Buscar y recuperar datos del Global Biodiverity Information Facilty (GBIF) utilizando R

Primer Seminario Grupo Ecoinformática de la AEET

Carlos Lara Romero (carlos.lara.romero@gmail.com)

09 febrero, 2022

Contents

rgbif es un paquete de R para buscar y recuperar datos del Global Biodiverity Information Facilty (GBIF). https://docs.ropensci.org/rgbif/ https://github.com/ropensci/rgbif/

1. Preparación sesión

Instalar rgbif desde CRAN

```
install.packages("rgbif")
```

O instalar la versión de desarrollo desde GitHub

```
remotes::install_github("ropensci/rgbif")
```

Cargar rgbif en la consola de R

```
library("rgbif")
```

2. Búsqueda de nombres de taxones

```
taxrank()
```

```
[1] "kingdom"       "phylum"        "class"         "order"
[5] "family"        "genus"         "species"       "subspecies"
[9] "infraspecific"
```

La nomenclatura científica en gbif es una maraña de nombres aceptados, nombres antiguos y sinonimias.

La función *name_lookup()* realiza una búsqueda que abarca el nombre científico y vernáculo, la descripción de la especie, la distribución y la clasificación completa en todos los usos del nombre de todas o algunas listas. Los resultados se ordenan por relevancia, ya que esta búsqueda suele devolver muchos resultados.

Búsqueda de una especie

```
tax <- name_lookup(query = 'Cedrus atlantica', rank="species")
names(tax)
```

```
[1] "meta"        "data"        "facets"      "hierarchies" "names"
```

```
tax$meta
```

```
# A tibble: 1 x 4
  offset limit endOfRecords count
   <int> <int> <lgl>        <int>
1      0    82 TRUE            82
```

```
tax$data
```

```
# A tibble: 82 x 42
        key scientificName  datasetKey   nubKey parentKey parent kingdom phylum
      <int> <chr>           <chr>          <int>     <int> <chr>  <chr>   <chr>
 1 145954271 Cedrus atlanti~ 3f5e930b-5~ 5284702 157137955 Cedrus Plantae Pinop~
 2 178920114 Cedrus atlanti~ 6b6b2923-0~ 5284702 178920103 Cedrus Viridi~ Strep~
 3 103018874 Cedrus atlanti~ fab88965-e~ 5284702 103018872 Cedrus Viridi~ Strep~
 4 180134137 Cedrus atlanti~ dbaa27eb-2~ 5284702 180134102 Cedrus Plantae Trach~
 5 131817678 Cedrus atlanti~ 91bc4859-0~ 5284702 131817666 Pinac~ Plantae <NA>
 6   8074435 Cedrus atlanti~ d7dddbf4-2~      NA   2685742 Cedrus Plantae Trach~
 7 127710596 Cedrus atlanti~ 91bc4859-0~ 5284702 131817666 Pinac~ Plantae <NA>
 8 127697674 Cedrus atlanti~ 91fecd78-0~ 5284702 163427218 Cedrus Plantae Pinop~
 9 105340171 Cedrus atlanti~ 046bbc50-c~ 8074435 105339285 Cedrus Plantae <NA>
10 180134154 Cedrus atlanti~ dbaa27eb-2~ 5284702    153727 Cedrus Plantae Trach~
# ... with 72 more rows, and 34 more variables: genus <chr>, species <chr>,
#   kingdomKey <int>, phylumKey <int>, classKey <int>, genusKey <int>,
#   speciesKey <int>, canonicalName <chr>, authorship <chr>, nameType <chr>,
#   taxonomicStatus <chr>, rank <chr>, origin <chr>, numDescendants <int>,
#   numOccurrences <int>, habitats <chr>, nomenclaturalStatus <chr>,
#   threatStatuses <chr>, synonym <lgl>, class <chr>, order <chr>,
#   family <chr>, orderKey <int>, familyKey <int>, taxonID <chr>, ...
```

```
head(tax$data)
```

```
# A tibble: 6 x 42
     key scientificName datasetKey  nubKey parentKey parent kingdom phylum genus
   <int> <chr>          <chr>        <int>     <int> <chr>  <chr>   <chr>  <chr>
```

```
1 1.46e8 Cedrus atlant~ 3f5e930b-~ 5284702 157137955 Cedrus Plantae Pinop~ Cedr~
2 1.79e8 Cedrus atlant~ 6b6b2923-~ 5284702 178920103 Cedrus Viridi~ Strep~ Cedr~
3 1.03e8 Cedrus atlant~ fab88965-~ 5284702 103018872 Cedrus Viridi~ Strep~ Cedr~
4 1.80e8 Cedrus atlant~ dbaa27eb-~ 5284702 180134102 Cedrus Plantae Trach~ Cedr~
5 1.32e8 Cedrus atlant~ 91bc4859-~ 5284702 131817666 Pinac~ Plantae <NA>   <NA>
6 8.07e6 Cedrus atlant~ d7dddbf4-~      NA   2685742 Cedrus Plantae Trach~ Cedr~
# ... with 33 more variables: species <chr>, kingdomKey <int>, phylumKey <int>,
#   classKey <int>, genusKey <int>, speciesKey <int>, canonicalName <chr>,
#   authorship <chr>, nameType <chr>, taxonomicStatus <chr>, rank <chr>,
#   origin <chr>, numDescendants <int>, numOccurrences <int>, habitats <chr>,
#   nomenclaturalStatus <chr>, threatStatuses <chr>, synonym <lgl>,
#   class <chr>, order <chr>, family <chr>, orderKey <int>, familyKey <int>,
#   taxonID <chr>, constituentKey <chr>, publishedIn <chr>, ...
```

Búsqueda de un género

```
cedrus <- name_lookup(query = 'Cedrus', rank="genus")
```

```
# A tibble: 67 x 37
      key scientificName datasetKey nubKey parentKey parent kingdom phylum order
    <int> <chr>          <chr>       <int>     <int> <chr>  <chr>   <chr>  <chr>
 1 1.90e8 Cedrus         91ad2b95-~ 2.69e6 190445000 Pinac~ Plantae Trach~ Pina~
 2 1.92e8 Cedrus         4dd32523-~ 2.69e6 192232334 Pinac~ Plantae Trach~ Pina~
 3 1.72e8 Cedrus         7ddf754f-~ 2.69e6 175150317 Pinac~ Plantae Trach~ Pina~
 4 1.63e8 Cedrus         91fecd78-~ 2.69e6 163427216 Pinac~ Plantae Pinop~ Pina~
 5 1.57e8 Cedrus         3f5e930b-~ 2.69e6 157137906 Pinop~ Plantae Pinop~ <NA>
 6 1.91e8 Cedrus         4dd32523-~ 2.69e6 190595271 Pinac~ Plantae Trach~ Pina~
 7 1.87e8 Cedrus         1ec61203-~ 2.69e6 187332763 Pinac~ Plantae Trach~ Pina~
 8 1.90e8 Cedrus         16c3f9cb-~ 2.69e6 190351215 Pinac~ <NA>    <NA>   Coni~
 9 1.92e8 Cedrus         19491596-~ 2.69e6 192024830 PINAC~ PLANTAE TRACH~ PINO~
10 1.92e8 Cedrus         d9a4eedb-~ 2.69e6 192375432 Pinac~ Plantae <NA>   <NA>
# ... with 57 more rows, and 28 more variables: family <chr>, genus <chr>,
#   kingdomKey <int>, phylumKey <int>, classKey <int>, orderKey <int>,
#   familyKey <int>, genusKey <int>, canonicalName <chr>, authorship <chr>,
#   nameType <chr>, taxonomicStatus <chr>, rank <chr>, origin <chr>,
#   numDescendants <int>, numOccurrences <int>, habitats <chr>,
#   nomenclaturalStatus <lgl>, threatStatuses <chr>, synonym <lgl>,
#   class <chr>, constituentKey <chr>, taxonID <chr>, acceptedKey <int>, ...
```

Para ayudarnos, *rgbif* cuenta con la función *name_backbone* que permite buscar nombres en la taxonomía de GBIF. La función devuelve un data.frame para la coincidencia de taxones sugerida. Un solo taxón con muchas columnas.

```
tax.backbone<-name_backbone(name='Cedrus atlantica', rank='species', kingdom='plants')
tax.backbone
```

```
# A tibble: 1 x 22
  usageKey scientificName       canonicalName  rank  status confidence matchType
*    <int> <chr>                <chr>          <chr> <chr>       <int> <chr>
1  5284702 Cedrus atlantica (E~ Cedrus atlant~ SPEC~ ACCEP~         99 EXACT
# ... with 15 more variables: kingdom <chr>, phylum <chr>, order <chr>,
#   family <chr>, genus <chr>, species <chr>, kingdomKey <int>,
#   phylumKey <int>, classKey <int>, orderKey <int>, familyKey <int>,
#   genusKey <int>, speciesKey <int>, synonym <lgl>, class <chr>
```

```
key<-tax.backbone$usageKey
```

Una estrategia útil es comprobar el listado de nombres aceptados en GBIF y usar su identificador único (key) para realizar la búsqueda de presencias. Para ello podemos usar la funicón *name_suggest*. Esta función nos proporciona un servicio rápido y sencillo que devuelve hasta 20 usos del nombre haciendo coincidir el prefijo con el nombre científico. Los resultados se ordenan por relevancia.

```
tax.suggest <- name_suggest(q="Cedrus atlantica", rank='species')
tax.suggest$data
```

```
# A tibble: 3 x 3
       key canonicalName    rank
     <int> <chr>            <chr>
1  8074435 Cedrus atlantica SPECIES
2  5284702 Cedrus atlantica SPECIES
3 10152153 <NA>             SPECIES
```

```
key2<-tax.suggest$data$key[1]
```

La función *occ_count* permite estimar la cantidad de ocurrencia. Es muy útil en el proceso de diseño de las descargas de GBIF.

```
occ_count(taxonKey=key)
```

```
[1] 6426
```

```
occ_count(taxonKey=key2)
```

```
[1] 111
```

3. Búsqueda y descarga de ocurrencias

La función *Occ_search()* permite buscar ocurrencias de GBIF. Por defecto se descargan más de 80 columnas de información. Accedemos a los datos a través de la ranura *data*.

```
cedrus<-occ_search(taxonKey=key, limit=500) #Limitamos la descarga a 500 datos para agilizar la descarg
cedrus$meta
```

```
$offset
[1] 300

$limit
[1] 200

$endOfRecords
[1] FALSE

$count
[1] 6426
```

```
dim(cedrus$data)
```

```
[1] 500 123
```

```
head(cedrus$data, n=10L)
```

```
# A tibble: 10 x 123
   key        scientificName  decimalLatitude decimalLongitude issues datasetKey
   <chr>      <chr>                     <dbl>            <dbl> <chr>  <chr>
 1 3447369515 Cedrus atlanti~            48.7             10.1  "cdro~ 6ac3f774-~
 2 3447391533 Cedrus atlanti~            48.9             10.3  "cdro~ 6ac3f774-~
 3 3447398380 Cedrus atlanti~            48.7             10.1  "cdro~ 6ac3f774-~
 4 3455872878 Cedrus atlanti~            41.8             12.5  "cdro~ 50c9509d-~
 5 3456555154 Cedrus atlanti~            34.0           -117.   "cdro~ 50c9509d-~
 6 3457140880 Cedrus atlanti~            43.7              3.08 "cdro~ 50c9509d-~
 7 3457176937 Cedrus atlanti~           -37.6            144.   "cdro~ 50c9509d-~
 8 3031654191 Cedrus atlanti~            48.1             -1.54 "cdro~ 50c9509d-~
 9 3117779404 Cedrus atlanti~            NA               NA    ""     50c9509d-~
10 3118262413 Cedrus atlanti~            47.0              3.15 "cdro~ 50c9509d-~
# ... with 117 more variables: publishingOrgKey <chr>, installationKey <chr>,
#   publishingCountry <chr>, protocol <chr>, lastCrawled <chr>,
#   lastParsed <chr>, crawlId <int>, hostingOrganizationKey <chr>,
#   basisOfRecord <chr>, occurrenceStatus <chr>, taxonKey <int>,
#   kingdomKey <int>, phylumKey <int>, classKey <int>, orderKey <int>,
#   familyKey <int>, genusKey <int>, speciesKey <int>, acceptedTaxonKey <int>,
#   acceptedScientificName <chr>, kingdom <chr>, phylum <chr>, order <chr>, ...
```

En el siguiente enlace puedes consultar una referencia fácil de leer de los términos recomendados actualmente por elGrupo de Mantenimiento de Darwin Core.

https://dwc.tdwg.org/terms/

Podemos seleccionar los campos que deseamos descargar

```
campos<-c("key","scientificName", "decimalLatitude","decimalLongitude",
          "issues", "datasetKey", "basisOfRecord","year")
cedrus<-occ_search(taxonKey =  key, fields= campos, limit = 5000)
dim(cedrus$data)
```

```
[1] 5000    8
```

```
names(cedrus$data)
```

```
[1] "key"              "datasetKey"       "basisOfRecord"    "scientificName"
[5] "decimalLongitude" "decimalLatitude"  "year"             "issues"
```

```
head(cedrus$data)
```

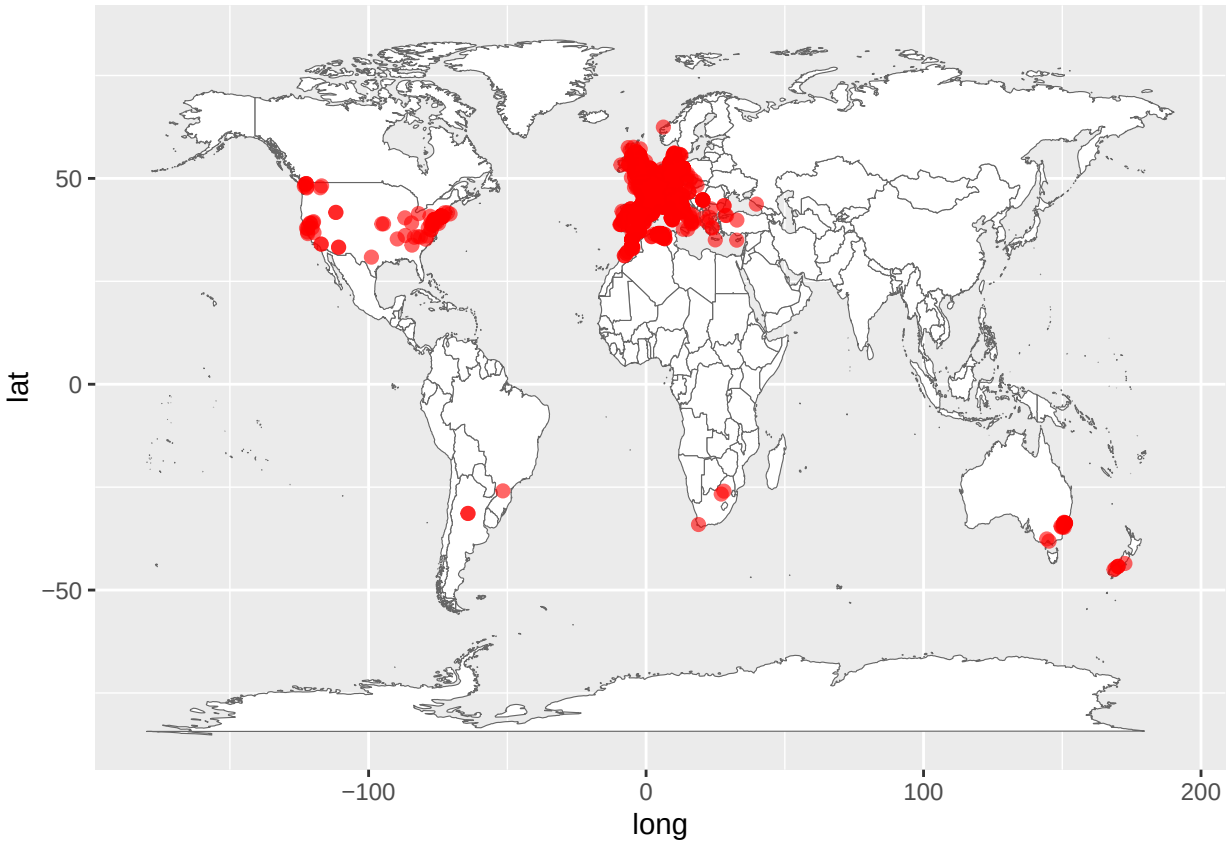
```
# A tibble: 6 x 8
  key        datasetKey basisOfRecord scientificName decimalLongitude decimalLatitude
  <chr>      <chr>      <chr>         <chr>                      <dbl>           <dbl>
```

```
1 3447369515 6ac3f774-~ HUMAN_OBSERV~ Cedrus atlant~            10.1             48.7
2 3447391533 6ac3f774-~ HUMAN_OBSERV~ Cedrus atlant~            10.3             48.9
3 3447398380 6ac3f774-~ HUMAN_OBSERV~ Cedrus atlant~            10.1             48.7
4 3455872878 50c9509d-~ HUMAN_OBSERV~ Cedrus atlant~            12.5             41.8
5 3456555154 50c9509d-~ HUMAN_OBSERV~ Cedrus atlant~          -117.              34.0
6 3457140880 50c9509d-~ HUMAN_OBSERV~ Cedrus atlant~             3.08            43.7
# ... with 2 more variables: year <int>, issues <chr>
```

Visualizar los datos en un mapa es sencillo

```
#Mapa con ggplot2
library(ggplot2)
world = map_data("world") #library maps

ggplot(world, aes(long, lat)) +
geom_polygon(aes(group = group), fill = "white",
             color = "gray40", size = .2)+
geom_jitter(data = cedrus$data,
aes(decimalLongitude, decimalLatitude), alpha=0.6,
            size = 2, color = "red")
```



occ_search() permite vectorizar los argumentos para hacer descargas múltiples. El resultado es una lista anidada.

```
occ <- occ_search(scientificName = c("Cedrus atlantica", "Eucalyptus nitens"),
                  fields= campos,
                  limit = 500,
                  hasCoordinate = TRUE)
is.list(occ)
```

```
[1] TRUE
```

```
names(occ)
```

```
[1] "Cedrus atlantica"  "Eucalyptus nitens"
```

```
head(occ$`Cedrus atlantica`$data)
```

```
# A tibble: 6 x 8
  key        datasetKey basisOfRecord scientificName decimalLongitude decimalLatitude
  <chr>      <chr>      <chr>         <chr>                      <dbl>           <dbl>
1 3447369515 6ac3f774-~ HUMAN_OBSERV~ Cedrus atlant~            10.1             48.7
2 3447391533 6ac3f774-~ HUMAN_OBSERV~ Cedrus atlant~            10.3             48.9
3 3447398380 6ac3f774-~ HUMAN_OBSERV~ Cedrus atlant~            10.1             48.7
4 3455872878 50c9509d-~ HUMAN_OBSERV~ Cedrus atlant~            12.5             41.8
5 3456555154 50c9509d-~ HUMAN_OBSERV~ Cedrus atlant~          -117.              34.0
6 3457140880 50c9509d-~ HUMAN_OBSERV~ Cedrus atlant~             3.08            43.7
# ... with 2 more variables: year <int>, issues <chr>
```

```
head(occ$`Eucalyptus nitens`$data)
```

```
# A tibble: 6 x 8
  key        datasetKey basisOfRecord scientificName decimalLongitude decimalLatitude
  <chr>      <chr>      <chr>         <chr>                      <dbl>           <dbl>
1 3097394389 50c9509d-~ HUMAN_OBSERV~ Eucalyptus ni~            173.            -41.8
2 3455916257 50c9509d-~ HUMAN_OBSERV~ Eucalyptus ni~            -73.2           -36.8
3 3325678345 50c9509d-~ HUMAN_OBSERV~ Eucalyptus ni~            170.            -43.9
4 2013746984 50c9509d-~ HUMAN_OBSERV~ Eucalyptus ni~            173.            -41.3
5 2437847974 4ce8e3f9-~ PRESERVED_SP~ Eucalyptus ni~            146.            -37.7
6 3130685708 37beb6ea-~ HUMAN_OBSERV~ Eucalyptus ni~            146.            -37.8
# ... with 2 more variables: year <int>, issues <chr>
```

```
occ.df<-rbind(occ$`Cedrus atlantica`$data, occ$`Eucalyptus nitens`$data) #Sólo funciona si comparten lo
occ.df
```

```
# A tibble: 1,000 x 8
   key        datasetKey      basisOfRecord scientificName      decimalLongitude
   <chr>      <chr>           <chr>         <chr>                          <dbl>
 1 3447369515 6ac3f774-d9fb-~ HUMAN_OBSERV~ Cedrus atlantica (~            10.1
 2 3447391533 6ac3f774-d9fb-~ HUMAN_OBSERV~ Cedrus atlantica (~            10.3
 3 3447398380 6ac3f774-d9fb-~ HUMAN_OBSERV~ Cedrus atlantica (~            10.1
 4 3455872878 50c9509d-22c7-~ HUMAN_OBSERV~ Cedrus atlantica (~            12.5
 5 3456555154 50c9509d-22c7-~ HUMAN_OBSERV~ Cedrus atlantica (~          -117.
 6 3457140880 50c9509d-22c7-~ HUMAN_OBSERV~ Cedrus atlantica (~             3.08
```

```
 7 3457176937 50c9509d-22c7-~ HUMAN_OBSERV~ Cedrus atlantica (~           144.
 8 3031654191 50c9509d-22c7-~ HUMAN_OBSERV~ Cedrus atlantica (~            -1.54
 9 3118262413 50c9509d-22c7-~ HUMAN_OBSERV~ Cedrus atlantica (~             3.15
10 3118394422 50c9509d-22c7-~ HUMAN_OBSERV~ Cedrus atlantica (~             8.56
# ... with 990 more rows, and 3 more variables: decimalLatitude <dbl>,
#   year <int>, issues <chr>
```

```
dim(occ.df)
```

```
[1] 1000    8
```

Se puede utilizar un bucle para descargar un archivo por cada especie. Luego podemos unirlos en un único data.frame. Esta manera de proceder permite generar un código más reproducible. Además, tiene la ventaja de que a la vez podemos descargar la cita y almacenarla en una carpeta específica. GBIF exige que se cite escrupulosamente los datos de presencia.

[https://www.gbif.org/citation-guidelines]

```
path<-"outputs/data/rgbif/occ_search/"
dir.create(path)
```

```
Warning in dir.create(path): 'outputs\data\rgbif\occ_search' already exists
```

```
splist<-c("Cedrus atlantica", "Eucalyptus nitens")

for (i in splist){
dat.search<- occ_search(scientificName=i,fields=campos, limit=50)
dat<- dat.search$data
saveRDS(dat, paste0(path,i,"_down.Rdata"))
cit<-gbif_citation(dat.search)
saveRDS(cit, paste0(path,i,"_citation",".Rdata"))
}
```

Podríamos modificar el bucle anterior para construir un único arreglo de datos y almacenar un único archivo en el disco duro. He manejado la información mediante objetos de R. En mi experiencia personal es una buena opción ya que limitas las posibilidades de que los datos se lean mal al importar/exportar y además suelen ocupar menos espacio en el disco duro que los archivos csv o txt. En todo caso, sería fácil modificar el código anterior cambiando la función *saveRDS* por *write.table*, *write.csv*, etc.

Generar un único objeto de R con la información descargada y almacenada de manera conjunta en una carpeta del PC.

```
files<-list.files(path="outputs/data/rgbif/descarga_ejemplo", full.names=TRUE)
files
```

```
[1] "outputs/data/rgbif/descarga_ejemplo/Cedrus atlantica_citation.Rdata"
[2] "outputs/data/rgbif/descarga_ejemplo/Cedrus atlantica_down.Rdata"
[3] "outputs/data/rgbif/descarga_ejemplo/Eucalyptus nitens_citation.Rdata"
[4] "outputs/data/rgbif/descarga_ejemplo/Eucalyptus nitens_down.Rdata"
```

```
files<-stringr::str_subset(files, "_down.Rdata") #Selecciono datos con ocurrencias

my.df<-data.frame(NULL)
for (i in files) {
temp<-readRDS(i)
my.df<-rbind(my.df,temp) #Columnas coinciden. Usar bind_rows(dplyr) si las columnas son distintas.
}
dim(my.df)
```

```
[1] 100   8
```

```
head(my.df)
```

```
# A tibble: 6 x 8
  key        datasetKey basisOfRecord scientificName decimalLongitude decimalLatitude
  <chr>      <chr>      <chr>         <chr>                      <dbl>           <dbl>
1 3447369515 6ac3f774-~ HUMAN_OBSERV~ Cedrus atlant~            10.1             48.7
2 3447391533 6ac3f774-~ HUMAN_OBSERV~ Cedrus atlant~            10.3             48.9
3 3447398380 6ac3f774-~ HUMAN_OBSERV~ Cedrus atlant~            10.1             48.7
4 3455872878 50c9509d-~ HUMAN_OBSERV~ Cedrus atlant~            12.5             41.8
5 3456555154 50c9509d-~ HUMAN_OBSERV~ Cedrus atlant~          -117.              34.0
6 3457140880 50c9509d-~ HUMAN_OBSERV~ Cedrus atlant~             3.08            43.7
# ... with 2 more variables: year <int>, issues <chr>
```

4. Las funciones *occ_download* como alternativa para grandes descargas de datos

rgbif sólo puede descargar datos de especies con menos de 100.000 ocurrencias. Si alimentamos *occ_search()* con más datos obtendremos un error. Podemos hacer un bucle sencillo para comprobar si las especies de estudio superan los 100.000 datos.

```
presencias<- as.vector(NULL)
for (i in splist){
tax.backbone<-name_backbone(name=i, rank='species', kingdom='plants')
key<-tax.backbone$usageKey
dat.search<- occ_count(taxonKey = key)
presencias <-c(presencias,dat.search)
}
```

Podemos visualizar los resultados y unirlos al listado de especies

```
presencias<-data.frame(splist,presencias)
presencias
```

```
             splist presencias
1  Cedrus atlantica       6426
2 Eucalyptus nitens        940
```

En los casos con más de 100.000 datos. Una forma de sortear esta limitación es aplicar un bucle que descargue los datos de manera secuencial. Por ejemplo, podemos decargar los datos de manera independiente para cada año, país de procedencia etc...

Sin embargo, la familia de funciones *occ_download()* son más apropiadas para solicitudes de datos más grandes. Los autores de rgbif aconsejan esta segunda opcion al considerarla más reproducible y más fácil una vez se aprende el funcionamiento.

La primera función importante es precisamente *occ_download()* Con ella puedes especificar qué consulta quieres realizar. Las interfaces de*occ_search()* y *occ_download()* son diferentes. Ten en cuenta que sólo puedes realizar 3 descargas simultáneas con *occ_download()*, así que planifica bien.

4.1 Registrarse en GBIF

Para poder utilizar occ_download tenemos que tener registrado en GBIF un nombre de usuario y una contraseña e email asociados.

[https://www.gbif.org/es/] (https://www.gbif.org/es/)

```
user<-escribe usuario gbif
pwd<-escribe contraseña gbif
email<-escribe email gbif
```

4.2 Iniciar una descarga

En lugar de pasar parámetros como *hasCoordinate = TRUE* en *occ_search*, para las descargas construimos consultas utilizando cualquiera de las funciones *pred* (predicados). Esto permite realizar consultas mucho más complejas que las que se pueden hacer con *occ_search*.

Se pueden utilizar operadores distintos de = (igual a). Algunas de las funcioes **pred** más comunes.

- pred: equals
- pred_lt: lessThan
- pred_lte: lessThanOrEquals
- pred_gt: greaterThan
- pred_gte: greaterThanOrEquals
- pred_not: not
- pred_like: like

Consulta la documentación de los predicados aquí: https://www.gbif.org/developer/occurrence#predicates

```
res <- occ_download(pred("taxonKey", 5284702), pred("hasCoordinate", TRUE), #Cedrus Atlantica, Sólo coo
                    user=user,
                    email=email,
                    pwd= pwd)
res
```

```
<<gbif download>>
  Username: carlos.lara.romero
  E-mail: carlos.lara.romero@gmail.com
  Format: DWCA
  Download key: 0133994-210914110416597
```

Con occ_download se trabaja de manera similar a una descarga desde la interfaz web. *occ_download* envía la solicitud a GBIF, ellos tienen que prepararla primero, luego cuando está hecha puedes descargarla.

Lo que *occ_download* devuelve son algunos metadatos útiles que informan sobre la descarga, y nos ayudan a comprobar y saber cuándo se ha realizado la descarga.

Comprobar el estado de la descarga

Para comprobar el estado de la descarga podemos pasar el objeto generado por *occ_download* a *occ_download_meta*

```
occ_download_meta(res)
```

```
<<gbif download metadata>>
  Status: PREPARING
  Format: DWCA
  Download key: 0133994-210914110416597
  Created: 2022-02-09T08:42:23.445+00:00
  Modified: 2022-02-09T08:42:23.445+00:00
  Download link: https://api.gbif.org/v1/occurrence/download/request/0133994-210914110416597.zip
  Total records: <NA>
  Request:
    type:  and
    predicates:
      > type: equals, key: TAXON_KEY, value: 5284702
      > type: equals, key: HAS_COORDINATE, value: true
```

Continua ejecutando occ_download_meta hasta que el valor del estado sea *SUCCEED* o *KILLED*.

4.3 Obtención de datos

Cuando cambie a *SUCCEED* se puede descargar los datos con *occ_download_get*

```
path<-"outputs/data/rgbif/occ_download/"
dir.create(path)
down.key <- occ_download_meta(res)$key
dat <- occ_download_get(key=down.key, path=path, overwrite = TRUE)
```

Para cargar los datos descargados en consola de R hay usar *occ_download_import*

```
dat.imp<-occ_download_import(dat)
```

4.4 Citando los datos de las descargas

Usando la función *gbif_citation()* podemos obtener citas para nuestras descargas. Esta funcón proporciona la citación global y las citas de cada conjunto de datos de manera independiente

```
cit<-gbif_citation(dat)
cit$download #Citación global
```

```
[1] "GBIF Occurrence Download https://doi.org/10.15468/dl.qjatxr Accessed from R via rgbif (https://gitl
```

```
length(cit$datasets) #157 conjunto de datos
```

```
[1] 156
```

```
cit$datasets [1] #Cita para el primer conjunto de datos
```

```
[[1]]
<<rgbif citation>>
   Citation: Merseyside BioBank (2020). Merseyside BioBank (unverified).
        Occurrence dataset https://doi.org/10.15468/iou2ld accessed via
        GBIF.org on 2022-02-09.. Accessed from R via rgbif
        (https://github.com/ropensci/rgbif) on 2022-02-09
   Rights: This work is licensed under a Creative Commons Attribution Non
        Commercial (CC-BY-NC) 4.0 License.
```

```
saveRDS(cit, paste0("outputs/data/rgbif/get/","dat2","_citation",".Rdata"))
```

4.5 Vectorizar búsquedas

Para descargar datos de varias especies necesitamos usar el predicado *pred_in*. Permite vectoriar y aplicar varios valores a una consulta

```
gbif_taxon_keys<-c(2977832,2977901,2977966,2977835,2977863)

multiple<-occ_download(
  pred_in("taxonKey", gbif_taxon_keys),
  format = "SIMPLE_CSV",
  user=user,pwd=pwd,email=email)
```

Para automatizar todo el proceso podemos crear un bucle para iterar mientras se está preparando nuetra solicitud. El bucle se detiene con éxito cuando se completa y *occ_download_meta* devuelve el resultado de "SUCCEED"

```
still_running <- TRUE
status_ping <- 30 #seconds

while (still_running) {
  meta <- occ_download_meta(multiple)
  status <- meta$status
  print(status)
  still_running <- status %in% c("PREPARING", "RUNNING")
  Sys.sleep(status_ping) # Suspender ejecución expresiones R durante intervalo de tiempo determinado.
}
```

```
[1] "RUNNING"
[1] "RUNNING"
[1] "SUCCEEDED"
```

```
occ_download_meta(multiple)
```

```
<<gbif download metadata>>
  Status: SUCCEEDED
  Format: SIMPLE_CSV
  Download key: 0134012-210914110416597
```

```
  Created: 2022-02-09T08:48:37.323+00:00
  Modified: 2022-02-09T08:49:19.522+00:00
  Download link: https://api.gbif.org/v1/occurrence/download/request/0134012-210914110416597.zip
  Total records: 392
  Request:
    type:  in
    predicates:
      > type: in, key: TAXON_KEY, value: 2977832,2977901,2977966,2977835,2977863
```

```
down.key <- occ_download_meta(multiple)$key
```

```
dat2<-occ_download_get(key=down.key, path=path,overwrite=TRUE)
dat2.imp<-occ_download_import(dat2)
unique(dat2.imp$scientificName)
```

```
 [1] "Abarema auriculata (Benth.) Barneby & J.W.Grimes"
 [2] "Abarema abbottii (Rose & Leonard) Barneby & J.W.Grimes"
 [3] "Abarema alexandri var. troyana (Urb.) Barneby & J.W.Grimes"
 [4] "Abarema alexandri var. alexandri"
 [5] "Abarema acreana (J.F.Macbr.) L.Rico"
 [6] "Abarema asplenifolia (Griseb.) Barneby & J.W.Grimes"
 [7] "Pithecellobium auriculatum Benth."
 [8] "Abarema alexandri (Urb.) Barneby & J.W.Grimes"
 [9] "Pithecellobium abbottii Rose & Leonard"
[10] "Pithecellobium alexandri (Urb.) Urb."
[11] "Pithecolobium asplenifolium Griseb."
[12] "Pithecellobium alexandri var. intermedium Urb."
[13] "Pithecellobium asplenifolium Griseb."
[14] "Pithecellobium alexandri var. troyanum Urb."
[15] "Pithecellobium jupunba var. alexandri Urb."
[16] "Pithecellobium asplenifolium subsp. mayarense Borhidi"
[17] "Pithecellobium malacotrichum Harms"
```

```
unique(dat2.imp$taxonKey)
```

```
 [1] 2977863 2977832 2977972 7225010 2977901 2977835 2977864 2977966 2977833
[10] 8075388 2977840 3796563 8429325 2977973 2977970 2977839 2977865
```

```
head(dat2.imp)
```

```
# A tibble: 6 x 50
  gbifID datasetKey occurrenceID kingdom phylum class order family genus species
  <int6> <chr>      <chr>        <chr>   <chr>  <chr> <chr> <chr>  <chr> <chr>
1   3.e9 ec76ce73-~ uniamazonia~ Plantae Trach~ Magn~ Faba~ Fabac~ Abar~ Abarem~
2   3.e9 d8cd16ba-~ MN305139     Plantae Trach~ Magn~ Faba~ Fabac~ Abar~ Abarem~
3   3.e9 d8cd16ba-~ MN305149     Plantae Trach~ Magn~ Faba~ Fabac~ Abar~ Abarem~
4   3.e9 d8cd16ba-~ MN305146     Plantae Trach~ Magn~ Faba~ Fabac~ Abar~ Abarem~
5   3.e9 d8cd16ba-~ MN305145     Plantae Trach~ Magn~ Faba~ Fabac~ Abar~ Abarem~
6   3.e9 d8cd16ba-~ MN990419     Plantae Trach~ Magn~ Faba~ Fabac~ Abar~ Abarem~
# ... with 40 more variables: infraspecificEpithet <chr>, taxonRank <chr>,
#   scientificName <chr>, verbatimScientificName <chr>,
```

```
#   verbatimScientificNameAuthorship <chr>, countryCode <chr>, locality <chr>,
#   stateProvince <chr>, occurrenceStatus <chr>, individualCount <int>,
#   publishingOrgKey <chr>, decimalLatitude <dbl>, decimalLongitude <dbl>,
#   coordinateUncertaintyInMeters <dbl>, coordinatePrecision <lgl>,
#   elevation <dbl>, elevationAccuracy <dbl>, depth <lgl>, ...
```

```
cit2<-gbif_citation(dat2)
dat2<-occ_download_import(dat2) # Import into R
dat2[sample(1:nrow(dat2),5),c("species","decimalLatitude","decimalLongitude","countryCode","issue")]
```

```
# A tibble: 5 x 5
  species              decimalLatitude decimalLongitude countryCode issue
  <chr>                          <dbl>            <dbl> <chr>       <chr>
1 Abarema auriculata                NA               NA BR          RECORDED_DA~
2 Abarema auriculata                NA               NA BR          CONTINENT_I~
3 Abarema auriculata                NA               NA BR          COLLECTION_~
4 Abarema abbottii                  NA               NA DO          COLLECTION_~
5 Abarema asplenifolia              NA               NA CU          TAXON_MATCH~
```

```
saveRDS(cit2, paste0(path,"dat2","_citation",".Rdata"))
```

5. Conjunto de datos derivados

GBIF tiene una herramienta para generar conjunto de datos derivados. Un conjunto de datos derivado es un registro citable (con un DOI único) que representa un conjunto de datos que no existe como una descarga convencional y no filtrada de GBIF.org.

https://www.gbif.org/es/derived-dataset%5Dhttps://www.gbif.org/es/derived-dataset

A través del enalce se accede a la herramienta de GBIF para generar el DOI a la que sólo necesitamos aportar: i) una lista de conjuntos de datos contribuyentes, ii) el recuento de registros de cada conjunto de daots y, en aras de la reproducibilidad, la url del repositorio público en el que se aloja el conjunto de datos derivado (figshare, dryad etc.). Los datos contribuyentes se pueden identificar con el campo datasetKey.

En resumen, hay que generar un archivo de este tipo:

- datasetKey1 25 records
- datasetKey2 545 records
- datasetKey3 200 records

GBIF recomienda el uso de conjuntos de datos derivados para citar los datos obtenidos mediante llamadas síncronas a la API como las utilizadas por rgbif en occ_data() y occ_search(). Pero aunque se utilice la función occ_download() para generar una única descarga de datos es posible que durante el proceso de limpieza y filtrado se eliminen registros de conjunto de datos completos (datasets). Por ello, es una herramienta muy útil para generar DOIs específicos para nuestros datos que permitan citar exactamente los datos utilizados.

6. Limpieza de datos

Los datos proporcionados por agregadores de grandes datos como GBIF son muy valiosos para la investigación. Sin embargo, existen algunos problemas relacionados con la calidad de los datos, sobre todo porque

estos datos se componen de una variedad de métodos de recolección diferentes, de diferentes fuentes y son digitalizados y editados por varias personas y algoritmos en diferentes momentos en el tiempo y el espacio.

Durante el proceso de indexación sobre los datos brutos, GBIF añade problemas y banderas a los registros con problemas comunes de calidad de datos. Esta información queda almacenada en el campo *issues*.

[https://data-blog.gbif.org/post/issues-and-flags/]https://data-blog.gbif.org/post/issues-and-flags/

```
unique(cedrus$data$issues)
```

```
 [1] "cdround,gass84"                  "cdround"
 [3] ""                                "fpwktinv"
 [5] "cudc,osiic"                      "cdround,cudc,osiic"
 [7] "colmafu,inmafu"                  "cdround,osiic"
 [9] "gass84,preswcd,colmafu,inmafu"   "osiic"
[11] "gass84,fpwktinv"                 "cdround,cudc"
[13] "gass84"                          "indci,fpwktinv"
[15] "colmafu"                         "colmano"
[17] "cudc,inmano"                     "bri,cudc"
[19] "colmano,inmafu"                  "inmano"
[21] "gass84,colmafu,inmafu"           "cdround,gass84,fpwktinv"
[23] "cdround,osu"                     "cudc"
[25] "gass84,colmafu"                  "bri,cdround,cudc"
[27] "ambinst"                         "gass84,inmano"
[29] "cdround,colmano,inmafu"          "cdround,diffown"
[31] "gass84,ambinst"                  "gass84,ambcol"
[33] "cdround,inmafu"                  "incomis"
[35] "cdround,gass84,cdpi"             "gass84,osiic,colmafu,inmafu"
[37] "osiic,colmafu,inmafu"            "gass84,cdpi,osiic,colmano,inmafu"
[39] "ambcol,inmafu"                   "gass84,osiic,colmano,inmafu"
[41] "gass84,diffown"                  "cudc,fpwktinv"
```

Con la función *gbif_issues()* se pueden consultar todos los tipos de problemas

```
head(gbif_issues())
```

```
   code                              issue
1   bri            BASIS_OF_RECORD_INVALID
2   ccm         CONTINENT_COUNTRY_MISMATCH
3   cdc CONTINENT_DERIVED_FROM_COORDINATES
4 conti                  CONTINENT_INVALID
5  cdiv                 COORDINATE_INVALID
6 cdout            COORDINATE_OUT_OF_RANGE
                                                                                                     desc
1 The given basis of record is impossible to interpret or seriously different from the recommended vocal
2                                                       The interpreted continent and country do not ma
3                   The interpreted continent is based on the coordinates, not the verbatim string infor
4                                                                          Uninterpretable continent values
5                                       Coordinate value given in some form but GBIF is unable to interpr
6                                         Coordinate has invalid lat/lon values out of their decimal max
        type
1 occurrence
2 occurrence
3 occurrence
```

```
4 occurrence
5 occurrence
6 occurrence
```

Filtrar datos por este campo es sencillo:

```
cedrus2<-cedrus$data[cedrus$data$issues!="bri",] #Elminamos ocurrencias sin origen del dato preciso
dim(cedrus2)
```

```
[1] 5000    8
```

Otro campo importante es *basisOfRecords* ya que nos permite filtra por la fuente de datos

```
table(cedrus2$basisOfRecord)
```

```
 HUMAN_OBSERVATION    LIVING_SPECIMEN    MATERIAL_SAMPLE        OBSERVATION
              4219                 29                  2                 18
        OCCURRENCE PRESERVED_SPECIMEN
                31                701
```

```
cedrus2<-cedrus2[cedrus2$basisOfRecord=="HUMAN_OBSERVATION",]
table(cedrus2$basisOfRecord)
```

```
HUMAN_OBSERVATION
             4219
```

Las posibilidades de filtrado de los datos son muy numerosas. Entrar en detalle en este proceso se escapa del objetivo de este seminario. En todo caso, existen algunos paquetes en R que nos proporcionan una línea de trabajo sobre cómo limpiar los registros de ocurrencia recuperados de GBIF (o de cualquier otra base de datos): *scrubr* (https://github.com/ropensci/scrubr), *biogeo* (https://markrobertson.co.za/biogeo/) o *CoordinateCleaner* (https://ropensci.github.io/CoordinateCleaner/articles/Cleaning_GBIF_data_with_CoordinateCleaner.html)

Los paquetes *scrubr* y *taxize* (https://github.com/ropensci/taxize) son dos muy buenas alernativas para la limpieza taxonómica