Package 'BeeBDC'

November 6, 2026

Title Occurrence Data Cleaning

Version 1.3.6

Description Flags and checks occurrence data that are in Darwin Core format. The package includes generic functions and data as well as some that are specific to bees. This package is meant to build upon and be complimentary to other excellent occurrence cleaning packages, including 'bdc' and 'CoordinateCleaner'. This package uses datasets from several sources and particularly from the Discover Life Website, created by Ascher and Pickering (2020). For further information, please see the original publication and package website. Publication - Dorey et al. (2023) <doi:10.1101/2023.06.30.547152> and package website - Dorey et al. (2023) <https://github.com/jbdorey/BeeBDC>.

License GPL (>= 3)

URL https://jbdorey.github.io/BeeBDC/
https://github.com/jbdorey/BeeBDC

BugReports https://github.com/jbdorey/BeeBDC/issues

Depends R (>= 4.1.0)

Imports circlize, CoordinateCleaner, cowplot, dplyr, forcats, ggplot2, ggspatial, here, igraph, lubridate, mgsub, openxlsx, paletteer, readr, rnaturalearth, sf, stringr, tidyselect

Suggests bdc, BiocManager, classInt, ComplexHeatmap, countrycode, curl, devtools, emld, formatR, galah, hexbin, htmltools, htmlwidgets, httr, iNEXT, janitor, kableExtra, knitr, leaflet, magrittr, mosaic, pkgdown, plotly, prettydoc, purrr, renv, rgnparser, rlang, rmarkdown, rmdformats, rnaturalearthdata, rvest, SpadeR, taxadb, terra, testthat, tibble, tidyr, withr, xml2

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

Encoding UTF-8

Language en-gb

LazyData TRUE

LazyDataCompression xz

NeedsCompilation no

Author James B. Dorey [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0003-2721-3842>),
Robert L. O'Reilly [aut] (ORCID:
<https://orcid.org/0000-0001-5291-7396>),
Silas Bossert [aut] (ORCID: <https://orcid.org/0000-0002-3620-5468>),
Erica E. Fischer [aut] (ORCID: <https://orcid.org/0000-0002-8202-158X>)

Maintainer James B. Dorey <jbdorey@me.com>

Repository CRAN

Date/Publication 2026-11-06 13:20:02 UTC

Contents

atlasDownloader 3
BeeBDCQuery 4
bees3sp 6
beesChecklist 12
beesCountrySubset 14
beesFlagged 15
beesRaw 21
beesTaxonomy 25
ChaoWrapper 27
chordDiagramR 29
ColTypeR 32
continentOutlieRs 33
coordUncerFlagR 34
countryHarmoniseR 35
countryNameCleanR 36
countryOutlieRs 37
dataProvTables 38
dataSaver 39
dateFindR 40
diagonAlley 41
dirMaker 43
dupePlotR 45
dupeSummary 46
fileFinder 49
flagAbsent 50
flagLicense 51
flagRecorder 52
flagSummaryTable 53
formattedCombiner 54
GBIFissues 55
ggRichnessWrapper 56

harmoniseR 59
idMatchR 61
importOccurrences 62
iNEXTwrapper 63
interactiveMapR 65
jbd_CfC_chunker 67
jbd_coordCountryInconsistent 69
jbd_coordinates_precision 71
jbd_coordinates_transposed 72
jbd_create_figures 74
jbd_Ctrans_chunker 76
manualOutlierFindeR 78
PaigeIntegrater 79
plotFlagSummary 81
readr_BeeBDC 84
repoFinder 87
repoMerge 88
richnessEstimateR 89
richnessPrepR 91
summaryFun 93
summaryMaps 94
taxadbToBeeBDC 96
testChecklist 98
testTaxonomy 99
USGS_formatter 100

Index **102**

`atlasDownloader` *Download occurrence data from the Atlas of Living Australia (ALA)*

Description

Downloads ALA data and creates a new file in the path to put those data. This function can also request downloads from other atlases (see: http://galah.ala.org.au/articles/choosing_an_atlas.html). However, it will only send the download to your email and you must do the rest yourself at this point.

Usage

```
atlasDownloader(
  path,
  userEmail = NULL,
  ALA_taxon,
  DL_reason = 4,
  atlas = "ALA"
)
```

Arguments

path	A character directory. The path to a folder where the download will be stored.
userEmail	A character string. The email used associated with the user's ALA account; user must make an ALA account to download data.
ALA_taxon	A character string. The taxon to download from ALA. Uses galah::galah_identify()
DL_reason	Numeric. The reason for data download according to galah::galah_config()
atlas	Character. The atlas to download occurrence data from - see here https://galah.ala.org.au/R/articles/choosi for details. Note: the default is "ALA" and is probably the only atlas which will work seamlessly with the rest of the workflow. However, different atlases can still be downloaded and a doi will be sent to your email.

Value

Completes an ALA data download and saves those data to the path provided.

Examples

```
## Not run:
atlasDownloader(path = DataPath,
                userEmail = "InsertYourEmail",
                ALA_taxon = "Apiformes",
                DL_reason = 4)

## End(Not run)
```

BeeBDCQuery *Query the bee taxonomy and country checklist*

Description

A simple function to return information about a particular species, including name validity and country occurrences.

Usage

```
BeeBDCQuery(
  beeName = NULL,
  searchChecklist = TRUE,
  printAllSynonyms = FALSE,
  beesChecklist = NULL,
  beesTaxonomy = NULL
)
```

Arguments

beeName	Character or character vector. A single or several bee species names to search for in the beesTaxonomy and beesChecklist tables.
searchChecklist	Logical. If TRUE (default), search the country checklist for each species.
printAllSynonyms	Logical. If TRUE, all synonyms will be printed out for each entered name. default = FALSE.
beesChecklist	A tibble. The bee checklist file for BeeBDC. If is NULL then beesChecklist() will be called internally to download the file. Default = NULL.
beesTaxonomy	A tibble. The bee taxonomy file for BeeBDC. If is NULL then beesTaxonomy() will be called internally to download the file. Default = NULL.

Value

Returns a list with the elements 'taxonomyReport' and 'SynonymReport'. IF searchChecklist is TRUE, then 'checklistReport' will also be returned.

Examples

```
  # For the sake of these examples, we will use the example taxonomy and checklist
  system.file("extdata", "testTaxonomy.rda", package="BeeBDC") |> load()
  system.file("extdata", "testChecklist.rda", package="BeeBDC") |> load()

  # Single entry example
testQuery <- BeeBDCQuery(
  beeName = "Lasioglossum bicingulatum",
  searchChecklist = TRUE,
  printAllSynonyms = TRUE,
  beesTaxonomy = testTaxonomy,
  beesChecklist = testChecklist)

  # Multiple entry example
testQuery <- BeeBDC::BeeBDCQuery(
  beeName = c("Lasioglossum bicingulatum", "Nomada flavopicta",
  "Lasioglossum fijiense (Perkins and Cheesman, 1928)"),
  searchChecklist = TRUE,
  printAllSynonyms = TRUE,
  beesTaxonomy = testTaxonomy,
  beesChecklist = testChecklist)

    # Example way to examine a report from the output list
  testQuery$checklistReport
```

bees3sp *A flagged dataset of 105 random bee occurrence records from the three species*

Description

This test dataset includes 105 random occurrence records from three bee species. The included species are: "Agapostemon tyleri Cockerell, 1917", "Centris rhodopus Cockerell, 1897", and "Perdita octomaculata (Say, 1824)".

Usage

```
data("bees3sp", package = "BeeBDC")
```

Format

An object of class `"tibble"`

database_id Occurrence code generated in bdc or BeeBDC

scientificName Full scientificName as shown on DiscoverLife

family Family name

subfamily Subfamily name

genus Genus name

subgenus Subgenus name

subspecies Full scientific name with subspecies name - ALA column

specificEpithet The species name (specific epithet) only

infraspecificEpithet The subspecies name (intraspecific epithet) only

acceptedNameUsage The full scientific name, with authorship and date information if known, of the currently valid (zoological) or accepted (botanical) taxon.

taxonRank The taxonomic rank of the most specific name in the scientificName column.

scientificNameAuthorship The authorship information for the scientificName column formatted according to the conventions of the applicable nomenclaturalCode.

identificationQualifier A brief phrase or a standard term ("cf.", "aff.") to express the determiner's doubts about the identification.

higherClassification A list (concatenated and separated) of taxon names terminating at the rank immediately superior to the taxon referenced in the taxon record.

identificationReferences A list (concatenated and separated) of references (e.g. publications, global unique identifier, URI, etc.) used in the identification of the occurrence.

typeStatus A list (concatenated and separated) of nomenclatural types (e.g. type status, typified scientific name, publication) applied to the occurrence.

previousIdentifications A list (concatenated and separated) of previous assignments of names to the occurrence.

verbatimIdentification This term is meant to allow the capture of an unaltered original identification/determination, including identification qualifiers, hybrid formulas, uncertainties, etc. This term is meant to be used in addition to scientificName (and identificationQualifier etc.), not instead of it.

identifiedBy A list (concatenated and separated) of names of people, groups, or organizations who assigned the Taxon to the subject.

dateIdentified The date on which the occurrence was identified as belonging to a taxon.

decimalLatitude The geographic latitude (in decimal degrees, using the spatial reference system given in geodeticDatum) of the geographic center of a location. Positive values are north of the Equator, negative values are south of it, and valid values lie between -90 and 90, inclusive.

decimalLongitude The geographic longitude (in decimal degrees, using the spatial reference system given in geodeticDatum) of the geographic center of a location. Positive values are east of the Greenwich Meridian, and negative values are west of it. Valid values lie between -180 and 180, inclusive.

stateProvince The name of the next smaller administrative region than country (e.g. state, province, canton, department, region, etc.) in which the location for the occurrence is found.

continent The name of the continent in which the location for the occurrence is found.

locality A specific description of the place the occurrence was found.

island The name of the island on or near which the location for the occurrence is found, if applicable.

county The full, unabbreviated name of the next smaller administrative region than stateProvince (e.g. county, shire, department, etc.) in which the location for the occurrence is found.

municipality The full, unabbreviated name of the next smaller administrative region than county (e.g. city, municipality, etc.) in which the location for the occurrence is found. Do not use this term for a nearby named place that does not contain the actual location for the occurrence.

license A legal document giving official permission to do something with the resource.

issue A GBIF-defined issue.

eventDate The time or interval during which the Event occurred. For occurrences, this is the time or interval when the event was recorded.

eventTime The time or interval during which an Event occurred.

day The integer day of the month on which the Event occurred. For occurrences, this is the day when the event was recorded.

month The integer month in which the Event occurred. For occurrences, this is the month of when the event was recorded.

year The four-digit year in which the Event occurred, according to the Common Era Calendar. For occurrences, this is the year when the event was recorded.

basisOfRecord The specific nature of the data record. Recommended best practice is to use the standard label of one of the Darwin Core classes.PreservedSpecimen, FossilSpecimen, LivingSpecimen, MaterialSample, Event, HumanObservation, MachineObservation, Taxon, Occurrence, MaterialCitation

country The name of the country or major administrative unit in which the location for the occurrence is found.

type The nature or genre of the resource. StillImage, MovingImage, Sound, PhysicalObject, Event, Text.

occurrenceStatus A statement about the presence or absence of a Taxon at a Location. present, absent.

recordNumber An identifier given to the Occurrence at the time it was recorded. Often serves as a link between field notes and an Occurrence record, such as a specimen collector's number.

recordedBy A list (concatenated and separated) of names of people, groups, or organizations responsible for recording the original Occurrence. The primary collector or observer, especially one who applies a personal identifier (recordNumber), should be listed first.

eventID An identifier for the set of information associated with an Event (something that occurs at a place and time). May be a global unique identifier or an identifier specific to the data set.

Location A spatial region or named place.

samplingProtocol The names of, references to, or descriptions of the methods or protocols used during an Event. Examples UV light trap, mist net, bottom trawl, ad hoc observation | point count, Penguins from space: faecal stains reveal the location of emperor penguin colonies, https://doi.org/10.1111/j.1466-8238.2009.00467.x, Takats et al. 2001.

samplingEffort The amount of effort expended during an Event. Examples 40 trap-nights, 10 observer-hours, 10 km by foot, 30 km by car.

individualCount The number of individuals present at the time of the Occurrence. Integer.

organismQuantity A number or enumeration value for the quantity of organisms. Examples 27 (organismQuantity) with individuals (organismQuantityType). 12.5 (organismQuantity) with percentage biomass (organismQuantityType). r (organismQuantity) with Braun Blanquet Scale (organismQuantityType). many (organismQuantity) with individuals (organismQuantityType).

coordinatePrecision A decimal representation of the precision of the coordinates given in the decimalLatitude and decimalLongitude.

coordinateUncertaintyInMeters The horizontal distance (in meters) from the given decimalLatitude and decimalLongitude describing the smallest circle containing the whole of the Location. Leave the value empty if the uncertainty is unknown, cannot be estimated, or is not applicable (because there are no coordinates). Zero is not a valid value for this term.

spatiallyValid Occurrence records in the ALA can be filtered by using the spatially valid flag. This flag combines a set of tests applied to the record to see how reliable are its spatial data components.

catalogNumber An identifier (preferably unique) for the record within the data set or collection.

gbifID The identifier assigned by GBIF for each record.

datasetID An identifier for the set of data. May be a global unique identifier or an identifier specific to a collection or institution.

institutionCode The name (or acronym) in use by the institution having custody of the object(s) or information referred to in the record. Examples MVZ, FMNH, CLO, UCMP.

datasetName The name identifying the data set from which the record was derived.

otherCatalogNumbers A list (concatenated and separated) of previous or alternate fully qualified catalog numbers or other human-used identifiers for the same Occurrence, whether in the current or any other data set or collection.

occurrenceID An identifier for the Occurrence (as opposed to a particular digital record of the occurrence). In the absence of a persistent global unique identifier, construct one from a combination of identifiers in the record that will most closely make the occurrenceID globally unique.

taxonKey The GBIF-assigned taxon identifier number.

collectionID An identifier for the collection or dataset from which the record was derived.

verbatimScientificName Scientific name as recorded on specimen label, not necessarily valid.

verbatimEventDate The verbatim original representation of the date and time information for an event. For occurrences, this is the date-time when the event was recorded as noted by the collector.

associatedTaxa A list (concatenated and separated) of identifiers or names of taxa and the associations of this occurrence to each of them.

associatedOrganisms A list (concatenated and separated) of identifiers of other Organisms and the associations of this occurrence to each of them.

fieldNotes One of (a) an indicator of the existence of, (b) a reference to (publication, URI), or (c) the text of notes taken in the field about the Event.

sex The sex of the biological individual(s) represented in the Occurrence.

rights A description of the usage rights applicable to the record.

rightsHolder A person or organization owning or managing rights over the resource.

accessRights Information about who can access the resource or an indication of its security status.

associatedReferences A list (concatenated and separated) of identifiers (publication, bibliographic reference, global unique identifier, URI) of literature associated with the Occurrence.

bibliographicCitation A bibliographic reference for the resource as a statement indicating how this record should be cited (attributed) when used.

references A related resource that is referenced, cited, or otherwise pointed to by the described resource.

informationWithheld Additional information that exists, but that has not been shared in the given record.

isDuplicateOf The code for another occerrence but for the same specimen.

hasCoordinate Variable indicating presence/absence of location coordinates.

hasGeospatialIssues Variable indicating validity of geospatial data associated with record.

occurrenceYear Year associated with Occurrence.

id Variable with identifying value for the Occurrenc.

duplicateStatus Variable indicating is Occurrence is duplicate or not.

associatedOccurrences A list (concatenated and separated) of identifiers of other occurrence records and their associations to this occurrence.

locationRemarks Comments or notes about the Location.

dataSource BeeBDC assigned source of the data. Often written when the data is formatted by a BeeBDC::xxx_readr function or similar.

verbatim_scientificName The verbatim (originally-provided) scientific name

.scientificName_empty Flag produced by bdc::bdc_scientificName_empty() where FALSE == no scientific name provided and TRUE means that there is text in that column.

.coordinates_empty Flag produced by bdc::bdc_coordinates_empty() where FALSE == no coordinates provided.

.coordinates_outOfRange Flag column produced by bdc::bdc_coordinates_outOfRange() where FALSE == coordinates represent a point off of the Earth. This is to say, the function identifies records with out-of-range coordinates (not between -90 and 90 for latitude; not between -180 and 180 for longitude).

.basisOfRecords_notStandard Flag produced by bdc::bdc_basisOfRecords_notStandard() where FALSE == an occurrence with a basisOfRecord not defined as acceptable by the user.

country_suggested A country name suggested by the bdc::bdc_country_standardized() function.

countryCode A country code suggested by the bdc::bdc_country_standardized() function.

coordinates_transposed A column indicating if coordinates were identified as being transposed by the function jbd_Ctrans_chunker() where FALSE == transposed.

.coordinates_country_inconsistent A flag generated by jbd_coordCountryInconsistent() where FALSE == an occurrence where the country name and coordinates did not match.

.occurrenceAbsent A flag generated by flagAbsent() where FALSE == occurrences marked as "ABSENT" in the "occurrenceStatus" column

.unLicensed A flag generated by flagLicense() where FALSE == those occurrences protected by a restrictive license.

.GBIFflags A flag generated by GBIFissues() where FALSE == an occurrence with user-specified GBIF issues to flag.

.uncer_terms A flag generated by bdc::bdc_clean_names() where FALSE == the presence of taxonomic uncertainty terms.

names_clean A column made by bdc::bdc_clean_names() indicating the cleaned scientificName

.invalidName A flag generated by harmoniseR() where FALSE == occurrences whose scientificName did not match the Discover Life taxonomy.

.rou A flag generated by CoordinateCleaner::clean_coordinates() where FALSE == rounded (probably imprecise) coordinates.

.val A flag generated by CoordinateCleaner::clean_coordinates() where FALSE == invalid coordinates.

.equ A flag generated by CoordinateCleaner::clean_coordinates() where FALSE == equal coordinates (e.g., 0.1, 0.1).

.zer A flag generated by CoordinateCleaner::clean_coordinates() where FALSE == zeros as coordinates

.cap A flag generated by CoordinateCleaner::clean_coordinates() where FALSE == records around country capital centroid.

.cen A flag generated by CoordinateCleaner::clean_coordinates() where FALSE == records around country or province centroids.

.gbf A flag generated by CoordinateCleaner::clean_coordinates() where FALSE == records around the GBIF headquarters.

.inst A flag generated by `CoordinateCleaner::clean_coordinates()` where FALSE == records around biodiversity institutions.

.sequential A flag generated by `diagonAlley()` where FALSE == records that are possibly the result of fill-down errors in sequence.

.lonFlag A flag generated by `CoordinateCleaner::cd_round()` where FALSE == potential gridding in the longitude column within dataset.

.latFlag A flag generated by `CoordinateCleaner::cd_round()` where FALSE == potential gridding in the latitude column within dataset.

.gridSummary A flag generated by `CoordinateCleaner::cd_round()` where FALSE == potential gridding in either the longitude or latitude columns within dataset.

.uncertaintyThreshold A flag generated by `coordUncerFlagR()` where FALSE == occurrences that did not pass a user-specified threshold in the "coordinateUncertaintyInMeters" column.

countryMatch A column made by `countryOutlieRs()`. Summarises the occurrence-level result: where the species is not known to occur in that country (noMatch), it is known from a bordering country (neighbour), or it is known to occur in that country (exact).

.countryOutlier A flag generated by `countryOutlieRs()` where FALSE == occurrences the do not occur in a country that concurs with the Discover Life country checklist OR an adjacent country.

.sea A flag generated by `countryOutlieRs()` where FALSE == occurrences that are in the ocean.

.summary A flag generated by `summaryFun()` where FALSE == occurrences flagged as FALSE in any of the .flag columns. In this example it excludes flags in the ".gridSummary", ".lonFlag", ".latFlag", and ".uncer_terms" columns.

.eventDate_empty A flag generated by `bdc::bdc_eventDate_empty()` where FALSE == occurrences with no eventDate provided.

.year_outOfRange A flag column generated by `bdc::bdc_year_outOfRange()` where FALSE == occurrences older than a threshold date. In the case of the bee dataset used in this package, the lower threshold is 1950

.duplicates A flag generated by `dupeSummary()` where FALSE == occurrences identified as duplicates. There will be an associated kept duplicate (.duplictes == TRUE) for all duplicate clusters.

Details

A small bee occurrence dataset with flags generated by BeeBDC which can be used to run the example script and to test functions. For data types, see `ColTypeR()`.

References

This data set was created by generating a random subset of 105 rows from the full BeeBDC dataset from the publication: Dorey, J.B., Fischer, E.E., Chesshire, P.R., Nava-Bolaños, A., O'Reilly, R.L., Bossert, S., Collins, S.M., Lichtenberg, E.M., Tucker, E., Smith-Pardo, A., Falcon-Brindis, A., Guevara, D.A., Ribeiro, B.R., de Pedro, D., Hung, J.K.-L., Parys, K.A., McCabe, L.M., Rogan, M.S., Minckley, R.L., Velzco, S.J.E., Griswold, T., Zarrillo, T.A., Jetz, W., Sica, Y.V., Orr, M.C., Guzman, L.M., Ascher, J., Hughes, A.C. & Cobb, N.S. (2023) A globally synthesised and flagged bee occurrence dataset and cleaning workflow. Scientific Data, 10, 1–17. https://www.doi.org/10.1038/S41597-023-02626-W

Examples

```
bees3sp <- BeeBDC::bees3sp
head(bees3sp)
```

`beesChecklist` *Download a country-level checklist of bees from Discover Life*

Description

Download the table contains taxonomic and country information for the bees of the world based on data collated on Discover Life. The data will be sourced from the BeeBDC article's Figshare.

Note that sometimes the download might not work without restarting R. In this case, you could alternatively download the dataset from the URL below and then read it in using `base::readRDS("filePath.Rda")`. Note that as of version 1.3.2, this function internally uses a modified version of the "download" function from the `downloader` package on CRAN. Additionally, BeeBDC will try a different download method with each failed attempt (for Windows: "auto" > "internal" > "libcurl" > "wget" > "curl"; or "auto" if any particular method is not available).

See `beesTaxonomy()` for further context.

Usage

```
beesChecklist(
  URL = "https://api.figshare.com/v2/file/download/60945823",
  mode = NULL,
  headers = NULL,
  token = NULL,
  alternateURL = FALSE,
  ...
)
```

Arguments

`URL`	A character vector to the FigShare location of the dataset. The default will be to the most-recent version.
`mode`	A character passed on to `utils::download.file()`. Default = "wb" for Windows or "w" for Mac/Linux.
`headers`	Character. Passed on to `utils::download.file()`. Default = NULL.
`token`	Character. Optional personal access token from FigShare account for authorisation. Default = NULL.
`alternateURL`	Logical. If TRUE then the function will use the an alernate version of the download URL. Might be worth trying if it is failing. Default = FALSE.
`...`	Extra variables that can be passed to `downloader::download()`.

Value

A downloaded beesChecklist.Rda file in the outPath and the same tibble returned to the environment.

**Column details **

validName The valid scientificName as it should occur in the scientificName column.

DiscoverLife_name The full country name as it occurs on Discover Life.

rNaturalEarth_name Country name from rnaturalearth's name_long.

shortName A short version of the country name.

DiscoverLife_ISO The ISO country name as it occurs on Discover Life.

Alpha-2 Alpha-2 from rnaturalearth.

Alpha-3 Alpha-3 from rnaturalearth.

official Official country name = "yes" or only a Discover Life name = "no".

Source A text strign denoting the source or author of the name-country pair.

matchCertainty Quality of the name's match to the Discover Life checklist.

canonical The valid species name without scientificNameAuthority.

canonical_withFlags The validName without the scientificNameAuthority but with Discover Life flags.

family Bee family.

subfamily Bee subfamily.

genus Bee genus.

subgenus Bee subgenus.

infraspecies Bee infraSpecificEpithet.

species Bee specificEpithet.

scientificNameAuthorship Bee scientificNameAuthorship.

taxon_rank Rank of the taxon name.

Notes Discover Life country name notes.

Previous checklists:

- 2026-01-12 **current**: https://open.flinders.edu.au/ndownloader/files/60945823
- 2024-06-17: https://open.flinders.edu.au/ndownloader/files/47092720
- Original: https://open.flinders.edu.au/ndownloader/files/42320598

References

This dataset was created using the Discover Life checklist and taxonomy. Dataset is from the publication: DOREY, J. B., CHESSHIRE, P. R., BOLAÑOS, A. N., O'REILLY, R. L., BOSSERT, S., COLLINS, S. M., LICHTENBERG, E. M., TUCKER, E., SMITH-PARDO, A., FALCON-BRINDIS, A., GUEVARA, D. A., RIBEIRO, B. R., DE PEDRO, D., FISCHER, E., HUNG, J. K.-L., PARYS, K. A., ROGAN, M. S., MINCKLEY, R. L., VELZCO, S. J. E., GRISWOLD, T., ZARRILLO, T. A., SICA, Y., ORR, M. C., GUZMAN, L. M., ASCHER, J., HUGHES, A. C. &

COBB, N. S. In review. A globally synthesised and flagged bee occurrence dataset and cleaning workflow. Scientific Data. The checklist data are mostly compiled from Discover Life data, www.discoverlife.org: ASCHER, J. S. & PICKERING, J. 2020. Discover Life bee species guide and world checklist (Hymenoptera: Apoidea: Anthophila). http://www.discoverlife.org/mp/20q?guide=Apoidea_species.

Examples

```
## Not run:
beesChecklist <- BeeBDC::beesChecklist()

## End(Not run)
```

`beesCountrySubset` *A simple scientificName and country_suggested test dataset*

Description

A small bee occurrence dataset with 1,488 scientificName-country_suggested combinations across four countries; Fiji, Uganda, Vietnam, and Zambia.

Usage

```
data("beesCountrySubset", package = "BeeBDC")
```

Format

An object of class `"tibble"`

scientificName Full scientificName as shown on DiscoverLife

country_suggested A country name suggested by the `bdc::bdc_country_standardized()` function.

References

This data set was created by during the writing of the paper: Dorey, J.B., (upcoming) How many bee species are there? A quantitative global estimate. Journal

Examples

```
beesCountrySubset <- BeeBDC::beesCountrySubset
head(beesCountrySubset)
```

`beesFlagged` *A flagged dataset of 100 random bee occurrence records*

Description

A small bee occurrence dataset with flags generated by BeeBDC used to run example script and test functions. For data types, see `ColTypeR()`.

Usage

```
data("beesFlagged", package = "BeeBDC")
```

Format

An object of class `"tibble"`

database_id Occurrence code generated in bdc or BeeBDC

scientificName Full scientificName as shown on DiscoverLife

family Family name

subfamily Subfamily name

genus Genus name

subgenus Subgenus name

subspecies Full name with subspecies name - ALA column

specificEpithet The species name only

infraspecificEpithet The subspecies name only

acceptedNameUsage The full name, with authorship and date information if known, of the currently valid (zoological) or accepted (botanical) taxon.

taxonRank The taxonomic rank of the most specific name in the scientificName.

scientificNameAuthorship The authorship information for the scientificName formatted according to the conventions of the applicable nomenclaturalCode.

identificationQualifier A brief phrase or a standard term ("cf.", "aff.") to express the determiner's doubts about the Identification.

higherClassification A list (concatenated and separated) of taxa names terminating at the rank immediately superior to the taxon referenced in the taxon record.)

identificationReferences A list (concatenated and separated) of references (publication, global unique identifier, URI) used in the Identification.

typeStatus A list (concatenated and separated) of nomenclatural types (type status, typified scientific name, publication) applied to the subject.

previousIdentifications A list (concatenated and separated) of previous assignments of names to the Organism.

verbatimIdentification This term is meant to allow the capture of an unaltered original identification/determination, including identification qualifiers, hybrid formulas, uncertainties, etc. This term is meant to be used in addition to scientificName (and identificationQualifier etc.), not instead of it.

identifiedBy A list (concatenated and separated) of names of people, groups, or organizations who assigned the Taxon to the subject.

dateIdentified The date on which the subject was determined as representing the Taxon.

decimalLatitude The geographic latitude (in decimal degrees, using the spatial reference system given in geodeticDatum) of the geographic center of a Location. Positive values are north of the Equator, negative values are south of it. Legal values lie between -90 and 90, inclusive.

decimalLongitude The geographic longitude (in decimal degrees, using the spatial reference system given in geodeticDatum) of the geographic center of a Location. Positive values are east of the Greenwich Meridian, negative values are west of it. Legal values lie between -180 and 180, inclusive.

stateProvince The name of the next smaller administrative region than country (state, province, canton, department, region, etc.) in which the Location occurs.

continent The name of the continent in which the Location occurs.

locality The specific description of the place.

island The name of the island on or near which the Location occurs.

county The full, unabbreviated name of the next smaller administrative region than stateProvince (county, shire, department, etc.) in which the Location occurs.

municipality The full, unabbreviated name of the next smaller administrative region than county (city, municipality, etc.) in which the Location occurs. Do not use this term for a nearby named place that does not contain the actual location.

license A legal document giving official permission to do something with the resource.

issue A GBIF-defined issue.

eventDate The date-time or interval during which an Event occurred. For occurrences, this is the date-time when the event was recorded. Not suitable for a time in a geological context.

eventTime The time or interval during which an Event occurred.

day The integer day of the month on which the Event occurred.

month The integer month in which the Event occurred.

year The four-digit year in which the Event occurred, according to the Common Era Calendar.

basisOfRecord The specific nature of the data record. Recommended best practice is to use the standard label of one of the Darwin Core classes.PreservedSpecimen, FossilSpecimen, LivingSpecimen, MaterialSample, Event, HumanObservation, MachineObservation, Taxon, Occurrence, MaterialCitation

country The name of the country or major administrative unit in which the Location occurs.

type The nature or genre of the resource. StillImage, MovingImage, Sound, PhysicalObject, Event, Text.

occurrenceStatus A statement about the presence or absence of a Taxon at a Location. present, absent.

recordNumber An identifier given to the Occurrence at the time it was recorded. Often serves as a link between field notes and an Occurrence record, such as a specimen collector's number.

recordedBy A list (concatenated and separated) of names of people, groups, or organizations responsible for recording the original Occurrence. The primary collector or observer, especially one who applies a personal identifier (recordNumber), should be listed first.

eventID An identifier for the set of information associated with an Event (something that occurs at a place and time). May be a global unique identifier or an identifier specific to the data set.

Location A spatial region or named place.

samplingProtocol The names of, references to, or descriptions of the methods or protocols used during an Event. Examples UV light trap, mist net, bottom trawl, ad hoc observation | point count, Penguins from space: faecal stains reveal the location of emperor penguin colonies, https://doi.org/10.1111/j.1466-8238.2009.00467.x, Takats et al. 2001.

samplingEffort The amount of effort expended during an Event. Examples 40 trap-nights, 10 observer-hours, 10 km by foot, 30 km by car.

individualCount The number of individuals present at the time of the Occurrence. Integer.

organismQuantity A number or enumeration value for the quantity of organisms. Examples 27 (organismQuantity) with individuals (organismQuantityType). 12.5 (organismQuantity) with percentage biomass (organismQuantityType). r (organismQuantity) with Braun Blanquet Scale (organismQuantityType). many (organismQuantity) with individuals (organismQuantityType).

coordinatePrecision A decimal representation of the precision of the coordinates given in the decimalLatitude and decimalLongitude.

coordinateUncertaintyInMeters The horizontal distance (in meters) from the given decimalLatitude and decimalLongitude describing the smallest circle containing the whole of the Location. Leave the value empty if the uncertainty is unknown, cannot be estimated, or is not applicable (because there are no coordinates). Zero is not a valid value for this term.

spatiallyValid Occurrence records in the ALA can be filtered by using the spatially valid flag. This flag combines a set of tests applied to the record to see how reliable are its spatial data components.

catalogNumber An identifier (preferably unique) for the record within the data set or collection.

gbifID The identifier assigned by GBIF for each record.

datasetID An identifier for the set of data. May be a global unique identifier or an identifier specific to a collection or institution.

institutionCode The name (or acronym) in use by the institution having custody of the object(s) or information referred to in the record. Examples MVZ, FMNH, CLO, UCMP.

datasetName The name identifying the data set from which the record was derived.

otherCatalogNumbers A list (concatenated and separated) of previous or alternate fully qualified catalog numbers or other human-used identifiers for the same Occurrence, whether in the current or any other data set or collection.

occurrenceID An identifier for the Occurrence (as opposed to a particular digital record of the occurrence). In the absence of a persistent global unique identifier, construct one from a combination of identifiers in the record that will most closely make the occurrenceID globally unique.

taxonKey The GBIF-assigned taxon identifier number.

collectionID An identifier for the collection or dataset from which the record was derived.

verbatim_scientificName The verbatim (originally-provided) scientific name

verbatimEventDate The verbatim original representation of the date and time information for an Event.

associatedTaxa A list (concatenated and separated) of identifiers or names of taxa and the associations of this Occurrence to each of them.

associatedOrganisms A list (concatenated and separated) of identifiers of other Organisms and the associations of this Organism to each of them.

fieldNotes One of a) an indicator of the existence of, b) a reference to (publication, URI), or c) the text of notes taken in the field about the Event.

sex The sex of the biological individual(s) represented in the Occurrence.

rights A description of the usage rights applicable to the record.

rightsHolder A person or organization owning or managing rights over the resource.

accessRights Information about who can access the resource or an indication of its security status.

associatedReferences A list (concatenated and separated) of identifiers (publication, bibliographic reference, global unique identifier, URI) of literature associated with the Occurrence.

bibliographicCitation A bibliographic reference for the resource as a statement indicating how this record should be cited (attributed) when used.

references A related resource that is referenced, cited, or otherwise pointed to by the described resource.

informationWithheld Additional information that exists, but that has not been shared in the given record.

isDuplicateOf Additional information that exists, but that has not been shared in the given record.

hasCoordinate Variable indicating presence/absence of location coordinates.

hasGeospatialIssues Variable indicating validity of geospatial data associated with record.

occurrenceYear Year associated with Occurrence.

id Variable with identifying value for the Occurrenc.

duplicateStatus Variable indicating is Occurrence is duplicate or not.

associatedOccurrences A list (concatenated and separated) of identifiers of other Occurrence records and their associations to this Occurrence.

locationRemarks Comments or notes about the Location.

dataSource BeeBDC assigned source of the data. Often written when the data is formatted by a BeeBDC::xxx_readr function or similar.

verbatim_scientificName The verbatim (originally-provided) scientific name

.scientificName_empty Flag produced by `bdc::bdc_scientificName_empty()` where FALSE == no scientific name provided and TRUE means that there is text in that column.

.coordinates_empty Flag produced by `bdc::bdc_coordinates_empty()` where FALSE == no coordinates provided.

.coordinates_outOfRange Flag produced by `bdc::bdc_coordinates_outOfRange()` where FALSE == point off the earth. This function identifies records with out-of-range coordinates (not between -90 and 90 for latitude; between -180 and 180 for longitude).

.basisOfRecords_notStandard Flag produced by `bdc::bdc_basisOfRecords_notStandard()` where FALSE == an occurrence with a basisOfRecord not defined as acceptable by the user.

country_suggested A country name suggested by the `bdc::bdc_country_standardized()` function.

countryCode A country code suggested by the `bdc::bdc_country_standardized()` function.

coordinates_transposed A column indicating if coordinates were tansposed by `jbd_Ctrans_chunker()` where FALSE == transposed.

.coordinates_country_inconsistent A flag generated by `jbd_coordCountryInconsistent()` where FALSE == an occurrence where the country name and coordinates did not match.

.occurrenceAbsent A flag generated by `flagAbsent()` where FALSE == occurrences marked as "ABSENT" in the "occurrenceStatus" column

.unLicensed A flag generated by `flagLicense()` where FALSE == those occurrences protected by a restrictive license.

.GBIFflags A flag generated by `GBIFissues()` where FALSE == an occurrence with user-specified GBIF issues to flag.

.uncer_terms A flag generated by `bdc::bdc_clean_names()` where FALSE == the presence of taxonomic uncertainty terms.

names_clean A column made by `bdc::bdc_clean_names()` indicating the cleaned scientificName

.invalidName A flag generated by `harmoniseR()` where FALSE == occurrences whose scientificName did not match the Discover Life taxonomy.

.rou A flag generated by `CoordinateCleaner::clean_coordinates()` where FALSE == rounded (probably imprecise) coordinates.

.val A flag generated by `CoordinateCleaner::clean_coordinates()` where FALSE == invalid coordinates.

.equ A flag generated by `CoordinateCleaner::clean_coordinates()` where FALSE == equal coordinates (e.g., 0.1, 0.1).

.zer A flag generated by `CoordinateCleaner::clean_coordinates()` where FALSE == zeros as coordinates

.cap A flag generated by `CoordinateCleaner::clean_coordinates()` where FALSE == records around country capital centroid.

.cen A flag generated by `CoordinateCleaner::clean_coordinates()` where FALSE == records around country or province centroids.

.gbf A flag generated by `CoordinateCleaner::clean_coordinates()` where FALSE == records around the GBIF headquarters.

.inst A flag generated by `CoordinateCleaner::clean_coordinates()` where FALSE == records around biodiversity institutions.

.sequential A flag generated by `diagonAlley()` where FALSE == records that are possibly the result of fill-down errors in sequence.

.lonFlag A flag generated by `CoordinateCleaner::cd_round()` where FALSE == potential gridding in the longitude column within dataset.

.latFlag A flag generated by `CoordinateCleaner::cd_round()` where FALSE == potential gridding in the latitude column within dataset.

.gridSummary A flag generated by `CoordinateCleaner::cd_round()` where FALSE == potential gridding in either the longitude or latitude columns within dataset.

.uncertaintyThreshold A flag generated by `coordUncerFlagR()` where FALSE == occurrences that did not pass a user-specified threshold in the "coordinateUncertaintyInMeters" column.

countryMatch A column made by `countryOutlieRs()`. Summarises the occurrence-level result: where the species is not known to occur in that country (noMatch), it is known from a bordering country (neighbour), or it is known to occur in that country (exact).

.countryOutlier A flag generated by `countryOutlieRs()` where FALSE == occurrences the do not occur in a country that concurs with the Discover Life country checklist OR an adjacent country.

.sea A flag generated by `countryOutlieRs()` where FALSE == occurrences that are in the ocean.

.summary A flag generated by `summaryFun()` where FALSE == occurrences flagged as FALSE in any of the .flag columns. In this example it excludes flags in the ".gridSummary", ".lonFlag", ".latFlag", and ".uncer_terms" columns.

.eventDate_empty A flag generated by `bdc::bdc_eventDate_empty()` where FALSE == occurrences with no eventDate provided.

.year_outOfRange A flag generated by `bdc::bdc_year_outOfRange()` where FALSE == occurrences older than a threshold date. In this case 1950.

.duplicates A flag generated by `dupeSummary()` where FALSE == occurrences identified as duplicates. There will be an associated kept duplicate (.duplictes == TRUE) for all duplicate clusters.

References

This data set was created by generating a random subset of 100 rows from the full BeeBDC dataset from the publication: Dorey, J.B., Fischer, E.E., Chesshire, P.R., Nava-Bolaños, A., O'Reilly, R.L., Bossert, S., Collins, S.M., Lichtenberg, E.M., Tucker, E., Smith-Pardo, A., Falcon-Brindis, A., Guevara, D.A., Ribeiro, B.R., de Pedro, D., Hung, J.K.-L., Parys, K.A., McCabe, L.M., Rogan, M.S., Minckley, R.L., Velzco, S.J.E., Griswold, T., Zarrillo, T.A., Jetz, W., Sica, Y.V., Orr, M.C., Guzman, L.M., Ascher, J., Hughes, A.C. & Cobb, N.S. (2023) A globally synthesised and flagged bee occurrence dataset and cleaning workflow. Scientific Data, 10, 1–17. https://www.doi.org/10.1038/S41597-023-02626-W

Examples

```
beesFlagged <- BeeBDC::beesFlagged
head(beesFlagged)
```

`beesRaw` *A dataset of 100 random bee occurrence records without flags or filters applied*

Description

A small bee occurrence dataset with flags generated by BeeBDC used to run example script and test functions. For data types, see `ColTypeR()`.

Usage

```
data("beesRaw", package = "BeeBDC")
```

Format

An object of class `"tibble"`

database_id Occurrence code generated in bdc or BeeBDC

scientificName Full scientificName as shown on DiscoverLife

family Family name

subfamily Subfamily name

genus Genus name

subgenus Subgenus name

subspecies Full name with subspecies name - ALA column

specificEpithet The species name only

infraspecificEpithet The subspecies name only

acceptedNameUsage The full name, with authorship and date information if known, of the currently valid (zoological) or accepted (botanical) taxon.

taxonRank The taxonomic rank of the most specific name in the scientificName.

scientificNameAuthorship The authorship information for the scientificName formatted according to the conventions of the applicable nomenclaturalCode.

identificationQualifier A brief phrase or a standard term ("cf.", "aff.") to express the determiner's doubts about the Identification.

higherClassification A list (concatenated and separated) of taxa names terminating at the rank immediately superior to the taxon referenced in the taxon record.)

identificationReferences A list (concatenated and separated) of references (publication, global unique identifier, URI) used in the Identification.

typeStatus A list (concatenated and separated) of nomenclatural types (type status, typified scientific name, publication) applied to the subject.

previousIdentifications A list (concatenated and separated) of previous assignments of names to the Organism.

verbatimIdentification This term is meant to allow the capture of an unaltered original identification/determination, including identification qualifiers, hybrid formulas, uncertainties, etc. This term is meant to be used in addition to scientificName (and identificationQualifier etc.), not instead of it.

identifiedBy A list (concatenated and separated) of names of people, groups, or organizations who assigned the Taxon to the subject.

dateIdentified The date on which the subject was determined as representing the Taxon.

decimalLatitude The geographic latitude (in decimal degrees, using the spatial reference system given in geodeticDatum) of the geographic center of a Location. Positive values are north of the Equator, negative values are south of it. Legal values lie between -90 and 90, inclusive.

decimalLongitude The geographic longitude (in decimal degrees, using the spatial reference system given in geodeticDatum) of the geographic center of a Location. Positive values are east of the Greenwich Meridian, negative values are west of it. Legal values lie between -180 and 180, inclusive.

stateProvince The name of the next smaller administrative region than country (state, province, canton, department, region, etc.) in which the Location occurs.

continent The name of the continent in which the Location occurs.

locality The specific description of the place.

island The name of the island on or near which the Location occurs.

county The full, unabbreviated name of the next smaller administrative region than stateProvince (county, shire, department, etc.) in which the Location occurs.

municipality The full, unabbreviated name of the next smaller administrative region than county (city, municipality, etc.) in which the Location occurs. Do not use this term for a nearby named place that does not contain the actual location.

license A legal document giving official permission to do something with the resource.

issue A GBIF-defined issue.

eventDate The date-time or interval during which an Event occurred. For occurrences, this is the date-time when the event was recorded. Not suitable for a time in a geological context.

eventTime The time or interval during which an Event occurred.

day The integer day of the month on which the Event occurred.

month The integer month in which the Event occurred.

year The four-digit year in which the Event occurred, according to the Common Era Calendar.

basisOfRecord The specific nature of the data record. Recommended best practice is to use the standard label of one of the Darwin Core classes.PreservedSpecimen, FossilSpecimen, LivingSpecimen, MaterialSample, Event, HumanObservation, MachineObservation, Taxon, Occurrence, MaterialCitation

country The name of the country or major administrative unit in which the Location occurs.

type The nature or genre of the resource. StillImage, MovingImage, Sound, PhysicalObject, Event, Text.

occurrenceStatus A statement about the presence or absence of a Taxon at a Location. present, absent.

recordNumber An identifier given to the Occurrence at the time it was recorded. Often serves as a link between field notes and an Occurrence record, such as a specimen collector's number.

recordedBy A list (concatenated and separated) of names of people, groups, or organizations responsible for recording the original Occurrence. The primary collector or observer, especially one who applies a personal identifier (recordNumber), should be listed first.

eventID An identifier for the set of information associated with an Event (something that occurs at a place and time). May be a global unique identifier or an identifier specific to the data set.

Location A spatial region or named place.

samplingProtocol The names of, references to, or descriptions of the methods or protocols used during an Event. Examples UV light trap, mist net, bottom trawl, ad hoc observation | point count, Penguins from space: faecal stains reveal the location of emperor penguin colonies, https://doi.org/10.1111/j.1466-8238.2009.00467.x, Takats et al. 2001.

samplingEffort The amount of effort expended during an Event. Examples 40 trap-nights, 10 observer-hours, 10 km by foot, 30 km by car.

individualCount The number of individuals present at the time of the Occurrence. Integer.

organismQuantity A number or enumeration value for the quantity of organisms. Examples 27 (organismQuantity) with individuals (organismQuantityType). 12.5 (organismQuantity) with percentage biomass (organismQuantityType). r (organismQuantity) with Braun Blanquet Scale (organismQuantityType). many (organismQuantity) with individuals (organismQuantityType).

coordinatePrecision A decimal representation of the precision of the coordinates given in the decimalLatitude and decimalLongitude.

coordinateUncertaintyInMeters The horizontal distance (in meters) from the given decimalLatitude and decimalLongitude describing the smallest circle containing the whole of the Location. Leave the value empty if the uncertainty is unknown, cannot be estimated, or is not applicable (because there are no coordinates). Zero is not a valid value for this term.

spatiallyValid Occurrence records in the ALA can be filtered by using the spatially valid flag. This flag combines a set of tests applied to the record to see how reliable are its spatial data components.

catalogNumber An identifier (preferably unique) for the record within the data set or collection.

gbifID The identifier assigned by GBIF for each record.

datasetID An identifier for the set of data. May be a global unique identifier or an identifier specific to a collection or institution.

institutionCode The name (or acronym) in use by the institution having custody of the object(s) or information referred to in the record. Examples MVZ, FMNH, CLO, UCMP.

datasetName The name identifying the data set from which the record was derived.

otherCatalogNumbers A list (concatenated and separated) of previous or alternate fully qualified catalog numbers or other human-used identifiers for the same Occurrence, whether in the current or any other data set or collection.

occurrenceID An identifier for the Occurrence (as opposed to a particular digital record of the occurrence). In the absence of a persistent global unique identifier, construct one from a combination of identifiers in the record that will most closely make the occurrenceID globally unique.

taxonKey The GBIF-assigned taxon identifier number.

collectionID An identifier for the collection or dataset from which the record was derived.

verbatim_scientificName The verbatim (originally-provided) scientific name

verbatimEventDate The verbatim original representation of the date and time information for an Event.

associatedTaxa A list (concatenated and separated) of identifiers or names of taxa and the associations of this Occurrence to each of them.

associatedOrganisms A list (concatenated and separated) of identifiers of other Organisms and the associations of this Organism to each of them.

fieldNotes One of a) an indicator of the existence of, b) a reference to (publication, URI), or c) the text of notes taken in the field about the Event.

sex The sex of the biological individual(s) represented in the Occurrence.

rights A description of the usage rights applicable to the record.

rightsHolder A person or organization owning or managing rights over the resource.

accessRights Information about who can access the resource or an indication of its security status.

associatedReferences A list (concatenated and separated) of identifiers (publication, bibliographic reference, global unique identifier, URI) of literature associated with the Occurrence.

bibliographicCitation A bibliographic reference for the resource as a statement indicating how this record should be cited (attributed) when used.

references A related resource that is referenced, cited, or otherwise pointed to by the described resource.

informationWithheld Additional information that exists, but that has not been shared in the given record.

isDuplicateOf Additional information that exists, but that has not been shared in the given record.

hasCoordinate Variable indicating presence/absence of location coordinates.

hasGeospatialIssues Variable indicating validity of geospatial data associated with record.

occurrenceYear Year associated with Occurrence.

id Variable with identifying value for the Occurrenc.

duplicateStatus Variable indicating is Occurrence is duplicate or not.

associatedOccurrences A list (concatenated and separated) of identifiers of other Occurrence records and their associations to this Occurrence.

locationRemarks Comments or notes about the Location.

dataSource BeeBDC assigned source of the data. Often written when the data is formatted by a BeeBDC::xxx_readr function or similar.

verbatim_scientificName The verbatim (originally-provided) scientific name

References

This data set was created by generating a random subset of 100 rows from the full, unfiltered and unflagged, BeeBDC dataset from the publication: Dorey, J.B., Fischer, E.E., Chesshire, P.R., Nava-Bolaños, A., O'Reilly, R.L., Bossert, S., Collins, S.M., Lichtenberg, E.M., Tucker, E., Smith-Pardo,

A., Falcon-Brindis, A., Guevara, D.A., Ribeiro, B.R., de Pedro, D., Hung, J.K.-L., Parys, K.A., McCabe, L.M., Rogan, M.S., Minckley, R.L., Velzco, S.J.E., Griswold, T., Zarrillo, T.A., Jetz, W., Sica, Y.V., Orr, M.C., Guzman, L.M., Ascher, J., Hughes, A.C. & Cobb, N.S. (2023) A globally synthesised and flagged bee occurrence dataset and cleaning workflow. Scientific Data, 10, 1–17. https://www.doi.org/10.1038/S41597-023-02626-W

Examples

```
beesRaw <- BeeBDC::beesRaw
head(beesRaw)
```

---

beesTaxonomy *Download a nearly complete taxonomy of bees globally*

Description

Downloads the taxonomic information for the bees of the world. Source of taxonomy is listed under "source" but are mostly derived from the Discover Life website. The data will be sourced from the BeeBDC article's Figshare.

Note that sometimes the download might not work without restarting R. In this case, you could alternatively download the dataset from the URL below and then read it in using `base::readRDS("filePath.Rda")`. Note that as of version 1.3.2, this function internally uses a modified version of the "download" function from the `downloader` package on CRAN. Additionally, for Windows machines, BeeBDC will try a different download method with each failed attempt (for Windows: "auto" > "internal" > "libcurl" > "wget" > "curl"; or "auto" if any particular method is not available).

Usage

```
beesTaxonomy(
  URL = "https://api.figshare.com/v2/file/download/64193731",
  mode = NULL,
  headers = NULL,
  token = NULL,
  alternateURL = FALSE,
  ...
)
```

Arguments

URL	A character vector to the FigShare location of the dataset. The default will be to the most-recent version.
mode	A character passed on to `utils::download.file()`. Default = "wb" for Windows or "w" for Mac/Linux.
headers	Character. Passed on to `utils::download.file()`. Default = NULL.
token	Character. Optional personal access token from FigShare account for authorisation. Default = NULL.

`alternateURL`	Logical. If TRUE then the function will use the an alernate version of the download URL. Might be worth trying if it is failing. Default = FALSE.
`...`	Extra variables that can be passed to `downloader::download()`.

Details

Column details

flags Flags or comments about the taxon name.

taxonomic_status Taxonomic status. Values are "accepted" or "synonym"

source Source of the name.

accid The id of the accepted taxon name or "0" if taxonomic_status == accepted.

id The id number for the taxon name.

kingdom The biological kingdom the taxon belongs to. For bees, kingdom == Animalia.

phylum The biological phylum the taxon belongs to. For bees, phylum == Arthropoda.

class The biological class the taxon belongs to. For bees, class == Insecta.

order The biological order the taxon belongs to. For bees, order == Hymenoptera.

family The family of bee which the species belongs to.

subfamily The subfamily of bee which the species belongs to.

tribe The tribe of bee which the species belongs to.

subtribe The subtribe of bee which the species belongs to.

validName The valid scientific name as it should occur in the "scientificName" column in a Darwin Core file.

canonical The scientificName without the scientificNameAuthority.

canonical_withFlags The scientificName without the scientificNameAuthority and with Discover Life taxonomy flags.

genus The genus the bee species belongs to.

subgenus The subgenus the bee species belongs to.

species The specific epithet for the bee species.

infraspecies The infraspecific epithet for the bee addressed.

authorship The author who described the bee species.

taxon_rank Rank for the bee taxon addressed in the entry.

notes Additional notes about the name/taxon.

Previous taxonomies:

- 2026-05-04 **current**: https://api.figshare.com/v2/file/download/64193731
- 2026-01-12: https://open.flinders.edu.au/ndownloader/files/60945820
- 2024-06-17: https://open.flinders.edu.au/ndownloader/files/47089969
- 2023-11-29: https://open.flinders.edu.au/ndownloader/files/43331472
- 2023-10-10: https://open.flinders.edu.au/ndownloader/files/42613126
- 2023-09-20: https://open.flinders.edu.au/ndownloader/files/42402264
- Original: https://open.flinders.edu.au/ndownloader/files/42320595

Value

A downloaded beesTaxonomy.Rda file in the `tempdir()` and the same tibble returned to the environment.

References

This dataset was created using the Discover Life taxonomy. Dataset is from the publication: DOREY, J. B., CHESSHIRE, P. R., BOLAÑOS, A. N., O'REILLY, R. L., BOSSERT, S., COLLINS, S. M., LICHTENBERG, E. M., TUCKER, E., SMITH-PARDO, A., FALCON-BRINDIS, A., GUEVARA, D. A., RIBEIRO, B. R., DE PEDRO, D., FISCHER, E., HUNG, J. K.-L., PARYS, K. A., ROGAN, M. S., MINCKLEY, R. L., VELZCO, S. J. E., GRISWOLD, T., ZARRILLO, T. A., SICA, Y., ORR, M. C., GUZMAN, L. M., ASCHER, J., HUGHES, A. C. & COBB, N. S. In review. A globally synthesised and flagged bee occurrence dataset and cleaning workflow. Scientific Data. The taxonomy data are mostly compiled from Discover Life data, www.discoverlife.org: ASCHER, J. S. & PICKERING, J. 2020. Discover Life bee species guide and world checklist (Hymenoptera: Apoidea: Anthophila). http://www.discoverlife.org/mp/20q?guide=Apoidea_species.

See Also

`taxadbToBeeBDC()` to download any other taxonomy (of any taxa or of bees) and `harmoniseR()` for the taxon-cleaning function where these taxonomies are implemented.

Examples

```
## Not run:
beesTaxonomy <- BeeBDC::beesTaxonomy()

## End(Not run)
```

`ChaoWrapper` *Parallel estimation of species richness in a community using iChao*

Description

ChaoSpecies: Estimation of species richness in a single community based on five types of data: Type (1) abundance data (datatype="abundance"), Type (1A) abundance-frequency counts (datatype="abundance_freq_count Type (2) incidence-frequency data (datatype = "incidence_freq"), Type (2A) incidence-frequency counts (datatype="incidence_freq_count"), and Type (2B) incidence-raw data (datatype="incidence_raw"); see SpadeR-package details for data input formats.

Usage

```
ChaoWrapper(
  data = NULL,
  datatype = "abundance",
```

```
  k = 10,
  conf = 0.95,
  mc.cores = 1
)
```

Arguments

data	A data frame or tibble. A data frame containing "abundance"-type data per variable (population, country, species...) in columns.
datatype	Character. The type of input data, "abundance", "abundance_freq_count", "incidence_freq", "incidence_freq_count" or "incidence_raw". So far only tested with "abundance" data. Default = "abundace".
k	Numeric. the cut-off point (default = 10), which separates species into "abundant" and "rare" groups for abundance data for the estimator ACE; it separates species into "frequent" and "infrequent" groups for incidence data for the estimator ICE. Default = 10.
conf	Numeric. A positive number equal to or less than 1 specifying the level of confidence interval. Default = 0.95.
mc.cores	Numeric. If > 1, the function will run in parallel using mclapply using the number of cores specified. If = 1 then it will be run using a serial loop. NOTE: Windows machines must use a value of 1 (see ?parallel::mclapply). Additionally, be aware that each thread can use large chunks of memory. Default = 1.

Value

Returns a list containing two tibbles. The first is a tibble that concatenates the outputs from the basic data and rare species information in columns per input variable (column). The second is a tibble that concatenates the various species richness estimates, with input variables in chunks of rows. Additionally a console output will list the variables (columns) that lacked sufficient data to be analysed.

See Also

SpadeR::ChaoSpecies()

Examples

```
## Not run:
 # Read in some example data and use [BeeBDC::richnessPrepR()] to create the example input data
 #' data(beesCountrySubset)

estimateDataExample <- BeeBDC::richnessPrepR(
  data = beesCountrySubset,
  # Download the taxonomy
  taxonomyFile = BeeBDC::beesTaxonomy(),
  # Download the checklist
  checklistFile = BeeBDC::beesChecklist(),
  curveFunction = function(x) (228.7531 * x * x^-log(12.1593)),
  sampleSize = 10000,
```

```
  countryColumn = "country_suggested",
  limitGlobal = NULL,
  outPath = tempdir()
)

  # Transform the data for input
inputTestData <- estimateDataExample$site_speciesCounts %>%
  dplyr::select(scientificName, country_suggested, n) %>%
  tidyr::pivot_wider(names_from = country_suggested,
                     values_from = n,
                     values_fill = 0) %>%
  # Create the rownames
tibble::column_to_rownames("scientificName") %>%
  dplyr::tibble()

iChaoOut <- ChaoWrapper(
data = inputTestData,
datatype = "abundance",
k = 10,
conf = 0.95,
mc.cores = 1)

## End(Not run)
```

---

chordDiagramR *Build a chord diagram of duplicate occurrence links*

Description

This function outputs a figure which shows the relative size and direction of occurrence points duplicated between data providers, such as, SCAN, GBIF, ALA, etc. This function requires the outputs generated by dupeSummary().

Usage

```
chordDiagramR(
  dupeData = NULL,
  outPath = NULL,
  fileName = NULL,
  width = 7,
  height = 6,
  bg = "white",
  smallGrpThreshold = 3,
  title = "Duplicated record sources",
  palettes = c("cartography::blue.pal", "cartography::green.pal",
    "cartography::sand.pal", "cartography::orange.pal", "cartography::red.pal",
    "cartography::purple.pal", "cartography::brown.pal"),
  canvas.ylim = c(-1, 1),
  canvas.xlim = c(-0.6, 0.25),
```

```
  text.col = "black",
  legendX = grid::unit(6, "mm"),
  legendY = grid::unit(18, "mm"),
  legendJustify = c("left", "bottom"),
  niceFacing = TRUE,
  self.link = 2
)
```

Arguments

dupeData	A tibble or data frame. The duplicate file produced by dupeSummary().
outPath	Character. The path to a directory (folder) in which the output should be saved.
fileName	Character. The name of the output file, ending in '.pdf'.
width	Numeric. The width of the figure to save (in inches). Default = 7.
height	Numeric. The height of the figure to save (in inches). Default = 6.
bg	The plot's background colour. Default = "white".
smallGrpThreshold	Numeric. The upper threshold of sub-dataSources to be listed as "other". Default = 3.
title	A character string. The figure title. Default = "Duplicated record sources".
palettes	A vector of the palettes to be used. One palette for each major dataSource and "other" using the paletteer package. Default = c("cartography::blue.pal", "cartography::green.pal", "cartography::sand.pal", "cartography::orange.pal", "cartography::red.pal", "cartography::purple.pal", "cartography::brown.pal")
canvas.ylim	Canvas limits from circlize::circos.par(). Default = c(-1.0,1.0).
canvas.xlim	Canvas limits from circlize::circos.par(). Default = c(-0.6, 0.25).
text.col	A character string. Text colour
legendX	The x position of the legends, as measured in current viewport. Passed to ComplexHeatmap::draw(). Default = grid::unit(6, "mm").
legendY	The y position of the legends, as measured in current viewport. Passed to ComplexHeatmap::draw(). Default = grid::unit(18, "mm").
legendJustify	A character vector declaring the justification of the legends. Passed to ComplexHeatmap::draw(). Default = c("left", "bottom").
niceFacing	TRUE/FALSE. The niceFacing option automatically adjusts the text facing according to their positions in the circle. Passed to circlize::highlight.sector().
self.link	1 or 2 (numeric). Passed to circlize::chordDiagram(): if there is a self link in one sector, 1 means the link will be degenerated as a 'mountain' and the width corresponds to the value for this connection. 2 means the width of the starting root and the ending root all have the width that corresponds to the value for the connection.

Value

Saves a figure to the provided file path.

Examples

```
## Not run:
  # Create a basic example dataset of duplicates to visualise
basicData <- dplyr::tribble(
                            ~dataSource,    ~dataSource_keep,
                       "GBIF_Halictidae",         "USGS_data",
                       "GBIF_Halictidae",         "USGS_data",
                       "GBIF_Halictidae",         "USGS_data",
                       "GBIF_Halictidae",         "USGS_data",
                       "GBIF_Halictidae",         "USGS_data",
                       "GBIF_Halictidae",         "USGS_data",
                       "SCAN_Halictidae",   "GBIF_Halictidae",
                    "iDigBio_halictidae",   "GBIF_Halictidae",
                    "iDigBio_halictidae",   "SCAN_Halictidae",
                    "iDigBio_halictidae",   "SCAN_Halictidae",
                       "SCAN_Halictidae",   "GBIF_Halictidae",
                        "iDigBio_apidae",       "SCAN_Apidae",
                           "SCAN_Apidae",    "Ecd_Anthophila",
                        "iDigBio_apidae",    "Ecd_Anthophila",
                           "SCAN_Apidae",    "Ecd_Anthophila",
                        "iDigBio_apidae",    "Ecd_Anthophila",
                     "SCAN_Megachilidae", "SCAN_Megachilidae",
                       "CAES_Anthophila",   "CAES_Anthophila",
                       "CAES_Anthophila",   "CAES_Anthophila"
 )


 chordDiagramR(
dupeData = basicData,
outPath = tempdir(),
fileName = "ChordDiagram.pdf",
# These can be modified to help fit the final pdf that's exported.
width = 9,
height = 7.5,
bg = "white",
# How few distinct dataSources should a group have to be listed as "other"
smallGrpThreshold = 3,
title = "Duplicated record sources",
# The default list of colour palettes to choose from using the paleteer package
palettes = c("cartography::blue.pal", "cartography::green.pal",
             "cartography::sand.pal", "cartography::orange.pal", "cartography::red.pal",
             "cartography::purple.pal", "cartography::brown.pal"),
canvas.ylim = c(-1.0,1.0),
canvas.xlim = c(-0.6, 0.25),
text.col = "black",
legendX = grid::unit(6, "mm"),
legendY = grid::unit(18, "mm"),
legendJustify = c("left", "bottom"),
niceFacing = TRUE)
## End(Not run)
```

ColTypeR *Sets up column names and types*

Description

This function uses `readr::cols_only()` to assign a column name and the type of data (e.g., `readr::col_character()`, and `readr::col_integer()`). To see the default columns simply run `ColTypeR()`. This is intended for use with `readr::read_csv()`. Columns that are not present will NOT be included in the resulting tibble unless they are specified using ....

Usage

```
ColTypeR(..., standardFormat = NULL)
```

Arguments

...	Additional arguments. These can be specified in addition to the ones default to the function. For example: • newCharacterColumn = `readr::col_character()`, • newNumericColumn = `readr::col_integer()`, • newLogicalColumn = `readr::col_logical()`
standardFormat	Character. Some taxa may have a standard format for data. Presently, Only bees have had this encoded here as "bee". Default = NULL.

Value

Returns an object of class col_spec. See `readr::as.col_spec()` for additional context and explication.

References

For using the bee standard — Clos, B. D., Seltmann, K. C., Turley, N. E., Maffei, C., Tucker, E. M., Lane, I. G., Levenson, H. K., & Woodard, H. S. (2025). Improving the standardization of wild bee occurrence data: towards a formal wild bee data standard. Authorea. doi: https://doi.org/10.22541/au.173862402.22787949/v

Examples

```
  # You can simply return the below for default values
  library(dplyr)
BeeBDC::ColTypeR()

  # To add new columns you can write
ColTypeR(newCharacterColumn = readr::col_character(),
         newNumericColumn = readr::col_integer(),
         newLogicalColumn = readr::col_logical())

# Try reading in one of the test datasets as an example:
```

```
beesFlagged %>% dplyr::as_tibble(col_types = BeeBDC::ColTypeR())
  # OR
beesRaw %>% dplyr::as_tibble(col_types = BeeBDC::ColTypeR())

  # OR, using the bee standard format from:

beesRaw %>% dplyr::as_tibble(col_types = BeeBDC::ColTypeR(standardFormat = "bee"))
```

`continentOutlieRs` *Flag continent-level outliers with a provided checklist.*

Description

This function flags continent-level outliers using the checklist provided with this package. For additional context and column names, see `beesChecklist()`.

Usage

```
continentOutlieRs(
  checklist = NULL,
  data = NULL,
  keepAdjacentContinent = FALSE,
  pointBuffer = NULL,
  scale = 50,
  stepSize = 1e+06,
  mc.cores = 1
)
```

Arguments

`checklist`	A data frame or tibble. The formatted checklist which was built based on the Discover Life website.
`data`	A data frame or tibble. The a Darwin Core occurrence dataset.
`keepAdjacentContinent`	Logical. If TRUE, occurrences in continents that are adjacent to checklist continents will be kept. If FALSE, they will be flagged. Defualt = FALSE.
`pointBuffer`	Numeric. A buffer around points to help them align with a continent or coastline. This provides a good way to retain points that occur right along the coast or borders of the maps in rnaturalearth
`scale`	Numeric. The value fed into the map scale parameter for `rnaturalearth::ne_countries()`'s scale parameter: Scale of map to return, one of 110, 50, 10 or 'small', 'medium', 'large', where smaller numbers are higher resolution. WARNING: This function is tested on 110 and 50.
`stepSize`	Numeric. The number of occurrences to process in each chunk. Default = 1000000.

mc.cores — Numeric. If > 1, the function will run in parallel using mclapply using the number of cores specified. If = 1 then it will be run using a serial loop. NOTE: Windows machines must use a value of 1 (see ?parallel::mclapply). Additionally, be aware that each thread can use large chunks of memory. If the cores throw issues, consider setting mc.cores to 1. Default = 1.

Value

The input data with two new columns, .continentOutlier or .sea. There are three possible values for the new column: TRUE == passed, FALSE == failed (not in continent or in the ocean), NA == did not overlap with rnaturalearth map.

See Also

countryOutlieRs() for implementation at the country level. Country-level implementation will be more data-hungry and, where data do not yet exist, difficult to implement. Additionally, see beesChecklist() for input data. Note, not all columns are necessary if you are building your own dataset. At a minimum you will need *validName* and *continent*.

Examples

```
if(requireNamespace("rnaturalearthdata", quietly = TRUE)){
library(magrittr)
  # Load in the test dataset
beesRaw <- BeeBDC::beesRaw
  # For the sake of this example, use the testChecklist
system.file("extdata", "testChecklist.rda", package="BeeBDC") |> load()
  # For real examples, you might download the beesChecklist from FigShare using
  #  [BeeBDC::beesChecklist()]

beesRaw_out <- continentOutlieRs(checklist = testChecklist,
                                 data = beesRaw %>%
                                 dplyr::filter(dplyr::row_number() %in% 1:50),
                                 keepAdjacentContinent = FALSE,
                                 pointBuffer = 1,
                                 scale = 50,
                                 stepSize = 1000000,
                                 mc.cores = 1)
table(beesRaw_out$.continentOutlier, useNA = "always")
} # END if require
```

---

coordUncerFlagR *Flag occurrences with an uncertainty threshold*

Description

To use this function, the user must choose a column, probably "coordinateUncertaintyInMeters" and a threshold above which occurrences will be flagged for geographic uncertainty.

Usage

```
coordUncerFlagR(
  data = NULL,
  uncerColumn = "coordinateUncertaintyInMeters",
  threshold = NULL
)
```

Arguments

data	A data frame or tibble. Occurrence records as input.
uncerColumn	Character. The column to flag uncertainty in.
threshold	Numeric. The uncertainty threshold. Values equal to, or greater than, this threshold will be flagged.

Value

The input data with a new column, .uncertaintyThreshold.

Examples

```
# Run the function
beesRaw_out <- coordUncerFlagR(data = beesRaw,
                               uncerColumn = "coordinateUncertaintyInMeters",
                               threshold = 1000)
# View the output
table(beesRaw_out$.uncertaintyThreshold, useNA = "always")
```

---

countryHarmoniseR *Match and harmonise country names*

Description

A small function that is useful for harmonising country names so that they mathc between datasets. This relies on the country names matching one of the exceptions pre-coded in and is so far used internally.

Usage

```
countryHarmoniseR(
  data = NULL,
  countryColumn = NULL,
  shorterNames = TRUE,
  continentAnalysis = FALSE
)
```

Arguments

data	A data frame or tibble.
countryColumn	Character. The column that contains country names to be harmonised.
shorterNames	Logical. If TRUE, some very long country names will be shortened. This can be helpful for writing country names in plots where long names are annoying.
continentAnalysis	Logical. Set to TRUE in order to match small entities to continents; limited use for the moment.

Value

Returns the original data frame or tibble but with harmonised country names

Examples

```
# load in the test dataset
system.file("extdata", "testTaxonomy.rda", package="BeeBDC") |> load()
harmonisedCountries <- countryHarmoniseR(
  data = testTaxonomy,
  countryColumn = "country_suggested"
)
```

---

countryNameCleanR *Fix country name issues using a user-input list*

Description

This function is basic for a user to manually fix some country name inconsistencies.

Usage

```
countryNameCleanR(data = NULL, ISO2_table = NULL, commonProblems = NULL)
```

Arguments

data	A data frame or tibble. Occurrence records as input.
ISO2_table	A data frame or tibble with the columns ISO2 and long names for country names. Default is a static version from Wikipedia.
commonProblems	A data frame or tibble. It must have two columns: one containing the user-identified problem and one with a user-defined fix

Value

Returns the input data, but with countries occurring in the user-supplied problem column ("commonProblems") replaced with those in the user-supplied fix column

Examples

```
beesFlagged_out <- countryNameCleanR(
data = BeeBDC::beesFlagged,
commonProblems = dplyr::tibble(problem = c('U.S.A.', 'US','USA','usa','UNITED STATES',
                        'United States','U.S.A','MX','CA','Bras.','Braz.',
                        'Brasil','CNMI','USA TERRITORY: PUERTO RICO'),
                        fix = c('United States of America','United States of America',
                                'United States of America','United States of America',
                                'United States of America','United States of America',
                                'United States of America','Mexico','Canada','Brazil',
                           'Brazil','Brazil','Northern Mariana Islands','PUERTO.RICO')))
```

`countryOutlieRs` *Flag country-level outliers with a provided checklist.*

Description

This function flags country-level outliers using the checklist provided with this package. For additional context and column names, see `beesChecklist()`.

Usage

```
countryOutlieRs(
  checklist = NULL,
  data = NULL,
  keepAdjacentCountry = TRUE,
  pointBuffer = NULL,
  scale = 50,
  stepSize = 1e+06,
  mc.cores = 1
)
```

Arguments

`checklist`	A data frame or tibble. The formatted checklist which was built based on the Discover Life website.
`data`	A data frame or tibble. The a Darwin Core occurrence dataset.
`keepAdjacentCountry`	Logical. If TRUE, occurrences in countries that are adjacent to checklist countries will be kept. If FALSE, they will be flagged.
`pointBuffer`	Numeric. A buffer around points to help them align with a country or coastline. This provides a good way to retain points that occur right along the coast or borders of the maps in rnaturalearth
`scale`	Numeric. The value fed into the map scale parameter for `rnaturalearth::ne_countries()`'s scale parameter: Scale of map to return, one of 110, 50, 10 or 'small', 'medium', 'large', where smaller numbers are higher resolution. WARNING: This function is tested on 110 and 50.

stepSize — Numeric. The number of occurrences to process in each chunk. Default = 1000000.

mc.cores — Numeric. If > 1, the function will run in parallel using mclapply using the number of cores specified. If = 1 then it will be run using a serial loop. NOTE: Windows machines must use a value of 1 (see ?parallel::mclapply). Additionally, be aware that each thread can use large chunks of memory. If the cores throw issues, consider setting mc.cores to 1. Default = 1.

Value

The input data with two new columns, .countryOutlier or .sea. There are three possible values for the new column: TRUE == passed, FALSE == failed (not in country or in the ocean), NA == did not overlap with rnaturalearth map.

See Also

`continentOutlieRs()` for implementation at the continent level. Implementation at the continent level may be lighter and more manageable on the data side of things where country-level checklists don't exist. Additionally, see `beesChecklist()` for input data. Note, not all columns are necessary if you are building your own dataset. At a minimum you will need *validName*, *country*, *iso_a3_eh* (to match rnaturalearth).

Examples

```
if(requireNamespace("rnaturalearthdata", quietly = TRUE)){
library(magrittr)
  # Load in the test dataset
beesRaw <- BeeBDC::beesRaw
  # For the sake of this example, use the testChecklist
system.file("extdata", "testChecklist.rda", package="BeeBDC") |> load()
  # For real examples, you might download the beesChecklist from FigShare using
  #  [BeeBDC::beesChecklist()]

beesRaw_out <- countryOutlieRs(checklist = testChecklist,
                               data = beesRaw %>%
                               dplyr::filter(dplyr::row_number() %in% 1:50),
                               keepAdjacentCountry = TRUE,
                               pointBuffer = 1,
                               scale = 50,
                               stepSize = 1000000,
                               mc.cores = 1)
table(beesRaw_out$.countryOutlier, useNA = "always")
} # END if require
```

---

dataProvTables *Build a table of data providers for bee occurrence records*

Description

This function will attempt to find and build a table of data providers that have contributed to the input data, especially using the 'institutionCode' column. It will also look for a variety of other columns to find data providers using an internally set sequence of if-else statements. Hence, this function is quite specific for bee data, but should work for other taxa in similar institutions.

Usage

```
dataProvTables(
  data = NULL,
  runBeeDataChecks = FALSE,
  outPath = OutPath_Report,
  fileName = NULL
)
```

Arguments

data	A data frame or tibble. Occurrence records as input.
runBeeDataChecks	Logical. If TRUE, will search in other columns for specific clues to determine the institution.
outPath	A character path. The path to the directory in which the figure will be saved. Default = OutPath_Report.
fileName	Character. The name of the file to be saved, ending in ".csv".

Value

Returns a table with the data providers, an specimen count, and a species count.

Examples

```
data(beesFlagged)

testOut <- dataProvTables(
data = beesFlagged,
runBeeDataChecks = TRUE,
outPath = tempdir(),
fileName = "testFile.csv")
```

---

dataSaver *Simple function to save occurrence AND EML data as a list*

Description

Used at the end of 1.x in the example workflow in order to save the occurrence dataset and its associated eml metadata.

Usage

```
dataSaver(
  path = NULL,
  save_type = NULL,
  occurrences = NULL,
  eml_files = NULL,
  file_prefix = NULL
)
```

Arguments

path	Character. The main file path to look for data in.
save_type	Character. The file format in which to save occurrence and EML data. Either "R_file" or "CSV_file"
occurrences	The occurrences to save as a data frame or tibble.
eml_files	A list of the EML files.
file_prefix	Character. A prefix for the resulting output file.

Value

This function saves both occurrence and EML data as a list when save_type = "R_File" or as individual csv files when save_type = "CSV_file".

Examples

```
## Not run:
dataSaver(path = tempdir(),# The main path to look for data in
save_type = "CSV_file", # "R_file" OR "CSV_file"
occurrences = Complete_data$Data_WebDL, # The existing datasheet
eml_files = Complete_data$eml_files, # The existing EML files
file_prefix = "Fin_") # The prefix for the file name

## End(Not run)
```

---

dateFindR *Find dates in other columns*

Description

A function made to search other columns for dates and add them to the eventDate column. The function searches the columns locality, fieldNotes, locationRemarks, and verbatimEventDate for the relevant information. Additionally, for date ranges in the eventDate column, these will Be translated into startDayOfYear and endDayOfYear and the eventDate will be kept as the last date, but formatted in the correct format. Ambiguous dates that can't be parsed reliably as day-month-year or month-day-year will be rounded to the nearest certainty; if month or day can be identified with a spelled-out month or a day >12, these wil be correctly formatted. Year-month-day... is the most-reliable format

Usage

```
dateFindR(data = NULL, maxYear = lubridate::year(Sys.Date()), minYear = 1700)
```

Arguments

data	A data frame or tibble. Occurrence records as input.
maxYear	Numeric. The maximum year considered reasonable to find. Default = lubridate::year(Sys.Date()).
minYear	Numeric. The minimum year considered reasonable to find. Default = 1700.

Value

The function results in the input occurrence data with but with updated eventDate, year, month, and day columns for occurrences where these data were a) missing and b) located in one of the searched columns.

Examples

```
# Using the example dataset, you may not find any missing eventDates are rescued (dependent on
# which version of the example dataset the user inputs.
beesRaw_out <- dateFindR(data = beesRaw,
                         # Years above this are removed (from the recovered dates only)
                         maxYear = lubridate::year(Sys.Date()),
                         # Years below this are removed (from the recovered dates only)
                         minYear = 1700)
```

---

diagonAlley *Find fill-down errors*

Description

A simple function that looks for potential latitude and longitude fill-down errors by identifying consecutive occurrences with coordinates at regular intervals. This is accomplished by using a sliding window with the length determined by minRepeats.

Usage

```
diagonAlley(
  data = NULL,
  minRepeats = NULL,
  groupingColumns = c("eventDate", "recordedBy", "datasetName"),
  ndec = 3,
  stepSize = 1e+06,
  mc.cores = 1
)
```

Arguments

data	A data frame or tibble. Occurrence records as input.
minRepeats	Numeric. The minimum number of lat or lon repeats needed to flag a record
groupingColumns	Character. The column(s) to group the analysis by and search for fill-down errors within. Default = c("eventDate", "recordedBy", "datasetName").
ndec	Numeric. The number of decimal places below which records will not be considered in the diagonAlley function. This is fed into jbd_coordinates_precision(). Default = 3.
stepSize	Numeric. The number of occurrences to process in each chunk. Default = 1000000.
mc.cores	Numeric. If > 1, the function will run in parallel using mclapply using the number of cores specified. If = 1 then it will be run using a serial loop. NOTE: Windows machines must use a value of 1 (see ?parallel::mclapply). Additionally, be aware that each thread can use large chunks of memory. Default = 1.

Details

The sliding window (and hence fill-down errors) will only be examined within the user-defined groupingColumns; if any of those columns are empty, that record will be excluded.

Value

The function returns the input data with a new column, .sequential, where FALSE = records that have consecutive latitudes or longitudes greater than or equal to the user-defined threshold.

Examples

```
# Read in the example data
  data(beesRaw)
 # Run the function
  beesRaw_out <- diagonAlley(
    data = beesRaw,
    # The minimum number of repeats needed to find a sequence in for flagging
    minRepeats = 4,
    groupingColumns = c("eventDate", "recordedBy", "datasetName"),
    ndec = 3,
    stepSize = 1000000,
    mc.cores = 1)
```

`dirMaker` *Set up global directory paths and create folders*

Description

This function sets up a directory for saving outputs (i.e. data, figures) generated through the use of the BeeBDC package, if the required folders do not already exist.

Usage

```
dirMaker(
  RootPath = RootPath,
  ScriptPath = NULL,
  DataPath = NULL,
  DataSubPath = "/Data_acquisition_workflow",
  DiscLifePath = NULL,
  OutPath = NULL,
  OutPathName = "Output",
  Report = TRUE,
  Check = TRUE,
  Figures = TRUE,
  Intermediate = TRUE,
  RDoc = NULL,
  useHere = TRUE
)
```

Arguments

`RootPath`	A character String. The `RootPath` is the base path for your project, and all other paths should ideally be located within the `RootPath`. However, users may specify paths not contained in the RootPath
`ScriptPath`	A character String. The `ScriptPath` is the path to any additional functions that you would like to read in for use with BeeBDC.
`DataPath`	A character string. The path to the folder containing bee occurrence data to be flagged and/or cleaned
`DataSubPath`	A character String. If a `DataPath` is not provided, this will be used as the `DataPath` folder name within the `RootPath`. Default is "/Data_acquisition_workflow"
`DiscLifePath`	A character String. The path to the folder which contains data from Ascher and Pcikering's Discover Life website.
`OutPath`	A character String. The path to the folder where output data will be saved.
`OutPathName`	A character String. The name of the `OutPath` subfolder located within the `RootPath`. Default is "Output".
`Report`	Logical. If TRUE, function creates a "Report" folder within the OutPath-defined folder. Default = TRUE.

Check	Logical. If TRUE, function creates a "Check" folder within the OutPath-defined folder. Default = TRUE.
Figures	Logical. If TRUE, function creates a "Figures" folder within the OutPath-defined folder. Default = TRUE.
Intermediate	Logical. If TRUE, function creates a "Intermediate" folder within the OutPath-defined folder in which to save intermediate datasets. Default = TRUE.
RDoc	A character String. The path to the current script or report, relative to the project root. Passing an absolute path raises an error. This argument is used by here::i_am() and incorrectly setting this may result in bdc figures being saved to your computer's root directory
useHere	Logical. If TRUE, dirMaker will use here::i_am() to declare the relative path to 'RDoc'. This is aimed at preserving some functionality with where bdc saves summary figures and tables. Default = TRUE.

Value

Results in the generation of a list containing the BeeBDC-required directories in your global environment. This function should be run at the start of each session. Additionally, this function will create the BeeBDC-required folders if they do not already exist in the supplied directory

Examples

```
  # load dplyr
  library(dplyr)
# Standard/basic usage:
RootPath <- tempdir()
dirMaker(
RootPath = RootPath,
# Input the location of the workflow script RELATIVE to the RootPath
RDoc = NULL,
useHere = FALSE) %>%
  # Add paths created by this function to the environment()
  list2env(envir = environment())

# Custom OutPathName provided
  dirMaker(
 RootPath = RootPath,
 # Set some custom OutPath info
 OutPath = NULL,
 OutPathName = "T2T_Output",
 # Input the location of the workflow script RELATIVE to the RootPath
 RDoc = NULL,
 useHere = FALSE) %>%
   # Add paths created by this function to the environment()
   list2env(envir = environment())
 # Set the working directory

# Further customisations are also possible
dirMaker(
  RootPath = RootPath,
```

```
  ScriptPath = "...path/Bee_SDM_paper/BDC_repo/BeeBDC/R",
  DiscLifePath = "...path/BDC_repo/DiscoverLife_Data",
  OutPathName = "AsianPerspective_Output",
  # Input the location of the workflow script RELATIVE to the RootPath
  RDoc = NULL,
  useHere = FALSE) %>%
  # Add paths created by this function to the environment()
  list2env(envir = environment())
```

`dupePlotR` *Create a compound bar graph of duplicate data sources*

Description

Creates a plot with two bar graphs. One shows the absolute number of duplicate records for each data source while the other shows the proportion of records that are duplicated within each data source. This function requires a dataset that has been run through `dupeSummary()`.

Usage

```
dupePlotR(
  data = NULL,
  outPath = NULL,
  fileName = NULL,
  legend.position.inside = c(0.85, 0.8),
  base_height = 7,
  base_width = 7,
  ...,
  dupeColours = c("#F2D2A2", "#B9D6BC", "#349B90"),
  returnPlot = FALSE
)
```

Arguments

`data`	A data frame or tibble. Occurrence records as input.
`outPath`	Character. The path to a directory (folder) in which the output should be saved.
`fileName`	Character. The name of the output file, ending in '.pdf'.
`legend.position.inside`	The position of the legend as coordinates. Default = c(0.85, 0.8).
`base_height`	Numeric. The height of the plot in inches. Default = 7.
`base_width`	Numeric. The width of the plot in inches. Default = 7.
`...`	Other arguments to be used to change factor levels of data sources.
`dupeColours`	A vector of colours for the levels duplicate, kept duplicate, and unique. Default = c("#F2D2A2","#B9D6BC", "#349B90").
`returnPlot`	Logical. If TRUE, return the plot to the environment. Default = FALSE.

Value

Outputs a .pdf figure.

Examples

```
# This example will show a warning for the factor levels taht are not present in the specific
# test dataset
dupePlotR(
  data = beesFlagged,
  # The outPath to save the plot as
    # Should be something like: #paste0(OutPath_Figures, "/duplicatePlot_TEST.pdf"),
  outPath = tempdir(),
  fileName = "duplicatePlot_TEST.pdf",
  # Colours in order: duplicate, kept duplicate, unique
  dupeColours = c("#F2D2A2","#B9D6BC", "#349B90"),
  # Plot size and height
  base_height = 7, base_width = 7,
  legend.position.inside = c(0.85, 0.8),
  # Extra variables can be fed into forcats::fct_recode() to change names on plot
  GBIF = "GBIF", SCAN = "SCAN", iDigBio = "iDigBio", USGS = "USGS", ALA = "ALA",
  ASP = "ASP", CAES = "CAES", 'B. Mont.' = "BMont", 'B. Minckley' = "BMin", Ecd = "Ecd",
  Gaiarsa = "Gai", EPEL = "EPEL", Lic = "Lic", Bal = "Bal", Arm = "Arm"
  )
```

---

dupeSummary *Identifies duplicate occurrence records*

Description

This function uses user-specified inputs and columns to identify duplicate occurrence records. Duplicates are identified iteratively and will be tallied up, duplicate pairs clustered, and sorted at the end of the function. The function is designed to work with Darwin Core data with a database_id column, but it is also modifiable to work with other columns.

Usage

```
dupeSummary(
  data = NULL,
  path = NULL,
  duplicatedBy = NULL,
  completeness_cols = NULL,
  idColumns = NULL,
  collectionCols = NULL,
  collectInfoColumns = NULL,
  CustomComparisonsRAW = NULL,
  CustomComparisons = NULL,
  sourceOrder = NULL,
  prefixOrder = NULL,
```

```
  dontFilterThese = c(".gridSummary", ".lonFlag", ".latFlag", ".uncer_terms",
    ".uncertaintyThreshold", ".unLicensed"),
  characterThreshold = 2,
  numberThreshold = 3,
  numberOnlyThreshold = 5,
  catalogSwitch = TRUE
)
```

Arguments

data	A data frame or tibble. Occurrence records as input.
path	A character path to the location where the duplicateRun_ file will be saved.
duplicatedBy	A character vector. Options are c("ID", "collectionInfo", "both"). "ID" columns runs through a series of ID-only columns defined by idColumns. "collectionInfo" runs through a series of columns defined by collectInfoColumns, which are checked in combination with collectionCols. "both" runs both of the above.
completeness_cols	A character vector. A set of columns that are used to order and select duplicates by. For each occurrence, this function will calculate the sum of `complete.cases()`. Within duplicate clusters occurrences with a greater number of the completeness_cols filled in will be kept over those with fewer.
idColumns	A character vector. The columns to be checked individually for internal duplicates. Intended for use with ID columns only.
collectionCols	A character vector. The columns to be checked in combination with each of the completeness_cols.
collectInfoColumns	A character vector. The columns to be checked in combinatino with all of the collectionCols columns.
CustomComparisonsRAW	A list of character vectors. Custom comparisons - as a list of columns to iteratively compare for duplicates. These differ from the CustomComparisons in that they ignore the minimum number and character thresholds for IDs.
CustomComparisons	A list of character vectors. Custom comparisons - as a list of columns to iteratively compare for duplicates. These comparisons are made after character and number thresholds are accounted for in ID columns.
sourceOrder	A character vector. The order in which you want to KEEP duplicated based on the dataSource column (i.e. what order to prioritize data sources). NOTE: These dataSources are simplified to the string prior to the first "_". Hence, "GBIF_Anthophyla" becomes "GBIF."
prefixOrder	A character vector. Like sourceOrder, except based on the database_id prefix, rather than the dataSource. Additionally, this is only examined if prefixOrder != NULL. Default = NULL.
dontFilterThese	A character vector. This should contain the flag columns to be ignored in the creation or updating of the .summary column. Passed to `summaryFun()`.

characterThreshold
: Numeric. The complexity threshold for ID letter length. This is the minimum number of characters that need to be present in ADDITION TO the numberThreshold for an ID number to be tested for duplicates. Ignored by CustomComparisonsRAW. The columns that are checked are occurrenceID, recordId, id, catalogNumber, and otherCatalogNumbers. Default = 2.

numberThreshold
: Numeric. The complexity threshold for ID number length. This is the minimum number of numeric characters that need to be present in ADDITION TO the characterThreshold for an ID number to be tested for duplicates. Ignored by CustomComparisonsRAW. The columns that are checked are occurrenceID, recordId, id, catalogNumber, and otherCatalogNumbers. Default = 3.

numberOnlyThreshold
: Numeric. As numberThreshold except the characterThreshold is ignored. Default = 5.

catalogSwitch
: Logical. If TRUE, and the catalogNumber is empty the function will copy over the otherCatalogNumbers into catalogNumber and visa versa. Hence, the function will attempt to matchmore catalog numbers as both of these functions can be problematic. Default = TRUE.

Value

Returns data with an additional column called .duplicates where FALSE occurrences are duplicates and TRUE occurrences are either kept duplicates or unique. Also exports a .csv to the user-specified location with information about duplicate matching. This file is used by other functions including `manualOutlierFindeR()` and `chordDiagramR()`

See Also

`chordDiagramR()` for creating a chord diagram to visualise linkages between dataSources and `dupePlotR()` to visualise the numbers and proportions of duplicates in each dataSource.

Examples

```
beesFlagged_out <- dupeSummary(
data = BeeBDC::beesFlagged,
  # Should start with paste0(DataPath, "/Output/Report/"), instead of tempdir():
path = paste0(tempdir(), "/"),
# options are "ID","collectionInfo", or "both"
duplicatedBy = "collectionInfo", # I'm only running ID for the first lot because we might
# recover other info later
# The columns to generate completeness info from
completeness_cols = c("decimalLatitude",  "decimalLongitude",
                      "scientificName", "eventDate"),
# idColumns = c("gbifID", "occurrenceID", "recordId","id"),
# The columns to ADDITIONALLY consider when finding duplicates in collectionInfo
collectionCols = c("decimalLatitude", "decimalLongitude", "scientificName", "eventDate",
                   "recordedBy"),
# The columns to combine, one-by-one with the collectionCols
collectInfoColumns = c("catalogNumber", "otherCatalogNumbers"),
```

```
# Custom comparisons - as a list of columns to compare
# RAW custom comparisons do not use the character and number thresholds
CustomComparisonsRAW = dplyr::lst(c("catalogNumber", "institutionCode", "scientificName")),
# Other custom comparisons use the character and number thresholds
CustomComparisons = dplyr::lst(c("gbifID", "scientificName"),
                               c("occurrenceID", "scientificName"),
                               c("recordId", "scientificName"),
                               c("id", "scientificName")),
# The order in which you want to KEEP duplicated based on data source
# try unique(check_time$dataSource)
sourceOrder = c("CAES", "Gai", "Ecd","BMont", "BMin", "EPEL", "ASP", "KP", "EcoS", "EaCO",
                "FSCA", "Bal", "SMC", "Lic", "Arm",
                "USGS", "ALA", "GBIF","SCAN","iDigBio"),
# !!!!!! BELS > GeoLocate
# Set the complexity threshold for id letter and number length
# minimum number of characters when WITH the numberThreshold
characterThreshold = 2,
# minimum number of numbers when WITH the characterThreshold
numberThreshold = 3,
# Minimum number of numbers WITHOUT any characters
numberOnlyThreshold = 5)
```

`fileFinder` *Finds files within a directory*

Description

A function which can be used to find files within a user-defined directory based on a user-provided character string.

Usage

```
fileFinder(path, fileName)
```

Arguments

`path`	A directory as character. The directory to recursively search.
`fileName`	A character/regex string. The file name to find.

Value

Returns a directory to the most-recent file that matches the provide file. Using regex can greatly improve specificity. The function will also write into the console the file that it has found - it is worthwhile to check that this is the correct file to avoid complications down the line

Examples

```
# load dplyr
library(dplyr)

 # Make the RootPath to the tempdir for this example
  RootPath <- tempdir()

 # Load the example data
 data("beesRaw", package = "BeeBDC")

# Save and example dataset to the temp dir
  readr::write_csv(beesRaw, file = paste0(RootPath, "/beesRaw.csv"))

 # Now go find it!
fileFinder(path = RootPath, fileName = "beesRaw")
# more specifically the .csv version
fileFinder(path = RootPath, fileName = "beesRaw.csv")
```

`flagAbsent` *Flags occurrences that are marked as absent*

Description

Flags occurrences that are "ABSENT" for the occurrenceStatus (or some other user-specified) column.

Usage

```
flagAbsent(data = NULL, PresAbs = "occurrenceStatus")
```

Arguments

`data`	A data frame or tibble. Occurrence records as input.
`PresAbs`	Character. The column in which the function will find "ABSENT" or "PRESENT" records. Default = "occurrenceStatus"

Value

The input data with a new column called ".occurrenceAbsent" where FALSE == "ABSENT" records.

Examples

```
  # Bring in the data
data(beesRaw)
  # Run the function
beesRaw_out <- flagAbsent(data = beesRaw,
PresAbs = "occurrenceStatus")
  # See the result
table(beesRaw_out$.occurrenceAbsent, useNA = "always")
```

`flagLicense` *Flag license protected records*

Description

This function will search for strings that indicate a record is restricted in its use and will flag the restricted records.

Usage

```
flagLicense(data = NULL, strings_to_restrict = "all", excludeDataSource = NULL)
```

Arguments

`data`	A data frame or tibble. Occurrence records as input.
`strings_to_restrict`	A character vector. Should contain the strings used to detect protected records. Default = c("All Rights Reserved", "All rights reserved", "All rights reserved.", "ND", "Not for public")
`excludeDataSource`	Optional. A character vector. A vector of the data sources (dataSource) that will not be flagged as protected, even if they are. This is useful if you have a private dataset that should be listed as "All rights reserved" which you want to be ignored by this flag.

Value

Returns the data with a new column, .unLicensed, where FALSE = records that are protected by a license.

Examples

```
  # Read in the example data
data("beesRaw")
  # Run the function
beesRaw_out <- flagLicense(data = beesRaw,
                        strings_to_restrict = "all",
                        # DON'T flag if in the following data# source(s)
                        excludeDataSource = NULL)
```

`flagRecorder` *Loads, appends, and saves occurrence flag data*

Description

This function is used to save the flag data for your occurrence data as you run the BeeBDC script. It will read and append existing files, if asked to. Your flags should also be saved in the occurrence file itself automatically.

Usage

```
flagRecorder(
  data = NULL,
  outPath = NULL,
  fileName = NULL,
 idColumns = c("database_id", "id", "catalogNumber", "occurrenceID", "dataSource"),
  append = NULL,
  printSummary = FALSE
)
```

Arguments

`data`	A data frame or tibble. Occurrence records as input.
`outPath`	A character path. Where the file should be saved.
`fileName`	Character. The name of the file to be saved
`idColumns`	A character vector. The names of the columns that are to be kept along with the flag columns. These columns should be useful for identifying unique records with flags. Default = c("database_id", "id", "catalogNumber", "occurrenceID", "dataSource").
`append`	Logical. If TRUE, this will find and append an existing file generated by this function.
`printSummary`	Logical. If TRUE, print a `base::summary()` of all filter columns - i.e. those which tidyselect::starts_with(".")

Value

Saves a file with id and flag columns and returns this as an object.

Examples

```
# Load the example data
data("beesFlagged")

  # Run the function
  OutPath_Report <- tempdir()
flagFile <- flagRecorder(
```

```
  data = beesFlagged,
  outPath = paste(OutPath_Report, sep =""),
  fileName = paste0("flagsRecorded_", Sys.Date(), ".csv"),
  # These are the columns that will be kept along with the flags
  idColumns = c("database_id", "id", "catalogNumber", "occurrenceID", "dataSource"),
  # TRUE if you want to find a file from a previous part of the script to append to
  append = FALSE)
```

`flagSummaryTable` *Build a per-species summary for each and all flags*

Description

Takes a flagged dataset and returns the total number of fails (FALSE) per flag (columns starting with ".") and per species. It will ignore the .scientificName_empty and .invalidName columns as species are not assigned. Users may define the column to group the summary by. While it is intended to work with the scientificName column, users may select any grouping column (e.g., country).

Usage

```
flagSummaryTable(
  data = NULL,
  column = "scientificName",
  outPath = OutPath_Report,
  fileName = "flagTable.csv",
  percentImpacted = TRUE,
  percentThreshold = 0
)
```

Arguments

`data`	A data frame or tibble. The flagged dataset.
`column`	Character. The name of the column to group by and summarise the failed occurrences. Default = "scientificName".
`outPath`	A character path. The path to the directory in which the figure will be saved. Default = OutPath_Report. If is NULL then no file will be saved to the disk.
`fileName`	Character. The name of the file to be saved, ending in ".csv". Default = "flagTable.csv".
`percentImpacted`	Logical. If TRUE (the default), the program will write the percentage of species impacted and over the percentThreshold for each flagging column.
`percentThreshold`	Numeric. A number between 0 and 100 to indicate the percent of individuals (>; within each species) that is impacted by a flag, and to be included in the percentImpacted. Default = 0.

Value

A tibble with a column for each flag column (starting with ".") showing the number of failed (FALSE) occurrences per group. Also shows the (i) total number of records, (ii) total number of failed records, and (iii) the percentage of failed records.

Examples

```
# Load the toy flagged bee data
data("beesFlagged")

  # Run the function and build the flag table
flagTibble <- flagSummaryTable(data = beesFlagged,
                               column = "scientificName",
                               outPath = paste0(tempdir()),
                               fileName = "flagTable.csv")
```

---

formattedCombiner *Combine the formatted USGS data with the main dataset*

Description

Merges the Darwin Core version of the USGS dataset that was created using `USGS_formatter()` with the main dataset.

Usage

```
formattedCombiner(path, strings, existingOccurrences, existingEMLs)
```

Arguments

`path`	A directory as character. The directory to look in for the formatted USGS data.
`strings`	A regex string. The string to find the most-recent formatted USGS dataset.
`existingOccurrences`	A data frame. The existing occurrence dataset.
`existingEMLs`	An EML file. The existing EML data file to be appended.

Value

A list with the combined occurrence dataset and the updated EML file.

Examples

```
## Not run:
DataPath <- tempdir()
strings = c("USGS_DRO_flat_27-Apr-2022")
    # Combine the USGS data and the existing big dataset
Complete_data <- formattedCombiner(path = DataPath,
                                   strings = strings,
                                   # This should be the list-format with eml attached
                                   existingOccurrences = DataImp$Data_WebDL,
                                   existingEMLs = DataImp$eml_files)

## End(Not run)
```

GBIFissues *Flags records with GBIF issues*

Description

This function will flag records which are subject to a user-specified vector of GBIF issues.

Usage

```
GBIFissues(data = NULL, issueColumn = "issue", GBIFflags = NULL)
```

Arguments

data	A data frame or tibble. Occurrence records as input.
issueColumn	Character. The column in which to look for GBIF issues. Default = "issue".
GBIFflags	Character vector. The GBIF issues to flag. Users may choose their own vector of issues to flag or use a pre-set vector or vectors, including c("allDates", "allMetadata", "allObservations", "allSpatial", "allTaxo", or "all"). Default = c("COORDINATE_INVALID", "PRESUMED_NEGATED_LONGITUDE", "PRESUMED_NEGATED_LATITUDE", "COUNTRY_COORDINATE_MISMATCH", "ZERO_COORDINATE")

Value

Returns the data with a new column, ".GBIFflags", where FALSE = records with any of the provided GBIFflags.

Examples

```
# Import the example data
data(beesRaw)
# Run the function
beesRaw_Out <- GBIFissues(data = beesRaw,
   issueColumn = "issue",
   GBIFflags = c("COORDINATE_INVALID", "ZERO_COORDINATE"))
```

ggRichnessWrapper *ggplot2 extension for a Chao- and iNEXT-wrapper outputs*

Description

BeeBDC::ggRichnessWrapper() takes data output from BeeBDC::iNEXTwrapper() and BeeBDC::ChaoWrapper() to produce plots for multiple groups in one go. The plots can be multiple per page and across multiple pages.

Usage

```
ggRichnessWrapper(
  iNEXT_in = country_iNEXT,
  iChao_in = NULL,
  filterOut = NULL,
  type = 1,
  se = TRUE,
  facet.var = "None",
  color.var = "Order.q",
  grey = FALSE,
  legendPerPlot = FALSE,
  show_iNEXT = TRUE,
  showPercent = TRUE,
  ChaoColour = "#55AD9B",
  iNEXTcolour = "#FD9B63",
  Chao_estimate = "iChao1 (Chiu et al. 2014)",
  nrow = 3,
  ncol = 4,
  labels = NULL,
  fileName = "richnessPlots.pdf",
  outPath = tempdir(),
  base_width = 8.3,
  base_height = 11.7,
  dpi = 300,
  ...
)
```

Arguments

iNEXT_in	a list of iNEXT objects computed by BeeBDC::iNEXTwrapper().
iChao_in	a list of R data created BeeBDC::ChaoWrapper().
filterOut	Character. A list of sites/countries to exclude from plotting; for example, because sample size was inadequate. Default = NULL.

type	three types of plots: sample-size-based rarefaction/extrapolation curve (type = 1); sample completeness curve (type = 2); coverage-based rarefaction/extrapolation curve (type = 3). From iNEXT::ggiNEXT()
se	a logical variable to display confidence interval around the estimated sampling curve. From iNEXT::ggiNEXT()
facet.var	create a separate plot for each value of a specified variable: no separation (facet.var="None"); a separate plot for each diversity order (facet.var="Order.q"); a separate plot for each assemblage (facet.var="Assemblage"); a separate plot for each combination of order x assemblage (facet.var="Both"). From iNEXT::ggiNEXT()
color.var	create curves in different colors for values of a specified variable: all curves are in the same color (color.var="None"); use different colors for diversity orders (color.var="Order.q"); use different colors for sites (color.var="Assemblage"); use different colors for combinations of order x assemblage (color.var="Both"). From iNEXT::ggiNEXT()
grey	a logical variable to display grey and white ggplot2 theme. From iNEXT::ggiNEXT()
legendPerPlot	Logical. If TRUE, remove the legend from each plot. Default = FALSE
show_iNEXT	Logical. If TRUE, show the estimate and 95% CIs if specified for iNEXT. Default = TRUE.
showPercent	Logical. If TRUE, show the prrcentage increases. Default = TRUE.
ChaoColour	Character. The to be used to graph Chao estimates (95% confidence intervals are shown with reduced opacity). Default = "#55AD9B".
iNEXTcolour	Character. The to be used to graph iNEXT estimates (95% confidence intervals are shown with reduced opacity). Default = "#FD9B63".
Chao_estimate	Character. The name of the Chao estimate to use from those calculated in SpadeR::ChaoSpecies(). The options are "Homogeneous Model","Homogeneous (MLE)", "Chao1 (Chao, 1984)","Chao1-bc","iChao1 (Chiu et al. 2014)","ACE (Chao & Lee, 1992)", "ACE-1 (Chao & Lee, 1992)","1st order jackknife","2nd order jackknife". Default = "iChao1 (Chiu et al. 2014)".
nrow	Numeric. The number of rows per figure. Figures (that don't fit in the nrow*ncol grid) will be saved into additional files. Default = 3.
ncol	Numeric. The number of columns per figure. Figures (that don't fit in the nrow*ncol grid) will be saved into additional files. Default = 4.
labels	Character. The labels for each sub-plot (a, b, c, ...). The default is NULL, which will provide labels a-z as required.
fileName	Character. Prefix to the output files. Default = "richnessPlots.pdf". Changing the suffix is a convenient way to alter the file format.
outPath	Character. The fodler in which to save the plots. Default = tempdir()
base_width	Numeric. The width, in inches, to save the plot. Default = 8.3.
base_height	Numeric. The height, in inches, to save the plot. Default = 11.7.
dpi	Numeric. Plot resolution. Also accepts a string input: "retina" (320), "print" (300), or "screen" (72). Applies only to raster output types. Default = 300.
...	other arguments passed on to methods. Not currently used. From iNEXT::ggiNEXT()

Value

Saves pdf objects and returns a summary table for all levels

Examples

```
## Not run:

data(beesCountrySubset)

   # Transform data for iNEXT
 data_nextWrapper <- beesCountrySubset %>%
   dplyr::group_by(scientificName, country_suggested) %>%
   dplyr::count()

 # Calculate iNEXT with the wrapper function
 output_iNEXTwrapper <- BeeBDC::iNEXTwrapper(data = data_nextWrapper,
                                             variableColumn = "country_suggested",
                                             valueColumn = "n",
                                             mc.cores = 1)

 # Transform data for iChao
data_iChao <- beesCountrySubset %>%
  dplyr::group_by(scientificName, country_suggested) %>%
  dplyr::count() %>%
  dplyr::select(scientificName, country_suggested, n) %>%
  tidyr::pivot_wider(names_from = country_suggested,
                     values_from = n,
                     values_fill = 0) %>%
  ## Create the rownames
  tibble::column_to_rownames("scientificName") %>%
  dplyr::tibble()

  # Run the wrapper function
output_iChaowrapper <- BeeBDC::ChaoWrapper(data = data_iChao,
                                             datatype = "abundance",
                                             k = 10,
                                             conf = 0.95,
                                             mc.cores = 1)

   # Make the plots!
plot_summary <- BeeBDC::ggRichnessWrapper(
iNEXT_in = output_iNEXTwrapper,
iChao_in = output_iChaowrapper,
nrow = 2,
ncol = 2,
labels = NULL,
fileName = "speciesRichnessPlots.pdf",
outPath = tempdir(),
base_width = 8.3,
base_height = 11.7,
dpi = 300)
```

```
## End(Not run)
```

---

harmoniseR *Harmonise taxonomy of occurrence data*

Description

Uses BeeBDC-formatted taxonomy data to harmonise occurrence records and flag those that do not match the taxonomy harmoniseR() prefers to use the names_clean columns that is generated by bdc::bdc_clean_names(). While this is not required, you may find better results by running that function on your dataset first. It is possible to download taxonomy file for other taxa using taxadbToBeeBDC() which can download taxonomies from ITIS, GBIF, and more. You could also match the format of the beesTaxonomy() file.

Usage

```
harmoniseR(
  data = NULL,
  path = NULL,
  taxonomy = BeeBDC::beesTaxonomy(),
  speciesColumn = "scientificName",
  rm_names_clean = TRUE,
  checkVerbatim = FALSE,
  relaxAmbiguous = FALSE,
  matchHigherTaxonomy = TRUE,
  stepSize = 1e+06,
  mc.cores = 1
)
```

Arguments

data	A data frame or tibble. Occurrence records as input.
path	A directory as character. The path to a folder that the output can be saved.
taxonomy	A data frame or tibble. The taxonomy file to use. Default = beesTaxonomy(); for other taxa see first taxadbToBeeBDC().
speciesColumn	Character. The name of the column containing species names. Default = "scientificName".
rm_names_clean	Logical. If TRUE then the names_clean column will be removed at the end of this function to help reduce confusion about this column later. Default = TRUE
checkVerbatim	Logical. If TRUE then the verbatimScientificName will be checked as well for species matches. This matching will ONLY be done after harmoniseR has failed for the other name columns. NOTE: this column is *not* first run through bdc::bdc_clean_names. Default = FALSE

`relaxAmbiguous` Logical. If TRUE, then ambiguous names will be returned as "TRUE" for .invalidName. However, they will be listed as "ambiguouslyMatchedName" under the "scientificNameAuthorship" column and the speciesColumn will NOT include the authority information. Hence, the names can be used but in should remain clear in the data that they must be used with caution. Default = FALSE.

`matchHigherTaxonomy` Logical. If TRUE, then BeeBDC will try to match higher taxonomies (e.g., family names) using genera and such. This will work better if you have run the data through `bdc::bdc_clean_names` beforehand. The function will update the taxonRank column, if it is otherwise empty. Default = TRUE.

`stepSize` Numeric. The number of occurrences to process in each chunk. Default = 1000000.

`mc.cores` Numeric. If > 1, the function will run in parallel using mclapply using the number of cores specified. If = 1 then it will be run using a serial loop. NOTE: Windows machines must use a value of 1 (see ?parallel::mclapply). Additionally, be aware that each thread can use large chunks of memory. Default = 1.

Value

The occurrences are returned with update taxonomy columns, including: scientificName, species, family, subfamily, genus, subgenus, specificEpithet, infraspecificEpithet, and scientificNameAuthorship. A new column, .invalidName, is also added and is FALSE when the occurrence's name did not match the supplied taxonomy.

See Also

`taxadbToBeeBDC()` to download any taxonomy (of any taxa or of bees) and `beesTaxonomy()` for the bee taxonomy download.

Examples

```
# load in the test dataset
system.file("extdata", "testTaxonomy.rda", package="BeeBDC") |> load()

# See also
?BeeBDC::taxadbToBeeBDC()

beesRaw_out <- BeeBDC::harmoniseR(
  #The path to a folder that the output can be saved
path = tempdir(),
# The formatted taxonomy file
taxonomy = testTaxonomy,
data = BeeBDC::beesFlagged,
speciesColumn = "scientificName")
table(beesRaw_out$.invalidName, useNA = "always")
```

idMatchR *Attempt to match database_ids from a prior run*

Description

This function attempts to match database_ids from a prior bdc or BeeBDC run in order to keep this column somewhat consistent between iterations. However, not all records contain sufficient information for this to work flawlessly.

Usage

```
idMatchR(
  currentData = NULL,
  priorData = NULL,
  matchBy = NULL,
  completeness_cols = NULL,
  excludeDataset = NULL
)
```

Arguments

currentData	A data frame or tibble. The NEW occurrence records as input.
priorData	A data frame or tibble. The PRIOR occurrence records as input.
matchBy	A list of character vectors Should contain the columns to iteratively compare.
completeness_cols	A character vector. The columns to check for completeness, arrange, and assign the relevant prior database_id.
excludeDataset	A character vector. The dataSources that are to be excluded from data matching. These should be static dataSources from minor providers.

Value

The input data frame returned with an updated database_id column that shows the database_ids as in priorData where they could be matched. Additionally, a columnd called idContinuity is returned where TRUE indicates a match to a prior database_id and FALSE indicates that a new database_id was assigned.

Examples

```
# Get the example data
data("beesRaw", package = "BeeBDC")
# Which datasets are static and should be excluded from matching?
excludeDataset <- c("BMin", "BMont", "CAES", "EaCO", "Ecd", "EcoS",
                    "Gai", "KP", "EPEL", "USGS", "FSCA", "SMC", "Bal", "Lic", "Arm", "BBD",
                      "MEPB")
  # Match the data to itself just as an example of running the code.
beesRaw_out <- idMatchR(
```

```
  currentData = beesRaw,
  priorData = beesRaw,
  # First matches will be given preference over later ones
  matchBy = dplyr::lst(c("gbifID"),
                       c("catalogNumber", "institutionCode", "dataSource"),
                       c("occurrenceID", "dataSource"),
                       c("recordId", "dataSource"),
                       c("id"),
                       c("catalogNumber", "institutionCode")),
 # You can exclude datasets from prior by matching their prefixs - before first underscore:
  excludeDataset = excludeDataset)
```

`importOccurrences` *Imports the most-recent repoMerge data*

Description

Looks for and imports the most-recent version of the occurrence data created by the `repoMerge()` function.

Usage

```
importOccurrences(path = path, fileName = "^BeeData_")
```

Arguments

`path`	A directory as a character. The directory to recursively look in for the above data.
`fileName`	Character. A String of text to look for the most-recent dataset. Default = "^BeeData_". Find faults by modifying `fileFinder()` and logic-checking the file that's found.

Value

A list with a data frame of merged occurrence records, "Data_WebDL", and a list of EML files contained in "eml_files".

Examples

```
## Not run:
DataImp <- importOccurrences(path = DataPath)

## End(Not run)
```

`iNEXTwrapper` *Parallel estimation of species richness in a community using iNEXT*

Description

A wrapper for `iNEXT::iNEXT()` to interpolate and extrapolate Hill numbers with order q (rarify species richness). The wrapper has the ability to estimate species richness for multiple sites (or countries) at once and to do this using multiple cores.

Usage

```
iNEXTwrapper(
  data = NULL,
  variableColumn = "groupVariable",
  valueColumn = "n",
  q = 0,
  datatype = "abundance",
  conf = 0.95,
  se = TRUE,
  nboot = 50,
  size = NULL,
  endpoint = NULL,
  knots = 40,
  mc.cores = 1
)
```

Arguments

`data`	A data frame or tibble. A data frame containing "abundance"-type data per variable (population, country, species...) in columns.
`variableColumn`	Character. The column to be used to group the data. Probably "country" or "site". Default = "groupVariable".
`valueColumn`	Character. The column containing the count data. Defualt = "n".
`q`	a number or vector specifying the richness order(s) of Hill numbers.
`datatype`	data type of input data: individual-based abundance data (datatype = "abundance"), sampling-unit-based incidence frequencies data (datatype = "incidence_freq") or species by sampling-units incidence matrix (datatype = "incidence_raw").
`conf`	a positive number < 1 specifying the level of confidence interval; default is 0.95.
`se`	a logical variable to calculate the bootstrap standard error and conf confidence interval.
`nboot`	an integer specifying the number of replications; default is 50.
`size`	an integer vector of sample sizes (number of individuals or sampling units) for which richness estimates will be computed. If NULL, then richness estimates will be computed for those sample sizes determined by the specified/default endpoint and knots.

endpoint — an integer specifying the sample size that is the endpoint for rarefaction/extrapolation. If NULL, then endpoint = double the reference sample size.

knots — an integer specifying the number of equally-spaced knots (say K, default is 40) between size 1 and the endpoint; each knot represents a particular sample size for which richness estimate will be calculated. If the endpoint is smaller than the reference sample size, then iNEXT() computes only the rarefaction esimates for approximately K evenly spaced knots. If the endpoint is larger than the reference sample size, then iNEXT() computes rarefaction estimates for approximately K/2 evenly spaced knots between sample size 1 and the reference sample size, and computes extrapolation estimates for approximately K/2 evenly spaced knots between the reference sample size and the endpoint.

mc.cores — Numeric. If > 1, the function will run in parallel using mclapply using the number of cores specified. If = 1 then it will be run using a serial loop. NOTE: Windows machines must use a value of 1 (see ?parallel::mclapply). Additionally, be aware that each thread can use large chunks of memory. Default = 1.

Value

Returns a list containing two tibbles. The first is a tibble that concatenates the outputs from the basic data and rare species information in columns per input variable (column). The second is a tibble that concatenates the various species richness estimates, with input variables in chunks of rows. Additionally a console output will list the variables (columns) that lacked sufficient data to be analysed.

See Also

iNEXT::iNEXT()

Examples

```
## Not run:
 # Read in some example data and use [BeeBDC::richnessPrepR()] to create the example input data
 #' data(beesCountrySubset)

estimateDataExample <- BeeBDC::richnessPrepR(
  data = beesCountrySubset,
  # Download the taxonomy
  taxonomyFile = BeeBDC::beesTaxonomy(),
  # Download the checklist
  checklistFile = BeeBDC::beesChecklist(),
  curveFunction = function(x) (228.7531 * x * x^-log(12.1593)),
  sampleSize = 10000,
  countryColumn = "country_suggested",
  limitGlobal = NULL,
  outPath = tempdir()
)

  # In this function we can directly feed in estimateDataExample$site_speciesCounts
 iNextOut <- iNEXTwrapper(data = estimateDataExample$site_speciesCounts,
                          variableColumn = "country_suggested",
```

```
                          valueColumn = "n",
                          q = 0,
                          datatype = "abundance",
                          conf = 0.95,
                          se = TRUE,
                          nboot = 50,
                          size = NULL,
                          endpoint = NULL,
                          knots = 40,
                          mc.cores = 1)

## End(Not run)
```

`interactiveMapR` *Creates interactive html maps for species*

Description

Uses the occurrence data (preferably uncleaned) and outputs interactive .html maps that can be opened in your browser to a specific directory. The maps can highlight if an occurrence has passed all filtering (.summary == TRUE) or failed at least one filter (.summary == FALSE). This can be modified by first running `summaryFun()` to set the columns that you want to be highlighted. It can also highlight occurrences flagged as expert-identified or country outliers.

Usage

```
interactiveMapR(
  data = NULL,
  outPath = NULL,
  lon = "decimalLongitude",
  lat = "decimalLatitude",
  speciesColumn = "scientificName",
  speciesList = "ALL",
  countryList = NULL,
  jitterValue = NULL,
  onlySummary = TRUE,
  overWrite = TRUE,
  customColumn1 = NULL,
  customColumn2 = NULL,
  TrueAlwaysTop = FALSE,
  excludeSpecies = c("Apis mellifera Linnaeus, 1758"),
  pointColours = c("blue", "darkred", "#ff7f00", "black"),
  returnPlot = FALSE
)
```

Arguments

`data`	A data frame or tibble. Occurrence records to use as input.

outPath	A directory as character. Directory where to save output maps.
lon	Character. The name of the longitude column. Default = "decimalLongitude".
lat	Character. The name of the latitude column. Default = "decimalLatitude".
speciesColumn	Character. The name of the column containing species names (or another factor) to build individual maps from. Default = "scientificName".
speciesList	A character vector. Should contain species names as they appear in the speciesColumn to make maps of. User can also specify "ALL" in order to make maps of all species present in the data. Hence, a user may first filter their data and then use "ALL". Default = "ALL".
countryList	A character vector. Country names to map, or NULL for to map ALL countries.
jitterValue	Numeric. The amount, in decimal degrees, to jitter the map points by - this is important for separating stacked points with the same coordinates.
onlySummary	Logical. If TRUE, the function will not look to plot country or expert-identified outliers in different colours.
overWrite	Logical. If TRUE, the function will overwrite existing files in the provided directory that have the same name. Default = TRUE.
customColumn1	Character. Allows the user to report on a column of their choosing in the output.
customColumn2	Character. Allows the user to report on a column of their choosing in the output.
TrueAlwaysTop	If TRUE, the quality (TRUE) points will always be displayed on top of other points. If FALSE, then whichever layer was turned on most-recently will be displayed on top.
excludeSpecies	Character. A character vector of species names to exclude, especially if they are so large as to become a problem when making maps. For bee data, for example, "Apis mellifera Linnaeus, 1758" has too many data points and is default excluded.
pointColours	A character vector of colours. In order provide colour for TRUE, FALSE, countryOutlier, and customOutlier. Default = c("blue", "darkred","#ff7f00", "black").
returnPlot	Logical. If TRUE, return the plot to the environment. Default = FALSE.

Value

Exports .html interactive maps of bee occurrences to the specified directory.

Examples

```
if(require("htmlwidgets") && require("leaflet")){

OutPath_Figures <- tempdir()

interactiveMapR(
# occurrence data - start with entire dataset, filter down to these species
data = BeeBDC::bees3sp, # %>%
  # Select only those species in the 100 randomly chosen
  # dplyr::filter(scientificName %in% beeData_interactive$scientificName),
```

```
  # Select only one species to map
  # dplyr::filter(scientificName %in% "Agapostemon sericeus (Forster, 1771)"),
# Directory where to save files
outPath = paste0(OutPath_Figures, "/interactiveMaps_TEST"),
# lat long columns
lon = "decimalLongitude",
lat = "decimalLatitude",
# Occurrence dataset column with species names
speciesColumn = "scientificName",
# Which species to map - a character vector of names or "ALL"
# Note: "ALL" is defined AFTER filtering for country
speciesList = "ALL",
# studyArea
countryList = NULL,
# Point jitter to see stacked points - jitters an amount in decimal degrees
jitterValue = 0.01,
# If TRUE, it will only map the .summary column. Otherwise, it will map .summary
# which will be over-written by countryOutliers and manualOutliers
onlySummary = TRUE,
excludeSpecies = c("Apis mellifera Linnaeus, 1758"),
overWrite = TRUE,
  # Colours for points which are flagged as TRUE, FALSE, countryOutlier, and customOutlier
pointColours = c("blue", "darkred","#ff7f00", "black")
)

} # END if require
```

---

jbd_CfC_chunker *Get country names from coordinates*

Description

Because the bdc::bdc_country_from_coordinates() function is very RAM-intensive, this wrapper allows a user to specify chunk-sizes and only analyse a small portion of the occurrence data at a time. The prefix jbd_ is used to highlight the difference between this function and the original bdc::bdc_country_from_coordinates().

Usage

```
jbd_CfC_chunker(
  data = NULL,
  lat = "decimalLatitude",
  lon = "decimalLongitude",
  country = "country",
  stepSize = 1e+06,
  chunkStart = 1,
  scale = "medium",
  path = tempdir(),
  mc.cores = 1
)
```

Arguments

data	A data frame or tibble. Occurrence records to use as input.
lat	Character. The name of the column to use as latitude. Default = "decimalLatitude".
lon	Character. The name of the column to use as longitude. Default = "decimalLongitude".
country	Character. The name of the column containing country names. Default = "country.
stepSize	Numeric. The number of occurrences to process in each chunk. Default = 1000000.
chunkStart	Numeric. The chunk number to start from. This can be > 1 when you need to restart the function from a certain chunk. For example, can be used if R failed unexpectedly.
scale	Passed to rnaturalearth's ne_countries(). Scale of map to return, one of 110, 50, 10 or 'small', 'medium', 'large'. Default = "large".
path	Character. The directory path to a folder in which to save the running countrylist csv file.
mc.cores	Numeric. If > 1, the function will run in parallel using mclapply using the number of cores specified. If = 1 then it will be run using a serial loop. NOTE: Windows machines must use a value of 1 (see ?parallel::mclapply). Additionally, be aware that each thread can use large chunks of memory. Default = 1.

Value

A data frame containing database_ids and a country column that needs to be re-merged with the data input.

Examples

```
if(requireNamespace("rnaturalearthdata", quietly = TRUE)){
library("dplyr")
data(beesFlagged)
HomePath = tempdir()
# Tibble of common issues in country names and their replacements
commonProblems <- dplyr::tibble(problem = c('U.S.A.', 'US','USA','usa','UNITED STATES',
'United States','U.S.A','MX','CA','Bras.','Braz.','Brasil','CNMI','USA TERRITORY: PUERTO RICO'),
                            fix = c('United States of America','United States of America',
                                  'United States of America','United States of America',
                                  'United States of America','United States of America',
                           'United States of America','Mexico','Canada','Brazil','Brazil',
                                  'Brazil','Northern Mariana Islands','Puerto Rico'))

beesFlagged <- beesFlagged %>%
      # Replace a name to test
   dplyr::mutate(country = stringr::str_replace_all(country, "Brazil", "Brasil"))

beesFlagged_out <- countryNameCleanR(
  data = beesFlagged,
```

```
    commonProblems = commonProblems)

  suppressWarnings(
    countryOutput <- jbd_CfC_chunker(data = beesFlagged_out,
                                     lat = "decimalLatitude",
                                     lon = "decimalLongitude",
                                     country = "country",
                                     # How many rows to process at a time
                                     stepSize = 1000000,
                                     # Start row
                                     chunkStart = 1,
                                     path = HomePath,
                                     scale = "medium"),
    classes = "warning")


  # Left join these datasets
  beesFlagged_out <- left_join(beesFlagged_out, countryOutput, by = "database_id")  %>%
    # merge the two country name columns into the "country" column
    dplyr::mutate(country = dplyr::coalesce(country.x, country.y)) %>%
    # remove the now redundant country columns
    dplyr::select(!c(country.x, country.y)) %>%
    # put the column back
    dplyr::relocate(country) %>%
    # Remove duplicates if they arose!
    dplyr::distinct()

  # Remove illegal characters
  beesFlagged_out$country <- beesFlagged_out$country %>%
    stringr::str_replace(., pattern = paste("\\[", "\\]", "\\?",
                                            sep=  "|"), replacement = "")
  } # END if require
```

`jbd_coordCountryInconsistent` *Flags coordinates that are inconsistent with the stated country name*

Description

Compares stated country name in an occurrence record with record's coordinates using rnaturalearth data. The prefix, jbd_ is meant to distinguish this function from the original `bdc::bdc_coordinates_country_inconsister` This functions will preferably use the countryCode and country_suggested columns generated by `bdc::bdc_country_standardized()`; please run it on your dataset prior to running this function.

Usage

```
jbd_coordCountryInconsistent(
  data = NULL,
  lon = "decimalLongitude",
```

```
  lat = "decimalLatitude",
  scale = 50,
  pointBuffer = 0.01,
  stepSize = 1e+06,
  mc.cores = 1
)
```

Arguments

data	A data frame or tibble. Occurrence records as input.
lon	Character. The name of the column to use as longitude. Default = "decimalLongitude".
lat	Character. The name of the column to use as latitude. Default = "decimalLatitude".
scale	Numeric or character. To be passed to rnaturalearth::ne_countries()'s scale. Scale of map to return, one of 110, 50, 10 or "small", "medium", "large". Smaller values return higher-resolution maps.
pointBuffer	Numeric. Amount to buffer points, in decimal degrees. If the point is outside of a country, but within this point buffer, it will not be flagged. Default = 0.01.
stepSize	Numeric. The number of occurrences to process in each chunk. Default = 1000000.
mc.cores	Numeric. If > 1, the st_intersects function will run in parallel using mclapply using the number of cores specified. If = 1 then it will be run using a serial loop. NOTE: Windows machines must use a value of 1 (see ?parallel::mclapply). Additionally, be aware that each thread can use large chunks of memory. Default = 1.

Value

The input occurrence data with a new column, .coordinates_country_inconsistent

Examples

```
if(requireNamespace("rnaturalearthdata", quietly = TRUE)){
beesRaw_out <- jbd_coordCountryInconsistent(
  data = BeeBDC::beesRaw,
  lon = "decimalLongitude",
  lat = "decimalLatitude",
  scale = 50,
  pointBuffer = 0.01)
} # END if require
```

`jbd_coordinates_precision` *Flags coordinates for imprecision*

Description

This function flags occurrences where BOTH latitude and longitude values are rounded. This contrasts with the original function, bdc::bdc_coordinates_precision() that will flag occurrences where only one of latitude OR longitude are rounded. The BeeBDC approach saves occurrences that may have had terminal zeros rounded in one coordinate column.

Usage

```
jbd_coordinates_precision(
  data,
  lat = "decimalLatitude",
  lon = "decimalLongitude",
  ndec = NULL,
  quieter = FALSE
)
```

Arguments

`data`	A data frame or tibble. Occurrence records as input.
`lat`	Character. The name of the column to use as latitude. Default = "decimalLatitude".
`lon`	Character. The name of the column to use as longitude. Default = "decimalLongitude".
`ndec`	Numeric. The number of decimal places to flag in decimal degrees. For example, argument value of 2 would flag occurrences with nothing in the hundredths place (0.0x).
`quieter`	Logical. If TRUE, the functino will run a little quieter. Default = FALSE.

Value

Returns the input data frame with a new column, .rou, where FALSE indicates occurrences that failed the test.

Examples

```
beesRaw_out <- jbd_coordinates_precision(
  data = BeeBDC::beesRaw,
  lon = "decimalLongitude",
  lat = "decimalLatitude",
    # number of decimals to be tested
  ndec = 2
)
table(beesRaw_out$.rou, useNA = "always")
```

jbd_coordinates_transposed *Identify transposed geographic coordinates*

Description

This function flags and corrects records when latitude and longitude appear to be transposed. This function will preferably use the countryCode column generated by `bdc::bdc_country_standardized()`.

Usage

```
jbd_coordinates_transposed(
  data,
  idcol = "database_id",
  sci_names = "scientificName",
  lat = "decimalLatitude",
  lon = "decimalLongitude",
  country = "country",
  countryCode = "countryCode",
  border_buffer = 0.2,
  save_outputs = FALSE,
  fileName = NULL,
  scale = "large",
  path = NULL,
  mc.cores = 1
)
```

Arguments

data	A data frame or tibble. Containing a unique identifier for each record, geographical coordinates, and country names. Coordinates must be expressed in decimal degrees and WGS84.
idcol	A character string. The column name with a unique record identifier. Default = "database_id".
sci_names	A character string. The column name with species' scientific names. Default = "scientificName".
lat	A character string. The column name with latitudes. Coordinates must be expressed in decimal degrees and WGS84. Default = "decimalLatitude".
lon	A character string. The column name with longitudes. Coordinates must be expressed in decimal degrees and WGS84. Default = "decimalLongitude".
country	A character string. The column name with the country assignment of each occurrence record. Default = "country".
countryCode	A character string. The column name containing an ISO-2 country code for each record.

border_buffer	Numeric. Must have value greater than or equal to 0. A distance in decimal degrees used to created a buffer around each 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. Indicates if a table containing transposed coordinates should be saved for further inspection. Default = FALSE.
fileName	A character string. The out file's name.
scale	Passed to rnaturalearth's ne_countries(). Scale of map to return, one of 110, 50, 10 or 'small', 'medium', 'large'. Default = "large".
path	A character string. A path as a character vector for where to create the directories and save the figures. If no path is provided (the default), the directories will be created using here::here().
mc.cores	Numeric. If > 1, the jbd_correct_coordinates function will run in parallel using mclapply using the number of cores specified. If = 1 then it will be run using a serial loop. NOTE: Windows machines must use a value of 1 (see ?parallel::mclapply). Additionally, be aware that each thread can use large chunks of memory. Default = 1.#'

Details

This test identifies transposed coordinates based on mismatches between the country provided for a record and the record's latitude and longitude 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::bdc_country_standardized.

Value

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

Examples

```
if(require("rnaturalearthdata") && require("bdc")){
database_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(
  database_id, scientificName, decimalLatitude,
```

```
  decimalLongitude, country
)

# Get country codes
x <- bdc::bdc_country_standardized(data = x, country = "country")

jbd_coordinates_transposed(
  data = x,
  idcol = "database_id",
  sci_names = "scientificName",
  lat = "decimalLatitude",
  lon = "decimalLongitude",
  country = "country_suggested",
  countryCode = "countryCode",
  border_buffer = 0.2,
  save_outputs = FALSE,
  scale = "medium"
)
} # END if require
 # RND donttest
```

`jbd_create_figures` *Create figures reporting the results of the bdc/BeeBDC packages*

Description

Creates figures (i.e., bar plots, maps, and histograms) reporting the results of data quality tests implemented the bdc and BeeBDC packages. Works like `bdc::bdc_create_figures()`, but it allows the user to specify a save path.

Usage

```
jbd_create_figures(
  data,
  path = OutPath_Figures,
  database_id = "database_id",
  workflow_step = NULL,
  save_figures = FALSE
)
```

Arguments

`data`	A data frame or tibble. Needs to contain the results of data quality tests; that is, columns starting with ".".
`path`	A character directory. The path to a directory in which to save the figures. Default = OutPath_Figures.
`database_id`	A character string. The column name with a unique record identifier. Default = "database_id".

workflow_step A character string. Name of the workflow step. Options available are "prefilter", "space", and "time".

save_figures Logical. Indicates if the figures should be saved for further inspection or use. Default = FALSE.

Details

This function creates figures based on the results of data quality tests. 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 tests implemented in one module of bdc/BeeBDC. When save_figures = TRUE, figures are also saved locally in a .png format.

Examples

```
if(require("bdc")){

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 <-
jbd_create_figures(
  data = x,
  database_id = "database_id",
  workflow_step = "prefilter",
  save_figures = FALSE
)

} # End if require
 # End dont test
```

jbd_Ctrans_chunker *Wraps jbd_coordinates_transposed to identify and fix transposed occurrences*

Description

Because the jbd_coordinates_transposed() function is very RAM-intensive, this wrapper allows a user to specify chunk-sizes and only analyse a small portion of the occurrence data at a time. The prefix jbd_ is used to highlight the difference between this function and the original bdc::bdc_coordinates_transposed(). This function will preferably use the countryCode column generated by bdc::bdc_country_standardized().

Usage

```
jbd_Ctrans_chunker(
  data = NULL,
  lat = "decimalLatitude",
  lon = "decimalLongitude",
  idcol = "databse_id",
  country = "country_suggested",
  countryCode = "countryCode",
  sci_names = "scientificName",
  border_buffer = 0.2,
  save_outputs = TRUE,
  stepSize = 1e+06,
  chunkStart = 1,
  progressiveSave = TRUE,
  path = tempdir(),
  append = TRUE,
  scale = "large",
  mc.cores = 1
)
```

Arguments

data	A data frame or tibble. Occurrence records as input.
lat	Character. The column with latitude in decimal degrees. Default = "decimalLatitude".
lon	Character. The column with longitude in decimal degrees. Default = "decimalLongitude".
idcol	Character. The column name with a unique record identifier. Default = "database_id".
country	Character. The name of the column containing country names. Default = "country".
countryCode	Character. Identifies the column containing ISO-2 country codes Default = "countryCode".

sci_names	Character. The column containing scientific names. Default = "scientificName".
border_buffer	Numeric. The buffer, in decimal degrees, around points to help match them to countries. Default = 0.2 (~22 km at equator).
save_outputs	Logical. If TRUE, transposed occurrences will be saved to their own file.
stepSize	Numeric. The number of occurrences to process in each chunk. Default = 1000000.
chunkStart	Numeric. The chunk number to start from. This can be > 1 when you need to restart the function from a certain chunk; for example if R failed unexpectedly.
progressiveSave	Logical. If TRUE then the country output list will be saved between each iteration so that append can be used if the function is stopped part way through.
path	Character. The path to a file in which to save the 01_coordinates_transposed_ output.
append	Logical. If TRUE, the function will look to append an existing file.
scale	Passed to rnaturalearth's ne_countries(). Scale of map to return, one of 110, 50, 10 or 'small', 'medium', 'large'. Default = "large".
mc.cores	Numeric. If > 1, the jbd_correct_coordinates function will run in parallel using mclapply using the number of cores specified. If = 1 then it will be run using a serial loop. NOTE: Windows machines must use a value of 1 (see ?parallel::mclapply). Additionally, be aware that each thread can use large chunks of memory. Default = 1.#'

Value

Returns the input data frame with a new column, coordinates_transposed, where FALSE = columns that had coordinates transposed.

Examples

```
if(requireNamespace("rnaturalearthdata", quietly = TRUE)){
library(dplyr)
  # Import and prepare the data
data(beesFlagged)
beesFlagged <- beesFlagged %>% dplyr::select(!c(.val, .sea)) %>%
  # Cut down the dataset to un example quicker
dplyr::filter(dplyr::row_number() %in% 1:20)
  # Run the function
beesFlagged_out <- jbd_Ctrans_chunker(
# bdc_coordinates_transposed inputs
data = beesFlagged,
idcol = "database_id",
lat = "decimalLatitude",
lon = "decimalLongitude",
country = "country_suggested",
countryCode = "countryCode",
# in decimal degrees (~22 km at the equator)
border_buffer = 1,
save_outputs = FALSE,
```

```
sci_names = "scientificName",
# chunker inputs
# How many rows to process at a time
stepSize = 1000000,
# Start row
chunkStart = 1,
# Progressively save the output between each iteration?
progressiveSave = FALSE,
path = tempdir(),
# If FALSE it may overwrite existing dataset
append = FALSE,
  # Users should select scale = "large" as it is more thoroughly tested
scale = "medium",
mc.cores = 1
)
table(beesFlagged_out$coordinates_transposed, useNA = "always")
} # END if require
```

`manualOutlierFindeR` *Finds outliers, and their duplicates, as determined by experts*

Description

Uses expert-identified outliers with source spreadsheets that may be edited by users. The function will also use the duplicates file made using `dupeSummary()` to identify duplicates of the expert-identified outliers and flag those as well. The function will add a flagging column called `.expertOutlier` where records that are FALSE are the expert outliers.

Usage

```
manualOutlierFindeR(
  data = NULL,
  DataPath = NULL,
  PaigeOutliersName = "removedBecauseDeterminedOutlier.csv",
  newOutliersName = "All_outliers_ANB.xlsx",
  ColombiaOutliers_all = "All_Colombian_OutlierIDs.csv",
  duplicates = NULL,
  NearTRUE = NULL,
  NearTRUE_threshold = 5
)
```

Arguments

`data`	A data frame or tibble. Occurrence records as input.
`DataPath`	A character path to the directory that contains the outlier spreadsheets.
`PaigeOutliersName`	A character patch. Should lead to outlier spreadsheet from Paige Chesshire (csv file).

`newOutliersName`
: A character path. Should lead to appropriate outlier spreadsheet (xlsx file).

`ColombiaOutliers_all`
: A character path. Should lead to spreadsheet of bee outliers from Colombia (csv file).

`duplicates`
: A data frame or tibble. The duplicate file produced by `dupeSummary()`.

`NearTRUE`
: Optional. A character file name to the csv file. If you want to remove expert outliers that are too close to TRUE points, use the name of the NearTRUE.csv. Note: This implementation is only basic for now unless there is a greater need in the future.

`NearTRUE_threshold`
: Numeric. The threshold (in km) for the distance to TRUE points to keep expert outliers.

Value

Returns the data with a new column, `.expertOutlier` where records that are FALSE are the expert outliers.

Examples

```
## Not run:
  # Read example data
  data(beesFlagged)
# Read in the most-recent duplicates file as well
if(!exists("duplicates")){
  duplicates <- fileFinder(path = DataPath,
                           fileName = "duplicateRun_") %>%
    readr::read_csv()}
# identify the outliers and get a list of their database_ids
beesFlagged_out <- manualOutlierFindeR(
  data = beesFlagged,
  DataPath = DataPath,
  PaigeOutliersName = "removedBecauseDeterminedOutlier.csv",
  newOutliersName = "^All_outliers_ANB_14March.xlsx",
  ColombiaOutliers_all = "All_Colombian_OutlierIDs.csv",
  duplicates = duplicates)

## End(Not run)
```

PaigeIntegrater *Integrate manually-cleaned data from Paige Chesshire*

Description

Replaces publicly available data with data that has been manually cleaned and error-corrected for use in the paper Chesshire, P. R., Fischer, E. E., Dowdy, N. J., Griswold, T., Hughes, A. C., Orr, M. J., . . . McCabe, L. M. (In Press). Completeness analysis for over 3000 United States bee species identifies persistent data gaps. Ecography.

Usage

```
PaigeIntegrater(db_standardized = NULL, PaigeNAm = NULL, columnStrings = NULL)
```

Arguments

db_standardized	A data frame or tibble. Occurrence records as input.
PaigeNAm	A data frame or tibble. The Paige Chesshire dataset.
columnStrings	A list of character vectors. Each vector is a set of columns that will be used to iteratively match the public dataset against the Paige dataset.

Value

Returns db_standardized (input occurrence records) with the Paige Chesshire data integrated.

Examples

```
## Not run:
library(dplyr)
# set the DataPath to tempdir for this example
DataPath <- tempdir()
# Integrate Paige Chesshire's cleaned dataset.
PaigeNAm <- readr::read_csv(paste(DataPath, "Paige_data", "NorAmer_highQual_only_ALLfamilies.csv",
                                  sep = "/"), col_types = ColTypeR()) %>%
 # Change the column name from Source to dataSource to match the rest of the data.
 dplyr::rename(dataSource = Source) %>%
 # add a NEW database_id column
 dplyr::mutate(
   database_id = paste0("Paige_data_", 1:nrow(.)),
   .before = scientificName)

 # Set up the list of character vectors to iteratively check for matches with public data.
columnList <- list(
 c("decimalLatitude", "decimalLongitude",
   "recordNumber", "recordedBy", "individualCount", "samplingProtocol",
   "associatedTaxa", "sex", "catalogNumber", "institutionCode", "otherCatalogNumbers",
   "recordId", "occurrenceID", "collectionID"), # Iteration 1
 c("catalogNumber", "institutionCode", "otherCatalogNumbers",
   "recordId", "occurrenceID", "collectionID"), # Iteration 2
 c("decimalLatitude", "decimalLongitude",
   "recordedBy", "genus", "specificEpithet"), # Iteration 3
 c("id", "decimalLatitude", "decimalLongitude"), # Iteration 4
 c("recordedBy", "genus", "specificEpithet", "locality"), # Iteration 5
 c("recordedBy", "institutionCode", "genus",
   "specificEpithet","locality"),# Iteration 6
 c("occurrenceID","decimalLatitude", "decimalLongitude"), # Iteration 7
 c("catalogNumber","decimalLatitude", "decimalLongitude"), # Iteration 8
 c("catalogNumber", "locality") # Iteration 9
)

# Merge Paige's data with downloaded data
```

```
db_standardized <- BeeBDC::PaigeIntegrater(
 db_standardized = db_standardized,
 PaigeNAm = PaigeNAm,
 columnStrings = columnList)

## End(Not run)
```

---

plotFlagSummary *Generate a plot summarising flagged data*

Description

Creates a compound bar plot that shows the proportion of records that pass or fail each flag (rows) and for each data source (columns). The function can also optionally return a point map for a user-specified species when plotMap = TRUE. This function requires that your dataset has been run through some filtering functions - so that is can display logical columns starting with ".".

Usage

```
plotFlagSummary(
  data = NULL,
  flagColours = c("#127852", "#A7002D", "#BDBABB"),
  fileName = NULL,
  outPath = OutPath_Figures,
  width = 15,
  height = 9,
  units = "in",
  dpi = 300,
  bg = "white",
  device = "pdf",
  speciesName = NULL,
  saveFiltered = FALSE,
  filterColumn = ".summary",
  nameColumn = NULL,
  plotMap = FALSE,
  mapAlpha = 0.5,
  xbuffer = c(0, 0),
  ybuffer = c(0, 0),
  ptSize = 1,
  saveTable = FALSE,
  jitterValue = NULL,
  returnPlot = FALSE,
  ...
)
```

Arguments

data	A data frame or tibble. Occurrence records as input.
flagColours	A character vector. Colours in order of pass (TRUE), fail (FALSE), and NA. Default = c("#127852", "#A7002D", "#BDBABB").
fileName	Character. The name of the file to be saved, ending in ".pdf". If saving as a different file type, change file type suffix - See device.
outPath	A character path. The path to the directory in which the figure will be saved. Default = OutPath_Figures.
width	Numeric. The width of the output figure in user-defined units Default = 15.
height	Numeric. The height of the output figure in user-defined units Default = 9.
units	Character. The units for the figure width and height passed to ggplot2::ggsave() ("in", "cm", "mm", or "px"). Default = "in".
dpi	Numeric. Passed to ggplot2::ggsave(). Plot resolution. Also accepts a string input: "retina" (320), "print" (300), or "screen" (72). Applies only to raster output types. Default = 300.
bg	Character. Passed to ggplot2::ggsave(). Background colour. If NULL, uses the plot.background fill value from the plot theme. Default = "white."
device	Character. Passed to ggplot2::ggsave(). Device to use. Can either be a device function (e.g. png), or one of "eps", "ps", "tex" (pictex), "pdf", "jpeg", "tiff", "png", "bmp", "svg" or "wmf" (windows only). Default = "pdf". If not using default, change file name suffix in fileName argument.
speciesName	Optional. Character. A species name, as it occurs in the user-input nameColumn. If provided, the data will be filtered to this species for the plot.
saveFiltered	Optional. Logical. If TRUE, the filtered data will be saved to the computer as a .csv file.
filterColumn	Optional. The flag column to display on the map. Default = .summary.
nameColumn	Optional. Character. If speciesName is not NULL, enter the column to look for the species in. A User might realise that, combined with speciesName, figures can be made for a variety of factors.
plotMap	Logical. If TRUE, the function will produce a point map. Tested for use with one species at a time; i.e., with speciesName is not NULL.
mapAlpha	Optional. Numeric. The opacity for the points on the map.
xbuffer	Optional. Numeric vector. A buffer in degrees of the amount to increase the min and max bounds along the x-axis. This may require some experimentation, keeping in mind the negative and positive directionality of hemispheres. Default = c(0,0).
ybuffer	Optional. Numeric vector. A buffer in degrees of the amount to increase the min and max bounds along the y-axis. This may require some experimentation, keeping in mind the negative and positive directionality of hemispheres. Default = c(0,0).
ptSize	Optional. Numeric. The size of the points as passed to ggplot2. Default = 1.
saveTable	Optional. Logical. If TRUE, the function will save the data used to produce the compound bar plot.

jitterValue	Optional. Numeric. The value to jitter points by in the map in decimal degrees.
returnPlot	Logical. If TRUE, return the plot to the environment. Default = FALSE.
...	Optional. Extra variables to be fed into forcats::fct_recode() to change names on plot. For example... 'B. Mont.' = "BMont", 'B. Minkley' = "BMin", Ecd = "Ecd", Gaiarsa = "Gai"

Value

Exports a compound bar plot that summarises all flag columns. Optionally can also return a point map for a particular species in tandem with the summary plot.

Examples

```
# import data
data(beesFlagged)
OutPath_Figures <- tempdir()
 # Visualise all flags for each dataSource (simplified to the text before the first underscore)
plotFlagSummary(
  data = beesFlagged,
  # Colours in order of pass (TRUE), fail (FALSE), and NA
  flagColours = c("#127852", "#A7002D", "#BDBABB"),
  fileName = paste0("FlagsPlot_TEST_", Sys.Date(),".pdf"),
  outPath = OutPath_Figures,
  width = 15, height = 9,
  # OPTIONAL:
  #\   #  # Filter to species
  #\   speciesName = "Holcopasites heliopsis",
  #\   # column to look in
  #\   nameColumn = "species",
  #\   # Save the filtered data
  #\   saveFiltered = TRUE,
  #\   # Filter column to display on map
  #\   filterColumn = ".summary",
  #\   plotMap = TRUE,
  #\   # amount to jitter points if desired, e.g. 0.25 or NULL
  #\   jitterValue = NULL,
  #\   # Map opacity value for points between 0 and 1
  #\   mapAlpha = 1,
  # Extra variables can be fed into forcats::fct_recode() to change names on plot
  GBIF = "GBIF", SCAN = "SCAN", iDigBio = "iDigBio", USGS = "USGS", ALA = "ALA",
  ASP = "ASP", CAES = "CAES", 'B. Mont.' = "BMont", 'B. Minkley' = "BMin", Ecd = "Ecd",
  Gaiarsa = "Gai", EPEL = "EPEL"
)
```

readr_BeeBDC *A wrapper for all of the data readr_functions*

Description

Read in a variety of data files that are specific to certain smaller data providers. There is an internal readr function for each dataset and each one of these functions is called by readr_BeeBDC. While these functions are internal, they are displayed in the documentation of readr_BeeBDC for clarity.

Usage

```
readr_BeeBDC(
  dataset = NULL,
  path = NULL,
  inFile = NULL,
  outFile = NULL,
  dataLicense = NULL,
  sheet = NULL
)

readr_EPEL(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_ASP(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_BMin(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_BMont(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_Ecd(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_Gai(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_CAES(
  path = NULL,
  inFile = NULL,
  outFile = NULL,
  dataLicense = NULL,
  sheet = "Sheet1"
)

readr_KP(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_EcoS(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_GeoL(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_EaCO(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)
```

```
readr_MABC(
  path = NULL,
  inFile = NULL,
  outFile = NULL,
  dataLicense = NULL,
  sheet = "Hoja1"
)

readr_Col(
  path = NULL,
  inFile = NULL,
  outFile = NULL,
  dataLicense = NULL,
  sheet = sheet
)

readr_FSCA(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_SMC(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_Bal(
  path = NULL,
  inFile = NULL,
  outFile = NULL,
  dataLicense = NULL,
  sheet = "animal_data"
)

readr_Lic(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_Arm(
  path = NULL,
  inFile = NULL,
  outFile = NULL,
  dataLicense = NULL,
  sheet = "Sheet1"
)

readr_Dor(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_MEPB(
  path = NULL,
  inFile = NULL,
  outFile = NULL,
  dataLicense = NULL,
  sheet = NULL
)
```

```
readr_BBD(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_MPUJ(
  path = NULL,
  inFile = NULL,
  outFile = NULL,
  dataLicense = NULL,
  sheet = sheet
)

readr_STRI(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_PALA(path = NULL, inFile = NULL, outFile = NULL, dataLicense = NULL)

readr_JoLa(
  path = NULL,
  inFile = NULL,
  outFile = NULL,
  dataLicense = NULL,
  sheet = c("pre-1950", "post-1950")
)

readr_VicWam(
  path = NULL,
  inFile = NULL,
  outFile = NULL,
  dataLicense = NULL,
  sheet = "Combined"
)
```

Arguments

dataset	Character. The name of the dataset to be read in. For example readr_CAES can be called using "readr_CAES" or "CAES". This is not caps sensitive.
path	A character path. The path to the directory containing the data.
inFile	Character or character path. The name of the file itself (can also be the remainder of a path including the file name).
outFile	Character or character path. The name of the Darwin Core format file to be saved.
dataLicense	Character. The license to accompany each record in the Darwin Core 'license' column.
sheet	A character String. For those datasets read from an .xlsx format, provide the sheet name. NOTE: This will be ignored for .csv readr_ functions and required for .xlsx readr_ functions.

Details

This function wraps several internal readr functions. Users may call readr_BeeBDC and select the dataset name to import a certain dataset. These datasets include:

Excel (.xlsx) formatted datasets: CAES, MABC, Col, Bal, MEPB, MUPJ, Arm, JoLa, and VicWam.

CSV (.csv) formatted datasets: EPEL, ASP, BMin, BMont, Ecd, Gai, KP, EcoS, GeoL, EaCo, FSCA, SMC, Lic, Dor, BBD, STRI, and PALA

See Dorey et al. 2023 BeeBDC... for further details.

Value

A data frame that is in Darwin Core format.

Examples

```
## Not run:
# An example using a .xlsx file
Arm_Data <- readr_BeeBDC(
    dataset = "Arm",
    path = paste0(tempdir(), "/Additional_Datasets"),
    inFile = "/InputDatasets/Bee database Armando_Final.xlsx",
    outFile = "jbd_Arm_Data.csv",
    sheet = "Sheet1",
    dataLicense = "https://creativecommons.org/licenses/by-nc-sa/4.0/")


    # An example using a .csv file
EPEL_Data <- readr_BeeBDC(
  dataset = "readr_EPEL",
  path = paste0(tempdir(), "/Additional_Datasets"),
  inFile = "/InputDatasets/bee_data_canada.csv",
  outFile = "jbd_EPEL_data.csv",
  dataLicense = "https://creativecommons.org/licenses/by-nc-sa/4.0/")

## End(Not run)
```

---

repoFinder *Find GBIF, ALA, iDigBio, and SCAN files in a directory*

Description

Find GBIF, ALA, iDigBio, and SCAN files in a directory

Usage

```
repoFinder(path)
```

Arguments

path	A directory as character. The path within which to recursively look for GBIF, ALA, iDigBio, and SCAN files.

Value

Returns a list of directories to each of the above data downloads

Examples

```
## Not run:
# Where DataPath is made by [BeeBDC::dirMaker()]
BeeBDC::repoFinder(path = DataPath)

## End(Not run)
```

---

repoMerge *Import occurrences from GBIF, ALA, iDigBio, and SCAN downloads*

Description

Locates data from GBIF, ALA, iDigBio, and SCAN within a directory and reads it in along with its eml metadata. Please keep the original download folder names and architecture unchanged. NOTE: This function uses family-level data to identify taxon downloads. If this, or something new, becomes an issue, please contact James Dorey (the developer) as there are likely to be exceptions to how files are downloaded. current as of versions 1.0.4.

Usage

```
repoMerge(path, save_type, occ_paths)
```

Arguments

path	A directory as a character. The directory to recursively look in for the above data.
save_type	Character. The data type to save the resulting file as. Options are: csv_files" or "R_file".
occ_paths	A list of directories. Preferably produced using `repoFinder()` the function asks for a list of paths to the relevant input datasets. You can fault-find errors in this function by checking the output of `repoFinder()`.

Value

A list with a data frame of merged occurrence records, "Data_WebDL", and a list of eml files contained in "eml_files". Also saves these files in the requested format.

Examples

```
## Not run:
if(require("emld")){

DataImp <- repoMerge(path = DataPath,
# Find data - Many problems can be solved by running [BeeBDC::repoFinder(path = DataPath)]
# And looking for problems
occ_paths = BeeBDC::repoFinder(path = DataPath),
save_type = "R_file")
} # END if require

## End(Not run) # END donttest
```

`richnessEstimateR` *Estimate country, continental, and global species richnesses*

Description

Takes an output dataset from `richnessPrepR()` to estimate species richness using iChao and iNEXT (hill numbers) for countries, continents, and/or the entire globe.

Usage

```
richnessEstimateR(
  data = NULL,
  sampleSize = 10000,
  countrySamples = 1,
  continentSamples = 1,
  globalSamples = 1,
  countriesToExclude = NULL,
  mc.cores = 1,
  k = 10,
  filterToRecordedCountries = TRUE,
  outPath = tempdir(),
  fileName = "continentSampled.pdf"
)
```

Arguments

`data`	an RData file created using the `richnessPrepR()` function.
`sampleSize`	Numeric. The size of the sample randomly drawn from the provided curve. See `curveFunction`. Default = 10000.
`countrySamples`	Numeric. The number of times to sample the country species richness for both iChao and iNEXT. If equal to zero (0), then this will not be analysed. Default = 5.

`continentSamples`
: Numeric. The number of times to sample the continent species richness for both iChao and iNEXT. If equal to zero (0), then this will not be analysed. Default = 5.

`globalSamples`
: Numeric. The number of times to sample the global species richness for both iChao and iNEXT. If equal to zero (0), then this will not be analysed. Default = 5.

`countriesToExclude`
: Character vector. You may decide to excluse some countries if they are being problematic or their sample sizes are too small. Default = NULL.

`mc.cores`
: Numeric. If > 1, the function will run in parallel using mclapply using the number of cores specified. If = 1 then it will be run using a serial loop. NOTE: Windows machines must use a value of 1 (see ?parallel::mclapply). Additionally, be aware that each thread can use large chunks of memory. Default = 1.

`k`
: Numeric. For iChao; the cut-off point (default = 10), which separates species into "abundant" and "rare" groups for abundance data for the estimator ACE; it separates species into "frequent" and "infrequent" groups for incidence data for the estimator ICE. Default = 10.

`filterToRecordedCountries`
: Logical. If TRUE, the checklist will be filtered to the countries Where occurrence records were found. Default = TRUE. Change at your own peril.

`outPath`
: A directory as character. Directory where to save output figure. Default = tempdir().

`fileName`
: A character vector with file name for the output figure, ending with '.pdf'. Default = "continentSampled.pdf".

Value

Outputs an R file with four tables ("Summary", "SiteOutput", "ContinentOutput", and "GlobalOutput"; depending on the number required). The summary table shows the Median overall estimates, while the remaining three shows the outputs from each iteration (useful for plotting, see relevant vignette). Some figures may also be saved to the selected outPath.

See Also

`countryHarmoniseR()` to harmonise country names based on a short list; `richnessPrepR()` to produce the input data and for the required column names; as well as `ChaoWrapper()` and `iNEXTwrapper()` for the parallelised implementation of `SpadeR` and `iNEXT` functions.

Examples

```
## Not run:

  # Use the example data
data(beesCountrySubset)

  # First,
estimateDataExample <- BeeBDC::richnessPrepR(
```

```
  data = beesCountrySubset,
  # Download the taxonomy
  taxonomyFile = BeeBDC::beesTaxonomy(),
  # Download the checklist
  checklistFile = BeeBDC::beesChecklist(),
  curveFunction = function(x) (228.7531 * x * x^-log(12.1593)),
  sampleSize = 10000,
  countryColumn = "country_suggested",
  limitGlobal = NULL,
  outPath = tempdir()
)

 exampleEstimate <- richnessEstimateR(
   data = estimateDataExample,
   sampleSize = 10000,
   countrySamples = 1,
   continentSamples = 1,
   globalSamples = 1,
   filterToRecordedCountries = TRUE,
   mc.cores = 1,
   # Directory where to save files
   outPath = tempdir(),
   fileName = "Sampled.pdf"
 )


## End(Not run) # END dontrun
```

`richnessPrepR` *Prepare occurrence, taxonomy, and checklist data for richness estimation*

Description

Takes your occurrence dataset along with a taxonomy and checklist in order to produce a file that's ready to be passed into the `richnessEstimateR()` function in order to estimate species richness using iChao and iNEXT (hill numbers) for countries, continents, or the entire globe.

Usage

```
richnessPrepR(
  data = NULL,
  taxonomyFile = BeeBDC::beesTaxonomy(),
  checklistFile = BeeBDC::beesChecklist(),
  curveFunction = function(x) (228.7531 * x * x^-log(12.1593)),
  sampleSize = 10000,
  countryColumn = "country_suggested",
  limitGlobal = NULL,
  outPath = tempdir()
)
```

Arguments

data	A data frame or tibble. Occurrence records as input. Needs to include the scientificName column and a "country" column (see countryColumn)
taxonomyFile	A data frame or tibble. The taxonomy file to use. Default = beesTaxonomy() but see taxadbToBeeBDC() for other taxa.
checklistFile	A data frame or tibble. The taxonomy to use. Default = beesChecklist(); use this as a template to convert other taxa checklists.
curveFunction	A function. The mathematical function that describes the curve used to randomly sample from in order to fill empty sample sizes for species present in the checklist but not present in the occurrence dataset. Default is function(x) (228.7531 * x * x^-log(12.1593)), which is taken from the paper "How many bee species are there? A quantitative global estimate" by Dorey et al. Models can be fit using mosaic::fitModel().
sampleSize	Numeric. The size of the sample randomly drawn from the provided curve. See curveFunction. Default = 10000.
countryColumn	Character. The column from which country names should be sought.
limitGlobal	Character vector. A character vector of the countries to filter the data to in order limit the extent of the global-level analysis. Defualt = NULL.
outPath	Character. The output path where the curve plot (curvePlot.pdf) and the output Rdata (richnessInputs.Rda) file will be saved. Default = tempdir().

Value

Saves an Rdata file with the data needed to feed into the richnessEstimateR() function. The richnessEstimateR() function will then use iChao and/or iNEXT to estimate species richness for countries, continents, and the globe. Also returns the RData file to the environment.

See Also

countryHarmoniseR() to harmonise country names based on a short list.

Examples

```
## Not run:
data(beesCountrySubset)
if(require("htmlwidgets"){

estimateDataExample <- richnessPrepR(
  data = beesCountrySubset,
  # Download the taxonomy
  taxonomyFile = BeeBDC::beesTaxonomy(),
  # Download the checklist
  checklistFile = BeeBDC::beesChecklist(),
  curveFunction = function(x) (228.7531 * x * x^-log(12.1593)),
  sampleSize = 10000,
  countryColumn = "country_suggested",
  limitGlobal = NULL,
  outPath = tempdir()
```

```
)
} # End if require

## End(Not run) # End dontrun
```

---

summaryFun *Create or update the .summary flag column*

Description

Using all flag columns (column names starting with "."), this function either creates or updates the .summary flag column which is FALSE when ANY of the flag columns are FALSE. Columns can be excluded and removed after creating the .summary column. Additionally, the occurrence dataset can be filtered to only those where .summary = TRUE at the end of the function.

Usage

```
summaryFun(
  data = NULL,
  dontFilterThese = NULL,
  onlyFilterThese = NULL,
  removeFilterColumns = FALSE,
  filterClean = FALSE
)
```

Arguments

data	A data frame or tibble. Occurrence records to use as input.
dontFilterThese	A character vector of flag columns to be ignored in the creation or updating of the .summary column. Cannot be specified with onlyFilterThese.
onlyFilterThese	A character vector. The inverse of dontFilterThese, where columns identified here will be filtered and no others. Cannot be specified with dontFilterThese.
removeFilterColumns	Logical. If TRUE all columns starting with "." will be removed in the output data. This only makes sense to use when filterClean = TRUE. Default = FALSE.
filterClean	Logical. If TRUE, the data will be filtered to only those occurrence where .summary = TRUE (i.e., completely clean according to the used flag columns). Default = FALSE.

Value

Returns a data frame or tibble of the input data but modified based on the above parameters.

Examples

```
# Read in example data
data(beesFlagged)

# To only update the .summary column
beesFlagged_out <- summaryFun(
    data = beesFlagged,
   dontFilterThese = c(".gridSummary", ".lonFlag", ".latFlag", ".uncer_terms", ".unLicensed"),
    removeFilterColumns = FALSE,
    filterClean = FALSE)
  # View output
table(beesFlagged_out$.summary, useNA = "always")

# Now filter to only the clean data and remove the flag columns
beesFlagged_out <- summaryFun(
  data = BeeBDC::beesFlagged,
 dontFilterThese = c(".gridSummary", ".lonFlag", ".latFlag", ".uncer_terms", ".unLicensed"),
  removeFilterColumns = TRUE,
  filterClean = TRUE)
# View output
table(beesFlagged_out$.summary, useNA = "always")
```

---

summaryMaps *Create country-level summary maps of species and occurrence numbers*

Description

Builds an output figure that shows the number of species and the number of occurrences per country. Breaks the data into classes for visualisation. Users may filter data to their taxa of interest to produce figures of interest.

Usage

```
summaryMaps(
  data = NULL,
  class_n = 15,
  class_Style = "fisher",
  outPath = NULL,
  fileName = NULL,
  width = 5,
  height = 10,
  dpi = 300,
  returnPlot = FALSE,
  scale = 110,
  pointBuffer = 0.01
)
```

Arguments

data	A data frame or tibble. Occurrence records as input.
class_n	Numeric. The number of categories to break the data into.
class_Style	Character. The class style passed to classInt::classIntervals(). Options are chosen style: one of "fixed", "sd", "equal", "pretty", "quantile", "kmeans", "hclust", "bclust", "fisher", "jenks", "dpih", "headtails", or "maximum". Default = "fisher"
outPath	A character vector the path to the save location for the output figure.
fileName	A character vector with file name for the output figure, ending with '.pdf'.
width	Numeric. The width, in inches, of the resulting figure. Default = 5.
height	Numeric. The height, in inches, of the resulting figure. Default = 10.
dpi	Numeric. The resolution of the resulting plot. Default = 300.
returnPlot	Logical. If TRUE, return the plot to the environment. Default = FALSE.
scale	Numeric or character. Passed to rnaturalearth's ne_countries(). Scale of map to return, one of 110, 50, 10 or 'small', 'medium', 'large'. Default = 110.
pointBuffer	Numeric. Amount to buffer points, in decimal degrees. If the point is outside of a country, but within this point buffer, it will count towards that country. It's a good idea to keep this value consistent with the prior flags applied. Default = 0.01.

Value

Saves a figure to the user-specified outpath and name with a global map of bee occurrence species and count data from the input dataset.

Examples

```
if(requireNamespace("rnaturalearthdata", quietly = TRUE)){
# Read in data
data(beesFlagged)
OutPath_Figures <- tempdir()
# This simple example using the test data has very few classes due to the small amount of input
# data.
summaryMaps(
data = beesFlagged,
width = 10, height = 10,
class_n = 4,
class_Style = "fisher",
outPath = OutPath_Figures,
fileName = paste0("CountryMaps_fisher_TEST.pdf"),
)
} # END if require
```

taxadbToBeeBDC *Import and convert taxadb taxonomies to BeeBDC format*

Description

Uses the taxadb R package to download a requested taxonomy and then transforms it into the input BeeBDC format. This means that any taxonomy in their databases can be used with BeeBDC. You can also save the output to your computer and to the R environment for immediate use. See details below for a list of providers or see `taxadb::td_download()`.

Usage

```
taxadbToBeeBDC(
  name = NULL,
  rank = NULL,
  provider = "gbif",
  version = taxadb::available_versions(db = taxadb::td_connect()),
  collect = TRUE,
  ignore_case = TRUE,
  db = NULL,
  removeEmptyNames = TRUE,
  outPath = getwd(),
  fileName = NULL,
  overwrite = FALSE,
  ...
)
```

Arguments

`name`	Character. Taxonomic scientific name (e.g. "Aves"). As defined by `taxadb::filter_rank()`.
`rank`	Character. Taxonomic rank name. (e.g. "class"). As defined by `taxadb::filter_rank()`.
`provider`	Character. From which provider should the hierarchy be returned? Default is 'gbif', which can also be configured using options(default_taxadb_provide = ..."). See `taxadb::td_download()` for a list of recognized providers. NOTE: gbif seems to have the most-complete columns, especially in terms of scientificNameAuthorship, which is important for matching ambiguous names. As defined by `taxadb::filter_rank()`.
`version`	Character. Which version of the taxadb provider database should we use? defaults to latest. See available_versions() for details. Default = 22.12. As defined by `taxadb::filter_rank()`.
`collect`	Logical. Should we return an in-memory data.frame (default, usually the most convenient), or a reference to lazy-eval table on disk (useful for very large tables on which we may first perform subsequent filtering operations.). Default = TRUE. As defined by `taxadb::filter_rank()`.
`ignore_case`	Logical. should we ignore case (capitalization) in matching names? Can be significantly slower to run. Default = TRUE. As defined by `taxadb::filter_rank()`.

db	a connection to the taxadb database. See details of `taxadb::filter_rank()`. Default = Null which should work. As defined by `taxadb::filter_rank()`.
removeEmptyNames	Logical. If True (default), it will remove entries without an entry for specificEpithet.
outPath	Character. The path to a directory (folder) in which the output should be saved.
fileName	Character. The name of the output file, ending in '.csv'.
overwrite	Logical. Should we re-download files that are already present? Default FALSE.
...	Arguments passed to `taxadb::td_download()`.

Value

Returns a taxonomy file (to the R environment and to the disk, if a fileName is provided) as a tibble that can be used with `BeeBDC::harmoniseR()`.

See Also

`beesTaxonomy()` for the bee taxonomy and `harmoniseR()` for the taxon-cleaning function where these taxonomies are implemented.

Examples

```
## Not run:

if(require("taxadb")){

  # Run the function using the bee genus Apis as an example...
ApisTaxonomy <- BeeBDC::taxadbToBeeBDC(
  name = "Apis",
  rank = "Genus",
  provider = "gbif",
  version = "22.12",
  removeEmptyNames = TRUE,
  outPath = getwd(),
  fileName = NULL,
  overwrite = FALSE,
  ...
  )

  } # END if require

## End(Not run) # End dontrun
```

`testChecklist` *An example of the beesChecklist file*

Description

A small test checklist file for package tests. This dataset was built by filtering the checklist data from the three test datasets, beesFlagged, beesRaw, bees3sp.

Usage

```
data("testChecklist", package = "BeeBDC")
```

Format

An object of class `"tibble"`

validName The valid scientificName as it should occur in the scientificName column.

DiscoverLife_name The full country name as it occurs on Discover Life.

rNaturalEarth_name Country name from rnaturalearth's name_long and type = "map_units".

shortName A short version of the country name.

continent The continent where that country is found.

DiscoverLife_ISO The ISO country name as it occurs on Discover Life.

Alpha-2 Alpha-2 from rnaturalearth.

iso_a3_eh iso_a3_eh from rnaturalearth.

official Official country name = "yes" or only a Discover Life name = "no".

Source A text strign denoting the source or author of the name-country pair.

matchCertainty Quality of the name's match to the Discover Life checklist.

canonical The valid species name without scientificNameAuthority.

canonical_withFlags The validName without the scientificNameAuthority but with Discover Life flags.

family Bee family.

subfamily Bee subfamily.

genus Bee genus.

subgenus Bee subgenus.

infraspecies Bee infraSpecificEpithet.

species Bee specificEpithet.

scientificNameAuthorship Bee scientificNameAuthorship.

taxon_rank Rank of the taxon name.

Notes Discover Life country name notes.

References

This dataset is a subset of the beesChecklist file described in: Dorey, J.B., Fischer, E.E., Chesshire, P.R., Nava-Bolaños, A., O'Reilly, R.L., Bossert, S., Collins, S.M., Lichtenberg, E.M., Tucker, E., Smith-Pardo, A., Falcon-Brindis, A., Guevara, D.A., Ribeiro, B.R., de Pedro, D., Hung, J.K.-L., Parys, K.A., McCabe, L.M., Rogan, M.S., Minckley, R.L., Velzco, S.J.E., Griswold, T., Zarrillo, T.A., Jetz, W., Sica, Y.V., Orr, M.C., Guzman, L.M., Ascher, J., Hughes, A.C. & Cobb, N.S. (2023) A globally synthesised and flagged bee occurrence dataset and cleaning workflow. Scientific Data, 10, 1–17. https://www.doi.org/10.1038/S41597-023-02626-W

Examples

```
beesRaw <- BeeBDC::testChecklist
head(testChecklist)
```

---

testTaxonomy *An example of the beesTaxonomy file*

Description

A small test taxonomy file for package tests. This dataset was built by filtering the taxonomy data from the three test datasets, beesFlagged, beesRaw, bees3sp.

Usage

```
data("testTaxonomy", package = "BeeBDC")
```

Format

An object of class `"tibble"`

taxonomic_status Taxonomic status. Values are "accepted" or "synonym"

source Source of the name.

accid The id of the accepted taxon name or "0" if taxonomic_status == accepted.

id The id number for the taxon name.

kingdom The biological kingdom the taxon belongs to. For bees, kingdom == Animalia.

phylum The biological phylum the taxon belongs to. For bees, phylum == Arthropoda.

class The biological class the taxon belongs to. For bees, class == Insecta.

order The biological order the taxon belongs to. For bees, order == Hymenoptera.

family The family of bee which the species belongs to.

subfamily The subfamily of bee which the species belongs to.

tribe The tribe of bee which the species belongs to.

subtribe The subtribe of bee which the species belongs to.

validName The valid scientific name as it should occur in the 'scientificName" column in a Darwin Core file.

canonical The scientificName without the scientificNameAuthority.

canonical_withFlags The scientificName without the scientificNameAuthority and with Discover Life taxonomy flags.

genus The genus the bee species belongs to.

subgenus The subgenus the bee species belongs to.

species The specific epithet for the bee species.

infraspecies The infraspecific epithet for the bee addressed.

authorship The author who described the bee species.

taxon_rank Rank for the bee taxon addressed in the entry.

notes Additional notes about the name/taxon.

References

This dataset is a subset of the beesTaxonomy file described in: Dorey, J.B., Fischer, E.E., Chesshire, P.R., Nava-Bolaños, A., O'Reilly, R.L., Bossert, S., Collins, S.M., Lichtenberg, E.M., Tucker, E., Smith-Pardo, A., Falcon-Brindis, A., Guevara, D.A., Ribeiro, B.R., de Pedro, D., Hung, J.K.-L., Parys, K.A., McCabe, L.M., Rogan, M.S., Minckley, R.L., Velzco, S.J.E., Griswold, T., Zarrillo, T.A., Jetz, W., Sica, Y.V., Orr, M.C., Guzman, L.M., Ascher, J., Hughes, A.C. & Cobb, N.S. (2023) A globally synthesised and flagged bee occurrence dataset and cleaning workflow. Scientific Data, 10, 1–17. https://www.doi.org/10.1038/S41597-023-02626-W

Examples

```
beesRaw <- BeeBDC::testTaxonomy
head(testTaxonomy)
```

USGS_formatter *Find, import, and format USGS data to Darwin Core*

Description

The function finds, imports, formats, and creates metadata for the USGS dataset.

Usage

```
USGS_formatter(path, pubDate)
```

Arguments

path	A character path to a directory that contains the USGS data, which will be found using `fileFinder()`. The function will look for "USGS_DRO_flat".
pubDate	Character. The publication date of the dataset to update the metadata and citation.

Value

Returns a list with the occurrence data, "USGS_data", and the EML data, "EML_attributes".

Examples

```
## Not run:
USGS_data <- USGS_formatter(path = DataPath, pubDate = "19-11-2022")

## End(Not run)
```

Index

∗ **datasets**
bees3sp, 6
beesCountrySubset, 14
beesFlagged, 15
beesRaw, 21
testChecklist, 98
testTaxonomy, 99
∗ **prefilter**
jbd_coordinates_transposed, 72
. . . , *32*

atlasDownloader, 3

base::summary(), *52*
bdc::bdc_basisOfRecords_notStandard(), *10*, *19*
bdc::bdc_clean_names(), *10*, *19*, *59*
bdc::bdc_coordinates_country_inconsistent(), *69*
bdc::bdc_coordinates_empty(), *10*, *18*
bdc::bdc_coordinates_outOfRange(), *19*
bdc::bdc_coordinates_transposed(), *76*
bdc::bdc_country_from_coordinates(), *67*
bdc::bdc_country_standardized, *73*
bdc::bdc_country_standardized(), *10*, *14*, *19*, *72*, *76*
bdc::bdc_create_figures(), *74*
bdc::bdc_eventDate_empty(), *11*, *20*
bdc::bdc_scientificName_empty(), *10*, *18*
bdc::bdc_year_outOfRange(), *11*, *20*
BeeBDCQuery, 4
bees3sp, 6
beesChecklist, 12
beesChecklist(), *5*, *33, 34*, *37, 38*, *92*
beesCountrySubset, 14
beesFlagged, 15
beesRaw, 21
beesTaxonomy, 25
beesTaxonomy(), *5*, *12*, *59, 60*, *92*, *97*
ChaoWrapper, 27
ChaoWrapper(), *90*
chordDiagramR, 29
chordDiagramR(), *48*
circlize::chordDiagram(), *30*
circlize::circos.par(), *30*
circlize::highlight.sector(), *30*
classInt::classIntervals(), *95*
ColTypeR, 32
ColTypeR(), *11*, *15*, *21*, *32*
complete.cases(), *47*
continentOutlieRs, 33
continentOutlieRs(), *38*
CoordinateCleaner::cd_round(), *11*, *20*
CoordinateCleaner::clean_coordinates(), *10, 11*, *19*
coordUncerFlagR, 34
coordUncerFlagR(), *11*, *20*
countryHarmoniseR, 35
countryHarmoniseR(), *90*, *92*
countryNameCleanR, 36
countryOutlieRs, 37
countryOutlieRs(), *11*, *20*, *34*

dataProvTables, 38
dataSaver, 39
dateFindR, 40
diagonAlley, 41
diagonAlley(), *11*, *19*
dirMaker, 43
dupePlotR, 45
dupePlotR(), *48*
dupeSummary, 46
dupeSummary(), *11*, *20*, *29, 30*, *45*, *78, 79*

fileFinder, 49
fileFinder(), *62*, *100*
flagAbsent, 50
flagAbsent(), *10*, *19*
flagLicense, 51

flagLicense(), *10*, *19*
flagRecorder, 52
flagSummaryTable, 53
forcats::fct_recode(), *83*
formattedCombiner, 54

galah::galah_config(), *4*
galah::galah_identify(), *4*
GBIFissues, 55
GBIFissues(), *10*, *19*
ggplot2::ggsave(), *82*
ggRichnessWrapper, 56

harmoniseR, 59
harmoniseR(), *10*, *19*, *27*, *59*, *97*
here::here(), *73*
here::i_am(), *44*

idMatchR, 61
importOccurrences, 62
iNEXT::iNEXT(), *64*
iNEXTwrapper, 63
iNEXTwrapper(), *90*
interactiveMapR, 65

jbd_CfC_chunker, 67
jbd_coordCountryInconsistent, 69
jbd_coordCountryInconsistent(), *10*, *19*
jbd_coordinates_precision, 71
jbd_coordinates_precision(), *42*
jbd_coordinates_transposed, 72
jbd_coordinates_transposed(), *76*
jbd_create_figures, 74
jbd_Ctrans_chunker, 76
jbd_Ctrans_chunker(), *10*, *19*

manualOutlierFindeR, 78
manualOutlierFindeR(), *48*
mosaic::fitModel(), *92*

PaigeIntegrater, 79
plotFlagSummary, 81

readr::as.col_spec(), *32*
readr::col_character(), *32*
readr::col_integer(), *32*
readr::col_logical(), *32*
readr::cols_only(), *32*
readr::read_csv(), *32*
readr_Arm *(*readr_BeeBDC*)*, 84
readr_ASP *(*readr_BeeBDC*)*, 84
readr_Bal *(*readr_BeeBDC*)*, 84
readr_BBD *(*readr_BeeBDC*)*, 84
readr_BeeBDC, 84
readr_BMin *(*readr_BeeBDC*)*, 84
readr_BMont *(*readr_BeeBDC*)*, 84
readr_CAES *(*readr_BeeBDC*)*, 84
readr_Col *(*readr_BeeBDC*)*, 84
readr_Dor *(*readr_BeeBDC*)*, 84
readr_EaCO *(*readr_BeeBDC*)*, 84
readr_Ecd *(*readr_BeeBDC*)*, 84
readr_EcoS *(*readr_BeeBDC*)*, 84
readr_EPEL *(*readr_BeeBDC*)*, 84
readr_FSCA *(*readr_BeeBDC*)*, 84
readr_Gai *(*readr_BeeBDC*)*, 84
readr_GeoL *(*readr_BeeBDC*)*, 84
readr_JoLa *(*readr_BeeBDC*)*, 84
readr_KP *(*readr_BeeBDC*)*, 84
readr_Lic *(*readr_BeeBDC*)*, 84
readr_MABC *(*readr_BeeBDC*)*, 84
readr_MEPB *(*readr_BeeBDC*)*, 84
readr_MPUJ *(*readr_BeeBDC*)*, 84
readr_PALA *(*readr_BeeBDC*)*, 84
readr_SMC *(*readr_BeeBDC*)*, 84
readr_STRI *(*readr_BeeBDC*)*, 84
readr_VicWam *(*readr_BeeBDC*)*, 84
repoFinder, 87
repoFinder(), *88*
repoMerge, 88
repoMerge(), *62*
richnessEstimateR, 89
richnessEstimateR(), *91*, *92*
richnessPrepR, 91
richnessPrepR(), *89*, *90*
rnaturalearth::ne_countries(), *33*, *37*, *70*

SpadeR::ChaoSpecies(), *28*
summaryFun, 93
summaryFun(), *11*, *20*, *47*, *65*
summaryMaps, 94

taxadbToBeeBDC, 96
taxadbToBeeBDC(), *27*, *59*, *60*, *92*
tempdir(), *27*
testChecklist, 98
testTaxonomy, 99

USGS_formatter, 100
USGS_formatter(), *54*