

# Package ‘BeeBDC’

February 14, 2024

**Title** Occurrence Data Cleaning

**Version** 1.0.4

**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](https://doi.org/10.1101/2023.06.30.547152)> and package website - Dorey et al. (2023) <<https://github.com/jbdorey/BeeBDC>>.

**License** GPL (>= 3)

**BugReports** <https://github.com/jbdorey/BeeBDC/issues>

**Depends** R (>= 2.10)

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

**Suggests** bdc, BiocManager, classInt, ComplexHeatmap, countrycode, devtools, emld, formatR, galah, hexbin, htmltools, htmlwidgets, httr, janitor, knitr, leaflet, magrittr, pkgdown, plotly, prettydoc, purrr, R.utils, renv, rgnparser, rlang, rmarkdown, rmdformats, rnaturalearthdata, rvest, terra, testthat, tidyr, utils, xml2

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-gb

**LazyData** TRUE

**LazyDataCompression** xz

**RoxygenNote** 7.3.1

**NeedsCompilation** no

**Author** James B. Dorey [aut, cre, cph]

(<<https://orcid.org/0000-0003-2721-3842>>),

Robert L. O'Reilly [aut] (<<https://orcid.org/0000-0001-5291-7396>>),

Silas Bossert [aut] (<<https://orcid.org/0000-0002-3620-5468>>),

Erica E. Fischer [aut] (<<https://orcid.org/0000-0002-8202-158X>>)

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

**Repository** CRAN

**Date/Publication** 2024-02-14 04:42:08 UTC

## R topics documented:

atlasDownloader	3
BeeBDCQuery	4
bees3sp	5
beesChecklist	11
beesFlagged	13
beesRaw	19
beesTaxonomy	23
chordDiagramR	25
ColTypeR	27
coordUncerFlagR	28
countryNameCleanR	29
countryOutlieRs	30
dataProvTables	31
dataSaver	32
dateFindR	33
diagonAlley	34
dirMaker	36
dupePlotR	38
dupeSummary	39
fileFinder	42
flagAbsent	43
flagLicense	44
flagRecorder	45
flagSummaryTable	46
formattedCombiner	47
GBIFissues	48
harmoniseR	49
idMatchR	50
importOccurrences	51
interactiveMapR	52
jbd_CfC_chunker	54
jbd_coordCountryInconsistent	56
jbd_coordinates_precision	57
jbd_coordinates_transposed	58

jbd_create_figures . . . . .	61
jbd_Ctrans_chunker . . . . .	62
manualOutlierFindeR . . . . .	64
PaigeIntegrater . . . . .	66
plotFlagSummary . . . . .	67
readr_BeeBDC . . . . .	70
repoFinder . . . . .	75
repoMerge . . . . .	75
summaryFun . . . . .	76
summaryMaps . . . . .	77
USGS_formatter . . . . .	79

<b>Index</b>	<b>80</b>
--------------	-----------

---

atlasDownloader	<i>Download occurrence data from the Atlas of Living Australia (ALA)</i>
-----------------	--

---

## 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](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 <code>galah::galah_identify()</code>
DL_reason	Numeric. The reason for data download according to <code>galah::galah_config()</code>
atlas	Character. The atlas to download occurrence data from - see here <a href="https://galah.ala.org.au/R/articles/choosing_an_atlas.html">https://galah.ala.org.au/R/articles/choosing_an_atlas.html</a> 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 <code>beesChecklist()</code> will be called internally to download the file. Default = NULL.
beesTaxonomy	A tibble. The bee taxonomy file for BeeBDC. If is NULL then <code>beesTaxonomy()</code> 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 <- 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 occurrence 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 Occurrence.
- duplicateStatus** Variable indicating if 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 scientific-Name
- .invalidName** A flag generated by `harmoniseR()` where FALSE == occurrences whose scientific-Name 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 (.duplctes == 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., 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.

## Examples

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

## 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")`.

See `beesTaxonomy()` for further context.

## Usage

```
beesChecklist(
  URL =
    "https://figshare.com/ndownloader/files/42320598?private_link=bce1f92848c2ced313ee",
  ...
)
```

## Arguments

URL	A character vector to the FigShare location of the dataset. The default will be to the most-recent version.
...	Extra variables that can be passed to <code>utils::download.file()</code>

## 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 string 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 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., FALCONBRINDIS, 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](http://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](http://www.discoverlife.org/mp/20q?guide=Apoidea_species).

## Examples

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

## End(Not run)
```

---

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 Occurrence.
- 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 transposed 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 scientific-Name
- .invalidName** A flag generated by `harmoniseR()` where FALSE == occurrences whose scientific-Name 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 `countryOutlierRs()`. 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 `countryOutlierRs()` 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 `countryOutlierRs()` 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., 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.

## Examples

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

---

beesRaw	<i>A dataset of 100 random bee occurrence records without flags or filters applied</i>
---------	--

---

## 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 Occurrence.

**duplicateStatus** Variable indicating if 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., 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.

## 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")`.

## Usage

```
beesTaxonomy(
  URL = "https://open.flinders.edu.au/ndownloader/files/43331472",
  ...
)
```

## Arguments

URL A character vector to the FigShare location of the dataset. The default will be to the most-recent version.

... Extra variables that can be passed to `utils::download.file()`

## 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.

## 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](http://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](http://www.discoverlife.org/mp/20q?guide=Apoidea_species).

### Examples

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

## 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**

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

## 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()`

## Value

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

## 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())
```

---

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")
```

---

countryNameCleanR	<i>Fix country name issues using a user-input list</i>
-------------------	--

---

**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')))
```

---

countryOutlierRs

*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**

```
countryOutlierRs(
  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 <code>rnaturalearth</code>

scale	Numeric. The value fed into the map scale parameter for <code>rnaturalearth::ne_countries()</code> '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 <code>mclapply</code> 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 <code>?parallel::mclapply</code> ). Additionally, be aware that each thread can use large chunks of memory. If the cores throw issues, consider setting <code>mc.cores</code> 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.

### Examples

```
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 <- countryOutlierRs(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")
```

---

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,  
  outputPath = 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.
outputPath	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,  
  outputPath = 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	<i>Find dates in other columns</i>
-----------	------------------------------------

---

**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.

**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

<code>data</code>	A data frame or tibble. Occurrence records as input.
<code>minRepeats</code>	Numeric. The minimum number of lat or lon repeats needed to flag a record
<code>groupingColumns</code>	Character. The column(s) to group the analysis by and search for fill-down errors within. Default = <code>c("eventDate", "recordedBy", "datasetName")</code> .
<code>ndec</code>	Numeric. The number of decimal places below which records will not be considered in the <code>diagonAlley</code> function. This is fed into <code>jbd_coordinates_precision()</code> . Default = 3.
<code>stepSize</code>	Numeric. The number of occurrences to process in each chunk. Default = 1000000.
<code>mc.cores</code>	Numeric. If > 1, the function will run in parallel using <code>mclapply</code> 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 <code>?parallel::mclapply</code> ). 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 Peikering'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 <code>here::i_am()</code> and incorrectly setting this may result in bdc figures being saved to your computer's root directory
useHere	Logical. If TRUE, dirMaker will use <code>here::i_am()</code> 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 = 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	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 = 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

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



- 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 match more 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\(\)](#)

## 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 provided file. Using regex can greatly improve specificity. 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	<i>Flag license protected records</i>
-------------	---------------------------------------

---

### 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	<i>Loads, appends, and saves occurrence flag data</i>
--------------	---

---

### 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 <a href="#">summary()</a> 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**

<code>data</code>	A data frame or tibble. Occurrence records as input.
<code>issueColumn</code>	Character. The column in which to look for GBIF issues. Default = "issue".
<code>GBIFflags</code>	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 <code>c("allDates", "allMetadata", "allObservations", "allSpatial", "allTaxo", or "all")</code> . Default = <code>c("COORDINATE_INVALID", "PRESUMED_NEGATED_LONGITUDE", "PRESUMED_NEGATED_LATITUDE", "COUNTRY_COORDINATE_MISMATCH", "ZERO_COORDINATE")</code>

**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"))
```



---

 harmoniseR

*Harmonise taxonomy of bee occurrence data*


---

## Description

Uses the Discover Life taxonomy to harmonise bee occurrences and flag those that do not match the checklist. `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. This function could be hijacked to service other taxa if a user matched the format of the `beesTaxonomy()` file.

## Usage

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

## Arguments

<code>data</code>	A data frame or tibble. Occurrence records as input.
<code>path</code>	A directory as character. The path to a folder that the output can be saved.
<code>taxonomy</code>	A data frame or tibble. The bee taxonomy to use. Default = <code>beesTaxonomy()</code> .
<code>speciesColumn</code>	Character. The name of the column containing species names. Default = "scientificName".
<code>rm_names_clean</code>	Logical. If TRUE then the <code>names_clean</code> column will be removed at the end of this function to help reduce confusion about this column later. Default = TRUE
<code>checkVerbatim</code>	Logical. If TRUE then the <code>verbatimScientificName</code> will be checked as well for species matches. This matching will ONLY be done after <code>harmoniseR</code> has failed for the other name columns. NOTE: this column is <i>not</i> first run through <code>bdc::bdc_clean_names</code> . Default = FALSE
<code>stepSize</code>	Numeric. The number of occurrences to process in each chunk. Default = 1000000.
<code>mc.cores</code>	Numeric. If > 1, the function will run in parallel using <code>mclapply</code> 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 <code>?parallel::mclapply</code> ). 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.

**Examples**

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

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 column 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 prefixes - 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**

<code>path</code>	A directory as a character. The directory to recursively look in for the above data.
<code>fileName</code>	Character. A String of text to look for the most-recent dataset. Default = " <code>^Bee-Data_</code> ". Find faults by modifying <a href="#"><code>fileFinder()</code></a> 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)
```

---

<code>interactiveMapR</code>	<i>Creates interactive html maps for species</i>
------------------------------	--

---

**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 = NULL,
  countryList = NULL,
  jitterValue = NULL,
  onlySummary = TRUE,
  overWrite = TRUE,
  TrueAlwaysTop = FALSE,
  excludeApis_mellifera = TRUE,
  pointColours = c("blue", "darkred", "#ff7f00", "black")
)
```

**Arguments**

<code>data</code>	A data frame or tibble. Occurrence records to use as input.
<code>outPath</code>	A directory as character. Directory where to save output maps.
<code>lon</code>	Character. The name of the longitude column. Default = "decimalLongitude".
<code>lat</code>	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".
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.
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.
excludeApis_mellifera	Logical. If TRUE, will not map records for Apis mellifera. Note: in most cases A. mellifera has too many points, and the resulting map will take a long time to make and be difficult to open. Default = TRUE.
pointColours	A character vector of colours. In order provide colour for TRUE, FALSE, countryOutlier, and customOutlier. Default = c("blue", "darkred", "#ff7f00", "black").

## Value

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

## Examples

```
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,
excludeApis_mellifera = TRUE,
overWrite = TRUE,
  # Colours for points which are flagged as TRUE, FALSE, countryOutlier, and customOutlier
pointColours = c("blue", "darkred", "#ff7f00", "black")
)

```

---

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

<code>data</code>	A data frame or tibble. Occurrence records to use as input.
<code>lat</code>	Character. The name of the column to use as latitude. Default = "decimalLatitude".
<code>lon</code>	Character. The name of the column to use as longitude. Default = "decimalLongitude".
<code>country</code>	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 rnatuarearth'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

```
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 = "")

```

---

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_inconsistent`. 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

<code>data</code>	A data frame or tibble. Occurrence records as input.
<code>lon</code>	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 <code>rnaturalearth::ne_countries()</code> '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 <code>st_intersects</code> function will run in parallel using <code>mclapply</code> 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 <code>?parallel::mclapply</code> ). 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

```
beesRaw_out <- jbd_coordCountryInconsistent(
  data = BeeBDC::beesRaw,
  lon = "decimalLongitude",
  lat = "decimalLatitude",
  scale = 50,
  pointBuffer = 0.01)
```

---

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**

<code>data</code>	A data frame or tibble. Occurrence records as input.
<code>lat</code>	Character. The name of the column to use as latitude. Default = "decimalLatitude".
<code>lon</code>	Character. The name of the column to use as longitude. Default = "decimalLongitude".
<code>ndec</code>	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).
<code>quieter</code>	Logical. If TRUE, the function 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 create a buffer around each country. Records within a given country and at a specified distance from the border will 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 <code>rnaturalearth::ne_countries()</code> . 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 <a href="#">here::here()</a> .
mc.cores	Numeric. If > 1, the <code>jbd_correct_coordinates</code> function will run in parallel using <code>mclapply</code> 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 <code>?parallel::mclapply</code> ). 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

```
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"
)
```

---

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 in 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

<code>data</code>	A data frame or tibble. Needs to contain the results of data quality tests; that is, columns starting with ".".
<code>path</code>	A character directory. The path to a directory in which to save the figures. Default = <code>OutPath_Figures</code> .
<code>database_id</code>	A character string. The column name with a unique record identifier. Default = <code>"database_id"</code> .
<code>workflow_step</code>	A character string. Name of the workflow step. Options available are <code>"prefilter"</code> , <code>"space"</code> , and <code>"time"</code> .
<code>save_figures</code>	Logical. Indicates if the figures should be saved for further inspection or use. Default = <code>FALSE</code> .

## 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

```

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_empty <- 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_empty,
  .coordinates_empty,
  .invalid_basis_of_records,
  .summary
)

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

```

---

jbd_Ctrans_chunker	<i>Wraps jbd_coordinates_transposed to identify and fix transposed occurrences</i>
--------------------	--

---

## 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 = "database_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 rnatualearth'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**

```
library(dplyr)
# Import and prepare the data
data(beesFlagged)
beesFlagged <- beesFlagged %>% dplyr::select(!c(.val, .sea)) %>%
  # Cut down the dataset to an 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")
```

---

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**

<code>data</code>	A data frame or tibble. Occurrence records as input.
<code>DataPath</code>	A character path to the directory that contains the outlier spreadsheets.
<code>PaigeOutliersName</code>	A character patch. Should lead to outlier spreadsheet from Paige Chesshire (csv file).
<code>newOutliersName</code>	A character path. Should lead to appropriate outlier spreadsheet (xlsx file).
<code>ColombiaOutliers_all</code>	A character path. Should lead to spreadsheet of bee outliers from Colombia (csv file).
<code>duplicates</code>	A data frame or tibble. The duplicate file produced by <code>dupeSummary()</code> .
<code>NearTRUE</code>	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 <code>NearTRUE.csv</code> . Note: This implementation is only basic for now unless there is a greater need in the future.
<code>NearTRUE_threshold</code>	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)
```

---

 PaigeIntegrator

*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

```
PaigeIntegrator(db_standardized = NULL, PaigeNAM = NULL, columnStrings = NULL)
```

### Arguments

<code>db_standardized</code>	A data frame or tibble. Occurrence records as input.
<code>PaigeNAM</code>	A data frame or tibble. The Paige Chesshire dataset.
<code>columnStrings</code>	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,
  PaigeNA = PaigeNA,
  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 it 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 <code>ggplot2::ggsave()</code> ("in", "cm", "mm", or "px"). Default = "in".
dpi	Numeric. Passed to <code>ggplot2::ggsave()</code> . 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 <code>ggplot2::ggsave()</code> . Background colour. If NULL, uses the <code>plot.background fill</code> value from the plot theme. Default = "white."
device	Character. Passed to <code>ggplot2::ggsave()</code> . 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 <code>fileName</code> argument.
speciesName	Optional. Character. A species name, as it occurs in the user-input <code>nameColumn</code> . 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 <code>forcats::fct_recode()</code> 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, MUJ, 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.

**Functions**

- readr\_EPEL(): Reads specific data files into Darwin Core format
- readr\_ASP(): Reads specific data files into Darwin Core format
- readr\_BMin(): Reads specific data files into Darwin Core format
- readr\_BMont(): Reads specific data files into Darwin Core format
- readr\_Ecd(): Reads specific data files into Darwin Core format
- readr\_Gai(): Reads specific data files into Darwin Core format
- readr\_CAES(): Reads specific data files into Darwin Core format
- readr\_KP(): Reads specific data files into Darwin Core format
- readr\_EcoS(): Reads specific data files into Darwin Core format
- readr\_GeoL(): Reads specific data files into Darwin Core format
- readr\_EaCO(): Reads specific data files into Darwin Core format

- readr\_MABC(): Reads specific data files into Darwin Core format
- readr\_Col(): Reads specific data files into Darwin Core format
- readr\_FSCA(): Reads specific data files into Darwin Core format
- readr\_SMC(): Reads specific data files into Darwin Core format
- readr\_Bal(): Reads specific data files into Darwin Core format
- readr\_Lic(): Reads specific data files into Darwin Core format
- readr\_Arm(): Reads specific data files into Darwin Core format
- readr\_Dor(): Reads specific data files into Darwin Core format
- readr\_MEPB(): Reads specific data files into Darwin Core format
- readr\_BBD(): Reads specific data files into Darwin Core format
- readr\_MPUJ(): Reads specific data files into Darwin Core format
- readr\_STRI(): Reads specific data files into Darwin Core format
- readr\_PALA(): Reads specific data files into Darwin Core format
- readr\_JoLa(): Reads specific data files into Darwin Core format
- readr\_VicWam(): Reads specific data files into 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	<i>Find GBIF, ALA, iDigBio, and SCAN files in a directory</i>
------------	---

---

**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	<i>Import occurrences from GBIF, ALA, iDigBio, and SCAN downloads</i>
-----------	---

---

**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 <code>repoFinder()</code> 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 <code>repoFinder()</code> .

**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:
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(Not run)
```

---

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,
  removeFilterColumns = FALSE,
  filterClean = FALSE
)
```

**Arguments**

<code>data</code>	A data frame or tibble. Occurrence records to use as input.
<code>dontFilterThese</code>	A character vector of flag columns to be ignored in the creation or updating of the <code>.summary</code> column.
<code>removeFilterColumns</code>	Logical. If TRUE all columns starting with "." will be removed in the output data. This only makes sense to use when <code>filterClean = TRUE</code> . Default = FALSE.
<code>filterClean</code>	Logical. If TRUE, the data will be filtered to only those occurrence where <code>.summary = TRUE</code> (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")
```

**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 = 10,
  height = 5,
  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 <code>classInt::classIntervals()</code> . 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 = 10.
height	Numeric. The height, in inches, of the resulting figure. Default = 5.
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 <code>rnaturalearth::ne_countries()</code> . 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

```
# 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"),
)
```

---

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 <code>fileFinder()</code> . 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, 5
  - beesFlagged, 13
  - beesRaw, 19
- \* **prefilter**
  - jbd\_coordinates\_transposed, 58
- ..., 28
- atlasDownloader, 3
  
- bdc::bdc\_basisOfRecords\_notStandard(),  
9, 17
- bdc::bdc\_clean\_names(), 10, 17, 49
- bdc::bdc\_coordinates\_country\_inconsistent(),  
56
- bdc::bdc\_coordinates\_empty(), 9, 17
- bdc::bdc\_coordinates\_outOfRange(), 17
- bdc::bdc\_coordinates\_transposed(), 62
- bdc::bdc\_country\_from\_coordinates(),  
54
- bdc::bdc\_country\_standardized, 60
- bdc::bdc\_country\_standardized(), 9, 17,  
58, 62
- bdc::bdc\_create\_figures(), 61
- bdc::bdc\_eventDate\_empty(), 11, 18
- bdc::bdc\_scientificName\_empty(), 9, 17
- bdc::bdc\_year\_outOfRange(), 11, 18
- BeeBDCQuery, 4
- bees3sp, 5
- beesChecklist, 11
- beesChecklist(), 4, 30
- beesFlagged, 13
- beesRaw, 19
- beesTaxonomy, 23
- beesTaxonomy(), 4, 12, 49
  
- chordDiagramR, 25
- chordDiagramR(), 41
- circlize::chordDiagram(), 26
- circlize::circos.par(), 26
  
- circlize::highlight.sector(), 26
- classInt::classIntervals(), 78
- ColTypeR, 27
- ColTypeR(), 11, 13, 19, 28
- complete.cases(), 40
- CoordinateCleaner::cd\_round(), 10, 18
- CoordinateCleaner::clean\_coordinates(),  
10, 18
- coordUncerFlagR, 28
- coordUncerFlagR(), 10, 18
- countryNameCleanR, 29
- countryOutliers, 30
- countryOutliers(), 10, 11, 18
  
- dataProvTables, 31
- dataSaver, 32
- dateFindR, 33
- diagonAlley, 34
- diagonAlley(), 10, 18
- dirMaker, 36
- dupePlotR, 38
- dupeSummary, 39
- dupeSummary(), 11, 18, 25, 26, 38, 64, 65
  
- fileFinder, 42
- fileFinder(), 51, 79
- flagAbsent, 43
- flagAbsent(), 10, 17
- flagLicense, 44
- flagLicense(), 10, 17
- flagRecorder, 45
- flagSummaryTable, 46
- forcats::fct\_recode(), 69
- formattedCombiner, 47
  
- galah::galah\_config(), 3
- galah::galah\_identify(), 3
- GBIFissues, 48
- GBIFissues(), 10, 17
- ggplot2::ggsave(), 68



harmoniseR, 49  
 harmoniseR(), 10, 17, 49  
 here::here(), 59  
 here::i\_am(), 37  
  
 idMatchR, 50  
 importOccurrences, 51  
 interactiveMapR, 52  
  
 jbd\_CfC\_chunker, 54  
 jbd\_coordCountryInconsistent, 56  
 jbd\_coordCountryInconsistent(), 9, 17  
 jbd\_coordinates\_precision, 57  
 jbd\_coordinates\_precision(), 35  
 jbd\_coordinates\_transposed, 58  
 jbd\_coordinates\_transposed(), 62  
 jbd\_create\_figures, 61  
 jbd\_Ctrans\_chunker, 62  
 jbd\_Ctrans\_chunker(), 9, 17  
  
 manualOutlierFinder, 64  
 manualOutlierFinder(), 41  
  
 PaigeIntegrater, 66  
 plotFlagSummary, 67  
  
 readr::as\_col\_spec(), 28  
 readr::col\_character(), 27, 28  
 readr::col\_integer(), 27, 28  
 readr::col\_logical(), 28  
 readr::cols\_only(), 27  
 readr::read\_csv(), 28  
 readr\_Arm (readr\_BeeBDC), 70  
 readr\_ASP (readr\_BeeBDC), 70  
 readr\_Bal (readr\_BeeBDC), 70  
 readr\_BBD (readr\_BeeBDC), 70  
 readr\_BeeBDC, 70  
 readr\_BMin (readr\_BeeBDC), 70  
 readr\_BMont (readr\_BeeBDC), 70  
 readr\_CAES (readr\_BeeBDC), 70  
 readr\_Col (readr\_BeeBDC), 70  
 readr\_Dor (readr\_BeeBDC), 70  
 readr\_EaCO (readr\_BeeBDC), 70  
 readr\_Ecd (readr\_BeeBDC), 70  
 readr\_EcoS (readr\_BeeBDC), 70  
 readr\_EPEL (readr\_BeeBDC), 70  
 readr\_FSCA (readr\_BeeBDC), 70  
 readr\_Gai (readr\_BeeBDC), 70  
 readr\_GeoL (readr\_BeeBDC), 70  
 readr\_JoLa (readr\_BeeBDC), 70  
 readr\_KP (readr\_BeeBDC), 70  
 readr\_Lic (readr\_BeeBDC), 70  
 readr\_MABC (readr\_BeeBDC), 70  
 readr\_MEPB (readr\_BeeBDC), 70  
 readr\_MPUJ (readr\_BeeBDC), 70  
 readr\_PALA (readr\_BeeBDC), 70  
 readr\_SMC (readr\_BeeBDC), 70  
 readr\_STRI (readr\_BeeBDC), 70  
 readr\_VicWam (readr\_BeeBDC), 70  
 repoFinder, 75  
 repoFinder(), 76  
 repoMerge, 75  
 repoMerge(), 51  
 rnaturalearth::ne\_countries(), 31, 57  
  
 summary(), 45  
 summaryFun, 76  
 summaryFun(), 11, 18, 40, 52  
 summaryMaps, 77  
  
 tempdir(), 24  
  
 USGS\_formatter, 79  
 USGS\_formatter(), 47  
 utils::download.file(), 12, 23