parseGBIF Manual

Pablo Hendrigo Alves de Melo[[1]](#footnote-1)

Nadia Bystriakova[[2]](#footnote-2)

Alexandre Monro[[3]](#footnote-3)

2024-03-01

# parseGBIF Manual

The parseGBIF package is designed to repackage [Global Biodiversity Information Facility - GBIF](https://www.gbif.org/) 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](https://github.com/pablopains/parseGBIF). To install parseGBIF, run

devtools::install\_github("pablopains/parseGBIF")

Please site parseGBIF as:

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

## 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](gbif.org)

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…  
## # 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).

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](https://www.gbif.org/article/5i3CQEZ6DuWiycgMaaakCo/gbif-infrastructure-data-processing)
* [An enumeration of validation rules for single occurrence records](https://gbif.github.io/gbif-api/apidocs/org/gbif/api/vocabulary/OccurrenceIssue.html)

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>

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/collectorDictionary/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)  
  
names(digital\_voucher)

## [1] "occ\_digital\_voucher" "occ\_results"

NROW(digital\_voucher$occ\_digital\_voucher)

## [1] 34099

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'  
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

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"

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'

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

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

1. Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, [pablopains@yahoo.com.br](mailto:pablopains@yahoo.com.br) [↑](#footnote-ref-1)
2. Natural History Museum, London, [n\_bystriakova@yahoo.com](mailto:n_bystriakova@yahoo.com) [↑](#footnote-ref-2)
3. Royal Botanic Gardens, Kew, [a.monro@kew.org](mailto:a.monro@kew.org) [↑](#footnote-ref-3)