Package: taxonAlign (via r-universe)

November 15, 2026

Title Align and Update Taxonomic Names Against Any Taxonomic Reference

Version 0.1.0

Description Matches lists of raw taxon names, including misspelt, informal and morphospecies names, to a taxonomic reference, and updates them to their currently accepted names. The reference can be one or more released datasets (versioned, so results are reproducible) or the user's own tables, in a documented format, at any combination of taxonomic ranks. The companion package 'taxonAlign.dataprep' builds and releases those datasets, and helps users turn their own references into the format.

License MIT + file LICENSE

URL https://traitecoevo.github.io/taxonAlign/, https://github.com/traitecoevo/taxonAlign

BugReports https://github.com/traitecoevo/taxonAlign/issues

Encoding UTF-8

Language en-GB

Roxygen list(markdown = TRUE)

Depends R (>= 4.5.0)

Imports APCalign, cli, curl, dplyr, jsonlite, nanoparquet, rlang, stats, stringdist, tools, utils

Suggests arrow, knitr, readr, rmarkdown, stringr, testthat (>= 3.1.9), tibble, withr

VignetteBuilder knitr

Remotes traitecoevo/APCalign

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libssl-dev

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

Date/Publication 2026-11-15 09:26:49 UTC

RemoteUrl https://github.com/traitecoevo/taxonAlign

RemoteRef HEAD

RemoteSha 10aa2d7138b143aabb5101afe9b29d758a5982c2

Contents

align_taxa 2
check_taxonomic_reference 4
create_taxonomic_update_lookup 5
data_release_datasets 7
data_release_versions 8
download_data_release_files 9
load_taxonomic_resources 10
prepare_taxonomic_resources 11
taxonomic_reference_schema 13
update_taxa 14

Index **17**

`align_taxa` *Align raw taxon names to a prepared taxonomic reference*

Description

Attempts to match each name in `original_name` to a name in `resources` (produced by `prepare_taxonomic_resources()`), trying, in order, exact matches (scientific name with authorship, then canonical name), then progressively more lenient fuzzy matches, at species level first and then at each higher taxonomic rank present in `resources`. See `match_taxa()` for the full sequence of match attempts.

Usage

```
align_taxa(
  original_name,
  resources = NULL,
  identifier = NA_character_,
  fuzzy_abs_dist = 3,
  fuzzy_rel_dist = 0.2,
  fuzzy_matches = TRUE,
  imprecise_fuzzy_matches = FALSE,
  taxon_ranks_to_check = NULL,
  hybrids = TRUE,
  intergrades_affinis = TRUE,
  consider_english_name_endings = FALSE,
  include_bracketed_info = TRUE,
  progress = FALSE,
  full = FALSE,
  quiet = FALSE
)
```

Arguments

original_name	Character vector of raw taxon names to align.
resources	The taxonomic reference to align against: the nested list produced by prepare_taxonomic_resources() or a reference table (or a list of them, such as load_taxonomic_resources()'s output), which is run through prepare_taxonomic_resources() automatically. Preparing it yourself first saves repeating that work when aligning several name lists against the same reference.
identifier	A dataset, location or other identifier associated with each name in original_name – used only to disambiguate genus sp.-style aligned names (see prepare_taxonomic_resources()/mat as belonging to a specific dataset/location. Recycled if length 1; otherwise must be the same length as original_name. Defaults to NA.
fuzzy_abs_dist	The number of characters allowed to differ for a fuzzy match.
fuzzy_rel_dist	The proportion of characters allowed to differ for a fuzzy match.
fuzzy_matches	Logical; whether fuzzy matching is attempted at all. Defaults to TRUE.
imprecise_fuzzy_matches	Logical; reserved for a more lenient fuzzy-matching pass (not yet implemented – see match_taxa()). Defaults to FALSE.
taxon_ranks_to_check	Optional character vector of taxonomic ranks (besides species) to attempt higher-rank matches against. Defaults to NULL, which uses every higher rank present in resources.
hybrids	Logical; if TRUE (the default, as APCalign always does), a hybrid name (Genus x species) that isn't itself a listed name is resolved to genus rank ("Genus x [original name]") rather than left for later match attempts – with FALSE, "Banksia serrata x ericifolia" aligns to *Banksia serrata*. See ?match_taxa.
intergrades_affinis	Logical; if TRUE (the default, as APCalign always does), names suggesting an intergrade (--), an indecision between taxa (/), or a graded/"affinis"/"cf." identification are resolved to genus rank the same way, unless the whole name is a listed name. See ?match_taxa.
consider_english_name_endings	Logical; if TRUE, before any fuzzy matching, try substituting a recognised informal English vernacular name ending (e.g. "-id", "-ine", "-oid") for its formal Latin equivalent and attempt an exact match on the corrected name. Defaults to FALSE. See ?match_taxa.
include_bracketed_info	Logical; TRUE (the default) gives every higher-rank-only match APCalign's "<rank name> sp. [<original name>; <identifier>]" form, e.g. "Banksia sp. [Banksia]" for a bare "Banksia". FALSE drops the brackets when the name is nothing more than the matched rank's own name (a bare single word, or a bare "Genus (Subgenus)"), giving just "Banksia"; original_name keeps the raw input either way, and a name with anything more keeps the brackets regardless. See ?match_taxa.
progress	Logical; if TRUE, prints a text progress bar tracking what fraction of original_name has been resolved so far. Useful for large inputs, where matching can take a while. Defaults to FALSE. See ?match_taxa.

full	Logical; if TRUE, return every intermediate column match_taxa() computes (useful for inspecting *why* a name matched the way it did). If FALSE (the default), return just the key output columns.
quiet	Accepted for compatibility with APCalign's function of the same name; this function prints nothing either way.

Value

A tibble with one row per element of original_name (preserving its length, order and any duplicates/NAs), with columns original_name, cleaned_name, aligned_name, taxonomic_dataset, taxon_rank, taxonomic_status, taxon_ID, accepted_name_usage_ID, aligned_reason, alignment_code, chars_changed and identifier (or, if full = TRUE, every column match_taxa() computes along the way). taxon_ID/accepted_name_usage_ID identify which row of resources a name matched to, and are what update_taxa() uses to resolve a matched synonym forward to its current accepted name. chars_changed is the number of characters that differed between the name being matched and the reference name it matched – 0 for every exact match, the actual edit distance (Damerau-Levenshtein) for a fuzzy match, and NA when nothing matched at all.

Examples

```
reference <- data.frame(
  canonical_name = c("Boronia", "Boronia serrulata", "Boronia oldname"),
  scientific_name = c("Boronia Sm.", "Boronia serrulata Sm.", "Boronia oldname Sm."),
  taxon_rank = c("genus", "species", "species"),
  taxonomic_status = c("accepted", "accepted", "synonym"),
  taxonomic_dataset = "MY_LIST", genus = "Boronia",
  taxon_ID = c("1", "2", "3"), accepted_name_usage_ID = c("1", "2", "2")
)
# a misspelling and a synonym
aligned <- align_taxa(c("Boronia serulata", "Boronia oldname"), reference)
aligned[, c("original_name", "aligned_name", "taxonomic_status", "aligned_reason")]
```

---

check_taxonomic_reference

Check a table is in taxonAlign's taxonomic reference format

Description

Checks a reference table against taxonomic_reference_schema(): every required column is present, every record has a name, rank, status, dataset and ID, taxon_ID is unique, and every record leads somewhere in the table – an accepted record to itself, any other to another record's taxon_ID. Every data release load_taxonomic_resources() downloads passes this check.

Usage

```
check_taxonomic_reference(x, name = deparse(substitute(x)))
```

Arguments

x	A data frame.
name	What to call x in the error message. Defaults to the expression passed as x.

Details

prepare_taxonomic_resources() is more forgiving than this: it renames Darwin Core columns, and drops (with a warning) a record with no canonical_name. Use this check on a reference you're going to share or release, so that other people get the same results from it that you do.

Value

x, invisibly, if it passes. Otherwise an error listing every problem found.

Examples

```
reference <- data.frame(
  canonical_name = c("Boronia serrulata", "Boronia oldname"),
  scientific_name = c("Boronia serrulata Sm.", "Boronia oldname Sm."),
  taxon_rank = "species", taxonomic_status = c("accepted", "synonym"),
  taxonomic_dataset = "MY_LIST", genus = "Boronia",
  taxon_ID = c("1", "2"), accepted_name_usage_ID = c("1", "1")
)
check_taxonomic_reference(reference)
```

---

create_taxonomic_update_lookup

Align and update a list of taxon names in one call

Description

The one-call convenience pipeline: align_taxa() followed by update_taxa(). Matches each name in original_name against resources, then resolves that match forward to its current accepted name.

Usage

```
create_taxonomic_update_lookup(
  original_name,
  resources = NULL,
  identifier = NA_character_,
  fuzzy_abs_dist = 3,
  fuzzy_rel_dist = 0.2,
  fuzzy_matches = TRUE,
  imprecise_fuzzy_matches = FALSE,
  taxon_ranks_to_check = NULL,
```

```
  hybrids = TRUE,
  intergrades_affinis = TRUE,
  consider_english_name_endings = FALSE,
  include_bracketed_info = TRUE,
  progress = FALSE,
  taxonomic_splits = c("most_likely_species", "collapse_to_higher_taxon"),
  full = FALSE,
  quiet = FALSE
)
```

Arguments

original_name	Character vector of raw taxon names to align and update.
resources	The nested list of reference tibbles produced by prepare_taxonomic_resources(), or a reference table (or list of them) – see align_taxa().
identifier, fuzzy_abs_dist, fuzzy_rel_dist, fuzzy_matches, imprecise_fuzzy_matches, taxon_ranks_to_check, hybrids, intergrades_affinis, consider_english_name_endings, include_bracketed_info, progress	Forwarded to align_taxa(); see its documentation.
taxonomic_splits	Passed to update_taxa(): how to report a name that leads to more than one accepted name ("most_likely_species", the default, or "collapse_to_higher_taxon").
full	Logical; if TRUE, return every intermediate column align_taxa()/update_taxa() compute. If FALSE (the default), return just the key output columns: original_name, aligned_name, accepted_name, suggested_name, genus, family, taxon_rank, taxonomic_dataset, taxonomic_status, taxonomic_status_aligned, aligned_reason, update_reason and chars_changed (the number of characters that differed between the name being matched and the reference name it matched – 0 for an exact match, the edit distance for a fuzzy match, NA when nothing matched).
quiet	Logical; if FALSE (the default), ends by printing a summary, as APCalign's does: how many distinct names aligned to a species or infraspecific taxon (by dataset), only to a genus or a family, or to another rank; how many couldn't be aligned; how many were updated to a different name; and how many could refer to more than one accepted taxon. TRUE prints nothing.

Value

A tibble with one row per element of original_name; see update_taxa() for column details.

Examples

```
reference <- data.frame(
  canonical_name = c("Boronia", "Boronia serrulata", "Boronia oldname"),
  scientific_name = c("Boronia Sm.", "Boronia serrulata Sm.", "Boronia oldname Sm."),
  taxon_rank = c("genus", "species", "species"),
  taxonomic_status = c("accepted", "accepted", "synonym"),
  taxonomic_dataset = "MY_LIST", genus = "Boronia",
```

```
  taxon_ID = c("1", "2", "3"), accepted_name_usage_ID = c("1", "2", "2")
)
create_taxonomic_update_lookup(
  c("Boronia serulata", "Boronia oldname", "Boronia sp."),
  reference
)
```

`data_release_datasets` *List the datasets in a taxonAlign data release*

Description

Reads a data release's manifest: the datasets `load_taxonomic_resources()` can load from it, and where each came from. Cite the sources of the datasets you use.

Usage

```
data_release_datasets(
  version = NULL,
  path = NULL,
  refresh_cache = FALSE,
  cache_dir = tools::R_user_dir("taxonAlign", "cache"),
  quiet = FALSE
)
```

Arguments

`version`	Data release to load, a date (`"YYYY-MM-DD"`); see `data_release_versions()`. Defaults to the latest release, or, with no internet connection, the newest release already downloaded.
`path`	Optional directory holding your own copy of a release's files (`manifest.csv` and the `<dataset>.parquet` files). `version` is then ignored.
`refresh_cache`	Logical; if `TRUE`, re-downloads the files even if they're cached. Defaults to `FALSE`.
`cache_dir`	Directory downloaded release files are cached in. Defaults to `tools::R_user_dir("taxonAlign", "cache")`.
`quiet`	Logical; suppress progress messages. Defaults to `FALSE`.

Value

A tibble with one row per dataset: `dataset`, `asset` (the release file), `rows`, `source`, `retrieved`, `licence` and `citation`. The release is recorded in its `"data_release"` attribute.

Examples

```
# needs the internet
if (curl::has_internet()) {
  data_release_datasets(version = "2026-11-14", cache_dir = tempfile())
}
```

`data_release_versions` *List the available taxonAlign data releases*

Description

The reference data `load_taxonomic_resources()` loads is published as dated releases, so that an alignment can be reproduced exactly by naming the release it used. This lists the releases available online, most recent first; `data_release_datasets()` lists what's in one.

Usage

```
data_release_versions(refresh = FALSE)
```

Arguments

`refresh`	Logical; if `TRUE`, query GitHub again rather than reusing the list already fetched in this session. Defaults to `FALSE`.

Value

A character vector of release versions (dates, `"YYYY-MM-DD"`), most recent first; empty if GitHub can't be reached.

Examples

```
data_release_versions()
```

download_data_release_files *Download files from a taxonAlign data release*

Description

Downloads (or finds already cached) any files from a data release – the ready datasets `load_taxonomic_resources()` reads, or the verbatim source exports stored alongside them (e.g. `"nsl_fungi_taxon.parquet"`), which `taxonAlign.dataprep` rebuilds references from. Most users never need this directly. `data_release_versions()` lists the releases; each release's `manifest.csv` lists its files.

Usage

```
download_data_release_files(
  files,
  version = NULL,
  refresh_cache = FALSE,
  cache_dir = tools::R_user_dir("taxonAlign", "cache"),
  quiet = FALSE
)
```

Arguments

`files`	Character vector of file names in the release.
`version`	Data release to load, a date (`"YYYY-MM-DD"`); see `data_release_versions()`. Defaults to the latest release, or, with no internet connection, the newest release already downloaded.
`refresh_cache`	Logical; if `TRUE`, re-downloads the files even if they're cached. Defaults to `FALSE`.
`cache_dir`	Directory downloaded release files are cached in. Defaults to `tools::R_user_dir("taxonAlign", "cache")`.
`quiet`	Logical; suppress progress messages. Defaults to `FALSE`.

Value

A named character vector of local file paths, one per file. Downloaded files are cached under `<cache_dir>/data_release/<version>/`; if the latest release can't be downloaded, the newest release already cached that has every file is used instead (never when `version` is given).

Examples

```
# needs the internet: the release's manifest, into a temporary cache
if (curl::has_internet()) {
 download_data_release_files("manifest.csv", version = "2026-11-14", cache_dir = tempfile())
}
```

load_taxonomic_resources

Load released taxonomic reference data

Description

Downloads one or more taxonomic reference datasets from a dated taxonAlign data release, ready to pass to `create_taxonomic_update_lookup()`, `align_taxa()` or `prepare_taxonomic_resources()`. Naming the release (`version`) makes an alignment reproducible. Downloaded files are cached, so each is only downloaded once.

Usage

```
load_taxonomic_resources(
  datasets,
  version = NULL,
  include_extinct = FALSE,
  path = NULL,
  refresh_cache = FALSE,
  cache_dir = tools::R_user_dir("taxonAlign", "cache"),
  quiet = FALSE
)
```

Arguments

`datasets`	Character vector of datasets to load, e.g. `c("NSL_animals", "GBIF_AU")`.
`version`	Data release to load, a date (`"YYYY-MM-DD"`); see `data_release_versions()`. Defaults to the latest release, or, with no internet connection, the newest release already downloaded.
`include_extinct`	Logical; if `TRUE`, also keep taxa flagged as extinct, and their synonyms. Defaults to `FALSE`.
`path`	Optional directory holding your own copy of a release's files (`manifest.csv` and the `<dataset>.parquet` files). `version` is then ignored.
`refresh_cache`	Logical; if `TRUE`, re-downloads the files even if they're cached. Defaults to `FALSE`.
`cache_dir`	Directory downloaded release files are cached in. Defaults to `tools::R_user_dir("taxonAlign", "cache")`.
`quiet`	Logical; suppress progress messages. Defaults to `FALSE`.

Details

The datasets in the first release are the Australian National Species List exports – `"NSL_animals"` (the Australian Faunal Directory), `"NSL_algae"`, `"NSL_bryophytes"`, `"NSL_fungi"` and `"NSL_lichens"` – and `"GBIF_AU"`, every animal taxon in the GBIF Backbone with occurrence records in Australia,

plus its synonyms. GBIF has since replaced its Backbone with the Catalogue of Life Extended Release, so later releases have `"CatalogueOfLife_AU"` instead: that release's animal names GBIF has Australian records of, plus their synonyms. `data_release_datasets()` lists what a release holds, with each dataset's source, retrieval date, licence and citation.

When several datasets are loaded, they're returned in the order given, which is the order they're preferred in when a name appears in more than one.

Some datasets flag extinct taxa (an `extinct` column; `"GBIF_AU"`). (`"CatalogueOfLife_AU"` has no flag; it leaves out names GBIF knows only from fossil specimens instead.) GBIF's occurrence records include fossil specimens, so these are left out by default, as they only produce spurious matches against modern names. GBIF's flag doesn't distinguish fossils from recent extinctions, though – the thylacine (*Thylacinus cynocephalus*) is flagged extinct too – so `include_extinct = TRUE` keeps everything, with the `extinct` column showing which are which.

The releases are built by the `taxonAlign.dataprep` package, which also has the functions for building a reference from other sources (a live GBIF fetch, taxadb, the AFD's own export, APC).

Value

A named list of tibbles, one per dataset, each in taxonAlign's reference format (see `taxonomic_reference_schema()`). The release it came from is recorded in each tibble's `"data_release"` attribute.

Examples

```
# needs the internet: the lichen list from the National Species List (a few MB), into a
# temporary cache; a misspelling ("inermsi") is corrected
if (curl::has_internet()) {
 lichens <- load_taxonomic_resources("NSL_lichens", version = "2026-11-14", cache_dir = tempfile())
  create_taxonomic_update_lookup(c("Cladia aggregata", "Usnea inermsi"), lichens)
}
```

`prepare_taxonomic_resources` *Prepare a combined taxonomic reference table for name matching*

Description

Takes one or more taxonomic reference tables – loaded with `load_taxonomic_resources()`, built yourself, or any mix of these – and turns them into the single nested `resources` list that `align_taxa()`/`match_taxa()` expect: tables are combined if more than one was supplied, and the result is split by rank (and, for species-level rows, by taxonomic status). `align_taxa()`, `update_taxa()` and `create_taxonomic_update_lookup()` call this themselves when given a table (or a list of tables) rather than prepared `resources`; call it yourself to prepare once and reuse.

Usage

```
prepare_taxonomic_resources(
  taxonomic_resources = NULL,
  taxon_ranks_to_check = NULL
)
```

Arguments

`taxonomic_resources`
: A tibble, a path to a CSV file (read with readr, which must be installed), or a (optionally named) list of either – one element per taxonomic reference table to combine. Each table must be in taxonAlign's taxonomic reference format (see `taxonomic_reference_schema()`): at least the columns `canonical_name`, `scientific_name`, `taxon_rank`, `taxonomic_status`, `taxonomic_dataset`, `genus`, `taxon_ID` and `accepted_name_usage_ID`, matching APCalign's own naming convention (see Details). A row whose `canonical_name` is `NA` has no usable name and is dropped, with a warning. `accepted_name_usage_ID` must be self-referential (equal to `taxon_ID`) for already-accepted rows, not `NA` – this is what lets `update_taxa()` resolve a matched name forward to its current accepted name regardless of whether it matched an accepted name or a synonym. To map a table with other column names onto this format interactively, see `taxonAlign.dataprep::build_taxonomic_referenc`

`taxon_ranks_to_check`
: Optional character vector restricting which taxonomic ranks (besides species/infraspecific ranks) are retained in the returned `resources` for higher-rank matching (e.g. `c("genus", "family")`). Defaults to `NULL`, which keeps every rank present in `taxonomic_resources`.

Details

Unlike APCalign's `load_taxonomic_resources()`, which only ever assembles the fixed APC/APNI combination, this function works on whatever combined reference data the user supplies – at whatever set of taxonomic ranks it happens to contain, not just genus/species/family.

Column names are normalised on the way in, exactly the way `APCalign::load_taxonomic_resources()` normalises APC/APNI's raw Darwin Core column names to its own convention – e.g. a raw `canonicalName`/`taxonRank`/`taxonomicStatus`/`scientificName` column is renamed to `canonical_name`/`taxon_rank`/`taxo` This keeps `canonical_name` the *only* name field taxonAlign ever refers to.

When multiple tables are supplied, whichever is supplied first is preferred when a name exists in more than one, whatever taxonomic status each gives it. Within one table, a name listed more than once is matched to its most reliable status first (accepted, then the various kinds of synonym, then misapplications, ...).

Two matching conventions for subgenus names are supported side by side: some input name lists write the subgenus alone (e.g. `"Podosemum"`), others write the `Genus (Subgenus)` bracketed convention (e.g. `"Boronia (Podosemum)"`). To support both, when `taxonomic_resources` contains subgenus-rank rows, `resources$subgenus` holds the plain subgenus names (used for the first convention) and `resources$subgenus_v2` additionally holds a `genus_and_subgenus` column (used for the second).

Value

A named list of tibbles (and, for `species`, a further named list split by `taxonomic_status`): one element per taxonomic rank present in `taxonomic_resources` (after applying `taxon_ranks_to_check`, if supplied), plus `subgenus_v2` when subgenus-rank rows are present (see Details).

Examples

```
reference <- data.frame(
  canonical_name = c("Boronia", "Boronia serrulata", "Boronia oldname"),
  scientific_name = c("Boronia Sm.", "Boronia serrulata Sm.", "Boronia oldname Sm."),
  taxon_rank = c("genus", "species", "species"),
  taxonomic_status = c("accepted", "accepted", "synonym"),
  taxonomic_dataset = "MY_LIST", genus = "Boronia",
  taxon_ID = c("1", "2", "3"), accepted_name_usage_ID = c("1", "2", "2")
)
# prepare once, then reuse for several lookups
resources <- prepare_taxonomic_resources(reference)
names(resources)
align_taxa("Boronia serulata", resources)
```

---

taxonomic_reference_schema

The taxonomic reference format taxonAlign matches against

Description

Describes the columns a taxonomic reference table needs before it can be passed to `prepare_taxonomic_resources()`, `align_taxa()`, `update_taxa()` or `create_taxonomic_update_lookup()`. Every reference taxonAlign loads (`load_taxonomic_resources()`) is in this format, and any table you build yourself – by hand, with your own scripts, or with the `taxonAlign.dataprep` package – just needs to be in it too.

Usage

```
taxonomic_reference_schema()
```

Details

Eight columns are required: `canonical_name`, `scientific_name`, `taxon_rank`, `taxonomic_status`, `taxonomic_dataset`, `genus`, `taxon_ID` and `accepted_name_usage_ID`. The names follow AP-Calign's own; where a Darwin Core name exists (`canonicalName`, `taxonRank`, ...), `prepare_taxonomic_resources()` renames it automatically.

Any other column is kept. A character column named after a rank (`family`, `order`, `tribe`, ...) is read as that rank's value for each row: `prepare_taxonomic_resources()` adds a row for every family/order/... named this way that has no row of its own. Such an added row's `taxon_ID` is `<taxonomic_dataset>_<rank>_<name>`, e.g. `"WCVP_family_Annonaceae"` (the dataset of the

first row naming it) – built from the name alone, so it's the same every time the same reference is prepared. The optional columns listed by `taxonomic_reference_schema()` (`scientific_name_ID`, `scientific_name_authorship`, `taxon_distribution`, `extinct`, ...) are never read as ranks. `extinct` (logical) is described in `load_taxonomic_resources()`.

One record per row. A synonym (any `taxonomic_status` other than `"accepted"`) has its own row, whose `accepted_name_usage_ID` is the `taxon_ID` of the accepted record it leads to. If a table has several references in it, it's ordered from most to least preferred: when the same name appears twice with the same status, taxonAlign uses the first.

Value

A tibble with one row per column: `column`, `required` (logical), `darwin_core` (the Darwin Core name it's renamed from, or `NA`) and `description`.

See Also

`check_taxonomic_reference()` to check a table is in this format.

Examples

```
taxonomic_reference_schema()
```

---

update_taxa *Resolve aligned names forward to their current accepted name*

Description

Takes the output of `align_taxa()` and, for every row that matched something in `resources` (whether an accepted name or a synonym), looks up the *current* record for that match via `accepted_name_usage_ID` and reports its accepted name (when it resolves to one).

Usage

```
update_taxa(
  aligned_data,
  resources = NULL,
  taxonomic_splits = c("most_likely_species", "collapse_to_higher_taxon"),
  quiet = FALSE
)
```

Arguments

`aligned_data`	A tibble in the shape `align_taxa()` returns – in particular, it must have `aligned_name`, `taxonomic_status`, `taxon_rank`, `taxonomic_dataset`, `taxon_ID` and `accepted_name_usage_ID` columns (both `align_taxa()`'s default, `full = FALSE`, output and its `full = TRUE` output include these).

`resources` The nested list of reference tibbles produced by `prepare_taxonomic_resources()` (the same one passed to `align_taxa()`), or a reference table (or list of them) – see `align_taxa()`.

`taxonomic_splits` How to report a name that leads to more than one accepted name: `"most_likely_species"` (the default) suggests one and lists the others in brackets; `"collapse_to_higher_taxon"` collapses it to genus when all candidates share one. See Details.

`quiet` Accepted for compatibility with APCalign's function of the same name; this function prints nothing either way.

Details

Unlike APCalign's `update_taxonomy()` – five separate functions, each hand-written for one specific rank/dataset combination (`update_taxonomy_APC_genus`, `update_taxonomy_APC_family`, ...) – this is a **single, rank-agnostic** lookup: every rank present in `resources` is resolved the same way, by matching each row's `accepted_name_usage_ID` against a `taxon_ID` in a table combining every rank/status sublist in `resources`. A row that's already accepted resolves to itself (its own `accepted_name_usage_ID` is self-referential, per `prepare_taxonomic_resources()`'s requirements); a synonym resolves to whatever its `accepted_name_usage_ID` points to.

A synonym that leads to more than one accepted taxon (a species or infraspecific name listed as a synonym of two or more different accepted taxa – a split, or a pro parte synonym) follows APCalign's `taxonomic_splits` convention. One accepted name is chosen – by taxonomic status precedence, then (for a tie) an accepted name sharing the name's epithet, then one of the same rank, then one published no later than the name itself, then the oldest, then the order `resources` lists them in – and the others are listed with their status, both in `suggested_name` (`"X [alternative possible names: Y (synonym) | Z (misapplied)]"`) and in a separate `alternative_possible_names` column. With `taxonomic_splits = "collapse_to_higher_taxon"`, a name whose candidates all share one genus is collapsed to that genus instead (`"Genus sp. [collapsed names: ...]"`, `taxon_rank = "genus"`, `accepted_name = NA`). Only a match to a synonym record is treated this way: a match to an accepted name, or to a name given with its authorship (which identifies one record, e.g. one of two homonyms), is not.

One thing this deliberately does **not** do, relative to APCalign's `update_taxonomy()`:

- **No genus-substring surgery.** APCalign's `update_taxonomy_APC_genus()` reconstructs a species' suggested name by splicing just the updated genus portion into the aligned name, when only the genus (not the species) has changed. That's a nice refinement but doesn't obviously generalize across arbitrary ranks the way the rest of this does, so it's omitted here.

Value

`aligned_data` with `taxon_rank`/`genus`/`family`/`taxonomic_dataset` refreshed to whatever the current record has (a synonym's may be outdated), the pre-update `taxonomic_status` renamed to `taxonomic_status_aligned`, and new columns `accepted_name` (the current accepted name, when the match resolves to one, otherwise `NA`), `suggested_name` (`accepted_name` if available, otherwise falls back to `aligned_name`), `taxonomic_status` (the *current* record's status) and `update_reason` (a short explanation, mirroring `aligned_reason`'s style), and `alternative_possible_names` (the other candidates for a name that leads to more than one accepted name, otherwise `NA`).

It also has every column APCalign's `update_taxonomy()` returns, meaning the same thing, all describing the taxon `suggested_name` names (its accepted record, or for a split collapsed to genus, that genus's):

- `taxon_ID`: that accepted record's ID (`NA` when the name doesn't resolve to an accepted name). The ID of the record the name *matched* (a synonym's own ID, say), which `align_taxa()` returns as `taxon_ID`, is kept as `aligned_taxon_ID`.
- `taxon_ID_genus`: the ID of its genus's accepted record.
- `scientific_name` and `scientific_name_ID`: its name with authorship, and the reference's `scientific_name_ID` for it (e.g. an IPNI LSID for the WCVP; `NA` if the reference has none). For a name that matched but doesn't resolve to an accepted name (an unplaced name, say), those of the matched record.
- `subclass`, `taxon_distribution`: the reference's own columns of those names, `NA` if it has none.
- `canonical_name`: the same as `suggested_name`, as in APCalign.
- `row_number`: the row's position in `aligned_data`.
- `number_of_collapsed_taxa`: for a name leading to more than one accepted name, 1 when one is suggested, or the number of candidates when they're collapsed to genus; `NA` otherwise.

Examples

```
reference <- data.frame(
  canonical_name = c("Boronia", "Boronia serrulata", "Boronia oldname"),
  scientific_name = c("Boronia Sm.", "Boronia serrulata Sm.", "Boronia oldname Sm."),
  taxon_rank = c("genus", "species", "species"),
  taxonomic_status = c("accepted", "accepted", "synonym"),
  taxonomic_dataset = "MY_LIST", genus = "Boronia",
  taxon_ID = c("1", "2", "3"), accepted_name_usage_ID = c("1", "2", "2")
)
aligned <- align_taxa(c("Boronia serulata", "Boronia oldname"), reference)
# "Boronia oldname" is a synonym, updated to its accepted name
updated <- update_taxa(aligned, reference)
updated[, c("original_name", "aligned_name", "accepted_name", "update_reason")]
```

Index

align_taxa, 2
align_taxa(), *5, 6*, *10, 11*, *13–15*
APCalign::load_taxonomic_resources(), *12*

check_taxonomic_reference, 4
check_taxonomic_reference(), *14*
create_taxonomic_update_lookup, 5
create_taxonomic_update_lookup(), *10, 11*, *13*

data_release_datasets, 7
data_release_datasets(), *8*, *11*
data_release_versions, 8
data_release_versions(), *7*, *9, 10*
download_data_release_files, 9

load_taxonomic_resources, 10
load_taxonomic_resources(), *3, 4*, *7–9*, *11*, *13, 14*

prepare_taxonomic_resources, 11
prepare_taxonomic_resources(), *2, 3*, *5, 6*, *10*, *13*, *15*

taxonomic_reference_schema, 13
taxonomic_reference_schema(), *4*, *11, 12*

update_taxa, 14
update_taxa(), *5, 6*, *11*, *13*