parseGBIF Manual

Pablo Hendrigo Alves de Melo[1]

Nadia Bystriakova[2]

Alexandre Monro[3]

2023-07-12

parseGBIF Manual

The parseGBIF package is designed to repackage Global Biodiversity Information Facility - GBIF species occurrence records into a format that optimises its use in further analyses. Currently occurrence records in GBIF can include several duplicate digital records, and in the case of vascular plants, for several physical duplicates of unique collection events (biological collections). parseGBIF aims to parse these records to a single, synthetic, record corresponding to a unique collection event to which a standardized scientific name is associated. It does so by providing tools to verify and standardize species scientific names, score the quality of both the naming of a record and of its associated spatial data, and to use those scores to synthesise and parse duplicate records into unique collection events. This Manual provides a brief introduction to parseGBIF, with more information available from Help pages accessed via the help fuction. We believe that this package will be of particular use for analyses of plant occurrence data.

Installation

You can install the development version of parseGBIF from GitHub. To install parseGBIF, run

```r
devtools::install_github("pablopains/parseGBIF")
```

Please site parseGBIF as:

```r
print(citation("parseGBIF"), bibtex = FALSE)
```

[1] Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, pablopains@yahoo.com.br

[2] Natural History Museum, London, n_bystriakova@yahoo.com

[3] Royal Botanic Gardens, Kew, a.monro@kew.org

```
## To cite package 'parseGBIF' in publications use:
##
##   Melo P, Bystriakova N, Monro A (2023). "ParsGBIF: An R package for
##   parsing species occurrence records." _Methods in Ecology and
##   Evolution_, *1*(11), 1-11. doi:doi..... <https://doi.org/doi.....>.
```

Example

Getting species occurrence records from GBIF

1. GBIF data preparation

1.1. Getting occurrence data of the species records from GBIF

1.1.1. Access a registered account in GBIF

1.1.2. Filter occurrences using available fields, for instance:

- Basis of record: *Preserved specimen*
- Occurrence status: *present*
- Scientific name: *Botanical family name* (e.g. Achatocarpaceae) or **filter by other fields**

1.1.3. Request to download information in **DARWIN CORE ARCHIVE FORMAT**

1.1.4. Download compressed file and unzip downloaded file

1.1.5. Use the **occurrence.txt** file as input to the prepare_gbif_occurrence_data(gbif_occurrece_file = 'occurrence.txt') function

1.2. Preparing occurrence data downloaded from GBIF

To prepare occurrence data downloaded from GBIF to be used by parseGBIF functions, run prepare_gbif_occurrence_data.

```
library(parseGBIF)

occ_file <-
'https://raw.githubusercontent.com/pablopains/parseGBIF/main/dataGBIF/Ach
atocarpaceae/occurrence.txt'

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

head(occ)

## # A tibble: 6 × 55
##   Ctrl_gbifID Ctrl_bibliographicCitation Ctrl_language
Ctrl_institutionCode
##         <dbl> <chr>                      <chr>         <chr>
## 1   931015583 <NA>                       es            Universidad de
```

```
Antioquia…
## 2   931008720 <NA>                        es           Universidad de
Antioquia…
## 3   931008021 <NA>                        es           Universidad de
Antioquia…
## 4   931002305 <NA>                        es           Universidad de
Antioquia…
## 5   931001019 <NA>                        es           Universidad de
Antioquia…
## 6   930997693 <NA>                        es           Universidad de
Antioquia…
## # ℹ 51 more variables: Ctrl_collectionCode <chr>, Ctrl_datasetName
<chr>,
## #   Ctrl_basisOfRecord <chr>, Ctrl_informationWithheld <chr>,
## #   Ctrl_dataGeneralizations <chr>, Ctrl_occurrenceID <chr>,
## #   Ctrl_catalogNumber <chr>, Ctrl_recordNumber <chr>, Ctrl_recordedBy
<chr>,
## #   Ctrl_georeferenceVerificationStatus <chr>, Ctrl_occurrenceStatus
<chr>,
## #   Ctrl_eventDate <dttm>, Ctrl_year <dbl>, Ctrl_month <dbl>, Ctrl_day
<dbl>,
## #   Ctrl_habitat <chr>, Ctrl_fieldNotes <chr>, Ctrl_eventRemarks
<chr>, …
```

When parsing data, the user can choose to select “standard” or “all” fields (columns). The “standard” format has 55 data fields (columns), and the “all” format, 257 data fields (columns).

```r
col_standard <- parseGBIF::select_gbif_fields(columns = 'standard')

str(col_standard)

##  chr [1:55] "gbifID" "bibliographicCitation" "language"
"institutionCode" ...

col_all <- parseGBIF::select_gbif_fields(columns = 'all')

str(col_all)

##  chr [1:257] "gbifID" "abstract" "accessRights" "accrualMethod" ...
```

1.3. Extracting GBIF issues

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.

```
data(EnumOccurrenceIssue)

colnames(EnumOccurrenceIssue)

## [1] "constant"        "description"     "definition"      "type"
## [5] "priority"        "score"           "selection_score" "reasoning"
## [9] "notes"

head(dplyr::arrange(EnumOccurrenceIssue,desc(score)))

## # A tibble: 6 × 9
##   constant description definition type  priority score selection_score
reasoning
##   <chr>    <chr>       <chr>      <chr> <chr>    <dbl>           <dbl>
<chr>
## 1 COORDIN… Coordinate… A coordin… geos… High         3              -9
Records …
## 2 COORDIN… Coordinate… The suppl… geos… High         3              -9
Records …
## 3 COUNTRY… The interp… The inter… geos… High         3              -9
Records …
## 4 ZERO_CO… Coordinate… Coordinat… geos… High         3              -9
Records …
## 5 COORDIN… The given … The given… geos… Medium       2              -3
Potentia…
## 6 COORDIN… Indicates … Indicates… geos… Medium       2              -3
Potentia…
## # ℹ 1 more variable: notes <chr>

occ_gbif_issue <- parseGBIF::extract_gbif_issue(occ = occ)

names(occ_gbif_issue)

## [1] "occ_gbif_issue" "summary"

head(occ_gbif_issue$summary)

##                                              issue n_occ
## 1                          INSTITUTION_MATCH_FUZZY  1919
## 2                     GEODETIC_DATUM_ASSUMED_WGS84  1883
## 3 OCCURRENCE_STATUS_INFERRED_FROM_INDIVIDUAL_COUNT  1336
## 4                CONTINENT_DERIVED_FROM_COORDINATES  1038
## 5                               COORDINATE_ROUNDED   978
## 6                               TYPE_STATUS_INVALID   858
```

2. Check species names against WCVP database

The World Checklist of Vascular Plants (WCVP) database is available from the (Royal Botanic Gardens, Kew)[https://powo.science.kew.org/about-wcvp]. It can be downloaded to a folder of the user's choice or into memory using get_wcvp function. The output has 33 columns.

```
data(wcvp_names_Achatocarpaceae)
wcvp_names <- wcvp_names_Achatocarpaceae

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

colnames(wcvp_names)

##  [1] "plant_name_id"            "ipni_id"
##  [3] "taxon_rank"               "taxon_status"
##  [5] "family"                   "genus_hybrid"
##  [7] "genus"                    "species_hybrid"
##  [9] "species"                  "infraspecific_rank"
## [11] "infraspecies"             "parenthetical_author"
## [13] "primary_author"           "publication_author"
## [15] "place_of_publication"     "volume_and_page"
## [17] "first_published"          "nomenclatural_remarks"
## [19] "geographic_area"          "lifeform_description"
## [21] "climate_description"      "taxon_name"
## [23] "taxon_authors"            "accepted_plant_name_id"
## [25] "basionym_plant_name_id"   "replaced_synonym_author"
## [27] "homotypic_synonym"        "parent_plant_name_id"
## [29] "powo_id"                  "hybrid_formula"
## [31] "reviewed"                 "TAXON_NAME_U"
## [33] "TAXON_AUTHORS_U"
```

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.

```
name.checked <- wcvp_check_name(searchedName = 'Achatocarpus mollis
H.Walter',
               wcvp_names = wcvp_names,
               if_author_fails_try_without_combinations = TRUE)
name.checked[,c(3:5,22,23,40)]

##   wcvp_taxon_rank wcvp_taxon_status     wcvp_family
wcvp_taxon_name
## 9         Species          Accepted Achatocarpaceae Achatocarpus
pubescens
##   wcvp_taxon_authors wcvp_searchNotes
## 9         C.H.Wright          Updated
```

To check names in a batch mode, there is wcvp_check_name_batch function. It uses the occurrence data (occ) and WCVP names list (wcvp_names) generated in the previous steps.

```
names.checked <- wcvp_check_name_batch(occ = occ,
                                                  wcvp_names =
wcvp_names,

if_author_fails_try_without_combinations = TRUE,
                                                  wcvp_selected_fields =
'standard')

names(names.checked)

## [1] "occ_wcvp_check_name" "summary"

head(names.checked$summary)

##    wcvp_plant_name_id wcvp_taxon_rank wcvp_taxon_status
wcvp_family
## 4              500156         Species          Accepted
Achatocarpaceae
## 22             500161            Form          Accepted
Achatocarpaceae
## 15             500146         Species          Accepted
Achatocarpaceae
## 9              500163         Species          Accepted
Achatocarpaceae
## 2              500150         Species          Accepted
Achatocarpaceae
## 16             500149         Species          Accepted
Achatocarpaceae
##                        wcvp_taxon_name       wcvp_taxon_authors
## 4             Achatocarpus nigricans                   Triana
## 22 Achatocarpus praecox f. obovatus (Schinz & Autran) Hauman
## 15             Achatocarpus balansae           Schinz & Autran
## 9             Achatocarpus pubescens               C.H.Wright
## 2              Achatocarpus gracilis                 H.Walter
## 16   Achatocarpus brevipedicellatus                 H.Walter
##    wcvp_accepted_plant_name_id wcvp_reviewed
## 4                       500156             Y
## 22                      500161             Y
## 15                      500146             Y
## 9                       500163             Y
## 2                       500150             Y
## 16                      500149             Y
##                             wcvp_searchedName
wcvp_taxon_status_of_searchedName
## 4            Achatocarpus nigricans Triana
<NA>
## 22   Achatocarpus obovatus Schinz & Autran
Synonym
## 15   Achatocarpus balansae Schinz & Autran
<NA>
## 9         Achatocarpus pubescens C.H.Wright
```

```
<NA>
## 2           Achatocarpus gracilis H.Walter
<NA>
## 16 Achatocarpus brevipedicellatus H.Walter
<NA>
##    wcvp_plant_name_id_of_searchedName
wcvp_taxon_authors_of_searchedName
## 4                                   NA
<NA>
## 22                              500158                    Schinz &
Autran
## 15                                  NA
<NA>
## 9                                   NA
<NA>
## 2                                   NA
<NA>
## 16                                  NA
<NA>
##    wcvp_verified_author wcvp_verified_speciesName wcvp_searchNotes
## 4                   100                       100         Accepted
## 22                  100                       100          Updated
## 15                  100                       100         Accepted
## 9                   100                       100         Accepted
## 2                   100                       100         Accepted
## 16                  100                       100         Accepted
```

To bring species' names into line with the format used by WCVP, the function standardize_scientificName inserts a space between the hybrid separator (x) and specific epithet, and also standardizes abbreviations of infrataxa (variety, subspecies, form).

```
# hybrid separator
standardize_scientificName('Leucanthemum ×superbum (Bergmans ex
J.W.Ingram) D.H.Kent')

## $searchedName
## [1] "Leucanthemum ×superbum (Bergmans ex J.W.Ingram) D.H.Kent"
##
## $standardizeName
## [1] "Leucanthemum × superbum"
##
## $taxonAuthors
## [1] "(Bergmans ex J.W.Ingram) D.H.Kent"
##
## $taxonAuthors_last
## [1] "D.H.Kent"

# variety

standardize_scientificName('Urera baccifera var. angustifolia Wedd.')
```

```
## $searchedName
## [1] "Urera baccifera var. angustifolia Wedd."
##
## $standardizeName
## [1] "Urera baccifera var. angustifolia"
##
## $taxonAuthors
## [1] "Wedd."
##
## $taxonAuthors_last
## [1] ""
```

```r
# subspecies
standardize_scientificName('Platymiscium pubescens subsp. fragrans
(Rusby) Klitg.')
```

```
## $searchedName
## [1] "Platymiscium pubescens subsp. fragrans (Rusby) Klitg."
##
## $standardizeName
## [1] "Platymiscium pubescens subsp. fragrans"
##
## $taxonAuthors
## [1] "(Rusby) Klitg."
##
## $taxonAuthors_last
## [1] "Klitg."
```

The function collectors_get_name returns the last name of the main collector in recordedBy field. It standardizes the text string to replace non-ascii characters.

```r
# library(parseGBIF)

collectors_get_name('Müller, W.')
```

```
## [1] "Muller"
```

```r
collectors_get_name("PEDRO ACEVEDO-RODRÍGUEZ|A. SIACA|GEORGE R.
PROCTOR|JULIE F. BARCELONA|J.A. CEDEÑO|P. LEWIS|R. O'REILLY|E. SANTIAGO")
```

```
## [1] "ACEVEDO-RODRIGUEZ"
```

```r
collectors_get_name("BORNMÜLLER, JOSEPH FRIEDRICH NICOLAUS")
```

```
## [1] "BORNMULLER"
```

```r
collectors_get_name("Botão, S.R.; Machado, F.P.")
```

```
## [1] "Botao"
```

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

```
## [1] "Melo"
```

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

## [1] "Monro"

collectors_get_name('Bystriakova, N., Monro, A.,Melo, P.H.A')

## [1] "Bystriakova"
```

3. Collectors Dictionary

To extract the last name 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 uses the occurrence data (occ) generated in the previous step.

3.1 Prepare dictionary collectors

```
collectorsDictionary.dataset <- collectors_prepare_dictionary(occ = occ)

head(collectorsDictionary.dataset)

##   Ctrl_nameRecordedBy_Standard
Ctrl_recordedBy
## 1                      ACEVEDO                         R. ACEVEDO R.;
C. REYES
## 2                       ACOSTA                     R. ACOSTA P.; F.
VÁZQUEZ B.
## 3                       ACOSTA                      R. ACOSTA P.; N.
ACOSTA B.
## 4                       ADARVE
J. ADARVE
## 5                      AGUILAR
AGUILAR, R.M.
## 6                      AGUILAR JOSÉ AGUILAR CANO;SANDRA
MEDINA;JHONATAN GUZMÁN
##   Ctrl_notes collectorDictionary Ctrl_update collectorName
Ctrl_fullName
## 1       <NA>                            <NA>          <NA>
<NA>
## 2       <NA>                            <NA>          <NA>
<NA>
## 3       <NA>                            <NA>          <NA>
<NA>
## 4       <NA>                            <NA>          <NA>
<NA>
## 5       <NA>                            <NA>          <NA>
<NA>
## 6       <NA>                            <NA>          <NA>
<NA>
##   Ctrl_fullNameII CVStarrVirtualHerbarium_PersonDetails
## 1            <NA>                                  <NA>
## 2            <NA>                                  <NA>
## 3               <NA>                                    <NA>
## 4               <NA>                                    <NA>
## 5               <NA>                                    <NA>
## 6               <NA>                                    <NA>
```

3.2 Check the main collector’s last name

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 all characters are replaced by uppercase and 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.

```
  file.collectorsDictionary.dataset <-
'file_collectorsDictionary_dataset.csv'

  write.csv(collectorsDictionary.dataset,
            file.collectorsDictionary.dataset,
            row.names = FALSE,
            fileEncoding = "UTF-8",
            na = "")
```

3.3 Generating the collection event key

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. in the future.

```
  occ_collectorsDictionary <- generate_collection_event_key(occ=occ,
                                        collectorDictionary_checked_file
= file.collectorsDictionary.dataset)

## [1] "Loading collectorDictionary..."

  names(occ_collectorsDictionary)
```

```
## [1] "occ_collectorsDictionary" "summary"
## [3] "collectorsDictionary_add"

  head(occ_collectorsDictionary$occ_collectorsDictionary[,c(1,3)])

## # A tibble: 6 × 2
##   Ctrl_nameRecordedBy_Standard Ctrl_key_family_recordedBy_recordNumber
##   <chr>                        <chr>
## 1 FONNEGRA                     ACHATOCARPACEAE_FONNEGRA_1629
## 2 ROLDAN                       ACHATOCARPACEAE_ROLDAN_957
## 3 FONNEGRA                     ACHATOCARPACEAE_FONNEGRA_1657
## 4 BUNCH                        ACHATOCARPACEAE_BUNCH_
## 5 DUQUE                        ACHATOCARPACEAE_DUQUE_4101
## 6 TRUJILLO                     ACHATOCARPACEAE_TRUJILLO_5470
```

4. Selecting the master digital voucher

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.

```
occ_digital_voucher <- parseGBIF::select_digital_voucher(occ = occ,
                                                          occ_gbif_issue =
occ_gbif_issue$occ_gbif_issue,

occ_wcvp_check_name = names.checked$occ_wcvp_check_name ,

occ_collectorsDictionary =
occ_collectorsDictionary$occ_collectorsDictionary)

  names(occ_digital_voucher)

## [1] "occ_digital_voucher" "occ_results"

  colnames(occ_digital_voucher$occ_digital_voucher)

##   [1] "COORDINATE_UNCERTAINTY_METERS_INVALID"
##   [2] "CONTINENT_COORDINATE_MISMATCH"
##   [3] "CONTINENT_COUNTRY_MISMATCH"
##   [4] "CONTINENT_DERIVED_FROM_COORDINATES"
##   [5] "CONTINENT_DERIVED_FROM_COUNTRY"
##   [6] "COORDINATE_ACCURACY_INVALID"
##   [7] "COORDINATE_PRECISION_INVALID"
##   [8] "COORDINATE_PRECISION_UNCERTAINTY_MISMATCH"
##   [9] "COORDINATE_REPROJECTED"
##  [10] "ELEVATION_NON_NUMERIC"
##  [11] "ELEVATION_NOT_METRIC"
##  [12] "ELEVATION_UNLIKELY"
##  [13] "CONTINENT_INVALID"
##  [14] "COUNTRY_DERIVED_FROM_COORDINATES"
##  [15] "COUNTRY_INVALID"
##  [16] "ELEVATION_MIN_MAX_SWAPPED"
##  [17] "COORDINATE_ROUNDED"
##  [18] "DEPTH_MIN_MAX_SWAPPED"
##  [19] "DEPTH_NON_NUMERIC"
##  [20] "DEPTH_NOT_METRIC"
##  [21] "DEPTH_UNLIKELY"
##  [22] "COUNTRY_MISMATCH"
##  [23] "COORDINATE_REPROJECTION_FAILED"
##  [24] "COORDINATE_REPROJECTION_SUSPICIOUS"
##  [25] "GEODETIC_DATUM_INVALID"
##  [26] "PRESUMED_NEGATED_LATITUDE"
##  [27] "PRESUMED_NEGATED_LONGITUDE"
##  [28] "PRESUMED_SWAPPED_COORDINATE"
##  [29] "GEODETIC_DATUM_ASSUMED_WGS84"
##  [30] "COORDINATE_INVALID"
##  [31] "COORDINATE_OUT_OF_RANGE"
##  [32] "COUNTRY_COORDINATE_MISMATCH"
##  [33] "ZERO_COORDINATE"
##  [34] "AMBIGUOUS_COLLECTION"
##  [35] "AMBIGUOUS_INSTITUTION"
##  [36] "BASIS_OF_RECORD_INVALID"
##  [37] "COLLECTION_MATCH_FUZZY"
##  [38] "COLLECTION_MATCH_NONE"
##  [39] "DIFFERENT_OWNER_INSTITUTION"
##  [40] "FOOTPRINT_SRS_INVALID"
##  [41] "FOOTPRINT_WKT_INVALID"
##  [42] "FOOTPRINT_WKT_MISMATCH"
##  [43] "GEOREFERENCED_DATE_INVALID"
##  [44] "GEOREFERENCED_DATE_UNLIKELY"
##  [45] "IDENTIFIED_DATE_INVALID"
##  [46] "IDENTIFIED_DATE_UNLIKELY"
##  [47] "INDIVIDUAL_COUNT_CONFLICTS_WITH_OCCURRENCE_STATUS"
##  [48] "INDIVIDUAL_COUNT_INVALID"
##  [49] "INSTITUTION_COLLECTION_MISMATCH"
##  [50] "INSTITUTION_MATCH_FUZZY"
##  [51] "INSTITUTION_MATCH_NONE"
##  [52] "INTERPRETATION_ERROR"
##  [53] "MODIFIED_DATE_INVALID"
##  [54] "MODIFIED_DATE_UNLIKELY"
##  [55] "MULTIMEDIA_DATE_INVALID"
##  [56] "MULTIMEDIA_URI_INVALID"
##  [57] "OCCURRENCE_STATUS_INFERRED_FROM_BASIS_OF_RECORD"
##  [58] "OCCURRENCE_STATUS_INFERRED_FROM_INDIVIDUAL_COUNT"
##  [59] "OCCURRENCE_STATUS_UNPARSABLE"
##  [60] "POSSIBLY_ON_LOAN"
##  [61] "RECORDED_DATE_INVALID"
##  [62] "RECORDED_DATE_MISMATCH"
##  [63] "RECORDED_DATE_UNLIKELY"
##  [64] "REFERENCES_URI_INVALID"
##  [65] "TAXON_MATCH_AGGREGATE"
##  [66] "TAXON_MATCH_FUZZY"
##  [67] "TAXON_MATCH_HIGHERRANK"
##  [68] "TAXON_MATCH_NONE"
##  [69] "TYPE_STATUS_INVALID"
##  [70] "Ctrl_gbifID"
##  [71] "Ctrl_bibliographicCitation"
##  [72] "Ctrl_language"
##  [73] "Ctrl_institutionCode"
##  [74] "Ctrl_collectionCode"
##  [75] "Ctrl_datasetName"
##  [76] "Ctrl_basisOfRecord"
##  [77] "Ctrl_informationWithheld"
##  [78] "Ctrl_dataGeneralizations"
##  [79] "Ctrl_occurrenceID"
##  [80] "Ctrl_catalogNumber"
##  [81] "Ctrl_recordNumber"
##  [82] "Ctrl_recordedBy"
##  [83] "Ctrl_georeferenceVerificationStatus"
##  [84] "Ctrl_occurrenceStatus"
##  [85] "Ctrl_eventDate"
##  [86] "Ctrl_year"
##  [87] "Ctrl_month"
##  [88] "Ctrl_day"
##  [89] "Ctrl_habitat"
##  [90] "Ctrl_fieldNotes"
##  [91] "Ctrl_eventRemarks"
##  [92] "Ctrl_locationID"
##  [93] "Ctrl_higherGeography"
##  [94] "Ctrl_islandGroup"
##  [95] "Ctrl_island"
##  [96] "Ctrl_countryCode"
##  [97] "Ctrl_stateProvince"
##  [98] "Ctrl_county"
##  [99] "Ctrl_municipality"
## [100] "Ctrl_locality"
## [101] "Ctrl_verbatimLocality"
## [102] "Ctrl_locationRemarks"
## [103] "Ctrl_decimalLatitude"
## [104] "Ctrl_decimalLongitude"
## [105] "Ctrl_verbatimCoordinateSystem"
## [106] "Ctrl_verbatimIdentification"
## [107] "Ctrl_identificationQualifier"
## [108] "Ctrl_typeStatus"
## [109] "Ctrl_identifiedBy"
## [110] "Ctrl_dateIdentified"
## [111] "Ctrl_scientificName"
## [112] "Ctrl_family"
## [113] "Ctrl_taxonRank"
## [114] "Ctrl_nomenclaturalCode"
## [115] "Ctrl_taxonomicStatus"
## [116] "Ctrl_issue"
## [117] "Ctrl_mediaType"
## [118] "Ctrl_hasCoordinate"
## [119] "Ctrl_hasGeospatialIssues"
## [120] "Ctrl_verbatimScientificName"
## [121] "Ctrl_level0Name"
## [122] "Ctrl_level1Name"
## [123] "Ctrl_level2Name"
## [124] "Ctrl_level3Name"
## [125] "wcvp_plant_name_id"
## [126] "wcvp_taxon_rank"
## [127] "wcvp_taxon_status"
## [128] "wcvp_family"
## [129] "wcvp_taxon_name"
## [130] "wcvp_taxon_authors"
## [131] "wcvp_accepted_plant_name_id"
## [132] "wcvp_reviewed"
## [133] "wcvp_searchedName"
## [134] "wcvp_taxon_status_of_searchedName"
## [135] "wcvp_plant_name_id_of_searchedName"
## [136] "wcvp_taxon_authors_of_searchedName"
## [137] "wcvp_verified_author"
## [138] "wcvp_verified_speciesName"
## [139] "wcvp_searchNotes"
## [140] "Ctrl_nameRecordedBy_Standard"
## [141] "Ctrl_recordNumber_Standard"
## [142] "Ctrl_key_family_recordedBy_recordNumber"
## [143] "Ctrl_key_year_recordedBy_recordNumber"
## [144] "Ctrl_geospatial_quality"
## [145] "Ctrl_verbatim_quality"
## [146] "Ctrl_moreInformativeRecord"
## [147] "parseGBIF_digital_voucher"
## [148] "parseGBIF_duplicates"
## [149] "parseGBIF_num_duplicates"
## [150] "parseGBIF_non_groupable_duplicates"
## [151] "parseGBIF_duplicates_grouping_status"
## [152] "Ctrl_coordinates_validated_by_gbif_issue"
```

```
  file.occ_digital_voucher <-  'occ_digital_voucher.csv'

  write.csv(occ_digital_voucher$occ_digital_voucher,
            file.occ_digital_voucher,
            row.names = FALSE,
            fileEncoding = "UTF-8",
            na = "")
```

5. Export of results

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?

- **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.

- **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

export_data function return a list with 10 data frames:

- **all_data** All records processed, merged Unique collection events complete / incomplete and their duplicates
- **useable_data_merge** Merged useable dataset
- **useable_data_raw** Raw useable dataset
- **duplicates** Duplicates of unique collection events of useable and useable datasets
- **unusable_data_merge** Merged unusable dataset. It is NA if merge_unusable_data is FALSE.
- **unusable_data_raw** Raw unusable dataset
- **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

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

names(results)
```

```
##  [1] "all_data"
##  [2] "useable_data_merge"
##  [3] "useable_data_raw"
##  [4] "duplicates"
##  [5] "unusable_data_merge"
##  [6] "unusable_data_raw"
##  [7] "parseGBIF_general_summary"
##  [8] "parseGBIF_merge_fields_summary"
##  [9] "parseGBIF_merge_fields_summary_useable_data"
## [10] "parseGBIF_merge_fields_summary_unusable_data"
```

```r
results$parseGBIF_general_summary
```

```
##                                                  question value
## 1                                 total number of records  3626
## 2                  total number of unique collection events  2777
## 3         total number of unique collection events complete  1527
## 4       total number of unique collection events incomplete  1250
## 5                              total number of duplicates   849
## 6 total unique collection events containing merged fields    86
```

```r
results$parseGBIF_merge_fields_summary
```

```
##                                  question value
## 1        Ctrl_habitat : total merge actions    41
## 2       Ctrl_locality : total merge actions    22
## 3   Ctrl_municipality : total merge actions    15
## 4  Ctrl_stateProvince : total merge actions    13
## 5     Ctrl_fieldNotes : total merge actions    12
## 6           Ctrl_year : total merge actions    11
## 7      Ctrl_eventDate : total merge actions    11
## 8     Ctrl_level0Name : total merge actions     2
## 9     Ctrl_level1Name : total merge actions     2
## 10    Ctrl_level2Name : total merge actions     2
## 11    Ctrl_level3Name : total merge actions     1
## 12   Ctrl_countryCode : total merge actions     0
```

```r
results$parseGBIF_merge_fields_summary_useable_data
```

```
##                            question value
## 1        Ctrl_habitat : merge actions    31
## 2       Ctrl_locality : merge actions    17
## 3   Ctrl_municipality : merge actions    12
## 4     Ctrl_fieldNotes : merge actions    11
## 5  Ctrl_stateProvince : merge actions    11
## 6           Ctrl_year : merge actions     8
## 7      Ctrl_eventDate : merge actions     8
## 8     Ctrl_level0Name : merge actions     2
## 9     Ctrl_level1Name : merge actions     2
## 10    Ctrl_level2Name : merge actions     2
## 11    Ctrl_level3Name : merge actions     1
## 12   Ctrl_countryCode : merge actions     0
```

```r
NROW(results$all_data)
```

```
## [1] 3626
```

```r
NROW(results$useable_data_merge)
```

```
## [1] 1527

NROW(results$useable_data_raw)

## [1] 1527

NROW(results$duplicates)

## [1] 849

file.all <- 'parseGBIF_all_data.csv'
write.csv(results$all_data,
          file.all,
          row.names = FALSE,
          fileEncoding = "UTF-8",
          na = "")

file.summary <- 'parseGBIF_general_summary.csv'
write.csv(results$parseGBIF_general_summary,
          file.summary,
          row.names = FALSE,
          fileEncoding = "UTF-8",
          na = "")

file.summary <- 'parseGBIF_merge_fields_summary.csv'
write.csv(results$parseGBIF_merge_fields_summary,
          file.summary,
          row.names = FALSE,
          fileEncoding = "UTF-8",
          na = "")

file.summary <- 'parseGBIF_merge_fields_summary_complete.csv'
write.csv(results$parseGBIF_merge_fields_summary_complete,
          file.summary,
          row.names = FALSE,
          fileEncoding = "UTF-8",
          na = "")
```

Accessing map of merged information and frequency of content in fields

- Merged information between fields of duplicates of a unique collection event

```
index <-
results$useable_data_merge$Ctrl_key_family_recordedBy_recordNumber %in%
'ACHATOCARPACEAE_ZARDINI_5592'

print('merged fields')

## [1] "merged fields"

print(jsonlite::fromJSON(results$useable_data_merge$parseGBIF_merged_fiel
ds[index==TRUE]))
```

```
## $Ctrl_locality
## [1] "4061323610"
```

```r
print('merged fields map')
```

```
## [1] "merged fields map"
```

```r
print(jsonlite::fromJSON(results$useable_data_merge$parseGBIF_duplicates_
map[index==TRUE]))
```

```
## $Ctrl_gbifID
## [1] "2996695631" "4061323610" "1258412909" "1095491559"
##
## $Ctrl_scientificName
## [1] "Achatocarpus praecox Griseb."
## [2] "Achatocarpus praecox var. bicornutus (Schinz & Autran) Botta"
## [3] "Achatocarpus praecox var. bicornutus (Schinz & Autran) Botta"
##
## $Ctrl_recordedBy
## [1] "Zardini, EM;Zardini, E.M." "Elsa M. Zardini"
##
## $Ctrl_recordNumber
## [1] "5592"         "Zardini 5592"
##
## $Ctrl_identifiedBy
## [1] "Liesner, R."         "Ronald Liesner (MO)" "Ronald Liesner (MO)"
##
## $Ctrl_dateIdentified
## [1] "1990-01-01" "1996-01-01"
##
## $Ctrl_institutionCode
## [1] "MBM" "FCQ" "MO"
##
## $Ctrl_collectionCode
## [1] "MBM"      "Tropicos"
##
## $Ctrl_datasetName
## [1] ""
## [2] "Tropicos"
## [3] "MBM - Herbário do Museu Botânico Municipal"
##
## $Ctrl_datasetName
## [1] ""
## [2] "Tropicos"
## [3] "MBM - Herbário do Museu Botânico Municipal"
##
## $Ctrl_language
## [1] ""   "pt"
##
## $wcvp_plant_name_id
## [1] "500159" "500160"
```

```
##
## $wcvp_taxon_rank
## [1] "Species" "Variety"
##
## $wcvp_taxon_name
## [1] "Achatocarpus praecox"
## [2] "Achatocarpus praecox var. bicornutus"
##
## $wcvp_taxon_authors
## [1] "Griseb."                 "(Schinz & Autran) Botta"
## [3] "(Schinz & Autran) Botta"
##
## $Ctrl_stateProvince
## [1] "Paraguari" "Paraguarí"
##
## $Ctrl_locality
## [1] ""                           "Cerro Palacios. In forest."
```

- Frequency of content in fields between fields of duplicates of a unique collection event

```
print('Frequency of content in fields')

## [1] "Frequency of content in fields"

print(jsonlite::fromJSON(results$useable_data_merge$parseGBIF_freq_duplic
ate_or_missing_data[index==TRUE]))

## $Ctrl_gbifID
##        value freq
## 1 1095491559    1
## 2 1258412909    1
## 3 2996695631    1
## 4 4061323610    1
##
## $Ctrl_scientificName
##                                                       value freq
## 1                                 Achatocarpus praecox Griseb.    2
## 2 Achatocarpus praecox var. bicornutus (Schinz & Autran) Botta    2
##
## $Ctrl_recordedBy
##                       value freq
## 1           Elsa M. Zardini    2
## 2               Zardini, EM    1
## 3 Zardini, EM;Zardini, E.M.    1
##
## $Ctrl_recordNumber
##          value freq
## 1         5592    2
## 2 Zardini 5592    2
##
## $Ctrl_identifiedBy
##                  value freq
## 1 Ronald Liesner (MO)    2
## 2          Liesner, R    1
## 3         Liesner, R.    1
##
## $Ctrl_dateIdentified
##        value freq
## 1 1990-01-01    2
## 2 1996-01-01    2
##
## $Ctrl_institutionCode
##   value freq
## 1   MBM    2
## 2   FCQ    1
## 3    MO    1
##
## $Ctrl_collectionCode
##      value freq
## 1      MBM    2
## 2 Tropicos    2
##
## $Ctrl_datasetName
##                                        value freq
## 1                                   Tropicos    2
## 2 MBM - Herbário do Museu Botânico Municipal    1
## 3                                      empty    1
##
## $Ctrl_datasetName
##                                        value freq
## 1                                   Tropicos    2
## 2 MBM - Herbário do Museu Botânico Municipal    1
## 3                                      empty    1
##
## $Ctrl_language
##   value freq
## 1    pt    1
## 2 empty    3
##
## $wcvp_plant_name_id
##    value freq
## 1 500159    2
## 2 500160    2
##
## $wcvp_taxon_rank
##     value freq
## 1 Species    2
## 2 Variety    2
##
## $wcvp_taxon_status
##      value freq
## 1 Accepted    4
##
## $wcvp_family
##             value freq
## 1 Achatocarpaceae    4
##
## $wcvp_taxon_name
##                                  value freq
## 1                 Achatocarpus praecox    2
## 2 Achatocarpus praecox var. bicornutus    2
##
## $wcvp_taxon_authors
##                     value freq
## 1 (Schinz & Autran) Botta    2
## 2                 Griseb.    2
##
## $wcvp_reviewed
##   value freq
## 1     Y    4
##
## $wcvp_searchNotes
##      value freq
## 1 Accepted    4
##
## $Ctrl_fieldNotes
##                           value freq
## 1 Tree 8m.; Forest. aLT.: 250m.    1
## 2                         empty    3
##
## $Ctrl_year
##   value freq
## 1  1988    4
##
## $Ctrl_stateProvince
##       value freq
## 1 Paraguari    2
## 2 Paraguarí    2
##
## $Ctrl_municipality
##            value freq
## 1 Cerro Palacios    2
## 2          empty    2
##
## $Ctrl_locality
##                        value freq
## 1 Cerro Palacios. In forest.    2
## 2                      empty    2
##
## $Ctrl_countryCode
##   value freq
## 1    PY    4
##
## $Ctrl_eventDate
##        value freq
## 1 1988-07-09    4
##
## $Ctrl_habitat
##   value freq
## 1 empty    4
##
## $Ctrl_level0Name
##      value freq
## 1 Paraguay    4
##
## $Ctrl_level1Name
##       value freq
## 1 Paraguarí    4
##
## $Ctrl_level2Name
##    value freq
## 1 Pirayú    4
##
## $Ctrl_level3Name
##   value freq
## 1 empty    4
```