Package: RuHere (via r-universe)

October 20, 2026

Title Flags Spatial Errors in Biological Collection Data Using Specialists' Information

Version 1.1.1

BugReports https://github.com/wevertonbio/RuHere/issues

Description Automatically flags common spatial errors in biological collection data using metadata and specialists' information. RuHere implements a workflow to manage occurrence data through six steps: dataset merging, metadata flagging, validation against expert-derived distribution maps, visualization of flagged records, and sampling bias exploration. It specifically integrates specialist-curated range information to identify geographic errors and introductions that often escape standard automated validation procedures. For details on the methodology, see: Trindade & Caron (2026) <doi:10.64898/2026.02.02.703373>.

Imports Rcpp, terra, data.table, faunabr (>= 1.1.2), florabr (>= 1.4.0), jsonlite, rgbif, rredlist, stringi, RPostgreSQL, DBI, ridigbio, fields, ggplot2, mapview, sf, ggnewscale, httr

Suggests pbapply, knitr, R.utils, rmarkdown, CoordinateCleaner

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.5)

LazyData true

LinkingTo Rcpp (>= 1.1.0), RcppArmadillo (>= 15.0.2.2)

VignetteBuilder knitr

URL https://wevertonbio.github.io/RuHere/

NeedsCompilation yes

Author Weverton C. F. Trindade [aut, cre] (ORCID: <https://orcid.org/0000-0003-2045-4555>), Fernanda S. Caron [aut] (ORCID: <https://orcid.org/0000-0002-1884-6157>)

Maintainer Weverton C. F. Trindade <wevertonf1993@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-10-20 13:20:20 UTC

RemoteUrl https://github.com/cran/RuHere

RemoteRef HEAD

RemoteSha 3a0b5b9502a9c2eeceeead29749c1edff7d97565

Contents

atlantic_amphibians . . . 4
available_datasets . . . 4
bien_here . . . 6
bind_here . . . 7
check_countries . . . 8
check_states . . . 9
cite_datapaper . . . 10
count_flags . . . 11
country_dictionary . . . 13
country_from_coords . . . 14
create_metadata . . . 15
cultivated . . . 17
fake_data . . . 18
faunabr_here . . . 19
fix_countries . . . 20
fix_states . . . 22
flag_bien . . . 24
flag_colors . . . 26
flag_consensus . . . 26
flag_cultivated . . . 28
flag_duplicates . . . 29
flag_env_moran . . . 30
flag_faunabr . . . 34
flag_florabr . . . 36
flag_fossil . . . 38
flag_geo_moran . . . 39
flag_inaturalist . . . 42
flag_iucn . . . 43
flag_names . . . 45
flag_wcvp . . . 46
flag_year . . . 47
florabr_here . . . 48
format_columns . . . 49
get_bien . . . 51
get_datapaper . . . 54
get_env_bins . . . 55
get_idigbio . . . 57

get_specieslink . . . 58
ggmap_here . . . 61
ggrid_here . . . 64
import_gbif . . . 65
inventory_completeness . . . 67
iucn_here . . . 69
map_here . . . 71
moranfast . . . 73
occ_bien . . . 74
occ_flagged . . . 75
occ_gbif . . . 76
occ_idig . . . 77
occ_splink . . . 77
occurrences . . . 78
plot_env_bins . . . 79
prepare_gbif_download . . . 80
prepared_metadata . . . 82
puma_atlanticr . . . 83
relocate_after . . . 83
remove_accent . . . 84
remove_flagged . . . 85
remove_invalid_coordinates . . . 86
request_gbif . . . 88
request_gbif_specieslist . . . 89
richness_here . . . 91
set_gbif_credentials . . . 93
set_iucn_credentials . . . 94
set_specieslink_credentials . . . 95
spatial_kde . . . 97
spatialize . . . 99
standardize_countries . . . 100
standardize_states . . . 102
states . . . 103
states_dictionary . . . 104
states_from_coords . . . 105
summarize_flags . . . 107
thin_env . . . 109
thin_geo . . . 110
wcvp_here . . . 112
world . . . 113
worldclim . . . 114

Index **115**

atlantic_amphibians *Amphibian communities from the Atlantic Forest*

Description

A data set of amphibian communities from the Atlantic Forests of South America sourced from Vancine et al. (2018).

Usage

```
atlantic_amphibians
```

Format

A `data.table` or `data.frame` with 8,254 rows and 3 columns:

species Character. Scientific name of the virtual species.

decimalLongitude Numeric. Georeferenced longitude in decimal degrees.

decimalLatitude Numeric. Georeferenced latitude in decimal degrees.

References

Vancine et al. 2018. ATLANTIC AMPHIBIANS: a data set of amphibian communities from the Atlantic Forests of South America. Ecology, 99(7), 1692-1692. doi:10.1002/ecy.2392

See Also

`inventory_completeness()`

Examples

```
# First rows
head(atlantic_amphibians)
```

available_datasets *Check the available distribution datasets for a set of species*

Description

This function checks which datasets contain distributional information for a given set of species, based on expert-curated sources. It searches the selected datasets and reports whether each species has available distribution data.

Usage

```
available_datasets(
  data_dir,
  species,
  datasets = "all",
  return_distribution = FALSE
)
```

Arguments

data_dir	(character) directory path where the datasets were saved. See *Details* for more information.
species	(character) vector with the species names to be checked for the availability of distributional information.
datasets	(character) vector indicating which datasets to search. Options are "all", "florabr", "wcvp", "iucn", "bien", and "faunabr". Default searches all datasets.
return_distribution	(logical) whether to return the spatial objects (SpatVector) representing the distribution regions of the species found in the selected datasets. Default is FALSE.

Details

The distribution datasets can be obtained using the functions florabr_here(), wcvp_here(), bien_here(), and faunabr_here(), which download and prepare the corresponding sources for use in RuHere.

Value

If return_distribution = FALSE, a data.frame containing the species names and the datasets where distributional information is available. If return_distribution = TRUE, it also returns a list containing the SpatVector objects representing the species ranges.

Examples

```
# Set directory where datasets were saved
# Here, we'll use the directory where the example datasets are stored
datadir <- system.file("extdata", "datasets",  package = "RuHere")
# Check available datasets
d <- available_datasets(data_dir = datadir,
                        species = c("Araucaria angustifolia",
                                    "Handroanthus serratifolius",
                                    "Cyanocorax caeruleus"))
# Check available datasets and return distribution
d2 <- available_datasets(data_dir = datadir,
                         species = c("Araucaria angustifolia",
                                     "Handroanthus serratifolius",
                                     "Cyanocorax caeruleus"),
                         return_distribution = TRUE)
```

bien_here *Download species distribution information from BIEN*

Description

This function downloads distribution information from the BIEN database, required for filtering occurrence records using specialists' range information via the `flag_bien()` function. Unlike the other range products supported by RuHere (IUCN, WCVP, florabr, faunabr), BIEN range polygons are derived from ecological niche models rather than direct taxonomic curation, and should be interpreted as model-based hypotheses of potential distribution.

Usage

```
bien_here(
  data_dir,
  species,
  synonyms = NULL,
  overwrite = TRUE,
  progress_bar = FALSE,
  verbose = TRUE
)
```

Arguments

`data_dir`	(character) directory to save the data downloaded from BIEN.
`species`	(character) a vector of species names for which to retrieve distribution information.
`synonyms`	(data.frame) an optional data.frame containing synonyms of the target species. The first column must contain the target species names, and the second column their corresponding synonyms. Default is `NULL`. See details for more information.
`overwrite`	(logical) whether to overwrite existing files. Default is `TRUE`.
`progress_bar`	(logical) whether to display a progress bar during processing. If TRUE, the 'pbapply' package must be installed. Default is `FALSE`.
`verbose`	(logical) whether to display progress messages. Default is `TRUE`.

Details

This function uses the `BIEN::BIEN_ranges_load_species()` function to retrieve polygons representing the distribution ranges of species available in the BIEN database. These polygons are derived from ecological niche models (Maitner et al., 2018) rather than from direct taxonomic curation.

Because taxonomic information in BIEN may be outdated, you can optionally provide a table of synonyms to broaden the search. The synonyms data.frame should have the accepted species in the first column and their synonyms in the second. See `RuHere::synonys` for an example.

Value

A data frame indicating whether the polygon(s) representing the species range are available in BIEN. If the range is available, a GeoPackage file (.gpkg) is saved in `data_dir/bien`. The file name corresponds to the species name, with an underscore ("_") replacing the space between the genus and the specific epithet.

Examples

```
# Define a directory to save the data
data_dir <- tempdir() # Here, a temporary directory

# Download species distribution information from BIEN
bien_here(data_dir = data_dir, species = "Handroanthus serratifolius")
```

`bind_here` *Bind occurrences after standardizing columns*

Description

Combines multiple occurrence data frames (for example, from GBIF, SpeciesLink, BIEN, or iDigBio) into a single standardized dataset. This is particularly useful after using `format_columns()` to ensure column compatibility across data sources.

Usage

```
bind_here(..., fill = FALSE, ignore.attr = FALSE)
```

Arguments

`...`	(data.frame) two or more data frames with occurrence records to combine.
`fill`	(logical) whether to fills missing columns with `NA`. Default is FALSE.
`ignore.attr`	(logical) whether to allow binding columns with different attributes (e.g. class). Default is FALSE.

Details

When `fill = TRUE`, columns not shared among the input data frames are added and filled with `NA`, ensuring that all columns align before binding. Internally, this function uses `data.table::rbindlist()` for efficient row binding.

Value

A `data.frame` containing all occurrence records combined.

Examples

```
# Import and standardize GBIF
data("occ_gbif", package = "RuHere") #Import data example
gbif_standardized <- format_columns(occ_gbif, metadata = "gbif")
# Import and standardize SpeciesLink
data("occ_splink", package = "RuHere") #Import data example
splink_standardized <- format_columns(occ_splink, metadata = "specieslink")
# Import and standardize BIEN
data("occ_bien", package = "RuHere") #Import data example
bien_standardized <- format_columns(occ_bien, metadata = "bien")
# Import and standardize idigbio
data("occ_idig", package = "RuHere") #Import data example
idig_standardized <- format_columns(occ_idig, metadata = "idigbio")
# Merge all
all_occ <- bind_here(gbif_standardized, splink_standardized,
                     bien_standardized, idig_standardized)
```

---

check_countries *Check if the records fall in the country assigned in the metadata*

Description

Check if the records fall in the country assigned in the metadata

Usage

```
check_countries(
  occ,
  long = "decimalLongitude",
  lat = "decimalLatitude",
  country_column,
  distance = 5,
  try_to_fix = FALSE,
  progress_bar = FALSE,
  verbose = TRUE
)
```

Arguments

occ	(data.frame) a dataset with occurrence records, preferably with country information standardized using `standardize_countries()`.
long	(character) column name with longitude. Default is 'decimalLongitude'.
lat	lat (character) column name with latitude. Default is 'decimalLatitude'.
country_column	(character) column name containing the country information.
distance	(numeric) maximum distance (in kilometers) a record can fall outside the country assigned in the `country_column`. Default is `5`.

try_to_fix	(logical) whether to check if coordinates are inverted or transposed (see fix_countries() for details). If TRUE, coordinates identified as inverted or transposed will be corrected. Default is FALSE.
progress_bar	(logical) whether to display a progress bar during processing. If TRUE, the 'pbapply' package must be installed. Default is FALSE.
verbose	(logical) whether to print messages about function progress. Default is TRUE.

Value

The original occ data.frame with an additional column (correct_country) indicating whether each record falls within the country specified in the metadata (TRUE) or not (FALSE). As with all other flagging functions in RuHere, TRUE indicates that the record passed this test and is eligible for retention; FALSE indicates it failed and is flagged as potentially problematic, and can be removed using remove_flagged().

Examples

```
# Load example data
data("occurrences", package = "RuHere") #Import data example
# Standardize country names
occ_country <- standardize_countries(occ = occurrences,
                                     return_dictionary = FALSE)
# Check whether records fall within assigned countries
occ_country_checked <- check_countries(occ = occ_country,
                                       country_column = "country_suggested")
```

---

check_states *Check if the records fall in the state assigned in the metadata*

Description

Check if the records fall in the state assigned in the metadata

Usage

```
check_states(
  occ,
  long = "decimalLongitude",
  lat = "decimalLatitude",
  state_column,
  distance = 5,
  try_to_fix = FALSE,
  progress_bar = FALSE,
  verbose = TRUE
)
```

Arguments

occ	(data.frame) a dataset with occurrence records, preferably with country information standardized using standardize_states().
long	(character) column name with longitude. Default is 'decimalLongitude'.
lat	lat (character) column name with latitude. Default is 'decimalLatitude'.
state_column	(character) column name containing the state information.
distance	(numeric) maximum distance (in kilometers) a record can fall outside the state assigned in the state_column. Default is 5.
try_to_fix	(logical) whether to check if coordinates are inverted or transposed (see fix_states() for details). If TRUE, coordinates identified as inverted or transposed will be corrected. Default is FALSE.
progress_bar	(logical) whether to display a progress bar during processing. If TRUE, the 'pbapply' package must be installed. Default is FALSE.
verbose	(logical) whether to print messages about function progress. Default is TRUE.

Value

The original occ data.frame with an additional column (correct_state) indicating whether each record falls within the state specified in the metadata (TRUE) or not (FALSE). As with all other flagging functions in RuHere, TRUE indicates that the record passed this test and is eligible for retention; FALSE indicates it failed and is flagged as potentially problematic, and can be removed using remove_flagged().

Examples

```
# Load example data
data("occurrences", package = "RuHere") #Import data example
# Subset occurrences for Araucaria angustifolia
occ <- occurrences[occurrences$species == "Araucaria angustifolia", ]
# Standardize country names
occ_country <- standardize_countries(occ = occ,
                                     return_dictionary = FALSE)
# Standardize state names
occ_state <- standardize_states(occ = occ_country,
                                country_column = "country_suggested",
                                return_dictionary = FALSE)
# Check whether records fall within assigned states
occ_state_checked <- check_states(occ = occ_state,
                                  state_column = "state_suggested")
```

---

cite_datapaper *Cite data papers retrieved from get_datapaper*

Description

Identifies the data sources present in the output of get_datapaper() and prints the appropriate bibliographic citations for use in academic and technical publications.

Usage

```
cite_datapaper(data)
```

Arguments

data	A data.frame or data.table obtained from get_datapaper(), containing the data_source column.

Value

Prints the formatted references to the console and invisibly returns a named character vector with the citations for the detected data sources.

Examples

```
## Not run:
# 1. Download data
dados <- get_datapaper(datapaper = "dryflor")

# 2. View citations
cite_datapaper(dados)

## End(Not run)
```

---

count_flags *Count the number of flags for each record*

Description

This function returns the original data frame with an additional column summarizing the total number of flags triggered by each record.

Usage

```
count_flags(
  occ = NULL,
  flagged_dir = NULL,
  output_format = ".gz",
  flags = "all",
  additional_flags = NULL
)
```

Arguments

occ	(data.frame or data.table) a dataset containing occurrence records that has been processed by one or more flagging functions. See *Details* for available flag types.

`flagged_dir` (character) optional path to a directory containing files with flagged records saved using the `remove_flagged()` function. Default is `NULL`.

`output_format` (character) output format used to read the removed records. Options are `".csv"` or `".gz"`. Only used when `flagged_dir` is not `NULL`. Default is `".gz"`.

`flags` (character) the flags to be summarized. Use `"all"` to display all available flags. See *Details* for all options. Default is `"all"`.

`additional_flags` (character) an optional named character vector with the names of additional logical columns to be used as flags. Default is `NULL`.

Details

This function expects an occurrence dataset that has already been processed by one or more flagging routines from **RuHere** or related packages such as **CoordinateCleaner**. Any logical column in `occ` can be used as a flag.

The following built-in flag names are recognized:

From RuHere: `correct_country`, `correct_state`, `cultivated`, `florabr`, `faunabr`, `wcvp`, `iucn`, `bien`, `duplicated`, `thin_geo`, `thin_env`, `consensus`

From CoordinateCleaner : `.val`, `.equ`, `.zer`, `.cap`, `.cen`, `.sea`, `.urb`, `.otl`, `.gbf`, `.inst`, `.aohi`

Users may also supply additional logical columns using `additional_flags`.

Value

The original data frame with an additional column summarizing the total number of flags triggered by each record.

Examples

```
# Load example data
data("occ_flagged", package = "RuHere")

# Count flags
sum_flags <- count_flags(occ = occ_flagged)

# Check the distribution of flags per record
table(sum_flags$total_flags)

# Plot histogram
hist(sum_flags$total_flags,
     main = "Distribution of Flags",
     xlab = "Number of Flags",
     col = "steelblue",
     breaks = seq(-0.5, max(sum_flags$total_flags) + 0.5, by = 1))
```

`country_dictionary` *Country dictionary for standardizing country names and codes*

Description

`country_dictionary` provides a set of lookup tables used to standardize country names and country codes in occurrence datasets.

The dictionary is built from `rnaturalearthdata::map_units110` and consolidates a wide variety of country name variants (in several languages and formats), as well as multiple coding systems, into a single suggested standardized name.

This object is used internally by functions that clean or harmonize country fields, ensuring that country names in occurrence datasets (e.g., `"Brasil"`,`"brasil"`, `"BR"`, `"BRA"`, `"République Française"`) are all mapped consistently to a single standardized form (`"brazil"`, `"france"`, etc.).

Usage

```
country_dictionary
```

Format

A named list of two data frames:

`country_name` A data frame with two columns:

- `country_name` Character. Lowercased and accent-stripped country name variants (from multiple `rnaturalearthdata` fields such as *name*, *name_long*, *abbrev*, *formal_en*, and alternative names in several languages).
- `country_suggested` Character. The standardized country name, derived from the `name` column of `map_units110`, also lowercased and accent-stripped.

`country_code` A data frame with two columns:

- `country_code` Character. Country codes from several systems, including ISO-2, ISO-3, FIPS, postal codes, and others, after filtering invalid or ambiguous codes.
- `country_suggested` Character. The standardized country name corresponding to each code.

Details

The dictionary is generated by:

- extracting multiple name and code fields from `rnaturalearthdata::map_units110`,
- converting names to lowercase and removing accents,
- converting codes to uppercase,
- removing invalid or ambiguous codes (e.g., `-99`, `"J"`, various country mismatches),
- and ensuring uniqueness across all entries.

Examples

```
data(country_dictionary)

head(country_dictionary$country_name)
head(country_dictionary$country_code)
```

`country_from_coords` *Extract country from coordinates*

Description

Extracts the country for each occurrence record based on coordinates.

Usage

```
country_from_coords(
  occ,
  long = "decimalLongitude",
  lat = "decimalLatitude",
  country_column = NULL,
  from = "all",
  output_column = "country_xy",
  append_source = FALSE
)
```

Arguments

`occ`	(data.frame) a dataset with occurrence records, preferably standardized using `format_columns()`.
`long`	(character) column name with longitude. Default is 'decimalLongitude'.
`lat`	(character) column name with latitude. Default is 'decimalLatitude'.
`country_column`	(character) the column name containing the country. Only applicable if `from = na_only`. Default is NULL.
`from`	(character) whether to extract the country for all records ('all') or only for records missing country information ('na_only'). If 'na_only', you must provide the name of the column with country information. Default is 'all'.
`output_column`	(character) column name created in `occ` to store the countries extracted. Default is 'country_xy'.
`append_source`	(logical) whether to create a new column in `occ` called 'country_source', which indicates whether the country was derived from coordinates. Default is FALSE.

Details

The countries are extracted from coordinates using a map retrieved from `rnaturalearthdata::map_units110`.

Value

The original occ data.frame with an additional column containing the countries extracted from coordinates.

Examples

```
# Import and standardize GBIF
data("occ_gbif", package = "RuHere") #Import data example
gbif_standardized <- format_columns(occ_gbif, metadata = "gbif")
gbif_countries <- country_from_coords(occ = gbif_standardized)
```

---

create_metadata *Create metadata template*

Description

This function creates a metadata template to be used in `format_columns()` for formatting and standardizing column names and classes in occurrence datasets. All column names specified as arguments must be present in the occ dataset.

If you obtained data from GBIF, SpeciesLink, BIEN or iDigBio using the functions provided in the RuHere package, you do not need to use this function, as the package already includes metadata templates for these datasets.

Usage

```
create_metadata(
  occ,
  scientificName,
  decimalLongitude,
  decimalLatitude,
  collectionCode = NA,
  catalogNumber = NA,
  coordinateUncertaintyInMeters = NA,
  elevation = NA,
  country = NA,
  stateProvince = NA,
  municipality = NA,
  locality = NA,
  year = NA,
  eventDate = NA,
  recordedBy = NA,
  identifiedBy = NA,
  basisOfRecord = NA,
  occurrenceRemarks = NA,
  habitat = NA,
  datasetName = NA,
  datasetKey = NA,
```

```
  key = NA
)
```

Arguments

occ	(data.frame or data.table) a dataset with occurrence records to be standardized.
scientificName	(character) column name in occ with the scientific name of the species.
decimalLongitude	(character) column name in occ with the longitude.
decimalLatitude	(character) column name in occ with the latitude.
collectionCode	(character) an optional column name in occ with the collection code.
catalogNumber	(character) an optional column name in occ with the catalog number.
coordinateUncertaintyInMeters	(character) an optional column name with the coordinate uncertainty in meters.
elevation	(character) an optional column name with the elevation information.
country	(character) an optional column name with the country of the record.
stateProvince	(character) an optional column name with the state or province of the record.
municipality	(character) an optional column name with the municipality of the record.
locality	(character) an optional column name with the locality description.
year	(character) an optional column name with the year when the occurrence was recorded.
eventDate	(character) an optional column name with the event date.
recordedBy	(character) an optional column name with the name of the collector or recorder.
identifiedBy	(character) an optional column name with the name of the identifier.
basisOfRecord	(character) an optional column name with the basis of record.
occurrenceRemarks	(character) an optional column name with remarks about the occurrence.
habitat	(character) an optional column name with the habitat description.
datasetName	(character) an optional column name with the dataset name.
datasetKey	(character) an optional column name with the dataset key.
key	(character) an optional column name with the unique occurrence identifier.

Value

A data.frame containing a metadata template that can be directly used in the format_columns() function.

Examples

```
# Load data example
# Occurrences of Puma concolor from the atlanticr R package
data("puma_atlanticr", package = "RuHere")
# Create metadata to standardize the occurrences
puma_metadata <- create_metadata(occ = puma_atlanticr,
                                 scientificName = "actual_species_name",
                                 decimalLongitude = "longitude",
                                 decimalLatitude = "latitude",
                                 elevation = "altitude",
                                 country = "country",
                                 stateProvince = "state",
                                 municipality = "municipality",
                                 locality = "study_location",
                                 year = "year_finish",
                                 habitat = "vegetation_type",
                                 datasetName = "reference")
# Now, we can use this metadata to standardize the columns
puma_occ <- format_columns(occ = puma_atlanticr, metadata = puma_metadata,
                           binomial_from = "actual_species_name",
                           data_source = "atlanticr")
```

---

cultivated *Dictionary of terms used to flag cultivated individuals*

Description

`cultivated` is a list of character vectors containing keywords used to identify whether an occurrence record refers to cultivated or non-cultivated individuals.

This object is used internally by `flag_cultivated()` to scan occurrence fields (such as notes, habitat descriptions, or remarks) and classify records as *cultivated* or *not cultivated* based on textual patterns.

The list combines terms from `plantR` (`plantR:::cultivated` and `plantR:::notCultivated`) with additional multilingual variants commonly found in herbarium metadata.

Usage

```
cultivated
```

Format

A named list with two elements:

`cultivated` Character vector. Terms that indicate an individual is cultivated. Imported from `plantR:::cultivated`.

`not_cultivated` Character vector. Terms suggesting an individual is *not* cultivated (e.g., "not cultivated", "not planted", "no plantada", "no cultivada"), including terms from `plantR:::notCultivated`.

Details

These terms are matched case-insensitively after text cleaning (e.g., lowercasing and accent removal).

References

de Lima, Renato AF, et al. plantR: An R package and workflow for managing species records from biological collections. **Methods in Ecology and Evolution**, 14.2 (2023): 332-339.

See Also

`flag_cultivated`

Examples

```
data(cultivated)

cultivated$cultivated
cultivated$not_cultivated
```

`fake_data` *Fake occurrence data for testing coordinate validation functions*

Description

`fake_data` is a synthetic dataset created for testing functions that validate and correct country- or state-level geographic coordinates.

Controlled coordinate errors were introduced (e.g., inverted signs, swapped values, combinations of swaps and inversions) to simulate common georeferencing mistakes.

This dataset is intended for automated testing of functions such as `check_countries()` and `check_states()`.

Usage

```
fake_data
```

Format

A data frame with the same structure as `all_occ`, containing occurrence records with intentionally manipulated coordinates. An additional column `data_source = "fake_data"` identifies these records.

Details

The coordinate errors include:

- **Inverted longitude**: multiplying longitude by -1.
- **Inverted latitude**: multiplying latitude by -1.
- **Both coordinates inverted**.
- **Swapped coordinates**: (`lon`, `lat`) → (`lat`, `lon`).
- **Swapped + inverted** in four combinations:
  - swapped only,
  - swapped + inverted longitude,
  - swapped + inverted latitude,
  - swapped + both inverted.

Examples

```
data(fake_data)
```

`faunabr_here` *Download the latest version of the Fauna do Brazil (Taxonomic Catalog of the Brazilian Fauna)*

Description

This function downloads the Taxonomic Catalog of the Brazilian Fauna database, which is required for filtering occurrence records using specialists' range information via the `flag_faunabr()` function.

Usage

```
faunabr_here(
  data_dir,
  data_version = "latest",
  get_fixed_version = TRUE,
  solve_discrepancy = TRUE,
  overwrite = TRUE,
  remove_files = TRUE,
  verbose = TRUE
)
```

Arguments

data_dir	(character) a directory to save the data downloaded from Fauna do Brazil.
data_version	(character) version of the Fauna do Brazil database to download. Use "latest" to get the most recent version, which is updated frequently. Alternatively, specify an older version (e.g., data_version="1.2"). Default value is "latest".
get_fixed_version	(logical) If TRUE, download the already-merged version 1.57 from Zenodo instead of downloading and processing an IPT archive. Default is TRUE.
solve_discrepancy	(logical) whether to resolve inconsistencies between species and subspecies information. When set to TRUE (default), species information is updated based on unique data from subspecies. For example, if a subspecies occurs in a certain state, it implies that the species also occurs in that state.
overwrite	(logical) If TRUE, data is overwritten. Default is TRUE.
remove_files	(logical) whether to remove the downloaded files used in building the final dataset. Default is TRUE.
verbose	(logical) whether to display messages during function execution. Set to TRUE to enable display, or FALSE to run silently. Default is TRUE.

Value

A message indicating that the data were successfully saved in the directory specified by data_dir.

Examples

```
## Not run:
# Define a directory to save the data
data_dir <- tempdir() # Here, a temporary directory

# Download the latest version of the Flora e Funga do Brazil database
faunabr_here(data_dir = data_dir)

## End(Not run)
```

---

fix_countries *Identify and correct coordinates based on country information*

Description

This function identifies and correct inverted and transposed coordinates based on country information

Usage

```
fix_countries(
  occ,
  long = "decimalLongitude",
  lat = "decimalLatitude",
  country_column,
  correct_country = "correct_country",
  distance = 5,
  progress_bar = FALSE,
  verbose = TRUE
)
```

Arguments

occ	(data.frame) a dataset with occurrence records, preferably with country information checked using check_countries().
long	(character) column name with longitude. Default is 'decimalLongitude'.
lat	lat (character) column name with latitude. Default is 'decimalLatitude'.
country_column	(character) name of the column containing the country information.
correct_country	(character) name of the column with logical value indicating whether each record falls within the country specified in the metadata. Default is 'correct_country'. See details.
distance	(numeric) maximum distance (in kilometers) a record can fall outside the country assigned in the country_column. Default is 5.
progress_bar	(logical) whether to display a progress bar during processing. If TRUE, the 'pbapply' package must be installed. Default is FALSE.
verbose	(logical) whether to print messages about function progress. Default is TRUE.

Details

The function checks and corrects coordinate errors in occurrence records by testing whether each point falls within the expected country polygon (from RuHere's internal world map).

The input occurrence data must contain a column (specified in the correct_country argument) with logical values indicating which records to check and fix — only those marked as FALSE will be processed. This column can be obtained by running the check_countries() function.

It runs a series of seven tests to detect common issues such as **inverted** signs or **swapped** latitude/longitude values. Inverted coordinates have their signs flipped (e.g., -45 instead of 45), placing the point in the opposite hemisphere, while swapped coordinates have latitude and longitude values exchanged (e.g., -47, -15 instead of -15, -47).

For each test, country borders are buffered by distance km to account for minor positional errors.

The type of issue (or "correct") is recorded in a new column, country_issues. Records that match their assigned country after any correction are updated accordingly, while remaining mismatches are labeled "incorrect".

This function can be used internally by check_countries() to automatically identify and fix common coordinate errors.

Value

The original `occ` data.frame with the coordinates in the `long` and `lat` columns corrected, and an additional column (`country_issues`) indicating whether the coordinates are:

- **correct**: the record falls within the assigned country;
- **inverted**: longitude and/or latitude have reversed signs;
- **swapped**: longitude and latitude are transposed (i.e., each appears in the other's column). **incorrect**: the record falls outside the assigned country and could not be corrected.

Examples

```
# Load example data
data("occurrences", package = "RuHere") # Import example data

# Standardize country names
occ_country <- standardize_countries(occ = occurrences,
                                     return_dictionary = FALSE)

# Check whether records fall within the assigned countries
occ_country_checked <- check_countries(occ = occ_country,
                                       country_column = "country_suggested")

# Fix records with incorrect or misassigned countries
occ_country_fixed <- fix_countries(occ = occ_country_checked,
                                   country_column = "country_suggested")
```

---

fix_states *Identify and correct coordinates based on state information*

Description

This function identifies and correct inverted and transposed coordinates based on state information.

Usage

```
fix_states(
  occ,
  long = "decimalLongitude",
  lat = "decimalLatitude",
  state_column,
  correct_state = "correct_state",
  distance = 5,
  progress_bar = FALSE,
  verbose = TRUE
)
```

Arguments

occ	(data.frame) a dataset with occurrence records, preferably with state information checked using state_countries().
long	(character) column name with longitude. Default is 'decimalLongitude'.
lat	lat (character) column name with latitude. Default is 'decimalLatitude'.
state_column	(character) name of the column containing the state information.
correct_state	(character) name of the column with logical value indicating whether each record falls within the state specified in the metadata. Default is 'correct_state'. See details.
distance	(numeric) maximum distance (in kilometers) a record can fall outside the state assigned in the state_column. Default is 5.
progress_bar	(logical) whether to display a progress bar during processing. If TRUE, the 'pbapply' package must be installed. Default is FALSE.
verbose	(logical) whether to print messages about function progress. Default is TRUE.

Details

The function checks and corrects coordinate errors in occurrence records by testing whether each point falls within the expected state polygon (from RuHere's internal world map).

The input occurrence data must contain a column (specified in the correct_state argument) with logical values indicating which records to check and fix — only those marked as FALSE will be processed. This column can be obtained by running the check_states() function.

It runs a series of seven tests to detect common issues such as **inverted** signs or **swapped** latitude/longitude values. Inverted coordinates have their signs flipped (e.g., -45 instead of 45), placing the point in the opposite hemisphere, while swapped coordinates have latitude and longitude values exchanged (e.g., -47, -15 instead of -15, -47).

For each test, state borders are buffered by distance km to account for minor positional errors.

The type of issue (or "correct") is recorded in a new column, state_issues. Records that match their assigned state after any correction are updated accordingly, while remaining mismatches are labeled "incorrect".

This function can be used internally by check_states() to automatically identify and fix common coordinate errors.

Value

The original occ data.frame with the coordinates in the long and lat columns corrected, and an additional column (state_issues) indicating whether the coordinates are:

- **correct**: the record falls within the assigned state;
- **inverted**: longitude and/or latitude have reversed signs;
- **swapped**: longitude and latitude are transposed (i.e., each appears in the other's column). **incorrect**: the record falls outside the assigned state and could not be corrected.

Examples

```
# Load example data
data("occurrences", package = "RuHere") # Import example data
# Subset records of Araucaria
occ <- occurrences[occurrences$species == "Araucaria angustifolia", ]
# Standardize country names
occ_country <- standardize_countries(occ = occ,
                                     return_dictionary = FALSE)

# Standardize state names
occ_state <- standardize_states(occ = occ_country,
                                country_column = "country_suggested",
                                return_dictionary = FALSE)

# Check whether records fall within the assigned states
occ_states_checked <- check_states(occ = occ_state,
                                   state_column = "state_suggested")

# Fix records with incorrect or misassigned states
occ_states_fixed <- fix_states(occ = occ_states_checked,
                               state_column = "state_suggested")
```

---

flag_bien *Identify records outside natural ranges according to BIEN*

Description

Flags occurrence records based on known distribution data from the Botanical Information and Ecology Network (BIEN) data. This function checks if an occurrence point for a given species falls within its documented distribution, allowing for user-defined buffers around the region. Records are flagged as TRUE if they fall inside the documented distribution (plus optional buffer) for the species in the BIEN dataset.

Usage

```
flag_bien(
  data_dir,
  occ,
  species = "species",
  long = "decimalLongitude",
  lat = "decimalLatitude",
  buffer = 10,
  progress_bar = FALSE,
  verbose = TRUE
)
```

Arguments

data_dir	(character) **Required** directory path where the BIEN data is saved
occ	(data.frame or data.table) a data frame containing the occurrence records to be flagged. Must contain columns for species, longitude, and latitude.
species	(character) the name of the column in occ that contains the species scientific names. Default is "species".
long	(character) the name of the column in occ that contains the longitude values. Default is "decimalLongitude".
lat	(character) the name of the column in occ that contains the latitude values. Default is "decimalLatitude".
buffer	(numeric) buffer distance (in kilometers) to be applied around the region of distribution. Default is 20 km.
progress_bar	(logical) whether to display a progress bar during processing. If TRUE, the 'pbapply' package must be installed. Default is FALSE.
verbose	(logical) if TRUE, prints messages about the progress and the number of species being checked. Default is TRUE.

Value

A data.frame that is the original occ data frame augmented with a new column named bien_flag. This column is logical (TRUE/FALSE) indicating whether the record falls within the expected distribution (plus buffer) based on the BIEN data. As with all other flagging functions in RuHere, TRUE indicates that the record passed this test and is eligible for retention; FALSE indicates it failed and is flagged as potentially problematic, and can be removed using remove_flagged(). Records for species not found in the BIEN data will have NA in the bien_flag column.

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Filter occurrences for golden trumpet tree
occ <- occurrences[occurrences$species == "Handroanthus serratifolius", ]
# Set folder where distributional datasets were saved
# Here, just a sample provided in the package
# You must run 'bien_here()' beforehand to download the necessary data files
dataset_dir <- system.file("extdata/datasets", package = "RuHere")

# Flag records using BIEN specialist information
occ_bien <- flag_bien(data_dir = dataset_dir, occ = occ)
```

flag_colors *Color palette for flagged records*

Description

flag_colors is a named character vector defining the default colors used to plot occurrence records flagged with mapview_here().

Usage

```
flag_colors
```

Format

A named character vector where:

names Flag labels corresponding to categories generated by the various flag_* and checking functions.

values Hex color codes or standard R color names used for plotting.

See Also

mapview_here

Examples

```
data(flag_colors)

# View all flag categories and their colors
flag_colors
```

flag_consensus *Get consensus across multiple flags*

Description

This functions creates a new column representing the consensus across multiple flag columns. The consensus can be computed in two ways:

- "all_true": A record passes the consensus (TRUE) only if **all** specified flags are TRUE.
- "any_true": A record passes the consensus (TRUE) if **at least one** specified flag is TRUE.

Usage

```
flag_consensus(
  occ,
  flags,
  consensus_rule = "all_true",
  flag_name = "consensus_flag",
  remove_flag_columns = FALSE
)
```

Arguments

occ	(data.frame or data.table) a dataset with occurrence records that has been processed by two or more flagging functions.
flags	(character) a string vector with the names of the flags to be used in the consensus evaluation. See details for see the options.
consensus_rule	(character) A string specifying how the consensus should be computed. Options are "all_true" (record passes the consensus only when **all** flags are TRUE) or "any_true" (record passes the consensus when **at least one** flag is TRUE). Default is "all_true"
flag_name	(character) name of the column that will store the consensus result. Default is "consensus_flag".
remove_flag_columns	(logical) whether to remove the original flag columns specified in flags from the final output. Default is FALSE.

Details

The following flags are available: correct_country, correct_state, cultivated, fossil, inaturalist, faunabr, florabr, wcvp, iucn, bien, duplicated, thin_geo, thin_env, year, .val, .equ, .zer, .cap, .cen, .sea, .urb, .otl, .gbf, .inst, and .aohi. Flags may be combined from any category (metadata-based, range-based, or thinning-related).

Value

The original occ with an additional logical column defined by flag_name, indicating the consensus result based on the selected consensus_rule. As with all other flagging functions in RuHere, TRUE indicates that the record passed the consensus criterion and is eligible for retention, while FALSE indicates it failed and is flagged as potentially problematic.

Examples

```
# Load example data
data("occ_flagged", package = "RuHere")

# Get consensus using florabr, wcvp, and iucn flags
 # Valid (TRUE) only when all flags are TRUE
occ_consensus_all <- flag_consensus(occ = occ_flagged,
                                    flags = c("florabr", "wcvp", "iucn"),
```

```
                                    consensus_rule = "all_true")
# Valid (TRUE) when at least one flag is TRUE
occ_consensus_any <- flag_consensus(occ = occ_flagged,
                                    flags = c("florabr", "wcvp", "iucn"),
                                    consensus_rule = "any_true")
```

`flag_cultivated` *Flag occurrence records of cultived individuals*

Description

This function identifies records of cultivated individuals based on record description.

Usage

```
flag_cultivated(
  occ,
  columns = c("occurrenceRemarks", "habitat", "locality"),
  cultivated_terms = NULL,
  not_cultivated_terms = NULL
)
```

Arguments

`occ`	(data.frame) a data frame containing the occurrence records to be examined, preferably standardized using `format_columns()`. Must contain the columns specified in `columns`.
`columns`	columns (character) vector of column names in `occ` where the function will search for cultivated-related expressions. Default is `c("occurrenceRemarks", "habitat", "locality")`.
`cultivated_terms`	(character) optional vector of additional terms that indicate a cultivated individual. Default is NULL, meaning it will use the cultivated-related expressions available in `RuHere::cultivated$cultivated`.
`not_cultivated_terms`	(character) optional vector of additional terms that indicate a non-cultivated individual. Default is NULL, meaning it will use the non cultivated-related expressions available in `RuHere::cultivated$not_cultivated`.

Value

A `data.frame` that is the original `occ` data frame augmented with a new column named `cultivated_flag`. Records identified as cultivated receive `FALSE`, while all other records receive `TRUE`. As with all other flagging functions in RuHere, `TRUE` indicates that the record passed this test and is eligible for retention; `FALSE` indicates it failed and is flagged as potentially problematic, and can be removed using `remove_flagged()`.

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Flag fossil records
occ_cultivated <- flag_cultivated(occ = occurrences)
```

`flag_duplicates` *Flag duplicated records*

Description

This function identifies duplicated records based on species name and coordinates, as well as user-defined additional columns or raster cells. Among duplicated records, the function keeps only one unflagged record, chosen according to a continuous variable (e.g., keeping the most recent), a categorical variable (e.g., prioritizing a specific data source), or randomly.

Usage

```
flag_duplicates(
  occ,
  species = "species",
  long = "decimalLongitude",
  lat = "decimalLatitude",
  additional_groups = NULL,
  continuous_variable = NULL,
  decreasing = TRUE,
  categorical_variable = NULL,
  priority_categories = NULL,
  by_cell = FALSE,
  raster_variable = NULL
)
```

Arguments

`occ`	(data.frame) a data frame containing the occurrence records to be examined, preferably standardized using `format_columns()`. Must contain the columns specified in `species`, `long` and `lat` arguments.
`species`	(character) the name of the column containing species names. Default is "species".
`long`	(character) the name of the column containing longitude values. Default is `"decimalLongitude"`.
`lat`	(character) the name of the column containing latitude values. Default is `"decimalLatitude"`.
`additional_groups`	(character) optional vector of additional column names to consider when identifying duplicates. For example, if `"year"` is included, records with the same coordinates but different collection years will not be flagged. Default is `NULL`.

continuous_variable
: (character) optional name of a numeric column used to sort duplicated records and select one to remain unflagged. Default is NULL, meaning that no sorting will occur and the unflagged record will be selected randomly.

decreasing
: (logical) whether to sort records in decreasing order using the continuous_variable (e.g., from most recent to oldest when the variable is "year"). Only applicable when continuous_variable is not NULL. Default is TRUE.

categorical_variable
: (character) (character) optional name of a categorical column used to sort duplicated records and select one to remain unflagged. If provided, the order of priority must be specified through priority_categories. Default is NULL.

priority_categories
: (character) vector of categories, in the desired order of priority, present in the column specified in categorical_variable. Only applicable when categorical_variable is not NULL. Default is NULL.

by_cell
: (logical) whether to use raster cells instead of raw coordinates to identify duplicates (i.e., all records inside the same raster cell are treated as duplicates). If TRUE, a SpatRaster must be supplied in raster_variable. Default is FALSE.

raster_variable
: (SpatRaster) a SpatRaster used to identify duplicated records by raster cell. Only applicable when by_cell is TRUE. Default is NULL.

Value

A data.frame that is the original occ data frame augmented with a new column named duplicated_flag. Records identified as duplicated receive FALSE, while all unique retained records receive TRUE. As with all other flagging functions in RuHere, TRUE indicates that the record passed this test and is eligible for retention; FALSE indicates it failed and is flagged as potentially problematic, and can be removed using remove_flagged().

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Duplicate some records as example
occurrences <- rbind(occurrences[1:1000, ], occurrences[1:100,])
# Flag duplicates
occ_dup <- flag_duplicates(occ = occurrences)
sum(!occ_dup$duplicated_flag) #Number of duplicated records
```

---

flag_env_moran *Select Environmentally Thinned Occurrences Using Moran's I Autocorrelation*

Description

This function evaluates multiple environmentally thinned datasets (produced using different number of blocks) and selects the one that best balances **low spatial autocorrelation** and **number of retained records**.

For each number of bins provided in `n_bins`, the function computes Moran's I for the selected environmental variables and summarizes autocorrelation using a chosen statistic (mean, median, minimum, or maximum). The best thinning level is then selected according to criteria described in *Details*.

Usage

```
flag_env_moran(
  occ,
  species = "species",
  long = "decimalLongitude",
  lat = "decimalLatitude",
  env_layers,
  n_bins,
  distance = "haversine",
  moran_summary = "mean",
  min_records = 10,
  min_imoran = 0.1,
  prioritary_column = NULL,
  decreasing = TRUE,
  do_pca = FALSE,
  mask = NULL,
  pca_buffer = 1000,
  flag_for_NA = FALSE,
  return_all = FALSE,
  verbose = TRUE
)
```

Arguments

`occ`	(data.frame or data.table) a data frame containing the occurrence records for a **single species**. Must contain columns for species, longitude, and latitude.
`species`	(character) the name of the column in `occ` that contains the species scientific names. Default is `"species"`.
`long`	(character) the name of the column in `occ` that contains the longitude values. Default is `"decimalLongitude"`.
`lat`	(character) the name of the column in `occ` that contains the latitude values. Default is `"decimalLatitude"`.
`env_layers`	(SpatRaster) object containing environmental variables for splitting in `n_bins` and for computing Moran's I.
`n_bins`	(numeric) vector of number of bins into which each environmental variable will be divided (e.g., c(5, 10, 15, 20)).

distance	(character) distance metric used to compute the weight matrix for Moran's I. One of "haversine" or "euclidean". Default: "haversine".
moran_summary	(character) summary statistic used to select the best thinning distance. One of "mean", "median", "max", or "min". Default: "mean".
min_records	(numeric) minimum number of records required for a dataset to be considered. Default: 10.
min_imoran	(numeric) minimum Moran's I required to avoid selecting datasets with extremely low spatial autocorrelation. Default: 0.1.
prioritary_column	(character) name of a numeric columns in occto define retention priority (e.g., quality score, year). See details.
decreasing	(logical) whether to sort records in decreasing order using the prioritary_column (e.g., from most recent to oldest when the variable is "year"). Only applicable when prioritary_column is not NULL. Default is TRUE.
do_pca	(logical) whether environmental variables should be summarized using PCA before computing Moran's I. Default: FALSE. See details.
mask	(SpatVector or SpatExtent) optional spatial object to mask the env_layers before computing PCA. Only applicable if do_pca = TRUE. Default is NULL.
pca_buffer	(numeric) buffer width (km) used when PCA is computed from the convex hull of records. Ignored if mask is provided. Default: 1000.
flag_for_NA	(logical) whether to treat records falling in NA cells of env_layers as passing (TRUE) or failing (FALSE) this test. Default is FALSE.
return_all	(logical) whether to return the full list of all thinned datasets. Default is FALSE.
verbose	(logical) whether to print messages about the progress. Default is TRUE

Details

This function is inspired by the approach used in Velazco et al. (2020), extending the procedure by allowing:

- prioritization of records based on a user-defined variable (e.g., year)
- optional PCA transformation of environmental layers
- selection rules that prevent datasets with too few records or extremely low Moran's I from being chosen.

Procedure overview

1. For each bin number in n_bins, generate a spatially thinned dataset using thin_env() function.
2. Extract environmental values for the retained records.
3. Compute Moran's I for each environmental variable.
4. Summarize autocorrelation per dataset (mean, median, min, or max).
5. Apply the selection criteria:
    - Keep only datasets with at least min_records records.

- Keep only datasets with Moran's I greater or equal to `min_imoran`.
- Round Moran's I to two decimal places and select the dataset with the **25th lowest** autocorrelation.
- If more than on dataset is selected, choose the dataset retaining **more records**.
- If still tied, choose the dataset with the **largest number of bins**.

Distance matrix for Moran's I Moran's I requires a weight matrix derived from pairwise distances among records. Two distance types are available:

- `"haversine"`: geographic distance computed with `fields::rdist.earth()` (default; recommended for longitude/latitude coordinates)
- `"euclidean"`: Euclidean distance computed with `stats::dist()`

Environmental PCA (optional) If `do_pca = TRUE`, the environmental layers are summarized using PCA before Moran's I is computed.

- If `mask` is provided, PCA is computed on masked layers.
- Otherwise, a convex hull around the records is buffered by `pca_buffer` kilometers to define the PCA area.
- It will select the axis that together explain more than 90% of the variation.

Value

A list with:

- **occ**: the selected thinned occurrence dataset with the column `thin_env_flag` indicating whether each record was retained (`TRUE`) or flagged as redundant (`FALSE`) in environmental space. As with all other flagging functions in RuHere, `TRUE` indicates that the record passed this test and is eligible for retention; `FALSE` indicates it failed and is flagged as potentially problematic, and can be removed using `remove_flagged()`.
- **imoran**: a table summarizing Moran's I for each thinning distance
- **n_bins**: the number of bins that produced the selected dataset
- **moran_summary**: the summary statistic used to select the dataset
- **all_thined**: (optional) list of thinned datasets for all bin numbers. Only returned if `return_all` was set to `TRUE`

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Subset occurrences from Araucaria
occ <- occurrences[occurrences$species == "Araucaria angustifolia", ]
# Load example of raster variables
data("worldclim", package = "RuHere")
# Unwrap Packed raster
r <- terra::unwrap(worldclim)
# Select thinned occurrences
occ_env_moran <- flag_env_moran(occ = occ,
                                n_bins = c(5, 10, 20, 30, 40, 50),
```

```
                                    env_layers = r)
# Selected number of bins
occ_env_moran$n_bins
# Number of flagged and unflagged records
sum(occ_env_moran$occ$thin_env_flag) #Retained
sum(!occ_env_moran$occ$thin_env_flag) #Flagged for thinning out
# Results os the spatial autocorrelation analysis
occ_env_moran$imoran
```

`flag_faunabr` *Identify records outside natural ranges according to Fauna do Brasil*

Description

Flags occurrence records based on known distribution data from the Catálogo Taxônomico da Fauna do Brasil (faunabr) data. This function checks if an occurrence point for a given species falls within its documented distribution, allowing for user-defined buffers around Brazilian states, or the entire country. Records are flagged as `TRUE` if they fall within the specified range for the distribution information available in the `faunabr` data.

Usage

```
flag_faunabr(
  data_dir,
  occ,
  species = "species",
  long = "decimalLongitude",
  lat = "decimalLatitude",
  origin = NULL,
  by_state = TRUE,
  buffer_state = 20,
  by_country = TRUE,
  buffer_country = 20,
  keep_columns = TRUE,
  spat_state = NULL,
  spat_country = NULL,
  progress_bar = FALSE,
  verbose = FALSE
)
```

Arguments

`data_dir`	(character) **Required** directory path where the `faunabr` data is saved.
`occ`	(data.frame or data.table) a data frame containing the occurrence records to be flagged. Must contain columns for species, longitude, and latitude.
`species`	(character) the name of the column in `occ` that contains the species scientific names. Default is `"species"`.

long	(character) the name of the column in occ that contains the longitude values. Default is "decimalLongitude".
lat	(character) the name of the column in occ that contains the latitude values. Default is "decimalLatitude".
origin	(character) filter the faunabr data by origin type before checking ("native", "cryptogenic", or "exotic"). Default is NULL (no filtering).
by_state	(logical) if TRUE, flags records based on their distance to known Brazilian state distributions. Default is TRUE.
buffer_state	(numeric) buffer distance (in kilometers) to be applied around the known state distribution boundaries. Records within this distance are flagged as TRUE. Default is 20 km.
by_country	(logical) if TRUE, flags records based on their distance to country distributions. Default is TRUE.
buffer_country	(numeric) buffer distance (in kilometers) to be applied around the country boundaries. Records within this distance are flagged as TRUE. Default is 20 km.
keep_columns	(logical) if TRUE, the returned data frame contains all original columns from occ. If FALSE, it returns only the key columns and the flag. Default is TRUE.
spat_state	(SpatVector) a SpatVector of the Brazilian states. By default, it uses the SpatVector provided by geobr::read_state(). It can be another Spatvector, but the structure must be identical to 'faunabr::states', with a column called "abbrev_state" identifying the states codes.
spat_country	(SpatVector) a SpatVector of the world countries. By default, it uses the SpatVector provided by rnaturalearth::ne_countries. It can be another Spatvector, but the structure must be identical to 'faunabr::world_fauna', with a column called "country_code" identifying the country codes.
progress_bar	(logical) whether to display a progress bar during processing. If TRUE, the 'pbapply' package must be installed. Default is FALSE.
verbose	(logical) if TRUE, prints messages about the progress and the number of species being checked. Default is FALSE.

Value

A data.frame that is the original occ data frame augmented with a new column named faunabr_flag. This column is logical (TRUE/FALSE) indicating whether the record falls within the expected distribution (plus buffer) based on the faunabr data. As with all other flagging functions in RuHere, TRUE indicates that the record passed this test and is eligible for retention; FALSE indicates it failed and is flagged as potentially problematic, and can be removed using remove_flagged(). Records for species not found in the faunabr data will have NA in the faunabr_flag column.

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Get only occurrences from Azure Jay
occ <- occurrences[occurrences$species == "Cyanocorax caeruleus", ]
# Set folder where distributional datasets were saved
```

```
# Here, just a sample provided in the package
# You must run 'faunabr_here()' beforehand to download the necessary data files for your species
dataset_dir <- system.file("extdata/datasets", package = "RuHere")
# Flag records using faunabr specialist information
occ_fauna <- flag_faunabr(data_dir = dataset_dir, occ = occ)
```

`flag_florabr` *Identify records outside natural ranges according to Flora e Funga do Brasil*

Description

Flags occurrence records based on known distribution data from the Flora e Funga do Brasil (florabr) data. This function checks if an occurrence point for a given species falls within its documented distribution, allowing for user-defined buffers around Brazilian states, biomes, or the entire country. Records are flagged as `TRUE` if they fall within the specified range for the distribution information available in the `florabr` data.

Usage

```
flag_florabr(
  data_dir,
  occ,
  species = "species",
  long = "decimalLongitude",
  lat = "decimalLatitude",
  origin = NULL,
  by_state = TRUE,
  buffer_state = 20,
  by_biome = TRUE,
  buffer_biome = 20,
  by_endemism = TRUE,
  buffer_brazil = 20,
  state_vect = NULL,
  state_column = NULL,
  biome_vect = NULL,
  biome_column = NULL,
  br_vect = NULL,
  keep_columns = TRUE,
  progress_bar = FALSE,
  verbose = FALSE
)
```

Arguments

`data_dir`	(character) directory path where the `florabr` data is saved **Required.**
`occ`	(data.frame) a data frame containing the occurrence records to be flagged. Must contain columns for species, longitude, and latitude.

species	(character) the name of the column in occ that contains the species scientific names. Default is "species".
long	(character) the name of the column in occ that contains the longitude values. Default is "decimalLongitude".
lat	(character) the name of the column in occ that contains the latitude values. Default is "decimalLatitude".
origin	(character or NULL) filter the florabr data by origin type before checking ("native", "cultivated", "naturalized", "unknown", or "not_found_in_brazil"). Default is NULL (no filtering).
by_state	(logical) if TRUE, flags records based on their distance to known Brazilian state distributions. Default is TRUE.
buffer_state	(numeric) buffer distance (in kilometers) to be applied around the known state distribution boundaries. Records within this distance are flagged as TRUE. Default is 20 km.
by_biome	(logical) if TRUE, flags records based on their distance to known Brazilian biome distributions. Default is TRUE.
buffer_biome	(numeric) buffer distance (in kilometers) to be applied around the known biome distribution boundaries. Records within this distance are flagged as TRUE. Default is 20 km.
by_endemism	(logical) if TRUE, includes a check against the entire Brazilian boundary. Default is TRUE.
buffer_brazil	(numeric) buffer distance (in kilometers) to be applied around the entire Brazilian boundary. Default is 20 km.
state_vect	(SpatVector) qn optional custom simple features (sf) vector representing Brazilian states/regions. If NULL, uses the default data loaded by florabr. Default is NULL.
state_column	(character) the name of the column in state_vect (or the default state vector) used to match distribution information. Default is NULL.
biome_vect	(SpatVector) an optional custom simple features (sf) vector representing Brazilian biomes. If NULL, uses the default data loaded by florabr. Default is NULL.
biome_column	(character) the name of the column in biome_vect (or the default biome vector) used to match distribution information. Default is NULL.
br_vect	(SpatVector) an optional custom simple features (sf) vector representing the entire Brazilian boundary. If NULL, uses the default data loaded by florabr. Default is NULL.
keep_columns	(logical) if TRUE, the returned data frame contains all original columns from occ. If FALSE, it returns only the key columns and the flag. Default is TRUE.
progress_bar	(logical) whether to display a progress bar during processing. If TRUE, the 'pbapply' package must be installed. Default is FALSE.
verbose	(logical) if TRUE, prints messages about the progress and the number of species being checked. Default is FALSE.

Value

A data.frame that is the original occ data frame augmented with a new column named florabr_flag. This column is logical (TRUE/FALSE) indicating whether the record falls within the expected distribution (plus buffer) based on the florabr data. As with all other flagging functions in RuHere, TRUE indicates that the record passed this test and is eligible for retention; FALSE indicates it failed and is flagged as potentially problematic, and can be removed using remove_flagged(). Records for species not found in the florabr data will have NA in the florabr_flag column.

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Get only occurrences from Araucaria
occ <- occurrences[occurrences$species == "Araucaria angustifolia", ]
# Set folder where distributional datasets were saved
# Here, just a sample provided in the package
# You must run 'florabr_here()' beforehand to download the necessary data files for your species
dataset_dir <- system.file("extdata/datasets", package = "RuHere")

# Flag records using specialist information from Flora do Brasil
occ_flora <- flag_florabr(data_dir = dataset_dir, occ = occ)
```

---

flag_fossil *Flag fossil records*

Description

This function identifies occurrence records that correspond to fossils, based on specific search terms found in selected columns.

Usage

```
flag_fossil(
  occ,
  columns = c("basisOfRecord", "occurrenceRemarks"),
  fossil_terms = NULL
)
```

Arguments

occ	(data.frame) a data frame containing the occurrence records to be examined, preferably standardized using format_columns(). Must contain the columns specified in columns.
columns	(character) vector of column names in occ where the function will search for the term "fossil" or other fossil-related expressions. Default is c("basisOfRecord", "occurrenceRemarks").
fossil_terms	(character) optional vector of additional terms that indicate a fossil record (e.g., "paleontological", "subfossil"). Default is NULL.

Value

A `data.frame` that is the original `occ` data frame augmented with a new column named `fossil_flag`. Records identified as fossils receive `FALSE`, while all other records receive `TRUE`. As with all other flagging functions in RuHere, `TRUE` indicates that the record passed this test and is eligible for retention; `FALSE` indicates it failed and is flagged as potentially problematic, and can be removed using `remove_flagged()`.

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Flag fossil records
occ_fossil <- flag_fossil(occ = occurrences)
```

`flag_geo_moran` *Select Spatially Thinned Occurrences Using Moran's I Autocorrelation*

Description

This function evaluates multiple geographically thinned datasets (produced using different thinning distances) and selects the one that best balances **low spatial autocorrelation** and **number of retained records**.

For each thinning distance provided in `d`, the function computes Moran's I for the selected environmental variables and summarizes autocorrelation using a chosen statistic (mean, median, minimum, or maximum). The best thinning level is then selected according to criteria described in *Details*.

Usage

```
flag_geo_moran(
  occ,
  species = "species",
  long = "decimalLongitude",
  lat = "decimalLatitude",
  d,
  distance = "haversine",
  moran_summary = "mean",
  min_records = 10,
  min_imoran = 0.1,
  prioritary_column = NULL,
  decreasing = TRUE,
  env_layers,
  do_pca = FALSE,
  mask = NULL,
  pca_buffer = 1000,
  return_all = FALSE,
  verbose = TRUE
)
```

Arguments

occ	(data.frame or data.table) a data frame containing the occurrence records for a **single species**. Must contain columns for species, longitude, and latitude.
species	(character) the name of the column in occ that contains the species scientific names. Default is "species".
long	(character) the name of the column in occ that contains the longitude values. Default is "decimalLongitude".
lat	(character) the name of the column in occ that contains the latitude values. Default is "decimalLatitude".
d	(numeric) vector of thinning distances in **kilometers** (e.g., c(5, 10, 15, 20)).
distance	(character) distance metric used to compute the weight matrix for Moran's I. One of "haversine" or "euclidean". Default: "haversine".
moran_summary	(character) summary statistic used to select the best thinning distance. One of "mean", "median", "max", or "min". Default: "mean".
min_records	(numeric) minimum number of records required for a dataset to be considered. Default: 10.
min_imoran	(numeric) minimum Moran's I required to avoid selecting datasets with extremely low spatial autocorrelation. Default: 0.1.
prioritary_column	(character) name of a numeric columns in occto define retention priority (e.g., quality score, year). See details.
decreasing	(logical) whether to sort records in decreasing order using the prioritary_column (e.g., from most recent to oldest when the variable is "year"). Only applicable when prioritary_column is not NULL. Default is TRUE.
env_layers	(SpatRaster) object containing environmental variables for computing Moran's I.
do_pca	(logical) whether environmental variables should be summarized using PCA before computing Moran's I. Default: FALSE. See details.
mask	(SpatVector or SpatExtent) optional spatial object to mask the env_layers before computing PCA. Only applicable if do_pca = TRUE. Default is NULL.
pca_buffer	(numeric) buffer width (km) used when PCA is computed from the convex hull of records. Ignored if mask is provided. Default: 1000.
return_all	(logical) whether to return the full list of all thinned datasets. Default: FALSE.
verbose	(logical) whether to print messages about the progress. Default is TRUE

Details

This function is inspired by the approach used in Velazco et al. (2021), extending the procedure by allowing:

- prioritization of records based on a user-defined variable (e.g., year)
- optional PCA transformation of environmental layers
- selection rules that prevent datasets with too few records or extremely low Moran's I from being chosen.

Procedure overview

1. For each distance in `d`, generate a spatially thinned dataset using `thin_geo()` function.
2. Extract environmental values for the retained records.
3. Compute Moran's I for each environmental variable.
4. Summarize autocorrelation per dataset (mean, median, min, or max).
5. Apply the selection criteria:
   - Keep only datasets with at least `min_records` records.
   - Keep only datasets with Moran's I higher than `min_imoran`.
   - Round Moran's I to two decimal places and select the dataset with the **25th lowest** autocorrelation.
   - If more than on dataset is selected, choose the dataset retaining **more records**.
   - If still tied, choose the dataset with the **smallest thinning distance**.

Distance matrix for Moran's I Moran's I requires a weight matrix derived from pairwise distances among records. Two distance types are available:

- `"haversine"`: geographic distance computed with `fields::rdist.earth()` (default; recommended for longitude/latitude coordinates)
- `"euclidean"`: Euclidean distance computed with `stats::dist()`

Environmental PCA (optional) If `do_pca = TRUE`, the environmental layers are summarized using PCA before Moran's I is computed.

- If `mask` is provided, PCA is computed on masked layers.
- Otherwise, a convex hull around the records is buffered by `pca_buffer` kilometers to define the PCA area.
- It will select the axis that together explain more than 90% of the variation.

Value

A list with:

- **occ**: the selected thinned occurrence dataset with the column `thin_geo_flag` indicating whether each record was retained (`TRUE`) or flagged as redundant (`FALSE`) in geographic space. As with all other flagging functions in RuHere, `TRUE` indicates that the record passed this test and is eligible for retention; `FALSE` indicates it failed and is flagged as potentially problematic, and can be removed using `remove_flagged()`.
- **imoran**: a table summarizing Moran's I for each thinning distance
- **distance**: the thinning distance that produced the selected dataset
- **moran_summary**: the summary statistic used to select the dataset
- **all_thined**: (optional) list of thinned datasets for all distances. Only returned if `return_all` was set to `TRUE`

References

- Velazco, S. J. E., Svenning, J. C., Ribeiro, B. R., & Laureto, L. M. O. (2021). On opportunities and threats to conserve the phylogenetic diversity of Neotropical palms. Diversity and Distributions, 27(3), 512–523. https://doi.org/10.1111/ddi.13215

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Subset occurrences from Araucaria
occ <- occurrences[occurrences$species == "Araucaria angustifolia", ]
# Load example of raster variables
data("worldclim", package = "RuHere")
# Unwrap Packed raster
r <- terra::unwrap(worldclim)
# Select thinned occurrences
occ_geo_moran <- flag_geo_moran(occ = occ, d = c(5, 10, 20, 30),
                                env_layers = r)
# Selected distance
occ_geo_moran$distance
# Number of flagged and unflagged records
sum(occ_geo_moran$occ$thin_geo_flag) #Retained
sum(!occ_geo_moran$occ$thin_geo_flag) #Flagged for thinning out
# Results os the spatial autocorrelation analysis
occ_geo_moran$imoran
```

---

flag_inaturalist *Flag occurrence records sourced from iNaturalist*

Description

This function identifies and flags occurrence records sourced from iNaturalist. It can flag all iNaturalist records or only those that do not have Research Grade status.

Usage

```
flag_inaturalist(occ, columns = "datasetName", research_grade = FALSE)
```

Arguments

occ	(data.frame) a data frame containing the occurrence records to be examined, preferably standardized using `format_columns()`. Must contain the columns specified in `columns`.
columns	(character) column name in `occ` where the function will search for the term "iNaturalist". Default is "datasetName".
research_grade	(logical) whether to flag *all* records from iNaturalist, including those with Research Grade status. Default is `FALSE`, meaning that only iNaturalist records **without** Research Grade will be flagged.

Details

According to iNaturalist, Observations become Research Grade when:

- the iNaturalist community agrees on species-level ID or lower, i.e. when more than 2/3 of identifiers agree on a taxon;
- the community taxon and the observation taxon agree;
- or the community agrees on an ID between family and species and votes that the community taxon is as good as it can be.

Value

A `data.frame` that is the original `occ` data frame augmented with a new column named `inaturalist_flag`. Flagged records receive `FALSE`, while all other records receive `TRUE`. As with all other flagging functions in RuHere, `TRUE` indicates that the record passed this test and is eligible for retention; `FALSE` indicates it failed and is flagged as potentially problematic, and can be removed using `remove_flagged()`.

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Flag only iNaturalist records without Research Grade
occ_inat <- flag_inaturalist(occ = occurrences, research_grade = FALSE)
table(occ_inat$inaturalist_flag) # Number of records flagged (FALSE)
# Flag all iNaturalist records (including Research Grade)
occ_inat <- flag_inaturalist(occ = occurrences, research_grade = TRUE)
table(occ_inat$inaturalist_flag) # Number of records flagged (FALSE)
```

`flag_iucn` *Identify records outside natural ranges according to the IUCN*

Description

Flags occurrence records based on known distribution data from the International Union for Conservation of Nature (IUCN) data. This function checks if an occurrence point for a given species falls within its documented distribution, allowing for user-defined buffers around the region. Records are flagged as `TRUE` if they fall inside the documented distribution (plus optional buffer) for the species in the IUCN dataset.

Usage

```
flag_iucn(
  data_dir,
  occ,
  species = "species",
  long = "decimalLongitude",
  lat = "decimalLatitude",
```

```
  origin = "native",
  presence = "all",
  buffer = 20,
  progress_bar = FALSE,
  verbose = FALSE
)
```

Arguments

data_dir	(character) **Required** directory path where the IUCN data is saved
occ	(data.frame or data.table) a data frame containing the occurrence records to be flagged. Must contain columns for species, longitude, and latitude.
species	(character) the name of the column in occ that contains the species scientific names. Default is "species".
long	(character) the name of the column in occ that contains the longitude values. Default is "decimalLongitude".
lat	(character) the name of the column in occ that contains the latitude values. Default is "decimalLatitude".
origin	(character) vector specifying which origin categories should be considered as part of the species' range. Options are: "native", "introduced", "reintroduced", "vagrant", "origin uncertain", "assisted colonisation", and "all". For example, if origin = "introduced", only regions where the species is considered introduced will be used to flag records. Default is "native".
presence	(character) vector specifying which presence type should be considered as part of the species' range. Options are: "extant", "probably extant", "possibly extant", "extinct", "possibly extinct", "presence uncertain", and "all". For example, if presence = "extinct", only regions where the species is considered extinct will be used to flag records. Default is "all", meaning all regions will be considered.
buffer	(numeric) buffer distance (in kilometers) to be applied around the region of distribution. Default is 20 km.
progress_bar	(logical) whether to display a progress bar during processing. If TRUE, the 'pbapply' package must be installed. Default is FALSE.
verbose	(logical) if TRUE, prints messages about the progress and the number of species being checked. Default is FALSE.

Value

A data.frame that is the original occ data frame augmented with a new column named iucn_flag. This column is logical (TRUE/FALSE) indicating whether the record falls within the expected distribution (plus buffer) based on the IUCN data. As with all other flagging functions in RuHere, TRUE indicates that the record passed this test and is eligible for retention; FALSE indicates it failed and is flagged as potentially problematic, and can be removed using remove_flagged(). Records for species not found in the IUCN data will have NA in the iucn_flag column.

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Get only occurrences from Araucaria
occ <- occurrences[occurrences$species == "Araucaria angustifolia", ]
# Set folder where distributional datasets were saved
# Here, just a sample provided in tha package
# You must run 'iucn_here()' beforehand to download the necessary data files
dataset_dir <- system.file("extdata/datasets", package = "RuHere")

# Flag records using IUCN specialist information
occ_iucn <- flag_iucn(data_dir = dataset_dir, occ = occ)
```

---

flag_names *Flag name dictionary*

Description

A named character vector used to convert internal flag column names (produced by the package's flagging functions) into human-readable labels.

Usage

```
flag_names
```

Format

A named character vector of length 25. The names correspond to the original flag codes (e.g., "correct_country", "duplicated_flag", ".cen", "consensus_flag"), and the values are the cleaned, human-readable labels (e.g., "Wrong country", "Duplicated", "Country/Province centroid", "consensus").

Details

This object is used internally by functions such as mapview_here() and remove_flagged()to display more intuitive flag names to users.

`flag_wcvp` *Identify records outside natural ranges according to the World Checklist of Vascular Plants*

Description

Flags occurrence records based on known distribution data from the World Checklist of Vascular Plants (WCVP) data. This function checks if an occurrence point for a given species falls within its documented distribution, allowing for user-defined buffers around the region. Records are flagged as `TRUE` if they fall inside the documented distribution (plus optional buffer) for the species in the WCVP dataset.

Usage

```
flag_wcvp(
  data_dir,
  occ,
  species = "species",
  long = "decimalLongitude",
  lat = "decimalLatitude",
  origin = "native",
  buffer = 20,
  progress_bar = FALSE,
  verbose = FALSE
)
```

Arguments

`data_dir`	(character) **Required** directory path where the `WCVP` data is saved
`occ`	(data.frame or data.table) a data frame containing the occurrence records to be flagged. Must contain columns for species, longitude, and latitude.
`species`	(character) the name of the column in `occ` that contains the species scientific names. Default is `"species"`.
`long`	(character) the name of the column in `occ` that contains the longitude values. Default is `"decimalLongitude"`.
`lat`	(character) the name of the column in `occ` that contains the latitude values. Default is `"decimalLatitude"`.
`origin`	(character) vector specifying which origin categories should be considered as part of the species' range. Options are: `"native"`, `"introduced"`, `"extinct"`, `"location_doubtful"`, and `"all"`. For example, if `origin = "introduced"`, only regions where the species is considered introduced will be used to flag records. Default is `"native"`.
`buffer`	(numeric) buffer distance (in kilometers) to be applied around the region of distribution. Default is 20 km.

progress_bar	(logical) whether to display a progress bar during processing. If TRUE, the 'pbapply' package must be installed. Default is FALSE.
verbose	(logical) if TRUE, prints messages about the progress and the number of species being checked. Default is FALSE.

Value

A data.frame that is the original occ data frame augmented with a new column named wcvp_flag. This column is logical (TRUE/FALSE) indicating whether the record falls within the expected distribution (plus buffer) based on the WCVP data. As with all other flagging functions in RuHere, TRUE indicates that the record passed this test and is eligible for retention; FALSE indicates it failed and is flagged as potentially problematic, and can be removed using remove_flagged(). Records for species not found in the WCVP data will have NA in the wcvp_flag column.

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Filter occurrences for Araucaria
occ <- occurrences[occurrences$species == "Araucaria angustifolia", ]
# Set folder where distributional datasets were saved
# Here, just a sample provided in the package
# You must run 'wcvp_here()' beforehand to download the necessary data files
dataset_dir <- system.file("extdata/datasets", package = "RuHere")

# Flag records using WCVP specialist information
occ_wcvp <- flag_wcvp(data_dir = dataset_dir, occ = occ)
```

---

flag_year *Flag records outside a year range*

Description

This function identifies occurrence records collected before or after user-specified years.

Usage

```
flag_year(
  occ,
  year_column = "year",
  lower_limit = NULL,
  upper_limit = NULL,
  flag_NA = FALSE
)
```

Arguments

occ	(data.frame) a dataset with occurrence records, preferably standardized using format_columns(). Must contain the column specified in year_column.
year_column	(character) name of the column containing the year in which the occurrence was recorded. This column must be numeric.
lower_limit	(numeric) the minimum acceptable year. Records collected before this value will be flagged. Default is NULL.
upper_limit	(numeric) the maximum acceptable year. Records collected after this value will be flagged. Default is NULL.
flag_NA	(character) whether to flag records with missing year information. Default is FALSE.

Value

A data.frame identical to occ but with an additional column named year_flag. Records collected outside the year range specified are assigned FALSE. As with all other flagging functions in RuHere, TRUE indicates that the record passed this test (i.e., falls within the specified year range) and is eligible for retention; FALSE indicates it failed and is flagged as potentially problematic, and can be removed using remove_flagged().

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Flag records collected before 1980 and after 2010
occ_year <- flag_year(occ = occurrences, lower_limit = 1980,
                      upper_limit = 2010)
```

---

florabr_here *Download the latest version of Flora e Funga do Brasil database*

Description

This function downloads the Flora e Funga do Brasil database, which is required for filtering occurrence records using specialists' range information via the flag_florabr() function.

Usage

```
florabr_here(
  data_dir,
  data_version = "latest",
  get_fixed_version = TRUE,
  solve_discrepancy = TRUE,
  overwrite = TRUE,
  remove_files = TRUE,
  verbose = TRUE
)
```

Arguments

data_dir	(character) a directory to save the data downloaded from Flora e Funga do Brasil.
data_version	(character) version of the Flora e Funga do Brasil database to download. Use "latest" to get the most recent version, updated weekly. Alternatively, specify an older version (e.g., data_version="393.319"). Default value is "latest".
get_fixed_version	If TRUE, download the fixed and already-merged version 393.432 from Zenodo instead of downloading and processing an IPT archive. Default is TRUE.
solve_discrepancy	(logical) whether to resolve discrepancies between species and subspecies/varieties information. When set to TRUE, species information is updated based on unique data from varieties and subspecies. For example, if a subspecies occurs in a certain biome, it implies that the species also occurs in that biome. Default is TRUE.
overwrite	(logical) if TRUE, data is overwritten. Default = TRUE.
remove_files	(logical) whether to remove the downloaded files used in building the final dataset. Default is TRUE.
verbose	(logical) whether to display messages during function execution. Set to TRUE to enable display, or FALSE to run silently. Default is TRUE.

Value

A message indicating that the data were successfully saved in the directory specified by data_dir.

Examples

```
## Not run:
# Define a directory to save the data
data_dir <- tempdir() # Here, a temporary directory

# Download the latest version of the Flora e Funga do Brasil database
florabr_here(data_dir = data_dir)

## End(Not run)
```

---

format_columns *Format and standardize column names and data types of an occurrence dataset*

Description

Format and standardize column names and data types of an occurrence dataset

Usage

```
format_columns(
  occ,
  metadata,
  extract_binomial = TRUE,
  binomial_from = NULL,
  include_subspecies = FALSE,
  include_variety = FALSE,
  check_numeric = TRUE,
  numeric_columns = NULL,
  check_encoding = TRUE,
  data_source = NULL,
  progress_bar = FALSE,
  verbose = FALSE
)
```

Arguments

occ	(data.frame or data.table) a dataset with occurrence records, preferably obtained from `import_gbif()`, `get_specieslink()`, `get_bien()`, or `get_idigbio()`.
metadata	(character or data.frame) if a character, one of 'gbif', 'specieslink', 'bien', or 'idigbio', specifying which metadata template to use (the corresponding data frames are available in `RuHere::prepared_metadata`). If a data.frame is provided, it must have 21 columns (see **Details**).
extract_binomial	(logical) whether to create a column with the binomial name of the species. If FALSE, it will create a column "species" with the exact name stored in the scientificName column. Default is TRUE.
binomial_from	(character) the column name in metadata from which to extract the binomial name. Only applicable if `extract_binomial = TRUE`. If `metadata` corresponds to one of the predefined sources ('gbif', specieslink', 'bien', or 'idigbio'), predefined columns will be used automatically. Default is "scientificName".
include_subspecies	(logical) whether to include subspecies in the binomial name. Only applicable if `extract_binomial = TRUE`. If TRUE, the function includes any infraspecific epithet after the pattern "subsp.". Default if FALSE.
include_variety	(logical) whether to include variety in the binomial name. Only applicable if `extract_binomial = TRUE`. If TRUE, the function includes any infraspecific epithet after the pattern "var.". Default if FALSE.
check_numeric	(logical) whether to check and coerce the columns specified in `numeric_columns` to numeric type. Default is TRUE.
numeric_columns	(character) a vector of column names that must be numeric. Default is NULL, meaning that if `check_numeric = TRUE`, the following columns will be coerced: 'decimalLongitude', 'decimalLatitude', 'coordinateUncertaintyInMeters', 'elevation', and 'year'.

check_encoding	(logical) whether to check and fix the encoding of columns that typically contain special characters (see **Details**). Default is TRUE.
data_source	(character) the source of the occurrence records. Default is NULL, meaning it will use the same string provided in metadata. If metadata is a user-defined data.frame, this argument must be specified.
progress_bar	(logical) whether to display a progress bar during processing. If TRUE, the 'pbapply' package must be installed. Default is FALSE.
verbose	(logical) whether to print messages about the progress. Default is FALSE.

Details

If a user-defined metadata data.frame is provided, it must include the following 21 columns: 'scientificName', 'collectionCode', 'catalogNumber', 'decimalLongitude', 'decimalLatitude', 'coordinateUncertaintyInMeters', 'elevation', 'country', 'stateProvince', 'municipality', 'locality', 'year', 'eventDate', 'recordedBy', 'identifiedBy', 'basisOfRecord', 'occurrenceRemarks', 'habitat', 'datasetName', 'datasetKey', and 'key'.

If check_encoding = TRUE, the function will inspect and, if necessary, fix the encoding of these columns: 'collectionCode', 'catalogNumber', 'country', 'stateProvince', municipality', 'locality', 'eventDate','recordedBy', 'identifiedBy', 'basisOfRecord', and 'datasetName'.

Value

A data.frame with standardized column names and data types according to the specified metadata.

Examples

```
# Example with GBIF
data("occ_gbif", package = "RuHere") #Import data example
gbif_standardized <- format_columns(occ_gbif, metadata = "gbif")
# Example with SpeciesLink
data("occ_splink", package = "RuHere") #Import data example
splink_standardized <- format_columns(occ_splink, metadata = "specieslink")
# Example with BIEN
data("occ_bien", package = "RuHere") #Import data example
bien_standardized <- format_columns(occ_bien, metadata = "bien")
# Example with idigbio
data("occ_idig", package = "RuHere") #Import data example
idig_standardized <- format_columns(occ_idig, metadata = "idigbio")
```

---

get_bien *Download occurrence records from BIEN*

Description

Wrapper function to access and download occurrence records from the Botanical Information and Ecology Network (BIEN) database. It provides a unified interface to query BIEN data by species, genus, family, or by geographic or political boundaries.

Usage

```
get_bien(by = "species", cultivated = FALSE,
new.world = NULL, all.taxonomy = FALSE, native.status = FALSE,
natives.only = TRUE, observation.type = FALSE, political.boundaries = TRUE,
collection.info = TRUE, only.geovalid = TRUE, min.lat = NULL, max.lat = NULL,
min.long = NULL, max.long = NULL, species = NULL, genus = NULL,
country = NULL, country.code = NULL, state = NULL, county = NULL,
state.code = NULL, county.code = NULL, family = NULL, sf = NULL, dir,
filename = "bien_output", file.format = "csv", compress = FALSE,
save = FALSE, verbose = TRUE, ...)
```

Arguments

by	(character) type of query to perform ("box", "country", "county", "family", "genus", "records_per_species", "species", "sf", or "state"). Default is species.
cultivated	(logical) whether to include cultivated records or exclude them. Default is FALSE.
new.world	(logical) if TRUE, restricts records to the New World, if FALSE, to the Old World, and if NULL, no restriction. Default is NULL.
all.taxonomy	(logical) if TRUE, returns all taxonomic levels available, otherwise, limits results to accepted names. Default is FALSE.
native.status	(logical) if TRUE, includes information about native versus non-native status of occurrences. Default is FALSE.
natives.only	(logical) if TRUE, restricts results to native species only. Default is TRUE.
observation.type	(logical) if TRUE, includes information on observation types. Default is FALSE.
political.boundaries	(logical) if TRUE, restricts the search to defined political boundaries. Default is TRUE.
collection.info	(logical) if TRUE, includes collection-level metadata. Default is TRUE.
only.geovalid	(logical) if TRUE, restricts output to georeferenced and spatially valid records. Default is TRUE.
min.lat	(numeric) the minimum latitude (in decimal degrees) for a bounding-box query when by = "box".
max.lat	(numeric) the maximum latitude (in decimal degrees) for a bounding-box query when by = "box".
min.long	(numeric) the minimum longitude (in decimal degrees) for a bounding-box query when by = "box".
max.long	(numeric) the maximum longitude (in decimal degrees) for a bounding-box query when by = "box". Ignored otherwise. Default is NULL.
species	(character) species name(s) to query when by = "species" or "records_per_species". Default is NULL.

genus	(character) genus name(s) to query when by = "genus". Default is NULL.
country	(character) country name when by = "country", "state", or "county". Default is NULL.
country.code	(character) two-letter ISO country code corresponding to country. Default is NULL.
state	(character) state or province name when by = "state" or "county". Default is NULL.
county	(character) county or equivalent subdivision name when by = "county". Default is NULL.
state.code	(character) state or province code corresponding to state. Default is NULL.
county.code	(character) county or equivalent subdivision code corresponding to county. Default is NULL.
family	(character) family name(s) to query when by = "family". Default is NULL.
sf	(object of class sf) a spatial object defining an area of interest when by = "sf". Default is NULL.
dir	(character) directory path where the file will be saved. Required if save = TRUE.
filename	(character) name of the output file without extension. Default is "bien_output".
file.format	(character) file format for saving output ("csv", "rds"). Default is "csv".
compress	(logical) if TRUE and save = TRUE, compresses the output file as .csv.zip. Default is FALSE.
save	(logical) if TRUE, saves the results to a CSV file. Default is FALSE.
verbose	(logical) if TRUE, prints messages about the progress and the number of species being checked. Default is TRUE.
...	additional arguments passed to the underlying BIEN function.

Value

A data.frame containing BIEN occurrence records that match the specified query. The structure and available columns depend on the chosen by value and the corresponding BIEN function.

Examples

```
# Example: download occurrence records for a single species
res_test <- get_bien(
    by = "species",
    species = "Paubrasilia echinata",
    cultivated = TRUE,
    native.status = TRUE,
    observation.type = TRUE,
    only.geovalid = TRUE
)
```

get_datapaper *Download and merge biodiversity data papers*

Description

Downloads, reads, and merges published biodiversity data papers (see details for available options). The function allows downloading single or multiple datasets simultaneously, optionally filtering records by species name and saving the consolidated table to disk in different file formats.

Usage

```
get_datapaper(
  datapaper,
  species = NULL,
  save = FALSE,
  dir = NULL,
  file.format = "gz",
  overwrite = FALSE,
  verbose = TRUE
)
```

Arguments

datapaper	(character) vector with the names of the data papers to download. Valid options are: "atlantic", "brazil road-kill", "dryflor", "neotropical xenarthrans", "neotroptree", dry_diagonal_epiphyte, and gabi. See details.
species	(character) optional species name or vector of species names to filter the dataset. Default is NULL (returns all species).
save	(logical) whether to save the merged output file to disk. If FALSE, files are downloaded to a temporary session directory. Default is FALSE.
dir	(character) directory path where the output file will be saved if save = TRUE. Default is NULL.
file.format	(character) file format to save the merged table. Valid options are "gz" (compressed CSV), "csv", or "rds". Default is "gz"
overwrite	(logical) whether to overwrite existing downloaded files in the destination directory. Default is FALSE
verbose	(logical) whether to display messages during function execution. Set to TRUE to enable display, or FALSE to run silently. Default is TRUE.

Details

Available datasets that can be requested in datapaper:

- "atlantic": Atlantic data papers compilations.
- "brazil road-kill": Brazil road-kill dataset.

- "dryflor": Latin American and Caribbean Seasonally Dry Tropical Forests (DryFlor) dataset.
- "neotropical xenarthrans": Neotropical Xenarthrans occurrence dataset.
- "neotroptree": NeoTropTree tree species checklist database.
- "dry_diagonal_epiphyte": Vascular epiphytes of the South America Dry Diagonal.
- "gabi": Global Ant Biodiversity Informatics.

Value

A data.table containing the consolidated records from the selected data papers.

Note

When using any dataset retrieved by this function in publications or research reports, please cite the original authors. Use the function cite_datapaper() to obtain the complete bibliographic references.

Examples

```
 ## Not run:
# Download Atlantic dataset
res <- get_datapaper(datapaper = "atlantic")

# Download multiple datasets and filter by a specific species
res_sp <- get_datapaper(
  datapaper = c("atlantic", "dryflor"),
  species = "Araucaria angustifolia")

## End(Not run)
```

---

get_env_bins *Identify Environmental Blocks and Group Nearby Records in Environmental Space*

Description

This function creates a multidimensional grid in environmental space by splitting each environmental variable into n_bins equally sized intervals. It then assigns each occurrence record to an environmental block (bin combination) and identifies records that fall into the same block (i.e., records that are close to each other in environmental space).

The results can be visualized using the plot_env_bins() function.

Usage

```
get_env_bins(
  occ,
  species = "species",
  long = "decimalLongitude",
  lat = "decimalLatitude",
  env_layers,
  n_bins = 5
)
```

Arguments

occ	(data.frame or data.table) a data frame containing the occurrence records for a **single species**. Must contain columns for species, longitude, and latitude.
species	(character) the name of the column in occ that contains the species scientific names. Default is "species".
long	(character) the name of the column in occ that contains the longitude values. Default is "decimalLongitude".
lat	(character) the name of the column in occ that contains the latitude values. Default is "decimalLatitude".
env_layers	(SpatRaster) object containing environmental variables.
n_bins	(numeric) number of bins into which each environmental variable will be divided.

Value

A list with:

- **data**: a data frame including extracted environmental values, bin indices, and a unique block_id for each record.
- **breaks**: a named list of numeric vectors containing the break points for each variable (used by plot_env_bins()).

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Get only occurrences from Araucaria
occ <- occurrences[occurrences$species == "Araucaria angustifolia", ]
# Load example of raster variables
data("worldclim", package = "RuHere")
# Unwrap Packed raster
r <- terra::unwrap(worldclim)
# Get bins
b <- get_env_bins(occ = occ, env_layers = r, n_bins = 5)
```

get_idigbio *get_idigbio*

Description

Downloads species occurrence records from the iDigBio (Integrated Digitized Biocollections) database with flexible taxonomic and geographic filtering options.

Usage

```
get_idigbio(species = NULL, fields = "all", genus = NULL,
family = NULL, order = NULL, phylum = NULL, kingdom = NULL, country = NULL,
county = NULL, limit = NULL, offset = NULL, dir, filename = "idigbio_output",
save = FALSE, compress = FALSE, file.format = "csv", verbose = TRUE, ...)
```

Arguments

species	(character) scientific name(s) of species to search for. Default is NULL.
fields	(character) fields to retrieve from iDigBio. Default is "all".
genus	(character) genus name for filtering results. Default is NULL.
family	(character) family name for filtering results. Default is NULL.
order	(character) order name for filtering results. Default is NULL.
phylum	(character) phylum name for filtering results. Default is NULL.
kingdom	(character) kingdom name for filtering results. Default is NULL.
country	(character) country name for geographic filtering. Default is NULL.
county	(character) county name for geographic filtering. Default is NULL.
limit	(numeric) maximum number of records to retrieve. Default is NULL (no limit).
offset	(numeric) number of records to skip before starting retrieval. Default is NULL (starts at 0).
dir	(character) directory path where the file will be saved. Required if save = TRUE.
filename	(character) name of the output file without extension. Default is "idigbio_output".
save	(logical) if TRUE, saves the results to a CSV file. Default is FALSE.
compress	(logical) if TRUE and save = TRUE, compresses the output file as .csv.zip. Default is FALSE.
file.format	(character) file format for saving output ("csv", "rds"). Default is "csv"
verbose	(logical) if TRUE, prints messages about the progress and the number of species being checked. Default is TRUE.
...	additional arguments passed to ridigbio::idig_search_records().

Value

A data.frame containing occurrence records from iDigBio with the requested fields.

Examples

```
## search for a single species
records_basic <- get_idigbio(species = "Arecaceae")

## search for multiple species
records_multiple <- get_idigbio(
  species = c("Araucaria angustifolia"),
  limit = 100)

## save results as a compressed RDS file
records_saved_rds <- get_idigbio(
  species = "Anacardiaceae",
  limit = 50,
  dir = tempdir(),
  filename = "anacardiaceae_records",
  save = TRUE,
  compress = TRUE,
  file.format = "rds")
```

`get_specieslink` *Download occurrence records from SpeciesLink*

Description

Retrieves occurrence data from the speciesLink network using user-defined filters. The function allows querying by taxonomic, geographic, and collection-related parameters.

Usage

```
get_specieslink(species = NULL, key = NULL, dir,
                        filename = "specieslink_output",save = FALSE,
                      basisOfRecord = NULL, family = NULL, institutionCode = NULL,
                        collectionID = NULL, catalogNumber = NULL,
                        kingdom = NULL, phylum = NULL, class = NULL,
                        order = NULL, genus = NULL, specificEpithet = NULL,
                        infraspecificEpithet = NULL, collectionCode = NULL,
                        identifiedBy = NULL, yearIdentified = NULL,
                        country = NULL, stateProvince = NULL, county = NULL,
                        typeStatus = NULL, recordedBy = NULL,
                        recordNumber = NULL, yearCollected = NULL,
                        locality = NULL, occurrenceRemarks = NULL,
                        barcode = NULL, bbox = NULL, landuse_1 = NULL,
                        landuse_year_1 = NULL, landuse_2 = NULL,
                        landuse_year_2 = NULL, phonetic = FALSE,
                        coordinates = NULL, scope = NULL, synonyms = NULL,
                        typus = FALSE, images = FALSE, redlist = NULL,
```

```
                        limit = NULL, file.format = "csv",
                        compress = FALSE, verbose = TRUE)
```

Arguments

species	(character) species name. Default is NULL.
key	(character) API key or authentication token if required. Default is NULL.
dir	(character) directory where files will be saved (if save = TRUE).
filename	(character) name of the output file without extension. Default is "specieslink_output".
save	(logical) whether to save the results to file. Default is FALSE.
basisOfRecord	(character) filter by basis of record. Default is NULL.
family	(character) family name. Default is NULL.
institutionCode	(character) code of the institution that holds the specimen. Default is NULL.
collectionID	(character) unique identifier for the collection. Default is NULL.
catalogNumber	(character) catalog number of the specimen or record. Default is NULL.
kingdom	(character) kingdom name. Default is NULL.
phylum	(character) phylum name. Default is NULL.
class	(character) class name. Default is NULL.
order	(character) order name. Default is NULL.
genus	(character) genus name. Default is NULL.
specificEpithet	(character) specific epithet of the species. Default is NULL.
infraspecificEpithet	(character) infraspecific epithet. Default is NULL.
collectionCode	(character) code identifying the collection within an institution. Default is NULL.
identifiedBy	(character) name of the person who identified the specimen. Default is NULL.
yearIdentified	(numeric) year of identification. Default is NULL.
country	(character) country name. Default is NULL.
stateProvince	(character) state or province name. Default is NULL.
county	(character) county or municipality name. Default is NULL.
typeStatus	(character) type status. Default is NULL.
recordedBy	(character) collector name. Default is NULL.
recordNumber	(numeric) collector's record number. Default is NULL.
yearCollected	(numeric) year of collection. Default is NULL.
locality	(character) locality description. Default is NULL.
occurrenceRemarks	(character) text field for remarks about the occurrence. Default is NULL.
barcode	(character) barcode or unique specimen identifier. Default is NULL.

bbox	(character) bounding box coordinates in the format "lon_min+lat_min+lon_max+lat_max". Default is NULL.
landuse_1	(character) land use category for the first year. Default is NULL.
landuse_year_1	(numeric) year corresponding to landuse_1. Default is NULL.
landuse_2	(character) land use category for the second year. Default is NULL.
landuse_year_2	(numeric) year corresponding to landuse_2. Default is NULL.
phonetic	(logical) whether to use phonetic matching for taxon names. Default is FALSE.
coordinates	(character) whether to include only records with geographic coordinates ("yes", "no", "original", "automatic", "blocked", "consistent", "suspect")). Default is NULL.
scope	(character) scope of the query ("p", "a", "m", "f", "b"). Default is NULL.
synonyms	(chacarter) whether to include synonyms of the specified taxon ("sp2000", "flora2020", "MycoBank", "algaebase", "DSMZ", "moure"). Default is NULL.
typus	(logical) whether to filter only type specimens. Default is FALSE.
images	(logical) whether to restrict to records with associated images. Default is FALSE.
redlist	(character) filter by IUCN Red List category. Default is NULL.
limit	(numeric) maximum number of records to return. Default is NULL.
file.format	(character) file format for saving output ("csv", "rds"). Default is "csv".
compress	(logical) whether to compress the output file into .zip. Default is FALSE.
verbose	(logical) if TRUE, prints messages about the progress and the number of species being checked. Default is TRUE.

Details

The speciesLink API key can be set permanently using:

```
set_specieslink_credentials("your_api_key")
```

Value

A data.frame containing the occurrence data fields returned by speciesLink.

Note

This function requires a valid speciesLink API key. Set it in advance using set_specieslink_credentials() before running the examples below. You can find your API key at https://specieslink.net/aut/profile/apikeys.

Examples

```
## Not run:
# Set your speciesLink API key (required before running this function)
# set_specieslink_credentials("your_api_key")

# Retrieve records for Arecaceae in São Paulo
res <- get_specieslink(
```

```
  family = "Arecaceae",
  country = "Brazil",
  stateProvince = "São Paulo",
  basisOfRecord = "PreservedSpecimen",
  limit = 10
)

# Save results as compressed CSV
get_specieslink(
  family = "Arecaceae",
  country = "Brazil",
  save = TRUE,
  dir = tempdir(),
  filename = "arecaceae_sp",
  compress = TRUE
)

## End(Not run)
```

---

ggmap_here *Static Visualization of Occurrence Flags with ggplot*

Description

This function creates a static map of occurrence records using **ggplot2**, highlighting which points were flagged by data-validation functions. This visualization helps users quickly inspect spatial patterns of flagged and unflagged records and diagnose potential data-quality issues.

The function can also be used to plot the heatmap generated by the `spatial_kde()` function.

Usage

```
ggmap_here(
  occ,
  species = NULL,
  long = "decimalLongitude",
  lat = "decimalLatitude",
  flags = "all",
  additional_flags = NULL,
  names_additional_flags = NULL,
  col_additional_flags = NULL,
  show_no_flagged = TRUE,
  col_points = NULL,
  size_points = 1,
  heatmap = NULL,
  low_color = "blue",
  mid_color = "yellow",
  high_color = "red",
```

```
  midpoint = 0.5,
  alpha_heatmap = 0.5,
  continent = NULL,
  continent_fill = "gray70",
  continent_linewidth = 0.3,
  continent_border = "white",
  ocean_fill = "aliceblue",
  extension = NULL,
  facet_wrap = FALSE,
  remove_invalid = TRUE,
  theme_plot = ggplot2::theme_minimal(),
  ...
)
```

Arguments

occ	(data.frame or data.table) a dataset containing occurrence records that has been processed by one or more flagging functions. See *Details* for available flag types.
species	(character) name of the species to subset and plot. Default is NULL, meaning that all records for all species are plotted.
long	(character) the name of the column in occ that contains the longitude values. Default is "decimalLongitude".
lat	(character) the name of the column in occ that contains the latitude values. Default is "decimalLatitude".
flags	(character) the flags to be used for coloring the records. Use "all" to display all available flags. See *Details* for all options. Default is "all".
additional_flags	(character) an optional named character vector with the names of additional logical columns to be used as flags. Default is NULL.
names_additional_flags	(character) an optional different name to the flag provided in additional_flags to be shown in the map. Only applicable if additional_flags is not NULL.
col_additional_flags	(character) if additional_flags is provided, the color of the records flagged in this column. Obligatory if additional_flags is not NULL.
show_no_flagged	(logical) whether to display records that did not receive any flag.Default is TRUE.
col_points	(character) A **named vector** assigning colors to each flag. If NULL, default colors from the internal object RuHere::flag_colors() are used.
size_points	(numeric) point size for plotting occurrences. Default is 6.
heatmap	(SpatRaster) an optional heatmap containing the estimated density of occurrence records, typically generated by the spatial_kde() function. Default is NULL.
low_color	(character) color used for the lowest density values in the heatmap. Only applicable if a heatmap is provided. Default is "blue".

mid_color	(character) color used for the midpoint of the heatmap gradient. Default is "yellow".
high_color	(character) color used for the highest density values in the heatmap. Default is "red".
midpoint	(numeric) the central value of the heatmap gradient, corresponding to mid_color. Default is 0.5.
alpha_heatmap	(numeric) Alpha transparency applied to the heatmap layer, ranging from 0 (fully transparent) to 1 (fully opaque). Default is 0.5.
continent	(SpatVector) optional polygon layer representing continent boundaries. If NULL, a built-in simplified world map is used (see *Details*). Default: NULL.
continent_fill	(character) fill color for the continent polygons. Default is "gray70".
continent_linewidth	(numeric) line width for continent boundaries. Default is 0.3.
continent_border	(character) color of the continent polygon borders. Default is "white".
ocean_fill	(character) background color used to represent the ocean. Default is "aliceblue".
extension	(SpatExtent or numeric) optional map extent specified as a SpatExtent or as a numeric vector of length 4 in the order: xmin, xmax, ymin, ymax. Default is NULL (computed automatically from the extent of the occurrence data).
facet_wrap	(logical) whether to plots each flag in a separate panel using ggplot2::facet_wrap(). Default is FALSE.
remove_invalid	(logical) whether to remove records with coordinates outside the valid range for Earth (latitude > 90 or < -90, and longitude > 180 or < -180). Default is TRUE.
theme_plot	(theme) a ggplot2 theme object. Default is ggplot2::theme_minimal().
...	other arguments passed to ggplot2::theme().

Details

This function expects an occurrence dataset that has already been processed by one or more flagging routines from **RuHere** or related packages such as **CoordinateCleaner**. Any logical column in occ can be used as a flag.

The following built-in flag names are recognized: *From RuHere*: correct_country, correct_state, cultivated, florabr, faunabr, wcvp, iucn, bien, duplicated, thin_geo, thin_env, consensus

From CoordinateCleaner: .val, .equ, .zer, .cap, .cen, .sea, .urb, .otl, .gbf, .inst, .aohi

Users may also supply additional logical columns using additional_flags, optionally providing alternative display names (names_additional_flags) and colors (col_additional_flags).

If continent is not provided, the background map is a simplified world polygon included with the package (a modified version of rnaturalearthdata::map_units110). To inspect this object, run:

```
terra::unwrap(getExportedValue("RuHere", "world"))
```

When facet_wrap = TRUE, each flag is plotted in a separate panel, allowing direct comparison among different types of data issues.

Value

An ggplot object displaying flagged and optionally unflagged occurrence records.

Examples

```
# Load example data
data("occ_flagged", package = "RuHere")
# Visualize all flags with ggplot
ggmap_here(occ = occ_flagged)
# Visualize each flag in a separate panel
ggmap_here(occ = occ_flagged, facet_wrap = TRUE)
```

---

ggrid_here *Static Visualization of Richness and Trait Maps*

Description

This function is the dedicated plotting tool for outputs from `richness_here()`. It automatically handles single-layer rasters (e.g., species richness) and multi-layer rasters (e.g., multiple biological traits or flags), creating a standardized visual using **ggplot2**.

Usage

```
ggrid_here(
  raster,
  low_color = "blue",
  mid_color = "yellow",
  high_color = "red",
  alpha = 0.8,
  continent = NULL,
  continent_fill = "gray70",
  continent_linewidth = 0.3,
  continent_border = "white",
  ocean_fill = "aliceblue",
  extension = NULL,
  theme_plot = ggplot2::theme_minimal(),
  ...
)
```

Arguments

`raster`	(SpatRaster) A raster object generated by `richness_here()`.
`low_color`	(character) color for the lowest values. Default is "blue".
`mid_color`	(character) color for the midpoint. Default is "yellow".
`high_color`	(character) color for the highest values. Default is "red".
`alpha`	(numeric) transparency of the grid (0-1). Default is 0.8.

continent	(SpatVector) optional polygon layer for boundaries.
continent_fill	(character) fill color for continents. Default is "gray70".
continent_linewidth	(numeric) line width for continent boundaries. Default is 0.3.
continent_border	(character) color of the continent polygon borders. Default is "white".
ocean_fill	(character) background color for the ocean. Default is "aliceblue".
extension	(SpatExtent or numeric) optional map extent.
theme_plot	(theme) a ggplot2 theme object.
...	other arguments passed to ggplot2::theme().

Value

A ggplot object.

Examples

```
# Load example data
data("occ_flagged", package = "RuHere")

# Simple richness map
r_records <- richness_here(occ_flagged, summary = "records", res = 2)
ggrid_here(r_records)

# Density of specific flags
# Let's see where 'florabr' flags are concentrated
r_flags <- richness_here(occ_flagged, summary = "records",
                         field = "florabr_flag",
                         field_name = "Records flagged by florabr",
                         fun = function(x, ...) sum(!x, na.rm = TRUE),
                         res = 2)
ggrid_here(r_flags)
```

---

import_gbif *Import a download requested from GBIF*

Description

This function imports a dataset downloaded from GBIF using a request key generated by the request_gbif() function. It optionally allows saving the imported occurrences to disk in CSV or GZIP format.

Usage

```
import_gbif(
  request_key,
  write_file = FALSE,
  output_dir = NULL,
  file.format = "gz",
  select_columns = TRUE,
  columns_to_import = NULL,
  overwrite = FALSE,
  ...
)
```

Arguments

request_key	an object of class 'request_key' returned by the request_gbif() function.
write_file	whether to save the downloaded occurrences to disk. Default is FALSE. If TRUE, you must specify the output_dir.
output_dir	(character) a directory to save the data downloaded from GBIF. Only applicable if write_file = TRUE. Default is NULL.
file.format	(character) the format to save the file. Options available are 'csv' (comma-separated values) and 'gz' (compressed GZIP). Only applicable if write_file = TRUE. Default is 'gz'.
select_columns	(logical) whether to import only specific columns (TRUE) or all columns (FALSE) from the occurrence table. Default is TRUE.
columns_to_import	(character) vector of column names to import. Default is NULL, meaning it will import the column names specified in RuHere::gbif_columns data.
overwrite	(logical) whether to overwrite the file in the 'output_dir' if it already exists. Default is FALSE.
...	other arguments passed to rgbif::occ_download_import().

Value

A data frame containing the GBIF occurrence records. If write_file = TRUE, the function also saves the dataset to disk in the specified format.

Note

This function requires an active internet connection.

Examples

```
 ## Not run:
# Prepare data to request GBIF download
gbif_prepared <- prepare_gbif_download(species = "Araucaria angustifolia")
# Submit a request to download occurrences
gbif_requested <- request_gbif(gbif_info = gbif_prepared)
```

```
# Check progress
rgbif::occ_download_wait(gbif_requested)
# After succeeded, import data
occ_gbif <- import_gbif(request_key = gbif_requested)

## End(Not run)
```

`inventory_completeness` *Estimation of inventory completeness and coverage deficit*

Description

Estimates expected species richness, sample coverage (inventory completeness), and coverage deficit for spatial units based on the framework proposed by Chao & Jost (2012).

Usage

```
inventory_completeness(
  occ,
  species = "species",
  long = "decimalLongitude",
  lat = "decimalLatitude",
  raster_base,
  minimum_species = 3,
  maximum_expected = "equal_obs",
  remove_NA = TRUE,
  fill_NA = TRUE,
  return = c("completeness", "deficit")
)
```

Arguments

`occ`	(data.frame or data.table) a data frame containing the occurrence records. Must contain columns for species, longitude, and latitude.
`species`	(character) the name of the column in occ that contains the species names Default is "species".
`long`	(character) the name of the column in occ that contains the longitude values. Default is "decimalLongitude".
`lat`	(character) the name of the column in occ that contains the latitude values. Default is "decimalLatitude".
`raster_base`	(SpatRaster) a reference raster used to aggregate records into spatial units.
`minimum_species`	(numeric) the minimum number of species required in a cell to calculate completeness and deficit. If the number of observed species is lower than this threshold, the function sets completeness = 0 and deficit = 1. Default is 3.

`maximum_expected`
: (numeric or character) The upper limit for the estimated species richness (s_exp). Options include:

  - **"equal_obs"**: Limits s_exp to the maximum observed richness (sefault).
  - **"double_obs"**: Limits s_exp to twice the maximum observed richness found across all cells.
  - **"triple_obs"**: Limits s_exp to three times the maximum observed richness global.
  - **"free"**: No limit is applied to the Chao1 estimator.
  - **numeric**: A fixed integer defining the maximum number of species allowed for any cell.

  This prevents mathematically inflated estimates in cells with extremely low sampling coverage.

`remove_NA`
: (logical) whether to remove sampling units in raster_base where values are NA.

`fill_NA`
: (logical) if TRUE (default), cells within the `raster_base` without occurrence records are assigned completeness = 0 and deficit = 1. If FALSE, these cells remain NA.

`return`
: (character) metrics to return.. Available options are "n", "s_obs", "s_exp", "singletons", "doubletons", "completeness" and "deficit". See details.

Details

The function calculates metrics based on the frequency of rare species (singletons and doubletons) within each cell of the `raster_base`.

- **n**: Total number of records.
- **s_obs**: Observed species richness (number of sampled species).
- **s_exp**: Estimated asymptotic species richness based on the Chao1 estimator.
- **singletons**: Species represented by exactly one record.
- **doubletons**: Species represented by exactly two records.
- **completeness**: Sample coverage, representing the proportion of the total individuals in `occ` that belong to the species in the sample.
- **deficit**: Coverage deficit, which is the probability that the next sampled individual represents a previously unsampled species (1 - completeness)

Value

A SpatRaster object containing the spatialized metrics defined in `return`.

References

Chao A, Jost L (2012) Coverage-based rarefaction and extrapolation: standardizing samples by completeness rather than size. Ecology 93:2533–2547. https://doi.org/10.1890/11-1952.1

Examples

```
# Load example of raster variables
data("worldclim", package = "RuHere")
r <- terra::unwrap(worldclim)
# Aggregate cells
r_base <- terra::aggregate(r, 5)

# Import data set of amphibian communities from the Atlantic Forest
data("atlantic_amphibians", package = "RuHere")

# Run analysis
res <- inventory_completeness(occ = atlantic_amphibians,  raster_base = r_base)
terra::plot(res)
```

---

iucn_here *Download species distribution information from IUCN*

Description

This function downloads information on species distributions from the IUCN Red List, required for filtering occurrence records using specialists' range information via the `flag_iucn()` function.

Usage

```
iucn_here(
  data_dir,
  species,
  synonyms = NULL,
  iucn_credential = NULL,
  overwrite = FALSE,
  progress_bar = FALSE,
  verbose = FALSE,
  return_data = TRUE
)
```

Arguments

`data_dir`	(character) directory to save the data downloaded from IUCN.
`species`	(character) a vector of species names for which to retrieve distribution information.
`synonyms`	(data.frame) an optional data.frame containing synonyms of the target species. The first column must contain the target species names, and the second column their corresponding synonyms. Default is `NULL`. See details for more information.

`iucn_credential` (character) your IUCN API key. Default is `NULL`, in which case the function will attempt to read the API key from your R environment. You can set it in advance using the `set_iucn_credentials()` function.

`overwrite` (logical) whether to overwrite existing files. Default is `FALSE`.

`progress_bar` (logical) whether to display a progress bar during processing. If TRUE, the 'pbapply' package must be installed. Default is `FALSE`.

`verbose` (logical) whether to display progress messages. Default is `FALSE`.

`return_data` (logical) whether to return a data frame containing the species distribution information downloaded from IUCN. Default is `TRUE`.

Details

This function uses the `rredlist::rl_species()` function to retrieve distribution data from the IUCN Red List. The data include information at the country and regional levels, following the World Geographical Scheme for Recording Plant Distributions (WGSRPD) — but applicable to both plants and animals.

Unfortunately, the range polygons available at https://www.iucnredlist.org/resources/spatial-data-download cannot be accessed automatically.

Because taxonomic information in IUCN may be outdated, you can optionally provide a table of synonyms to broaden the search. The synonyms data.frame should have the accepted species in the first column and their synonyms in the second. See `RuHere::synonys` for an example.

The function also downloads the WGSRPD map used to represent distribution regions.

Value

A message indicating that the data were successfully saved in the directory specified by `data_dir`. If `return_data = TRUE`, the function additionally returns a data frame containing the species distribution information retrieved from IUCN.

Note

This function requires a valid IUCN Red List API key. Set it in advance using `set_iucn_credentials()` before running the examples below. You can check or generate your API key at https://api.iucnredlist.org/users/edit.

Examples

```
## Not run:
# Set your IUCN API key (required before running this function)
# set_iucn_credentials(iucn_key = "your_api_key")

# Define a directory to save the data
data_dir <- tempdir() # Here, a temporary directory

# Download species distribution information from IUCN
iucn_here(data_dir = data_dir, species = "Araucaria angustifolia")

## End(Not run)
```

map_here *Interactive Visualization of Occurrence Flags with mapview*

Description

This function creates an interactive map of occurrence records using **mapview**, visually highlighting flags. This tool helps users explore which records were flagged by one or more validation functions and inspect them directly on the map.

Usage

```
map_here(
  occ,
  species = NULL,
  long = "decimalLongitude",
  lat = "decimalLatitude",
  flags = "all",
  additional_flags = NULL,
  names_additional_flags = NULL,
  col_additional_flags = NULL,
  show_no_flagged = TRUE,
  cex = 6,
  lwd = 2,
  col_points = NULL,
  label = NULL,
 map.types = c("OpenStreetMap", "Esri.WorldImagery", "OpenTopoMap", "CartoDB.Positron",
    "CartoDB.DarkMatter"),
  ...
)
```

Arguments

occ	(data.frame or data.table) a dataset containing occurrence records that has been processed by one or more flagging functions. See *Details* for available flag types.
species	(character) name of the species to subset and plot. Default is NULL, meaning that all records for all species are plotted.
long	(character) the name of the column in occ that contains the longitude values. Default is "decimalLongitude".
lat	(character) the name of the column in occ that contains the latitude values. Default is "decimalLatitude".
flags	(character) the flags to be used for coloring the records. Use "all" to display all available flags. See *Details* for all options. Default is "all".
additional_flags	(character) an optional named character vector with the names of additional logical columns to be used as flags. Default is NULL.

`names_additional_flags`
: (character) an optional different name to the flag provided in `additional_flags` to be shown in the map. Only applicable if `additional_flags` is not NULL.

`col_additional_flags`
: (character) if `additional_flags` is provided, the color of the records flagged in this column. Obligatory if `additional_flags` is not NULL.

`show_no_flagged`
: (logical) whether to display records that did not receive any flag.Default is TRUE.

`cex`
: (numeric) point size for plotting occurrences. Default is 6.

`lwd`
: (numeric) line width for point borders. Default is 2.

`col_points`
: (character) A **named vector** assigning colors to each flag. If `NULL`, default colors from the internal object `RuHere::flag_colors()` are used.

`label`
: (character) column name in `occ` to use as the mouseover label. Default is `NULL`, in which case the record's row number is displayed.

`map.types`
: (character) background map layers to display. Defaults to c("OpenStreetMap", "Esri.WorldImagery", "OpenTopoMap", "CartoDB.Positron", "CartoDB.DarkMatter"). See `?mapview::mapview` for additional options.

`...`
: additional arguments passed to `mapview::mapview()`.

Details

The following flags are available: correct_country, correct_state, cultivated, fossil, inaturalist, faunabr, florabr, wcvp, iucn, duplicated, thin_geo, thin_env, .val, .equ, .zer, .cap, .cen, .sea, .urb, .otl, .gbf, .inst, and .aohi.

These flags are typically generated by functions in the `RuHere` or `CoordinateCleaner`workflow to identify potential data-quality issues in occurrence records.

Users may also supply additional logical columns using `additional_flags`, optionally providing alternative display names (`names_additional_flags`) and colors (`col_additional_flags`).

Value

An interactive mapview object displaying flagged and optionally unflagged occurrence records.

Examples

```
# Load example data
data("occ_flagged", package = "RuHere")
# Visualize flags interactively
map_here(occ = occ_flagged, label = "record_id")
```

`moranfast` *Fast Moran's I Autocorrelation Index*

Description

This function computes Moran's I autocorrelation coefficient for a numeric vector `x` using a matrix of weights. The method follows Gittleman and Kot (1990). This function is an implementation of `ape::Moran.I()`, but rewritten in C++ to be substantially faster and more memory-efficient.

Usage

```
moranfast(
  x,
  weight,
  na_rm = TRUE,
  scaled = FALSE,
  alternative = c("two.sided")
)
```

Arguments

`x`	(numeric) A numeric vector (e.g., environmental values extracted from occurrence records).
`weight`	(matrix) A matrix of spatial weights (e.g., a distance or inverse-distance matrix). The number of rows must be equal to the length of `x`.
`na_rm`	(logical) whether to remove missing values from `x`. Default is `TRUE`.
`scaled`	(logical) whether to scale Moran's I so that it ranges between –1 and +1. Default is `TRUE`.
`alternative`	(character) The alternative hypothesis tested against the null hypothesis of no autocorrelation. Must be one of `"two.sided"`, `"less"`, or `"greater"`. Default is `"two.sided"`.

Value

A list with the following components:

- **observed** – The observed Moran's I.
- **expected** – The expected value of Moran's I under the null hypothesis.
- **sd** – The standard deviation of Moran's I under the null hypothesis.
- **p.value** – The p-value of the test based on the chosen `alternative`.

References

Gittleman, J. L., & Kot, M. (1990). Adaptation: statistics and a null model for estimating phylogenetic effects. *Systematic Zoology*, 39(3), 227–241.

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Filter occurrences of Araucaria
occ <- occurrences[occurrences$species == "Araucaria angustifolia", ]
# Load example of raster variables
data("worldclim", package = "RuHere")
# Unwrap Packed raster
r <- terra::unwrap(worldclim)
# Extract values for bio_1
bio_1 <- terra::extract(r$bio_1,
                        occ[, c("decimalLongitude", "decimalLatitude")],
                        ID = FALSE, xy = TRUE)
#Remove NAs
bio_1 <- na.omit(bio_1)
# Convert values to numeric
v <- as.numeric(bio_1$bio_1)
# Compute geographic distance matrix
d <- fields::rdist.earth(x1 = as.matrix(bio_1[, c("x", "y")]), miles = FALSE)
# Inverse-distance weights
d <- 1/d
# Fill diagonal with 0
diag(d) <- 0
# Remove finite values
d[is.infinite(d)] <- 0
# Compute Moran's I
m <- moranfast(x = v, weight = d, scale = TRUE)
# Print results
m
```

---

occ_bien *Occurrence records of Yellow Trumpet Tree from BIEN*

Description

A cleaned dataset of occurrence records for Yellow Trumpet Tree (*Handroanthus serratifolius*) retrieved from the BIEN database. The raw data were downloaded using `get_bien()`

The dataset was subsequently processed with the package's internal flagging workflow (`flag_duplicates()` and `remove_flagged()`) to remove duplicated records.

Usage

```
occ_bien
```

Format

A data frame containing spatial coordinates, taxonomic information, and metadata returned by BIEN, after cleaning. Columns include (but may not be limited to):

- scrubbed_species_binomial: Cleaned species name
- longitude, latitude: Geographic coordinates
- country, state_province, and other political boundary fields

See Also

get_bien()

Examples

```
# View dataset
head(occ_bien)

# Number of records
nrow(occ_bien)
```

---

occ_flagged *Flagged occurrence records of* Araucaria angustifolia

---

Description

A dataset containing the occurrence records of *Araucaria angustifolia* after applying several of the package's flagging and data-quality assessment functions.

Usage

```
occ_flagged
```

Format

A data frame where each row corresponds to a georeferenced occurrence of *A. angustifolia*.

References

Derived dataset GBIF.org (23 August 2026) Filtered export of GBIF occurrence data https://doi.org/10.15468/dd.xezdws

See Also

occurrences, standardize_countries(), standardize_states(), flag_florabr(), flag_wcvp(), flag_iucn(), flag_cultivated(), flag_inaturalist(), flag_duplicates(), mapview_here()

Examples

```
# First rows
head(occ_flagged)

# Count flagged vs. unflagged records
table(occ_flagged$correct_country)
```

`occ_gbif` *Occurrence records of* Araucaria angustifolia *from GBIF*

Description

A cleaned dataset of occurrence records for *Araucaria angustifolia* (Parana pine) retrieved from GBIF.

Records were downloaded using the package's GBIF workflow (`prepare_gbif_download()`, `request_gbif()`, `import_gbif()`), and then cleaned using the internal flagging workflow (duplicate detection and removal).

Usage

```
occ_gbif
```

Format

A data frame containing georeferenced GBIF occurrence records for *A. angustifolia* after all cleaning steps.

References

Derived dataset GBIF.org (23 August 2026) Filtered export of GBIF occurrence data https://doi.org/10.15468/dd.aqxw37

See Also

`prepare_gbif_download()`, `request_gbif()`, `import_gbif()`, `flag_duplicates()`, `remove_flagged()`

Examples

```
# Preview dataset
head(occ_gbif)

# Number of cleaned records
nrow(occ_gbif)
```

occ_idig *Occurrence records of azure jay from iDigBio*

Description

A cleaned dataset of occurrence records for azure jay (*Cyanocorax caeruleus*) retrieved from the iDigBio using `get_idigbio()`.

Records were cleaned using the package's internal duplicate-flagging workflow.

Usage

```
occ_idig
```

Format

A data frame containing georeferenced iDigBio occurrence records for *C. caeruleus* after all cleaning steps.

See Also

`get_idigbio()`, `flag_duplicates()`, `remove_flagged()`

Examples

```
# First rows
head(occ_idig)

# Number of cleaned records
nrow(occ_idig)
```

occ_splink *Occurrence records of azure jay from SpeciesLink*

Description

A cleaned dataset of occurrence records for azure jay (*Cyanocorax caeruleus*) retrieved from the SpeciesLink using `get_specieslink()`.

Records were cleaned using the package's internal duplicate-flagging workflow.

Usage

```
occ_splink
```

Format

A data frame containing georeferenced SpeciesLink occurrence records for *C. caeruleus* after all cleaning steps.

See Also

`get_specieslink()`, `flag_duplicates()`, `remove_flagged()`

Examples

```
# First rows
head(occ_splink)

# Number of cleaned records
nrow(occ_splink)
```

---

occurrences *Integrated occurrence dataset for three example species*

Description

A harmonized, multi-source occurrence dataset containing cleaned georeferenced records for three species:

- *Araucaria angustifolia* (Parana pine)
- *Cyanocorax caeruleus* (Azure jay)
- *Handroanthus serratifolius* (Yellow trumpet tree)

Records were retrieved from **GBIF**, **speciesLink**, **BIEN**, and **iDigBio**, standardized through the package workflow, merged, and cleaned to remove duplicates.

Usage

```
occurrences
```

Format

A data frame where each row represents a georeferenced occurrence record for one of the three species.

Columns correspond to the standardized output of `format_columns()`, including:

- `species`: Cleaned binomial species name
- `decimalLongitude`, `decimalLatitude`: Coordinates
- `year`: Year of collection/observation
- Various taxonomic, temporal, locality, and metadata fields
- Source identifiers added by `format_columns()` (e.g., `data_source`)

References

Derived dataset GBIF.org (23 August 2026) Filtered export of GBIF occurrence data https://doi.org/10.15468/dd.6nyzke

See Also

`format_columns()`, `bind_here()`, `flag_duplicates()`, `remove_flagged()`

Examples

```
# Show the first rows
head(occurrences)

# Number of occurrences per species
table(occurrences$species)
```

---

plot_env_bins *Plot Environmental Bins (2D Projection)*

Description

Visualize the output of `get_env_bins()` by plotting environmental blocks (bins) along two selected environmental variables. Each block is shown as a colored rectangle, and points falling inside the same rectangle share the same `block_id`.

Usage

```
plot_env_bins(
  env_bins,
  x_var,
  y_var,
  alpha_blocks = 0.3,
  color_points = "black",
  size_points = 2,
  alpha_points = 0.5,
  stroke_points = 1,
  xlab = NULL,
  ylab = NULL,
  theme_plot = ggplot2::theme_minimal()
)
```

Arguments

`env_bins`	(list) output list from `get_env_bins()`. Must contain: • `data`: data.frame with environmental values, bin indices, and block_id • `breaks`: named list with breakpoints for each variable
`x_var`	(character) name of the environmental variable used on the x-axis.

y_var	(character) name of the environmental variable used on the y-axis.
alpha_blocks	(numeric) transparency level of the block rectangles. Must be between 0 and 1. Default is 0.3.
color_points	(character) color of the points representing occurrence records. Default is "black".
size_points	(numeric) size of the points representing occurrence records. Default is 2.
alpha_points	(numeric) transparency level of the points. Must be between 0 and 1. Default is 0.5..
stroke_points	(numeric) size of the border of the points. Default is 1.
xlab	(character) label for the x-axis. Default is NULL, meaning the name provided in x_var will be used.
ylab	(character) label for the y-axis. Default is NULL, meaning the name provided in y_var will be used.
theme_plot	(theme) a ggplot2 theme object. Default is ggplot2::theme_minimal().

Value

A ggplot object showing the environmental blocks (colored rectangles) and the occurrence records in the selected environmental space.

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Get only occurrences from Araucaria
occ <- occurrences[occurrences$species == "Araucaria angustifolia", ]
# Load example of raster variables
data("worldclim", package = "RuHere")
# Unwrap Packed raster
r <- terra::unwrap(worldclim)
# Get bins
b <- get_env_bins(occ = occ, env_layers = r, n_bins = 10)
# Plot
plot_env_bins(b, x_var = "bio_1", y_var = "bio_12",
              xlab = "Temperature", ylab = "Precipitation")
```

---

prepare_gbif_download *Prepare data to request GBIF download*

Description

Prepare data to request GBIF download

Usage

```
prepare_gbif_download(
  species,
  rank = NULL,
  kingdom = NULL,
  phylum = NULL,
  class = NULL,
  order = NULL,
  family = NULL,
  genus = NULL,
  strict = FALSE,
  progress_bar = FALSE,
  ...
)
```

Arguments

species	(character) a vector of species name(s).
rank	(character) optional taxonomic rank (for example, 'species' or 'genus'). Default is NULL, meaning it will return species matched across all ranks.
kingdom	(character) optional taxonomic kingdom (for example, 'Plantae' or 'Animalia'). Default is NULL, meaning it will return species matched across all kingdoms.
phylum	(character) optional taxonomic phylum. Default is NULL, meaning it will return species matched across all phyla.
class	(character) optional taxonomic class. Defaults is NULL, meaning it will return species matched across all classes.
order	(character) optional taxonomic order. Defaults is NULL, meaning it will return species matched across all orders
family	(character) optional taxonomic family. Defaults is NULL, meaning it will return species matched across all families.
genus	(character) optional taxonomic genus. Defaults is NULL, meaning it will return species matched across all genus.
strict	(logical) If TRUE, it (fuzzy) matches only the given name, but never a taxon in the upper classification. Default is FALSE.
progress_bar	(logical) whether to display a progress bar during processing. If TRUE, the 'pbapply' package must be installed. Default is FALSE.
...	other parameters passed to rgbif::occ_count().

Value

A data.frame with species information, including the number of occurrences and other related details.

Note

This function requires an active internet connection to access GBIF data.

Examples

```
gbif_prepared <- prepare_gbif_download(species = "Araucaria angustifolia")
```

`prepared_metadata` *Metadata templates used internally by* `format_columns()`

Description

A named list of data frames containing metadata templates for the main biodiversity data providers supported by the package (GBIF, SpeciesLink, iDigBio, and BIEN).

These templates are used internally by `format_columns()` to harmonize columns.

Usage

```
prepared_metadata
```

Format

A named list of four data frames:

- `$gbif` — template for GBIF dataset.
- `$specieslink` — template for SpeciesLink dataset.
- `$idigbio` — template for iDigBio dataset.
- `$bien` — template for BIEN dataset.

Details

Each element of `prepared_metadata` is a single-row data frame where:

- **column names** correspond to the package's standardized output fields
- **values in the row** represent the original column names used by each data provider

These mappings allow `format_columns()` to:

- rename fields (e.g., `scientificname` → `scientificName`)
- identify which variables are missing or provider-specific
- coerce classes consistently (e.g., dates, coordinates)
- ensure compatibility when combining datasets from different sources

See Also

`format_columns()`

Examples

```
# View template for GBIF records
prepared_metadata$gbif
```

puma_atlanticr *Occurrence records of* Puma concolor *from AtlanticR*

Description

A subset of Atlantic mammals records obtained from the `atlanticr::atlantic_mammals` dataset, containing occurrences of *Puma concolor*.

This dataset is provided as an example to illustrate how to create user-defined metadata templates for occurrence records from external sources using the package's `create_metadata()` function.

Usage

```
puma_atlanticr
```

Format

A data frame where each row represents a single occurrence record of *Puma concolor*. Columns include species name, location, and other relevant metadata fields provided by the `atlantic_mammals` dataset.

See Also

`create_metadata()`, `format_columns()`

Examples

```
# Preview first rows
head(puma_atlanticr)

# Count occurrences per year
table(puma_atlanticr$year)
```

relocate_after *Relocate a column in a data frame*

Description

These functions move one column to a new position in a data frame, either immediately *after* or *before* another column, while preserving the order of all remaining columns. They are lightweight base-R utilities equivalent to `dplyr::relocate()`, but without external dependencies.

Usage

```
relocate_after(df, col, after)

relocate_before(df, col, before)
```

Arguments

df	(data.frame) a data.frame whose columns will be reordered.
col	(character) the name of the column to move.
after	(character) for `relocate_after()`: the column after which `col` will be placed.
before	(character) for `relocate_before()`: the column before which `col` will be placed.

Value

A data.frame with columns reordered.

---

remove_accent *Remove accents and special characters from strings*

Description

This function removes accents and replaces special characters from strings, returning a plain-text version suitable for data cleaning or standardization.

Usage

```
remove_accent(s)
```

Arguments

s	(character) a character vector containing the strings to process.

Value

A vector string without accents or special characters.

Examples

```
remove_accent(c("Colômbia", "São Paulo"))
```

remove_flagged *Remove flagged records*

Description

This function removes occurrence records flagged as invalid by one or more flagging functions. Additional manual control is available to force keeping or removing specific records, regardless of their flag values.

Usage

```
remove_flagged(
  occ,
  flags = "all",
  additional_flags = NULL,
  force_keep = NULL,
  force_remove = NULL,
  remove_NA = FALSE,
  column_id = "record_id",
  save_flagged = FALSE,
  output_dir = NULL,
  overwrite = FALSE,
  output_format = ".gz"
)
```

Arguments

occ	(data.frame or data.table) a dataset with occurrence records that has been processed by two or more flagging functions. See details.
flags	(character) a character vector with the names of the flag columns to be used for filtering records. See *details* for the available options. Default is "all".
additional_flags	(character) an optional named character vector with the names of additional logical columns to be used as flags. Default is NULL.
force_keep	(character) an optional character vector with the IDs of records that were flagged but should still be kept. Default is NULL.
force_remove	(character) an optional character vector with the IDs of records that were not flagged but should still be removed. Default is NULL.
remove_NA	(logical) whether to remove records that have NA in the flags specified. Default is FALSE.
column_id	(character) the name of the column containing unique record IDs. Required if force_keep or force_remove is used. Default is NULL.
save_flagged	(logical) whether to save the flagged (removed) records. If TRUE, an output_dir must be provided. Default is FALSE.

output_dir	(character) path to an existing directory where removed flagged records will be saved. Only used when save_flagged = TRUE.
overwrite	(logical) whether to overwrite existing files in output_dir. Only used when save_flagged = TRUE. Default is FALSE.
output_format	(character) output format for saving removed records. Options are ".csv" or ".gz". Only used when save_flagged = TRUE. Default is ".gz".

Details

The following flags are available: correct_country, correct_state, cultivated, fossil, inaturalist, faunabr, florabr, wcvp, iucn, duplicated, thin_geo, thin_env, .val, .equ, .zer, .cap, .cen, .sea, .urb, .otl, .gbf, .inst, and .aohi.

Value

A data.frame containing only the valid (kept) records according to the flags and additional criteria.

Examples

```
# Load example data
data("occ_flagged", package = "RuHere")

# Remove all flagged records
occ_valid <- remove_flagged(occ = occ_flagged)

# Remove flagged records and force removal of some unflagged records
to_remove <- c("gbif_5987", "specieslink_2301", "gbif_18761")
occ_valid2 <- remove_flagged(occ = occ_flagged,
                             force_remove = to_remove)

# Remove flagged records but keep some flagged ones
to_keep <- c("gbif_14501", "gbif_12002", "gbif_5168")
occ_valid3 <- remove_flagged(occ = occ_flagged,
                            force_keep = to_keep)
```

---

remove_invalid_coordinates

Identify and remove invalid coordinates

Description

This function identifies and removes invalid geographic coordinates, including non-numeric values, NA or empty values, and coordinates outside the valid range for Earth (latitude > 90 or < -90, and longitude > 180 or < -180).

Usage

```
remove_invalid_coordinates(
  occ,
  long = "decimalLongitude",
  lat = "decimalLatitude",
  return_invalid = TRUE,
  save_invalid = FALSE,
  output_dir = NULL,
  overwrite = FALSE,
  output_format = ".gz",
  verbose = FALSE
)
```

Arguments

occ	(data.frame or data.table) a dataset with occurrence records.
long	(character) column name in occ with the longitude.
lat	(character) column name in occ with the latitude.
return_invalid	(logical) whether to return a list containing records that passed and failed this test. Default is TRUE.
save_invalid	(logical) whether to save the records that failed this test (i.e., flagged as FALSE). If TRUE, an output_dir must be provided. Default is FALSE.
output_dir	(character) path to an existing directory where records flagged as FALSE will be saved. Only used when save_invalid = TRUE.
overwrite	(logical) whether to overwrite existing files in output_dir. Only used when save_invalid = TRUE. Default is FALSE.
output_format	(character) output format for saving flagged records. Options are ".csv" or ".gz". Only used when save_invalid = TRUE. Default is ".gz".
verbose	(logical) whether to print messages about function progress. Default is TRUE.

Value

The input data.frame with an additional logical column indicating whether each record passed (TRUE) or failed (FALSE) the coordinate quality check (i.e., is not missing, non-numeric, or outside the possible range for Earth: latitude between -90 and 90, longitude between -180 and 180). As with all other flagging functions in RuHere, TRUE indicates that the record passed this test and is eligible for retention; FALSE indicates it failed and is flagged as potentially problematic, and can be removed using remove_flagged().

If return_invalid = FALSE, returns the occurrence dataset containing only records flagged as TRUE. If return_invalid = TRUE (default), returns a list with two elements:

- valid – the dataset with records flagged as TRUE.
- invalid – the dataset with records flagged as FALSE.

Examples

```
# Create fake data example
occ <- data.frame("species" = "spp",
                  "decimalLongitude" = c(10, -190, 20, 50, NA),
                  "decimalLatitude" = c(20, 20, 240, 50, NA))
# Split valid and invalid coordinates
occ_valid <- remove_invalid_coordinates(occ)
```

---

request_gbif *Submit a request to download occurrence data from GBIF.*

Description

Submit a request to download occurrence data from GBIF.

Usage

```
request_gbif(gbif_info, hasCoordinate = TRUE,
                    hasGeospatialIssue = FALSE, format = "DWCA",
                    gbif_user = NULL, gbif_pwd = NULL, gbif_email = NULL,
                    additional_predicates = NULL)
```

Arguments

gbif_info	an object of class 'gbif_info' resulted by the prepare_gbif_download() function.
hasCoordinate	(logical) whether to retrieve only records with coordinates. Default is TRUE.
hasGeospatialIssue	(logical) whether to retrieve records identified with geospatial issue. Default is FALSE.
format	(character) the download format. Options available are 'DWCA', 'SIMPLE_CSV', or 'SPECIES_LIST', Default is DWCA'.
gbif_user	(character) user name within GBIF's website. Default is NULL, meaning it will try to obtain this information from the R enviroment. (check set_gbif_credentials()) for more details.
gbif_pwd	(character) user password within GBIF's website. Default is NULL, meaning it will try to obtain this information from the R enviroment.
gbif_email	(character) user email within GBIF's website. Default is NULL, meaning it will try to obtain this information from the R enviroment.
additional_predicates	(character or occ_predicate) additional supported predicates that can be combined to build more complex download requests. See rgbif::pred() for details.

Details

You can use the object returned by this function to check the download request progress with `rgbif::occ_download_wait()`

Value

A download request key returned by the GBIF API, which can be used to monitor or retrieve the download.

Note

This function requires an active internet connection and valid GBIF credentials. Set them in advance using `set_gbif_credentials()`.

Examples

```
## Not run:
# Set your GBIF credentials (required before running this function)
# set_gbif_credentials(gbif_username = "your_username",
#                      gbif_email = "your_email@example.com",
#                      gbif_password = "your_password")

# Prepare data to request GBIF download
gbif_prepared <- prepare_gbif_download(species = "Araucaria angustifolia")
# Submit a request to download occurrences
gbif_requested <- request_gbif(gbif_info = gbif_prepared)
# Check progress
rgbif::occ_download_wait(gbif_requested)

## End(Not run)
```

---

request_gbif_specieslist

Download a list of species recorded inside a spatial polygon from GBIF.

Description

Download a list of species recorded inside a spatial polygon from GBIF, with support for optional higher-taxon filtering.

Usage

```
request_gbif_specieslist(
  spatial_polygon,
  kingdom = NULL,
  phylum = NULL,
  class = NULL,
```

```
  order = NULL,
  family = NULL,
  genus = NULL,
  species = NULL,
  tolerance = 0.01,
  gbif_user = NULL,
  gbif_pwd = NULL,
  gbif_email = NULL,
  verbose = TRUE
)
```

Arguments

spatial_polygon	an object of class SpatVector representing the area of interest.
kingdom, phylum, class, order, family, genus, species	(character) optional taxonomic filters. Default is NULL.
tolerance	(numeric) tolerance in degrees for geometry simplification. Default is 0.01.
gbif_user	(character) user name within GBIF's website. Default is NULL, meaning it will try to obtain this information from the R enviroment. (check set_gbif_credentials()) for more details.
gbif_pwd	(character) user password within GBIF's website. Default is NULL, meaning it will try to obtain this information from the R enviroment.
gbif_email	(character) user email within GBIF's website. Default is NULL, meaning it will try to obtain this information from the R enviroment.
verbose	(logical) if TRUE, prints messages about the progress. Default is FALSE.

Details

This function converts the input SpatVector polygon into Well-Known Text (WKT) format and submits an asynchronous query using GBIF's SPECIES_LIST download format.

You can use the object returned by this function to check the download request progress with rgbif::occ_download_wait()

Value

A download request key returned by the GBIF API, which can be used to monitor or retrieve the download of the species list.

Note

This function requires an active internet connection and valid GBIF credentials. Set them in advance using set_gbif_credentials().

Examples

```
## Not run:
# Set your GBIF credentials (required before running this function)
# set_gbif_credentials(gbif_username = "your_username",
#                      gbif_email = "your_email@example.com",
#                      gbif_password = "your_password")

# Create a sample polygon
coords <- matrix(c(-48, -16, -47, -16, -47, -15, -48, -15, -48, -16),
                 ncol = 2, byrow = TRUE)
poly <- vect(coords, type = "polygons", crs = "EPSG:4326")

# Submit a request to download list of Malvaceae species in the area
gbif_requested_sl <- request_gbif_specieslist(spatial_polygon = poly,
                                              family = "Malvaceae")
# Monitor download progress
rgbif::occ_download_wait(gbif_requested_sl)

# Import completed species list
sl <- import_gbif(gbif_requested_sl, select_columns = FALSE)

## End(Not run)
```

---

richness_here *Species Richness and Occurrence Summary Mapping*

Description

This function generates spatial grids (rasters) of species richness, record density, or summarized biological traits from occurrence data. It supports custom resolutions, masking, and automatic coordinate reprojection to match reference rasters.

Usage

```
richness_here(
  occ,
  species = "species",
  long = "decimalLongitude",
  lat = "decimalLatitude",
  records = "record_id",
  raster_base = NULL,
  res = NULL,
  crs = "epsg:4326",
  mask = NULL,
  summary = "records",
  field = NULL,
  field_name = NULL,
  fun = mean,
```

```
  verbose = TRUE
)
```

Arguments

occ	(data.frame) a dataset containing occurrence records. Must include columns for species names and geographic coordinates.
species	(character) the name of the column in occ that contains the species scientific names. Default is "species".
long	(character) the name of the column in occ that contains the longitude values. Default is "decimalLongitude".
lat	(character) the name of the column in occ that contains the latitude values. Default is "decimalLatitude".
records	(character) the name of the column in occ that contains the record names. Default is "record_id".
raster_base	(SpatRaster) an optional reference raster. If provided, the output will match its resolution, extent, and CRS. Default is NULL.
res	(numeric) the desired resolution (in decimal degrees if WGS84) for the output grid. Only used if raster_base is NULL.
crs	(character) the coordinate reference system of the raster. (see ?terra::crs). Default is "epsg:4326". Only applicable if raster_base is not provided.
mask	(SpatRaster or SpatVector) an optional layer to mask the final output. Default is NULL.
summary	(character) the type of summary to calculate. Either "records" (number of occurrences per cell) or "species" (number of unique species per cell). Default is "records".
field	(character or named vector) column in occ to summarize (e.g., traits). If a named vector is provided, names must match species in occ. Used to summarize traits or flags in both 'species' and 'records' modes. Default is NULL.
field_name	(character) a custom name used to build the legend when plotting the result with ggrid_here(). Only applicable if field is provided. Default is NULL, meaning it will use the same column name provided in field.
fun	(function) the function to aggregate field values (e.g., mean, max, sum). Default is mean.
verbose	(logical) whether to print messages about the progress. Default is TRUE.

Value

A SpatRaster object representing the calculated richness, density, or trait summary.

Examples

```
# Load example data
data("occ_flagged", package = "RuHere")
```

```
# Mapping the density of records
r_density <- richness_here(occ_flagged, summary = "records", res = 0.5)
ggrid_here(r_density)

# We can also summarize key features:
# 1. Identifying problematic regions by summing error flags
# We create a variable to store the sum of logical flags (TRUE = 1, FALSE = 0)
total_flags <- occ_flagged$florabr_flag +
 occ_flagged$wcvp_flag +
 occ_flagged$iucn_flag +
 occ_flagged$cultivated_flag +
 occ_flagged$inaturalist_flag +
 occ_flagged$duplicated_flag
names(total_flags) <- occ_flagged$record_id

# Using summary = "records" with to see the average accumulation of errors
# with fun = mean to see the average accumulation
r_flags <- richness_here(occ_flagged, summary = "records",
                         field = total_flags,
                         field_name = "Number of flags",
                         fun = mean, res = 0.5)
ggrid_here(r_flags)

# 2. Or we can summarize organisms traits spatially
# Simulating a trait (e.g., mass) for each unique record
spp <- unique(occ_flagged$record_id)
sim_mass <- setNames(runif(length(spp), 10, 50), spp)

r_trait <- richness_here(occ_flagged, summary = "records",
                         field = sim_mass, field_name = "Mass",
                         fun = mean, res = 0.5)
ggrid_here(r_trait)
```

---

set_gbif_credentials *Store GBIF credentials*

Description

This function sets GBIF credentials (username, email and password) as environment variables in the R environment. These credentials are required to retrieve occurrence records from GBIF.

Usage

```
set_gbif_credentials(
  gbif_username,
  gbif_email,
  gbif_password,
  permanently = FALSE,
  overwrite = FALSE,
  open_Renviron = FALSE,
```

```
  verbose = TRUE
)
```

Arguments

gbif_username	(character) your GBIF username.
gbif_email	(character) your GBIF email address.
gbif_password	(character) your GBIF password.
permanently	(logical) whether to add the GBIF credentials permanently to the R environment. Default is FALSE, meaning it will be added only temporarily for the current session.
overwrite	(logical) whether to overwrite GBIF credentials if they already exist. Only applicable if permanently is set to TRUE. Default is FALSE.
open_Renviron	(logical) whether to open the .Renviron file after saving the credentials. Only applicable if permanently is set to TRUE. Default is FALSE.
verbose	(logical) if TRUE, prints messages about the progress and the number of species being checked. Default is TRUE.

Value

If permanently and open_Renviron are set to TRUE, it opens the .Renviron file. Otherwise, the credentials are saved silently.

Examples

```
## Not run:
set_gbif_credentials(gbif_username = "my_username",
                     gbif_email = "my_email@example.com",
                     gbif_password = "my_password")

## End(Not run)
```

---

set_iucn_credentials *Store SpeciesLink credential*

Description

This function sets the IUCN API key as an environment variable in the R environment. This key is required to obtain distributional data from IUCN.

Usage

```
set_iucn_credentials(
  iucn_key,
  permanently = FALSE,
  overwrite = FALSE,
  open_Renviron = FALSE,
  verbose = TRUE
)
```

Arguments

iucn_key	(character) your IUCN API key. See Details.
permanently	(logical) whether to add the SpeciesLink API key permanently to the R environment. Default is FALSE, meaning it will be added only temporarily for the current session.
overwrite	(logical) whether to overwrite IUCN credential if it already exists. Only applicable if permanently is set to TRUE. Default is FALSE.
open_Renviron	(logical) whether to open the .Renviron file after saving the credential. Only applicable if permanently is set to TRUE. Default is FALSE.
verbose	(logical) if TRUE, prints messages about the progress and the number of species being checked. Default is TRUE.

Details

To check your API key, visit: https://api.iucnredlist.org/users/edit.

Value

If permanently and open_Renviron are set to TRUE, it opens the .Renviron file. Otherwise, the credentials are saved silently.

Examples

```
## Not run:
set_iucn_credentials(iucn_key = "my_key")

## End(Not run)
```

---

set_specieslink_credentials

Store SpeciesLink credential

Description

This function sets the SpeciesLink API key as an environment variable in the R environment. This API key is required to retrieve occurrence records from SpeciesLink.

Usage

```
set_specieslink_credentials(
  specieslink_key,
  permanently = FALSE,
  overwrite = FALSE,
  open_Renviron = FALSE,
  verbose = TRUE
)
```

Arguments

specieslink_key	(character) your SpeciesLink API key.
permanently	(logical) whether to add the SpeciesLink API key permanently to the R environment. Default is FALSE, meaning it will be added only temporarily for the current session.
overwrite	(logical) whether to overwrite SpeciesLink credential if it already exists. Only applicable if permanently is set to TRUE. Default is FALSE.
open_Renviron	(logical) whether to open the .Renviron file after saving the credential. Only applicable if permanently is set to TRUE. Default is FALSE.
verbose	(logical) if TRUE, prints messages about the progress and the number of species being checked. Default is TRUE.

Details

To check your API key, visit: https://specieslink.net/aut/profile/apikeys.

Value

If permanently and open_Renviron are set to TRUE, it opens the .Renviron file. Otherwise, the credentials are saved silently.

Examples

```
## Not run:
# Set your speciesLink API key (required before running this function)
# set_specieslink_credentials(specieslink_key = "your_api_key")

# Retrieve records for Arecaceae in São Paulo
res <- get_specieslink(
  family = "Arecaceae",
  country = "Brazil",
  stateProvince = "São Paulo",
  basisOfRecord = "PreservedSpecimen",
  limit = 10
)

# Save results as compressed CSV
get_specieslink(
  family = "Arecaceae",
  country = "Brazil",
  save = TRUE,
  dir = tempdir(),
  filename = "arecaceae_sp",
  compress = TRUE
)

## End(Not run)
```

spatial_kde *Kernel Density Estimation (Heatmap) for occurrence data*

Description

This function creates density heatmaps using kernel density estimation. The algorithm is inspired by the **SpatialKDE** R package and the "Heatmap" tool from QGIS. Each occurrence contributes to the density surface within a circular neighborhood defined by a specified radius.

Usage

```
spatial_kde(
  occ,
  long = "decimalLongitude",
  lat = "decimalLatitude",
  radius = 0.2,
  resolution = NULL,
  buffer_extent = 500,
  crs = "epsg:4326",
  raster_ref = NULL,
  kernel = "quartic",
  scaled = TRUE,
  decay = 1,
  mask = NULL,
  zero_as_NA = FALSE,
  weights = NULL
)
```

Arguments

occ	(data.frame, data.table, or SpatVector) a data frame or SpatVector containing the occurrences. Must contain columns longitude and latitude.
long	(character) the name of the column in occ that contains the longitude values. Default is "decimalLongitude".
lat	(character) the name of the column in occ that contains the latitude values. Default is "decimalLatitude".
radius	(numeric) a positive numeric value specifying the smoothing radius for the kernel density estimate. This parameter determines the circular neighborhood around each point where that point will have an influence. See details. Default is 0.2.
resolution	(numeric) a positive numeric value specifying the resolution (in degrees or meters, depending on the crs provided) of the resultant raster heatmap. Required if raster_ref is not provided. Default is NULL.
buffer_extent	(numeric) width of the buffer (in kilometers) to draw around the occurrences to define the area for computing the heatmap. Default is 500.
crs	(character) the coordinate reference system of the raster heatmap (see ?terra::crs). Default is "epsg:4326". Only applicable if raster_ref is not provided.

raster_ref	(SpatRaster) an optional raster to use as reference for resolution, CRS, and extent. Default is NULL.
kernel	(character) type of kernel to use. Available kernerls are "uniform", "quartic", "triweight", "epanechnikov", or "triangular". Default is "quartic".
scaled	(logical) whether to scale output values to vary between 0 and 1. Default is TRUE.
decay	(numeric) decay parameter for "triangular" kernel. Only applicable if `kernel = triangular`.
mask	(SpatRaster or SpatExtent) optional spatial object to define the extent of the area for the heatmap. Default is NULL, in which case the extent is derived from `raster_ref` (if provided) or from the convex hull of occurrences plus `buffer_extent`.
zero_as_NA	(logical) whether to convert regions with value 0 to NA. Default is FALSE.
weights	(numeric) optional vector of weights for individual points. Must be the same length as the number of occurrences in `occ`. Default is NULL.

Details

The `radius` parameter controls how far the influence of each observation extends. Smaller values produce fine-grained peaks; larger values produce smoother, more spread-out heatmaps. Units depend on the CRS: degrees for geographic coordinates (default), meters for projected coordinates.

If `raster_ref` is not provided, the extent is calculated from the convex hull of `occ` plus `buffer_extent`.

Kernels define the weight decay of points: "uniform" = constant, "quartic"/"triweight"/"epanechnikov" = smooth, and "triangular" = linear decay (using `decay` parameter).

Value

A SpatRaster containing the kernel density values.

References

- Hart, T., & Zandbergen, P. (2014). Kernel density estimation and hotspot mapping: Examining the influence of interpolation method, grid cell size, and radius on crime forecasting. **Policing: An International Journal of Police Strategies & Management**, 37(2), 305-323.
- Nelson, T. A., & Boots, B. (2008). Detecting spatial hot spots in landscape ecology. **Ecography**, 31(5), 556-566.
- Chainey, S., Tompson, L., & Uhlig, S. (2008). The utility of hotspot mapping for predicting spatial patterns of crime. **Security journal**, 21(1), 4-28.
- Caha J (2023). SpatialKDE: Kernel Density Estimation for Spatial Data. **https://jancaha.github.io/SpatialKDE/index**

Examples

```
# Load example data
data("occ_flagged", package = "RuHere")
# Remove flagged records
occ <- remove_flagged(occ_flagged)
```

```
# Generate heatmap
heatmap <- spatial_kde(occ = occ, resolution = 0.25, buffer_extent = 50,
                       radius = 2)
# Plot heatmap with terra
terra::plot(heatmap)
# Plot heatmap with ggplot
ggmap_here(occ = occ, heatmap = heatmap)
```

`spatialize` *Spatialize occurrence records*

Description

Convert a data.frame (or data.table) of occurrence records into a **SpatVector** object.

Usage

```
spatialize(
  occ,
  long = "decimalLongitude",
  lat = "decimalLatitude",
  crs = "epsg:4326",
  force_numeric = TRUE
)
```

Arguments

`occ`	(data.frame or data.table) a data frame containing the occurrence records to be flagged. Must contain columns for longitude, and latitude.
`long`	(character) the name of the column in `occ` that contains the longitude values. Default is `"decimalLongitude"`.
`lat`	(character) the name of the column in `occ` that contains the latitude values. Default is `"decimalLatitude"`.
`crs`	(character) the coordinate reference system (see `?terra::crs`). Default is "epsg:4326".
`force_numeric`	(logical) whether to coerce the longitude and latitude columns to numeric if they are not already. Default is `TRUE`.

Value

A **SpatVector** object containing the spatialized occurrence records.

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Spatialize the occurrence records
pts <- spatialize(occurrences)
# Plot the resulting SpatVector
terra::plot(pts)
```

`standardize_countries` *Standardize country names*

Description

This function standardizes country names using both names and codes present in a specified column.

Usage

```
standardize_countries(
  occ,
  country_column = "country",
  max_distance = 0.1,
  user_dictionary = NULL,
  lookup_na_country = FALSE,
  long = "decimalLongitude",
  lat = "decimalLatitude",
  return_dictionary = TRUE
)
```

Arguments

`occ`	(data.frame) a dataset with occurrence records, preferably standardized using `format_columns()`.
`country_column`	(character) the column name containing the country information.
`max_distance`	(numeric) maximum allowed distance (as a fraction) when searching for suggestions for misspelled country names. Can be any value between 0 and 1. Higher values return more suggestions. See `agrep()` for details. Default is 0.1.
`user_dictionary`	(data.frame) optional data.frame with two columns: 'country_name' and 'country_suggested'. If provided, this dictionary will be combined with the package's default country dictionary (`RuHere::country_dictionary`). Default is NULL.
`lookup_na_country`	(logical) whether to extract the country from coordinates when the country column has missing values. If TRUE, longitude and latitude columns must be provided. Default is FALSE.

long (character) column name with longitude. Only applicable if lookup_na_country = TRUE. Default is "decimalLongitude".

lat (character) column name with latitude. Only applicable if lookup_na_country = TRUE. Default is "decimalLatitude".

return_dictionary (logical) whether to return the dictionary of countries that were (fuzzy) matched.

Details

Country names are first standardized by exact matching against a list of country names in several languages from rnaturalearthdata::map_units110. Any unmatched names are then processed using a fuzzy matching algorithm to find potential candidates for misspelled country names. If unmatched names remain and lookup_na_country = TRUE, the country is extracted from coordinates using a map retrieved from rnaturalearthdata::map_units110.

Value

A list with two elements:

data The original occ data.frame with an additional column (country_suggested) containing the suggested country names based on exact match, fuzzy match, and/or coordinates.

dictionary If return_dictionary = TRUE, a data.frame with the original country names and the suggested matches.

Examples

```
# Import and standardize GBIF
data("occ_gbif", package = "RuHere") #Import data example
gbif_standardized <- format_columns(occ_gbif, metadata = "gbif")
# Import and standardize SpeciesLink
data("occ_splink", package = "RuHere") #Import data example
splink_standardized <- format_columns(occ_splink, metadata = "specieslink")
# Import and standardize BIEN
data("occ_bien", package = "RuHere") #Import data example
bien_standardized <- format_columns(occ_bien, metadata = "bien")
# Import and standardize idigbio
data("occ_idig", package = "RuHere") #Import data example
idig_standardized <- format_columns(occ_idig, metadata = "idigbio")
# Merge all
all_occ <- bind_here(gbif_standardized, splink_standardized,
                     bien_standardized, idig_standardized)
# Standardize countries
occ_standardized <- standardize_countries(occ = all_occ)
```

`standardize_states` *Standardize state names*

Description

This function standardizes state names using both names and codes present in a specified column.

Usage

```
standardize_states(
  occ,
  state_column = "stateProvince",
  country_column = "country_suggested",
  max_distance = 0.1,
  lookup_na_state = FALSE,
  long = "decimalLongitude",
  lat = "decimalLatitude",
  return_dictionary = TRUE
)
```

Arguments

`occ`	(data.frame) a dataset with occurrence records, preferably standardized using `format_columns()`.
`state_column`	(character) the column name containing the state information.
`country_column`	(character) the column name containing the country information.
`max_distance`	(numeric) maximum allowed distance (as a fraction) when searching for suggestions for misspelled state names. Can be any value between 0 and 1. Higher values return more suggestions. See `agrep()` for details. Default is 0.1.
`lookup_na_state`	(logical) whether to extract the state from coordinates when the state column has missing values. If TRUE, longitude and latitude columns must be provided. Default is FALSE.
`long`	(character) column name with longitude. Only applicable if `lookup_na_state = TRUE`. Default is "decimalLongitude".
`lat`	(character) column name with latitude. Only applicable if `lookup_na_state = TRUE`. Default is "decimalLatitude".
`return_dictionary`	(logical) whether to return the dictionary of states that were (fuzzy) matched.

Details

States names are first standardized by exact matching against a list of state names in several languages from `rnaturalearthdata::states50`. Any unmatched names are then processed using a fuzzy matching algorithm to find potential candidates for misspelled state names. If unmatched names remain and `lookup_na_state = TRUE`, the state is extracted from coordinates using a map retrieved from `rnaturalearthdata::states50`.

Value

A list with two elements:

data	The original occ data.frame with an additional column (state_suggested) containing the suggested state names based on exact match, fuzzy match, and/or coordinates.
dictionary	If return_dictionary = TRUE, a data.frame with the original state names and the suggested matches.

Examples

```
# Import and standardize GBIF
data("occ_gbif", package = "RuHere") #Import data example
gbif_standardized <- format_columns(occ_gbif, metadata = "gbif")
# Import and standardize SpeciesLink
data("occ_splink", package = "RuHere") #Import data example
splink_standardized <- format_columns(occ_splink, metadata = "specieslink")
# Import and standardize BIEN
data("occ_bien", package = "RuHere") #Import data example
bien_standardized <- format_columns(occ_bien, metadata = "bien")
# Import and standardize idigbio
data("occ_idig", package = "RuHere") #Import data example
idig_standardized <- format_columns(occ_idig, metadata = "idigbio")
# Merge all
all_occ <- bind_here(gbif_standardized, splink_standardized,
                     bien_standardized, idig_standardized)
# Standardize countries
occ_standardized <- standardize_countries(occ = all_occ)
# Standardize states
occ_standardized2 <- standardize_states(occ = occ_standardized$occ)
```

---

states *Administrative Units (States, Provinces, and Regions)*

Description

A simplified PackedSpatVector containing state-level polygons (e.g., provinces, departments, regions) for countries worldwide. Names and parent countries (geonunit) were cleaned (lowercase, accents removed).

Usage

```
states
```

Format

A PackedSpatVector object with polygons of administrative divisions and one attribute:

name State/province/region name.

Details

The dataset was generated from `rnaturalearth::ne_states()`. The following processing steps were applied:

- kept only administrative types: `"Province"`, `"State"`, `"Department"`, `"Region"`, `"Federal District"`;
- selected only `"name"` and `"geonunit"` columns;
- both fields were cleaned via `tolower()` and `remove_accent()`;
- records where state name = country name were removed;
- geometries were simplified using `terra::simplifyGeom(tolerance = 0.05)`;
- wrapped with `terra::wrap()` for internal storage.

Source

Natural Earth data, via **rnaturalearth**.

Examples

```
data(states)
states <- terra::unwrap(states)
terra::plot(states)
```

---

states_dictionary *States dictionary for standardizing state and province names and codes*

Description

Provides lookup tables used to standardize subnational administrative units (states and provinces) in occurrence datasets.

Generated from `rnaturalearth::ne_states()`, it includes a wide range of name variants (in multiple languages, transliterations, and common abbreviations), as well as postal codes for each unit.

This dictionary allows consistent mapping of user-provided names such as `"são paulo"`, `"sao paulo"`, `"SP"`, `"illinois"`, `"ill."`, `"bayern"`, `"bavaria"` to a single standardized state or province name.

Usage

```
states_dictionary
```

Format

A named list with two data frames:

states_name A data frame with columns:

- **state_name** Character. Name variants of states or provinces from `ne_states()`, lowercased and accent-stripped.
- **state_suggested** Character. Standardized state/province name, also lowercased and accent-stripped.
- **country** Character. Country associated with the state/province, lowercased and accent-stripped.

states_code A data frame with columns:

- **state_code** Character. Postal codes from `ne_states()`, cleaned and converted to uppercase.
- **state_suggested** Character. Standardized state/province name corresponding to the code.
- **country** Character. Country associated with the code.

Details

The dictionary is constructed by:

- selecting administrative units of type `"State"` or `"Province"`;
- extracting multiple name fields, including alternative names and multilingual fields;
- normalizing names to lowercase and removing accents;
- normalizing codes to uppercase;
- removing duplicates and ambiguous entries;
- removing rows with missing names or codes.

Examples

```
data(states_dictionary)
head(states_dictionary$states_name)
head(states_dictionary$states_code)
```

---

states_from_coords *Extract state from coordinates*

Description

Extracts the state for each occurrence record based on coordinates.

Usage

```
states_from_coords(
  occ,
  long = "decimalLongitude",
  lat = "decimalLatitude",
  from = "all",
  state_column = "stateProvince",
  output_column = "state_xy",
  append_source = FALSE
)
```

Arguments

occ	(data.frame) a dataset with occurrence records, preferably standardized using `format_columns()`.
long	(character) column name with longitude. Default is 'decimalLongitude'.
lat	(character) column name with latitude. Default is 'decimalLatitude'.
from	(character) whether to extract the state for all records ('all') or only for records missing state information ('na_only'). If 'na_only', you must provide the name of the column with state information. Default is 'all'.
state_column	(character) the column name containing the state. Only applicable if `from = na_only`. Default is NULL.
output_column	(character) column name created in `occ` to store the states extracted. Default is 'state_xy'.
append_source	(logical) whether to create a new column in `occ` called 'state_source', which indicates whether the state was derived from coordinates. Default is FALSE.

Details

The states are extracted from coordinates using a map retrieved from `rnaturalearthdata::states50`.

Value

The original `occ` data.frame with an additional column containing the states extracted from coordinates.

Examples

```
# Import and standardize GBIF
data("occ_gbif", package = "RuHere") #Import data example
gbif_standardized <- format_columns(occ_gbif, metadata = "gbif")
gbif_states <- states_from_coords(occ = gbif_standardized)
```

summarize_flags *Summarize flags*

Description

This functions returns a dataframe and a bar plot summarizing the number of records flagged by each flagging function.

Usage

```
summarize_flags(
  occ = NULL,
  flagged_dir = NULL,
  output_format = ".gz",
  flags = "all",
  additional_flags = NULL,
  names_additional_flags = NULL,
  plot = TRUE,
  show_unflagged = TRUE,
  occ_unflagged = NULL,
  fill = "#0072B2",
  sort = TRUE,
  decreasing = TRUE,
  add_n = TRUE,
  size_n = 3.5,
  theme_plot = ggplot2::theme_minimal(),
  ...
)
```

Arguments

occ	(data.frame or data.table) a dataset containing occurrence records that has been processed by one or more flagging functions. See *Details* for available flag types.
flagged_dir	(character) optional path to a directory containing files with flagged records saved using the remove_flagged() function. Default is NULL.
output_format	(character) output format used to read the removed records. Options are ".csv" or ".gz". Only used when flagged_dir is not NULL. Default is ".gz".
flags	(character) the flags to be summarized. Use "all" to display all available flags. See *Details* for all options. Default is "all".
additional_flags	(character) an optional named character vector with the names of additional logical columns to be used as flags. Default is NULL.
names_additional_flags	(character) an optional different name to the flag provided in additional_flags to be shown in the bar plot. Only applicable if additional_flags is not NULL.

plot	(logical) whether to return a ggplot2 bar plot showing the number of flagged records. Default is TRUE.
show_unflagged	(logical) whether to include the number of unflagged records in the plot. Default is TRUE.
occ_unflagged	(data.frame or data.table) an optional dataset containing unflagged occurrence records. Only applicable if occ is NULL and show_unflagged is TRUE.
fill	(character) fill color for the bar plot. Default is "#0072B2".
sort	(logical) whether to sort bars according to the number of records. Default is TRUE.
decreasing	(logical) whether to sort bars in decreasing order (flags with more records appear at the top of the plot). Default is TRUE.
add_n	(logical) whether to display the number of flagged records on the bars. Default is TRUE.
size_n	(numeric) size of the text showing the number of records. Only used when add_n = TRUE. Default is 3.5.
theme_plot	(theme) a ggplot2 theme object. Default is ggplot2::theme_minimal().
...	additional arguments passed to ggplot2::theme().

Details

This function expects an occurrence dataset that has already been processed by one or more flagging routines from **RuHere** or related packages such as **CoordinateCleaner**. Any logical column in occ can be used as a flag.

The following built-in flag names are recognized: *From RuHere*: correct_country, correct_state, cultivated, florabr, faunabr, wcvp, iucn, bien, duplicated, thin_geo, thin_env, consensus

From CoordinateCleaner: .val, .equ, .zer, .cap, .cen, .sea, .urb, .otl, .gbf, .inst, .aohi

Users may also supply additional logical columns using additional_flags, optionally providing alternative display names (names_additional_flags) and colors (col_additional_flags).

Value

If plot = TRUE, a list with two elements:

df_summary A data frame summarizing the number of records per flag.

plot_summary A ggplot2 object showing the summary as a bar plot.

If plot = FALSE, only the summary data frame is returned.

Examples

```
# Load example data
data("occ_flagged", package = "RuHere")
# Summarize flags
sum_flags <- summarize_flags(occ = occ_flagged)
# Plot
sum_flags$plot_summary
```

thin_env *Flag records that are close to each other in the enviromnetal space*

Description

Flags occurrence records for thinning by keeping only one record per species within the same environmental block/bin.

Usage

```
thin_env(
  occ,
  species = "species",
  long = "decimalLongitude",
  lat = "decimalLatitude",
  env_layers,
  n_bins = 5,
  prioritary_column = NULL,
  decreasing = TRUE,
  flag_for_NA = FALSE
)
```

Arguments

occ	(data.frame or data.table) a data frame containing the occurrence records. Must contain columns for species, longitude, and latitude.
species	(character) the name of the column in occ that contains the species scientific names. Default is "species".
long	(character) the name of the column in occ that contains the longitude values. Default is "decimalLongitude".
lat	(character) the name of the column in occ that contains the latitude values. Default is "decimalLatitude".
env_layers	(SpatRaster) object containing environmental variables.
n_bins	(numeric) number of bins into which each environmental variable will be divided.
prioritary_column	(character) name of a numeric columns in occto define retention priority (e.g., quality score, year). See details.
decreasing	(logical) whether to sort records in decreasing order using the prioritary_column (e.g., from most recent to oldest when the variable is "year"). Only applicable when prioritary_column is not NULL. Default is TRUE.
flag_for_NA	(logical) whether to treat records falling in NA cells of env_layers as passing (TRUE) or failing (FALSE) this test. Default is FALSE.

Details

This function used `get_env_bins()` to create a multidimensional grid in environmental space by splitting each environmental variable into `n_bins` equally sized intervals. Records falling into the same environmental bin are considered redundant; only one is kept (based on retention priority when provided), and the remaining records are flagged.

Value

The original `occ` data frame with two additional columns:

- `thin_env_flag`: logical indicating whether each record was retained (`TRUE`) or flagged as redundant (`FALSE`). As with all other flagging functions in RuHere, `TRUE` indicates that the record passed this test and is eligible for retention; `FALSE` indicates it failed and is flagged as potentially problematic, and can be removed using `remove_flagged()`.
- `bin`: environmental bin ID assigned to each record. Each component of the ID corresponds to the bin of one environmental variable.

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Get only occurrences from Araucaria
occ <- occurrences[occurrences$species == "Araucaria angustifolia", ]
# Load example of raster variables
data("worldclim", package = "RuHere")
# Unwrap Packed raster
r <- terra::unwrap(worldclim)
# Flag records that are close to each other in the enviromnetal space
occ_env_thin <- thin_env(occ = occ, env_layers = r)
# Number of flagged (redundant) records
sum(!occ_env_thin$thin_env_flag) #Number of flagged records
```

---

thin_geo *Flag records that are close to each other in the geographic space*

Description

Marks occurrence records for thinning by keeping only one record per species within a radius of 'd' kilometers.

Usage

```
thin_geo(
  occ,
  species = "species",
  long = "decimalLongitude",
  lat = "decimalLatitude",
  d,
```

```
  prioritary_column = NULL,
  decreasing = TRUE,
  remove_invalid = TRUE,
  optimize_memory = FALSE,
  verbose = TRUE
)
```

Arguments

occ	(data.frame or data.table) a data frame containing the occurrence records to be flagged. Must contain columns for species, longitude, and latitude.
species	(character) the name of the column in occ that contains the species scientific names. Default is "species".
long	(character) the name of the column in occ that contains the longitude values. Default is "decimalLongitude".
lat	(character) the name of the column in occ that contains the latitude values. Default is "decimalLatitude".
d	(numeric) thinning distance in **kilometers** (e.g., 10 for 10km).
prioritary_column	(character) name of a numeric columns in occto define retention priority (e.g., quality score, year). See details.
decreasing	(logical) whether to sort records in decreasing order using the prioritary_column (e.g., from most recent to oldest when the variable is "year"). Only applicable when prioritary_column is not NULL. Default is TRUE.
remove_invalid	(logical) whether to remove records with missing or impossible coordinates. Default is TRUE.
optimize_memory	(logical) whether to compute the distance matrix using a C++ implementation that reduces memory usage at the cost of increased computation time. Recommended for large datasets (> 10,000 records). Default is FALSE.
verbose	(logical) whether to display messages during function execution. Set to TRUE to enable display, or FALSE to run silently. Default is TRUE.

Details

This function is similar to the thin() function from the **spThin** package, but with an important difference: it allows specifying a priority order for retaining records.

When a thinning distance is provided (e.g., 10 km), the function identifies clusters of records within this distance. Within each cluster, it keeps the record with the highest priority according to the column defined in prioritary_column (for example, keeping the most recent record if prioritary_column = "year"), and flags the remaining nearby records for removal.

If prioritary_column is NULL, the priority follows the original order of rows in the input occ data.frame.

Value

The original occ data frame augmented with a new logical column named thin_geo_flag. Records that are retained after thinning receive TRUE, while records identified as too close to a higher-priority record receive FALSE. As with all other flagging functions in RuHere, TRUE indicates that the record passed this test and is eligible for retention; FALSE indicates it failed and is flagged as potentially problematic, and can be removed using remove_flagged().

Examples

```
# Load example data
data("occurrences", package = "RuHere")
# Subset occurrences for Araucaria angustifolia
occ <- occurrences[occurrences$species == "Araucaria angustifolia", ]
# Thin records using a 10 km distance threshold
occ_thin <- thin_geo(occ = occ, d = 10)
sum(!occ_thin$thin_geo_flag)  # Number of records flagged for removal
# Prioritizing more recent records within each cluster
occ_thin_recent <- thin_geo(occ = occ, d = 10, prioritary_column = "year")
sum(!occ_thin_recent$thin_geo_flag)  # Number of records flagged for removal
```

---

wcvp_here *Download distribution data from the World Checklist of Vascular Plants (WCVP)*

Description

This function downloads the World Checklist of Vascular Plants database, which is required for filtering occurrence records using specialists' range information via the flag_wcvp() function.

Usage

```
wcvp_here(
  data_dir,
  overwrite = TRUE,
  remove_files = TRUE,
  timeout = 300,
  verbose = TRUE
)
```

Arguments

data_dir	(character) a directory to save the data downloaded from WCVP.
overwrite	(logical) If TRUE, data is overwritten. Default is TRUE.
remove_files	(logical) whether to remove the downloaded files used in building the final dataset. Default is TRUE.

timeout	(numeric) maximum time (in seconds) allowed for downloading. Default is 300. Slower internet connections may require higher values.
verbose	(logical) whether to display messages during function execution. Set to TRUE to enable display, or FALSE to run silently. Default is TRUE.

Value

A message indicating that the data were successfully saved in the directory specified by `data_dir`.

Examples

```
# Define a directory to save the data
data_dir <- tempdir() # Here, a temporary directory

# Download the WCVP database
wcvp_here(data_dir = data_dir)
```

---

world *World Countries*

Description

A `"PackedSpatVector"` containing country polygons from **Natural Earth**, processed and cleaned for use within the package. Country names were converted to lowercase and had accents removed.

Usage

```
world
```

Format

A `PackedSpatVector` object with country polygons and one attribute:

name Country name.

Details

The dataset is sourced from `rnaturalearthdata::map_units110`, then:

- converted to a `SpatVector` using **terra**,
- attribute `"name"` cleaned (`tolower()`, `remove_accent()`),
- wrapped using `terra::wrap()` for robust internal storage.

Source

Natural Earth data, via **rnaturalearthdata**.

Examples

```
data(world)
world <- terra::unwrap(world)
terra::plot(world)
```

`worldclim` *Bioclimatic Variables from WorldClim (bio_1, bio_7, bio_12)*

Description

A `PackedSpatRaster` containing three bioclimatic variables from the WorldClim, cropped to a region of interest South America.

Usage

```
worldclim
```

Format

A `SpatRaster` with 3 layers and the following characteristics:

Dimensions 151 rows × 183 columns

Resolution 0.08333333° × 0.08333333°

Extent xmin = -57.08333, xmax = -41.83333, ymin = -32.08333, ymax = -19.5

CRS WGS84 (EPSG:4326)

Layers **bio_1** Mean Annual Temperature (°C × 10)
bio_7 Temperature Annual Range (°C × 10)
bio_12 Annual Precipitation (mm)

Details

This raster corresponds to three standard bioclimatic variables from the **WorldClim 2.1** dataset.

Source

https://www.worldclim.org/

Examples

```
data(worldclim)
bioclim <- terra::unwrap(worldclim)
terra::plot(bioclim)
```

Index

∗ **datasets**
atlantic_amphibians, 4
country_dictionary, 13
cultivated, 17
fake_data, 18
flag_colors, 26
flag_names, 45
occ_bien, 74
occ_flagged, 75
occ_gbif, 76
occ_idig, 77
occ_splink, 77
occurrences, 78
prepared_metadata, 82
puma_atlanticr, 83
states, 103
states_dictionary, 104
world, 113
worldclim, 114

atlantic_amphibians, 4
available_datasets, 4

bien_here, 6
bind_here, 7

check_countries, 8
check_states, 9
cite_datapaper, 10
count_flags, 11
country_dictionary, 13
country_from_coords, 14
create_metadata, 15
cultivated, 17

fake_data, 18
faunabr_here, 19
fix_countries, 20
fix_states, 22
flag_bien, 24
flag_colors, 26
flag_consensus, 26
flag_cultivated, 28
flag_duplicates, 29
flag_env_moran, 30
flag_faunabr, 34
flag_florabr, 36
flag_fossil, 38
flag_geo_moran, 39
flag_inaturalist, 42
flag_iucn, 43
flag_names, 45
flag_wcvp, 46
flag_year, 47
florabr_here, 48
format_columns, 49

get_bien, 51
get_datapaper, 54
get_env_bins, 55
get_idigbio, 57
get_specieslink, 58
ggmap_here, 61
ggrid_here, 64

import_gbif, 65
inventory_completeness, 67
iucn_here, 69

map_here, 71
moranfast, 73

occ_bien, 74
occ_flagged, 75
occ_gbif, 76
occ_idig, 77
occ_splink, 77
occurrences, 78

plot_env_bins, 79
prepare_gbif_download, 80

prepared_metadata, 82
puma_atlanticr, 83

relocate_after, 83
relocate_before *(*relocate_after*)*, 83
remove_accent, 84
remove_flagged, 85
remove_invalid_coordinates, 86
request_gbif, 88
request_gbif_specieslist, 89
richness_here, 91

set_gbif_credentials, 93
set_iucn_credentials, 94
set_specieslink_credentials, 95
spatial_kde, 97
spatialize, 99
standardize_countries, 100
standardize_states, 102
states, 103
states_dictionary, 104
states_from_coords, 105
summarize_flags, 107

thin_env, 109
thin_geo, 110

wcvp_here, 112
world, 113
worldclim, 114