Package 'dwctaxon'

December 13, 2023

Title Edit and Validate Darwin Core Taxon Data

Version 2.0.3

Description Edit and validate taxonomic data in compliance with Darwin Core standards (Darwin Core 'Taxon' class <https://dwc.tdwg.org/terms/#taxon>).

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports assertthat, digest, dplyr, glue, purrr, rlang, settings, stringr, tibble

Suggests testthat (>= 3.0.0), mockery, readr, usethis, knitr, rmarkdown, patrick, stringi, english, tidyr, utils, curl, httr

Depends R (>= 2.10)

Config/testthat/edition 3

URL https://docs.ropensci.org/dwctaxon/, https://github.com/ropensci/dwctaxon

BugReports https://github.com/ropensci/dwctaxon/issues

VignetteBuilder knitr

NeedsCompilation no

Author Joel H. Nitta [aut, cre, cph] (<https://orcid.org/0000-0003-4719-7472>), Wataru Iwasaki [ctb] (<https://orcid.org/0000-0002-9169-9245>), Collin Schwantes [rev] (Collin reviewed the package (v. 1.0.0.9000) for rOpenSci, see <https://github.com/ropensci/software-review/issues/574>), Stephen Formel [rev] (Stephen reviewed the package (v. 1.0.0.9000) for rOpenSci, see <https://github.com/ropensci/software-review/issues/574>)

Maintainer Joel H. Nitta <joelnitta@gmail.com>

Repository CRAN

Date/Publication 2023-12-13 17:20:02 UTC

R topics documented:

`dct_add_row` *Add row(s) to a taxonomic database*

Description

Add one or more rows to a taxonomic database in Darwin Core (DwC) format.

Usage

```
dct_add_row(
  tax_dat,
  taxonID = NULL,
  scientificName = NULL,
  taxonomicStatus = NULL,
  acceptedNameUsageID = NULL,
  acceptedNameUsage = NULL,
  new_dat = NULL,
  fill_taxon_id = dct_options()$fill_taxon_id,
  fill_usage_id = dct_options()$fill_usage_id,
  taxon_id_length = dct_options()$taxon_id_length,
  stamp_modified = dct_options()$stamp_modified,
  strict = dct_options()$strict,
  ...
)
```

Arguments

`tax_dat`	Dataframe; taxonomic database in DwC format.
`taxonID`	Character or numeric vector; values to add to taxonID column. Ignored if `new_dat` is not `NULL`.

`scientificName` Character vector; values to add to scientificName column. Ignored if `new_dat` is not `NULL`.

`taxonomicStatus` Character vector; values to add to taxonomicStatus column. Ignored if `new_dat` is not `NULL`.

`acceptedNameUsageID` Character or numeric vector; values to add to acceptedNameUsageID column. Ignored if `new_dat` is not `NULL`.

`acceptedNameUsage` Character vector; values to add to acceptedNameUsage column. Ignored if `new_dat` is not `NULL`.

`new_dat` A dataframe including columns corresponding to one or more of the above arguments, except for `tax_dat`. Other DwC terms can also be included as additional columns. All rows in `new_dat` will be appended to the input data (`tax_dat`).

`fill_taxon_id` Logical vector of length 1; if `taxon_id` is not provided, should values in the taxonID column be filled in by generating them automatically from the scientificName? If the `taxonID` column does not yet exist it will be created. Default `TRUE`.

`fill_usage_id` Logical vector of length 1; if `usage_id` is not provided, should values in the acceptedNameUsageID column be filled in by matching acceptedNameUsage to scientificName? If the `acceptedNameUsageID` column does not yet exist it will be created. Default `TRUE`.

`taxon_id_length` Numeric vector of length 1; how many characters should be included in automatically generated values of taxonID? Must be between 1 and 32, inclusive. Default `32`.

`stamp_modified` Logical vector of length 1; should the `modified` column of any newly created or modified row include a timestamp with the date and time of its creation/modification? If the `modified` column does not yet exist it will be created. Default `TRUE`.

`strict` Logical vector of length 1; should taxonomic checks be run on the updated taxonomic database? Default `FALSE`.

`...` Additional data to add, specified as sets of named character or numeric vectors; e.g., `parentNameUsageID = "6SH4"`. The name of each must be a valid column name for data in DwC format. Ignored if `new_dat` is not `NULL`.

Details

`fill_taxon_id` and `fill_usage_id` only act on the newly added data (they do not fill columns in `tax_dat`).

If "taxonID" is not provided for the new row and `fill_taxon_id` is `TRUE`, a value for taxonID will be automatically generated from the md5 hash digest of the scientific name.

To modify settings used for validation if `strict` is `TRUE`, use `dct_options()`.

Value

Dataframe; taxonomic database in DwC format.

Examples

```
tibble::tibble(
  taxonID = "123",
  scientificName = "Foogenus barspecies",
  acceptedNameUsageID = NA_character_,
  taxonomicStatus = "accepted"
) |>
  dct_add_row(
    scientificName = "Foogenus barspecies var. bla",
    parentNameUsageID = "123",
    nameAccordingTo = "me",
    strict = TRUE
  )
```

`dct_check_mapping` *Check mapping of usage taxonomic IDs*

Description

Check that values of terms like 'acceptedUsageID' map properly to taxonID in Darwin Core (DwC) taxonomic data.

Usage

```
dct_check_mapping(
  tax_dat,
  on_fail = dct_options()$on_fail,
  on_success = dct_options()$on_success,
  col_select = "acceptedNameUsageID",
  quiet = dct_options()$quiet
)
```

Arguments

`tax_dat`	Dataframe; taxonomic database in DwC format.
`on_fail`	Character vector of length 1, either "error" or "summary". Describes what to do if the check fails. Default `"error"`.
`on_success`	Character vector of length 1, either "logical" or "data". Describes what to do if the check passes. Default `"data"`.
`col_select`	Character vector of length 1; the name of the column (DwC term) to check. Default `"acceptedNameUsageID"`.
`quiet`	Logical vector of length 1; should warnings be silenced? Default `FALSE`.

Details

The following rules are enforced:

- Value of taxonID may not be identical to that of the selected column within a single row (in other words, a name cannot be its own accepted name, parent taxon, or basionym).
- Every value in the selected column must have a corresponding taxonID.

`col_select` can take one of the following values:

- `"acceptedNameUsageID"`: taxonID corresponding to the accepted name (of a synonym).
- `"parentNameUsageID"`: taxonID corresponding to the immediate parent taxon of a name (for example, for a species, this would be the genus).
- `"originalNameUsageID"`: taxonID corresponding to the basionym of a name.

Value

Depends on the result of the check and on values of `on_fail` and `on_success`:

- If the check passes and `on_success` is "logical", return `TRUE`
- If the check passes and `on_success` is "data", return the input dataframe
- If the check fails and `on_fail` is "error", return an error
- If the check fails and `on_fail` is "summary", issue a warning and return a dataframe with a summary of the reasons for failure

Examples

```
# The bad data has an acceptedNameUsageID (third row, "4") that lacks a
# corresponding taxonID
bad_dat <- tibble::tribble(
  ~taxonID, ~acceptedNameUsageID, ~taxonomicStatus, ~scientificName,
  "1", NA, "accepted", "Species foo",
  "2", "1", "synonym", "Species bar",
  "3", "4", "synonym", "Species bat"
)

dct_check_mapping(bad_dat, on_fail = "summary", quiet = TRUE)
```

`dct_check_sci_name` *Check scientificName*

Description

Check for correctly formatted scientificName column in Darwin Core taxonomic data.

Usage

```
dct_check_sci_name(
  tax_dat,
  on_fail = dct_options()$on_fail,
  on_success = dct_options()$on_success,
  quiet = dct_options()$quiet
)
```

Arguments

tax_dat	Dataframe; taxonomic database in DwC format.
on_fail	Character vector of length 1, either "error" or "summary". Describes what to do if the check fails. Default "error".
on_success	Character vector of length 1, either "logical" or "data". Describes what to do if the check passes. Default "data".
quiet	Logical vector of length 1; should warnings be silenced? Default FALSE.

Details

The following rules are enforced:

- scientificName may not be missing (NA)
- scientificName must be unique

Value

Depends on the result of the check and on values of on_fail and on_success:

- If the check passes and on_success is "logical", return TRUE
- If the check passes and on_success is "data", return the input dataframe
- If the check fails and on_fail is "error", return an error
- If the check fails and on_fail is "summary", issue a warning and return a dataframe with a summary of the reasons for failure

Examples

```
dct_check_sci_name(
  data.frame(scientificName = NA_character_),
  on_fail = "summary", quiet = TRUE
)

dct_check_sci_name(data.frame(scientificName = "a"))
```

`dct_check_taxon_id` *Check taxonID*

Description

Check for correctly formatted taxonID column in Darwin Core taxonomic data.

Usage

```
dct_check_taxon_id(
  tax_dat,
  on_fail = dct_options()$on_fail,
  on_success = dct_options()$on_success,
  quiet = dct_options()$quiet
)
```

Arguments

`tax_dat`	Dataframe; taxonomic database in DwC format.
`on_fail`	Character vector of length 1, either "error" or "summary". Describes what to do if the check fails. Default `"error"`.
`on_success`	Character vector of length 1, either "logical" or "data". Describes what to do if the check passes. Default `"data"`.
`quiet`	Logical vector of length 1; should warnings be silenced? Default `FALSE`.

Details

The following rules are enforced:

- taxonID may not be missing (NA)
- taxonID must be unique

Value

Depends on the result of the check and on values of `on_fail` and `on_success`:

- If the check passes and `on_success` is "logical", return `TRUE`
- If the check passes and `on_success` is "data", return the input dataframe
- If the check fails and `on_fail` is "error", return an error
- If the check fails and `on_fail` is "summary", issue a warning and return a dataframe with a summary of the reasons for failure

Examples

```
dct_check_taxon_id(
  data.frame(taxonID = NA_character_),
  on_fail = "summary", quiet = TRUE
)
dct_check_taxon_id(data.frame(taxonID = 1))
```

`dct_check_tax_status` *Check that taxonomicStatus is within valid values in Darwin Core taxonomic data*

Description

Check that taxonomicStatus is within valid values in Darwin Core taxonomic data

Usage

```
dct_check_tax_status(
  tax_dat,
  on_fail = dct_options()$on_fail,
  on_success = dct_options()$on_success,
  valid_tax_status = dct_options()$valid_tax_status,
  quiet = dct_options()$quiet
)
```

Arguments

`tax_dat`	Dataframe; taxonomic database in DwC format.
`on_fail`	Character vector of length 1, either "error" or "summary". Describes what to do if the check fails. Default `"error"`.
`on_success`	Character vector of length 1, either "logical" or "data". Describes what to do if the check passes. Default `"data"`.
`valid_tax_status`	Character vector of length 1; valid values for `taxonomicStatus`. Each value must be separated by a comma. Default `accepted, synonym, variant, NA`. `"NA"` indicates that missing (NA) values are valid. Case-sensitive. (see Examples).
`quiet`	Logical vector of length 1; should warnings be silenced? Default `FALSE`.

Value

Depends on the result of the check and on values of `on_fail` and `on_success`:

- If the check passes and `on_success` is "logical", return `TRUE`
- If the check passes and `on_success` is "data", return the input dataframe
- If the check fails and `on_fail` is "error", return an error
- If the check fails and `on_fail` is "summary", issue a warning and return a dataframe with a summary of the reasons for failure

References

https://dwc.tdwg.org/terms/#dwc:taxonomicStatus

Examples

```
# The bad data has an taxonomicStatus (third row, "foo") that is not
# a valid value
bad_dat <- tibble::tribble(
  ~taxonID, ~acceptedNameUsageID, ~taxonomicStatus, ~scientificName,
  "1", NA, "accepted", "Species foo",
  "2", "1", "synonym", "Species bar",
  "3", NA, "foo", "Species bat"
)

dct_check_tax_status(bad_dat, on_fail = "summary", quiet = TRUE)

# Example of setting valid values of taxonomicStatus via dct_options()

# First store existing settings, including any changes made by the user
old_settings <- dct_options()

# Change options for valid_tax_status
dct_options(valid_tax_status = "provisionally accepted, synonym, NA")
tibble::tribble(
  ~taxonID, ~acceptedNameUsageID, ~taxonomicStatus, ~scientificName,
  "1", NA, "provisionally accepted", "Species foo",
  "2", "1", "synonym", "Species bar",
  "3", NA, NA, "Strange name"
) |>
  dct_check_tax_status()

# Reset options to those before this example was run
do.call(dct_options, old_settings)
```

---

dct_drop_row *Drop row(s) of a taxonomic database*

Description

Drop one or more rows from a taxonomic database in Darwin Core (DwC) format by taxonID or scientificName.

Usage

```
dct_drop_row(tax_dat, taxonID = NULL, scientificName = NULL)
```

Arguments

tax_dat	Dataframe; taxonomic database in DwC format.
taxonID	Character or numeric vector; taxonID of the row(s) to be dropped.
scientificName	Character vector; scientificName of the row(s) to be dropped.

Details

Only works if values of taxonID or scientificName are unique and non-missing in the taxonomic database (tax_dat).

Either taxonID or scientificName should be provided, but not both.

Value

Dataframe; taxonomic database in DwC format

Examples

```
# Can drop rows by scientificName or taxonID
dct_filmies |>
  dct_drop_row(scientificName = "Cephalomanes atrovirens Presl")

dct_filmies |>
  dct_drop_row(taxonID = "54133783")

# Can drop multiple rows at once by providing multiple values for
# scientificName or taxonID
dct_filmies |>
  dct_drop_row(
    scientificName = c(
      "Cephalomanes atrovirens Presl",
      "Trichomanes crassum Copel."
    )
  )

dct_filmies |>
  dct_drop_row(
    taxonID = c(
      "54133783", "54133783"
    )
  )
```

---

dct_fill_col *Fill a column of a taxonomic database*

Description

Fill a column in a taxonomic database in Darwin Core (DwC) format.

Usage

```
dct_fill_col(
  tax_dat,
  fill_to = "acceptedNameUsage",
  fill_from = "scientificName",
  match_to = "taxonID",
  match_from = "acceptedNameUsageID",
  stamp_modified = dct_options()$stamp_modified
)
```

Arguments

`tax_dat`	Dataframe; taxonomic database in DwC format.
`fill_to`	Character vector of length 1; name of column to fill. If the column does not yet exist it will be created.
`fill_from`	Character vector of length 1; name of column to copy values from when filling.
`match_to`	Character vector of length 1; name of column to match to.
`match_from`	Character vector of length 1; name of column to match from.
`stamp_modified`	Logical vector of length 1; should the `modified` column of any newly created or modified row include a timestamp with the date and time of its creation/modification? If the `modified` column does not yet exist it will be created. Default `TRUE`.

Details

Several terms (columns) in DwC format come in pairs of "term" and "termID"; for example, "acceptedNameUsage" and "acceptedNameUsageID", where the first is the value in a human-readable form (in this case, scientific name of the accepted taxon) and the second is the value used by a machine (in this case, taxonID of the accepted taxon). Other pairs include "parentNameUsage" and "parentNameUsageID", "scientificName" and "scientificNameID", etc. None are required to be used in a given DwC dataset.

Often when updating data, the user may only fill in one value or the other (e.g., "acceptedNameUsage" or "acceptedNameUsageID"), but not both. The purpose of `dct_fill_col()` is to fill the missing column.

`match_from` and `match_to` are used to locate the values used for filling each cell. The values in the `match_to` column must be unique.

The default settings are to fill acceptedNameUsage with values from scientificName by matching acceptedNameUsageID to taxonID (see Example).

When adding timestamps with `stamp_modified`, any row that differs from the original data (`tax_dat`) is considered modified. This includes when a new column is added, in which case all rows will be considered modified.

Value

Dataframe; taxonomic database in DwC format.

Examples

```
# Fill acceptedNameUsage with values from scientificName by
# matching acceptedNameUsageID to taxonID
(head(dct_filmies, 5)) |>
  dct_fill_col(
    fill_to = "acceptedNameUsage",
    fill_from = "scientificName",
    match_to = "taxonID",
    match_from = "acceptedNameUsageID"
  )
```

---

dct_filmies *Taxonomic data of filmy ferns*

Description

Taxonomic data of filmy ferns (family Hymenophyllaceae) in Darwin Core format. Non-ASCII characters have been converted to ASCII, so some author names may not be as expected. Meant for demonstration purposes only, not formal data analysis.

Usage

```
dct_filmies
```

Format

Dataframe (tibble), with 2451 rows and 5 columns. For details about data format, see https://dwc.tdwg.org/terms/#taxon.

Details

Modified from data downloaded from the Catalog of Life under the Creative Commons Attribution (CC BY) 4.0 license.

Source

https://www.catalogueoflife.org/

Examples

```
dct_filmies
```

dct_modify_row *Modify row(s) of a taxonomic database*

Description

Modify one or more rows in a taxonomic database in Darwin Core (DwC) format.

Usage

```
dct_modify_row(
  tax_dat,
  taxonID = NULL,
  scientificName = NULL,
  taxonomicStatus = NULL,
  acceptedNameUsageID = NULL,
  acceptedNameUsage = NULL,
  clear_usage_id = dct_options()$clear_usage_id,
  clear_usage_name = dct_options()$clear_usage_name,
  fill_usage_name = dct_options()$fill_usage_name,
  remap_names = dct_options()$remap_names,
  remap_variant = dct_options()$remap_variant,
  stamp_modified = dct_options()$stamp_modified,
  strict = dct_options()$strict,
  quiet = dct_options()$quiet,
  args_tbl = NULL,
  ...
)
```

Arguments

tax_dat	Dataframe; taxonomic database in DwC format.
taxonID	Character or numeric vector of length 1; taxonID of the row to be modified (the selected row).
scientificName	Character vector of length 1; scientificName of the row to be modified if taxonID is NULL, OR the scientificName to assign to the selected row if taxonID is provided (see Details).
taxonomicStatus	Character vector of length 1; taxonomicStatus to assign to the selected row.
acceptedNameUsageID	Character or numeric vector of length 1; acceptedNameUsageID to assign to the selected row.
acceptedNameUsage	Character vector of length 1; acceptedNameUsage to assign to the selected row.
clear_usage_id	Logical vector of length 1; should acceptedNameUsageID of the selected row be set to NA if the word "accepted" is detected in tax_status (not case-sensitive)? Default TRUE.

`clear_usage_name`
: Logical vector of length 1; should acceptedNameUsageID of the selected row be set to `NA` if the word "accepted" is detected in tax_status (not case-sensitive)? Default `TRUE`.

`fill_usage_name`
: Logical vector of length 1; should the acceptedNameUsage of the selected row be set to the scientificName corresponding to its acceptedNameUsageID? Default `TRUE`.

`remap_names`
: Logical vector of length 1; should the acceptedNameUsageID be updated (remapped) for rows with the same acceptedNameUsageID as the taxonID of the row to be modified? Default `TRUE`.

`remap_variant`
: Same as `remap_names`, but applies specifically to rows with taxonomicStatus of "variant". Default `FALSE`.

`stamp_modified`
: Logical vector of length 1; should the `modified` column of any newly created or modified row include a timestamp with the date and time of its creation/modification? If the `modified` column does not yet exist it will be created. Default `TRUE`.

`strict`
: Logical vector of length 1; should taxonomic checks be run on the updated taxonomic database? Default `FALSE`..

`quiet`
: Logical vector of length 1; should warnings be silenced? Default `FALSE`..

`args_tbl`
: A dataframe including columns corresponding to one or more of the above arguments, except for `tax_dat`. In this case, the input taxonomic database will be modified sequentially over each row of input in `args_tbl`. Other DwC terms can also be included as additional columns, similar to using `...` to modify a single row.

`...`
: other DwC terms to modify, specified as sets of named values. Each element of the vector must have a name corresponding to a valid DwC term; see dct_terms.

Details

`taxonID` is only used to identify the row(s) to modify and is not itself modified. `scientificName` can be used in the same way if `taxonID` is not provided (as long as `scientificName` matches a single row). If both `taxonID` and `scientificName` are provided, `scientificName` will be assigned to the scientificName of the row identified by `taxonID`, replacing any value that already exists.

`acceptedNameUsageID` and `acceptedNameUsage` must match existing values of acceptedNameUsageID and acceptedNameUsage in the input data (`tax_dat`). On default settings, either can be used and the other will be filled in automatically (`fill_usage_id` and `fill_usage_name` are both `TRUE`).

Any other arguments provided that are DwC terms will be assigned to the selected row (i.e., they will modify the row).

If `remap_names` is `TRUE` (default) and `acceptedNameUsageID` is provided, any names that have an acceptedNameUsageID matching the taxonID of the selected row (i.e., synonyms of that row) will also have their acceptedNameUsageID replaced with the new acceptedNameUsageID. This behavior is not applied to names with taxonomicStatus of "variant" by default, but can be turned on for such names with `remap_variant`.

If `clear_usage_id` or `clear_usage_name` is `TRUE` and `taxonomicStatus` includes the word "accepted", acceptedNameUsageID or acceptedNameUsage will be set to NA respectively, regardless of the values of `acceptedNameUsageID`, `acceptedNameUsage`, or `fill_usage_name`.

Can either modify a single row in the input taxonomic database if each argument is supplied as a vector of length 1, or can apply a set of changes to the taxonomic database if the input is supplied as a dataframe via `args_tbl`.

Value

Dataframe; taxonomic database in DwC format

Examples

```
# Swap the accepted / synonym status of
# Cephalomanes crassum (Copel.) M. G. Price
# and Trichomanes crassum Copel.
dct_filmies |>
  dct_modify_row(
    scientificName = "Cephalomanes crassum (Copel.) M. G. Price",
    taxonomicStatus = "synonym",
    acceptedNameUsage = "Trichomanes crassum Copel."
  ) |>
  dct_modify_row(
    scientificName = "Trichomanes crassum Copel.",
    taxonomicStatus = "accepted"
  ) |>
  dct_validate(
    check_tax_status = FALSE,
    check_mapping_accepted_status = FALSE,
    check_sci_name = FALSE
  )
# Sometimes changing one name will affect others, if they map
# to the new synonym
dct_modify_row(
  tax_dat = dct_filmies |> head(),
  scientificName = "Cephalomanes crassum (Copel.) M. G. Price",
  taxonomicStatus = "synonym",
  acceptedNameUsage = "Cephalomanes densinervium (Copel.) Copel."
)
# Apply a set of changes
library(tibble)
updates <- tibble(
  scientificName = c(
    "Cephalomanes atrovirens Presl",
    "Cephalomanes crassum (Copel.) M. G. Price"
  ),
  taxonomicStatus = "synonym",
  acceptedNameUsage = "Trichomanes crassum Copel."
)
dct_filmies |>
  dct_modify_row(args_tbl = updates) |>
  dct_modify_row(
```

```
  scientificName = "Trichomanes crassum Copel.",
  taxonomicStatus = "accepted"
)
```

`dct_options` *Get and set function arguments via options*

Description

Changes the default values of function arguments.

Usage

```
dct_options(reset = FALSE, ...)
```

Arguments

`reset`	Logical vector of length 1; if TRUE, reset all options to their default values.
`...`	Any number of `argument = value` pairs, where the left side is the name of the argument and the right side is its value. See Details and Examples.

Details

Use this to change the default values of function arguments. That way, you don't have to type the same thing each time you call a function.

The arguments that can be set with this function are as follows:

Validation arguments:

- `check_col_names`: Logical vector of length 1; should all column names be required to be a valid DwC term? Default `TRUE`.
- `check_mapping_accepted_status`: Logical vector of length 1; should rules about mapping of variants and synonyms be enforced? Default `FALSE`. (See `dct_validate()`).
- `check_mapping_accepted`: Logical vector of length 1; should all values of `acceptedNameUsageID` be required to map to the `taxonID` of an existing name? Default `TRUE`.
- `check_mapping_original`: Logical vector of length 1; should all values of `originalNameUsageID` be required to map to the `taxonID` of an existing name? Default `TRUE`.
- `check_mapping_parent`: Logical vector of length 1; should all values of `parentNameUsageID` be required to map to the `taxonID` of an existing name? Default `TRUE`.
- `check_sci_name`: Logical vector of length 1; should all instances of `scientificName` be required to be non-missing and unique? Default `TRUE`.
- `check_status_diff`: Logical vector of length 1; should each scientific name be allowed to have only one taxonomic status? Default `FALSE`.
- `check_tax_status`: Logical vector of length 1; should all taxonomic names be required to have a valid value for taxonomic status (by default, "accepted", "synonym", or "variant")? Default `TRUE`.

- check_taxon_id: Logical vector of length 1; should all instances of taxonID be required to be non-missing and unique? Default TRUE.
- extra_cols: Character vector; names of columns that should be allowed beyond those defined by the DwC taxon standard. Default NULL. Providing column name(s) that are valid DwC taxon column(s) has no effect.
- on_fail: Character vector of length 1, either "error" or "summary". Describes what to do if the check fails. Default "error".
- on_success: Character vector of length 1, either "logical" or "data". Describes what to do if the check passes. Default "data".
- skip_missing_cols: Logical vector of length 1; should checks be silently skipped if any of the columns they inspect are missing? Default FALSE.
- valid_tax_status: Character vector of length 1; valid values for taxonomicStatus. Each value must be separated by a comma. Default accepted, synonym, variant, NA. "NA" indicates that missing (NA) values are valid. Case-sensitive.

Editing arguments:

- clear_usage_id: Logical vector of length 1; should acceptedNameUsageID of the selected row be set to NA if the word "accepted" is detected in tax_status (not case-sensitive)? Default TRUE.
- clear_usage_name: Logical vector of length 1; should acceptedNameUsage of the selected row be set to NA if the word "accepted" is detected in tax_status (not case-sensitive)? Default TRUE.
- fill_taxon_id: Logical vector of length 1; if taxon_id is not provided, should values in the taxonID column be filled in by generating them automatically from the scientificName? If the taxonID column does not yet exist it will be created. Default TRUE.
- fill_usage_id: Logical vector of length 1; if usage_id is not provided, should values in the acceptedNameUsageID column be filled in by matching acceptedNameUsage to scientificName? If the acceptedNameUsageID column does not yet exist it will be created. Default TRUE.
- fill_usage_name: Logical vector of length 1; should the acceptedNameUsage of the selected row be set to the scientificName corresponding to its acceptedNameUsageID? Default TRUE.
- remap_names: Logical vector of length 1; should the acceptedNameUsageID be updated (remapped) for rows with the same acceptedNameUsageID as the taxonID of the row to be modified? Default TRUE.
- remap_variant: Same as remap_names, but applies specifically to rows with taxonomicStatus of "variant". Default FALSE.
- stamp_modified: Logical vector of length 1; should the modified column of any newly created or modified row include a timestamp with the date and time of its creation/modification? If the modified column does not yet exist it will be created. Default TRUE.
- taxon_id_length: Numeric vector of length 1; how many characters should be included in automatically generated values of taxonID? Must be between 1 and 32, inclusive. Default 32.

General arguments:

- quiet: Logical vector of length 1; should warnings be silenced? Default FALSE.
- strict: Logical vector of length 1; should taxonomic checks be run on the updated taxonomic database? Default FALSE.

Value

Nothing; used for its side-effect.

Examples

```
# Show all options
dct_options()

# Store existing settings, including any changes made by the user
old_settings <- dct_options()

# View one option
dct_options()$valid_tax_status

# Change one option
dct_options(valid_tax_status = "accepted, weird, whatever")
dct_options()$valid_tax_status

# Reset to default values
dct_options(reset = TRUE)
dct_options()$valid_tax_status

# Multiple options may also be set at once
dct_options(check_taxon_id = FALSE, check_status_diff = TRUE)

# Reset options to those before this example was run
do.call(dct_options, old_settings)
```

---

dct_terms *Darwin Core Taxon terms*

Description

A table of valid Darwin Core terms. Only terms in the Taxon class or at the record-level are included.

Usage

```
dct_terms
```

Format

Dataframe (tibble), including two columns:

- `group`: Darwin Core term group; either "taxon" (terms in the Taxon class) or "record-level" (terms that are generic in that they might apply to any type of record in a dataset.)
- `term`: Darwin Core term

with two additional attributes:

- `retrieved`: Date the terms were obtained
- `url`: URL from which the terms were obtained

Details

Modified from data downloaded from TDWG Darwin Core under the Creative Commons Attribution (CC BY) 4.0 license.

Source

https://dwc.tdwg.org/terms/#taxon

Examples

```
dct_terms
```

`dct_validate` *Validate a taxonomic database*

Description

Runs a series of automated checks on a taxonomic database in Darwin Core (DwC) format.

Usage

```
dct_validate(
  tax_dat,
  check_taxon_id = dct_options()$check_taxon_id,
  check_tax_status = dct_options()$check_tax_status,
  check_mapping_accepted = dct_options()$check_mapping_accepted,
  check_mapping_parent = dct_options()$check_mapping_parent,
  check_mapping_original = dct_options()$check_mapping_original,
  check_mapping_accepted_status = dct_options()$check_mapping_accepted_status,
  check_sci_name = dct_options()$check_sci_name,
  check_status_diff = dct_options()$check_status_diff,
  check_col_names = dct_options()$check_col_names,
  valid_tax_status = dct_options()$valid_tax_status,
  extra_cols = dct_options()$extra_cols,
  on_success = dct_options()$on_success,
  on_fail = dct_options()$on_fail,
  skip_missing_cols = dct_options()$skip_missing_cols,
  quiet = dct_options()$quiet
)
```

Arguments

tax_dat	Dataframe; taxonomic database in DwC format.
check_taxon_id	Logical vector of length 1; should all instances of taxonID be required to be non-missing and unique? Default TRUE.
check_tax_status	Logical vector of length 1; should all taxonomic names be required to have a valid value for taxonomic status (by default, "accepted", "synonym", or "variant")? Default TRUE.
check_mapping_accepted	Logical vector of length 1; should all values of acceptedNameUsageID be required to map to the taxonID of an existing name? Default TRUE.
check_mapping_parent	Logical vector of length 1; should all values of parentNameUsageID be required to map to the taxonID of an existing name? Default TRUE.
check_mapping_original	Logical vector of length 1; should all values of originalNameUsageID be required to map to the taxonID of an existing name? Default TRUE.
check_mapping_accepted_status	Logical vector of length 1; should rules about mapping of variants and synonyms be enforced? Default FALSE. (see Details).
check_sci_name	Logical vector of length 1; should all instances of scientificName be required to be non-missing and unique? Default TRUE.
check_status_diff	Logical vector of length 1; should each scientific name be allowed to have only one taxonomic status? Default FALSE.
check_col_names	Logical vector of length 1; should all column names be required to be a valid DwC term? Default TRUE.
valid_tax_status	Character vector of length 1; valid values for taxonomicStatus. Each value must be separated by a comma. Default accepted, synonym, variant, NA. "NA" indicates that missing (NA) values are valid. Case-sensitive.
extra_cols	Character vector; names of columns that should be allowed beyond those defined by the DwC taxon standard. Default NULL. Providing column name(s) that are valid DwC taxon column(s) has no effect.
on_success	Character vector of length 1, either "logical" or "data". Describes what to do if the check passes. Default "data".
on_fail	Character vector of length 1, either "error" or "summary". Describes what to do if the check fails. Default "error".
skip_missing_cols	Logical vector of length 1; should checks be silently skipped if any of the columns they inspect are missing? Default FALSE.
quiet	Logical vector of length 1; should warnings be silenced? Default FALSE.

Details

For check_mapping_accepted_status and check_status_diff, "accepted", "synonym", and "variant" are determined by string matching of taxonomicStatus; so "provisionally accepted" is counted as "accepted", "ambiguous synonym" is counted as "synonym", etc. (case-sensitive).

For check_mapping_accepted_status, the following rules are enforced:

- Rows with taxonomicStatus of "synonym" (synonyms) must have an acceptedNameUsageID matching the taxonID of an accepted name (taxonomicStatus of "accepted")
- Rows with taxonomicStatus of "variant" (orthographic variants) must have an acceptedNameUsageID matching the taxonID of an accepted name or synonym (but not another variant)
- Rows with taxonomicStatus of "accepted" must not have any value entered for acceptedNameUsageID
- Rows with a value for acceptedNameUsageID must have a valid value for taxonomicStatus.

Default settings of all arguments can be modified with dct_options() (see Examples).

Most columns are expected to be vectors of class character, but this is not checked for all columns. Columns (DwC terms) with names including 'ID', for example 'taxonID', may be character, numeric, or integer.

Value

Depends on the result of the check and on values of on_fail and on_success:

- If the check passes and on_success is "logical", return TRUE
- If the check passes and on_success is "data", return the input dataframe
- If the check fails and on_fail is "error", return an error
- If the check fails and on_fail is "summary", issue a warning and return a dataframe with a summary of the reasons for failure

Examples

```
# The example dataset dct_filmies is already correctly formatted and passes
# validation
dct_validate(dct_filmies)

# So make some bad data on purpose with a duplicated scientific name
bad_dat <- dct_filmies
bad_dat$scientificName[1] <- bad_dat$scientificName[2]

# The incorrectly formatted data won't pass
try(
  dct_validate(bad_dat)
)

# It will pass if we allow duplicated scientific names though
dct_validate(bad_dat, check_sci_name = FALSE)

# Individual checks can also be turned or off with dct_options()
```

```
# First save the current settings before making any changes
old_settings <- dct_options()

# Let's allow duplicated scientific names by default
dct_options(check_sci_name = FALSE)

# The data passes validation as before, but we don't have to specify
# `check_sci_name = FALSE` in the function call
dct_validate(bad_dat)

# Reset options to those before this example was run
do.call(dct_options, old_settings)
```

Index