

# Package ‘parseGBIF’

July 11, 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,  
rnatualearthdata,  
jsonlite,  
sqldf,  
DT,  
downloader,  
tidyselect,  
utils

**Remotes** github::pablopains/parseGBIF

**Depends** R (>= 3.5.0)

R topics documented:

|   |           |
|---|-----------|
| collectors_get_name . . . . .           | 2         |
| collectors_prepare_dictionary . . . . . | 3         |
| collectors_update_dictionary . . . . .  | 4         |
| EnumOccurrenceIssue . . . . .           | 5         |
| export_data . . . . .                   | 6         |
| extract_gbif_issue . . . . .            | 8         |
| prepare_gbif_occurrence_data . . . . .  | 9         |
| select_digital_voucher . . . . .        | 10        |
| select_gbif_fields . . . . .            | 13        |
| standardize_scientificName . . . . .    | 20        |
| wcvp_check_name . . . . .               | 21        |
| wcvp_check_name_batch . . . . .         | 23        |
| wcvp_get_data . . . . .                 | 25        |
| <b>Index</b>                            | <b>27</b> |

---

|                     |  |
|---------------------|--|
| collectors_get_name | <i>Get the last name of the main collector in recordedBy field</i> |
|---------------------|--|

---

Description

Returns the last name of the main collector

Usage

collectors\_get\_name(x)

Arguments

x                      recordedBy field

Details

Returns the last name of the main collector in recordedBy field

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 main collector's last name*

---

**Description**

Returns the list with the last name of the main collector associated with the unique key recordedBy.

**Usage**

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

**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 <a href="https://raw.githubusercontent.com/pablopains/parseGBIF/main/collectorDictionary/CollectorsDictionary">https://raw.githubusercontent.com/pablopains/parseGBIF/main/collectorDictionary/CollectorsDictionary</a> |

**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](#), [update\\_collectorsDictionary](#)

**Examples**

```
help(collectors_prepare_dictionary)

occ <- prepare_gbif_occurrence_data(gbif_occurrence_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 = "")
```

---

`collectors_update_dictionary`

*Create a key to group duplicates of a sample*

---

**Description**

Include recordedByStandardized field with verified main collector's last name. Include recordNumber\_Standard field with only numbers from recordNumber. Create a key to group duplicates in the key\_family\_recordedBy\_recordNumber field, composed of the fields: family + recordedByStandardized + recordNumber\_Standard.

**Usage**

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

**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

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/CollectorsDictionary>

## Details

Fields created for each incident record: nameRecordedBy\_Standard, recordNumber\_Standard, key\_family\_recordedBy\_Standard, 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')
```

---

|                     |   |
|---------------------|---|
| EnumOccurrenceIssue | <i>Enumeration GBIF issue An enumeration of validation rules for single occurrence records.</i> |
|---------------------|---|

---

## Description

There are many things that can go wrong and we continuously 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 merely 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 | <i>Export of results</i> |
|-------------|--------------------------|

---

## Description

from the selected digital voucher:

- select the taxonomic identification of the sample,
- Select geographic coordinates,
- Merge information between fields of duplicates of a sample,
- Compare the frequency of content in fields
- Generate data summary
- Export results

## Usage

```
export_data(
  occ_digital_voucher_file = "",
  occ_digital_voucher = NA,
  merge_occ_out = 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")
)

```

## Arguments

`occ_digital_voucher_file`  
 CSV file result of function `select_digital_voucher()` `$occ_digital_voucher`

`occ_digital_voucher`  
 data frame result of function `select_digital_voucher()` `$occ_digital_voucher`

`merge_occ_out` include records flagged `out_to_recover` in merge processing

`fields_to_merge`  
 fields to merge

`fields_to_compare`  
 fields to compare content frequency

`fields_to_parse`  
 all fields

## Details

Each data frame should be used as needed

## Value

list with six or seven (`merge_occ_out == TRUE`) data frames `occ_all` - all records processed with `parseGBIF_dataset_result` field indicating records flagged as in, `out_to_recover`, and duplicates. `occ_in_merge` - records inside - samples with data merged between duplicates `occ_in_raw` - records inside - samples with original digital voucher data `occ_dup` - duplicates `occ_out_to_recover_raw` - records out - samples with original digital voucher data `occ_out_to_recover_merge` - records out - (only if `lala == TRUE`) samples with data merged between duplicates `summary`

## Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

**See Also**

[batch\\_checkName\\_wcvp](#), [extract\\_gbif\\_issue](#)

**Examples**

```
help(export_data)

occ_results <- export_data(occ_digital_voucher_file = file.occ_digital_voucher, merge_occ_out = T)

names(occ_results)

NROW(occ_results$occ_all)
NROW(occ_results$occ_in_merge)
NROW(occ_results$occ_in_raw)
NROW(occ_results$occ_out_to_recover_merge)
NROW(occ_results$occ_out_to_recover_raw)
NROW(occ_results$occ_dup)

occ_results$summary
```

---

|                    |  |
|--------------------|--|
| extract_gbif_issue | <i>Extract GBIF ussue occurrence records</i> |
|--------------------|--|

---

**Description**

Extract GBIF validation rules for occurrence records

**Usage**

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

**Arguments**

|                     |  |
|---------------------|--|
| occ                 | GBIF occurrence table with selected columns as <code>select_gbif_fields(columns = 'standard')</code>                                       |
| enumOccurrenceIssue | An enumeration of validation rules for single occurrence records by GBIF file, if NA, will be used, <code>data(EnumOccurrenceIssue)</code> |

**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_occurrence_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)
```

---

```
prepare_gbif_occurrence_data
```

*Prepare occurrence data from GBIF to use in package*

---

**Description**

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

**Usage**

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

**Arguments**

|                      |   |
|----------------------|---|
| gbif_occurrence_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_occurrence_file = occ_file,
                                     columns = 'standard')

colnames(occ)

head(occ)
```

---

```
select_digital_voucher
```

*Select a sample among available duplicates*

---

**Description**

To group duplicates and choose the digital voucher:

1. If the key for grouping duplicates is complete with collector information and collection number, sample duplicates can be grouped. In this case, the voucher with the highest score is selected among the duplicates in the sample.

2. If the key to group duplicates is incomplete, sample duplicates cannot be grouped due to missing collector information and/or collection number. In this case, each record is considered a sample, without duplicates, and a voucher is selected for each sample.

How is the information score calculated?

moreInformativeRecord = sum of textual quality + quality of geospatial information.

How is the quality of textual information calculated?

The Text quality is the sum of the number of flags with text quality equal to TRUE.

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 collection site? Is there information about the municipality of collection? Is there information about the state/province of collection? Is there information about the bibliographic citation?

How is the quality of geospatial information calculated?

The quality of geospatial information is based on geographic issues made available by GBIF.

GBIF issues on the quality of geospatial information were classified into three levels.

- Not applicable, with selection\_score equal to 0
- Does not affect coordinating accuracy, with selection\_score equal to -1
- Potentially affect coordinate accuracy, with selection\_score equal to -3
- Records to be excluded from spatial analysis, with selection\_score equal to -9

How is the taxonomic identification of the sample chosen?

1. When the key to group the duplicates is complete: The accepted TAXON\_NAME identified at or below the specified level and the most frequent among the duplicates is chosen.

In case of a tie in frequency, in alphabetical order, the first accepted TAXON\_NAME identified up to or below the specific level is chosen.

If there is no identification, equal to or less than the specific level, for the sample, the sample is indicated as unidentified.

1. When the key to group the duplicates is incomplete: If so, the accepted TAXON\_NAME identified at or below the specified level is used. If there is no identification, equal to or less than the specific level, the sample is indicated as unidentified.

## Usage

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

## 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)

### Details

- matchStatusDuplicates - "groupable", "not groupable: no recordedBy and no recordNumber", "not groupable: no recordNumber" or "not groupable: no recordedBy"
- numberTaxonNamesSample - count of the different accepted scientific names, identified up to or below the specific level, listed in the sample duplicates, or Zero, if there is no identification, equal to or below the specific level, for the sample.
- sampleTaxonName - TAXON\_name accepted and identified up to or below the specific level selected for the sample.
- sampleIdentificationStatus - 'Identified', 'divergent identifications', or 'unidentified'

### Value

list with two data frames: occ\_digital\_voucher\_and: occ\_digital\_voucher, only with selection result fields and occ\_join\_results, with all data processing fields.

### Author(s)

Pablo Hendrigo Alves de Melo, Nadia Bystrakova & 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$occ_wcvp_check_name,
                                occ_collectorsDictionary = res_collectorsDictionary$occ_collectorsDictionary,
                                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)

head(res_digital_voucher_and_sample_identification$occ_join_results)
colnames(res_digital_voucher_and_sample_identification$occ_join_results)
```

---

|                    |                           |
|--------------------|---------------------------|
| select_gbif_fields | <i>select_gbif_fields</i> |
|--------------------|---------------------------|

---

## 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'
- 'namePublishedIn Year'
- '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 (WCVF) 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 WCVF format. Separate generic epithet, specific epithet, variety, subspecies, form, hybrid and author, in the scientific name, if any. Standardize, according to WCVF, abbreviation of infrataxon, if any: variety to var., subspecies to subsp., FORM to f., hybrid separator separate x from the specific epithet.

**Value**

searchedName, standardizedName, taxonAuthors, taxonAuthors\_last

**Author(s)**

Pablo Hendrigo Alves de Melo, Nadia Bystriakova & Alexandre Monro

**See Also**

[get\\_wcvf](#), [checkName\\_wcvf](#)

## Examples

```
# standardize_scientificName()

help(standardize_scientificName)

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 | <i>Use the World Checklist of Vascular Plants (WCVp) database to check accepted names and update synonyms.</i> |
|-----------------|--|

---

## Description

Use the **World Checklist of Vascular Plants database (about WCVp)** to check accepted names and update synonyms.

## 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 <a href="http://sftp.kew.org/pub/data-repositories/WCVp/">http://sftp.kew.org/pub/data-repositories/WCVp/</a><br>If NA, automatically load the latest version of the database by the function <code>parseGBIF::wcvp_get_data(read_only_to_memory = TRUE)\$wcvp_names</code> . |
| 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> search-Notes 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 WCVF 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

In batch, use the **World Checklist of Vascular Plants database (about WCVp)** to check accepted names and update synonyms

## Usage

```
wcvp_check_name_batch(
```

```

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

```

### Arguments

|   |   |
|---|---|
| <code>occ</code>                                      | GBIF occurrence table with selected columns as <code>select_gbif_fields(columns = 'standard')</code>  |
| <code>wcvp_names</code>                               | get data frame in <code>parseGBIF::wcvp_get_data(read_only_to_memory = TRUE)\$wcvp_names</code> or configure function to save a copy on local disk to optimize loading, see details in <code>help(wcvp_get_data)</code> |
| <code>if_author_fails_try_without_combinations</code> | option for partial verification of the authorship of the species. Remove the authors of combinations, in parentheses.   |
| <code>wcvp_selected_fields</code>                     | WCVF fields selected as return, 'standard' basic columns, 'all' all available columns. The default is 'standard'  |

### Details

See `help(checkName_wcvp)` and [about WCVF database](#)

### Value

list with two data frames: summary, species list and `occ_wcvp_check_name`, with WCVF 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/occurrence'

occ <- prepare_gbif_occurrence_data(gbif_occurrence_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 | <i>Get WCVP database</i> |
|---------------|--------------------------|

---

## 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

|                                  |  |
|----------------------------------|--|
| <code>url_source</code>          | <code>http://sftp.kew.org/pub/data-repositories/WCVP/</code>   |
| <code>read_only_to_memory</code> | TRUE to in-memory read-only, not writing a copy to local disk  |
| <code>path_results</code>        | download destination folder, if <code>read_only_to_memory</code> FALSE   |
| <code>update</code>              | TRUE to update and load files, FALSE to keep local version and load files, if <code>read_only_to_memory</code> FALSE |
| <code>load_distribution</code>   | TRUE to load file with geographical distribution of species, if <code>read_only_to_memory</code> 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/WCPV/',
                      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

## \* datasets

EnumOccurrenceIssue, [5](#)

batch\_checkName\_wcvp, [8](#), [12](#)

checkName\_wcvp, [20](#)

collectors\_get\_name, [2](#), [4](#), [5](#)

collectors\_prepare\_dictionary, [2](#), [3](#)

collectors\_update\_dictionary, [2](#), [4](#)

EnumOccurrenceIssue, [5](#)

export\_data, [6](#)

extract\_gbif\_issue, [8](#), [8](#), [10](#), [12](#), [20](#)

get\_wcvp, [20](#)

prepare\_collectorsDictionary, [5](#)

prepare\_gbif\_occurrence\_data, [9](#), [9](#), [20](#)

select\_digital\_voucher, [10](#)

select\_gbif\_fields, [9](#), [10](#), [13](#)

standardize\_scientificName, [20](#)

update\_collectorsDictionary, [4](#)

wcvp\_check\_name, [21](#), [24](#), [26](#)

wcvp\_check\_name\_batch, [22](#), [23](#), [26](#)

wcvp\_get\_data, [22](#), [24](#), [25](#)