Package 'parseGBIF'

August 4, 2023

Type Package

Title An R package for parsing species occurrence records

Version 0.0.1

Date 2023-06-04

Maintainer Pablo Melo <pablopains@yahoo.com.br>

Description parseGBIF package is designed to convert GBIF species occurrence data to a more comprehensible format to be used for further analysis, e.g. spatial.
The package provides tools for verifying and standardizing species scientific names and for selecting the most informative species records when duplicates are available.

License GPL (>= 2) | file LICENSE

Encoding UTF-8

LazyData true

LazyDataCompression xz

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports plyr,
    readxl,
    dplyr,
    tidyr,
    readr,
    stringr,
    textclean,
    googledrive,
    rvest,
    lubridate,
    rnaturalearthdata,
    jsonlite,
    sqldf,
    DT,
    downloader,
    tidyselect,
    utils

Remotes github::pablopains/parseGBIF

Depends R (>= 3.5.0)

R topics documented:

---

collectors_get_name   *Get the last name of the main collector*

Description

Get the last name of the main collector in recordedBy field

Usage

```
collectors_get_name(x)
```

Arguments

x   recordedBy field

Details

Returns the last name

Value

last name of the main collector

Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

See Also

[collectors_prepare_dictionary](), [collectors_update_dictionary]()

Examples

```
help(collectors_get_name)

collectors_get_name('Melo, P.H.A & Monro, A.')

collectors_get_name('Monro, A. & Melo, P.H.A')
```

```
collectors_prepare_dictionary
```
Prepare the list with the last name of the main collector

Description

Returns the list with the last name of the main collector associated with the unique key recordedBy. A necessary step for parsing duplicate records is generating a robust key for each unique collecting event (aka 'gathering') that will support the recognition of duplicate records. For this purpose we generate a string combining the plant family name + first collector's surname + the collection number. It is therefore essential to consistently record the collector surname and for this purpose we provide a collector dictionary. To extract the surname of the main collector based on the recordedBy field and assemble a list relating the last name of the main collector and the raw data from the recordedBy, use the collectors_prepare_dictionary function.

It is recommended to check the main collector's last name in the nameRecordedBy_Standard field. Our goal is to standardize the main collector's last name, which is automatically extracted from the recordedBy field. We do so by standardizing the text string so that it begins with an uppercase character and to replace non-ascii characters, so that collector reponsible for a collection event is always recorded using the same string of characters. If the searched recordedBy entry is present in the collector's dictionary, the function retrieves the last name of the main collector with reference to the recordedBy field (in which case the CollectorDictionary field will be flagged as 'checked'), otherwise, the function will return the last name of the main collector, extracted automatically from the recordedBy field .

Once verified, the collector's dictionary can be reused in the future.

Usage

```
collectors_prepare_dictionary(
  occ = NA,
  collectorDictionary_file =
   "https://raw.githubusercontent.com/pablopains/parseGBIF/main/collectorDictionary/CollectorsDi
  silence = TRUE
)
```

Arguments

occ	GBIF occurrence table with selected columns as select_gbif_fields(columns = 'standard')
collectorDictionary_file	
	Collector dictionary file - point to a file on your local disk or upload via git at https://raw.githubusercontent.com/pablopains/parseGBIF/main/collectorDictionary/CollectorsDictio
silence	if TRUE does not display progress messages

Details

If recordedBy is present in the collector's dictionary, it returns the checked name, if not, it returns the last name of the main collector, extracted from the recordedBy field. If recordedBy is present in the collector's dictionary, returns the main collector's last name associated with the single recordedBy key, otherwise, returns the main collector's last name, extracted from the recordedBy field. It is recommended to curate the main collector's surname, automatically extracted from the recordedBy field. The objective is to standardize the last name of the main collector. That the primary botanical collector of a sample is always recognized by the same last name, standardized in capital letters and non-ascii characters replaced

Value

Ctrl_nameRecordedBy_Standard, Ctrl_recordedBy, Ctrl_notes, collectorDictionary, Ctrl_update, collectorName, Ctrl_fullName, Ctrl_fullNameII, CVStarrVirtualHerbarium_PersonDetails

Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

See Also

collectors_get_name, generate_collection_event_key

Examples

```
help(collectors_prepare_dictionary)

occ <- prepare_gbif_occurrence_data(gbif_occurrece_file = 'https://raw.githubusercontent.com/pablopains/par
                                    columns = 'standard')

collectorsDictionaryFromDataset <- collectors_prepare_dictionary(occ = occ,
                                        collectorDictionary_file = 'https://raw.githubusercontent.com/pa

colnames(collectorsDictionaryFromDataset)
head(collectorsDictionaryFromDataset)

collectorDictionary_checked_file <- paste0(tempdir(),'/','collectorsDictionaryFromDataset.csv')

collectorDictionary_checked_file

write.csv(collectorsDictionaryFromDataset,
          collectorDictionary_checked_file,
          row.names = FALSE,
          fileEncoding = "UTF-8",
          na = "")
```

EnumOccurrenceIssue   *Enumeration GBIF issue An enumeration of validation rules for single*
                           occurrence records.

Description

There are many things that can go wrong and we continously encounter unexpected data. In order to help us and publishers improve the data, we flag records with various issues that we have encountered. This is also very useful for data consumers as you can include these issues as filters in occurrence searches. Not all issues indicate bad data. Some are merley flagging the fact that GBIF has altered values during processing. On the details page of any occurrence record you will see the list of issues in the notice at the bottom.

Usage

```
data(EnumOccurrenceIssue)
```

Format

A data frame with 69 rows and 9 columns

Details

constant GBIF issue constant

description GBIF issue description

definition Our definition for classifying geographic issues

type Type issue

priority Impact of the issue for the use of geospatial information

score Impact, in number, of the issue for the use of geospatial information

selection_score Value used to calculate the quality of the geospatial information according to the classification of the issue

reasoning Reasoning of the impact of the theme for the use of geospatial information

notes Notes

Source

- GBIF Infrastructure: Data processing
- An enumeration of validation rules for single occurrence records

---

export_data   *Export of results*

Description

For each unique collection event key, complete or incomplete, outputs will be created which combine information from duplicate records and generate a single unique collection event record to replace them. The main output fields relating to taxonomic identification and geographic coordinates:

- parseGBIF_sample_taxon_name = scientific name chosen as taxonomic identification for unique collection event
- parseGBIF_number_taxon_names = number of scientific names found in duplicates of unique collection event

- parseGBIF_sample_taxon_name_status = status of choice of 'identified', 'divergent identifications', 'unidentified'

- parseGBIF_unidentified_sample = if unique collection event has taxonomic identification

- parseGBIF_decimalLatitude = latitude in decimal degrees

- parseGBIF_decimalLongitude = longitude in decimal degrees

- parseGBIF_useful_for_spatial_analysis = whether the coordinates are useful for spatial analysis. **How is the taxon binomial attributed to the unique collection event selected?**

1. Where the unique collection event key is complete: The accepted TAXON_NAME selected is that which is most frequently applied to the duplicate vouchers at or below the rank of species. Where two named are applied with equal frequency then a mechanical approach, using alphabetical order, is applied, the first listed TAXON_NAME being chosen. Where there is no identification, at or below the rank of species, then the unique collection event, the unique collection event is indicated as unidentified.

2. Where the unique collection event key is incomplete: Where the unique collection event key is incomplete, then each record is treated as a unique collection event. If there is no identification, at or below the rank of species, then the unique collection event is classified as unidentified. __Geospatial information __ If the master voucher does not have geographic coordinates, we will seek coordinates from the duplicate records associated with it. Finally, the records are separated into three sets of data: **useable_data** - Where unique collection event with taxonomic identification and geographic coordinates are complete. This represents the useable dataset. **unusable_data** - Where unique collection event without taxonomic identification and/or geographic coordinates. **duplicates** The duplicates of unique collection events complete / incomplete

With this, it is possible to perform: Merge information between fields of duplicates of a unique collection event to create a synthetic record for each unique collection event, Compare the frequency of content in fields Generate a work package summary

For each complete unique collection event key, data fields that are empty in the digital voucher record will be populated with data from the respective duplicates. During content merging, we indicate fields associated with the description, location, and data of the unique collection event. By default, fields_to_merge parameter of export_data function contains:

- Ctrl_fieldNotes

- Ctrl_year

- Ctrl_stateProvince

- Ctrl_municipality

- Ctrl_locality

- Ctrl_countryCode

- Ctrl_eventDate

- Ctrl_habitat

- Ctrl_level0Name

- Ctrl_level1Name

- Ctrl_level2Name

- Ctrl_level3Name

Usage

```
export_data(
  occ_digital_voucher_file = "",
  occ_digital_voucher = NA,
  merge_unusable_data = FALSE,
  fields_to_merge = c("Ctrl_fieldNotes", "Ctrl_year", "Ctrl_stateProvince",
    "Ctrl_municipality", "Ctrl_locality", "Ctrl_countryCode", "Ctrl_eventDate",
    "Ctrl_habitat", "Ctrl_level0Name", "Ctrl_level1Name", "Ctrl_level2Name",
    "Ctrl_level3Name"),
  fields_to_compare = c("Ctrl_gbifID", "Ctrl_scientificName", "Ctrl_recordedBy",
    "Ctrl_recordNumber", "Ctrl_identifiedBy", "Ctrl_dateIdentified",
    "Ctrl_institutionCode", "Ctrl_collectionCode", "Ctrl_datasetName",
   "Ctrl_datasetName", "Ctrl_language", "wcvp_plant_name_id", "wcvp_taxon_rank",
    "wcvp_taxon_status", "wcvp_family", "wcvp_taxon_name", "wcvp_taxon_authors",
    "wcvp_reviewed", "wcvp_searchNotes"),
  fields_to_parse = c("Ctrl_gbifID", "Ctrl_bibliographicCitation", "Ctrl_language",
    "Ctrl_institutionCode", "Ctrl_collectionCode", "Ctrl_datasetName",
   "Ctrl_basisOfRecord", "Ctrl_catalogNumber", "Ctrl_recordNumber", "Ctrl_recordedBy",
   "Ctrl_occurrenceStatus", "Ctrl_eventDate", "Ctrl_year", "Ctrl_month", "Ctrl_day",
    "Ctrl_habitat", "Ctrl_fieldNotes", "Ctrl_eventRemarks", "Ctrl_countryCode",
    "Ctrl_stateProvince", "Ctrl_municipality", "Ctrl_county", "Ctrl_locality",
    "Ctrl_level0Name", "Ctrl_level1Name", "Ctrl_level2Name",
     "Ctrl_level3Name",
    "Ctrl_identifiedBy", "Ctrl_dateIdentified", "Ctrl_scientificName",
   "Ctrl_decimalLatitude", "Ctrl_decimalLongitude", "Ctrl_nameRecordedBy_Standard",
    "Ctrl_recordNumber_Standard", "Ctrl_key_family_recordedBy_recordNumber",
   "Ctrl_geospatial_quality", "Ctrl_verbatim_quality", "Ctrl_moreInformativeRecord",
   "Ctrl_coordinates_validated_by_gbif_issue", "wcvp_plant_name_id", "wcvp_taxon_rank",
    "wcvp_taxon_status", "wcvp_family", "wcvp_taxon_name", "wcvp_taxon_authors",
    "wcvp_reviewed",
     "wcvp_searchNotes", "parseGBIF_digital_voucher",
    "parseGBIF_duplicates", "parseGBIF_num_duplicates",
   "parseGBIF_non_groupable_duplicates", "parseGBIF_duplicates_grouping_status"),
  silence = TRUE
)
```

Arguments

occ_digital_voucher_file
              CSV fila result of function select_digital_voucher()$occ_digital_voucher

occ_digital_voucher
              data frame result of function select_digital_voucher()$occ_digital_voucher

merge_unusable_data
              include records unique collection events incomplete in merge processing

fields_to_merge
              fields to merge

fields_to_compare
              fields to compare content frequency

fields_to_parse
              all fields

silence   if TRUE does not display progress messages

Details

Each data frame should be used as needed

Value

list with 10 data frames

- **all_data** All records processed, merged Unique collection events complete / incomplete and their duplicates

- **useable_data_merge** Merged Unique collection events complete

- **useable_data_raw** Raw Unique collection events complete

- **duplicates** Duplicates of unique collection events complete / incomplete

- **unusable_data_merge** Merged Unique collection events incomplete, It is NA if merge_unusable_data is FALSE.

- **unusable_data_raw** Raw Unique collection events incomplete

- **parseGBIF_general_summary**

- **parseGBIF_merge_fields_summary**

- **parseGBIF_merge_fields_summary_useable_data**

- **parseGBIF_merge_fields_summary_unusable_data** It is NA if merge_unusable_data is FALSE

Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

See Also

batch_checkName_wcvp, extract_gbif_issue

Examples

```
help(export_data)

results <- export_data(occ_digital_voucher_file = file.occ_digital_voucher,
                       merge_unusable_data = TRUE)

names(results)

results$parseGBIF_general_summary
results$parseGBIF_merge_fields_summary
results$parseGBIF_merge_fields_summary_complete
NROW(results$all_data)
NROW(results$unique_collection_event_complete_merge)
NROW(results$unique_collection_event_incomplete_raw)
NROW(results$duplicates)
```

export_data_v2.3	*Export of results*

Description

For each unique collection event key, complete or incomplete, outputs will be created which combine information from duplicate records and generate a single unique collection event record to replace them. The main output fields relating to taxonomic identification and geographic coordinates:

- parseGBIF_sample_taxon_name = scientific name chosen as taxonomic identification for unique collection event
- parseGBIF_number_taxon_names = number of scientific names found in duplicates of unique collection event
- parseGBIF_sample_taxon_name_status = status of choice of 'identified', 'divergent identifications', 'unidentified'
- parseGBIF_unidentified_sample = if unique collection event has taxonomic identification
- parseGBIF_decimalLatitude = latitude in decimal degrees
- parseGBIF_decimalLongitude = longitude in decimal degrees
- parseGBIF_useful_for_spatial_analysis = whether the coordinates are useful for spatial analysis. **How is the taxon binomial attributed to the unique collection event selected?**

1. Where the unique collection event key is complete: The accepted TAXON_NAME selected is that which is most frequently applied to the duplicate vouchers at or below the rank of species. Where two named are applied with equal frequency then a mechanical approach, using alphabetical order, is applied, the first listed TAXON_NAME being chosen. Where there is no identification, at or below the rank of species, then the unique collection event, the unique collection event is indicated as unidentified.

2. Where the unique collection event key is incomplete: Where the unique collection event key is incomplete, then each record is treated as a unique collection event. If there is no identification, at or below the rank of species, then the unique collection event is classified as unidentified. __Geospatial information __ If the master voucher does not have geographic coordinates, we will seek coordinates from the duplicate records associated with it. Finally, the records are separated into three sets of data: **useable_data** - Where unique collection event with taxonomic identification and geographic coordinates are complete. This represents the useable dataset. **unusable_data** - Where unique collection event without taxonomic identification and/or geographic coordinates. **duplicates** The duplicates of unique collection events complete / incomplete

With this, it is possible to perform: Merge information between fields of duplicates of a unique collection event to create a synthetic record for each unique collection event, Compare the frequency of content in fields Generate a work package summary

For each complete unique collection event key, data fields that are empty in the digital voucher record will be populated with data from the respective duplicates. During content merging, we indicate fields associated with the description, location, and data of the unique collection event. By default, fields_to_merge parameter of export_data function contains:

- Ctrl_fieldNotes
- Ctrl_year

- Ctrl_stateProvince
- Ctrl_municipality
- Ctrl_locality
- Ctrl_countryCode
- Ctrl_eventDate
- Ctrl_habitat
- Ctrl_level0Name
- Ctrl_level1Name
- Ctrl_level2Name
- Ctrl_level3Name

Usage

```
export_data_v2.3(
  occ_digital_voucher_file = "",
  occ_digital_voucher = NA,
  merge_unusable_data = FALSE,
  fields_to_merge = c("Ctrl_fieldNotes", "Ctrl_year", "Ctrl_stateProvince",
    "Ctrl_municipality", "Ctrl_locality", "Ctrl_countryCode", "Ctrl_eventDate",
    "Ctrl_habitat", "Ctrl_level0Name", "Ctrl_level1Name", "Ctrl_level2Name",
    "Ctrl_level3Name"),
  fields_to_compare = c("Ctrl_gbifID", "Ctrl_scientificName", "Ctrl_recordedBy",
    "Ctrl_recordNumber", "Ctrl_identifiedBy", "Ctrl_dateIdentified",
    "Ctrl_institutionCode", "Ctrl_collectionCode", "Ctrl_datasetName",
   "Ctrl_datasetName", "Ctrl_language", "wcvp_plant_name_id", "wcvp_taxon_rank",
    "wcvp_taxon_status", "wcvp_family", "wcvp_taxon_name", "wcvp_taxon_authors",
    "wcvp_reviewed", "wcvp_searchNotes"),
  fields_to_parse = c("Ctrl_gbifID", "Ctrl_bibliographicCitation", "Ctrl_language",
    "Ctrl_institutionCode", "Ctrl_collectionCode", "Ctrl_datasetName",
   "Ctrl_basisOfRecord", "Ctrl_catalogNumber", "Ctrl_recordNumber", "Ctrl_recordedBy",
   "Ctrl_occurrenceStatus", "Ctrl_eventDate", "Ctrl_year", "Ctrl_month", "Ctrl_day",
    "Ctrl_habitat", "Ctrl_fieldNotes", "Ctrl_eventRemarks", "Ctrl_countryCode",
    "Ctrl_stateProvince", "Ctrl_municipality", "Ctrl_county", "Ctrl_locality",
    "Ctrl_level0Name", "Ctrl_level1Name", "Ctrl_level2Name",
     "Ctrl_level3Name",
    "Ctrl_identifiedBy", "Ctrl_dateIdentified", "Ctrl_scientificName",
   "Ctrl_decimalLatitude", "Ctrl_decimalLongitude", "Ctrl_nameRecordedBy_Standard",
    "Ctrl_recordNumber_Standard", "Ctrl_key_family_recordedBy_recordNumber",
   "Ctrl_geospatial_quality", "Ctrl_verbatim_quality", "Ctrl_moreInformativeRecord",
   "Ctrl_coordinates_validated_by_gbif_issue", "wcvp_plant_name_id", "wcvp_taxon_rank",
    "wcvp_taxon_status", "wcvp_family", "wcvp_taxon_name", "wcvp_taxon_authors",
    "wcvp_reviewed",
     "wcvp_searchNotes", "parseGBIF_digital_voucher",
    "parseGBIF_duplicates", "parseGBIF_num_duplicates",
   "parseGBIF_non_groupable_duplicates", "parseGBIF_duplicates_grouping_status",
    "parseGBIF_unidentified_sample", "parseGBIF_sample_taxon_name",
    "parseGBIF_sample_taxon_name_status", "parseGBIF_number_taxon_names",
    "parseGBIF_useful_for_spatial_analysis", "parseGBIF_decimalLatitude",
    "parseGBIF_decimalLongitude", "parseGBIF_dataset_result"),
  silence = TRUE
)
```

Arguments

`occ_digital_voucher_file`

> CSV fila result of function select_digital_voucher()$occ_digital_voucher

`occ_digital_voucher`

> data frame result of function select_digital_voucher()$occ_digital_voucher

`merge_unusable_data`

> include records unique collection events incomplete in merge processing

`fields_to_merge`

> fields to merge

`fields_to_compare`

> fields to compare content frequency

`fields_to_parse`

> all fields

`silence`   if TRUE does not display progress messages

Details

Each data frame should be used as needed

Value

list with 10 data frames

- **all_data** All records processed, merged Unique collection events complete / incomplete and their duplicates

- **useable_data_merge** Merged Unique collection events complete

- **useable_data_raw** Raw Unique collection events complete

- **duplicates** Duplicates of unique collection events complete / incomplete

- **unusable_data_merge** Merged Unique collection events incomplete, It is NA if merge_unusable_data is FALSE.

- **unusable_data_raw** Raw Unique collection events incomplete

- **parseGBIF_general_summary**

- **parseGBIF_merge_fields_summary**

- **parseGBIF_merge_fields_summary_useable_data**

- **parseGBIF_merge_fields_summary_unusable_data** It is NA if merge_unusable_data is FALSE

Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

See Also

batch_checkName_wcvp, extract_gbif_issue

Examples

```
help(export_data)

results <- export_data(occ_digital_voucher_file = file.occ_digital_voucher,
                       merge_unusable_data = TRUE)

names(results)

results$parseGBIF_general_summary
results$parseGBIF_merge_fields_summary
results$parseGBIF_merge_fields_summary_complete
NROW(results$all_data)
NROW(results$unique_collection_event_complete_merge)
NROW(results$unique_collection_event_incomplete_raw)
NROW(results$duplicates)
```

---

extract_gbif_issue   *Extracting GBIF issues*

Description

Extract GBIF validation rules for occurrence records

GBIF recognises and documents several issues relating to the data fields for an individual record. The issue field stores terms that represent an enumeration of GBIF validation rules. Issues can lead to errors or unexpected data. The issues fields are therefore a valuable source of information when assessing the quality of a record. In order to help GBIF and the data publishers improve the data, GBIF flag records with various issues that they have encountered. These issues can be used as filters applied to occurrence searches. Not all issues indicate bad data, some flagthe fact that GBIF has altered values during processing. The values of EnumOccurrenceIssue will be used by the function extract_gbif_issue as a model to tabulate the GBIF issues of each record, individualizing them, in columns.TRUE or FALSE, flagging whether the issue applies or not for each record.

Usage

```
extract_gbif_issue(occ = NA, enumOccurrenceIssue = NA)
```

Arguments

occ   GBIF occurrence table with selected columns as select_gbif_fields(columns = 'standard')

enumOccurrenceIssue
                    An enumeration of validation rules for single occurrence records by GBIF file, if NA, will be used, data(EnumOccurrenceIssue)

Details

https://gbif.github.io/parsers/apidocs/org/gbif/api/vocabulary/OccurrenceIssue.html

Value

list with two data frames: summary, with the frequency of issues in the records and occ_gbif_issue, with issues in columns with TRUE or FALSE for each record.

Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

See Also

prepare_gbif_occurrence_data, select_gbif_fields

Examples

```
library(ParsGBIF)

help(extract_gbif_issue)

occ_file <- 'https://raw.githubusercontent.com/pablopains/ParsGBIF/main/dataGBIF/Achatocarpaceae/occurrence

occ <- prepare_gbif_occurrence_data(gbif_occurrece_file = occ_file,
                                    columns = 'standard')

data(EnumOccurrenceIssue)

colnames(EnumOccurrenceIssue)

head(EnumOccurrenceIssue)

occ_gbif_issue <- extract_gbif_issue(occ = occ)

names(occ_gbif_issue)

head(occ_gbif_issue$summary)

colnames(occ_gbif_issue$occ_gbif_issue)

head(occ_gbif_issue$occ_gbif_issue)
```

---

generate_collection_event_key
Generating the collection event key

Description

This generates a key to identify the physical and digital duplicates, of a given collection event. It combines the primary collector's surname, the collector's number and the botanical family, a key is created (family + recordByStandardized + recordNumber_Standard) that allows grouping the duplicates of the same unique collection event.

It also identifiesnew collectors to be added to the collector dictionary and that can be reused in the future.

Include recordedByStandardized field with verified main collector's last name. Include recordNumber_Standard field with only numbers from recordNumber. Create the collection event key to group duplicates in the key_family_recordedBy_recordNumber field, composed of the fields: family + recordedByStandardized + recordNumber_Standard.

Usage

```
generate_collection_event_key(
  occ = NA,
  collectorDictionary_checked_file = NA,
  collectorDictionary_checked = NA,
  collectorDictionary_file =
   "https://raw.githubusercontent.com/pablopains/parseGBIF/main/collectorDictionary/CollectorsDi
  silence = TRUE
)
```

Arguments

occ	GBIF occurrence table with selected columns as select_gbif_fields(columns = 'standard')
collectorDictionary_checked_file	
	Verified collector dictionary file - point to a file on your local disk (use file or data frame)
collectorDictionary_checked	
	Verified collector dictionary data frame (use file or data frame)
collectorDictionary_file	
	Collector dictionary file - point to a file on your local disk or upload via git at https://raw.githubusercontent.com/pablopains/parseGBIF/main/collectorDictionary/CollectorsDictio
silence	if TRUE does not display progress messages

Details

Fields created for each incident record: nameRecordedBy_Standard, recordNumber_Standard, key_family_recordedBy_ key_year_recordedBy_recordNumber

Value

list with three data frames: occ_collectorsDictionary, with update result fields only, summary and CollectorsDictionary_add, with new collectors that can be added to the collector dictionary that can be reused in the future.

Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

See Also

collectors_get_name, prepare_collectorsDictionary

Examples

```
collectorsDictionaryFromDataset <- prepare_lastNameRecordedBy(occ=occ,
                                        collectorDictionary_checked_file='collectorDictionary_checked.csv

names(collectorsDictionaryFromDataset)
```

---

parseGBIF_summary   *Selecting the master digital voucher*

Description

...

Usage

```
parseGBIF_summary(
  parseGBIF_all_data = NA,
  file.parseGBIF_all_data = "",
  fields_to_merge = c("Ctrl_fieldNotes", "Ctrl_year", "Ctrl_stateProvince",
    "Ctrl_municipality", "Ctrl_locality", "Ctrl_countryCode", "Ctrl_eventDate",
    "Ctrl_habitat", "Ctrl_level0Name", "Ctrl_level1Name", "Ctrl_level2Name",
    "Ctrl_level3Name")
)
```

Arguments

parseGBIF_all_data
                  ...
file.parseGBIF_all_data
                  ...
fields_to_merge
                  fields to merge
fields_to_compare
                  fields to compare content frequency
fields_to_parse
                  all fields
silence   if TRUE does not display progress messages

Details

...

Value

- **parseGBIF_general_summary**
- **parseGBIF_merge_fields_summary**
- **parseGBIF_merge_fields_summary_useable_data**
- **parseGBIF_merge_fields_summary_unusable_data**

Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

See Also

[batch_checkName_wcvp](), [extract_gbif_issue]()

Examples

```
results <- export_data_v2.1(occ_digital_voucher_file = '',
occ_digital_voucher = occ_digital$all_data,
merge_unusable_data = TRUE,
silence = FALSE)

names(results)

head(results$occ_all)
colnames(results$occ_all)
```

---

prepare_gbif_occurrence_data
                              Preparing occurrence data downloaded from GBIF for use by parseG-
                              BIF

Description

Prepare occurrence data downloaded from GBIF to be used by ParsGBIF functions

Usage

```
prepare_gbif_occurrence_data(gbif_occurrece_file = "", columns = "standard")
```

Arguments

gbif_occurrece_file

                 The name of the file from which the with occurrence data downloaded from GBIF (by default "occurrence.txt")

columns   Character vector of strings to indicate column names of the GBIF occurrence file. Use 'standard' to select basic columns for use in the package, 'all' to select all available columns. The default is 'standard'

Details

Select data fields and rename field names prefixed with "Ctrl_"

Value

data.frame with renamed selected fields with prefix "Ctrl_"

Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

See Also

select_gbif_fields, extract_gbif_issue

Examples

```
library(ParsGBIF)

help(prepare_gbif_occurrence_data)

occ_file <- 'https://raw.githubusercontent.com/pablopains/ParsGBIF/main/dataGBIF/Achatocarpaceae/occurrence

occ <- prepare_gbif_occurrence_data(gbif_occurrece_file = occ_file,
                                    columns = 'standard')

colnames(occ)

head(occ)
```

---

select_digital_voucher

Selecting the master digital voucher

Description

To group duplicates and choose the digital voucher: Unique collection events can result in many 'duplicate' GBIF records. We designate one of these 'duplicate' records as the master digital voucher, to which data from other duplicate vouchers can be merged (see export_data):

Where the collection event key for grouping duplicates is complete, then duplicates can be grouped / parsed. To do so, we evaluate record completeness. Record completeness is calculated based on data-quality scores for the information in the following fields: recordedBy, recordNumber, year, institutionCode, catalogNumber, locality, municipality, countryCode, stateProvince and fieldNotes. The spatial coordinates associated with each duplicate are ranked using a score for the quality of the geospatial information. This score is calculated using the issues listed in the GBIF table, EnumOccurrenceIssue. A score is calculated based on these issues (see above). The duplicate with the highest total score is assigned as the master voucher for the unique collection event. Missing information contained in duplicate records of the unique collection event can then be merged into the master digital voucher (see export_data).

Where the collection event key is incomplete, unique collection event duplicates cannot be parsed. In this case, each record is considered as a unique collection event, without duplicates. However, to know the integrity of the information, record completeness and quality of the geospatial information, are evaluated as described above.

How is the quality score calculated? parseGBIF_digital_voucher = The duplicate with the highest total score, sum of record completeness + quality of geospatial information.

How is record completeness calculated? The quality of the duplicate records associated with each collection event key is measured as the completeness of a record, using the sum of a number of flags (see below) equal to TRUE.

Flags used to calculate record completeness

- Is there information about the collector?
- Is there information about the collection number?
- Is there information about the year of collection?
- Is there information about the institution code?
- Is there information about the catalog number?
- Is there information about the locality?
- Is there information about the municipality of collection?
- Is there information about the state/province of collection?
- Is there information about the field notes?

The quality of geospatial information is based on geographic issues raised by GBIF. GIBF issues relating to geospatial data were classified into three classes based on the data quality scores that we assigned to each of the following GBIF issues recorded in the EnumOccurrenceIssue.

- Issue does not affect coordinating accuracy, with selection_score equal to -1
- Issue has potential to affect coordinate accuracy, with selection_score equal to -3
- Records with a selection_score equal to -9 are excluded.

Usage

```
select_digital_voucher(
  occ = NA,
  occ_gbif_issue = NA,
  occ_wcvp_check_name = NA,
  occ_collectorsDictionary = NA,
  enumOccurrenceIssue = NA,
  silence = TRUE
)
```

Arguments

occ	GBIF occurrence table with selected columns as select_gbif_fields(columns = 'standard')
occ_gbif_issue	= result of function extract_gbif_issue()$occ_gbif_issue
occ_wcvp_check_name	
	= result of function batch_checkName_wcvp()$occ_wcvp_check_name
occ_collectorsDictionary	
	= result of function update_collectorsDictionary()$occ_collectorsDictionary
enumOccurrenceIssue	
	An enumeration of validation rules for single occurrence records by GBIF file, if NA, will be used, data(EnumOccurrenceIssue)
silence	if TRUE does not display progress messages

Details

- parseGBIF_duplicates_grouping_status - "groupable", "not groupable: no recordedBy and no recordNumber", "not groupable: no recordNumber" or "not groupable: no recordedBy"
- parseGBIF_num_duplicates number of duplicates records
- parseGBIF_duplicates TRUE/FALSE
- parseGBIF_non_groupable_duplicates TRUE/FALSE

Value

list with two data frames: occ_digital voucher_and: occ_digital_voucher, with all data processing fields and occ_results, only result fields.

Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

See Also

[batch_checkName_wcvp](), [extract_gbif_issue]()

Examples

```
help(select_digital_voucher)

head(occ)
head(res_gbif_issue$occ_gbif_issue)
head(res_checkName_wcvp$occ_wcvp_check_name)
head(res_collectorsDictionary$occ_collectorsDictionary)
res_digital_voucher_and_sample_identification <- select_digital_voucher(occ = occ,
                                        occ_gbif_issue = res_gbif_issue$occ_gbif_issue,
                                        occ_wcvp_check_name = res_checkName_wcvp$occ_wcvp_check_nam
                                        occ_collectorsDictionary = res_collectorsDictionary$occ_col
                                        enumOccurrenceIssue = EnumOccurrenceIssue)

names(res_digital_voucher_and_sample_identification)

head(res_digital_voucher_and_sample_identification$occ_digital_voucher)
colnames(res_digital_voucher_and_sample_identification$occ_digital_voucher)
```

---

select_digital_voucher_v2.1

Selecting the master digital voucher

Description

To group duplicates and choose the digital voucher: Unique collection events can result in many 'duplicate' GBIF records. We designate one of these 'duplicate' records as the master digital voucher, to which data from other duplicate vouchers can be merged (see export_data):

Where the collection event key for grouping duplicates is complete, then duplicates can be grouped / parsed. To do so, we evaluate record completeness. Record completeness is calculated

based on data-quality scores for the information in the following fields: recordedBy, recordNumber, year, institutionCode, catalogNumber, locality, municipality, countryCode, stateProvince and fieldNotes. The spatial coordinates associated with each duplicate are ranked using a score for the quality of the geospatial information. This score is calculated using the issues listed in the GBIF table, EnumOccurrenceIssue. A score is calculated based on these issues (see above). The duplicate with the highest total score is assigned as the master voucher for the unique collection event. Missing information contained in duplicate records of the unique collection event can then be merged into the master digital voucher (see export_data).

Where the collection event key is incomplete, unique collection event duplicates cannot be parsed. In this case, each record is considered as a unique collection event, without duplicates. However, to know the integrity of the information, record completeness and quality of the geospatial information, are evaluated as described above.

How is the quality score calculated? parseGBIF_digital_voucher = The duplicate with the highest total score, sum of record completeness + quality of geospatial information.

How is record completeness calculated? The quality of the duplicate records associated with each collection event key is measured as the completeness of a record, using the sum of a number of flags (see below) equal to TRUE.

Flags used to calculate record completeness

- Is there information about the collector?
- Is there information about the collection number?
- Is there information about the year of collection?
- Is there information about the institution code?
- Is there information about the catalog number?
- Is there information about the locality?
- Is there information about the municipality of collection?
- Is there information about the state/province of collection?
- Is there information about the field notes?

The quality of geospatial information is based on geographic issues raised by GBIF. GIBF issues relating to geospatial data were classified into three classes based on the data quality scores that we assigned to each of the following GBIF issues recorded in the EnumOccurrenceIssue.

- Issue does not affect coordinating accuracy, with selection_score equal to -1
- Issue has potential to affect coordinate accuracy, with selection_score equal to -3
- Records with a selection_score equal to -9 are excluded.

Usage

```
select_digital_voucher_v2.1(
  occ = NA,
  occ_gbif_issue = NA,
  occ_wcvp_check_name = NA,
  occ_collectorsDictionary = NA,
  enumOccurrenceIssue = NA,
  silence = TRUE
)
```

Arguments

`occ`	GBIF occurrence table with selected columns as select_gbif_fields(columns = 'standard')
`occ_gbif_issue`	= result of function extract_gbif_issue()$occ_gbif_issue
`occ_wcvp_check_name`	
	= result of function batch_checkName_wcvp()$occ_wcvp_check_name
`occ_collectorsDictionary`	
	= result of function update_collectorsDictionary()$occ_collectorsDictionary
`enumOccurrenceIssue`	
	An enumeration of validation rules for single occurrence records by GBIF file, if NA, will be used, data(EnumOccurrenceIssue)
`silence`	if TRUE does not display progress messages

Details

- parseGBIF_duplicates_grouping_status - "groupable", "not groupable: no recordedBy and no recordNumber", "not groupable: no recordNumber" or "not groupable: no recordedBy"

- parseGBIF_num_duplicates number of duplicates records

- parseGBIF_duplicates TRUE/FALSE

- parseGBIF_non_groupable_duplicates TRUE/FALSE

Value

list with two data frames: occ_digital voucher_and: occ_digital_voucher, with all data processing fields and occ_results, only result fields.

Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

See Also

[batch_checkName_wcvp](), [extract_gbif_issue]()

Examples

```
help(select_digital_voucher)

head(occ)
head(res_gbif_issue$occ_gbif_issue)
head(res_checkName_wcvp$occ_wcvp_check_name)
head(res_collectorsDictionary$occ_collectorsDictionary)
res_digital_voucher_and_sample_identification <- select_digital_voucher(occ = occ,
                                        occ_gbif_issue = res_gbif_issue$occ_gbif_issue,
                                        occ_wcvp_check_name = res_checkName_wcvp$occ_wcvp_check_nam
                                        occ_collectorsDictionary = res_collectorsDictionary$occ_col
                                        enumOccurrenceIssue = EnumOccurrenceIssue)

names(res_digital_voucher_and_sample_identification)

head(res_digital_voucher_and_sample_identification$occ_digital_voucher)
colnames(res_digital_voucher_and_sample_identification$occ_digital_voucher)
```

---

select_gbif_fields   *select_gbif_fields*

Description

Select columns in GBIF occurrence data

Usage

```
select_gbif_fields(columns = "standard")
```

Arguments

columns   'standard' basic columns about what, when, where, and who collected, 'all' all
                 available columns or list column names

Details

"standard" : indicated by **(standard)**

or

'all':

- 'gbifID' **(standard)**
- 'abstract'
- 'accessRights'
- 'accrualMethod'
- 'accrualPeriodicity'
- 'accrualPolicy'
- 'alternative'
- 'audience'
- 'available'
- 'bibliographicCitation' **(standard)**
- 'conformsTo'
- 'contributor'
- 'coverage'
- 'created'
- 'creator'
- 'date'
- 'dateAccepted'
- 'dateCopyrighted'
- 'dateSubmitted'
- 'description'

- 'educationLevel'
- 'extent'
- 'format'
- 'hasFormat'
- 'hasPart'
- 'hasVersion'
- 'identifier'
- 'instructionalMethod'
- 'isFormatOf'
- 'isPartOf'
- 'isReferencedBy'
- 'isReplacedBy'
- 'isRequiredBy'
- 'isVersionOf'
- 'issued'
- 'language' **(standard)**
- 'license'
- 'mediator'
- 'medium'
- 'modified'
- 'provenance'
- 'publisher'
- 'references'
- 'relation'
- 'replaces'
- 'requires'
- 'rights'
- 'rightsHolder'
- 'source'
- 'spatial'
- 'subject'
- 'tableOfContents'
- 'temporal'
- 'title'
- 'type'
- 'valid'
- 'institutionID'
- 'collectionID'
- 'datasetID'
- 'institutionCode' **(standard)**

- 'collectionCode' **(standard)**
- 'datasetName' **(standard)**
- 'ownerInstitutionCode'
- 'basisOfRecord' **(standard)**
- 'informationWithheld' **(standard)**
- 'dataGeneralizations' **(standard)**
- 'dynamicProperties'
- 'occurrenceID' **(standard)** # occ_search(occurrenceId='BRA:UNEMAT:HPAN:6089')
- 'catalogNumber' **(standard)**
- 'recordNumber' **(standard)**
- 'recordedBy' **(standard)**
- 'recordedByID'
- 'individualCount'
- 'organismQuantity'
- 'organismQuantityType'
- 'sex'
- 'lifeStage'
- 'reproductiveCondition'
- 'behavior'
- 'establishmentMeans'
- 'degreeOfEstablishment'
- 'pathway'
- 'georeferenceVerificationStatus' **(standard)**
- 'occurrenceStatus' **(standard)**
- 'preparations'
- 'disposition'
- 'associatedOccurrences'
- 'associatedReferences'
- 'associatedSequences'
- 'associatedTaxa'
- 'otherCatalogNumbers'
- 'occurrenceRemarks'
- 'organismID'
- 'organismName'
- 'organismScope'
- 'associatedOrganisms'
- 'previousIdentifications'
- 'organismRemarks'
- 'materialSampleID'
- 'eventID'

- 'parentEventID'
- 'fieldNumber'
- 'eventDate' **(standard)**
- 'eventTime'
- 'startDayOfYear'
- 'endDayOfYear'
- 'year' **(standard)**
- 'month' **(standard)**
- 'day' **(standard)**
- 'verbatimEventDate'
- 'habitat' **(standard)**
- 'samplingProtocol'
- 'sampleSizeValue'
- 'sampleSizeUnit'
- 'samplingEffort'
- 'fieldNotes' **(standard)**
- 'eventRemarks' **(standard)**
- 'locationID' **(standard)**
- 'higherGeographyID'
- 'higherGeography' **(standard)**
- 'continent'
- 'waterBody'
- 'islandGroup' **(standard)**
- 'island' **(standard)**
- 'countryCode' **(standard)**
- 'stateProvince' **(standard)**
- 'county' **(standard)**
- 'municipality' **(standard)**
- 'locality' **(standard)**
- 'verbatimLocality' **(standard)**
- 'verbatimElevation'
- 'verticalDatum'
- 'verbatimDepth'
- 'minimumDistanceAboveSurfaceInMeters'
- 'maximumDistanceAboveSurfaceInMeters'
- 'locationAccordingTo'
- 'locationRemarks' **(standard)**
- 'decimalLatitude' **(standard)**
- 'decimalLongitude' **(standard)**
- 'coordinateUncertaintyInMeters'

- 'coordinatePrecision'
- 'pointRadiusSpatialFit'
- 'verbatimCoordinateSystem' **(standard)**
- 'verbatimSRS'
- 'footprintWKT'
- 'footprintSRS'
- 'footprintSpatialFit'
- 'georeferencedBy'
- 'georeferencedDate'
- 'georeferenceProtocol'
- 'georeferenceSources'
- 'georeferenceRemarks'
- 'geologicalContextID'
- 'earliestEonOrLowestEonothem'
- 'latestEonOrHighestEonothem'
- 'earliestEraOrLowestErathem'
- 'latestEraOrHighestErathem'
- 'earliestPeriodOrLowestSystem'
- 'latestPeriodOrHighestSystem'
- 'earliestEpochOrLowestSeries'
- 'latestEpochOrHighestSeries'
- 'earliestAgeOrLowestStage'
- 'latestAgeOrHighestStage'
- 'lowestBiostratigraphicZone'
- 'highestBiostratigraphicZone'
- 'lithostratigraphicTerms'
- 'group'
- 'formation'
- 'member'
- 'bed'
- 'identificationID'
- 'verbatimIdentification' **(standard)**
- 'identificationQualifier' **(standard)**
- 'typeStatus' **(standard)**
- 'identifiedBy' **(standard)**
- 'identifiedByID'
- 'dateIdentified' **(standard)**
- 'identificationReferences'
- 'identificationVerificationStatus'
- 'identificationRemarks'

- 'taxonID'
- 'scientificNameID'
- 'acceptedNameUsageID'
- 'parentNameUsageID'
- 'originalNameUsageID'
- 'nameAccordingToID'
- 'namePublishedInID'
- 'taxonConceptID'
- 'scientificName' **(standard)**
- 'acceptedNameUsage'
- 'parentNameUsage'
- 'originalNameUsage'
- 'nameAccordingTo'
- 'namePublishedIn'
- 'namePublishedInYear'
- 'higherClassification'
- 'kingdom'
- 'phylum'
- 'class'
- 'order'
- 'family' **(standard)**
- 'subfamily'
- 'genus'
- 'genericName'
- 'subgenus'
- 'infragenericEpithet'
- 'specificEpithet'
- 'infraspecificEpithet'
- 'cultivarEpithet'
- 'taxonRank' **(standard)**
- 'verbatimTaxonRank'
- 'vernacularName'
- 'nomenclaturalCode' **(standard)**
- 'taxonomicStatus' **(standard)**
- 'nomenclaturalStatus'
- 'taxonRemarks'
- 'datasetKey'
- 'publishingCountry'
- 'lastInterpreted'
- 'elevation'

- 'elevationAccuracy'
- 'depth'
- 'depthAccuracy'
- 'distanceAboveSurface'
- 'distanceAboveSurfaceAccuracy'
- 'issue' **(standard)**
- 'mediaType' **(standard)**
- 'hasCoordinate' **(standard)**
- 'hasGeospatialIssues' **(standard)**
- 'taxonKey'
- 'acceptedTaxonKey'
- 'kingdomKey'
- 'phylumKey'
- 'classKey'
- 'orderKey'
- 'familyKey'
- 'genusKey'
- 'subgenusKey'
- 'speciesKey'
- 'species'
- 'acceptedScientificName'
- 'verbatimScientificName' **(standard)**
- 'typifiedName'
- 'protocol'
- 'lastParsed'
- 'lastCrawled'
- 'repatriated'
- 'relativeOrganismQuantity'
- 'level0Gid'
- 'level0Name' **(standard)**
- 'level1Gid'
- 'level1Name' **(standard)**
- 'level2Gid'
- 'level2Name' **(standard)**
- 'level3Gid'
- 'level3Name' **(standard)**
- 'iucnRedListCategory'

Value

list of the columns names

Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

See Also

[extract_gbif_issue](), [prepare_gbif_occurrence_data]()

Examples

```
# select_gbif_fields()

help(select_gbif_fields)

col_sel <- select_gbif_fields(columns = 'all')

col_sel <- select_gbif_fields(columns = 'standard')
```

---

standardize_scientificName

standardize_scientificName

Description

standardize binomial name, variety, subspecies, form and hybrids, authorship to allow comparison with names of taxa in the World Checklist of Vascular Plants (WCVP) database

Usage

```
standardize_scientificName(
  searchedName = "Alomia angustata (Gardner) Benth. ex Baker"
)
```

Arguments

searchedName   scientific name, with or without author

Details

Standardize scientific name according to WCVP format. Separate generic epithet, specific epithet, variety, subspecies, form, hybrid and author, in the scientific name, if any. Standardize, according to WCVP, abbreviation of infrataxon, if any: variety to var., subspecies to subsp., FORM to f., hybrid separator separate x from the specific epithet.

Value

searchedName, standardizeName, taxonAuthors, taxonAuthors_last

Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

See Also

get_wcvp, checkName_wcvp

Examples

```
standardize_scientificName('Leucanthemum ×superbum (Bergmans ex J.W.Ingram) D.H.Kent')
standardize_scientificName('Alomia angustata (Gardner) Benth. ex Baker')
standardize_scientificName('Centaurea ×aemiliae Font Quer')
```

---

wcvp_check_name	*Check species names against World Checklist of Vascular Plants (WCVP) database*

Description

Use the World Checklist of Vascular Plants WCVP database to check accepted names and update synonyms.

The World Checklist of Vascular Plants (WCVP) database is available from the Royal Botanic Gardens, Kew. It can be downloaded to a folder of the user's choice or into memory using the get_wcvp function. The output has 33 columns.

Usage

```
wcvp_check_name(
  searchedName = "Hemistylus brasiliensis Wedd.",
  wcvp_names = "",
  if_author_fails_try_without_combinations = TRUE
)
```

Arguments

searchedName	scientific name, with or without author
wcvp_names	WCVP table, wcvp_names.csv file from http://sftp.kew.org/pub/data-repositories/WCVP/ If NA, automatically load the latest version of the database by the function parseGBIF::wcvp_get_data(read_only_to_memory = TRUE)$wcvp_names.
if_author_fails_try_without_combinations	
	option for partial verification of the authorship of the species. Remove the authors of combinations, in parentheses

Details

About the World Checklist of Vascular Plants https://powo.science.kew.org/about-wcvp searchNotes values:

- Accepted - When only one authorless scientific name is present in the list of TAXON_name with and TAXON_STATUS equal to "Accepted", verified_speciesName = 100.
- Accepted among homonyms - When more than one authorless scientific name is present in the TAXON_name list, but only one of the homonyms displays TAXON_STATUS equal to "Accepted", verified_speciesName = number of matches/100.

- Homonyms - When more than one authorless scientific name is present in the TAXON_name list and more than one, or none among the homonyms, display TAXON_STATUS equal to "Accepted", verified_speciesName = number of matches/100. Before searching for homonyms, there was a failure in trying to find the matching match between authorless scientific name in TAXON_name and author in TAXON_AUTHORS, in these cases verified_author equal to 0 (zero),

- Not Found: When the authorless scientific name is not present in the TAXON_NAME LIST

- Unplaced: o When only one authorless scientific name is present in the list of TAXON_name with and TAXON_STATUS = "Unplaced"

- Updated: When only one authorless scientific name is present in the list of TAXON_name and ACCEPTED_PLANT_NAME_ID are not empty (and ACCEPTED_PLANT_NAME_ID is different from the ID of the species consulted) taxon_status_of_searchedName, plant_name_id_of_searchedName and taxon_authors_of_searchedName values:

  – When searchNotes equals "Updated" – The fields record the information of the scientific name originally consulted.

  – When searchNotes equals "Homonyms" - Fields record the information of homonymous synonyms separated by "|".

- verified_author values:

  – When value equal to 100 – when there is matched match between authorless scientific name in TAXON_name and author in TAXON_AUTHORS.

  – When value equal to 50 – when there is combined correspondence between authorless scientific name in TAXON_name and author, without (combination), in TAXON_AUTHORS.

  – When value equal to 0 – regardless of the correspondence between authorless scientific name in TAXON_name, author is not present in TAXON_AUTHORS.

Value

Data frame with WCVP fields

Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

See Also

wcvp_check_name_batch, wcvp_get_data

Examples

```
# These examples take >10 seconds to run and require 'parseGBIF::wcvp_get_data()'

library(parseGBIF)

help(wcvp_check_name)

wcvp_names <- wcvp_get_data(read_only_to_memory = TRUE)$wcvp_names

# 1) Updated
wcvp_check_name(searchedName = 'Hemistylus brasiliensis Wedd.',
                wcvp_names = wcvp_names,
                if_author_fails_try_without_combinations = TRUE)
```

```
# 2) Accepted
wcvp_check_name(searchedName = 'Hemistylus boehmerioides Wedd. ex Warm.',
                wcvp_names = wcvp_names,
                if_author_fails_try_without_combinations = TRUE)

# 3) Unplaced - taxon_status = Unplaced
wcvp_check_name(searchedName = 'Leucosyke australis Unruh',
                wcvp_names = wcvp_names,
                if_author_fails_try_without_combinations = TRUE)

# 4) Accepted among homonyms - When author is not informed. In this case, one of the homonyms, taxon_status is ac
wcvp_check_name(searchedName = 'Parietaria cretica',
                wcvp_names = wcvp_names,
                if_author_fails_try_without_combinations = TRUE)

# When author is informed
wcvp_check_name(searchedName = 'Parietaria cretica L.',
                wcvp_names = wcvp_names,
                if_author_fails_try_without_combinations = TRUE)

# When author is informed
wcvp_check_name(searchedName = 'Parietaria cretica Moris',
                wcvp_names = wcvp_names,
                if_author_fails_try_without_combinations = TRUE)

# 5) Homonyms - When author is not informed. In this case, none of the homonyms, taxon_status is Accepted
wcvp_check_name(searchedName = 'Laportea peltata',
                wcvp_names = wcvp_names,
                if_author_fails_try_without_combinations = TRUE)

# When author is informed
wcvp_check_name(searchedName = 'Laportea peltata Gaudich. & Decne.',
                wcvp_names = wcvp_names,
                if_author_fails_try_without_combinations = TRUE)

# When author is informed
wcvp_check_name(searchedName = 'Laportea peltata (Blume) Gaudich.',
                wcvp_names = wcvp_names,
                if_author_fails_try_without_combinations = TRUE)
```

wcvp_check_name_batch	*In batch, use the WCVP database to check accepted names and update synonyms*

Description

Species' names can be checked against WCVP database one by one, or in a batch mode. To verify individual names, the function wcvp_check_name is used.

Usage

```
wcvp_check_name_batch(
```

```
    occ = NA,
    wcvp_names = "",
    if_author_fails_try_without_combinations = TRUE,
    wcvp_selected_fields = "standard",
    silence = TRUE
)
```

Arguments

occ	GBIF occurrence table with selected columns as select_gbif_fields(columns = 'standard')
wcvp_names	get data frame in parseGBIF::wcvp_get_data(read_only_to_memory = TRUE)$wcvp_names or configure function to save a copy on local disk to optimize loading, see details in help(wcvp_get_data)
if_author_fails_try_without_combinations	
	option for partial verification of the authorship of the species. Remove the authors of combinations, in parentheses.
wcvp_selected_fields	
	WCVP fields selected as return, 'standard' basic columns, 'all' all available columns. The default is 'standard'
silence	if TRUE does not display progress messages

Details

See help(checkName_wcvp)

- about WCVP database
- World Checklist of Vascular Plants
- WCVP database
- (about WCVP)

Value

list with two data frames: summary, species list and occ_wcvp_check_name, with WCVP fields

Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

See Also

wcvp_get_data, wcvp_check_name

Examples

```
# These examples take >10 minutes to run and require 'parseGBIF::wcvp_get_data()'


library(parseGBIF)

help(wcvp_check_name_batch)

occ_file <- 'https://raw.githubusercontent.com/pablopains/parseGBIF/main/dataGBIF/Achatocarpaceae/occurrenc
```

```
occ <- prepare_gbif_occurrence_data(gbif_occurrece_file = occ_file,
                                    columns = 'standard')

# wcvp_names <- wcvp_get_data(read_only_to_memory = TRUE)$wcvp_names
data(wcvp_names_Achatocarpaceae)

head(wcvp_names)

res_wcvp_check_name_batch <- wcvp_check_name_batch(occ = occ,
                                          wcvp_names =  wcvp_names,
                              if_author_fails_try_without_combinations = TRUE,
                                          wcvp_selected_fields = 'standard',
                                          show_process = TRUE)

names(res_wcvp_check_name_batch)

head(res_wcvp_check_name_batch$summary)

head(res_wcvp_check_name_batch$occ_wcvp_check_name)
```

---

wcvp_get_data   *Get WCVP database*

Description

Download World Checklist of Vascular Plants (WCVP) database

Usage

```
wcvp_get_data(
  url_source = "http://sftp.kew.org/pub/data-repositories/WCVP/",
  read_only_to_memory = FALSE,
  path_results = "C:/parseGBIF",
  update = FALSE,
  load_distribution = FALSE
)
```

Arguments

url_source   http://sftp.kew.org/pub/data-repositories/WCVP/

read_only_to_memory
                    TRUE to in-memory read-only, not writing a copy to local disk

path_results   download destination folder, if read_only_to_memory FALSE

update   TRUE to update and load files, FALSE to keep local version and load files, if
                    read_only_to_memory FALSE

load_distribution
                    TRUE to load file with geographical distribution of species, if read_only_to_memory
                    FALSE

Details

http://sftp.kew.org/pub/data-repositories/WCVP/ This is the public SFTP (Secure File Transfer Protocol) site of the Royal Botanic Gardens, Kew. This space contains data resources publicly accessible to the user 'anonymous'. No password required for access. Use of data made available via this site may be subject to legal and licensing restrictions. The README in the top-level directory for each data resource provides specific information about its terms of use.

Value

list with two data frames: wcvp_names and wcvp_distribution

Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

See Also

[wcvp_check_name](#), [wcvp_check_name_batch](#)

Examples

```
# load package
library(parseGBIF)

help(wcvp_get_data)

# Download wcvp database to local disk
path_data <- tempdir() # you can change this folder

wcvp <- wcvp_get_data(url_source = 'http://sftp.kew.org/pub/data-repositories/WCVP/',
                read_only_to_memory = FALSE,
                path_results = path_data,
                update = FALSE,
                load_distribution = TRUE)

names(wcvp)

head(wcvp$wcvp_names)
colnames(wcvp$wcvp_names)

head(wcvp$wcvp_distribution)
colnames(wcvp$wcvp_distribution)
```

Index