# Package: bdc (via r-universe)

September 10, 2024

**Title** Biodiversity Data Cleaning

**Version** 1.1.5

**Description** It brings together several aspects of biodiversity data-cleaning in one place. 'bdc' is organized in thematic modules related to different biodiversity dimensions, including 1) Merge datasets: standardization and integration of different datasets; 2) Pre-filter: flagging and removal of invalid or non-interpretable information, followed by data amendments; 3) Taxonomy: cleaning, parsing, and harmonization of scientific names from several taxonomic groups against taxonomic databases locally stored through the application of exact and partial matching algorithms; 4) Space: flagging of erroneous, suspect, and low-precision geographic coordinates; and 5) Time: flagging and, whenever possible, correction of inconsistent collection date. In addition, it contains features to visualize, document, and report data quality – which is essential for making data quality assessment transparent and reproducible. The reference for the methodology is Bruno et al. (2022) <doi:10.1111/2041-210X.13868>.

**License** GPL (>= 3)

**URL** https://brunobrr.github.io/bdc/ (website)
https://github.com/brunobrr/bdc

**BugReports** https://github.com/brunobrr/bdc/issues

**Imports** CoordinateCleaner, doParallel, dplyr, DT, foreach, fs, ggplot2, here, magrittr, purrr, qs, readr, rgnparser, rnaturalearth, sf (>= 1.0.5), stringdist, stringi, stringr, taxadb (>= 0.1.3), tibble, tidyselect

**Suggests** contentid (>= 0.0.15), covr, cowplot, DBI, duckdb (>= 0.3.2), knitr (>= 1.31), maps, markdown, rappdirs, raster, remotes, rlang (>= 1.0.1), rmarkdown, rnaturalearthdata, rvest, sp, testthat (>= 3.0.0), xml2

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-gb

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.1

**Repository** https://brunobrr.r-universe.dev

**RemoteUrl** https://github.com/brunobrr/bdc

**RemoteRef** HEAD

**RemoteSha** 2b8ce0e45cca46559aadcccfa7bae37f6e084c65

# Contents



---

`bdc_basisOfRecords_notStandard`

*Identify records from doubtful source (e.g., 'fossil', MachineObservation')*

---

## Description

This function flags records with an informed basis of records (i.e., the records type, for example, a specimen, a human observation, or a fossil specimen) not interpretable, which does not comply with Darwin Core vocabulary, or unreliable or unsuitable for specific analyses.

## Usage

```
bdc_basisOfRecords_notStandard(
  data,
  basisOfRecord = "basisOfRecord",
  names_to_keep = "all"
)
```

## Arguments

| | |
|---|---|
| data | data.frame. Containing information about the basis of records. |
| basisOfRecord | character string. The column name with information about basis of records. Default = "basisOfRecord". |
| names_to_keep | character string. Elements of the column BasisOfRecords to keep. Default is "all", which considers a selected list of recommended standard Darwin Core classes (and their spelling variations, see details). By default, records missing (i.e., NA) or with "unknown" information about basis of records are kept. |

## Details

Users are encourage to select the set of basis of records classes to keep. Default = c("Event","HUMAN_OBSERVATION", "HumanObservation", "LIVING_SPECIMEN", "LivingSpecimen", "MACHINE_OBSERVATION", "MachineObservation", "MATERIAL_SAMPLE", "O", "Occurrence", "MaterialSample", "OBSERVATION", "Preserved Specimen", "PRESERVED_SPECIMEN", "preservedspecimen Specimen", "Preservedspecimen", "PreservedSpecimen", "preservedspecimen", "S", "Specimen", "Taxon", "UNKNOWN", "", NA)

## Value

A data.frame containing the column ".basisOfRecords_notStandard" .Compliant (TRUE) if 'basisOfRecord' is standard; otherwise "FALSE".

## See Also

Other prefilter: bdc_coordinates_country_inconsistent(), bdc_coordinates_empty(), bdc_coordinates_from_loc bdc_coordinates_outOfRange(), bdc_coordinates_transposed(), bdc_country_standardized(), bdc_scientificName_empty()

## Examples

```
x <- data.frame(basisOfRecord = c(
  "FOSSIL_SPECIMEN", "UNKNOWN",
  "RON", NA, "Specimen", "PRESERVED_SPECIMEN"
))

bdc_basisOfRecords_notStandard(
  data = x,
  basisOfRecord = "basisOfRecord",
  names_to_keep = "all"
)
```

`bdc_clean_names` *Clean and parse scientific names*

## Description

This function is composed of a series of name-checking routines for cleaning and parsing scientific names; i.e., unify writing style. It removes 1) family names of animals or plants pre-pended to species names, 2) qualifiers denoting the uncertain or provisional status of taxonomic identification (e.g., confer, species, affinis), and 3) infraspecific terms, for example, variety (var.), subspecies (subsp), forma (f.), and their spelling variations. It also includes applications to 4) standardize names, i.e., capitalize only the first letter of the genus name and remove extra whitespaces), and 5) parse names, i.e., separate author, date, annotations from taxon name.

## Usage

```
bdc_clean_names(sci_names, save_outputs = FALSE)
```

## Arguments

| | |
|---|---|
| `sci_names` | character string. Containing scientific names. |
| `save_outputs` | logical. Should the outputs be saved? Default = FALSE. |

## Details

Terms denoting uncertainty or provisional status of taxonomic identification as well as infraspecific terms were obtained from Sigoviniet al. (2016; doi: 10.1111/2041-210X.12594). More details about the names parse process can be found in gnparser. **Note: GNparser is not automatically installed. Please see guidelines to install gnparser.**

## Value

A five-column data.frame including

- scientificName: original names supplied
- .uncer_terms: indicates the presence of taxonomic uncertainty terms
- .infraesp_names: indicates the presence of infraspecific terms
- name_clean: scientific names resulting from the cleaning and parsing processes
- quality: an index indicating the quality of parsing process. It ranges from 0 to 4, being 1 no problem detected, 4 serious problems detected; a value of 0 indicates no interpretable name that was not parsed).

If save_outputs == TRUE, a data.frame containing all tests of the cleaning names process and the results of the parsing names process is saved in "Output/Check/02_parse_names.csv".

## See Also

Other taxonomy: `bdc_filter_out_names()`, `bdc_query_names_taxadb()`

## Examples

```
## Not run:
scientificName <- c(
  "Fridericia bahiensis (Schauer ex. DC.) L.G.Lohmann",
  "Peltophorum dubium (Spreng.) Taub. (Griseb.) Barneby",
  "Gymnanthes edwalliana (Pax & K.Hoffm.) Laurenio-Melo & M.F.Sales",
  "LEGUMINOSAE Senna aff. organensis (Glaz. ex Harms) H.S.Irwin & Barneby"
)

bdc_clean_names(scientificName, save_outputs = FALSE)

## End(Not run)
```

---

# bdc_coordinates_country_inconsistent

*Identify records within a reference country*

---

## Description

This function flags geographic coordinates within a reference country. A spatial buffer can be added to the reference country to ensure that records in mangroves, marshes, estuaries, and records with low coordinate precision are not flagged as invalid.

## Usage

```
bdc_coordinates_country_inconsistent(
  data,
  country_name,
  country = "country_suggested",
  lat = "decimalLatitude",
  lon = "decimalLongitude",
  dist = 0.1
)
```

## Arguments

| | |
|---|---|
| data | data.frame. Containing longitude and latitude. Coordinates must be expressed in decimal degrees and WGS84. |
| country_name | character string. Name of the country or countries to be considered. |
| country | character string. The column name with the country assignment of each record. It is recommended use a column with corrected and homogenized country names. Default = "country_suggested". |
| lat | character string. The column name with the latitude coordinates. Default = “decimallatitude”. |
| lon | character string. The column name with the longitude coordinates. Default = “decimallongitude”. |

`dist` numeric. The distance in decimal degrees used to created a buffer around the country. Default = 0.1 (~11 km at the equator).

## Details

Multiple countries can be informed, but they are tested separately. The distance reported in the argument 'dist' is used to create a buffer around the reference country. Records within the reference country or at a specified distance from the coastline of the reference country (i.e., records within the buffer) are flagged as valid (TRUE). Note that records within the buffer but in other countries are flagged as invalid (FALSE). Records with invalid (e.g., NA or empty) and out-of-range coordinates are not tested and returned as TRUE.

## Value

A data.frame containing the column '.coordinates_country_inconsistent'. Compliant (TRUE) if coordinates fall within the boundaries plus a specified distance (if 'dist' is supplied) of 'country_name'; otherwise "FALSE".

## See Also

Other prefilter: bdc_basisOfRecords_notStandard(), bdc_coordinates_empty(), bdc_coordinates_from_locality(
bdc_coordinates_outOfRange(), bdc_coordinates_transposed(), bdc_country_standardized(),
bdc_scientificName_empty()

## Examples

```
## Not run:
x <- data.frame(
  country = c("Brazil", "Brazil", "Bolivia", "Argentina", "Peru"),
  decimalLongitude = c(-40.6003, -39.6, -77.689288, NA, -76.352930),
  decimalLatitude = c(-19.9358, -13.016667, -20.5243, -35.345940, -11.851872)
)

bdc_coordinates_country_inconsistent(
  data = x,
  country_name = c("Brazil", "Peru", "Argentina"),
  country = "country",
  lon = "decimalLongitude",
  lat = "decimalLatitude",
  dist = 0.1
)

## End(Not run)
```

`bdc_coordinates_empty` *Identify records with empty geographic coordinates*

## Description

This function flags records missing latitude or longitude coordinates.

## Usage

```
bdc_coordinates_empty(data, lat = "decimalLatitude", lon = "decimalLongitude")
```

## Arguments

| | |
|---|---|
| `data` | data.frame. Containing geographical coordinates. |
| `lat` | character string. The column name with latitude in decimal degrees and WGS84. Default = "decimalLatitude". |
| `lon` | character string. The column with longitude in decimal degrees and WGS84. Default = "decimalLongitude". |

## Details

This test identifies records missing geographic coordinates (i.e., empty or not applicable NA longitude or latitude)

## Value

A data.frame containing the column ".coordinates_empty". Compliant (TRUE) if 'lat' and 'lon' are not empty; otherwise "FALSE".

## See Also

Other prefilter: `bdc_basisOfRecords_notStandard()`, `bdc_coordinates_country_inconsistent()`, `bdc_coordinates_from_locality()`, `bdc_coordinates_outOfRange()`, `bdc_coordinates_transposed()`, `bdc_country_standardized()`, `bdc_scientificName_empty()`

## Examples

```
x <- data.frame(
  decimalLatitude = c(19.9358, -13.016667, NA, ""),
  decimalLongitude = c(-40.6003, -39.6, -20.5243, NA)
)

bdc_coordinates_empty(
  data = x,
  lat = "decimalLatitude",
  lon = "decimalLongitude"
)
```

`bdc_coordinates_from_locality` *Identify records lacking or with invalid coordinates but containing locality information*

## Description

This function Identifies records whose coordinates can potentially be extracted from locality information.

## Usage

```
bdc_coordinates_from_locality(
  data,
  lat = "decimalLatitude",
  lon = "decimalLongitude",
  locality = "locality",
  save_outputs = FALSE
)
```

## Arguments

| | |
|---|---|
| `data` | data.frame. Containing geographical coordinates and the column "locality'. |
| `lat` | character string. The column name with latitude in decimal degrees and WGS84. Default = "decimalLatitude". |
| `lon` | character string. The column with longitude in decimal degrees and WGS84. Default = "decimalLongitude". |
| `locality` | character string. The column name with locality information. Default = "locality". |
| `save_outputs` | logical. Should a table containing transposed coordinates saved for further inspection? Default = FALSE. |

## Details

According to DarwinCore terminology, locality refers to "the specific description of the place" where an organism was recorded.

## Value

A data.frame containing records missing or with invalid coordinates but with potentially useful locality information. When save_outputs = FALSE the data.frame is saved in Output/Check/01_coordinates_from_locality.csv

## See Also

Other prefilter: `bdc_basisOfRecords_notStandard()`, `bdc_coordinates_country_inconsistent()`, `bdc_coordinates_empty()`, `bdc_coordinates_outOfRange()`, `bdc_coordinates_transposed()`, `bdc_country_standardized()`, `bdc_scientificName_empty()`

## Examples

```
x <- data.frame(
  lat = c(NA, NA, ""),
  lon = c("", NA, NA),
  locality = c("PARAGUAY: ALTO PARAGUAY: CO.; 64KM W PUERTO SASTRE",
               "Parque Estadual da Serra de Caldas Novas, Goias, Brazil",
               "Parque Nacional Iguazu"))

bdc_coordinates_from_locality(
data = x,
lat = "lat",
lon = "lon",
locality = "locality",
save_outputs = FALSE)
```

---

# bdc_coordinates_outOfRange

*Identify records with out-of-range geographic coordinates*

---

## Description

This function identifies records with out-of-range coordinates (not between -90 and 90 for latitude; between -180 and 180 for longitude).

## Usage

```
bdc_coordinates_outOfRange(
  data,
  lat = "decimalLatitude",
  lon = "decimalLongitude"
)
```

## Arguments

| | |
|---|---|
| data | data.frame. Containing geographical coordinates. Coordinates must be expressed in decimal degrees and WGS84. |
| lat | character string. The column name with latitude in decimal degree and in WGS84. Default = "decimalLatitude". |
| lon | character string. The column with longitude in decimal degree and in WGS84. Default = "decimalLongitude". |

## Value

A data.frame containing the column ".coordinates_outOfRange". Compliant (TRUE) if 'lat' and 'lon' are not out-of-range; otherwise "FALSE".

## See Also

Other prefilter: bdc_basisOfRecords_notStandard(), bdc_coordinates_country_inconsistent(), bdc_coordinates_empty(), bdc_coordinates_from_locality(), bdc_coordinates_transposed(), bdc_country_standardized(), bdc_scientificName_empty()

## Examples

```
x <- data.frame(
  decimalLatitude = c(-185.111, -43.34, "", -21.8069444),
  decimalLongitude = c(-45.4, -39.6, -20.5243, -440.9055555)
)

bdc_coordinates_outOfRange(
  data = x,
  lat = "decimalLatitude",
  lon = "decimalLongitude"
)
```

---

# bdc_coordinates_precision

*Flag low-precise geographic coordinates*

---

## Description

This function flags records with a coordinate precision below a specified number of decimal places. Coordinates with one, two, or three decimal places present a precision of~11.1 km, ~1.1 km, and ~111 m at the equator, respectively.

## Usage

```
bdc_coordinates_precision(
  data,
  lat = "decimalLatitude",
  lon = "decimalLongitude",
  ndec = c(0, 1, 2)
)
```

## Arguments

| | |
|---|---|
| data | data.frame. A data.frame containing geographic coordinates. |
| lat | character string. The column with latitude in decimal degrees and WGS84. Default = "decimalLatitude". |
| lon | character string. The column with longitude in decimal degrees and WGS84. Default = "decimalLongitude". |
| ndec | numeric. The minimum number of decimal places that the coordinates should have to be considered valid. Default = 2. |

## Value

A data.frame with logical values indicating whether values are equal or higher than the specified minimum decimal number (ndec). Coordinates flagged as FALSE in .rou column are considered imprecise.

## Examples

```
x <- data.frame(
  lat = c(-21.34, 23.567, 16.798, -10.468),
  lon = c(-55.38, -13.897, 30.8, 90.675)
)

bdc_coordinates_precision(
  data = x,
  lat = "lat",
  lon = "lon",
  ndec = 3
)
```

---

# bdc_coordinates_transposed

*Identify transposed geographic coordinates*

---

## Description

This function flags and corrects records when latitude and longitude appear to be transposed.

## Usage

```
bdc_coordinates_transposed(
  data,
  id = "database_id",
  sci_names = "scientificName",
  lat = "decimalLatitude",
  lon = "decimalLongitude",
  country = "country",
  countryCode = "countryCode",
  border_buffer = 0.2,
  save_outputs = FALSE
)
```

## Arguments

| | |
|---|---|
| data | data.frame. Containing a unique identifier for each record, geographical coordinates, and country names. Coordinates must be expressed in decimal degrees and WGS84. |

| | |
|---|---|
| id | character string. The column name with a unique record identifier. Default = "database_id". |
| sci_names | character string. The column name with species scientific name. Default = "scientificName". |
| lat | character string. The column name with latitude. Coordinates must be expressed in decimal degrees and WGS84. Default = "decimalLatitude". |
| lon | character string. The column with longitude. Coordinates must be expressed in decimal degrees and WGS84. Default = "decimalLongitude". |
| country | character string. The column name with the country assignment of each record. Default = "country". |
| countryCode | character string. The column name with an ISO-2 country code. |
| border_buffer | numeric >= 0. A distance in decimal degrees used to created a buffer around the country. Records within a given country and at a specified distance from the border will be not be corrected. Default = 0.2 (~22 km at the equator). |
| save_outputs | logical. Should a table containing transposed coordinates saved for further inspection? Default = FALSE. |

## Details

This test identifies transposed coordinates resulted from mismatches between the country informed for a record and coordinates. Transposed coordinates often fall outside of the indicated country (i.e., in other countries or in the sea). Different coordinate transformations are performed to correct country/coordinates mismatches. Importantly, verbatim coordinates are replaced by the corrected ones in the returned database. A database containing verbatim and corrected coordinates is created in "Output/Check/01_coordinates_transposed.csv" if save_outputs == TRUE. The columns "country" and "countryCode" can be retrieved by using the function bdc_country_standardized.

## Value

A data.frame containing the column "coordinates_transposed" indicating if verbatim coordinates were not transposed (TRUE). Otherwise records are flagged as (FALSE) and, in this case, verbatim coordinates are replaced by corrected coordinates.

## See Also

Other prefilter: bdc_basisOfRecords_notStandard(), bdc_coordinates_country_inconsistent(), bdc_coordinates_empty(), bdc_coordinates_from_locality(), bdc_coordinates_outOfRange(), bdc_country_standardized(), bdc_scientificName_empty()

## Examples

```
## Not run:
id <- c(1, 2, 3, 4)
scientificName <- c(
  "Rhinella major", "Scinax ruber",
  "Siparuna guianensis", "Psychotria vellosiana"
)
decimalLatitude <- c(63.43333, -14.43333, -41.90000, -46.69778)
```

```
decimalLongitude <- c(-17.90000, -67.91667, -13.25000, -13.82444)
country <- c("BOLIVIA", "bolivia", "Brasil", "Brazil")

x <- data.frame(
  id, scientificName, decimalLatitude,
  decimalLongitude, country
)

# Get country code
x <- bdc_country_standardized(data = x, country = "country")

bdc_coordinates_transposed(
  data = x,
  id = "id",
  sci_names = "scientificName",
  lat = "decimalLatitude",
  lon = "decimalLongitude",
  country = "country_suggested",
  countryCode = "countryCode",
  border_buffer = 0.2,
  save_outputs = FALSE
)

## End(Not run)
```

---

`bdc_country_from_coordinates`

*Get country names from coordinates*

---

## Description

Country names derived from valid geographic coordinates are added to records missing country names.

## Usage

```
bdc_country_from_coordinates(
  data,
  lat = "decimalLatitude",
  lon = "decimalLongitude",
  country = "country"
)
```

## Arguments

| | |
|---|---|
| `data` | data.frame. Containing geographical coordinates and country names. |
| `lat` | character string. The column name with latitude in decimal degrees and WGS84. Default = "decimalLatitude". |

| | |
|---|---|
| lon | character string. The column with longitude in decimal degrees and WGS84. Default = "decimalLongitude". |
| country | character string. The column name with the country assignment of each record. Default = "country". If no column name is provided a new column "country" is created. |

## Details

This function assigns a country name for records missing such information. Country names are extracted from valid geographic coordinates using a high-quality map of the world (rnaturalearth package). No country name is added to records whose coordinates are in the sea.

## Value

A tibble containing country names for records missing such information.

## Examples

```
## Not run:
x <- data.frame(
  decimalLatitude = c(-22.9834, -39.857030, -17.06811, -46.69778),
  decimalLongitude = c(-69.095, -68.443588, 37.438108, -13.82444),
  country = c("", NA, NA, "Brazil"))

bdc_country_from_coordinates(
  data = x,
  lat = "decimalLatitude",
  lon = "decimalLongitude",
  country = "country"
)

## End(Not run)
```

---

# bdc_country_standardized

*Standardizes country names and gets country code*

---

## Description

This function standardizes country names and adds a new column to the database containing two-letter country codes (ISO 3166-1 alpha-2).

## Usage

```
bdc_country_standardized(data, country = "country")
```

### Arguments

| | |
|---|---|
| data | data.frame. Containing country names |
| country | character string. The column name with the country assignment of each record. Default = "country". |

### Details

Country names are standardized using an exact matching against a list of country names in several languages from International Organization for Standardization. If any unmatched names remain, a fuzzy matching algorithm is used to find potential candidates for each misspelled countries names.

### Value

A data.frame containing two columns: country_suggested (standardized country names) and country_code (two-letter country codes; more details in World Countries, International Organization for Standardization).

### See Also

Other prefilter: bdc_basisOfRecords_notStandard(), bdc_coordinates_country_inconsistent(), bdc_coordinates_empty(), bdc_coordinates_from_locality(), bdc_coordinates_outOfRange(), bdc_coordinates_transposed(), bdc_scientificName_empty()

### Examples

```
## Not run:
country <- c("BOLIVIA", "bolivia", "Brasil", "Brazil", "BREZIL")
x <- data.frame(country)

bdc_country_standardized(
  data = x,
  country = "country"
)

## End(Not run)
```

---

## bdc_create_figures *Create figures reporting the results of the bdc package*

---

### Description

Creates figures (i.e., bar plots, maps, and histograms) reporting the results of data quality tests implemented in the bdc package.

## Usage

```
bdc_create_figures(
  data,
  database_id = "database_id",
  workflow_step = NULL,
  save_figures = FALSE
)
```

## Arguments

| | |
|---|---|
| data | data.frame. Containing the results of data quality tests; that is, columns starting wit ".". |
| database_id | character string. The column name with a unique record identifier. Default = "database_id". |
| workflow_step | character string. Name of the workflow step. Options available are "prefilter", "space", and "time". |
| save_figures | logical. Should the figures be saved for further inspection? Default = FALSE. |

## Details

This function creates figures based on the results of data quality tests implemented. A pre-defined list of test names is used for creating figures depending on the name of the workflow step informed. Figures are saved in "Output/Figures" if save_figures == TRUE.

## Value

List containing figures showing the results of data quality test implemented in one module of bdc. When save_figures = TRUE, figures are also saved locally in a png format.

## Examples

```
## Not run:
database_id <- c("GBIF_01", "GBIF_02", "GBIF_03", "FISH_04", "FISH_05")
lat <- c(-19.93580, -13.01667, -22.34161, -6.75000, -15.15806)
lon <- c(-40.60030, -39.60000, -49.61017, -35.63330, -39.52861)
.scientificName_emptys <- c(TRUE, TRUE, TRUE, FALSE, FALSE)
.coordinates_empty <- c(TRUE, TRUE, TRUE, TRUE, TRUE)
.invalid_basis_of_records <- c(TRUE, FALSE, TRUE, FALSE, TRUE)
.summary <- c(TRUE, FALSE, TRUE, FALSE, FALSE)

x <- data.frame(
  database_id,
  lat,
  lon,
  .scientificName_emptys,
  .coordinates_empty,
  .invalid_basis_of_records,
  .summary
)
```

```
figures <-
bdc_create_figures(
  data = x,
  database_id = "database_id",
  workflow_step = "prefilter",
  save_figures = FALSE
)

## End(Not run)
```

---

bdc_create_report *Create a report summarizing the results of data quality tests*

---

## Description

Create a report summarizing the results of data quality tests

## Usage

```
bdc_create_report(
  data,
  database_id = "database_id",
  workflow_step,
  save_report = FALSE
)
```

## Arguments

| | |
|---|---|
| data | data.frame. Containing a unique identifier for each record and the results of data quality tests. |
| database_id | character string. The column name with a unique record identifier.Default = "database_id". |
| workflow_step | character string containing the following options("prefilter", "taxonomy", "space" or "time"). |
| save_report | logical. Should the report be saved for further inspection? Default = FALSE. |

## Value

A data.frame containing a report summarizing the results of data quality assessment.

## Examples

```
## Not run:
database_id <- c("test_1", "test_2", "test_3", "test_4", "test_5")
.missing_names <- c(TRUE, TRUE, TRUE, FALSE, FALSE)
.missing_coordinates <- c(TRUE, FALSE, FALSE, TRUE, FALSE)
.basisOfRecords_notStandard <- c(TRUE, TRUE, FALSE, TRUE, TRUE)
```

```
.summary <- c(TRUE, FALSE, FALSE, FALSE, FALSE)

x <- data.frame(
  database_id,
  .missing_names,
  .missing_coordinates,
  .basisOfRecords_notStandard,
  .summary
)

report <-
bdc_create_report(
  data = x,
  database_id = "database_id",
  workflow_step = "prefilter",
  save_report = FALSE
)

## End(Not run)
```

---

`bdc_eventDate_empty` *Identify records with empty event date*

---

## Description

This function identifies records missing information on an event date (i.e., when a record was collected or observed).

## Usage

```
bdc_eventDate_empty(data, eventDate = "eventDate")
```

## Arguments

| | |
|---|---|
| `data` | A data frame containing column with event date information. |
| `eventDate` | Numeric or date. The column with event date information. |

## Details

This test identifies records missing event date information (i.e., empty or not applicable NA).

## Value

A data.frame containing the column ".eventDate_empty". Compliant (TRUE) if 'eventDate' is not empty; otherwise "FALSE".

## See Also

Other time: `bdc_year_from_eventDate()`, `bdc_year_outOfRange()`

## Examples

```
collection_date <- c(
  NA, "31/12/2015", "2013-06-13T00:00:00Z", "2013-06-20",
  "", "2013", "0001-01-00"
)
x <- data.frame(collection_date)

bdc_eventDate_empty(data = x, eventDate = "collection_date")
```

---

# bdc_filter_out_flags *Remove columns with the results of data quality tests*

---

## Description

This function filters out columns containing the results of data quality tests (i.e., columns starting with '.') or other columns specified.

## Usage

```
bdc_filter_out_flags(data, col_to_remove = "all")
```

## Arguments

| | |
|---|---|
| data | data.frame. Containing columns to be removed. |
| col_to_remove | logical. Which columns should be removed? Default = "all", which means that all columns containing the results of data quality tests are removed. |

## Value

A data.frame without columns specified in 'col_to_remove'.

## Examples

```
x <- data.frame(
  database_id = c("test_1", "test_2", "test_3", "test_4", "test_5"),
   kindom = c("Plantae", "Plantae", "Animalia", "Animalia", "Plantae"),
  .bdc_scientificName_empty = c(TRUE, TRUE, TRUE, FALSE, FALSE),
  .bdc_coordinates_empty = c(TRUE, FALSE, FALSE, FALSE, FALSE),
  .bdc_coordinates_outOfRange = c(TRUE, FALSE, FALSE, FALSE, FALSE),
  .summary = c(TRUE, FALSE, FALSE, FALSE, FALSE)
)

bdc_filter_out_flags(
  data = x,
  col_to_remove = "all"
)
```

`bdc_filter_out_names` *Filter out records according to their taxonomic status*

## Description

This function is useful for selecting records according to their taxonomic status. By default, only records with accepted scientific names are returned.

## Usage

```
bdc_filter_out_names(
  data,
  col_name = "notes",
  taxonomic_status = "accepted",
  opposite = FALSE
)
```

## Arguments

| | |
|---|---|
| `data` | data.frame. Containing the column "notes" with information on the taxonomic status of scientific names. |
| `col_name` | character string. The column name containing notes about the taxonomic status of a name. Default = "notes". |
| `taxonomic_status` | character string. Taxonomic status of a name. Default = "accepted". |
| `opposite` | logical. Should taxonomic status different from those listed in 'taxonomic_status' be returned? Default = FALSE |

## Details

By default, only records with accepted scientific names are kept in the database. Such records are listed in the column 'taxonomic_status' as "accepted", "accepted | replaceSynonym", "accepted | wasMisspelled" or "accepted | wasMisspelled | replaceSynonym". It is also possible to customize the list of taxonomic notes to be kept in the argument 'taxonomic_status'. See 'notes' in the data.frame resulted from the function `bdc_create_report`. If 'opposite' is TRUE, records with notes different from names listed in 'taxonomic_status' are returned.

## Value

A data.frame filtered out according to names listed in 'taxonomic_status'.

## See Also

Other taxonomy: `bdc_clean_names()`, `bdc_query_names_taxadb()`

## Examples

```
df_notes <-
  data.frame(
    notes = c(
      "notFound", "accepted", "accepted | replaceSynonym",
      "accepted | wasMisspelled",
      "accepted | wasMisspelled | replaceSynonym",
      "multipleAccepted",
      "heterotypic synonym"
    )
  )

bdc_filter_out_names(
  data = df_notes,
  taxonomic_status = "accepted",
  col_name = "notes",
  opposite = FALSE
)
```

---

# bdc_query_names_taxadb

*Harmonizing taxon names against local stored taxonomic databases*

---

## Description

Harmonizing taxon names against local stored taxonomic databases

## Usage

```
bdc_query_names_taxadb(
  sci_name,
  replace_synonyms = TRUE,
  suggest_names = TRUE,
  suggestion_distance = 0.9,
  db = "gbif",
  rank_name = NULL,
  rank = NULL,
  parallel = FALSE,
  ncores = 2,
  export_accepted = FALSE
)
```

## Arguments

| | |
|---|---|
| sci_name | character string. Containing scientific names to be queried. |
| replace_synonyms | logical. Should synonyms be replaced by accepted names? Default = TRUE. |

suggest_names logical. Tries to find potential candidate names for misspelled names not resolved by an exact match. Default = TRUE.

suggestion_distance numeric. A threshold value determining the acceptable orthographical distance between searched and candidate names. Names with matching distance value lower threshold informed are returned as NA. Default = 0.9.

db character string. The name of the taxonomic database to be used in harmonizing taxon names. Default = "gbif". Use "all" to install all available taxonomic databases automatically.

rank_name character string. Taxonomic rank name (e.g. "Plantae", "Animalia", "Aves", "Carnivora". Default = NULL.

rank character string. A taxonomic rank used to filter the taxonomic database. Options available are: "kingdom", "phylum", "class", "order", "family", and "genus".

parallel logical. Should a parallelization process be used? Default=FALSE

ncores numeric. The number of cores to run in parallel.

export_accepted logical. Should a table containing records with names linked to multiple accepted names saved for further inspection. Default = FALSE.

## Details

The taxonomic harmonization is based upon one taxonomic authority database. The lastest version of each database is used to perform queries, but note that only older versions are available for some taxonomic databases. The database version is shown in parenthesis. Note that some databases are momentary unavailable in taxadb.

- **itis**: Integrated Taxonomic Information System (v. 2022)
- **ncbi**: National Center for Biotechnology Information (v. 2022)
- **col**: Catalogue of Life (v. 2022)
- **tpl**: The Plant List (v. 2019)
- **gbif**: Global Biodiversity Information Facility (v. 2022)
- **fb**: FishBase (v. 2019)
- **slb**: SeaLifeBase (unavailable)
- **wd**: Wikidata (unavailable)
- **ott**: OpenTree Taxonomy (v. 2021)
- **iucn**: International Union for Conservation of Nature (v. 2019)

The bdc_query_names_taxadb processes as this:

### Creation of a local taxonomic database

This is a one-time setup used to download, extract, and import the taxonomic databases specified in the argument "db". The downloading process may take a few minutes depending on your connection and database size. By default, the "gbif" database following a Darwin Core schema is installed. (see ?taxadb::td_create for details).

### Taxonomic harmonization

The taxonomic harmonization is divided into two distinct phases according to the matching type to be undertaken.

**Exact matching**

Firstly, the algorithm attempts to find an exact matching for each original scientific name supplied using the function "filter_name" from taxadb package. If an exact matching cannot be found, names are returned as Not Available (NA). Also, it is possible that a scientific name match multiple accepted names. In such cases, the "bdc_clean_duplicates" function is used to flag and remove names with multiple accepted names.

Information on higher taxa (e.g., kingdom or phylum) can be used to disambiguate names linked to multiple accepted names. For example, the genus "Casearia" is present in both Animalia and Plantae kingdoms. When handling names of Plantae, it would be helpful to get rid of names belonging to the Animalia to avoid flagging "Caseria" as having multiple accepted names. Following Norman et al. (2020), such cases are left to be fixed by the user. If "export_accepted" = TRUE a database containing a list of all records with names linked to multiple accepted names is saved in the "Output" folder.

**Fuzzy matching**

Fuzzy matching will be applied when "suggest_names" is TRUE and only for names not resolved by an exact match. In such cases, a fuzzy matching algorithm processes name-matching queries to find a potential matching candidate from the specified taxonomic database. Fuzzy matching identifies probable names (here identified as suggested names) for original names via a measure of orthographic similarity (i.e., distance). Orthographic distance is calculated by optimal string alignment (restricted Damerau-Levenshtein distance) that counts the number of deletions, insertions, substitutions, and adjacent characters' transpositions. It ranges from 0 to 1, being 1 an indicative of a perfect match. A threshold distance, i.e. the lower value of match acceptable, can be informed by user (in the "suggest_distance" argument). If the distance of a candidate name is equal or higher than the distance informed by user, the candidate name is returned as suggested name. Otherwise, names are returned as NA.

To increase the probability of finding a potential match candidate and to save time, two steps are taken before conducting fuzzy matching. First, if supplied, information on higher taxon (e.g., kingdom, family) is used to filter the taxonomic database. This step removes matching ambiguity by avoiding matching names from unrelated taxonomic ranks (e.g., match a plant species against a taxonomic database containing animal names) and decreases the number of names in the taxonomic database used to calculate the matching distance. Then, the taxonomic database is filtered according to a set of firsts letters of all input names. This process reduces the number of names in the taxonomic database to which each original name should be compared When a same suggested name is returned for different input names, a warning is returned asking users to check whether the suggested name is valid.

**Report**

The name harmonization processes' quality can be accessed in the column "notes" placed in the table resulting from the name harmonization process. The column "notes" contains assertions on the name harmonization process based on Carvalho (2017). The notes can be grouped in two categories: accepted names and those with a taxonomic issue or warning, needing further inspections. Accepted names can be returned as "accepted" (valid accepted name), "replaceSynonym" (a synonym replaced by an accepted name), "wasMisspelled" (original name was misspelled), "wasMisspelled | replaceSynonym" (misspelled synonym replaced by an accepted name), and "synonym" (original names is a synonym without accepted names in the database). Similarly, the following notes are

used to flag taxonomic issues: "notFound" (no matching name found), "multipleAccepted" (name with multiple accepted names), "noAcceptedName" (no accepted name found), and ambiguous synonyms such as "heterotypic synonym", "homotypic synonym", and "pro-parte synonym". Ambiguous synonyms, names that have been published more than once describing different species, have more than one accepted name and cannot be resolved. Such cases are flagged and left to be determined by the user.

## Value

This function returns data.frame containing the results of the taxonomic harmonization process. The database is returned in the same order of sci_name.

## See Also

Other taxonomy: `bdc_clean_names()`, `bdc_filter_out_names()`

## Examples

```
if (interactive()) {
  sci_name <- c(
    "Polystachya estrellensis",
    "Tachigali rubiginosa",
    "Oxalis rhombeo ovata",
    "Axonopus canescens",
    "Prosopis",
    "Haematococcus salinus",
    "Monas pulvisculus",
    "Cryptomonas lenticulari",
    "Poincianella pyramidalis",
    "Hymenophyllum polyanthos"
  )

  names_harmonization <-
    bdc_query_names_taxadb(
      sci_name,
      replace_synonyms = TRUE,
      suggest_names = TRUE,
      suggestion_distance = 0.9,
      db = "gbif",
      parallel = TRUE,
      ncores = 2,
      export_accepted = FALSE
    )
}
```

bdc_quickmap *Create a map of points using ggplot2*

## Description

Creates a map of points using ggplot2 useful for inspecting the results of tests implemented in the bdc package.

## Usage

```
bdc_quickmap(
  data,
  lat = "decimalLatitude",
  lon = "decimalLongitude",
  col_to_map = "red",
  size = 1
)
```

## Arguments

| | |
|---|---|
| data | data.frame. Containing geographical coordinates. Coordinates must be expressed in decimal degree and in WGS84. |
| lat | character string. The column name with latitude. Coordinates must be expressed in decimal degree and in WGS84. Default = "decimalLatitude". |
| lon | character string. The column with longitude. Coordinates must be expressed in decimal degree and in WGS84. Default = "decimalLongitude". |
| col_to_map | character string. Defining the column or color used to map. It can be a color name (e.g., "red") or the name of a column of data. Default = "blue" |
| size | numeric. The size of the points. |

## Details

Only records with valid coordinates can be plotted. Records missing or containing invalid coordinates are removed prior creating the map.

## Value

A map of points created using ggplot2.

## Examples

```
decimalLatitude <- c(19.9358, -13.016667, -19.935800)
decimalLongitude <- c(-40.6003, -39.6, -40.60030)
.coordinates_out_country <- c(FALSE, TRUE, TRUE)
x <- data.frame(decimalLatitude, decimalLongitude, .coordinates_out_country)

bdc_quickmap(
```

```
  data = x,
  lat = "decimalLatitude",
  lon = "decimalLongitude",
  col_to_map = ".coordinates_out_country",
  size = 1
)
```

---

`bdc_scientificName_empty` *Identify records with empty scientific names*

---

## Description

Flags records with empty or not interpretable scientific names.

## Usage

```
bdc_scientificName_empty(data, sci_names = "scientificName")
```

## Arguments

| | |
|---|---|
| `data` | data.frame. Containing the species scientific names. |
| `sci_names` | character string. The column name with the species scientific name. Default = "scientificName". |

## Details

This test identifies records missing scientific names (i.e., empty or not applicable NA names)

## Value

A data.frame containing the column ".scientificName_empty". Compliant (TRUE) if 'sci_names' is not empty; otherwise "FALSE".

## See Also

Other prefilter: `bdc_basisOfRecords_notStandard()`, `bdc_coordinates_country_inconsistent()`, `bdc_coordinates_empty()`, `bdc_coordinates_from_locality()`, `bdc_coordinates_outOfRange()`, `bdc_coordinates_transposed()`, `bdc_country_standardized()`

## Examples

```
x <- data.frame(scientificName = c("Ocotea odorifera", NA, "Panthera onca", ""))
bdc_scientificName_empty(data = x, sci_names = "scientificName")
```

bdc_standardize_datasets

*Standardize datasets columns based on metadata*

## Description

This function's main goal is to merge and standardize different datasets into a new dataset with column names following the Darwin Core terminology. All the process is based on a metadata file provided by the user.

## Usage

```
bdc_standardize_datasets(
  metadata,
  format = "csv",
  overwrite = FALSE,
  save_database = FALSE
)
```

## Arguments

| | |
|---|---|
| metadata | A data frame with metadata containing information about the name, path, and columns of the original data set which need to be renamed. See @details. |
| format | a character setting the output file type. Option available are "csv" and "qs" (recommenced to save large datasets). Default == "csv". |
| overwrite | A logical vector indicating whether the final merged dataset should be overwritten. The default is FALSE. |
| save_database | logical. Should the standardized database be locally saved? Default = FALSE. |

## Details

`bdc_standardize_datasets()` facilitate the standardization of datasets with different column names by converting them into a new dataset following the Darwin Core terminology. The standardization process relies on a metadata file containing the name, path, and columns that need to be renamed. The metadata file can be constructed using built-in functions (e.g., `data.frame()`) or storing the information in a CSV file and importing it into R. Regardless of the method chosen, the data frame with metadata needs to contain the following column names (this is a list of required column names; for a comprehensive list of column names following Darwin Core terminology, see here

- `datasetName`: A short name identifying the dataset (e.g., GBIF)
- `fileName`: The relative path containing the name of the input dataset (e.g., Input_files/GBIF.csv)
- `scientificName`: Name of the column in the original database presenting the taxon scientific names with or without authorship information, depending on the format of the source dataset (e.g., Myrcia acuminata)

- `decimalLatitude`: Name of the column in the original database presenting the geographic latitude in decimal degrees (e.g., -6.370833)
- `decimalLongitude`: Name of the column in the original database presenting the geographic longitude in decimal degrees (e.g., -3.25500)

## Value

A merged data.frame with column names following Darwin Core terminology.

## Examples

```
## Not run:
metadata <- readr::read_csv(system.file("extdata/Config/DatabaseInfo.csv",
            package = "bdc"))

db_standardized <-
bdc_standardize_datasets(
  metadata = metadata,
  format = "csv",
  overwrite = TRUE,
  save_database = FALSE)

## End(Not run)
```

---

# bdc_summary_col *Create or update the column summarizing the results of data quality tests*

---

## Description

This function creates or updates the column ".summary" summarizing the results of data quality tests (i.e., columns starting with "."). Records that have failed in at least one test are flagged for further inspection (i.e., flagged as "FALSE") in the ".summary" column.

## Usage

```
bdc_summary_col(data)
```

## Arguments

| | |
|---|---|
| `data` | data.frame. Containing the results of data quality tests (i.e., columns starting with "."). |

## Details

If existing, the column ".summary" will be removed and then updated considering all test names available in the supplied database.

## Value

A data.frame containing a new or an updated column ".summary".

## Examples

```
.missing_names <- c(TRUE, TRUE, TRUE, FALSE, FALSE)
.missing_coordinates <- c(TRUE, FALSE, FALSE, TRUE, FALSE)
x <- data.frame(.missing_names, .missing_coordinates)

bdc_summary_col(data = x)
```

---

# bdc_year_from_eventDate

*Extract year from eventDate*

---

## Description

This function extracts a four-digit year from unambiguously interpretable collecting dates.

## Usage

```
bdc_year_from_eventDate(data, eventDate = "eventDate")
```

## Arguments

| | |
|---|---|
| data | A data frame containing a column with event date information. |
| eventDate | Numeric or date. The column with event date information. |

## Value

A data.frame containing the column "year". Year information is returned only if "eventDate" can be unambiguously interpretable from "eventDate". Years in the future (e.g., 2050) are returned as NA as well as years before 1600, which is the lower limit for collecting dates of biological specimens.

## See Also

Other time: bdc_eventDate_empty(), bdc_year_outOfRange()

## Examples

```
collection_date <- c(
  NA, "31/12/2015", "2013-06-13T00:00:00Z", "2019-05-20",
  "", "2013", "0001-01-00", "20", "1200"
)
x <- data.frame(collection_date)

bdc_year_from_eventDate(data = x, eventDate = "collection_date")
```

bdc_year_outOfRange *Identify records with year out-of-range*

## Description

This function identifies records out-of-range collecting year (e.g., in the future) or old records collected before a year informed in 'year_threshold'.

## Usage

```
bdc_year_outOfRange(data, eventDate, year_threshold = 1900)
```

## Arguments

| | |
|---|---|
| data | A data frame containing a column with event date information. |
| eventDate | numeric or date. The column containing event date information. |
| year_threshold | numeric. A four-digit year threshold used to flag old (potentially invalid) records. Default = 1900 |

## Details

Following the "VALIDATION:YEAR_OUTOFRANGE" Biodiversity data quality group, the results of this test are time-dependent. While the user may provide a lower limit to the year, the upper limit is defined based on the year when the test is run. Lower limits can be used to flag old, often imprecise, records. For example, records collected before GPS advent (1980). If 'year_threshold' is not provided, the lower limit to the year is by default 1600, a lower limit for collecting dates of biological specimens. Records with empty or NA 'eventDate' are not tested and returned as NA.

## Value

A data.frame containing the column ".year_outOfRange". Compliant (TRUE) if 'eventDate' is not out-of-range; otherwise "FALSE".

## See Also

Other time: bdc_eventDate_empty(), bdc_year_from_eventDate()

## Examples

```
collection_date <- c(
  NA, "31/12/2029", "2013-06-13T00:00:00Z", "2013-06-20",
  "", "2013", 1650, "0001-01-00"
)
x <- data.frame(collection_date)

bdc_year_outOfRange(
data = x,
eventDate = "collection_date",
```

```
    year_threshold = 1900)
```

# Index