# parseGBIF Manual

Pablo Hendrigo Alves de Melo[1]

Nadia Bystriakova[2]

Alexandre Monro[3]

2024-03-01

## 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

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

Please site parseGBIF as:

```
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

```
 library(parseGBIF)

 folder_download <- tempdir()

 download_gbif_data_from_doi(gbif_doi_url =
'https://doi.org/10.15468/dl.nbcqc6', folder = folder_download,
keep_only_occurrence_file = TRUE)
```

```
## [1]
"C:\\Users\\PABLOH~1\\AppData\\Local\\Temp\\RtmpMtvI2y/dataGBIF.zip"
## [2]
"C:\\Users\\PABLOH~1\\AppData\\Local\\Temp\\RtmpMtvI2y/file1c02423973d5c"
## [3]
"C:\\Users\\PABLOH~1\\AppData\\Local\\Temp\\RtmpMtvI2y/file1c02445486737"
## [4]
"C:\\Users\\PABLOH~1\\AppData\\Local\\Temp\\RtmpMtvI2y/file1c0244efd25bd"
## [5]
"C:\\Users\\PABLOH~1\\AppData\\Local\\Temp\\RtmpMtvI2y/file1c024726879b3"
## [6]
"C:\\Users\\PABLOH~1\\AppData\\Local\\Temp\\RtmpMtvI2y/occurrence.txt"
```

### *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.

```
occ_file <- paste0(folder_download,'\\','occurrence.txt')
# occ_file <-
'https://raw.githubusercontent.com/pablopains/parseGBIF/main/dataGBIF/Achatocarpaceae/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   931017641 <NA>                       es            Universidad de
Antioquia…
## 2   931016976 <NA>                       es            Universidad de
Antioquia…
## 3   931016635 <NA>                       es            Universidad de
Antioquia…
## 4   931016614 <NA>                       es            Universidad de
Antioquia…
## 5   931016599 <NA>                       es            Universidad de
Antioquia…
## 6   931014610 <NA>                       es            Universidad de
Antioquia…
## # ⓘ 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.

## 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

```
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…
## # ⓘ 1 more variable: notes <chr>
```

```
gbif_issue <- parseGBIF::extract_gbif_issue(occ = occ)

names(gbif_issue)

## [1] "occ_gbif_issue" "summary"

head(gbif_issue$summary)

##                                                issue n_occ
## 1                          INSTITUTION_MATCH_FUZZY 20379
## 2                       GEODETIC_DATUM_ASSUMED_WGS84 10751
## 3                    INSTITUTION_COLLECTION_MISMATCH  8388
## 4                              COLLECTION_MATCH_NONE  6755
## 5                       CONTINENT_DERIVED_FROM_COUNTRY  5761
## 6 OCCURRENCE_STATUS_INFERRED_FROM_INDIVIDUAL_COUNT  5723

file.name <- 'parseGBIF_1_occ_issue.csv'

write.csv(gbif_issue$occ_gbif_issue,
            file.name,
            row.names = FALSE,
            fileEncoding = "UTF-8",
            na = "")

file.name <- 'parseGBIF_1_summary_issue.csv'

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

### 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)$wcvp_names
# wcvp_names <-  wcvp_get_data_v2.1(read_only_to_memory = TRUE,
#                                   load_rda_data = TRUE)$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',
                                       silence = TRUE)



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
## 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>
##   wcvp_taxon_name wcvp_taxon_authors wcvp_accepted_plant_name_id
wcvp_reviewed
## 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>
##         wcvp_searchedName wcvp_taxon_status_of_searchedName
## 1  Carpotroche longifolia                                NA
## 2    Carpotroche pacifica                                NA
## 3  Hydnocarpus annamensis                                NA
## 4 Hydnocarpus hainanensis                                NA
## 5     Hydnocarpus alpinus                                NA
## 6      Gynocardia odorata                                NA
##   wcvp_plant_name_id_of_searchedName
wcvp_taxon_authors_of_searchedName
## 1                                 NA
NA
## 2                                 NA
NA
## 3                                 NA
NA
## 4                                 NA
NA
## 5                                 NA
NA
## 6                                 NA
NA
##   wcvp_verified_author wcvp_verified_speciesName wcvp_searchNotes
## 1                    0                         0        Not found
## 2                    0                         0        Not found
## 3                    0                         0        Not found
## 4                    0                         0        Not found
## 5                    0                         0        Not found
## 6                    0                         0        Not found

file.name <- 'parseGBIF_2_occ_wcvp_check_name.csv'
write.csv(names.checked$occ_wcvp_check_name,
```

```
            file.name,
            row.names = FALSE,
            fileEncoding = "UTF-8",
            na = "")
```

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] ""

# 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.

```
# library(parseGBIF)

collectors_get_name('Müller, W.')

## [1] "MULLER"

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"

collectors_get_name("BORNMÜLLER, JOSEPH FRIEDRICH NICOLAUS")

## [1] "BORNMULLER"

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

## [1] "BOTAO"

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,
collectorDictionary_file =
'https://raw.githubusercontent.com/pablopains/parseGBIF/main/collectorDic
tionary/CollectorsDictionary_parseGBIF.csv',
```

```
                                                                  silence =
TRUE)

NROW(collectorsDictionary.dataset)

## [1] 9776

head(collectorsDictionary.dataset)

##   Ctrl_nameRecordedBy_Standard                    Ctrl_recordedBy
Ctrl_notes
## 1                        <KEU>                      KEND.<KEU???>
<NA>
## 2                       AABGON                           E AABGON
<NA>
## 3                       ABAGON                           ABAGON E
<NA>
## 4                       ABBOTT                         ABBOTT, ATD
<NA>
## 5                    ABDULHADI                         ABDULHADI R
<NA>
## 6                        ABEID ABEID YS; KIBURE OA; NGUMWE; NJOU S
<NA>
##   collectorDictionary Ctrl_update collectorName Ctrl_fullName
Ctrl_fullNameII
## 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>
##   CVStarrVirtualHerbarium_PersonDetails
## 1                                  <NA>
## 2                                  <NA>
## 3                                  <NA>
## 4                                  <NA>
## 5                                  <NA>
## 6                                  <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.The collectorDictionary_file parameter, in the collectors_prepare_dictionary function, indicates the collector dictionary file on your local disk, or by default the collector dictionary will be indicated, verified and maintained by the parseGBIF team, downloaded via git from https://raw.githubusercontent.com/pablopains/parseGBIF/main/collectorDictionary/CollectorsDictionary.csv

```
  file.collectorsDictionary.dataset <-
'parseGBIF_3_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.

The collection event key for grouping duplicates is complete when the records match all parts of the key (the botanical family, the primary collector's surname and the collector's number). If part of the key is missing, the collection event key is incomplete.

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

```
  collectorsDictionary <- generate_collection_event_key(occ=occ,

collectorDictionary_checked_file = file.collectorsDictionary.dataset)

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

  names(collectorsDictionary)
```

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

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

```
## # A tibble: 6 × 2
##   Ctrl_nameRecordedBy_Standard Ctrl_key_family_recordedBy_recordNumber
##   <chr>                        <chr>
## 1 FONNEGRA                     ACHARIACEAE_FONNEGRA_1011
## 2 ACEVEDO                      ACHARIACEAE_ACEVEDO_11249
## 3 GOMEZ                        ACHARIACEAE_GOMEZ_540
## 4 GOMEZ                        ACHARIACEAE_GOMEZ_540
## 5 GOMEZ                        ACHARIACEAE_GOMEZ_540
## 6 GENTRY                       ACHARIACEAE_GENTRY_7215
```

```
  file.name <- 'parseGBIF_3_occ_collectorsDictionary.csv'
  write.csv(collectorsDictionary$occ_collectorsDictionary, file.name,
            row.names = FALSE,
            fileEncoding = "UTF-8",
            na = "")


  file.name <- 'parseGBIF_3_summary_collectorsDictionary.csv'
  write.csv(collectorsDictionary$summary, file.name,
            row.names = FALSE,
            fileEncoding = "UTF-8",
            na = "")

file.name <- 'parseGBIF_3_collectorsDictionary_add.csv'
  write.csv(collectorsDictionary$collectorsDictionary_add, file.name,
            row.names = FALSE,
            fileEncoding = "UTF-8",
            na = "")
```

### 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 to select the master digital voucher?** Master digital voucher is 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 country (using a GBIF issue COUNTRY_INVALID)?
- 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 If they are selected as a digital voucher (for example, due to lack of duplicates), in the export step, they will be classified as unusable.

**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, coordinates are not useful for spatial analysis. If they are selected as a digital voucher (for example, due to lack of duplicates), in the export step, they will be classified as unusable.

**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.

**How is the geospatial information selected?** If the master voucher does not have geographic coordinates, or if the quality of the geospatial information according to the classification of the GBIF issue is poor (-9), we will search for the coordinates in the duplicate records associated with it.

**The main output fields relating to taxonomic identification and geographic coordinates:**

- parseGBIF_digital_voucher = TRUE indicates the master digital voucher.
- parseGBIF_duplicates = TRUE indicates whether there are duplicates associated with the master digital voucher using the unique collection event key.

- parseGBIF_num_duplicates = number of duplicates associated with the master digital voucher using the unique collection event key.

- parseGBIF_non_groupable_duplicates = TRUE indicates where the collection event key is incomplete.

- parseGBIF_duplicates_grouping_status = duplicate grouping status as: “groupable”, “not groupable: no recordNumber”, “not groupable: no recordedBy” or “not groupable: no recordedBy and no recordNumber”.

- parseGBIF_unidentified_sample = if unique collection event has taxonomic identification.

- parseGBIF_sample_taxon_name = scientific name chosen as taxonomic identification for unique collection event.
- parseGBIF_sample_taxon_name_status = indicates the selection status of the binomial taxon attributed to the unique collection event as: “identified”, “divergent identifications”, or “not identified”.
- parseGBIF_number_taxon_names = number of scientific names found in duplicates of unique collection event.
- parseGBIF_useful_for_spatial_analysis = whether the coordinates are useful for spatial analysis.
- parseGBIF_decimalLatitude = latitude in decimal degrees.
- parseGBIF_decimalLongitude = longitude in decimal degrees.

- parseGBIF_dataset_result = indicates the datasets resulting from the parseGBIF classification with: usable data, unusable data and their duplicates.

- parseGBIF_wcvp_plant_name_id = Information from the World Checklist of Vascular Plants (WCVP) database on the taxon binomial attributed to the unique collection event.

- parseGBIF_wcvp_taxon_rank = as in previous.

- parseGBIF_wcvp_taxon_status = as in previous.
- parseGBIF_wcvp_family = as in previous.
- parseGBIF_wcvp_taxon_name = as in previous.
- parseGBIF_wcvp_taxon_authors = as in previous.
- parseGBIF_wcvp_reviewed = as in previous.

```
digital_voucher <- select_digital_voucher(occ = occ,
                                                    occ_gbif_issue =
gbif_issue$occ_gbif_issue,

occ_wcvp_check_name = names.checked$occ_wcvp_check_name,

occ_collectorsDictionary = collectorsDictionary$occ_collectorsDictionary,
                                                    silence = TRUE)
```

```r
names(digital_voucher)
```

```
## [1] "occ_digital_voucher" "occ_results"
```

```r
NROW(digital_voucher$occ_digital_voucher)
```

```
## [1] 34099
```

```r
colnames(digital_voucher$occ_digital_voucher)
```

```
##  [1] "Ctrl_gbifID"
##  [2] "Ctrl_bibliographicCitation"
##  [3] "Ctrl_language"
##  [4] "Ctrl_institutionCode"
##  [5] "Ctrl_collectionCode"
##  [6] "Ctrl_datasetName"
##  [7] "Ctrl_basisOfRecord"
##  [8] "Ctrl_catalogNumber"
##  [9] "Ctrl_recordNumber"
## [10] "Ctrl_recordedBy"
## [11] "Ctrl_occurrenceStatus"
## [12] "Ctrl_eventDate"
## [13] "Ctrl_year"
## [14] "Ctrl_month"
## [15] "Ctrl_day"
## [16] "Ctrl_habitat"
## [17] "Ctrl_fieldNotes"
## [18] "Ctrl_eventRemarks"
## [19] "Ctrl_countryCode"
## [20] "Ctrl_stateProvince"
## [21] "Ctrl_municipality"
## [22] "Ctrl_county"
## [23] "Ctrl_locality"
## [24] "Ctrl_level0Name"
## [25] "Ctrl_level1Name"
## [26] "Ctrl_level2Name"
## [27] "Ctrl_level3Name"
## [28] "Ctrl_identifiedBy"
## [29] "Ctrl_dateIdentified"
## [30] "Ctrl_scientificName"
## [31] "Ctrl_decimalLatitude"
## [32] "Ctrl_decimalLongitude"
## [33] "Ctrl_taxonRank"
## [34] "Ctrl_nameRecordedBy_Standard"
## [35] "Ctrl_recordNumber_Standard"
## [36] "Ctrl_key_family_recordedBy_recordNumber"
## [37] "Ctrl_geospatial_quality"
## [38] "Ctrl_verbatim_quality"
## [39] "Ctrl_moreInformativeRecord"
## [40] "Ctrl_coordinates_validated_by_gbif_issue"
```

```
## [41] "wcvp_plant_name_id"
## [42] "wcvp_taxon_rank"
## [43] "wcvp_taxon_status"
## [44] "wcvp_family"
## [45] "wcvp_taxon_name"
## [46] "wcvp_taxon_authors"
## [47] "wcvp_reviewed"
## [48] "wcvp_searchedName"
## [49] "wcvp_searchNotes"
## [50] "parseGBIF_digital_voucher"
## [51] "parseGBIF_duplicates"
## [52] "parseGBIF_num_duplicates"
## [53] "parseGBIF_non_groupable_duplicates"
## [54] "parseGBIF_duplicates_grouping_status"
## [55] "parseGBIF_unidentified_sample"
## [56] "parseGBIF_sample_taxon_name"
## [57] "parseGBIF_sample_taxon_name_status"
## [58] "parseGBIF_number_taxon_names"
## [59] "parseGBIF_useful_for_spatial_analysis"
## [60] "parseGBIF_decimalLatitude"
## [61] "parseGBIF_decimalLongitude"
## [62] "parseGBIF_dataset_result"
## [63] "parseGBIF_wcvp_plant_name_id"
## [64] "parseGBIF_wcvp_taxon_rank"
## [65] "parseGBIF_wcvp_taxon_status"
## [66] "parseGBIF_wcvp_family"
## [67] "parseGBIF_wcvp_taxon_name"
## [68] "parseGBIF_wcvp_taxon_authors"
## [69] "parseGBIF_wcvp_reviewed"

head(digital_voucher$occ_digital_voucher[,50:69])

##   parseGBIF_digital_voucher parseGBIF_duplicates
parseGBIF_num_duplicates
## 1                      TRUE                 TRUE
3
## 2                      TRUE                FALSE
1
## 3                      TRUE                 TRUE
3
## 4                      TRUE                 TRUE
4
## 5                      TRUE                FALSE
1
## 6                      TRUE                FALSE
1
##   parseGBIF_non_groupable_duplicates
parseGBIF_duplicates_grouping_status
## 1                              FALSE
groupable
```

```
## 2                                FALSE
groupable
## 3                                FALSE
groupable
## 4                                FALSE
groupable
## 5                                FALSE
groupable
## 6                                FALSE
groupable
##   parseGBIF_unidentified_sample parseGBIF_sample_taxon_name
## 1                          TRUE
## 2                          TRUE
## 3                          TRUE
## 4                          TRUE
## 5                          TRUE
## 6                          TRUE
##   parseGBIF_sample_taxon_name_status parseGBIF_number_taxon_names
## 1                       unidentified                            0
## 2                       unidentified                            0
## 3                       unidentified                            0
## 4                       unidentified                            0
## 5                       unidentified                            0
## 6                       unidentified                            0
##   parseGBIF_useful_for_spatial_analysis parseGBIF_decimalLatitude
## 1                                  TRUE                  7.246133
## 2                                  TRUE                  5.622622
## 3                                  TRUE                  6.540725
## 4                                  TRUE                  6.241858
## 5                                  TRUE                  5.986938
## 6                                  TRUE                  6.093710
##   parseGBIF_decimalLongitude parseGBIF_dataset_result
## 1                  -76.43966                 unusable
## 2                  -77.42660                 unusable
## 3                  -76.63161                 unusable
## 4                  -77.42627                 unusable
## 5                  -77.33291                 unusable
## 6                  -77.28808                 unusable
##   parseGBIF_wcvp_plant_name_id parseGBIF_wcvp_taxon_rank
## 1                                                  <NA>
## 2                                                  <NA>
## 3                                                  <NA>
## 4                                                  <NA>
## 5                                                  <NA>
## 6                                                  <NA>
##   parseGBIF_wcvp_taxon_status parseGBIF_wcvp_family
parseGBIF_wcvp_taxon_name
## 1                        <NA>                  <NA>
<NA>
## 2                        <NA>                  <NA>
```

```
<NA>
## 3                          <NA>                  <NA>
<NA>
## 4                          <NA>                  <NA>
<NA>
## 5                          <NA>                  <NA>
<NA>
## 6                          <NA>                  <NA>
<NA>
##   parseGBIF_wcvp_taxon_authors parseGBIF_wcvp_reviewed
## 1                         <NA>                    <NA>
## 2                         <NA>                    <NA>
## 3                         <NA>                    <NA>
## 4                         <NA>                    <NA>
## 5                         <NA>                    <NA>
## 6                         <NA>                    <NA>
```

```
digital_voucher$occ_digital_voucher$parseGBIF_dataset_result %>% unique()
```

```
## [1] "unusable"  "duplicate"
```

```
ind <- digital_voucher$occ_digital_voucher$parseGBIF_dataset_result ==
"useable"
NROW(digital_voucher$occ_digital_voucher[ind==TRUE,])
```

```
## [1] 0
```

```
ind <- digital_voucher$occ_digital_voucher$parseGBIF_dataset_result ==
"duplicate"
NROW(digital_voucher$occ_digital_voucher[ind==TRUE,])
```

```
## [1] 10428
```

```
ind <- digital_voucher$occ_digital_voucher$parseGBIF_dataset_result ==
"unusable"
NROW(digital_voucher$occ_digital_voucher[ind==TRUE,])
```

```
## [1] 23671
```

```
file.name <- 'parseGBIF_4_occ_digital_voucher.csv'
write.csv(digital_voucher$occ_digital_voucher,
            file.name,
            row.names = FALSE,
            fileEncoding = "UTF-8",
            na = "")
```

### 5. Export of results

For each unique collection event key, complete or incomplete, outputs will be generated a single unique collection event record. Where the unique collection event key is complete, a single unique collection event record will be created which combine information from duplicate records. 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 (where the unique collection event key is complete), 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 six data frames**:

- **all_data** All records processed, merged. To separate the records into three datasets by filtering parseGBIF_dataset_result field by"useable", "unusable" and "duplicates".
- **useable_data_merge** Merged useable dataset.
- **useable_data_raw** Raw useable dataset.
- **duplicates** Duplicates of unique collection events of useable and unusable datasets.
- **unusable_data_merge** Merged unusable dataset. It is NA if merge_unusable_data is FALSE.
- **unusable_data_raw** Raw unusable dataset.

```
results <- export_data(occ_digital_voucher_file = '',
                                occ_digital_voucher =
digital_voucher$occ_digital_voucher,
                                merge_unusable_data = TRUE,
                                silence = TRUE)


names(results)
```

```
## [1] "all_data"            "useable_data_merge"  "useable_data_raw"
## [4] "duplicates"          "unusable_data_merge" "unusable_data_raw"
```

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

```
## [1] 34099
```

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

```
## [1] 0
```

```
NROW(results$useable_data_raw)
```

```
## [1] 0
```

```
NROW(results$duplicates)
```

```
## [1] 10428
```

```
NROW(results$unusable_data_merge)
```

```
## [1] 23671
```

```
NROW(results$unusable_data_raw)
```

```
## [1] 23671
```

```
file.name <-  file.name <- 'parseGBIF_5_occ_all_data.csv'
write.csv(results$all_data,
              file.name,
              row.names = FALSE,
              fileEncoding = "UTF-8",
              na = "")


file.name <- 'parseGBIF_5_occ_useable_data_merge.csv'
write.csv(results$useable_data_merge,
              file.name,
              row.names = FALSE,
              fileEncoding = "UTF-8",
              na = "")


file.name <-  'parseGBIF_5_occ_useable_data_raw.csv'
write.csv(results$useable_data_raw,
              file.name,
              row.names = FALSE,
              fileEncoding = "UTF-8",
              na = "")


file.name <-  'parseGBIF_5_occ_duplicates.csv'
```

```r
write.csv(results$duplicates,
              file.name,
              row.names = FALSE,
              fileEncoding = "UTF-8",
              na = "")


file.name <-  'parseGBIF_5_occ_unusable_data_merge.csv'
write.csv(results$unusable_data_merge,
              file.name,
              row.names = FALSE,
              fileEncoding = "UTF-8",
              na = "")

file.name <-  'parseGBIF_5_occ_unusable_data_raw.csv'
write.csv(results$unusable_data_raw,
              file.name,
              row.names = FALSE,
              fileEncoding = "UTF-8",
              na = "")
```

### 6. ParseGBIF summary

parseGBIF_summary function return a list with four data frames:

- **parseGBIF_general_summary**
- **parseGBIF_merged_fields_summary** = frequency of merge actions in the fields
- **parseGBIF_merged_fields_summary_useable_data** = frequency of merge actions on fields in the usable dataset
- **parseGBIF_merged_fields_summary_unusable_data** = frequency of merge actions on fields in the unusable dataset. It is NA if merge_unusable_data is FALSE

Detail of the **general summary:**

- total number of records
- total number of unique collection events
- total number of duplicates records of unique collection events
- total number of useable records
- total number of useable records / consensus on identification
- total number of useable records / divergent identifications
- total number of unusable records
- total number of unusable records / unidentified
- total number of unusable records / not suitable for geospatial analysis
- total unique collection events containing merged fields

```r
summ <- parseGBIF_summary(parseGBIF_all_data = results$all_data)

names(summ)
```

```
## [1] "parseGBIF_general_summary"
## [2] "parseGBIF_merge_fields_summary"
## [3] "parseGBIF_merge_fields_summary_useable_data"
## [4] "parseGBIF_merge_fields_summary_unusable_data"
```

```r
head(summ$parseGBIF_general_summary)
```

```
##                                                                 question value
## 1                                            total number of records 34099
## 2                          total number of unique collection events 23671
## 3 total number of duplicates records of unique collection events 10428
## 4                                    total number of useable records     0
## 5  total number of useable records / consensus on identification     0
## 6    total number of useable records / divergent identifications     0
##
condition
## 1
all lines
## 2
where parseGBIF_digital_voucher = TRUE
## 3
where parseGBIF_dataset_result = 'duplicate'
## 4
where parseGBIF_dataset_result = 'useable'
## 5                where parseGBIF_dataset_result = 'useable' AND
parseGBIF_sample_taxon_name_status = 'identified'
## 6 where parseGBIF_dataset_result = 'useable' AND
parseGBIF_sample_taxon_name_status = 'divergent identifications'
```

```r
head(summ$parseGBIF_merge_fields_summary)
```

```
##                                      question value
## 1       Ctrl_habitat : total merge actions   418
## 2 Ctrl_stateProvince : total merge actions   210
## 3      Ctrl_locality : total merge actions   177
## 4  Ctrl_municipality : total merge actions   153
## 5    Ctrl_fieldNotes : total merge actions    78
## 6          Ctrl_year : total merge actions    42
##                             condition
## 1 frequency of Ctrl_verbatim_quality
## 2 frequency of Ctrl_verbatim_quality
## 3 frequency of Ctrl_verbatim_quality
## 4 frequency of Ctrl_verbatim_quality
## 5 frequency of Ctrl_verbatim_quality
## 6 frequency of Ctrl_verbatim_quality
```

```r
head(summ$parseGBIF_merge_fields_summary_useable_data)
```

```
## [1] question  value     condition
## <0 linhas> (ou row.names de comprimento 0)
```

```
file.name <-  file.name <- 'parseGBIF_6_general_summary.csv'
write.csv(summ$parseGBIF_general_summary,
              file.name,
              row.names = FALSE,
              fileEncoding = "UTF-8",
              na = "")

file.name <-  file.name <- 'parseGBIF_6_merge_fields_summary.csv'
write.csv(summ$parseGBIF_merge_fields_summary,
              file.name,
              row.names = FALSE,
              fileEncoding = "UTF-8",
              na = "")

file.name <-  file.name <-
'parseGBIF_6_merge_fields_summary_useable_data.csv'
write.csv(summ$parseGBIF_merge_fields_summary_useable_data,
              file.name,
              row.names = FALSE,
              fileEncoding = "UTF-8",
              na = "")

file.name <-  file.name <-
'parseGBIF_6_merge_fields_summary_unusable_data.csv'
write.csv(summ$parseGBIF_merge_fields_summary_unusable_data,
              file.name,
              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 key = ACHATOCARPACEAE_ZARDINI_5592

```
index <- results$all_data$Ctrl_key_family_recordedBy_recordNumber %in%
results$all_data$Ctrl_key_family_recordedBy_recordNumber[83]

print('merged fields')
```

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

```
print(jsonlite::fromJSON(results$all_data$parseGBIF_merged_fields[index==
TRUE]))
```

```
## $Ctrl_habitat
## [1] "1839829110"
##
```

```
## $Ctrl_habitat
## [1] "1839829088"
```

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

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

```
print(jsonlite::fromJSON(results$all_data$parseGBIF_duplicates_map[index==TRUE]))
```

```
## $Ctrl_gbifID
## [1] "2268998464" "1839829110" "1839829088"
##
## $Ctrl_scientificName
## [1] "Oncoba cuneatoacuminata De Wild. Hul & Breteler"
## [2] "Lindackeria cuneatoacuminata De Wild. Gilg"
## [3] "Lindackeria cuneatoacuminata De Wild. Gilg"
##
## $Ctrl_recordedBy
## [1] "L. Achten"   "Achten L.T." "Achten L.T."
##
## $Ctrl_recordNumber
## [1] "Achten 399A" "399A"        "399B"
##
## $Ctrl_institutionCode
## [1] "BR"      "MeiseBG" "MeiseBG"
##
## $Ctrl_collectionCode
## [1] "Tropicos" "BR"       "BR"
##
## $Ctrl_datasetName
## [1] "Tropicos"                       "Meise Botanic Garden Herbarium"
## [3] "Meise Botanic Garden Herbarium"
##
## $Ctrl_datasetName
## [1] "Tropicos"                       "Meise Botanic Garden Herbarium"
## [3] "Meise Botanic Garden Herbarium"
##
## $Ctrl_habitat
## [1] ""        "Savanna" "Savanna"
```

- 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$all_data$parseGBIF_freq_duplicate_or_missing_data[index==TRUE]))
```

```
## $Ctrl_gbifID
##        value freq
## 1 1839829088    1
## 2 1839829110    1
## 3 2268998464    1
##
## $Ctrl_scientificName
##                                                value freq
## 1      Lindackeria cuneatoacuminata (De Wild.) Gilg    2
## 2 Oncoba cuneatoacuminata (De Wild.) Hul & Breteler    1
##
## $Ctrl_recordedBy
##         value freq
## 1 Achten L.T.    2
## 2   L. Achten    1
##
## $Ctrl_recordNumber
##         value freq
## 1        399A    1
## 2        399B    1
## 3 Achten 399A    1
##
## $Ctrl_identifiedBy
##   value freq
## 1 empty    3
##
## $Ctrl_dateIdentified
##   value freq
## 1 empty    3
##
## $Ctrl_institutionCode
##     value freq
## 1 MeiseBG    2
## 2      BR    1
##
## $Ctrl_collectionCode
##      value freq
## 1       BR    2
## 2 Tropicos    1
##
## $Ctrl_datasetName
##                             value freq
## 1 Meise Botanic Garden Herbarium    2
## 2                       Tropicos    1
##
## $Ctrl_datasetName
##                             value freq
## 1 Meise Botanic Garden Herbarium    2
## 2                       Tropicos    1
##
```

```
## $Ctrl_language
##   value freq
## 1 empty    3
##
## $wcvp_plant_name_id
##   value freq
## 1 empty    3
##
## $wcvp_taxon_rank
##   value freq
## 1 empty    3
##
## $wcvp_taxon_status
##   value freq
## 1 empty    3
##
## $wcvp_family
##   value freq
## 1 empty    3
##
## $wcvp_taxon_name
##   value freq
## 1 empty    3
##
## $wcvp_taxon_authors
##   value freq
## 1 empty    3
##
## $wcvp_reviewed
##   value freq
## 1 empty    3
##
## $wcvp_searchNotes
##       value freq
## 1 Not found    3
##
## $Ctrl_fieldNotes
##   value freq
## 1 empty    3
##
## $Ctrl_year
##   value freq
## 1  1915    3
##
## $Ctrl_stateProvince
##      value freq
## 1 Kinshasa    1
## 2    empty    2
##
## $Ctrl_municipality
```

```
##   value freq
## 1 empty    3
##
## $Ctrl_locality
##         value freq
## 1 Leopoldstad    3
##
## $Ctrl_countryCode
##   value freq
## 1    CD    3
##
## $Ctrl_eventDate
##         value freq
## 1 -1712102400    2
## 2  1915-10-01    1
##
## $Ctrl_habitat
##     value freq
## 1 Savanna    2
## 2   empty    1
##
## $Ctrl_level0Name
##                                value freq
## 1 Democratic Republic of the Congo    3
##
## $Ctrl_level1Name
##      value freq
## 1 Kinshasa    3
##
## $Ctrl_level2Name
##      value freq
## 1 Kinshasa    3
##
## $Ctrl_level3Name
##   value freq
## 1 empty    3
```