Package 'gatoRs'

November 30, 2023

Type Package

Title Geographic and Taxonomic Occurrence R-Based Scrubbing

Version 1.0.1

Date 2023-11-29

Imports ridigbio, dplyr (>= 1.1.0), rgbif, magrittr, CoordinateCleaner (>= 3.0.1), terra, spThin, stringr, leaflet, parsedate, spatstat.geom

Encoding UTF-8

VignetteBuilder knitr

LazyData true

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, testthat

License GPL-3

Description Streamlines downloading and cleaning biodiversity data from Integrated Digitized Biocollections (iDigBio) and the Global Biodiversity Information Facility (GBIF).

Maintainer Natalie N. Patten <natalienpatten@gmail.com>

URL https://nataliepatten.github.io/gatoRs/, https://github.com/nataliepatten/gatoRs

BugReports https://github.com/nataliepatten/gatoRs/issues

Depends R (>= 3.5.0)

NeedsCompilation no

Author Natalie N. Patten [aut, cre] (<https://orcid.org/0000-0001-8090-1324>), Michelle L. Gaynor [aut] (<https://orcid.org/0000-0002-3912-6079>), Douglas E. Soltis [ctb] (<https://orcid.org/0000-0001-8638-4137>), Pamela S. Soltis [ctb] (<https://orcid.org/0000-0001-9310-8659>)

Repository CRAN

Date/Publication 2023-11-30 10:00:02 UTC

R topics documented:

`basic_locality_clean` *Locality Cleaning - Remove missing and improbable coordinates*

Description

The `basic_locality_clean()` function cleans locality by removing missing or impossible coordinates and correcting precision. This function requires columns named 'latitude' and 'longitude'. These columns should be of type 'numeric'.

Usage

```
basic_locality_clean(
  df,
  latitude = "latitude",
  longitude = "longitude",
  remove.zero = TRUE,
  precision = TRUE,
  digits = 2,
  remove.skewed = TRUE,
  info.withheld = "informationWithheld"
)
```

Arguments

df	Data frame of occurrence records returned from gators_download().
latitude	Default = "latitude". The name of the latitude column in the data frame.
longitude	Default = "longitude". The name of the longitude column in the data frame.
remove.zero	Default = TRUE. Indicates that points at (0.00, 0.00) should be removed.
precision	Default = TRUE. Indicates that coordinates should be rounded to match the coordinate uncertainty.
digits	Default = 2. Indicates digits to round coordinates to when precision = TRUE.
remove.skewed	Default = TRUE. Utilizes the remove_skewed() function to remove skewed coordinate values.
info.withheld	Default = "informationWithheld". The name of the information withheld column in the data frame.

Details

This function removes any records with missing coordinates, impossible coordinates, coordinates at (0,0), and any that are flagged as skewed. These skewed records are identified with the remove_skewed() function which identifies rows where the 'InformationWitheld' column includes the string "Coordinate uncertainty increased". We also provide the option to round the provided latitude and longitude values to a specified number of decimal places. This function requires no additional packages.

Value

Return data frame with specimen removed that had missing or improper coordinate values. Information about the columns in the returned data frame can be found in the documentation for gators_download().

Examples

```
cleaned_data <- basic_locality_clean(data)
```

---

basis_clean *Basis Cleaning - Removes records with certain record basis*

Description

The basis_clean() function removes records based on basisOfRecord column.

Usage

```
basis_clean(df, basis.list = NA, basis.of.record = "basisOfRecord")
```

Arguments

df	Data frame of occurrence records returned from gators_download().
basis.list	A list of basis to keep. If a list is not supplied, the filter will be interactive and users must respond to the function.
basis.of.record	Default = "basisOfRecord". The name of the basis of record column in the data frame.

Details

With the interactive method, the function will print all unique basisOfRecord values in the current data set and then ask the user to respond in the console to prompts regarding which records, if any, should be removed based on their basisOfRecord. This function requires no additional packages.

Value

Returns a data frame with records of desired record basis. Information about the columns in the returned data frame can be found in the documentation for gators_download().

Examples

```
cleaned_data <- basis_clean(data, basis.list = c("Preserved Specimen","Physical specimen"))
```

---

citation_bellow *Cite Data - Get GBIF citations*

Description

The citation_bellow function retrieves and returns the citation information for records from GBIF, or where aggregator = "GBIF".

Usage

```
citation_bellow(df, id = "ID", aggregator = "aggregator")
```

Arguments

df	Data frame of occurrence records returned from gators_download().
id	Default = "ID". The name of the id column in the data frame, which contains unique IDs defined from GBIF (keys) or iDigBio (UUID).
aggregator	Default = "aggregator". The name of the column in the data frame that identifies the aggregator that provided the record. This is equal to iDigBio or GBIF.

Details

This function requires the rgbif package.

Value

Returns a list with citation information for the GBIF data downloaded.

Examples

```
citations <- citation_bellow(data)
```

`data_chomp` *Subset Data - Get species, longitude, and latitude columns*

Description

The `data_chomp()` function "chomps" (subsets) a data frame of occurrence records to only contain the following columns: "species", "longitude", and "latitude". After using this function data will be ready for use in Maxent, for example.

Usage

```
data_chomp(
  df,
  accepted.name = NA,
  longitude = "longitude",
  latitude = "latitude"
)
```

Arguments

`df`	Data frame of occurrence records returned from `gators_download()`.
`accepted.name`	The accepted species name for the records.
`longitude`	Default = "longitude". The name of the longitude column in the data frame.
`latitude`	Default = "latitude". The name of the latitude column in the data frame.

Details

This function requires the package dplyr.

Value

Returns data frame with three columns: "species", "longitude", and "latitude". The "species" column is set by the accepted.name input. This data frame is ready for downstream applications such as Maxent.

Examples

```
chomped_data <- data_chomp(data, accepted.name = "Galax urceolata")
```

`filter_fix_names` *Used in gators_download() - Filter iDigBio results by scientific name*

Description

The `filter_fix_names()` function filters a data frame for relevant results, based on the scientific name given. Some downloaded results from iDigBio might contain occurrences of other species that have "notes" or "locality" strings that mentioning the desired species. Here we only retain those where the scientificName column is found to be a fuzzy match to a value in the user-provided list containing the scientific name and applicable synonym Hence, this function looks for relevant results that are actually occurrences of the desired species.

Usage

```
filter_fix_names(
  df,
  synonyms.list,
  filter = "fuzzy",
  scientific.name = "scientificName",
  accepted.name = NA
)
```

Arguments

`df`	Data frame with name column to be fixed.
`synonyms.list`	A list of synonyms for a species.
`filter`	Default = "fuzzy". Indicates the type of filter to be used–either "exact" or "fuzzy".
`scientific.name`	Default = "scientificName". The name of the scientificName column in the data frame.
`accepted.name`	The accepted scientific name for the species. If provided, an additional column will be added to the data frame with the accepted name for further manual comparison.

Details

This function requires no additional packages.

Value

Returns data frame with filtered results.

Examples

```
cleaned_data <- filter_fix_names(data, c("Galax urceolata", "Galax aphylla"), filter = "exact")
cleaned_data <- filter_fix_names(data, c("Galax urceolata", "Galax aphylla"),
accepted.name = "Galax urceolata")
```

`fix_columns` *Used in gators_download() - Fill out taxonomic name columns*

Description

The `fix_columns()` function fills out the taxonomic name columns based on available information in the data set. For example, if a row has a name provided for the scientificName column, this information will be used to generate the respective genus, specificEpithet, and infraspecificEpithet columns for that row.

Usage

```
fix_columns(
  df,
  scientific.name = "scientificName",
  genus = "genus",
  species = "specificEpithet",
  infraspecific.epithet = "infraspecificEpithet"
)
```

Arguments

`df`	Data frame of occurrence records.
`scientific.name`	Default = "scientificName". The name of the scientificName column in the data frame.
`genus`	Default = "genus". The name of the genus column in the data frame.
`species`	Default = "specificEpithet". The name of the specificEpithet column in the data frame.
`infraspecific.epithet`	Default = "infraspecificEpithet". The name of the infraspecificEpithet column in the data frame.

Details

This function requires package stringr.

Value

Returns the original data frame with the specified columns.

Examples

```
fixed_data <- fix_columns(data)
```

`fix_names` *Used in gators_download() - Fix taxonomic name capitalization*

Description

The `fix_names()` function fixes the capitalization of species names in the data frame provided to align with accepted capitalization standards.

Usage

```
fix_names(df, scientific.name = "scientificName")
```

Arguments

`df`	Data frame with name column to be fixed.
`scientific.name`	Default = "scientificName". The name of the scientificName column in the data frame.

Details

This function uses the `fixAfterPeriod()` function. This function requires package stringr.

Value

Returns df with fixed capitalization in name column.

Examples

```
fixed_data <- fix_names(data)
```

full_clean *Full Cleaning - Wrapper function to speed clean*

Description

The full_clean() function performs automated cleaning steps, including options for: removing duplicate data points, checking locality precision, removing points with skewed coordinates, removing plain zero records, removing records based on basis of record, and spatially thinning collection points. This function also provides the option to interactively inspect and remove types of basis of record.

Usage

```
full_clean(
  df,
  synonyms.list,
  event.date = "eventDate",
  year = "year",
  month = "month",
  day = "day",
  occ.id = "occurrenceID",
  remove.NA.occ.id = FALSE,
  remove.NA.date = FALSE,
  aggregator = "aggregator",
  id = "ID",
  taxa.filter = "fuzzy",
  scientific.name = "scientificName",
  accepted.name = NA,
  remove.zero = TRUE,
  precision = TRUE,
  digits = 2,
  remove.skewed = TRUE,
  basis.list = NA,
  basis.of.record = "basisOfRecord",
  latitude = "latitude",
  longitude = "longitude",
  remove.flagged = TRUE,
  thin.points = TRUE,
  distance = 5,
  reps = 100,
  one.point.per.pixel = TRUE,
  raster = NA,
  resolution = 0.5,
  remove.duplicates = TRUE
)
```

Arguments

df	Data frame of occurrence records.
synonyms.list	A list of synonyms for a species.
event.date	Default = "eventDate". The name of the event date column in the data frame.
year	Default = "year". The name of the year column in the data frame.
month	Default = "month". The name of the month column in the data frame.
day	Default = "day". The name of the day column in the data frame.
occ.id	Default = "occurrenceID". The name of the occurrenceID column in the data frame.
remove.NA.occ.id	Default = FALSE. This will remove records with missing occurrence IDs when set to TRUE.
remove.NA.date	Default = FALSE. This will remove records with missing event dates when set to TRUE.
aggregator	Default = "aggregator". The name of the column in the data frame that identifies the aggregator that provided the record. This is equal to iDigBio or GBIF.
id	Default = "ID". The name of the id column in the data frame, which contains unique IDs defined from GBIF (keys) or iDigBio (UUID).
taxa.filter	The type of filter to be used–either "exact", "fuzzy", or "interactive".
scientific.name	Default = "scientificName". The name of the scientificName column in the data frame.
accepted.name	The accepted scientific name for the species. If provided, an additional column will be added to the data frame with the accepted name for further manual comparison.
remove.zero	Default = TRUE. Indicates that points at (0.00, 0.00) should be removed.
precision	Default = TRUE. Indicates that coordinates should be rounded to match the coordinate uncertainty.
digits	Default = 2. Indicates digits to round coordinates to when precision = TRUE.
remove.skewed	Default = TRUE. Utilizes the remove_skewed() function to remove skewed coordinate values.
basis.list	A list of basis to keep. If a list is not supplied, this filter will not occur.
basis.of.record	Default = "basisOfRecord". The name of the basis of record column in the data frame.
latitude	Default = "latitude". The name of the latitude column in the data frame.
longitude	Default = "longitude". The name of the longitude column in the data frame.
remove.flagged	Default = TRUE. An option to remove points with problematic locality information.
thin.points	Default = TRUE. An option to spatially thin occurrence records.
distance	Default = 5. Distance in km to separate records.

reps Default = 100. Number of times to perform thinning algorithm.

one.point.per.pixel
Default = TRUE. An option to only retain one point per pixel.

raster Raster object which will be used for ecological niche comparisons. A SpatRaster should be used.

resolution Default = 0.5. Options - 0.5, 2.5, 5, and 10 (in min of a degree). 0.5 min of a degree is equal to 30 arc sec.

remove.duplicates
Default = TRUE. An option to remove duplicate points.

Details

This function is entirely automated and thus does not take advantage of the interactive options provided in the individual cleaning functions. Using this wrapper is recommended for data processing that does not require interactive/manual cleaning and inspection. All cleaning steps, except taxonomic harmonization, can be bypassed by setting their associated input variables to FALSE. This function requires packages dplyr, magrittr, and raster.

Value

df is a data frame with the cleaned data. Information about the columns in the returned data frame can be found in the documentation for `gators_download()`. An additional column named "accepted_name" will be returned if an accepted.name was provided.

Examples

```
cleaned_data <- full_clean(data, synonyms.list = c("Galax urceolata", "Galax aphylla"),
digits = 3, basis.list = c("Preserved Specimen","Physical specimen"),
accepted.name = "Galax urceolata", remove.flagged = FALSE)
```

---

gators_download *Download - Download specimen data from both iDigBio and GBIF*

Description

The `gators_download()` function downloads data from GBIF and iDigBio for your desired species.

Usage

```
gators_download(
  synonyms.list,
  write.file = FALSE,
  filename = NA,
  gbif.match = "fuzzy",
  gbif.prov = FALSE,
  idigbio.filter = TRUE,
  limit = 1e+05
)
```

Arguments

synonyms.list	A list of scientific names including the accepted scientific name and any synonyms for your desired species. For example, synonyms.list = c("Asclepias curtissii","Asclepias aceratoides", "Asclepias arenicola", "Oxypteryx arenicola", "Oxypteryx curtissii"). This parameter is required.
write.file	A parameter to choose whether to produce a .csv file containing search results. This parameter is not required and is assigned FALSE by default.
filename	The path and file name for the retrieved data. Note that this parameter should include the ".csv" extension as well. For example, filename = "base_folder/other_folder/my_file.cs The file path can be entered either as relative to the current working directory (example: "../my_file.csv") or as a full path. This parameter is required if write.file = TRUE.
gbif.match	A parameter to select either search by fuzzy matching of scientific name or to search by species code. For example, gbif.match = "fuzzy" will search by fuzzy match and gbif.match = "code" will search by code. This parameter is not required and is assigned "fuzzy" by default.
gbif.prov	A parameter to obtain the provider/verbatim columns from GBIF. This parameter is optional and is assigned FALSE by default.
idigbio.filter	A parameter to remove less relevant search results from iDigBio. Based on the search input, results may include data points for a different species that mention the desired species in the locality information, for example. Choosing idigbio.filter = TRUE will return the data frame with rows in which the name column fuzzy matches a name on the synonym list. This parameter is not required and is assigned TRUE by default.
limit	Default = 100,000 (maximum). Set limit to the number of records requested for each element in synonyms.list from each aggregator.

Details

With gators_download() you can obtain biodiversity records for your species of interest from both GBIF and iDigBio. This function is innovative in how it searches iDigBio. Unlike spocc::occ(), we do not query the iDigBio API using the scientific name field, as this will only return exact matches. Instead, we designed a "pseudo-fuzzy match" to search all fields for partial matches to the supplied scientific names. This function uses the get_idigbio(), get_gbif(), fix_columns(), fix_names(), and filter_fix_names() functions. This function requires packages magrittr, rgbif, dplyr, ridigbio, and stringr.

Value

Returns a data frame and writes a csv file as specified in the input. This csv file will contain search results for the desired species from the GBIF and iDigBio databases. The columns are as follows:

- scientificName
- genus
- specificEpithet
- infraspecificEpithet

- ID (contains unique IDs defined from GBIF or iDigBio)
- occurrenceID
- basisOfRecord
- eventDate
- year
- month
- day
- institutionCode
- recordedBy
- informationWithheld
- country
- county
- stateProvince
- locality
- latitude
- longitude
- coordinateUncertaintyInMeters
- habitat
- aggregator (either GBIF or iDigBio)

Examples

```
df <- gators_download(synonyms.list = c("Galax urceolata", "Galax aphylla"), limit = 10)
df <- gators_download(synonyms.list = "Galax urceolata", gbif.match = "code",
idigbio.filter = FALSE, limit = 10)
```

`gators_merge` *Merge Retained Data - Combined original data set with georeferenced or retained records.*

Description

The `gators_merge()` function combines two data sets with identical column names and returns a single data set.

Usage

```
gators_merge(df1, df2)
```

Arguments

df1	A data frame downloaded with gators_download() and prepared using remove_missing().
df2	A data frame with the same columns as df1, but with observations generated through georeferencing or through data requests.

Details

Prior to combining a data set with georeferenced or retrieved data, please use the remove_missing() function to limit duplicate records. This function requires no additional packages.

Value

A combined data set.

Examples

```
removed_missing <- remove_missing(data)
needs_geo <- need_to_georeference(data)
# fill in manually georeferenced data into needs_geo...
merged_data <- gators_merge(removed_missing, needs_geo)
needs_data <- needed_records(data)
# fill in missing information with a data request...
merged_data <- gators_merge(merged_data, needs_data)
```

---

get_gbif *Used in gators_download() - Download data from the Global Biodiversity Information Facility*

Description

The get_gbif() function queries the Global Biodiversity Information Facility (GBIF) for your desired species. Limited to 100,000 record downloads.

Usage

```
get_gbif(synonyms.list, gbif.match = "fuzzy", gbif.prov = FALSE, limit = 1e+05)
```

Arguments

synonyms.list	A list of affiliated names for your query.
gbif.match	Default = "fuzzy". Either "fuzzy" for fuzzy matching of name or "code" to search by species code.
gbif.prov	Default = FALSE. A parameter to obtain the provider/verbatim columns from GBIF.
limit	Default = 100,000 (maximum). Set limit to the number of records requested for each element in synonyms.list.

Details

This function uses the `correct_class()` function. This function requires the packages rgbif, magrittr, and dplyr.

Value

Returns a data frame with desired columns from GBIF.

Examples

```
df <- get_gbif(c("Galax urceolata", "Galax aphylla"), limit = 5)
df <- get_gbif(c("Galax urceolata", "Galax aphylla"),
gbif.match = "code", limit = 5)
df <- get_gbif(c("Galax urceolata", "Galax aphylla"), gbif.prov = TRUE, limit = 5)
```

---

get_idigbio *Used in gators_download() - Download data from Integrated Digitized Biocollections*

Description

The `get_idigbio()` function queries iDigBio for your desired species. Limited to 100,000 record downloads.

Usage

```
get_idigbio(synonyms.list, limit = 1e+05)
```

Arguments

`synonyms.list`	A list of affiliated names for your query.
`limit`	Default = 100,000 (maximum). Set limit to the number of records requested for each element in synonyms.list.

Details

This function uses the `correct_class()` function. This function requires the packages ridigbio, magrittr, and dplyr.

Value

A data frame with desired columns from iDigBio.

Examples

```
df <- get_idigbio(c("Galax urceolata", "Galax aphylla"), limit = 100)
```

needed_records *Identify Missing Information - Find records with redacted or missing data*

Description

The needed_records() function identifies records with flags, therefore withheld. This indicates that information is withheld from these records due to endangered species status, for example. Accessing this information may require a permit. Or, these records can be removed from the data set with remove_redacted().

Usage

```
needed_records(df, info.withheld = "informationWithheld")
```

Arguments

df	A data frame downloaded with gators_download().
info.withheld	Default = "informationWithheld". The name of the information withheld column in the data frame.

Details

This function requires no additional packages.

Value

A data frame with only records for which locality was flagged as redacted or missing. Information about the columns in the returned data frame can be found in the documentation for gators_download().

Examples

```
need_info <- needed_records(data)
```

need_to_georeference *Identify Missing Information - Find records which lack coordinate information*

Description

The need_to_georeference() function allows you to find records that are missing coordinates but contain locality information. These records can then be manually georeferenced.

Usage

```
need_to_georeference(
  df,
  longitude = "longitude",
  latitude = "latitude",
  locality = "locality"
)
```

Arguments

df	A data frame downloaded with gators_download().
longitude	Default = "longitude". The name of the longitude column in the data frame.
latitude	Default = "latitude". The name of the latitude column in the data frame.
locality	Default = "locality". The name of the locality column in the data frame.

Details

This function requires no additional packages.

Value

Returns a data frame of the points that need to be georeferenced. Information about the columns in the returned data frame can be found in the documentation for gators_download().

Examples

```
need_coords <- need_to_georeference(data)
```

---

one_point_per_pixel *Spatial Correction - One point per pixel*

Description

The one_point_per_pixel function retains only one point per raster pixel. This function is useful when creating present-absent models.

Usage

```
one_point_per_pixel(
  df,
  raster = NA,
  resolution = 0.5,
  precision = TRUE,
  digits = 2,
  longitude = "longitude",
  latitude = "latitude"
)
```

Arguments

df	Data frame of occurrence records.
raster	Raster object which will be used for ecological niche comparisons. A SpatRaster should be used.
resolution	Default = 0.5. Options - 0.5, 2.5, 5, and 10 (in min of a degree). 0.5 min of a degree is equal to 30 arc sec.
precision	Default = TRUE. Indicates that coordinates should be rounded to match the coordinate uncertainty.
digits	Default = 2. Indicates digits to round coordinates to when precision = TRUE.
longitude	Default = "longitude". The name of the longitude column in the data frame.
latitude	Default = "latitude". The name of the latitude column in the data frame.

Details

This function requires package raster and spatstat.geom.

Value

df is a data frame with only one point per pixel. Information about the columns in the returned data frame can be found in the documentation for gators_download().

Examples

```
ready_data <- one_point_per_pixel(data)
```

---

process_flagged *Locality Cleaning - Find possibly problematic occurrence records*

Description

The process_flagged() function allows you to visualize and inspect possible problematic points, as well as manually remove these points, if desired. By default, this function is interactive. When running the function interactively you can hover over a point to see the record's scientific name, and click on a point to see the record's coordinates.The interactive option plots flagged points in red and non-flagged points in blue.

Usage

```
process_flagged(
  df,
  interactive = TRUE,
  latitude = "latitude",
  longitude = "longitude",
  scientific.name = "scientificName"
)
```

Arguments

df	Data frame of occurrence records returned from gators_download().
interactive	Default = TRUE. The interactive option allows for a visual display of possible problematic points and the ability to manually remove these points. Setting interactive = FALSE will automatically remove these points from the data frame.
latitude	Default = "latitude". The name of the latitude column in the data frame.
longitude	Default = "longitude". The name of the longitude column in the data frame.
scientific.name	Default = "scientificName". The name of the scientificName column in the data frame.

Details

This function is a wrapper to visualize results for the CoordinateCleaner::clean_coordinates() function. Briefly, CoordinateCleaner::clean_coordinates() flags records with coordinates that are unlikely valid, spatial outliers, or in certain locations including the ocean, state capitals, country centroids, the GBIF headquarters, and biodiversity institutions (including botanical gardens, museums, herbaria, etc.). This function requires packages CoordinateCleaner, leaflet, and magrittr. This function requires interactive user input.

Value

Return cleaned data frame. Information about the columns in the returned data frame can be found in the documentation for gators_download().

Examples

```
cleaned_data <- process_flagged(data, interactive = FALSE)
```

---

remove_duplicates *Remove Duplicates - Remove records with identical event dates and coordinates*

Description

The remove_duplicates() function removes records with identical event dates and occurrence IDs. Prior to utilizing this function, longitude and latitude columns should be rounded to match the coordinate uncertainty using the basic_locality_clean() function.

Usage

```
remove_duplicates(
  df,
  event.date = "eventDate",
  aggregator = "aggregator",
  id = "ID",
  occ.id = "occurrenceID",
  year = "year",
  month = "month",
  day = "day",
  latitude = "latitude",
  longitude = "longitude",
  remove.NA.occ.id = FALSE,
  remove.NA.date = FALSE,
  remove.unparseable = FALSE
)
```

Arguments

df	Data frame of occurrence records returned from gators_download().
event.date	Default = "eventDate". The name of the event date column in the data frame.
aggregator	Default = "aggregator". The name of the column in the data frame that identifies the aggregator that provided the record. This is equal to iDigBio or GBIF.
id	Default = "ID". The name of the id column in the data frame, which contains unique IDs defined from GBIF (keys) or iDigBio (UUID).
occ.id	Default = "occurrenceID". The name of the occurrenceID column in the data frame.
year	Default = "year". The name of the year column in the data frame.
month	Default = "month". The name of the month column in the data frame.
day	Default = "day". The name of the day column in the data frame.
latitude	Default = "latitude". The name of the latitude column in the data frame.
longitude	Default = "longitude". The name of the longitude column in the data frame.
remove.NA.occ.id	Default = FALSE. This will remove records with missing occurrence IDs when set to TRUE.
remove.NA.date	Default = FALSE. This will remove records with missing event dates when set to TRUE.
remove.unparseable	Default = FALSE. If we cannot parse the event date into individual year, month, day categories the user can manually specify. Otherwise, if set to TRUE, these rows will simply be removed.

Details

Here we identify and remove both (1) specimen duplicates and (2) aggregator duplicates based on each specimens coordinates, occurrenceID, and eventDate. To leverage all date information available, set `remove.unparseable = FALSE` to manually populate the year, month, and day columnsl. Dates are parsed based on ISO 8601 which only includes time since the Unix epoch, or January 1st, 1970, therefore dates that occur before 1970 will not be automatically parsed. If we are unable to parse the included date for particular records, users can choose to manually enter the year, month, and day for these records when prompted. If the user chooses to manually enter the event date, the records eventDate will be printed and the user will be asked to manually enter the year, month, and day of this eventDate into the console. Users are only prompted to manually parse event dates for records where year, month, and day are absent, but eventDate is present and cannot be parsed. This function also we also confirm all ID (UUID and key) are unique to remove any within-aggregator duplicates that may accumulate due to processing errors. This function requires the parsedate and dplyr packages. Warning, this function will ignore missing occurrence ID and year, month, day columns if not provided in the data set.

Value

Return data frame with duplicates removed. Information about the columns in the returned data frame can be found in the documentation for `gators_download()`.

Examples

```
cleaned_data <- remove_duplicates(data)
cleaned_data <- remove_duplicates(data, remove.NA.occ.id = TRUE, remove.NA.date = TRUE)
cleaned_data <- remove_duplicates(data, remove.unparseable = TRUE)
```

---

remove_missing *Remove Missing Information - Prepare to merge a data frame with georeferenced and retrieved records*

Description

The `remove_missing()` function identifies and removes records identified with the `need_to_georeference()` and `needed_records()` functions. This function should be utilized prior to merging georeferenced or retrieved records.

Usage

```
remove_missing(
  df,
  remove.type = "both",
  info.withheld = "informationWithheld",
  longitude = "longitude",
  latitude = "latitude",
  locality = "locality",
  id = "ID"
)
```

Arguments

df	A data frame downloaded with gators_download().
remove.type	Default equal to "both" indicating records identified with the need_to_georeference() function and needed_records() function are removed from the data frame. If equal to "georeference" then only records identified by the need_to_georeference() function are removed. If equal to "withheld" then only records identified with the needed_records() function are removed.
info.withheld	Default = "informationWithheld". The name of the information withheld column in the data frame.
longitude	Default = "longitude". The name of the longitude column in the data frame.
latitude	Default = "latitude". The name of the latitude column in the data frame.
locality	Default = "locality". The name of the locality column in the data frame.
id	Default = "ID". The name of the id column in the data frame, which contains unique IDs defined from GBIF (keys) or iDigBio (UUID).

Details

This function requires no additional packages.

Value

A data frame with records containing missing information removed. Information about the columns in the returned data frame can be found in the documentation for gators_download().

Examples

```
cleaned_data <- remove_missing(data)
```

---

remove_redacted *Remove Redacted Information - Remove protected or private records prior to publication*

Description

The remove_redacted() function identifies and removes records where 'aggregator' is not equal to iDigBio or GBIF.

Usage

```
remove_redacted(df, aggregator = "aggregator")
```

Arguments

df	A data frame downloaded with gators_download().
aggregator	Default = "aggregator". The name of the column in the data frame that identifies the aggregator that provided the record. This is equal to iDigBio or GBIF.

Details

This function requires no additional packages.

Value

A data frame with redacted records removed. Information about the columns in the returned data frame can be found in the documentation for `gators_download()`.

Examples

```
cleaned_data <- remove_redacted(data)
```

---

remove_skewed *Used in basic_locality_clean() - Removed skewed locality*

Description

The `remove_skewed()` function identifies and removes records where locality has been skewed. Records are considered skewed if informationWithheld contains the string "Coordinate uncertainty increased".

Usage

```
remove_skewed(df, info.withheld = "informationWithheld")
```

Arguments

`df`	A data frame downloaded with `gators_download()`.
`info.withheld`	Default = "informationWithheld". The name of the information withheld column in the data frame.

Details

This function requires no additional packages.

Value

A data frame with records remove only records for which locality was skewed.

Examples

```
cleaned_data <- remove_skewed(data)
```

`taxa_clean` *Taxonomic Cleaning - Filter and resolve taxon names*

Description

The `taxa_clean()` function filters a data frame for relevant results, based on the scientific name given. Filtering can be done with scripts by exact or fuzzy match. Or, for a more controlled approach, this function provides interactive filtering by providing the user with prompts. The interactive method allows the user to manually determine whether they wish to keep results containing certain scientific names.

Usage

```
taxa_clean(
  df,
  synonyms.list,
  taxa.filter = "fuzzy",
  scientific.name = "scientificName",
  accepted.name = NA
)
```

Arguments

`df`	Data frame of occurrence records returned from `gators_download()`.
`synonyms.list`	A list of synonyms for a species.
`taxa.filter`	The type of filter to be used–either "exact", "fuzzy", or "interactive".
`scientific.name`	Default = "scientificName". The name of the scientificName column in the data frame.
`accepted.name`	The accepted scientific name for the species. If provided, an additional column will be added to the data frame with the accepted name for further manual comparison.

Details

If users select the interactive approach, the function will first print all unique scientific names in the current data set and then ask the user to respond in the console to prompts regarding which records, if any, should be removed based on their scientific name. After filtering, based on a user-provided taxonomy, an accepted name column can be defined with an optional argument. This function relies on the user-provided taxonomy, we do not utilize any taxonomic backbone. Additionally, this function requires no additional packages.

Value

Returns data frame with filtered results and new column with the accepted name labeled as "accepted_name". Information about the columns in the returned data frame can be found in the documentation for `gators_download()`. An additional column named "accepted_name" will be returned if an accepted.name was provided.

Examples

```
cleaned_data <- taxa_clean(data, c("Galax urceolata", "Galax aphylla"), taxa.filter = "exact")
cleaned_data <- taxa_clean(data, c("Galax urceolata", "Galax aphylla"),
accepted.name = "Galax urceolata")
```

---

thin_points *Spatial Correction - Spatially thin records*

Description

The thin_points function returns records based on coordinate thinning based on a minimum nearest neighbor distance approach.

Usage

```
thin_points(
  df,
  accepted.name = NA,
  distance = 5,
  reps = 100,
  latitude = "latitude",
  longitude = "longitude"
)
```

Arguments

df	Data frame of occurrence records.
accepted.name	Accepted name of your species. This argument is not required if the data frame already contains an accepted_name column.
distance	Default = 5. Distance in km to separate records.
reps	Default = 100. Number of times to perform thinning algorithm.
latitude	Default = "latitude". The name of the latitude column in the data frame.
longitude	Default = "longitude". The name of the longitude column in the data frame.

Details

This function is a wrapper for spatial thinning using the spThin package (Aiello-Lammens et al., 2015) In summary, the thinning algorithm provided by spThin calculates the pairwise distances between data points, then randomly samples a single point from all points less than or equal to the set minimum nearest neighbor distance. This process is repeated until the pairwise distances among points do not fall below the minimum nearest neighbor distance.

Value

df is a data frame with the cleaned data. Information about the columns in the returned data frame can be found in the documentation for gators_download().

Examples

```
thinned_data <- thin_points(data, accepted.name = "Galax urceolata")
```

Index