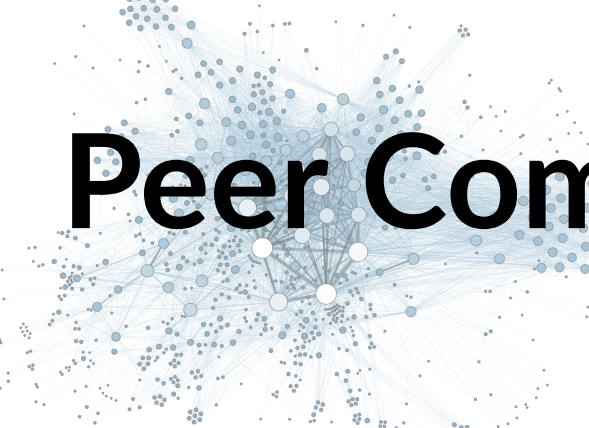




# Peer Community Journal

Section: Evolutionary Biology

Research article

Published  
2026-10-05

Cite as

Shinichi Nakagawa, Santiago Ortega, Ayumi Mizuno, Eduardo S. A. Santos, Malgorzata Lagisz, Bhavya Jain, Jimuel Jr Celeste and Sergio Poo Hernandez (2026) *prepR4pcm: An R package for preparing data and trees for phylogenetic comparative methods*, Peer Community Journal, 6: e104.

Correspondence

[snakagaw@ualberta.ca](mailto:snakagaw@ualberta.ca)

Peer-review

Peer reviewed and  
recommended by

PCI Evolutionary Biology,  
<https://doi.org/10.24072/pci.evolbiol.101496>



This article is licensed  
under the Creative Commons  
Attribution 4.0 License.

## prepR4pcm: An R package for preparing data and trees for phylogenetic comparative methods

Shinichi Nakagawa<sup>,1,2</sup>, Santiago Ortega<sup>,1</sup>, Ayumi Mizuno<sup>,1</sup>, Eduardo S. A. Santos<sup>,1</sup>, Malgorzata Lagisz<sup>,1,2</sup>, Bhavya Jain<sup>,1</sup>, Jimuel Jr Celeste<sup>,1</sup>, and Sergio Poo Hernandez<sup>,1</sup>

Volume 6 (2026), article e104

<https://doi.org/10.24072/pcjournal.809>

## Abstract

Phylogenetic comparative methods require species names in a trait dataset to match tip labels in a phylogenetic tree. Yet this apparently simple prerequisite is often one of the most fragile steps in a comparative workflow. Names may differ because of, for example, formatting, taxonomic revisions, synonyms, or spelling errors. If these differences are resolved informally, species can be lost from analyses, and the reasons for their loss can be difficult to reconstruct. Here, we present *prepR4pcm*, an R package for preparing data and trees for phylogenetic comparative methods. The package reconciles species names through a staged procedure: exact matching, normalised matching, synonym lookup with local taxonomic databases, and optional fuzzy matching for likely spelling errors. Each decision is stored in a reconciliation object with the original name, matched name, match type, confidence score, and a short explanation. This object turns name matching from a hidden pre-processing step into an auditable part of the analysis. *prepR4pcm* also supports the points where comparative workflows need human judgement. Users can inspect unresolved names, accept or reject suggested matches, add manual corrections, apply taxonomy crosswalks (which link names across taxonomic systems), compare reconciliation runs, and generate reports. The package then returns a matched data frame and pruned tree with the same species set, ready for phylogenetic generalised least squares, phylogenetic mixed models, phylogenetic meta-analysis, and related workflows. If users do not yet have a tree, *prepR4pcm* can retrieve trees from several sources, date trees when suitable information is available, and format tree-source citations. We illustrate the workflow using bundled datasets with realistic name mismatches. *prepR4pcm* is available at the CRAN (Comprehensive R Archive Network) with documentation and vignettes covering data and tree reconciliation, tree retrieval, multi-tree workflows, and phylogenetic meta-analysis.

<sup>1</sup>Collaboration for Open Science and Synthesis in Ecology and Evolution (COSSEE), Department of Biological Sciences, University of Alberta, Edmonton, AB, Canada, <sup>2</sup>Evolution & Ecology Research Centre, School of Biological, Earth and Environmental Sciences, University of New South Wales, Sydney, NSW, Australia

Peer Community Journal is a member of the  
Centre Mersenne for Open Scientific Publishing  
<http://www.centre-mersenne.org/>

e-ISSN 2804-3871



## Introduction

Phylogenetic comparative methods (PCMs) allow researchers to study trait variation while accounting for shared evolutionary history (Felsenstein, 1985; Grafen, 1989). These methods range from phylogenetic generalised least squares (PGLS; Freckleton et al., 2002), phylogenetic path analyses (Gonzalez-Voyer and von Hardenberg, 2014) to phylogenetic generalised linear mixed models or meta-analyses (e.g. Hadfield and Nakagawa, 2010; Mizuno et al., 2025; Nakagawa and Santos, 2012). Before fitting such models, researchers need two matched objects: a dataset with species names and a phylogenetic tree with tip labels representing the same species names. Yet this apparently simple prerequisite, matching species names, is often a stumbling block in a comparative workflow, because using scientific species names (binomial strings) as keys causes mismatches that can remove species from the analysis.

The name-matching problem often becomes visible only when the data and the tree are brought together. A species may be present in both sources, but fails to match because one source writes the name as “*Parus major*”, whereas another uses the tip label “*Parus\_major*”. In many cases, the two sources may use different names for the same or corresponding taxon, for example, because of an older genus name, a synonym, or a different taxonomic authority. Also, species may be absent from the tree. These cases could have different biological and practical implications, yet they may have similar effects if simple filtering is applied to remove mismatched species, thereby reducing the number of species in the analysis. For PCMs, this loss matters not only because the retained species set has less information and power, but also because non-random species loss could alter both the composition of the analysed dataset and its phylogenetic structure (Marcondes, 2019).

Several R (R Core Team, 2025) packages address parts of this problem. Taxonomic tools, such as `taxize` (Chamberlain and Szöcs, 2013), `taxadb` (Norman et al., 2020), `Taxonstand` (Cayuela et al., 2012), and `bdc` (Ribeiro et al., 2022), help users validate, standardise, or clean species names; the underlying Global Names Architecture provides language-agnostic name-parsing and verification services that several of these tools draw on. Its name parser is described by Mozzherin et al. (2017). Tree-retrieval tools, such as `rotl` (Michonneau et al., 2016), `rtrees` (Li, 2023), `clootl` (Miller et al., 2026), `fishtree` (Chang et al., 2019; Rabosky et al., 2018), and `datelife` (Sánchez Reyes et al., 2024), help users obtain phylogenies. PCM helper functions, such as `geiger::treedata()` (Harmon et al., 2008; Pennell et al., 2014) and `caper::comparative.data()` (Orme et al., 2025), help align data and trees for analysis. However, these tools are not designed to provide a single workflow that takes raw species names, from several data objects and produces aligned comparative objects together with an explicit, auditable record of name-matching decisions. The gap is therefore not the absence of useful tools, but the lack of a transparent conduit connecting between name reconciliation, tree preparation, and downstream PCM analysis (Table 1).

Here, we present `prepR4pcm`, an R package for preparing species-level data and phylogenetic trees for PCMs. The package connects three steps that are often handled separately: name reconciliation, tree preparation, and final data-tree alignment. First, it reconciles species names through a staged procedure that combines exact matching, name normalisation, synonym lookup, and optional fuzzy matching. Second, it records each decision in a reconciliation object, allowing users to inspect which names matched, changed, left unresolved, or corrected manually. Third, it applies the reconciliation to return a data frame and a pruned tree containing the same species set, ready for common comparative workflows, including PGLS, phylogenetic mixed models, and phylogenetic meta-analysis. When users do not yet have a suitable tree, `prepR4pcm` can also retrieve trees from several sources, record name substitutions, check the alignment between the tree and species-level data, and format tree-source citations to credit the original references.

**Table 1** – Existing R tools for taxonomic name handling and phylogenetic tree retrieval relevant to phylogenetic comparative methods. The first three columns identify each tool and place it in context. Tool = R package or function; Role = the main task the tool was designed to perform; Coverage = the main names, taxa, tree sources, databases, or input objects covered by the tool. The remaining columns summarise features relevant to preparing data and trees for PCMs. Tree = accepts or works directly with a user-supplied `ape::phylo tree`; Prov. = records provenance or an audit trail for name-matching, tree retrieval, or alignment decisions; Fuz. = supports fuzzy matching for likely spelling errors; Syn. = supports synonym lookup or accepted-name resolution; Map = supports taxonomy mapping tables or crosswalks between naming systems; Fetch = retrieves a tree from an external source; Multi = can return, handle, or compare more than one tree, such as posterior samples or alternative phylogenies. ✓ = full support; ~ = partial or context-dependent support; – = absent.

| Tool                                 | Role                             | Coverage                                                                       | Tree | Prov. | Fuz. | Syn. | Map | Fetch | Multi |
|--------------------------------------|----------------------------------|--------------------------------------------------------------------------------|------|-------|------|------|-----|-------|-------|
| <code>taxize</code>                  | Taxonomic name resolution        | Multiple online taxonomic databases                                            | –    | –     | ~    | ✓    | –   | –     | –     |
| <code>rgnparser</code> <sup>a</sup>  | Name parsing and verification    | Global Names services and linked name sources                                  | –    | ~     | ✓    | ✓    | ~   | –     | –     |
| <code>taxadb</code>                  | Local taxonomic name resolution  | Local copies of multiple taxonomic databases                                   | –    | –     | –    | ✓    | –   | –     | –     |
| <code>Taxonstand</code>              | Plant name standardisation       | Plant names, originally against The Plant List                                 | –    | ~     | ~    | ✓    | –   | –     | –     |
| <code>bdc</code>                     | Biodiversity data cleaning       | Occurrence records and associated taxonomic names                              | –    | ~     | ~    | ✓    | –   | –     | –     |
| <code>rotl</code>                    | Taxonomy and tree retrieval      | Open Tree of Life taxonomy and synthesis tree                                  | ~    | –     | –    | ✓    | –   | ✓     | –     |
| <code>Rtrees</code>                  | Megatree assembly                | Taxon-specific megatrees, including birds, mammals, fishes, plants, and others | ~    | ~     | –    | –    | –   | ✓     | ✓     |
| <code>clootl</code>                  | Avian tree retrieval             | Bird trees mapped to eBird/-Clements taxonomy                                  | ~    | –     | –    | –    | –   | ✓     | ~     |
| <code>fishtree</code>                | Fish tree retrieval              | Ray-finned fish phylogenies from the Fish Tree of Life                         | ~    | –     | –    | –    | –   | ✓     | ✓     |
| <code>datelife</code>                | Dated-tree synthesis             | Published dated phylogenies across taxa                                        | ~    | ~     | –    | ~    | –   | ✓     | ✓     |
| <code>geiger::treedata</code>        | Data–tree alignment              | User-supplied comparative dataset and tree                                     | ✓    | –     | –    | –    | –   | –     | –     |
| <code>caper::comparative.data</code> | PCM data-object creation         | User-supplied comparative dataset and tree                                     | ✓    | –     | –    | –    | –   | –     | –     |
| <b><code>prepR4pcm</code></b>        | Audited PCM preparation pipeline | User datasets and trees, with optional tree retrieval                          | ✓    | ✓     | ✓    | ✓    | ✓   | ✓     | ✓     |

<sup>a</sup>R wrapper for Global Names Architecture services (Mozzherin et al., 2017); `taxize` also uses the Global Names Verifier internally. This table compares broad workflow features rather than the quality, completeness, or taxonomic authority of each tool.

## Package description

### Overview and design

`prepR4pcm` was designed around three principles: conservative matching, transparent records, and reproducible outputs. These principles apply to both main parts of the package: reconciling species names among data objects and trees, and preparing trees for downstream PCMs.

First, `prepR4pcm` is conservative. It gives priority to high-confidence matches and does not replace them with weaker matches later in the workflow. Exact and normalised matches are attempted before synonym lookup and optional fuzzy matching. Names are not changed without being recorded, and uncertain matches are left for user review.

Second, the package is transparent. Each reconciliation returns a reconciliation object that records what happened to each name. This object stores the original and matched names, the match type, a score, a short explanation, the settings used in the run, and any manual overrides. Users can print the object for a summary or extract the full mapping table for inspection and reporting.

Third, the package is reproducible. The reconciliation object stores the settings used in the run, together with the package, database, and tree-source information where relevant. This design helps users repeat a workflow, compare alternative reconciliation runs, and report how the final data and tree objects were produced.

The package uses `ape` (Paradis and Schliep, 2019) for phylogenetic tree handling, `cli` for user messages, `rlang` for tidy evaluation, and `tibble` for data frames. Synonym matching uses `taxadb` (Norman et al., 2020) when this suggested dependency is available.

## Matching cascade

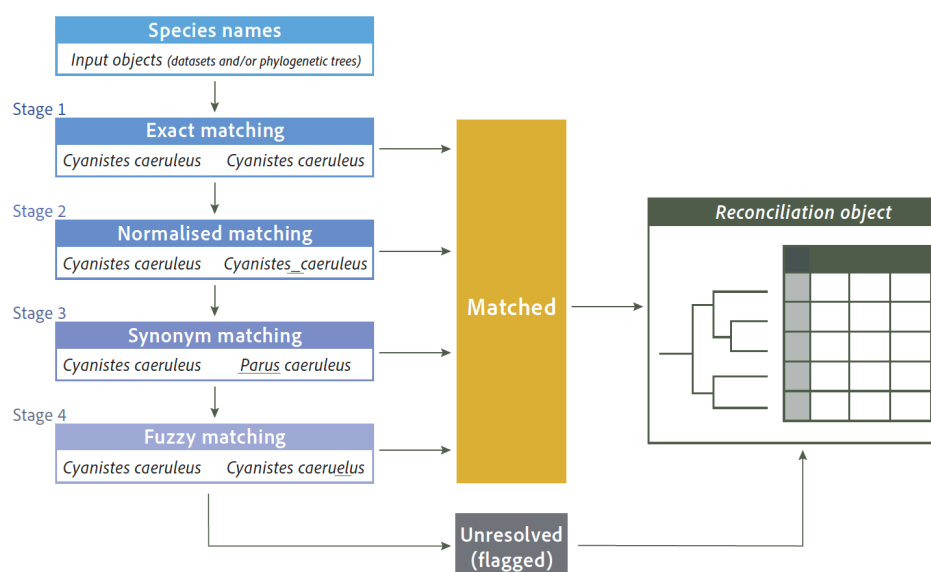
The core of `prepR4pcm` is a “four-stage matching cascade” (Figure 1). The cascade starts with the safest matches (exact name matching), then moves to those that require more interpretation. At each stage, only names that remain unmatched advance to the next. Thus, an exact match is never replaced by a weaker match later in the pipeline.

Stage 1: Exact matching.. The package first compares names exactly as written. These matches receive a score of 1.0 (ranging from 0 to 1).

Stage 2: Normalised matching.. The package then standardises names before matching them again. This step deals with common formatting differences, such as spaces versus underscores, inconsistent capitalisation, repeated spaces, authority strings, and Open Tree of Life suffixes. Intraspecific names are handled according to the `rank` argument. With `rank = "species"` (the default), infraspecific rank markers such as `subsp.`, `ssp.`, `var.`, and `f.`, together with the following epithet, are removed so that trinomials are reduced to binomials before matching. For example, *Parus major major* becomes *Parus major*. With `rank = "subspecies"`, trinomials are retained and recognised infraspecific rank markers are standardised; for example, `ssp.` is converted to `subsp.` Placeholder names such as *Genus* sp. are retained during normalisation rather than assigned to a named species. They can therefore match an equivalent placeholder in another source. Otherwise, they continue through any enabled later matching stages and remain available for user review if unresolved. These operations standardise name strings; they do not make taxonomic decisions about whether two names represent the same biological entity. Normalised matches receive a score of 1.0 because the underlying names are identical after these deterministic transformations, but they are labelled as “normalised” because the original strings differed.

Stage 3: Synonym matching.. Names that remain unmatched can be checked against local taxonomic databases through `taxadb`. This step is useful when the dataset uses one accepted name and the tree uses a synonym or an older name. Where possible, the package records which source was used. Synonym matches receive a score of 0.95 because they depend on the selected taxonomic authority. The `name_resolved` and `match_source` fields record the accepted name and taxonomic authority used to establish the match. When `resolve = "flag"`, indirect synonym matches that require both names to be resolved through the authority are flagged for user review.

Stage 4: Fuzzy matching.. Finally, the package can suggest near matches for likely spelling errors. Fuzzy matching compares the genus and species epithet using a component-based string distance. It is useful for finding typographical errors, but it can also suggest wrong matches. For this reason, fuzzy matching is optional, and uncertain matches are flagged for review rather than silently accepted. Fuzzy matches receive a component-based similarity score. By default, matches with scores of at least 0.90 are retained, with uncertain matches flagged for user review. The match type records that the result was obtained through fuzzy matching.



**Figure 1** – The four-stage matching cascade in `prepR4pcm`. Species names from input objects, such as datasets and phylogenetic trees, are passed through a conservative matching cascade. Exact matches are identified first, followed by matches after name normalisation, synonym-based matches, and optional fuzzy matches for likely spelling errors. The example species names illustrate the intended distinction among matching stages: identical spelling for exact matching (*Cyanistes caeruleus*), formatting differences for normalised matching (*Cyanistes\_caeruleus*), synonymy or older nomenclature for synonym matching (*Parus caeruleus*), and a likely typographical error for fuzzy matching (*Cyanistes caeruleus*). Names matched at any stage, as well as names retained as unresolved or flagged cases, are stored in the reconciliation object so that the process can be inspected, reported, and reproduced.

### The reconciliation object

The central output of `prepR4pcm` is a `reconciliation` object. This object records how names were matched, changed, left unresolved, or corrected by the user. It is produced not only when a dataset is reconciled with a tree, but also when users reconcile two datasets, two trees, several datasets against one tree, or one dataset against several trees. Thus, the same object structure is used across data–data, data–tree, tree–tree, multi-dataset, and multi-tree workflows.

A reconciliation object has four main components. The `$mapping` table records the names from the input objects and the outcome for each name. The exact columns depend on the workflow, but the table includes information such as original names, matched names, match status, match type, score, source, and notes. The `$meta` list stores the settings used in the run, together with timestamps and version information. The `$counts` table summarises how many names were matched, changed, or left unresolved. The `$overrides` table records manual changes made by the user.

The `reconciliation` object separates reconciliation from application. Users can first inspect the name-matching decisions, then decide whether to accept them, add corrections, or rerun the reconciliation with different settings. The object has `print()`, `summary()`, and `format()` methods. Printing gives a short summary of match coverage. The function `reconcile_mapping()` returns the full mapping table, which users can inspect, save, or include as supplementary material.

This mapping table is the main difference between a simple alignment and an auditable reconciliation. Standard PCM helper functions are useful for making data and trees ready for analysis, but they are not designed to record all name-matching decisions. `prepR4pcm` keeps this

record, so users can see which names matched, which names changed, which names remained unresolved, and which names were corrected by hand before the final data and tree objects are created.

### Curation, augmentation, and multi-input workflows

Because the reconciliation object keeps the matching decisions separate from the final aligned objects, users can curate the result before applying it. Automated matching should not be the final word in every case. Taxonomy changes, spelling errors, and species splits and lumps often require human judgement. Species splits and lumps are particularly challenging because they can create one-to-many or many-to-one relationships between taxonomies, which cannot generally be resolved by name matching alone. Users therefore need to decide how these taxa should be treated, and these decisions should be explicitly recorded as part of the reconciliation. `prepR4pcm` therefore provides tools for curation after the first reconciliation.

Users can correct individual matches with `reconcile_override()` or apply many corrections with `reconcile_override_batch()`. They can also use `reconcile_crosswalk()` to apply a taxonomic crosswalk: a table that links names in one taxonomy to names in another. For example, a crosswalk between BirdLife-based names and Jetz et al. / BirdTree tip labels can show which BirdLife names correspond to tip labels in Jetz et al. bird phylogeny (Jetz et al., 2012; Tobias et al., 2022). Some studies need to deal with species that are present in the data but absent from the tree. `reconcile_augment()` can add such species to a tree for exploratory analyses, for example, by placing an unresolved species with a close relative. Internally, `prepR4pcm` can also delegate grafting to the `V. PhyloMaker/U. PhyloMaker` suites when the required backbone and genus-family information are available (Jin and Qian, 2019, 2022, 2023). This step should be treated as a modelling assumption rather than an observed result. Users should therefore report which species were added and repeat the analysis with and without augmented tips.

Alternatively, users may substitute a closely related tree tip for which no data are available with a data species absent from the tree. In `prepR4pcm`, this can be recorded using `reconcile_override()` and applied using `reconcile_apply()`. This relabels the surrogate tip while preserving its phylogenetic position and branch lengths, leaving the numerical covariance values unchanged apart from their labels. Because this assumes that the focal species occupies the surrogate's phylogenetic position, the choice should be biologically justified and clearly reported.

The package also supports studies with more than one dataset or more than one tree. `reconcile_data()` reconciles two data frames, `reconcile_merge()` joins reconciled datasets, `reconcile_multi()` reconciles several datasets against one tree, and `reconcile_to_trees()` reconciles one dataset against several trees. These functions are useful when users want to compare alternative trees or combine trait datasets from different sources.

### Tree retrieval and dating

Many users begin with a trait dataset without a suitable phylogenetic tree. `pr_get_tree()` helps in this situation by retrieving a candidate phylogeny from public sources. At present, the package supports five backends: `rotl`, `rtrees`, `clootl`, `fishtree`, and `datelife`. These back-ends differ in taxonomic coverage, tree sources, and whether they return a single tree or a set of trees.

Before a backend is called, `pr_normalize_names()` standardises the input species names. This step helps avoid failures caused by simple formatting differences. When taxonomic name resolution changes an input name, the package records the replacement, linking the user's original name to the resolved name. The package also checks that the reported matched and unmatched names remain tied to the user's input list.

**Table 2** – The 31 available functions in `prepR4pcm` package, organised by functionalities. Functions prefixed `reconcile_` enable the matching and alignment pipeline; functions prefixed `pr_` enable tree retrieval, dating, comparison, and supporting utilities.

| Function                              | Family         | Description                                                                                                                    |
|---------------------------------------|----------------|--------------------------------------------------------------------------------------------------------------------------------|
| <code>reconcile_tree</code>           | Reconciliation | Reconcile a data frame against a phylogenetic tree                                                                             |
| <code>reconcile_data</code>           | Reconciliation | Reconcile two data frames                                                                                                      |
| <code>reconcile_trees</code>          | Reconciliation | Reconcile two phylogenetic trees                                                                                               |
| <code>reconcile_to_trees</code>       | Reconciliation | Reconcile one dataset against a list of trees                                                                                  |
| <code>reconcile_multi</code>          | Reconciliation | Reconcile multiple datasets against one tree                                                                                   |
| <code>reconcile_apply</code>          | Alignment      | Produce matched data frame and pruned tree                                                                                     |
| <code>reconcile_merge</code>          | Alignment      | Join two reconciled datasets                                                                                                   |
| <code>reconcile_override</code>       | Curation       | Manually correct an individual match                                                                                           |
| <code>reconcile_override_batch</code> | Curation       | Apply batch corrections from CSV or data frame                                                                                 |
| <code>reconcile_crosswalk</code>      | Curation       | Convert a taxonomy crosswalk to overrides                                                                                      |
| <code>reconcile_augment</code>        | Augmentation   | Graft missing species onto a phylogenetic tree for exploratory analyses                                                        |
| <code>reconcile_mapping</code>        | Inspection     | Extract the full mapping tibble                                                                                                |
| <code>reconcile_summary</code>        | Inspection     | Generate a detailed summary report                                                                                             |
| <code>reconcile_plot</code>           | Inspection     | Plot match composition (bar or pie chart)                                                                                      |
| <code>reconcile_suggest</code>        | Review         | Suggest fuzzy candidates for unresolved names                                                                                  |
| <code>reconcile_review</code>         | Review         | Interactively accept/reject flagged matches                                                                                    |
| <code>reconcile_diff</code>           | Review         | Compare two reconciliation runs                                                                                                |
| <code>reconcile_splits_lumps</code>   | Review         | Detect taxonomic splits and lumps                                                                                              |
| <code>reconcile_report</code>         | Reporting      | Produce self-contained HTML report                                                                                             |
| <code>reconcile_export</code>         | Reporting      | Write aligned data, tree, and mapping to disk                                                                                  |
| <code>pr_get_tree</code>              | Tree retrieval | Fetch a tree from external sources / backends:<br><code>rotl/rtrees/clootl/fishtree/datelife</code> (manual selection or auto) |
| <code>pr_get_tree_status</code>       | Tree retrieval | Probe which backends are installed and reachable                                                                               |
| <code>pr_date_tree</code>             | Dating         | Add DateLife calibration to an existing topology                                                                               |
| <code>pr_cite_tree</code>             | Citations      | Format citations for the sources of phylogenies used (select formats:<br><code>text/Markdown/BibTeX</code> )                   |
| <code>pr_tree_compare</code>          | Comparison     | Pairwise Jaccard, Robinson–Foulds, and bipartition-matched branch-length tree agreement                                        |
| <code>pr_phylo_cor</code>             | Correlation    | Build the phylogenetic correlation matrix from a tree for<br><code>metafor/MCMCglmm</code>                                     |
| <code>pr_tree_cache_dir</code>        | Cache          | Get or set the cache directory                                                                                                 |
| <code>pr_tree_cache_status</code>     | Cache          | List cache entries                                                                                                             |
| <code>pr_tree_cache_clear</code>      | Cache          | Wipe the cache                                                                                                                 |
| <code>pr_normalize_names</code>       | Utility        | Standardise scientific-name formatting                                                                                         |
| <code>pr_extract_tips</code>          | Utility        | Extract tip labels from a phylogenetic tree                                                                                    |

Multi-tree support depends on the backend. For example, some sources can return posterior samples of trees, whereas others return a single tree. The function `pr_date_tree()` can add time calibration to a topology using `datelife` where suitable information is available. The function `pr_cite_tree()` formats citations for tree sources in plain text, Markdown, or BibTeX.

**Reporting, comparison, and export**

Users can inspect a reconciliation with `reconcile_summary()` or write a self-contained HTML report with `reconcile_report()`. The function `reconcile_suggest()` proposes near matches for unresolved names, whereas `reconcile_diff()` compares two reconciliation runs. Finally, `reconcile_splits_lumps()` flags cases in which one taxonomy treats a name as one species and another treats it as several species, or vice versa.

Once users are satisfied with the reconciliation, `reconcile_apply()` returns the two objects required for most PCMs: a data frame and a pruned tree with the same set of species. `reconcile_export()` writes the aligned data, tree, and mapping table to disk. For tree comparisons, `pr_tree_compare()` summarises overlap in tip sets and differences in topology. For phylogenetic meta-analysis, `pr_phylo_cor()` builds a phylogenetic correlation matrix that

can be used in packages such as metafor (Viechtbauer, 2010). Together, these functions move users from raw species names to auditable, aligned comparative objects (Table 2).

## Worked examples

### Example 1: Reconciling a dataset with an existing tree

We first demonstrate the core workflow using a subset of AVONET bird morphological data (Tobias et al., 2022) and a subset of the Jetz et al. bird phylogeny (Jetz et al., 2012), both bundled with the package. This example is deliberately simple, but it illustrates a common problem in comparative workflows. AVONET uses spaces in species names, whereas the tree uses underscores. Exact matching therefore fails, even when the biological species are the same.

```
library(prepareR4pcm)
library(caper)

data(avonet_subset)
data(tree_jet2)

# Step 1: Reconcile species names
rec <- reconcile_tree(
  x = avonet_subset,
  tree = tree_jet2,
  x_species = "Species1",
  fuzzy = TRUE,
  resolve = "flag"
)

# Step 2: Inspect the matches
mapping <- reconcile_mapping(rec)
table(mapping$match_type)

# Step 3: Apply the reconciliation
aligned <- reconcile_apply(
  rec,
  data = avonet_subset,
  tree = tree_jet2,
  species_col = "Species1",
  drop_unresolved = TRUE
)

# Step 4: Run a comparative analysis
cd <- comparative.data(
  phy = aligned$tree,
  data = aligned$data,
  names.col = "Species1",
  vcv = TRUE
)

model <- pgl(
  log(Mass) ~ log(Wing.Length),
  data = cd
)
```

The output of `table(mapping$match_type)` shows that all retained species were matched through normalisation:

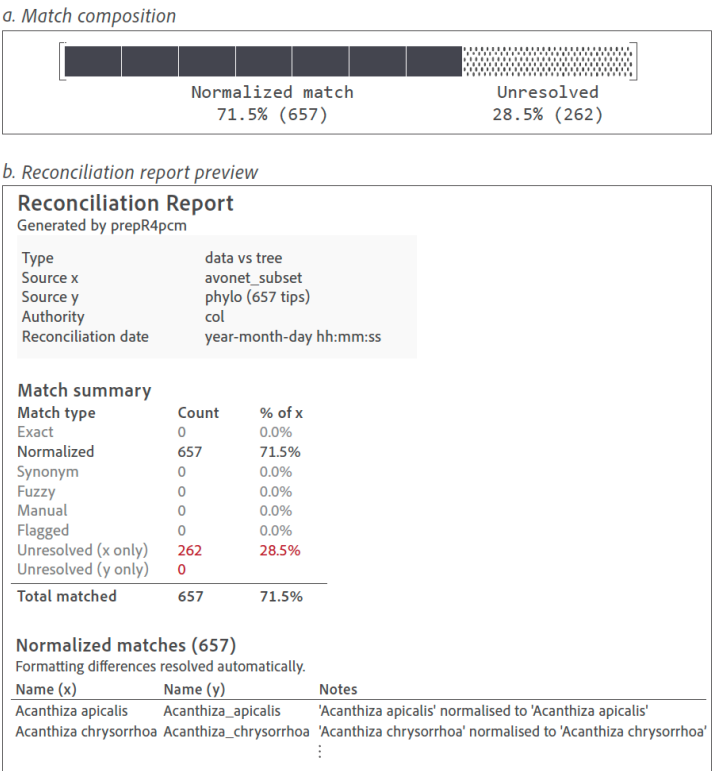
```
normalised unresolved_x
657 262
```

In this example, `reconcile_tree()` compares 919 data names with 657 tree tips. It matches 657 data names to tree tips, all through normalised matching, because the main difference is formatting. The remaining 262 species names from the AVONET data object are unresolved because they are present in the AVONET subset but not in the tree subset. Although fuzzy matching is enabled in the code, no retained species are matched by fuzzy matching in this example. After `reconcile_apply()` is run, both the data frame and the tree contain the same 657 species.

This example illustrates why an audit is useful. A simple final count would tell us that 657 species were retained. The reconciliation object tells us more: it shows that the retained species matched after a formatting change, that no exact matches occurred, and that 262 data rows were removed because they had no matching tree tip in this example.

The same audit trail becomes more important in less tidy examples. The reconciliation object can also record synonym matches, fuzzy suggestions, manual overrides, unresolved names, taxonomic sources, database versions, and user-supplied corrections. Users can inspect this information with `reconcile_mapping()`, summarise it with `reconcile_summary()`, or generate an HTML report with `reconcile_report()`.

Figure 2 shows two outputs from this workflow: the match composition and a schematic preview of the HTML report. Together, these outputs show both the numerical result of reconciliation and the record of decisions that led to the final data–tree pair.



**Figure 2** – Example outputs from a `prepR4pcm` reconciliation workflow. (a) Match composition for the worked example using an AVONET data subset containing 919 species names and a subset of the Jetz et al. (2012) phylogeny containing 657 tip labels. In this example, 657 data names were matched to the 657 tree tips through normalised matching, mainly because AVONET uses spaces in binomials, whereas the tree uses underscores. The remaining 262 species names in the AVONET data subset were unresolved data-only names, meaning that they were present in the data subset but absent from the tree subset. (b) Schematic preview of the HTML reconciliation report generated by `prepR4pcm`. The report summarises run metadata, source objects, taxonomic authority, match-type counts, unresolved names, and examples of names resolved through normalised matching.

## Example 2: Retrieving trees before reconciliation

The first example assumes that the user already has a phylogenetic tree. In many projects, however, users begin with a trait dataset but without a suitable tree. In this case, `prepR4pcm` can retrieve a candidate tree before the reconciliation step. The code below retrieves a posterior sample of fish trees and formats the relevant citations.

Tree-retrieval backends use optional packages that are not installed automatically with `prepR4pcm`. This example requires the `fishtree` package. Also, the optional `rotl` package is used for taxonomic name-resolution preflight when available; if it is absent, that preflight is skipped. Users can run `pr_get_tree_status()` to check which backends are installed and obtain installation instructions.

```
# Retrieve 50 trees for three fish species
fish_species <- c(
  "Salmo salar",
  "Esox lucius",
  "Oncorhynchus mykiss"
)

trees <- pr_get_tree(
  fish_species,
  source = "fishtree",
  n_tree = 50
)

# Check the posterior sample
class(trees$tree) # "multiPhylo"
length(trees$tree) # 50

# Format citations for the tree source
cat(pr_cite_tree(trees, format = "markdown"))
```

The retrieved tree object can then be used in the same way as a user-supplied tree. Users can pass it to `reconcile_tree()` and inspect the resulting reconciliation object. They can then use `reconcile_apply()` to create aligned data and tree objects.

Notably, the package code, documentation, and worked vignettes are available at <https://github.com/itchyshin/prepR4pcm>. The vignettes demonstrate data–tree reconciliation, tree retrieval, multi-tree workflows, tree comparison, bird and mammal trait workflows, and phylogenetic meta-analysis, which are also available at the `pkgdown` website (<https://itchyshin.github.io/prepR4pcm/>).

## Bundled data

Importantly, `prepR4pcm` includes small example datasets for vignettes, worked examples, and tests. These datasets are not intended to replace the full published datasets. Rather, they provide realistic examples in which species names do not always match perfectly.

The bird data examples include an AVONET data subset (Tobias et al., 2022), a NestTrait subset (Chia et al., 2023), a plumage lightness subset (Delhey et al., 2019), a Jetz et al. tree subset (Jetz et al., 2012), a subset of the complete avian phylogeny mapped to the eBird/Clements Checklist v2025 (Clements et al., 2025; McTavish et al., 2025), and a BirdLife to BirdTree taxonomy crosswalk distributed with AVONET (Jetz et al., 2012; Tobias et al., 2022). These objects support the worked example above and the bird-trait vignette.

The mammal examples include subsets from the Amniote life-history database (Myhrvold et al., 2015), PanTHERIA (Jones et al., 2009), and TetrapodTraits (Moura et al., 2024), along with a mammal tree used for demonstration (Upham et al., 2019). Users who rely upon these example datasets in their published work should cite the original data providers and, where needed, replace the example objects with the full original datasets.

The examples are useful because they are not perfectly clean. Some species names differ in formatting, some datasets follow different taxonomies, and some species are present in one source but not another. They therefore show the kinds of problems that users often face in real PCM workflows.

## Discussion

Preparing data for PCMs often begins with an important yet underreported step: matching species names between a dataset and a phylogenetic tree. Researchers usually report the final number of species analysed, but they may not report how many species were lost during matching, which names failed to match, or how ambiguous names were resolved. These missing details make it difficult to reproduce the analysis or to assess whether the final species set could affect the results.

`prepR4pcm` makes this step visible. The package reconciles names, records the decisions, and returns aligned objects for downstream analysis. The key outputs are not only a pruned tree and a matched data frame but also a reconciliation object that can be inspected, saved, shared, and reported. This object records exact matches, normalised matches, synonym matches, fuzzy suggestions, unresolved names, and manual corrections. It also allows users to report how many names were matched at each stage, which names remained unresolved, which decisions were corrected manually, and which taxonomic authority or database version was used. The full mapping table can be archived as supplementary material so that readers can trace how the final species set was obtained.

The same idea applies to tree retrieval. When users retrieve trees using `prepR4pcm`, the package records the tree source and the name substitutions made along the way. This record helps users write a clearer methods section and provides a more complete audit trail for the analysis. In this sense, `prepR4pcm` treats name matching, tree preparation, and final data alignment as parts of one workflow rather than as separate preprocessing steps.

Several limitations remain. Fuzzy matching can produce incorrect matches, so users should review fuzzy suggestions before accepting them. Tree augmentation also requires care. Here, tree augmentation means adding species that are present in the dataset but absent from the phylogenetic tree. Such additions may be based on genus-level placement, a close relative, a taxonomic crosswalk, or information from another more detailed published tree. These choices are assumptions, not observed phylogenetic relationships. Users should therefore report which species were added, how they were added, and what source or rule was used. They should also inspect the resulting tree visually, because plotting the augmented tree is often a useful sanity check. Where possible, analyses should be repeated with and without augmented species, and sometimes with alternative augmentation strategies.

Synonym matching also depends on the coverage and version of the taxonomic database used. Users should always report the authority, database version, and any manual changes made during reconciliation. Future versions will expose Global Names services as alternative backends, including `rgnparser` for the parsing step (Mozzherin et al., 2017) in `pr_normalize_names()` and the Global Names Verifier for synonym resolution against a larger set of sources, alongside the existing `taxadb`-based path.

By making reconciliation and related decisions explicit, `prepR4pcm` helps users move from raw species names to analysis-ready comparative objects. More importantly, it helps users show how they got there. This audit trail should make PCM workflows easier to verify, repeat, and report.

## Author contributions

### Author Statement

Conceptualization: SN, BJ, SPH. Data curation: SN, AM, SO. Formal analysis: SN, AM, SO. Funding Acquisition: SN, ML. Investigation: All. Methodology: SN. Project administration: SN, ESAS, SPH. Supervision: SN, ESAS, ML. Validation: All. Visualization: AM. Writing - Original Draft: SN. Writing - Review & Editing: All.

## Acknowledgements

We thank the developers of the upstream R packages that `prepR4pcm` wraps or builds on. Preprint version 3 of this article has been peer-reviewed and recommended by Peer Community in Evolutionary Biology (<https://doi.org/10.24072/pci.evolbiol.101496>; Kupczok, 2026).

## Statements

Conflict of interest: The authors declare no conflict of interest. Funding: SN was supported by the Canada Excellence Research Chair (CERC-2022-00074) and NSERC Discovery Grant (RGPIN-2025-04813). Peer review: The peer review history and recommendation are available at <https://doi.org/10.24072/pci.evolbiol.101496>. Data, scripts and code availability: The data, analysis scripts and package source code are archived together as `prepR4pcm` version 1.0.3 on Zenodo. Nakagawa et al. (2026) provides the archive record (<https://doi.org/10.5281/zenodo.22961118>). The development repository is available at <https://github.com/itchyshin/prepR4pcm>.

## Use of generative AI

Generative AI tools, including Claude Code (Anthropic) and Codex (OpenAI), were extensively used during software development to assist with code drafting, refactoring, documentation, and test development. These tools were not used as independent sources of scientific evidence. AI-assisted code was checked by the authors through unit tests, worked examples, vignettes, manual inspection, and comparisons with expected outputs. The authors take full responsibility for the accuracy of the package, analyses, and manuscript.

## References

- Cayuela L, Granzow-de la Cerda Í, Albuquerque FS, Golicher DJ (2012). Taxonstand: An R package for species names standardisation in vegetation databases. *Methods in Ecology and Evolution* **3**, 1078–1083. <https://doi.org/10.1111/j.2041-210X.2012.00232.x>.
- Chamberlain SA, Szöcs E (2013). taxize: taxonomic search and retrieval in R [version 2; peer review: 3 approved]. *F1000Research* **2**, 191. <https://doi.org/10.12688/f1000research.2-191.v2>.
- Chang J, Rabosky DL, Smith SA, Alfaro ME (2019). An R package and online resource for macroevolutionary studies using the ray-finned fish tree of life. *Methods in Ecology and Evolution* **10**, 1118–1124. <https://doi.org/10.1111/2041-210X.13182>.
- Chia SY, Fang YT, Su YT, Tsai PY, Hsieh C, Tsao SH, Juang JY, Hung CM, Tuanmu MN (2023). A global database of bird nest traits. *Scientific Data* **10**, 923. <https://doi.org/10.1038/s41597-023-02837-1>.
- Clements JF, Rasmussen PC, Schulenberg TS, Iliff MJ, Gerbracht JA, Lepage D, Spencer A, Billerman SM, Sullivan BL, Smith M, Wood CL (2025). The eBird/Clements checklist of Birds of the World: v2025. <https://www.birds.cornell.edu/clementschecklist/download/>. Accessed 26 September 2026.
- Delhey K, Dale J, Valcu M, Kempenaers B (2019). Reconciling ecogeographical rules: rainfall and temperature predict global colour variation in the largest bird radiation. *Ecology Letters* **22**, 726–736. <https://doi.org/10.1111/ele.13233>.

- Felsenstein J (1985). Phylogenies and the comparative method. *American Naturalist* **125**, 1–15. <https://doi.org/10.1086/284325>.
- Freckleton RP, Harvey PH, Pagel M (2002). Phylogenetic analysis and comparative data: a test and review of evidence. *American Naturalist* **160**, 712–726. <https://doi.org/10.1086/343873>.
- Gonzalez-Voyer A, von Hardenberg A (2014). An introduction to phylogenetic path analysis. In: *Modern phylogenetic comparative methods and their application in evolutionary biology: Concepts and practice*. Ed. by László Zsolt Garamszegi. Berlin, Heidelberg: Springer-Verlag, pp. 201–229. [https://doi.org/10.1007/978-3-662-43550-2\\_8](https://doi.org/10.1007/978-3-662-43550-2_8).
- Grafen A (1989). The phylogenetic regression. *Philosophical Transactions of the Royal Society of London. Series B* **326**, 119–157. <https://doi.org/10.1098/rstb.1989.0106>.
- Hadfield JD, Nakagawa S (2010). General quantitative genetic methods for comparative biology: phylogenies, taxonomies and multi-trait models for continuous and categorical characters. *Journal of Evolutionary Biology* **23**, 494–508. <https://doi.org/10.1111/j.1420-9101.2009.01915.x>.
- Harmon LJ, Weir JT, Brock CD, Glor RE, Challenger W (2008). GEIGER: investigating evolutionary radiations. *Bioinformatics* **24**, 129–131. <https://doi.org/10.1093/bioinformatics/btm538>.
- Jetz W, Thomas GH, Joy JB, Hartmann K, Mooers AO (2012). The global diversity of birds in space and time. *Nature* **491**, 444–448. <https://doi.org/10.1038/nature11631>.
- Jin Y, Qian H (2019). V.PhyloMaker: an R package that can generate very large phylogenies for vascular plants. *Ecography* **42**, 1353–1359. <https://doi.org/10.1111/ecog.04434>.
- 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**, 335–339. <https://doi.org/10.1016/j.pld.2022.05.005>.
- Jin Y, Qian H (2023). U.PhyloMaker: An R package that can generate large phylogenetic trees for plants and animals. *Plant Diversity* **45**, 347–352. <https://doi.org/10.1016/j.pld.2022.12.007>.
- Jones KE, Bielby J, Cardillo M, Fritz SA, O'Dell J, Orme CDL, Safi K, Sechrest W, Boakes EH, Carbone C, Connolly C, Cutts MJ, Foster JK, Grenyer R, Habib M, Plaster CA, Price SA, Rigby EA, Rist J, Teacher A, et al. (2009). PanTHERIA: a species-level database of life history, ecology, and geography of extant and recently extinct mammals. *Ecology* **90**, 2648. <https://doi.org/10.1890/08-1494.1>.
- Kupczok A (2026). Making phylogenetic comparative workflows more transparent and reproducible. *Peer Community in Evolutionary Biology*, 101496. <https://doi.org/10.24072/pci.evolbiol.101496>.
- Li D (2023). rtrees: an R package to assemble phylogenetic trees from megatrees. *Ecography* **2023**, e06643. <https://doi.org/10.1111/ecog.06643>. URL: <https://github.com/daijiang/rtrees>.
- Marcondes RS (2019). Realistic scenarios of missing taxa in phylogenetic comparative methods and their effects on model selection and parameter estimation. *PeerJ* **7**, e7917. <https://doi.org/10.7717/peerj.7917>.
- McTavish EJ, Gerbracht JA, Holder MT, Iliff MJ, Lepage D, Rasmussen PC, Redelings B, Sanchez-Reyes LL, Miller ET (2025). A complete and dynamic tree of birds. *Proceedings of the National Academy of Sciences* **122**, e2409658122. <https://doi.org/10.1073/pnas.2409658122>.
- Michonneau F, Brown JW, Winter DJ (2016). rotl: an R package to interact with the Open Tree of Life data. *Methods in Ecology and Evolution* **7**, 1476–1481. <https://doi.org/10.1111/2041-210X.12593>.
- Miller E, McTavish EJ, Sanchez Reyes LL (2026). clootl: Fetch and Explore the Cornell Lab of Ornithology Open Tree of Life Avian Phylogeny. R package version 0.1.4. CRAN. <https://doi.org/10.32614/CRAN.package.clootl>. URL: <https://CRAN.R-project.org/package=clootl>.
- Mizuno A, Drobniak SM, Williams C, Lagisz M, Nakagawa S (2025). Promoting the use of phylogenetic multinomial generalised mixed-effects model to understand the evolution of discrete traits. *Journal of Evolutionary Biology* **38**, 1699–1715. <https://doi.org/10.1093/jeb/voaf116>.

- Moura MR, Ceron K, Guedes JJM, Chen-Zhao R, Sica YV, Hart J, Dorman W, Portmann JM, González-del-Pliego P, Ranipeta A, Catenazzi A, Werneck FP, Toledo LF, Upham NS, Tonini JFR, Colston TJ, Guralnick R, Bowie RCK, Pyron RA, Jetz W (2024). A phylogeny-informed characterisation of global tetrapod traits addresses data gaps and biases. *PLOS Biology* **22**, e3002658. <https://doi.org/10.1371/journal.pbio.3002658>.
- Mozzherin DY, Myltsev AA, Patterson DJ (2017). “Gnparser”: a powerful parser for scientific names based on Parsing Expression Grammar. *BMC Bioinformatics* **18**, 279. <https://doi.org/10.1186/s12859-017-1663-3>.
- Myhrvold NP, Baldrige E, Chan B, Sivam D, Freeman DL, Ernest SKM (2015). An amniote life-history database to perform comparative analyses with birds, mammals, and reptiles. *Ecology* **96**, 3109. <https://doi.org/10.1890/15-0846R.1>.
- Nakagawa S, Ortega S, Mizuno A, Santos ES, Lagisz M, Jain B, Celeste JJ, Poo Hernandez S (2026). prepR4pcm 1.0.3: An R Package for Preparing Data and Trees for Phylogenetic Comparative Methods. Software version 1.0.3. Zenodo. <https://doi.org/10.5281/zenodo.22961118>. URL: <https://doi.org/10.5281/zenodo.22961118>.
- Nakagawa S, Santos ES (2012). Methodological issues and advances in biological meta-analysis. *Evolutionary Ecology* **26**, 1253–1274. <https://doi.org/10.1007/s10682-012-9555-5>.
- Norman KEA, Chamberlain S, Boettiger C (2020). taxadb: a high-performance local taxonomic database interface. *Methods in Ecology and Evolution* **11**, 1153–1159. <https://doi.org/10.1111/2041-210X.13440>.
- Orme D, Freckleton R, Thomas G, Petzoldt T, Fritz S, Isaac N, Pearse W (2025). caper: Comparative Analyses of Phylogenetics and Evolution in R. R package version 1.0.4. <https://doi.org/10.32614/CRAN.package.caper>.
- Paradis E, Schliep K (2019). ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R. *Bioinformatics* **35**, 526–528. <https://doi.org/10.1093/bioinformatics/bty633>.
- Pennell MW, Eastman JM, Slater GJ, Brown JW, Uyeda JC, FitzJohn RG, Alfaro ME, Harmon LJ (2014). geiger v2.0: an expanded suite of methods for fitting macroevolutionary models to phylogenetic trees. *Bioinformatics* **30**, 2216–2218. <https://doi.org/10.1093/bioinformatics/btu181>.
- R Core Team (2025). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. URL: <https://www.R-project.org/>.
- Rabosky DL, Chang J, Title PO, Cowman PF, Sallan L, Friedman M, Kaschner K, Garilao C, Near TJ, Coll M, Alfaro ME (2018). An inverse latitudinal gradient in speciation rate for marine fishes. *Nature* **559**, 392–395. <https://doi.org/10.1038/s41586-018-0273-1>.
- Ribeiro BR, Velazco SJE, Guidoni-Martins K, Tessarolo G, Jardim L, Bachman SP, Loyola R (2022). bdc: a toolkit for standardizing, integrating and cleaning biodiversity data. *Methods in Ecology and Evolution* **13**, 1421–1428. <https://doi.org/10.1111/2041-210X.13868>.
- Sánchez Reyes LL, McTavish EJ, O'Meara B (2024). DateLife: Leveraging databases and analytical tools to reveal the dated Tree of Life. *Systematic Biology* **73**, 470–485. <https://doi.org/10.1093/sysbio/syae015>.
- Tobias JA, Sheard C, Pigot AL, et al. (2022). AVONET: morphological, ecological and geographical data for all birds. *Ecology Letters* **25**, 581–597. <https://doi.org/10.1111/ele.13898>.
- Upham NS, Esselstyn JA, Jetz W (2019). Inferring the mammal tree: Species-level sets of phylogenies for questions in ecology, evolution, and conservation. *PLOS Biology* **17**, e3000494. <https://doi.org/10.1371/journal.pbio.3000494>.
- Viechtbauer W (2010). Conducting meta-analyses in R with the metafor package. *Journal of statistical software* **36**, 1–48. <https://doi.org/10.18637/jss.v036.i03>.