvus brachyrhynchos*). When `FALSE`, stages 1–3 still run.

`fuzzy_threshold`
Numeric in [0, 1]. Minimum genus-weighted similarity score for a fuzzy match to be accepted. Default `0.9` (roughly "no more than ~10% of characters differ"). Lower values (e.g. `0.7`) are more permissive but produce more false positives; always review fuzzy matches with `reconcile_suggest()` or `reconcile_review()` before trusting them.

`resolve` A length-1 character vector. What to do with borderline matches:

`"flag"` **(default)** Mark low-confidence fuzzy matches (score below `flag_threshold`) and names with indirect taxadb synonymy as `match_type = "flagged"` so you can audit them with `reconcile_review()` or `reconcile_suggest()`.

`"first"` Accept the highest-scoring candidate silently, without flagging. Faster but riskier; use only when you have already reviewed the ambiguities.

`quiet` Logical. Suppresses progress messages when `TRUE`. Default `FALSE`.

Value

A reconciliation object. The mapping tibble gains one logical column per input dataset (e.g. `in_morpho`, `in_nests`) indicating which datasets contained each species.

See Also

`reconcile_tree()` for the single-dataset case; `reconcile_merge()` to join two datasets after reconciliation.

Other reconciliation functions: `reconcile_apply()`, `reconcile_augment()`, `reconcile_crosswalk()`, `reconcile_data()`, `reconcile_diff()`, `reconcile_export()`, `reconcile_mapping()`, `reconcile_merge()`, `reconcile_override()`, `reconcile_override_batch()`, `reconcile_plot()`, `reconcile_report()`, `reconcile_review()`, `reconcile_splits_lumps()`, `reconcile_suggest()`, `reconcile_summary()`, `reconcile_to_trees()`, `reconcile_tree()`, `reconcile_trees()`

Examples

```
data(avonet_subset)
data(nesttrait_subset)
data(tree_jetz)
datasets <- list(
  morpho = avonet_subset,
  nests  = nesttrait_subset
)
result <- reconcile_multi(datasets, tree_jetz,
                          species_cols = c("Species1", "Scientific_name"),
```

```
                          authority = NULL)
print(result)
```

`reconcile_override` *Manually override a single name in a reconciliation*

Description

Apply a single hand-curated decision to a reconciliation object. Use this to accept a match the matching cascade rejected (typically a flagged fuzzy hit), remove a spurious match, or force a new mapping that the cascade missed. ("Cascade" here means the four-stage matching pipeline run by `reconcile_tree()` and `reconcile_data()` — exact, normalised, synonym, fuzzy — as described in ?prepR4pcm.) The override is recorded in the provenance log so that you and your reviewers can audit every manual decision.

Usage

```
reconcile_override(
  reconciliation,
  name_x,
  name_y = NULL,
  action = c("accept", "reject", "replace"),
  note = ""
)
```

Arguments

`reconciliation`	A reconciliation object.
`name_x`	A length-1 character vector. The name as it appears in source `x` (your data). Must match a value already present in `mapping$name_x`.
`name_y`	A length-1 character vector or `NULL`. The name in source `y` (the tree or reference dataset) that `name_x` should be mapped to. `NULL` is only valid when `action = "reject"`.
`action`	A length-1 character vector. What the override does: `"accept"` **(default)** Confirm a proposed match. Use after reviewing a flagged fuzzy or synonym hit. `"reject"` Remove an existing match and return both names to the unresolved pool. Use when the cascade over-matched (e.g. an aggressive fuzzy score linked the wrong species). `"replace"` Set a new match, overwriting whatever the cascade produced for `name_x`.
`note`	A length-1 character vector. A short justification for the override, stored in the provenance log and in `mapping$notes`. Strongly recommended — future you will want to know why this decision was made.

Details

For applying many overrides at once (e.g. from a curated CSV), see `reconcile_override_batch()`; for interactive decisions in the console, see `reconcile_review()`; for published taxonomy crosswalks, see `reconcile_crosswalk()`.

Value

An updated reconciliation object. The existing row for `name_x` is replaced with one whose `match_type` is `"manual"` and `match_source` is `"user_override"`.

See Also

`reconcile_override_batch()` for bulk overrides; `reconcile_suggest()` for near-miss candidates; `reconcile_crosswalk()` for published taxonomy crosswalks.

Other reconciliation functions: `reconcile_apply()`, `reconcile_augment()`, `reconcile_crosswalk()`, `reconcile_data()`, `reconcile_diff()`, `reconcile_export()`, `reconcile_mapping()`, `reconcile_merge()`, `reconcile_multi()`, `reconcile_override_batch()`, `reconcile_plot()`, `reconcile_report()`, `reconcile_review()`, `reconcile_splits_lumps()`, `reconcile_suggest()`, `reconcile_summary()`, `reconcile_to_trees()`, `reconcile_tree()`, `reconcile_trees()`

Examples

```
data(avonet_subset)
data(tree_jetz)
rec <- reconcile_tree(avonet_subset, tree_jetz,
                      x_species = "Species1", authority = NULL)

# Pick an unresolved species and hand-assign it for illustration
unresolved <- reconcile_mapping(rec)
unresolved <- unresolved[unresolved$match_type == "unresolved" &
                           unresolved$in_x, ]
if (nrow(unresolved) > 0) {
  rec <- reconcile_override(
    rec,
    name_x = unresolved$name_x[1],
    name_y = tree_jetz$tip.label[1],
    note   = "Demo: manual assignment"
  )
}
```

`reconcile_override_batch` *Apply many manual corrections to a reconciliation at once*

Description

A convenience wrapper around `reconcile_override()` for curated batches of manual decisions.

Usage

```
reconcile_override_batch(reconciliation, overrides, quiet = FALSE)
```

Arguments

reconciliation	A reconciliation object returned by reconcile_tree(), reconcile_data(), or a related matcher.
overrides	A data frame, or a length-1 character vector giving the path to a CSV file with the same columns: name_x **(required)** The original name in source x (your data). action One of "accept" (default), "reject", "replace". See reconcile_override() for the semantics. name_y The target name in source y; required for "accept" and "replace". note Optional free-text justification.
quiet	Logical. Suppresses per-override success messages when TRUE. Default FALSE.

Details

Typical workflow: generate a CSV of corrections (by hand, or with the help of reconcile_suggest()), check it into version control, and apply it on every run so the corrections are reproducible and reviewable.

Value

An updated reconciliation object with all overrides applied.

See Also

reconcile_override() for the single-override case; reconcile_crosswalk() for building an override table from a published taxonomy crosswalk.

Other reconciliation functions: reconcile_apply(), reconcile_augment(), reconcile_crosswalk(), reconcile_data(), reconcile_diff(), reconcile_export(), reconcile_mapping(), reconcile_merge(), reconcile_multi(), reconcile_override(), reconcile_plot(), reconcile_report(), reconcile_review(), reconcile_splits_lumps(), reconcile_suggest(), reconcile_summary(), reconcile_to_trees(), reconcile_tree(), reconcile_trees()

Examples

```
data(avonet_subset)
data(tree_jetz)
result <- reconcile_tree(avonet_subset, tree_jetz,
                         x_species = "Species1", authority = NULL)
# Create a batch of overrides
batch <- data.frame(
  name_x = reconcile_mapping(result)$name_x[
    reconcile_mapping(result)$match_type == "unresolved" &
    reconcile_mapping(result)$in_x][1:2],
  name_y = tree_jetz$tip.label[1:2],
```

```
  action = "accept",
  note = "Batch demo",
  stringsAsFactors = FALSE
)
batch <- batch[!is.na(batch$name_x), ]
if (nrow(batch) > 0) {
  result2 <- reconcile_override_batch(result, batch)
}
```

`reconcile_plot` *Plot the match composition of a reconciliation*

Description

Draw a one-glance bar or pie chart of how species names were resolved (exact, normalised, synonym, fuzzy, flagged, manual, unresolved). Uses base R graphics only, so no additional packages are required.

Usage

```
reconcile_plot(reconciliation, type = c("bar", "pie"), ...)
```

Arguments

`reconciliation`	A reconciliation object returned by `reconcile_tree()`, `reconcile_data()`, or a related matcher.
`type`	A length-1 character vector. Plot style:
	`"bar"` **(default)** Horizontal stacked bar chart. Best for slides, reports, and scripting.
	`"pie"` Pie chart. Useful when the match types are roughly balanced.
`...`	Additional arguments passed on to `graphics::barplot()` or `graphics::pie()` (e.g. `main`, `col`, `border`).

Value

The input `reconciliation`, invisibly, so you can use the function in a pipe.

See Also

`reconcile_summary()` for a textual breakdown; `reconcile_report()` for a full HTML audit trail.

Other reconciliation functions: `reconcile_apply()`, `reconcile_augment()`, `reconcile_crosswalk()`, `reconcile_data()`, `reconcile_diff()`, `reconcile_export()`, `reconcile_mapping()`, `reconcile_merge()`, `reconcile_multi()`, `reconcile_override()`, `reconcile_override_batch()`, `reconcile_report()`, `reconcile_review()`, `reconcile_splits_lumps()`, `reconcile_suggest()`, `reconcile_summary()`, `reconcile_to_trees()`, `reconcile_tree()`, `reconcile_trees()`

Examples

```
data(avonet_subset)
data(tree_jetz)
rec <- reconcile_tree(avonet_subset, tree_jetz,
                      x_species = "Species1", authority = NULL)
reconcile_plot(rec)
reconcile_plot(rec, type = "pie")
```

`reconcile_report` *Write a self-contained HTML reconciliation report*

Description

Produce an HTML file summarising a reconciliation object: provenance metadata, match-type breakdown, full mapping table, and a list of unresolved / flagged species. The file has no external dependencies (CSS is inlined), so it is suitable for sharing with collaborators, pasting into supplementary materials, or archiving next to analysis outputs.

Usage

```
reconcile_report(
  reconciliation,
  file,
  title = "Reconciliation Report",
  open = interactive()
)
```

Arguments

`reconciliation`	A reconciliation object returned by `reconcile_tree()`, `reconcile_data()`, or a related matcher.
`file`	A length-1 character vector. Output file path. Must end in `.html`.
`title`	A length-1 character vector. Report title shown at the top of the page. Default is generic.
`open`	Logical. Open the finished report in the default browser? Defaults to `TRUE` in interactive sessions, `FALSE` otherwise (so it does not block scripts).

Details

Value

The file path, invisibly.

Layout

The report opens with a run header (the originating `reconcile_tree()` / `reconcile_data()` call, timestamp, package version), the match-coverage summary, and a compact bar chart of match composition. Below those, per-match-type detail tables (normalised, synonym, fuzzy, flagged) and the unresolved-species list make each decision auditable. The `bird-workflow` vignette includes annotated screenshots of both sections.

See Also

`reconcile_summary()` for a console equivalent; `reconcile_export()` to additionally save aligned data and tree files.

Other reconciliation functions: `reconcile_apply()`, `reconcile_augment()`, `reconcile_crosswalk()`, `reconcile_data()`, `reconcile_diff()`, `reconcile_export()`, `reconcile_mapping()`, `reconcile_merge()`, `reconcile_multi()`, `reconcile_override()`, `reconcile_override_batch()`, `reconcile_plot()`, `reconcile_review()`, `reconcile_splits_lumps()`, `reconcile_suggest()`, `reconcile_summary()`, `reconcile_to_trees()`, `reconcile_tree()`, `reconcile_trees()`

Examples

```
data(avonet_subset)
data(tree_jetz)
rec <- reconcile_tree(avonet_subset, tree_jetz,
                      x_species = "Species1", authority = NULL)
f <- tempfile(fileext = ".html")
reconcile_report(rec, file = f, open = FALSE)
cat("Report written to:", f, "\n")
```

---

reconcile_review *Interactively review reconciliation matches*

Description

Presents matches one at a time for manual accept/reject decisions in an interactive R session. Each accepted or rejected match is applied via `reconcile_override()`, updating the reconciliation object in place. Useful for auditing fuzzy or flagged matches in the console or RStudio.

Usage

```
reconcile_review(
  reconciliation,
  type = c("flagged", "fuzzy", "all_unresolved"),
  suggest = TRUE,
  quiet = FALSE
)
```

Arguments

`reconciliation`	A reconciliation object returned by `reconcile_tree()`, `reconcile_data()`, or a related matcher.
`type`	A length-1 character vector. Which matches to review: `"flagged"` Only flagged matches (default). `"fuzzy"` Fuzzy and flagged matches. `"all_unresolved"` All unresolved species.
`suggest`	Logical. If `TRUE` and `type = "all_unresolved"`, show the closest fuzzy candidate (if any) alongside unresolved names. Default `TRUE`.
`quiet`	Logical. If `TRUE`, suppress the end-of-review summary. Default `FALSE`.

Details

This function requires an interactive session. In non-interactive contexts (e.g., scripts, CI), it warns and returns `reconciliation` unchanged.

At each prompt the user may enter:

- `a` Accept the proposed match (calls `reconcile_override()` with `action = "accept"`).
- `r` Reject the match (calls `reconcile_override()` with `action = "reject"`).
- `s` Skip – move to the next item without changes.
- `q` Quit – return the current state immediately.

Value

An updated reconciliation object reflecting accepted and rejected decisions.

See Also

`reconcile_override()` and `reconcile_override_batch()` for non-interactive corrections; `reconcile_suggest()` for shortlisting unresolved species before review.

Other reconciliation functions: `reconcile_apply()`, `reconcile_augment()`, `reconcile_crosswalk()`, `reconcile_data()`, `reconcile_diff()`, `reconcile_export()`, `reconcile_mapping()`, `reconcile_merge()`, `reconcile_multi()`, `reconcile_override()`, `reconcile_override_batch()`, `reconcile_plot()`, `reconcile_report()`, `reconcile_splits_lumps()`, `reconcile_suggest()`, `reconcile_summary()`, `reconcile_to_trees()`, `reconcile_tree()`, `reconcile_trees()`

Examples

```
if (interactive()) {
  # Interactive review in RStudio console:
  result <- reconcile_review(result, type = "flagged")
}
```

`reconcile_splits_lumps` *Flag taxonomic splits and lumps in a reconciliation*

Description

Taxonomic revisions often split a single species into several or lump several into one. When your data and your reference taxonomy disagree on such cases, the reconciliation mapping will show one name in one source linked to multiple accepted names in the other. `reconcile_splits_lumps()` scans a reconciliation for these cases and returns them as two tibbles, one for splits and one for lumps, so you can decide how to handle each before running your PCM (e.g. keep only one of the split taxa, pool traits across a lumped set, or exclude them entirely).

Usage

```
reconcile_splits_lumps(reconciliation, quiet = FALSE)
```

Arguments

`reconciliation`	A reconciliation object built with a non-`NULL` `authority` argument. The function inspects the `name_resolved` column, which is only populated when synonym resolution was performed.
`quiet`	Logical. Suppresses the console summary when `TRUE`. Default `FALSE`.

Details

Detection relies on the `name_resolved` column populated by synonym resolution — so `authority` must have been set (i.e. not `NULL`) when building the reconciliation.

Value

Invisibly, a list with two tibbles:

`splits` Cases where one name in source `x` corresponds to multiple accepted names in source `y`.

`lumps` Cases where several names in source `x` share a single accepted name in source `y`.

See Also

`reconcile_diff()` for comparing two reconciliations, which surfaces the same splits/lumps across taxonomy versions.

Other reconciliation functions: `reconcile_apply()`, `reconcile_augment()`, `reconcile_crosswalk()`, `reconcile_data()`, `reconcile_diff()`, `reconcile_export()`, `reconcile_mapping()`, `reconcile_merge()`, `reconcile_multi()`, `reconcile_override()`, `reconcile_override_batch()`, `reconcile_plot()`, `reconcile_report()`, `reconcile_review()`, `reconcile_suggest()`, `reconcile_summary()`, `reconcile_to_trees()`, `reconcile_tree()`, `reconcile_trees()`

Examples

```
# `reconcile_splits_lumps()` only surfaces rows that synonym lookup
# resolved (`match_type == "synonym"`), which requires `authority`
# to be non-NULL when building the reconciliation. The bundled-data
# call below uses `authority = NULL` for speed, so the output is
# empty:
data(avonet_subset)
data(tree_jetz)
rec <- reconcile_tree(avonet_subset, tree_jetz,
                      x_species = "Species1", authority = NULL,
                      quiet = TRUE)
sl <- reconcile_splits_lumps(rec, quiet = TRUE)
nrow(sl$splits); nrow(sl$lumps)   # 0 and 0

# To show what the output looks like when splits and lumps DO turn
# up, we hand-build a tiny reconciliation. In practice you would
# obtain this by calling reconcile_tree(..., authority = "col").
#
#   * Acanthiza pusilla (data) was split in CoL into A. pusilla and
#     A. apicalis  (1 x-name -> 2 y-names  ==>  split).
#   * Parus caeruleus and Cyanistes caeruleus (data: old + new names)
#     both map to Cyanistes caeruleus in CoL
#                 (2 x-names -> 1 y-name  ==>  lump).
demo_mapping <- tibble::tibble(
  name_x        = c("Acanthiza pusilla", "Acanthiza pusilla",
                    "Parus caeruleus",   "Cyanistes caeruleus"),
  name_y        = c("Acanthiza pusilla", "Acanthiza apicalis",
                    "Cyanistes caeruleus", "Cyanistes caeruleus"),
  name_resolved = c("Acanthiza pusilla", "Acanthiza pusilla",
                    "Cyanistes caeruleus", "Cyanistes caeruleus"),
  match_type    = "synonym",
  match_score   = 1,
  match_source  = "col",
  in_x          = TRUE,
  in_y          = TRUE,
  notes         = NA_character_
)
rec_demo <- structure(
  list(mapping   = demo_mapping,
       meta      = list(type = "data_tree", authority = "col"),
       counts    = list(),
       overrides = tibble::tibble()),
  class = "reconciliation"
)
sl <- reconcile_splits_lumps(rec_demo, quiet = TRUE)
sl$splits     # 1 row: Acanthiza pusilla split into 2 taxa
sl$lumps      # 1 row: Parus + Cyanistes lumped into 1 taxon
```

---

reconcile_suggest *Suggest near-miss matches for unresolved species*

Description

For every species that the four-stage cascade failed to resolve, `reconcile_suggest()` returns the top-`n` candidate matches in the reference source (`y`). The cascade is the exact -> normalised -> synonym -> fuzzy matching process run by `reconcile_tree()` and `reconcile_data()` (see `?prepR4pcm`). This is the most efficient way to audit orphan species: a typo or a species epithet that drifted by one letter will usually appear near the top of the list, and you can then feed the fix to `reconcile_override()` or `reconcile_override_batch()`.

Usage

```
reconcile_suggest(reconciliation, n = 3, threshold = 0.7, quiet = FALSE)
```

Arguments

`reconciliation`	A reconciliation object returned by `reconcile_tree()`, `reconcile_data()`, or a related matcher.
`n`	Integer. Maximum number of suggestions to return per unresolved species. Default `3`.
`threshold`	Numeric in [0, 1]. Minimum weighted similarity score for a candidate to be listed. Default `0.7` (quite permissive, because the idea is to surface candidates for review). Raise to `0.85` for a tighter shortlist.
`quiet`	Logical. Suppresses informational messages when `TRUE`. Default `FALSE`.

Details

Similarity is computed from the Levenshtein edit distance between normalised names — i.e., the minimum number of character insertions, deletions and substitutions needed to turn one name into the other, divided by the length of the longer name and subtracted from 1. The final score is weighted 60% genus, 40% specific epithet, which heavily penalises genus-level disagreement while tolerating small epithet differences.

For computational efficiency on large trees, `reconcile_suggest()` only compares a query name against reference names whose genus is within 2 character edits of the query genus. This can very occasionally miss a match where both the genus and the epithet are badly misspelled simultaneously; if you suspect that, lower the `threshold` and inspect manually.

Value

A tibble with one row per (unresolved, suggestion) pair:

`unresolved` The unresolved name from source `x`.

`suggestion` A candidate name from source `y`.

`score` Weighted similarity in [`threshold`, 1].

Rows are sorted by `unresolved` then descending `score`, so the first suggestion for each name is the best candidate.

See Also

reconcile_override() / reconcile_override_batch() to act on suggestions; reconcile_review() for an interactive alternative.

Other reconciliation functions: reconcile_apply(), reconcile_augment(), reconcile_crosswalk(), reconcile_data(), reconcile_diff(), reconcile_export(), reconcile_mapping(), reconcile_merge(), reconcile_multi(), reconcile_override(), reconcile_override_batch(), reconcile_plot(), reconcile_report(), reconcile_review(), reconcile_splits_lumps(), reconcile_summary(), reconcile_to_trees(), reconcile_tree(), reconcile_trees()

Examples

```
data(avonet_subset)
data(tree_jetz)
rec <- reconcile_tree(avonet_subset, tree_jetz,
                      x_species = "Species1", authority = NULL)

suggestions <- reconcile_suggest(rec, n = 2, threshold = 0.85)
head(suggestions, 10)
```

---

reconcile_summary *Print a reconciliation summary to the console*

Description

Produce a human-readable breakdown of a reconciliation object: how many names matched exactly, how many were rescued by normalisation, synonymy, or fuzzy matching, and which names remain unresolved. Usually the second function you call after reconcile_tree() or reconcile_data().

Usage

```
reconcile_summary(
  reconciliation,
  detail = c("full", "brief", "mismatches_only"),
  format = c("console", "data.frame"),
  file = NULL,
  ...
)
```

Arguments

reconciliation	A reconciliation object returned by reconcile_tree(), reconcile_data(), or a related matcher.
detail	A length-1 character vector. How much to show:
	"full" **(default)** Every match category, with the names belonging to each category listed out.

"brief" Counts only — a one-screen overview.

"mismatches_only" Non-exact matches and unresolved names. Useful once the easy cases are out of the way and you want to focus on what still needs review.

format — A length-1 character vector. Where the summary goes:

"console" **(default)** Pretty-printed to the screen.

"data.frame" Returns a list of tibbles silently; useful when writing a report or table in a larger script.

file — A length-1 character vector or NULL. If non-NULL, writes the console report to this file path in addition to printing it.

... — Additional arguments (currently unused).

Value

A reconciliation_summary object. The formatted report is attached to the object and rendered by print.reconciliation_summary(). R's REPL auto-printing means that calling the function at the prompt without assignment shows the full report; assigning the result to a variable shows nothing until you print(x) (or auto-print x). Use invisible(reconcile_summary(rec)) to suppress display at the prompt entirely.

See Also

reconcile_plot() for a visual summary; reconcile_report() for a shareable HTML audit trail; reconcile_mapping() for the full per-name tibble.

Other reconciliation functions: reconcile_apply(), reconcile_augment(), reconcile_crosswalk(), reconcile_data(), reconcile_diff(), reconcile_export(), reconcile_mapping(), reconcile_merge(), reconcile_multi(), reconcile_override(), reconcile_override_batch(), reconcile_plot(), reconcile_report(), reconcile_review(), reconcile_splits_lumps(), reconcile_suggest(), reconcile_to_trees(), reconcile_tree(), reconcile_trees()

Examples

```
data(avonet_subset)
data(tree_jetz)
rec <- reconcile_tree(avonet_subset, tree_jetz,
                      x_species = "Species1", authority = NULL)
reconcile_summary(rec, detail = "brief")
reconcile_summary(rec, detail = "mismatches_only")
```

---

reconcile_to_trees *Reconcile one dataset against multiple phylogenetic trees*

Description

Takes a single data frame and matches it against each tree in a named list, returning one reconciliation object per tree. This is the standard workflow for generating separate tree-compatible datasets aligned to different phylogenies (e.g., Clements 2023, 2024, 2025, Jetz 2012).

Usage

```
reconcile_to_trees(
  x,
  trees,
  x_species = NULL,
  authority = "col",
  rank = c("species", "subspecies"),
  overrides = NULL,
  db_version = NULL,
  fuzzy = FALSE,
  fuzzy_threshold = 0.9,
  resolve = c("flag", "first"),
  quiet = FALSE,
  x_label = NULL
)
```

Arguments

x	A data frame.
trees	A named list of ape::phylo objects or file paths.
x_species	A length-1 character vector. Column name in x containing species names. Auto-detected if NULL.
authority	A length-1 character vector, or NULL. Taxonomic authority used for synonym resolution (stage 3 of the cascade). One of:

"col" **(default)** Catalogue of Life — broad, curated, frequently updated. A sensible default for most taxa.

"itis" Integrated Taxonomic Information System — strong for North American vertebrates and plants.

"gbif" Global Biodiversity Information Facility backbone. Wider coverage; includes more recent synonymy.

"ncbi" NCBI Taxonomy — best when working with sequence data.

"ott" Open Tree of Life synthetic taxonomy. Useful when your downstream phylogeny is from the Open Tree synthesis.

"itis_test" A small bundled subset of ITIS, cached locally with **taxadb** for testing. Intended for examples and unit tests; not for analysis.

"gnverifier" HTTP-backed verification against ~100 sources via the Global Names verifier; no local database download. See vignette("getting-started") for the trade-off (wider coverage, requires network and the **httr2** package).

NULL Skip the synonym stage entirely. Useful for quick checks or when **taxadb** is unavailable. Stages 1, 2 and 4 still run.

Five authority codes that earlier versions of the package advertised — "iucn", "tpl", "fb", "slb", "wd" — are no longer accepted. Empirical testing against **taxadb** v22.12 showed that iucn errors with a schema mismatch and the others are not **taxadb** providers at all. Passing one of those values now produces a helpful migration error.

`rank` A length-1 character vector. Controls how trinomials are handled during normalisation:

- `"species"` **(default)** Strip infraspecific epithets so that `"Parus major major"` becomes `"Parus major"` before matching.
- `"subspecies"` Keep trinomials intact. Use this when your analysis operates at subspecies level.

`overrides` Optional pre-built corrections. Either a data frame with at least columns `name_x` and `name_y` (plus an optional `user_note` column), or a file path to a CSV with the same columns. Any name listed here bypasses the cascade and is recorded as `match_type = "manual"`. Useful for applying published crosswalks (see `reconcile_crosswalk()`) or for locking down decisions made in a previous run.

`db_version` A length-1 character vector. **taxadb** database snapshot to use (e.g. `"22.12"`). `NULL` (default) uses the latest available.

`fuzzy` Logical. Enables the fuzzy-matching stage when `TRUE`. Default `FALSE`. Turn this on to catch likely typos (*Corvus brachyrhnchos* -> *Corvus brachyrhynchos*). When `FALSE`, stages 1–3 still run.

`fuzzy_threshold`

Numeric in [0, 1]. Minimum genus-weighted similarity score for a fuzzy match to be accepted. Default `0.9` (roughly "no more than ~10% of characters differ"). Lower values (e.g. `0.7`) are more permissive but produce more false positives; always review fuzzy matches with `reconcile_suggest()` or `reconcile_review()` before trusting them.

`resolve` A length-1 character vector. What to do with borderline matches:

- `"flag"` **(default)** Mark low-confidence fuzzy matches (score below `flag_threshold`) and names with indirect taxadb synonymy as `match_type = "flagged"` so you can audit them with `reconcile_review()` or `reconcile_suggest()`.
- `"first"` Accept the highest-scoring candidate silently, without flagging. Faster but riskier; use only when you have already reviewed the ambiguities.

`quiet` Logical. Suppresses progress messages when `TRUE`. Default `FALSE`.

`x_label` A length-1 character vector or `NULL`. Human-readable label for source `x` stored in the reconciliation metadata and shown in `print()` / `format()`. Defaults to the expression passed as `x` (via `deparse(substitute())`). Set this explicitly when calling `reconcile_data()` inside another function so the label reflects the real data source rather than the local argument name.

Details

Species names in `x` are normalised once and reused across all trees, so synonym lookups are not repeated.

Value

A named list of reconciliation objects, one per tree, with the same names as `trees`.

See Also

reconcile_tree() for the single-tree case; reconcile_diff() to compare two reconciliations (e.g. to quantify how many species are gained or lost by switching taxonomies).

Other reconciliation functions: reconcile_apply(), reconcile_augment(), reconcile_crosswalk(), reconcile_data(), reconcile_diff(), reconcile_export(), reconcile_mapping(), reconcile_merge(), reconcile_multi(), reconcile_override(), reconcile_override_batch(), reconcile_plot(), reconcile_report(), reconcile_review(), reconcile_splits_lumps(), reconcile_suggest(), reconcile_summary(), reconcile_tree(), reconcile_trees()

Examples

```
data(avonet_subset)
data(tree_jetz)
data(tree_clements25)
results <- reconcile_to_trees(
  avonet_subset,
  trees = list(jetz = tree_jetz, clements = tree_clements25),
  x_species = "Species1",
  authority = NULL
)
# Compare overlap across trees
sapply(results, function(r) r$counts$n_exact)
```

---

reconcile_tree *Reconcile species names between a dataset and a phylogenetic tree*

Description

Match the species in a trait data frame (x) to the tip labels of a phylogenetic tree (tree), producing a reconciliation object ready to feed into reconcile_apply(), PGLS, phylogenetic GLMMs, ancestral state reconstruction, or any other phylogenetic comparative method (PCM). This is typically the first function you call in a prepR4pcm workflow.

Usage

```
reconcile_tree(
  x,
  tree,
  x_species = NULL,
  authority = "col",
  rank = c("species", "subspecies"),
  overrides = NULL,
  db_version = NULL,
  fuzzy = FALSE,
  fuzzy_threshold = 0.9,
  flag_threshold = 0.95,
```

```
  resolve = c("flag", "first"),
  quiet = FALSE,
  x_label = NULL
)
```

Arguments

x	A data frame containing the trait data. Must have one column of scientific names.
tree	An `ape::phylo` object, or a length-1 character vector giving the path to a Newick (`.nwk`, `.tre`, `.tree`) or Nexus (`.nex`, `.nexus`) file. File format is auto-detected.
x_species	A length-1 character vector. Name of the column in `x` containing scientific names (the same column referenced by `x` above; the term "species names" elsewhere in this help page is a synonym for the same scientific names). When `NULL`, the column is auto-detected from a small list of common labels (e.g. `species`, `Species1`, `scientific_name`); the list is not exhaustive — pass the column name explicitly if your data uses a non-standard label.
authority	A length-1 character vector, or `NULL`. Taxonomic authority used for synonym resolution (stage 3 of the cascade). One of: `"col"` **(default)** Catalogue of Life — broad, curated, frequently updated. A sensible default for most taxa. `"itis"` Integrated Taxonomic Information System — strong for North American vertebrates and plants. `"gbif"` Global Biodiversity Information Facility backbone. Wider coverage; includes more recent synonymy. `"ncbi"` NCBI Taxonomy — best when working with sequence data. `"ott"` Open Tree of Life synthetic taxonomy. Useful when your downstream phylogeny is from the Open Tree synthesis. `"itis_test"` A small bundled subset of ITIS, cached locally with **taxadb** for testing. Intended for examples and unit tests; not for analysis. `"gnverifier"` HTTP-backed verification against ~100 sources via the Global Names verifier; no local database download. See `vignette("getting-started")` for the trade-off (wider coverage, requires network and the **httr2** package). `NULL` Skip the synonym stage entirely. Useful for quick checks or when **taxadb** is unavailable. Stages 1, 2 and 4 still run. Five authority codes that earlier versions of the package advertised — `"iucn"`, `"tpl"`, `"fb"`, `"slb"`, `"wd"` — are no longer accepted. Empirical testing against **taxadb** v22.12 showed that `iucn` errors with a schema mismatch and the others are not **taxadb** providers at all. Passing one of those values now produces a helpful migration error.
rank	A length-1 character vector. Controls how trinomials are handled during normalisation: `"species"` **(default)** Strip infraspecific epithets so that `"Parus major major"` becomes `"Parus major"` before matching. `"subspecies"` Keep trinomials intact. Use this when your analysis operates at subspecies level.

`overrides` Optional pre-built corrections. Either a data frame with at least columns `name_x` and `name_y` (plus an optional `user_note` column), or a file path to a CSV with the same columns. Any name listed here bypasses the cascade and is recorded as `match_type = "manual"`. Useful for applying published crosswalks (see `reconcile_crosswalk()`) or for locking down decisions made in a previous run.

`db_version` A length-1 character vector. **taxadb** database snapshot to use (e.g. `"22.12"`). `NULL` (default) uses the latest available.

`fuzzy` Logical. Enables the fuzzy-matching stage when `TRUE`. Default `FALSE`. Turn this on to catch likely typos (*Corvus brachyrhnchos* -> *Corvus brachyrhynchos*). When `FALSE`, stages 1–3 still run.

`fuzzy_threshold`
Numeric in [0, 1]. Minimum genus-weighted similarity score for a fuzzy match to be accepted. Default `0.9` (roughly "no more than ~10% of characters differ"). Lower values (e.g. `0.7`) are more permissive but produce more false positives; always review fuzzy matches with `reconcile_suggest()` or `reconcile_review()` before trusting them.

`flag_threshold` Numeric in [0, 1]. When `resolve = "flag"`, fuzzy matches with a score below this value are recorded as `match_type = "flagged"` rather than `"fuzzy"`, marking them for manual review. Default `0.95`. Must be `>= fuzzy_threshold` to have any effect.

`resolve` A length-1 character vector. What to do with borderline matches:

`"flag"` **(default)** Mark low-confidence fuzzy matches (score below `flag_threshold`) and names with indirect taxadb synonymy as `match_type = "flagged"` so you can audit them with `reconcile_review()` or `reconcile_suggest()`.

`"first"` Accept the highest-scoring candidate silently, without flagging. Faster but riskier; use only when you have already reviewed the ambiguities.

`quiet` Logical. Suppresses progress messages when `TRUE`. Default `FALSE`.

`x_label` A length-1 character vector or `NULL`. Human-readable label for source `x` stored in the reconciliation metadata and shown in `print()` / `format()`. Defaults to the expression passed as `x` (via `deparse(substitute())`). Set this explicitly when calling `reconcile_data()` inside another function so the label reflects the real data source rather than the local argument name.

Details

Internally, `reconcile_tree()` treats the tree's tip labels as the `y` argument of `reconcile_data()` and runs the same four-stage matching cascade (exact -> normalized -> synonym -> fuzzy). Tip labels typically differ from data names only in formatting (underscores, capitalisation, authority strings), so even with `authority = NULL` you usually recover most matches at the *normalized* stage. Turn on `fuzzy = TRUE` to also catch spelling mistakes.

After reconciliation, the typical workflow is:

1. Inspect with `reconcile_summary()` or `reconcile_plot()`.
2. Investigate unresolved names with `reconcile_suggest()` and fix them with `reconcile_override()` or `reconcile_override_batch()`.

3. Produce an aligned data frame and pruned tree via `reconcile_apply()`.
4. Optionally, graft orphan species onto the tree with `reconcile_augment()` (exploratory only; always run sensitivity analyses).

Value

A reconciliation object with `meta$type == "data_tree"`. The `mapping` tibble has one row per unique name: matched species (`in_x & in_y`), data-only orphans (`in_x & !in_y`, candidates for `reconcile_augment()`), and tree-only orphans (`!in_x & in_y`, candidates for `reconcile_apply()` to prune).

References

Paradis, E. & Schliep, K. (2019) ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R. *Bioinformatics* 35:526–528. doi:10.1093/bioinformatics/bty633

See Also

`reconcile_apply()` to produce an aligned data-tree pair; `reconcile_augment()` to add orphan species back to the tree; `reconcile_to_trees()` to reconcile against several trees at once; `reconcile_data()` for the data-only counterpart.

Other reconciliation functions: `reconcile_apply()`, `reconcile_augment()`, `reconcile_crosswalk()`, `reconcile_data()`, `reconcile_diff()`, `reconcile_export()`, `reconcile_mapping()`, `reconcile_merge()`, `reconcile_multi()`, `reconcile_override()`, `reconcile_override_batch()`, `reconcile_plot()`, `reconcile_report()`, `reconcile_review()`, `reconcile_splits_lumps()`, `reconcile_suggest()`, `reconcile_summary()`, `reconcile_to_trees()`, `reconcile_trees()`

Examples

```
# Reconcile the bundled AVONET subset against the Jetz et al. (2012)
# bird tree. `authority = NULL` keeps the example offline; in a real
# analysis you would usually set `authority = "col"` (Catalogue of
# Life) to pick up taxonomic synonyms.
data(avonet_subset)
data(tree_jetz)

rec <- reconcile_tree(
  avonet_subset, tree_jetz,
  x_species = "Species1",
  authority = NULL,
  fuzzy     = TRUE          # also catch typos
)
rec                         # one-line status
reconcile_summary(rec)      # full breakdown by match type

# Produce aligned data + pruned tree ready for PGLS / PGLMM
aligned <- reconcile_apply(rec,
                           data = avonet_subset,
                           tree = tree_jetz,
                           species_col = "Species1",
                           drop_unresolved = TRUE)
```

```
nrow(aligned$data)
ape::Ntip(aligned$tree)
```

`reconcile_trees` *Reconcile tip labels between two phylogenetic trees*

Description

Compare the tip labels of two phylogenetic trees and report which species are shared, which differ only in formatting or synonymy, and which appear in only one of the two trees. Use this when assessing the impact of switching phylogenies (e.g., Jetz et al. 2012 vs Clements 2025) before deciding which tree to use in a downstream PCM.

Usage

```
reconcile_trees(
  tree1,
  tree2,
  authority = "col",
  rank = c("species", "subspecies"),
  overrides = NULL,
  db_version = NULL,
  fuzzy = FALSE,
  fuzzy_threshold = 0.9,
  resolve = c("flag", "first"),
  quiet = FALSE
)
```

Arguments

`tree1`	An `ape::phylo` object, or a character(1) path to a Newick/Nexus tree file.
`tree2`	An `ape::phylo` object, or a character(1) path to a Newick/Nexus tree file.
`authority`	A length-1 character vector, or `NULL`. Taxonomic authority used for synonym resolution (stage 3 of the cascade). One of: `"col"` **(default)** Catalogue of Life — broad, curated, frequently updated. A sensible default for most taxa. `"itis"` Integrated Taxonomic Information System — strong for North American vertebrates and plants. `"gbif"` Global Biodiversity Information Facility backbone. Wider coverage; includes more recent synonymy. `"ncbi"` NCBI Taxonomy — best when working with sequence data. `"ott"` Open Tree of Life synthetic taxonomy. Useful when your downstream phylogeny is from the Open Tree synthesis. `"itis_test"` A small bundled subset of ITIS, cached locally with **taxadb** for testing. Intended for examples and unit tests; not for analysis.

"gnverifier" HTTP-backed verification against ~100 sources via the Global Names verifier; no local database download. See vignette("getting-started") for the trade-off (wider coverage, requires network and the **httr2** package).

NULL Skip the synonym stage entirely. Useful for quick checks or when **taxadb** is unavailable. Stages 1, 2 and 4 still run.

Five authority codes that earlier versions of the package advertised — "iucn", "tpl", "fb", "slb", "wd" — are no longer accepted. Empirical testing against **taxadb** v22.12 showed that iucn errors with a schema mismatch and the others are not **taxadb** providers at all. Passing one of those values now produces a helpful migration error.

rank — A length-1 character vector. Controls how trinomials are handled during normalisation:

"species" **(default)** Strip infraspecific epithets so that "Parus major major" becomes "Parus major" before matching.

"subspecies" Keep trinomials intact. Use this when your analysis operates at subspecies level.

overrides — Optional pre-built corrections. Either a data frame with at least columns name_x and name_y (plus an optional user_note column), or a file path to a CSV with the same columns. Any name listed here bypasses the cascade and is recorded as match_type = "manual". Useful for applying published crosswalks (see reconcile_crosswalk()) or for locking down decisions made in a previous run.

db_version — A length-1 character vector. **taxadb** database snapshot to use (e.g. "22.12"). NULL (default) uses the latest available.

fuzzy — Logical. Enables the fuzzy-matching stage when TRUE. Default FALSE. Turn this on to catch likely typos (*Corvus brachyrhnchos* -> *Corvus brachyrhynchos*). When FALSE, stages 1–3 still run.

fuzzy_threshold

Numeric in [0, 1]. Minimum genus-weighted similarity score for a fuzzy match to be accepted. Default 0.9 (roughly "no more than ~10% of characters differ"). Lower values (e.g. 0.7) are more permissive but produce more false positives; always review fuzzy matches with reconcile_suggest() or reconcile_review() before trusting them.

resolve — A length-1 character vector. What to do with borderline matches:

"flag" **(default)** Mark low-confidence fuzzy matches (score below flag_threshold) and names with indirect taxadb synonymy as match_type = "flagged" so you can audit them with reconcile_review() or reconcile_suggest().

"first" Accept the highest-scoring candidate silently, without flagging. Faster but riskier; use only when you have already reviewed the ambiguities.

quiet — Logical. Suppresses progress messages when TRUE. Default FALSE.

Value

A reconciliation object with meta$type == "tree_tree".

See Also

`reconcile_diff()` to quantify gains/losses between two reconciliations; `reconcile_to_trees()` when you want to match a single dataset against many trees at once.

Other reconciliation functions: `reconcile_apply()`, `reconcile_augment()`, `reconcile_crosswalk()`, `reconcile_data()`, `reconcile_diff()`, `reconcile_export()`, `reconcile_mapping()`, `reconcile_merge()`, `reconcile_multi()`, `reconcile_override()`, `reconcile_override_batch()`, `reconcile_plot()`, `reconcile_report()`, `reconcile_review()`, `reconcile_splits_lumps()`, `reconcile_suggest()`, `reconcile_summary()`, `reconcile_to_trees()`, `reconcile_tree()`

Examples

```
data(tree_jetz)
data(tree_clements25)
rec <- reconcile_trees(tree_jetz, tree_clements25, authority = NULL)
rec
# How many tips are shared across both trees?
sum(reconcile_mapping(rec)$in_x & reconcile_mapping(rec)$in_y)
```

`tree_clements25` *Clements 2025 phylogenetic tree (subset)*

Description

A pruned version of the Clements 2025 taxonomy phylogenetic tree, containing ~850 species from the same families. Larger than tree_jetz because the Clements taxonomy recognises more species in these clades. Tip labels use underscores.

Usage

```
tree_clements25
```

Format

An object of class `phylo` (from the ape package).

Source

Clements et al. (2025) eBird/Clements Checklist of Birds of the World, v2025.

`tree_jetz` *Jetz (2012) phylogenetic tree (subset)*

Description

A pruned version of the BirdTree Stage 2 maximum clade credibility tree (Hackett backbone), containing ~660 species from the Corvoidea and allied passerine families. Deliberately smaller than avonet_subset (~920 species) so that reconciliation produces unresolved species suitable for `reconcile_augment()`. Tip labels use underscores.

Usage

```
tree_jetz
```

Format

An object of class `phylo` (from the ape package).

Source

Jetz et al. (2012) The global diversity of birds in space and time. *Nature* 491:444–448. doi:10.1038/nature11631

Index