dwctaxon, an R package for editing and validating taxonomic data in Darwin Core format

Joel H. Nitta [1] **and Wataru Iwasaki** [2,3,4]

1 Graduate School of Global and Transdisciplinary Studies, Chiba University, Japan **2** Department of Integrated Biosciences, Graduate School of Frontier Sciences, The University of Tokyo, Chiba, Japan **3** Department of Biological Sciences, Graduate School of Science, The University of Tokyo, Tokyo, Japan **4** Atmosphere and Ocean Research Institute, The University of Tokyo, Chiba, Japan

Summary

The dwctaxon R package provides functions to edit and validate taxonomic data in compliance with the widely used Darwin Core (DwC) standard (Wieczorek et al., 2012) for biodiversity data. dwctaxon automates several data editing steps, thereby simplifying workflows and decreasing the chance of human-generated data entry errors. Furthermore, it conducts automated checks to validate taxonomic datasets, thereby alerting database maintainers to problems as soon as possible. dwctaxon will accelerate the generation and maintenance of taxonomic data, a critical part of the effort to document and conserve earth's biodiversity. dwctaxon has passed code review at rOpenSci. Complete documentation is available at https://docs.ropensci.org/dwctaxon and source code is available at https://github.com/ropensci/dwctaxon.

Statement of need

Taxonomic names, or taxa, are a fundamental unit of biodiversity (Janzen, 2003). Darwin Core (DwC) is a data standard for taxonomic and other biodiversity data, and provides a common format for biodiversity databases to interface with each other (Wieczorek et al., 2012). Darwin Core has been widely adopted by many databases (e.g., the Global Biodiversity Information Facility (GBIF), Catalog of Life, and World Flora Online).

Taxonomic data are frequently updated due to, for example, discovery of new species or revision of existing taxonomic concepts. These taxonomic changes can be complicated. For example, changing the status of one name to the synonym of another could potentially involve modifying many other names if they are synonyms of the first. Because of these properties, maintaining a taxonomic database in compliance with DwC can be onerous and error-prone if done by hand, especially if the database is large. There is therefore a need for software to automate entry and validation of DwC taxonomic data.

The dwctaxon R package addresses this need. dwctaxon can be used to edit DwC taxonomic databases by adding new rows or modifying existing data in compliance with DwC. dwctaxon automatically updates synonymy, thereby decreasing errors and simplifying workflow. Furthermore, it runs >15 automated checks to validate that taxonomic data are in compliance with DwC and can be used for taxonomic name resolution. We anticipate dwctaxon will reduce friction in maintaining taxonomic databases in the face of frequent updates, and be used from smaller projects (e.g., local biodiversity surveys) to large, public databases (e.g., GBIF).

Installation

The stable version can be installed from the Comprehensive R Archive Network (CRAN):

```
install.packages("dwctaxon")
```

The development version can be installed from r-universe:

```r
options(repos = c(
  ropensci = "https://ropensci.r-universe.dev/",
  CRAN = "https://cran.rstudio.com/"
))
install.packages("dwctaxon", dep = TRUE)
```

Features

Data editing

dwctaxon offers several functions for entering new data or manipulating existing data. All functions are designed to maximize compliance with DwC.

dct_add_row() is the simplest editing function, and appends a new row to the data. It will automatically generate a unique identifier (taxonID) for the row if one is not provided by the user, and only allows entering values for official DwC taxon columns (terms).

dct_modify_row() modifies the contents of an existing row, allowing the user to assign new values for any DwC terms (such as scientificName, acceptedNameUsage, taxonomicStatus, etc.). For a complete list of DwC terms, see the Darwin Core Quick Reference Guide. Although DwC does not maintain a standardized list of allowed values for taxonomicStatus, these typically include values like "accepted", "synonym", "invalid", etc. Changing a name from the status of accepted to synonym can involve several related modifications if the original name included one or more synonyms, as these also need to be updated to point to the new accepted name. dct_modify_row() automatically modifies such synonyms. This can reduce error since a given name may include a large number of synonyms that would otherwise be onerous to edit.

dct_fill_col() automatically fills in the values of one column by matching from another pair of columns. This is needed because the DwC format includes some columns that have very closely related data, which may be difficult to manually keep up to date (here called "term-termID pairs"). For example, there are two columns that specify the accepted name for synonyms: acceptedNameUsage (the actual accepted scientific name) and acceptedNameUsageID (the unique identifier, or taxonID, of the accepted name). The reason for having both columns is that since taxonID is expected to be unique, it is more accurate as a reference value. However, due to the formatting of taxonID (for example, a typical value may be a series of numbers and letters like 36HMM), it is difficult for a human to immediately know what scientific name that taxonID refers to; hence it is also helpful to have the name itself available alongside. dct_fill_col() can be used to automatically fill in such term-termID pairs, which is expected to greatly decrease risk of data entry error.

The default output of all functions is a tibble (data frame). This means that a series of changes can be applied using the pipe operator (|> or %>%). Furthermore, the data editing functions can also use a data frame or vector to provide the new data so that entries do not have to be made one per function call.

Data validation

dct_validate() carries out automated data validation to ensure that the taxonomic database is in compliance with DwC. At a minimum, this involves checking that the names of columns (terms) match those used by DwC. While DwC makes recommendations for values of some terms, for most of these it does not specify the actual values allowed. However, it is typical for taxonomic databases to make assumptions about the data based on their values. For example, we would expect that every synonym should be linked to an accepted name. Therefore, validation in dwctaxon is flexible, allowing the user to select from multiple checks.

Validation checks in `dwctaxon` include the following (each is a logical argument of `dct_validate()`):

- `check_taxon_id`: Check that all values of `taxonID` are unique and non-missing.

- `check_tax_status`: Check that all values of `taxonomicStatus` are valid. Default valid values include "accepted", "synonym", or "variant", but these can be set via the `valid_tax_status` argument.

- `check_mapping`: Check that all values of `acceptedNameUsageID` map to the `taxonID` of an existing name.

- `check_mapping_strict`: Check that mapping of names follows expectations based on their taxonomic status (e.g., synonyms must map to accepted names, accepted names cannot have an `acceptedNameUsageID`, etc.).

- `check_sci_name`: Check that all instances of `scientificName` are non-missing and unique.

- `check_status_diff`: Check that each scientific name be allowed to have only one taxonomic status (typically only used if `check_sci_name` is not used).

- `check_col_names`: Check that all column names are valid Darwin Core terms.

There are multiple options for the output of `dct_validate()`:

- If the `on_fail` argument is set to `"error"`, an error will be raised and computation stopped at the first failing check.

- If `on_fail` is set to `"summary"`, if any check fails a summary table (tibble) will be returning describing failures for all checks, with a warning.

- If the `on_success` argument is set to `"data"` and all checks pass, the original data will be returned.

- If `on_success` is set to `"logical"` and all checks pass, a logical value of `TRUE` will be returned.

Each check can also be carried out as a separate function (e.g., `dct_check_taxon_id`), with the same output options.

Examples and documentation

Examples of code usage, an overview of the DwC taxon data format, and complete description of all functions are available on the `dwctaxon` website, https://docs.ropensci.org/dwctaxon. The examples include editing and validating a small dataset that comes with `dwctaxon` as well as a real-world example using the Database of Vascular Plants of Canada (Desmet & Brouilet, 2013).

Benchmark on real data

One potential limitation of data validation that relies on in-memory data loaded into R (via `load.csv()` etc.) is that extremely large databases could exceed R's memory capacity.

To test this, we ran `dct_validate()` on the GBIF backbone taxonomy, which is to our knowledge the largest single database in DwC format currently available online with 6,312,410 rows (names) in the most recent release (2023-08-28) as of writing. The test was carried out on an Apple iMac (2019, 16 GB RAM) with R v4.3.2 and `dwctaxon` v2.0.3. The code and results are reproduced below:

```
library(dwctaxon)
```

```r
# This assumes backbone.zip has been downloaded from
# https://doi.org/10.15468/39omei
# and unzipped in the current working directory

# Load GBIF backbone taxonomy
gbif_backbone <- readr::read_tsv("backbone/Taxon.tsv")

# Validate taxonomy
gbif_check <- dct_validate(
  gbif_backbone,
  # - set valid values for taxonomicStatus as used by GBIF
  valid_tax_status = paste(
    "accepted, doubtful, heterotypic synonym,",
    "homotypic synonym,", "proparte synonym,",
    "synonym"),
  on_fail = "summary")
```

```
Warning: taxonID detected whose acceptedNameUsageID value does not map to
taxonID of an existing name.
taxonID detected whose parentNameUsageID value does not map to taxonID
of an existing name.
taxonID detected whose originalNameUsageID value does not map to
taxonID of an existing name.
scientificName detected with duplicated value.
Invalid column names detected canonicalName.
```

```r
# Count types of errors detected
table(gbif_check$error)
```

```
Invalid column names detected: canonicalName
1
scientificName detected with duplicated value
14397
taxonID detected whose acceptedNameUsageID value does not map to
taxonID of an existing name.
454500
taxonID detected whose originalNameUsageID value does not map to
taxonID of an existing name.
117005
taxonID detected whose parentNameUsageID value does not map to taxonID
of an existing name.
1259875
```

We see that GBIF uses one non-standard column, `canonicalName`. 14,397 scientific names have duplicated values. 1,831,380 names have a value for `acceptedNameUsageID`, `originalNameUsageID`, or `parentNameUsageID` that does not map to any `taxonID`. It should be noted that these results do not necessarily indicate that the GBIF database is deficit in practice. It is up to the database maintainer to interpret the results of the validation and decide upon the appropriate course of action, if any.

The time to run `dct_validate()` even on this relatively large database was quite short: 14.9 seconds. Therefore, there are likely to be few situations where `dwctaxon` cannot be used due to database size.

Similar software

The only similar tool we are aware of is the GBIF data validator (https://www.gbif.org/tools/data-validator), an online system for validating data in DwC format. However, the GBIF data validator is not available for use as a local program and is limited to input sizes of 100 mb. It does not conduct editing of DwC compliant data, nor are we aware of any other software that does so.

Conclusion

With its automated yet flexible interface to update and validate taxonomic data in compliance with DwC, dwctaxon fills a critical gap in taxonomic data software. We hope it will become widely used and support the taxonomic infrastructure that undergirds efforts to document and preserve biodiversity.

Reproducibility

Code to generate this manuscript is available at https://github.com/joelnitta/dwctaxon_ms.

Acknowledgements

References

Desmet, P., & Brouilet, L. (2013). Database of vascular plants of canada (VASCAN): A community contributed taxonomic checklist of all vascular plants of canada, saint pierre and miquelon, and greenland. *PhytoKeys*, *25*, 55–67. https://doi.org/10.3897/phytokeys.25.3100

Janzen, D. H. (2003). How does an "all taxa biodiversity inventory (ATBI)" promote and facilitate local and global biodiversity conservation? *Biodiversity*, *4*(2), 4–10. https://doi.org/10.1080/14888386.2003.9712683

Wieczorek, J., Bloom, D., Guralnick, R., Blum, S., Döring, M., Giovanni, R., Robertson, T., & Vieglais, D. (2012). Darwin core: An evolving community-developed biodiversity data standard. *PLoS ONE*, *7*(1), e29715. https://doi.org/10.1371/journal.pone.0029715