Package 'EcoCleanR'

February 13, 2026

Title Enhancing Data Quality of Biogeographic Ranges with Application for Marine Invertebrates

Version 1.0.3

Language en-US

Description
Provides step-by-step automation for integrating biodiversity data from multiple online aggregators, merging and cleaning datasets while addressing challenges such as taxonomic inconsistencies, georeferencing issues, and spatial or environmental outliers. Includes functions to extract environmental data and to define the biogeographic ranges in which species are most likely to occur. For methodological details see the associated publication.<doi:10.1002/ecog.08203>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Suggests knitr, rgbif, robis, ridigbio, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports dplyr, geodata, geosphere, ggplot2, mregions2, patchwork, rlang, sdmpredictors, sf, worrms, terra, tidyr

Depends R (>= 3.5)

LazyData true

VignetteBuilder knitr

Config/Needs/website rmarkdown

URL `https://github.com/sonipri/EcoCleanR`, `https://sonipri.github.io/EcoCleanR/`, `https://doi.org/10.1002/ecog.08203`

BugReports `https://github.com/sonipri/EcoCleanR/issues`

NeedsCompilation no

Author Priyanka Soni [aut, cre] (ORCID: <`https://orcid.org/0000-0001-8358-1645`>),
Austin Hendy [ctb] (Manuscript mentoring and advising),
David Bottjer [ctb] (Manuscript mentoring and advising),
Vijay Barve [ctb] (Code development feedback and advising)

Maintainer Priyanka Soni <sonip@usc.edu>

Repository CRAN

Date/Publication 2026-02-13 09:50:21 UTC

Contents

`decimal_places` *Get Decimal Places of Coordinate Values*

Description

Get Decimal Places of Coordinate Values

Usage

```
decimal_places(coord)
```

Arguments

`coord`	A coordinate value in the numeric format of decimal degree

Value

a numerical value which represent the number of decimal places for the coordiante

Examples

```
decimal_places(12.7000000)
decimal_places(45.67788)
```

`distance_calc` *Calculate geographic distance and mahalanobis distance to estimate outlier probability of a data point*

Description

Calculate geographic distance and mahalanobis distance to estimate outlier probability of a data point

Usage

```
distance_calc(data, latitude, longitude, env_layers, itr = 15, k = 3)
```

Arguments

`data`	data table with spatial and environmental variables
`latitude`	nested input from ec_flag_outlier
`longitude`	nested input from ec_flag_outlier
`env_layers`	header names of env variables. env_layers <- c("Temperature", "pH")
`itr`	iteration to run the clustering 100 or 1000 times
`k`	number of cluster to choose in each iteration

Value

A list of results that shows result of calculated distance for each iteration

Examples

```
data <- data.frame(
  scientificName = "Mexacanthina lugubris",
  decimalLongitude = c(-117, -117.8, -116.9),
  decimalLatitude = c(32.9, 33.5, 31.9),
  temperature_mean = c(12, 13, 14),
  temperature_min = c(9, 6, 10),
  temperature_max = c(14, 16, 18)
)

env_layers <- c("temperature_mean", "temperature_min", " temperature_max")

result_list <- distance_calc(data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude",
  env_layers,
  itr = 100,
  k = 3
)
```

`ecodata` *dataset1: Documentation of data file - ecodata.rda*

Description

This data file is consider as raw data file after merging and removing duplicate records of all data sources. e.g. this file is an output of occurrence records of mollusc species "Mexacanthina lugubris" with all modern records extracted from GBIF, OBIS, IDIGBIO and InvertEBase

Usage

```
ecodata
```

Format

A data frame with 1115 rows and 19 variables:

X index

basisOfRecord Type of record (e.g., preserved specimen, fossil)

occurrenceStatus Presence or absence of the organism

institutionCode Code of the institution that holds the record

verbatimEventDate Original recorded date of the event

scientificName Full scientific name of the organism

individualCount Number of individuals observed

organismQuantity Reported quantity of the organism

abundance Calculated or standardized abundance value

decimalLatitude Latitude in decimal degrees

decimalLongitude Longitude in decimal degrees

coordinateUncertaintyInMeters Uncertainty in coordinates (meters)

locality Named place where the occurrence was recorded

verbatimLocality Original text for locality description

municipality Municipality or town of the occurrence

county County where the record was observed

stateProvince State or province name

country Country name

cleaned_catalog Standardized catalog number for de-duplication

Source

- used rgbif for GBIF, ridigbio for iDigBio, robis for OBIS and rsymbiota for InvertEBase

---

`ecodata_cleaned` *dataset4: Documentation of data file - ecodata_cleaned.rda*

Description

This data shows the final cleaned occurrence records

Usage

```
ecodata_cleaned
```

Format

A data frame with 698 rows and 35 variables:

X Index

basisOfRecord Type of occurrence record (e.g., preserved specimen, fossil)

occurrenceStatus Indicates presence or absence of the species

institutionCode Code of the institution that provided the record

verbatimEventDate Original text for the event or collection date

scientificName Scientific name of the organism

individualCount Number of individuals recorded

organismQuantity Reported quantity (unit may vary)

abundance Standardized or calculated abundance value

decimalLatitude Latitude in decimal degrees

decimalLongitude Longitude in decimal degrees

coordinateUncertaintyInMeters Spatial uncertainty of coordinates in meters

locality Named location where the record was collected

verbatimLocality Original locality text as provided by the source

municipality Municipality or town of occurrence

county County of occurrence

stateProvince State or province of occurrence

country Country of occurrence

cleaned_catalog Standardized catalog number used for de-duplication

lat_precision Number of decimal places in the latitude coordinate

lon_precision Number of decimal places in the longitude coordinate

flag_cordinate_precision Flag for low coordinate precision

flag_cc_val Flag for invalid or impossible coordinates

flag_cc_equal Flag for identical latitude and longitude (likely erroneous)

flag_cc_zero Flag for coordinates at (0,0)

flag_cc_cent Flag for coordinates placed at a country or region centroid

flag_cc_gbif Flag for coordinates matching GBIF headquarters (artifact)

flag_cc_inst Flag for coordinates matching institution location

flag_non_region Flag for coordinates outside the study region

outliers Flag for outliers based on clustering of spatial and environmental variables

BO_sstmean Mean sea surface temperature from Bio-ORACLE

BO_sstmax Maximum sea surface temperature from Bio-ORACLE

BO_sstmin Minimum sea surface temperature from Bio-ORACLE

BO_chloro Chlorophyll concentration from Bio-ORACLE

BO_dissox Dissolved oxygen level from Bio-ORACLE

Source

Generated after filtering outlier data points

`ecodata_corrected` *dataset2: Documentation of data file - ecodata_corrected.rda*

Description

This data file created by using GEOLocate tool and we only kept 4 columns. These georeference information will be merge back with the main data file ecodata

Usage

```
ecodata_corrected
```

Format

A data frame with 433 rows and 4 variables:

cleaned_catalog catalog number

corrected_latitude latitude values assigned by GEOLocate

corrected_longitude longitude values assigned by GEOLocate

corrected_uncertainty uncertainty values assigned by GEOLocate

Source

- this file was created manually after extracting the csv file from GEOLocate online software to assign coordiante and uncertainty values for the records has locality information

`ecodata_with_outliers` *dataset3: Documentation of data file - ecodata_with_outliers.rda*

Description

This data file created after running ec_flag_outlier function. It has records with outlier probability

Usage

```
ecodata_with_outliers
```

Format

A data frame with 713 rows and 35 variables:

X index
basisOfRecord Type of occurrence record (e.g., preserved specimen, fossil)
occurrenceStatus Indicates presence or absence of the species
institutionCode Code of the institution that provided the record
verbatimEventDate Original text for the event or collection date
scientificName Scientific name of the organism
individualCount Number of individuals recorded
organismQuantity Reported quantity (unit may vary)
abundance Standardized or calculated abundance value
decimalLatitude Latitude in decimal degrees
decimalLongitude Longitude in decimal degrees
coordinateUncertaintyInMeters Spatial uncertainty of coordinates in meters
locality Named location where the record was collected
verbatimLocality Original locality text as provided by the source
municipality Municipality or town of occurrence
county County of occurrence
stateProvince State or province of occurrence
country Country of occurrence
cleaned_catalog Standardized catalog number used for de-duplication
lat_precision Number of decimal places in the latitude coordinate
lon_precision Number of decimal places in the longitude coordinate
flag_cordinate_precision Flag for low coordinate precision
flag_cc_val Flag for invalid or impossible coordinates
flag_cc_equal Flag for identical latitude and longitude (likely erroneous)
flag_cc_zero Flag for coordinates at (0,0)
flag_cc_cent Flag for coordinates placed at a country or region centroid
flag_cc_gbif Flag for coordinates matching GBIF headquarters (artifact)
flag_cc_inst Flag for coordinates matching institution location
flag_non_region Flag for coordinates outside the study region
outliers Flag for outliers based clustering of spatial and env variables
BO_sstmean Mean sea surface temperature from Bio-ORACLE
BO_sstmax Maximum sea surface temperature from Bio-ORACLE
BO_sstmin Minimum sea surface temperature from Bio-ORACLE
BO_chloro Chlorophyll concentration from Bio-ORACLE
BO_dissox Dissolved oxygen level from Bio-ORACLE

Source

- this file was created manually after extracting the csv file from GEOLocate online software to assign coordiante and uncertainty values for the records has locality information

---

`ecodata_x` *dataset5: Documentation of data file - ecodata_x.rda*

Description

This data was created to get unique combination of coordinate values to extract env variables from bio-oracle and merge back in main data table - ecodata

Usage

```
ecodata_x
```

Format

A data frame with 705 rows and 6 variables:

species species name

decimalLatitude Latitude in decimal degrees

decimalLongitude Longitude in decimal degrees

temperature_mean_BO Mean sea surface temperature from Bio-ORACLE

temperature_max_BO Maximum sea surface temperature from Bio-ORACLE

temperature_min_BO Minimum sea surface temperature from Bio-ORACLE

Source

- this file has unique coordinate information with unique values of enviornemnt variables

---

`ec_db_merge` *Merge the Data sets Extracted from Various datasources.*

Description

condition to run this function: all the data frames should have same fields follwing DwC standards: e.g. attribute_list <- c("source","catalogNumber", "basisOfRecord", "occurrenceStatus", "institutionCode", "verbatimEventDate", "scientificName", "individualCount", "organismQuantity", "abundance", "decimalLatitude", "decimalLongitude", "coordinateUncertaintyInMeters", "locality", "verbatimLocality", "municipality", "county", "stateProvince", "country", "countryCode") Assign manually the source name in "source" field. example - gbif, obis, invertEBase etc Assign values of individual count or organism count into abundance. Most online sources has one of them updated with specimen count. this function depends on successful download of data files, it also allow to input csv files from local system

Usage

```
ec_db_merge(
  db_list,
  datatype = "modern",
  occurrenceStatus = "occurrenceStatus",
  basisOfRecord = "basisOfRecord"
)
```

Arguments

`db_list`	list of data frames which we want to merge. e.g. GBIF, iDigbio, InvertEBase and any local file.
`datatype`	default "modern". datatype accept text input as "modern" or "fossil"
`occurrenceStatus`	default name for occurrenceStatus column is occurrenceStatus but a different name can be inserted if required.
`basisOfRecord`	default name for basis of record column is basis of record but a different name can be inserted if required.

Value

A data frame of occurrence records filtered to include only those classified as "modern" or "fossil".

Examples

```
db1 <- data.frame(
  species = "A",
  decimalLongitude = c(-120, -117, NA, NA),
  decimalLatitude = c(20, 34, NA, NA),
  catalogNumber = c("12345", "89888", "LACM8898", "SDNHM6767"),
  occurrenceStatus = c("present", "", "ABSENT", "Present"),
  basisOfRecord = c("preserved_specimen", "", "fossilspecimen", "material_sample"),
  source = "db1",
  abundance = c(1, NA, 8, 23)
)

db2 <- data.frame(
  species = "A",
  decimalLongitude = c(-120.2, -117.1, NA, NA),
  decimalLatitude = c(20.2, 34.1, NA, NA),
  catalogNumber = c("123452", "898828", "LACM82898", "SDNHM62767"),
  occurrenceStatus = c("present", "", "ABSENT", "Present"),
  basisOfRecord = c("preserved_specimen", "", "fossilspecimen", "material_sample"),
  source = "db2",
  abundance = c(1, 2, 3, 19)
)
db_list <- list(db1, db2)
merge_modern_data <- ec_db_merge(db_list = db_list, "modern")
```

`ec_extract_env_layers` *Extract the Environmental data*

Description

Extract the Environmental data

Usage

```
ec_extract_env_layers(
  data,
  env_layers = env_layers,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)
```

Arguments

`data`	data table which has coordinate information
`env_layers`	make a list of enviornmental layers which need to be extracted, example :BO_sstmean, BO_sstmax, BO_sstmin, BO_chomean, BO_phosphate or marspec layer, must check list_layer to know exact name of the layer code.
`latitude`	default assigned as "decimalLatitude"
`longitude`	default assigned as "decimalLongitude"

Value

A data table which has unique coordinates and env predictors

Examples

```
## Not run:
env_layers <- c("BO_sstmean", "BO_chlomean", "BO_dissox", "BO_salinity")
data <- data.frame(
  scientificName = "Mexacanthina lugubris",
  decimalLongitude = c(-117, -117.8, -116.9),
  decimalLatitude = c(32.9, 33.5, 31.9)
)

data_x <- ec_extract_env_layers(data,
  env_layers = env_layers,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)

## End(Not run)
```

`ec_filter_by_uncertainty` *Flag the Occurrences those has Extreme Uncertainty Error Radius*

Description

Flag the Occurrences those has Extreme Uncertainty Error Radius

Usage

```
ec_filter_by_uncertainty(
  data,
  uncertainty_col = "coordinateUncertaintyInMeters",
  percentile = 0.96,
  ask = TRUE,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)
```

Arguments

`data`	data table which need to be cleaned with unwanted uncertainty values - extreme values
`uncertainty_col`	coordinateUncertaintyInMeters column
`percentile`	to derive threshold, e.g. extreme 5% uncertainty data points to be removed. give percentile value as 0.95
`ask`	this allow user to decide if the uncertainty threshold value is okay or too high/low
`latitude`	default set on decimalLatitude, this column is use to filter records those does not have georeferences.
`longitude`	default set on decimalLongitude.

Value

A data frame as result of removing extreme uncertain occurrences

Examples

```
data <- data.frame(
  species = "A",
  decimalLongitude = c(-120, -117, NA, NA),
  decimalLatitude = c(20, 34, NA, NA),
  cleaned_catalog = c("12345", "89888", "LACM8898", "SDNHM6767"),
  locality = c(NA, NA, "Los Angeles, CA", "San Pedro, CA"),
  coordinateUncertaintyInMeters = c(1000, 2000, 9999900, NA)
)
data <- ec_filter_by_uncertainty(
  data,
  uncertainty_col = "coordinateUncertaintyInMeters",
  latitude = "decimalLatitude",
  longitude = "decimalLongitude",
  percentile = 0.96,
  ask = TRUE
)
```

`ec_flag_non_east_atlantic` *Flag the occurrences those are not in east Atlantic and are inland*

Description

Flag the occurrences those are not in east Atlantic and are inland

Usage

```
ec_flag_non_east_atlantic(
  ocean_names,
  buffer_distance = 50000,
  data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)
```

Arguments

`ocean_names`	Insert the name of oceans: "South Pacific Ocean", "North Pacific Ocean", North Atlantic Ocean", "South Atlantic Ocean"
`buffer_distance`	Its a certain buffer distance to consider if a data point is inland. Beyond this distance data points consider as bad data points. e.g. buffer_distance <- 25000
`data`	Data table which has latitude and longitude information
`latitude`	default set to "decimalLatitude"
`longitude`	default set to "decimalLongitude"

Value

A new column with flagged values, 1 means bad records 0 means good record. Column name: flag_non_region

Examples

```
## Not run:
ocean_names <- c("North Atlantic Ocean", "South Atlantic Ocean")
buffer_distance <- 25000
data <- data.frame(
  species = "A",
  decimalLongitude = c(-120, -78, -110, -60, -75, -130, -10, 5),
  decimalLatitude = c(20, 34, 30, 10, 40, 25, 15, 35)
)
data$flag_non_region <- ec_flag_non_east_atlantic(
  ocean_names,
  buffer_distance,
  data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)

## End(Not run)
```

`ec_flag_non_east_pacific` *Flag occurrences those are not in east Pacific and are inland*

Description

Flag occurrences those are not in east Pacific and are inland

Usage

```
ec_flag_non_east_pacific(
  ocean_names,
  buffer_distance = 50000,
  data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)
```

Arguments

`ocean_names`	Insert the name of oceans: "South Pacific Ocean", "North Pacific Ocean", North Atlantic Ocean", "South Atlantic Ocean"
`buffer_distance`	Its a certain buffer distance to consider if a data point is inland. Beyond this distance data points consider as bad data points. e.g. buffer_distance <- 25000
`data`	Data table which has latitude and longitude information
`latitude`	default set to "decimalLatitude"
`longitude`	default set to "decimalLongitude"

Value

A new column with flagged values, 1 means bad records 0 means good record. Column name: flag_non_region

Examples

```
## Not run:
ocean_names <- c("North Pacific Ocean", "South Pacific Ocean")
buffer_distance <- 25000
data <- data.frame(
  species = "A",
  decimalLongitude = c(-120, -78, -110),
  decimalLatitude = c(20, 34, 30)
)
data$flag_non_region <- ec_flag_non_east_pacific(
  ocean_names,
  buffer_distance,
  data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)

## End(Not run)
```

---

ec_flag_non_region *Flag Occurrences those are in wrong ocean basins and are inland*

Description

Flag Occurrences those are in wrong ocean basins and are inland

Usage

```
ec_flag_non_region(
  direction,
  ocean,
  buffer = 50000,
  data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)
```

Arguments

`direction`	values as "east" or "west". These values help to filter the shape files for east or west of select ocean (e.g. pacific) for both north and south hemisphere.
`ocean`	values such as "pacific" or "atlantic"

`buffer`	Its a certain buffer distance to consider if a data point is inland. Beyond this distance data points consider as bad data points. e.g. buffer <- 25000
`data`	Data table which has latitude and longitude information
`latitude`	default set to "decimalLatitude"
`longitude`	default set to "decimalLongitude"

Value

A new column with flagged values, 1 means bad records 0 means good record. Column name: flag_non_region

Examples

```
## Not run:
direction <- "east"
buffer <- 25000
ocean <- "pacific"
data <- data.frame(
  species = "A",
  decimalLongitude = c(-120, -78, -110, -60, -75, -130, -10, 5),
  decimalLatitude = c(20, 34, 30, 10, 40, 25, 15, 35)
)
data$flag_non_region <- ec_flag_non_region(
  direction,
  ocean,
  buffer = 50000,
  data
)

## End(Not run)
```

`ec_flag_non_west_atlantic`

Flag Occurrences those are not in west Atlantic and are inland

---

Description

Flag Occurrences those are not in west Atlantic and are inland

Usage

```
ec_flag_non_west_atlantic(
  ocean_names,
  buffer_distance = 50000,
  data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)
```

Arguments

`ocean_names`	Insert the name of oceans: "South Pacific Ocean", "North Pacific Ocean", North Atlantic Ocean", "South Atlantic Ocean"
`buffer_distance`	Its a certain buffer distance to consider if a data point is inland. Beyond this distance data points consider as bad data points. e.g. buffer_distance <- 25000
`data`	Data table which has latitude and longitude information
`latitude`	default set to "decimalLatitude"
`longitude`	default set to "decimalLongitude"

Value

A new column with flagged values, 1 means bad records 0 means good record. Column name: flag_non_region

Examples

```
## Not run:
ocean_names <- c("North Atlantic Ocean", "South Atlantic Ocean")
buffer_distance <- 25000
data <- data.frame(
  species = "A",
  decimalLongitude = c(-120, -78, -110, -60, -75, -130, -10, 5),
  decimalLatitude = c(20, 34, 30, 10, 40, 25, 15, 35)
)
data$flag_non_region <- ec_flag_non_west_atlantic(
  ocean_names,
  buffer_distance,
  data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)

## End(Not run)
```

`ec_flag_non_west_pacific` *Flag occurrences those are not in east Pacific and are inland*

Description

Flag occurrences those are not in east Pacific and are inland

Usage

```
ec_flag_non_west_pacific(
  ocean_names,
  buffer_distance = 50000,
  data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)
```

Arguments

`ocean_names`	Insert the name of oceans: "South Pacific Ocean", "North Pacific Ocean", North Atlantic Ocean", "South Atlantic Ocean"
`buffer_distance`	Its a certain buffer distance to consider if a data point is inland. Beyond this distance data points consider as bad data points. e.g. buffer_distance <- 25000
`data`	Data table which has latitude and longitude information
`latitude`	default set to "decimalLatitude"
`longitude`	default set to "decimalLongitude"

Value

A new column with flagged values, 1 means bad records 0 means good record. Column name: flag_non_region

Examples

```
## Not run:
ocean_names <- c("North Pacific Ocean", "South Pacific Ocean")
buffer_distance <- 25000
data <- data.frame(
  species = "A",
  decimalLongitude = c(-120, -78, -110),
  decimalLatitude = c(20, 34, 30)
)
data$flag_non_region <- ec_flag_non_west_pacific(
  ocean_names,
  buffer_distance,
  data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)

## End(Not run)
```

ec_flag_outlier *Flag Outlier Occurrences - using Spatial and Non-spatial Attributes*

Description

Flag Outlier Occurrences - using Spatial and Non-spatial Attributes

Usage

```
ec_flag_outlier(
  data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude",
  env_layers,
  itr = 50,
  k = 3,
  geo_quantile = 0.99,
  maha_quantile = 0.99
)
```

Arguments

`data`	data table with spatial and environmental variables
`latitude`	default set to "deciamlLatitude"
`longitude`	default set to "decimalLongitude"
`env_layers`	header names of env variables. env_layers <- c("Temperature", "pH")
`itr`	iteration to run the clustering 100 or 1000 times
`k`	number of cluster to choose in each iteration
`geo_quantile`	value with geo_quantile percentile would consider has threshold for geo_distance to derive the outlier. e.g. default 0.99
`maha_quantile`	value with maha_quantile percentile would consider has threshold for maha_distance to derive the outlier. e.g. default 0.99

Value

A column call flag_outlier which has outlier probability from 0 to 1. 1 is more towards outlier, 0 more towards good data points.

Examples

```
data <- data.frame(
  scientificName = "Mexacanthina lugubris",
  decimalLongitude = c(-117, -117.8, -116.9),
  decimalLatitude = c(32.9, 33.5, 31.9),
  BO_sstmean = c(12, 13, 14),
  BO_sstmin = c(9, 6, 10),
  BO_sstmax = c(14, 16, 18)
)

env_layers <- c("BO_sstmean", "BO_sstmin", "BO_sstmax")
res <- ec_flag_outlier(data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude",
  env_layers,
  itr = 100,
  k = 3,
  geo_quantile = 0.99,
  maha_quantile = 0.99
)
data$outlier <- res$outlier
iteration_list <- res$result$list
```

`ec_flag_precision` *Flag occurrences those has bad precision*

Description

Flag occurrences those has bad precision

Usage

```
ec_flag_precision(
  data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude",
  threshold = 2
)
```

Arguments

`data`	dataframe
`latitude`	decimalLatitude, this a field in the data file. We prefer to use decimalLatitude as accepeted name based on TDWG standards
`longitude`	decimalLongitude, this a field in the data file. We prefer to use decimalLongitude as accepeted name based on TDWG standards
`threshold`	set on 2

Value

A column which has flagged records represents bad records based on low precision as well as rounding

Examples

```
data <- data.frame(
  species = "A",
  decimalLongitude = c(-120.67, -78, -110, -60, -75.5, -130.78, -10.2, 5.4),
  decimalLatitude = c(20.7, 34.6, 30.0, 10.5, 40.4, 25.66, 15.0, 35.9)
)

data$flag_cordinate_precision <- ec_flag_precision(
  data,
  latitude = "decimalLongitude",
  longitude = "decimalLatitude",
  threshold = 2
)
```

`ec_flag_with_locality` *Filter records to georeference using GEOLocate*

Description

Filter records to georeference using GEOLocate

Usage

```
ec_flag_with_locality(
  data,
  uncertainty = "coordinateUncertaintyInMeters",
  locality = "locality",
  verbatimLocality = "verbatimLocality"
)
```

Arguments

`data`	data table with occurrence information
`uncertainty`	Mendatory to have coordinateUncertaintyInMeters column in the data table
`locality`	Mandatory to have locality column in the data table.
`verbatimLocality`	Mandatory to have verbatimLocality in the data table.

Details

Records those does not have coordinates assigned but has locality and varbatim locality information to assign coordinates by using external tools such as GEOLocate

Value

A column with flagged records as 1, which means these records has potential to be georeferenced.

Examples

```
data <- data.frame(
  coordinateUncertaintyInMeters = c(NA, "N/A", 50, "30", NA, "N/A", NA),
  locality = c("Santa Cruz", NA, "Los Angeles", "N/A", "", "San Diego", NA),
  verbatimLocality = c(NA, "CA coast", "", "N/A", "Long Beach", NA, "")
)
data$flag_check_geolocate <- ec_flag_with_locality(
data, uncertainty = "coordinateUncertaintyInMeters",
locality = "locality",
verbatimLocality = "verbatimLocality"
)
```

`ec_geographic_map` *Map view of occurrence data points*

Description

Map view of occurrence data points

Usage

```
ec_geographic_map(
  data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)
```

Arguments

`data`	Data table
`latitude`	default set to "decimalLatitude"
`longitude`	default set to "decimalLongitude"

Value

A map view shows occurrence records.

Examples

```
## Not run:
data <- data.frame(
  scientificName = "Mexacanthina lugubris",
  decimalLongitude = c(-117, -117.8, -116.9),
  decimalLatitude = c(32.9, 33.5, 31.9),
  temperature_mean = c(12, 13, 14),
  temperature_min = c(9, 6, 10),
  temperature_max = c(14, 16, 18)
)
ec_geographic_map(data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)

## End(Not run)
```

`ec_geographic_map_w_flag`

Map view to visualize data points with outlier probability 0 to 1 on a map view

---

Description

Map view to visualize data points with outlier probability 0 to 1 on a map view

Usage

```
ec_geographic_map_w_flag(
  data,
  flag_column,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)
```

Arguments

`data`	Data table which has information of coordinates (decimalLongitude and decimalLatitude) and a column which has flags 0 to 1
`flag_column`	column name which has flag, e.g. flag_outlier
`latitude`	default set on "decimalLatitude", change if the name of column is different.
`longitude`	default set on "decimalLongitude", change if the name of column is different.

Value

A geographic map which shows occurrence data points with the color gradient to show flagged records in warm color.

Examples

```
## Not run:
data <- data.frame(
  scientificName = "Mexacanthina lugubris",
  decimalLongitude = c(-117, -117.8, -116.9),
  decimalLatitude = c(32.9, 33.5, 31.9),
  temperature_mean = c(12, 13, 14),
  temperature_min = c(9, 6, 10),
  temperature_max = c(14, 16, 18),
  flag_outlier = c(0, 0.5, 1)
)
ec_geographic_map_w_flag(data,
  flag_column = "flag_outlier",
  latitude = "decimalLatitude",
  longitude = "decimalLongitude"
)

## End(Not run)
```

---

ec_impute_env_values *Impute Environmental Variables using Mean Values of occurrences within a certain radius*

Description

Impute Environmental Variables using Mean Values of occurrences within a certain radius

Usage

```
ec_impute_env_values(
  data_x,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude",
  radius_km = 10,
  iter = 3
)
```

Arguments

data_x	this is data_x which is the output of ec_extract_env_layers
latitude	default set to "decimalLatitude"
longitude	default set to "decimalLongitude"
radius_km	radius to average the values of data points within the circle to imput the values for missing datta points
iter	number of times to iterate the imputation, e.g. 1 or 2 or 3

Value

An updated table of data_x which has imputed values for the missing env variables, condition applies that the this imputation wont work if the data points are too sparse.

Examples

```
data_x <- data.frame(
  scientificName = "Mexacanthina lugubris",
  decimalLongitude = c(-117, -117.8, -116.9),
  decimalLatitude = c(32.9, 33.5, 31.9),
  BO_sstmean = c(12, NA, 14),
  BO_sstmin = c(9, NA, 10),
  BO_sstmax = c(14, NA, 18)
)
radius_km <- 10
iter <- 3
data_x <- ec_impute_env_values(data_x,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude",
  radius_km, iter
)
```

---

ec_merge_corrected_coordinates

Merge the Update Georeferenced Occurrence Points back to the Main Data File.

Description

Merge the Update Georeferenced Occurrence Points back to the Main Data File.

Usage

```
ec_merge_corrected_coordinates(
  data_corrected,
  data,
  catalog = "cleaned_catalog",
  latitude = "decimalLatitude",
  longitude = "decimalLongitude",
  uncertainty_col = "coordinateUncertaintyInMeters"
)
```

Arguments

data_corrected	After assigning coordinate values using online georeference tools such as GeoLocate, upload the csv file back to R with the name call data_corrected, we hardcoded the field names as "corrected_longitude", "corrected_latitude" and "corrected_uncertainty" and "cleaned_catalog" for column names of data_corrected dataset" which will be merge with "decimalLongitude", "decimalLantitude", "coordinateUncertaintyInMeters" and "cleaned_catalog" of data table.
data	data table which needs to updated with the assign coordiantes

`catalog`	this is an important attribute to use matching the records back to the main data file.
`latitude`	default set to "decimalLatitude", this is a column name of data
`longitude`	default set to "decimalLongitude", this is a column name of data
`uncertainty_col`	this is a column name of data and default set to "coordinateUncertaintyInMeters"

Value

A data frame with updated coordinate information

Examples

```
data <- data.frame(
  species = "A",
  decimalLongitude = c(-120, -119.8, NA, NA),
  decimalLatitude = c(20, 34, NA, NA),
  cleaned_catalog = c("12345", "89888", "LACM8898", "SDNHM6767"),
  locality = c(NA, NA, "Los Angeles, CA", "San Pedro, CA"),
  coordinateUncertaintyInMeters = c(9999, NA, NA, NA)
)
data_corrected <- data.frame(
corrected_longitude = c(-120, -119.8, 118, 118.3),
 corrected_latitude = c(20, 34, 33, 32.9),
 cleaned_catalog = c("12345", "89888", "LACM8898", "SDNHM6767"),
 corrected_uncertainty = c(9999, NA, 5000, 1000)
)


data<- ec_merge_corrected_coordinates(data_corrected, data,
catalog = "cleaned_catalog",
latitude = "decimalLatitude",
longitude = "decimalLongitude",
uncertainty_col = "coordinateUncertaintyInMeters" )
```

`ec_plot_distance` *Scatter Plot between geo_distance vs maha_distance with geo- and maha- Quantile Threshold to Demonstrate the Outliers outside those threshold.*

Description

Scatter Plot between geo_distance vs maha_distance with geo- and maha- Quantile Threshold to Demonstrate the Outliers outside those threshold.

Usage

```
ec_plot_distance(
  x,
  geo_quantile = 0.99,
  maha_quantile = 0.99,
  iterative = TRUE,
  geo_distance = "geo_distance",
  maha_distance = "maha_distance"
)
```

Arguments

`x`	iteration_list derived from ec_flag_outlier can be used to plot these scatter plots between geo_distance vs maha_distance
`geo_quantile`	value with geo_quantile percentile would consider has threshold for geo_distance to derive the outlier. e.g. default 0.99
`maha_quantile`	value with maha_quantile percentile would consider has threshold for maha_distance to derive the outlier. e.g. default 0.99
`iterative`	= TRUE/FALSE, default set on TRUE, which provide a iterative loop to check maps of each iteration of listed outcome of outlier probability, if it is FALSE, loop exit with first iteration outcome of outlier probability.
`geo_distance`	default set on "geo_distance", this column has calculated distance - output of ec_flag_outlier
`maha_distance`	default set on "maha_distance", this column has calculated distance - output of ec_flag_outlier

Value

A list of plots for each iteration outcome

Examples

```
df1 <- data.frame(
  latitude = runif(5, 30, 35),
  longitude = runif(5, -120, -115),
  temperature = rnorm(5, 15, 2),
  pH = rnorm(5, 8, 0.1),
  geo_distance = runif(5, 0, 100),
  maha_distance = runif(5, 0, 10)
)
df2 <- data.frame(
 latitude = runif(5, 30, 35),
 longitude = runif(5, -120, -115),
 temperature = rnorm(5, 16, 2),
 pH = rnorm(5, 7.9, 0.1),
 geo_distance = runif(5, 0, 100),
 maha_distance = runif(5, 0, 10)
)
```

```
iteration_list <- list(df1, df2)#Store both data frames in a list

iteration_list <- list(df1, df2)
plot <- ec_plot_distance(iteration_list, geo_quantile = 0.99, maha_quantile = 0.99,
iterative = TRUE)
```

---

ec_plot_var_range *Plot cleaned data overlay overall occurrence data to demonstrate accepted ranges of spatial and non-spatial attributes*

Description

Plot cleaned data overlay overall occurrence data to demonstrate accepted ranges of spatial and non-spatial attributes

Usage

```
ec_plot_var_range(
  data,
  summary_df,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude",
  env_layers
)
```

Arguments

`data`	data table which even has outlier data points
`summary_df`	summmary output of final cleaned data, after executing function ec_var_summary
`latitude`	default set to "decimalLatitude"
`longitude`	default set to "decimalLongitude"
`env_layers`	list of environmental variables

Value

A plot which shows spatial and environmental variables with the acceptable range for species habitability

Examples

```
data <- data.frame(
  scientificName = "Mexacanthina lugubris",
  decimalLongitude = c(-117, -117.8, -116.9, -116.5),
  decimalLatitude = c(32.9, 33.5, 31.9, 32.4),
  temperature_mean = c(12, 13, 14, 11),
  temperature_min = c(9, 6, 10, 10),
  temperature_max = c(14, 16, 18, 17),
  flag_outlier = c(0, 0.5, 1, 0.7)
) # this data table has data points which was considered as outliers

data_x <- data.frame(
  scientificName = "Mexacanthina lugubris",
  decimalLongitude = c(-117, -117.8, -116.5),
  decimalLatitude = c(32.9, 33.5, 32.4),
  temperature_mean = c(12, 13, 11),
  temperature_min = c(9, 6, 10),
  temperature_max = c(14, 16, 17),
  flag_outlier = c(0, 0.5, 0.7)
)
# cleaned data base after removing outliers >x probability.
# in this example, removed data points >0.7 probability to be
# considering outliers


env_layers <- c("temperature_mean", "temperature_min", "temperature_max")
summary_df <- ec_var_summary(data_x,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude",
  env_layers
)
# this is the final cleaned data table which
# will be used to derive summary of acceptable niche

ec_plot_var_range(data,
  summary_df,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude",
  env_layers
)
```

`ec_rm_duplicate` *Remove Duplicate Records from the Merged Data*

Description

Remove Duplicate Records from the Merged Data

Usage

```
ec_rm_duplicate(data, catalogNumber = "catalogNumber", abundance = "abundance")
```

Arguments

`data`	this is merge data frame which is a output file after running ec_db_merge
`catalogNumber`	this is a mandatory field which consider unique for each occurrence record.
`abundance`	this is a mandatory field which has created while data extraction by combining individual count and quantity fields (may vary from one source to another, we aim to standardize those as "abundance").

Details

This function will provide a cleaned_catalog column as output, which has catalog numbers standardize and removed duplicates based on generated cleaned_catalog and abundance columns of data. mandatory fields are catalogNumber, source and abundance

Value

A data frame which has unique catalog numbers. the output file will have cleaned_catalog field instead of catalogNumber. Also the unique record will be chosen with the abundance value if there is any.

Examples

```
db1 <- data.frame(
  species = "A",
  decimalLongitude = c(-120.2, -117.1, NA, NA),
  decimalLatitude = c(20.2, 34.1, NA, NA),
  catalogNumber = c("12345", "89888", "LACM8898", "SDNHM6767"),
  occurrenceStatus = c("present", "", "ABSENT", "Present"),
  basisOfRecord = c("preserved_specimen", "", "fossilspecimen", "material_sample"),
  source = "db1",
  abundance = c(1, NA, 8, 23)
)

db2 <- data.frame(
  species = "A",
  decimalLongitude = c(-120.2, -117.1, NA, NA),
  decimalLatitude = c(20.2, 34.1, NA, NA),
  catalogNumber = c("123452", "898828", "LACM82898", "SDNHM62767"),
  occurrenceStatus = c("present", "", "ABSENT", "Present"),
  basisOfRecord = c("preserved_specimen", "", "fossilspecimen", "material_sample"),
  source = "db2",
  abundance = c(1, 2, 3, 19)
)
db_list <- list(db1, db2)
merge_modern_data <- ec_db_merge(db_list = db_list, "modern")
ecodata <- ec_rm_duplicate(merge_modern_data,
  catalogNumber = "catalogNumber",
  abundance = "abundance"
)
```

ec_rm_duplicate_occurid

Remove Duplicate Records from the Merged Data based on occurrenceID

Description

Remove Duplicate Records from the Merged Data based on occurrenceID

Usage

```
ec_rm_duplicate_occurid(
  data,
  occurrenceID = "occurrenceID",
  abundance = "abundance"
)
```

Arguments

`data`	this is merge data frame which is a output file after running ec_db_merge
`occurrenceID`	this is a mandatory field which consider unique for each occurrence record.
`abundance`	this is a mandatory field which has created while data extraction by combining individual count and quantity fields (may vary from one source to another, we aim to standardize those as "abundance").

Details

This function will provide a cleaned_occurrenceID column as output, which has occurrenceID standardize and removed duplicates based on generated cleaned_occurrenceID and abundance columns of data. mandatory fields are occurrenceID, source and abundance

Value

A data frame which has unique occurrenceID. the output file will have cleaned_occurrenceID field instead of occurrenceID. Also the unique record will be chosen with the abundance value if there is any.

Examples

```
db1 <- data.frame(
  species = "A",
  decimalLongitude = c(-120.2, -117.1, NA, NA),
  decimalLatitude = c(20.2, 34.1, NA, NA),
  occurrenceID = c("12345", "898828", "LACM8289", "SDNHM6276"),
  occurrenceStatus = c("present", "", "ABSENT", "Present"),
  basisOfRecord = c("preserved_specimen", "", "fossilspecimen", "material_sample"),
  source = "db1",
  abundance = c(1, NA, 8, 23)
)

db2 <- data.frame(
  species = "A",
  decimalLongitude = c(-120.2, -117.1, NA, NA),
  decimalLatitude = c(20.2, 34.1, NA, NA),
  occurrenceID = c("12345", "898828", "LACM82898", "SDNHM62767"),
  occurrenceStatus = c("present", "", "ABSENT", "Present"),
  basisOfRecord = c("preserved_specimen", "", "fossilspecimen", "material_sample"),
  source = "db2",
  abundance = c(1, 2, 3, 19)
)
db_list <- list(db1, db2)
merge_modern_data <- ec_db_merge(
  db_list = db_list, "modern"
)
ecodata <- ec_rm_duplicate_occurid(
  merge_modern_data,
  occurrenceID = "occurrenceID",
  abundance = "abundance"
)
```

ec_trail_zero   *Trail Zeros from the Coordinate Values*

Description

Trail Zeros from the Coordinate Values

Usage

```
ec_trail_zero(coord)
```

Arguments

coord	A coordinate value in the numeric format of decimal degree

Value

A numerical trailed coordinate value.

Examples

```
ec_trail_zero(12.7000000)
ec_trail_zero(45.000000)
```

`ec_var_summary` *A Summary Table of Final Cleaned Spatial and Environmental Variables*

Description

A Summary Table of Final Cleaned Spatial and Environmental Variables

Usage

```
ec_var_summary(
  data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude",
  env_layers
)
```

Arguments

`data`	data table after cleaning the records
`latitude`	default set to "decimalLatitude"
`longitude`	default set to "decimalLongitude"
`env_layers`	an array of col names of enviornmental layers

Value

A summary table with the mean, min and max values of final cleaned spatial and environmental variables

Examples

```
data <- data.frame(
  scientificName = "Mexacanthina lugubris",
  decimalLongitude = c(-117, -117.8, -116.9, -116.5),
  decimalLatitude = c(32.9, 33.5, 31.9, 32.4),
  BO_sstmean = c(12, 13, 14, 11),
  BO_sstmin = c(9, 6, 10, 10),
  BO_sstmax = c(14, 16, 18, 17)
)
env_layers <- c("BO_sstmean", "BO_sstmin", "BO_sstmax")
ec_var_summary(data,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude",
  env_layers
)
```

ec_worms_synonym *Check Accepted Synonyms from WoRMs Taxonomy*

Description

Check Accepted Synonyms from WoRMs Taxonomy

Usage

```
ec_worms_synonym(
  species_name,
  data,
  scientificName = "scientificName",
  verbose = TRUE
)
```

Arguments

species_name	input species name.e.g. Mexacanthina lugubris
data	data table which has information of all occurrence data of the selected species
scientificName	default set to scientificName, this is a column in the data extracted from online sources, may have various synonyms of species_name.
verbose	default value as TRUE

Value

A table with two columns, column one represent the accepted synonyms, and column two demonstrate the unique species names from the occurrence data base with the number of records tagged under species names.

Examples

```
## Not run:
species_name <- "Mexacanthina lugubris"
data <- data.frame(
  scientificName = "Mexacanthina lugubris",
  decimalLongitude = c(-120, -78, -110, -60, -75, -130, -10, 5),
  decimalLatitude = c(20, 34, 30, 10, 40, 25, 15, 35)
)
comparison <- ec_worms_synonym(
  species_name,
  data,
  scientificName = "scientificName",
  verbose = TRUE
)
print(comparison)

## End(Not run)
```

`example_sp_invertebase` *dataset6: Documentation of data file - example_sp_invertebase.rda*

Description

This is a data dump downloaded from invertEbase, as the R package link with InverEbase is currently archive and not maintained, we are providing an example file.

Usage

```
example_sp_invertebase
```

Format

A data frame with 710 rows and 20 variables:

source invertEbase

catalogNumber CatalogNumber

basisOfRecord type of observations

occurrenceStatus presence or absent

institutionCode Institution code

verbatimEventDate when was this occurrence created

scientificName species name

individualCount abundance

organismQuantity abundance

abundance abundance

decimalLatitude Latitude in decimal degrees

decimalLongitude Longitude in decimal degrees

coordinateUncertaintyInMeters uncertainty of coordiantes

locality location information

verbatimLocality verbatim location information

municipality municipality

country country

stateProvince State or Provinces

county county

countryCode country code

Source

- this file is downloaded file from invertEBase for species - "Mexacanthina lugubris" and modified field names based on TDWG standard.

haversine_kmeans *Calculate Harversine distance*

Description

Calculate Harversine distance

Usage

```
haversine_kmeans(data, latitude, longitude, k)
```

Arguments

`data`	is a dataframe with spatial attributes - Latitude and Logitude
`latitude`	nested imput from ec_flag_outlier
`longitude`	nested imput from ec_flag_outlier
`k`	is number of cluster required for the data set you have. Normally visual inspection can give a sense on number of clusters. Cautious to have more than expected clusters to fit all data points, as overfitting can end up inluding bad data points in the analysis. e.g. k = 3

Value

A data frame with centroid and clusters using Harversine distance matrix

Examples

```
data_x <- data.frame(
  scientificName = "Mexacanthina lugubris",
  decimalLongitude = c(-117, -117.8, -116.9),
  decimalLatitude = c(32.9, 33.5, 31.9),
  BO_sstmean = c(12, 13, 14),
  BO_sstmin = c(9, 6, 10),
  BO_sstmax = c(14, 16, 18)
)

result <- haversine_kmeans(
  data_x,
  latitude = "decimalLatitude",
  longitude = "decimalLongitude",
  k = 3
)
```

Index