geocleaMT: An R package to cleaning geographical data from electronic biodatabases

Viviana R-Alarcon and Daniel R Miranda-Esquivel (dmiranda@uis.edu.co)

2016-01-15

geocleaMT is a cleaning protocol for distributional data and an automated tool to provide the necessary data mining process before any diversity analysis. The protocol takes into account the most common mistakes found in distributional records and attempts to minimize its possibility, increasing users confidence in their data. All functions use as input a basic table with at least three columns species/ decimalLatitude/ decimalLongitude, and save the results in the same format after a specific task is performed. Additionally, all functions return a descriptive report about the process made, that can be also saved as data frame object. Although, the functions could be used in the suggested pipelines, each function is suited to perform an specific task or a set of them, given the package a high level of customization. See R-Alarcon and Miranda-Esquivel (submitted).

1.0 A worked example for *geocleaMT* package (R-Alarcon and Miranda-Esquivel, submitted)

1.1 Preliminars

First, we remove everything to start with a clean session. If there is an open graphic device, we close it:

```
rm(list = ls())

if (dev.cur() != 1){
    dev.off()}
```

Then, we get the latest version of ***geocleaMT*** from GitHub repository, and load the library:

```
## You might need to uncomment these lines

#install.packages(c('devtools', 'plyr', 'rgbif', 'RCurl', 'vegan', 'modeest',
#                   'maptools', 'data.table', 'RJSONIO', 'reshape2'),
#                   dependencies = T, repos = 'https://www.icesi.edu.co/CRAN/')

library('devtools')
install_github('alarconvv/geocleaMT', dependencies = TRUE)
library('geocleaMT')

library('plyr')
library('rgbif')
library('RCurl')
library('vegan')
library('modeest')
library('maptools')
library('data.table')
library('RJSONIO')
library('reshape2')
```

Now we are ready to work. We must change our working directory to a suitable place, assuming you want to store your data/results in a directory called '~/geocleanMTtest/' in your home directory; first, we create it (if exists there is no problem, just some warnings that we do not show):

```
Sys.chmod(paths = '~/', mode = '7777', use_umask = TRUE)

dir.create(path = '~/geocleanMTtest/', showWarnings = FALSE, recursive = FALSE, mode = '7777')

pathToFiles <- '~/geocleanMTtest/'

dir.create(path = pathToFiles, showWarnings = FALSE)

setwd(pathToFiles)

# To check everything is fine:

getwd()
```

```
## [1] "/home/vra/geocleanMTtest"
```

We create the directory structure to save the results in separated files. This function will create a folder for each function used.

```
pathStructure(path.dir = '')
```

If the protocol is performed for several groups, for example the classes: Mammalia, Reptilia, etc., the user could assign these group names in the group argument as c('Mammalia','Reptilia', 'Amphibia') to separate the results by group.

```
pathStructure(path.dir = '', group = 'Mammalia')
```

To obtain the occurrences reported for the Biogeographic Choco in the Global Biodiversity Information Facility (GBIF) (Telenius2011), we will go to GBIF website and follow the next pathway:

```
>  Data
>  Explore occurences (Fig. 1)
```

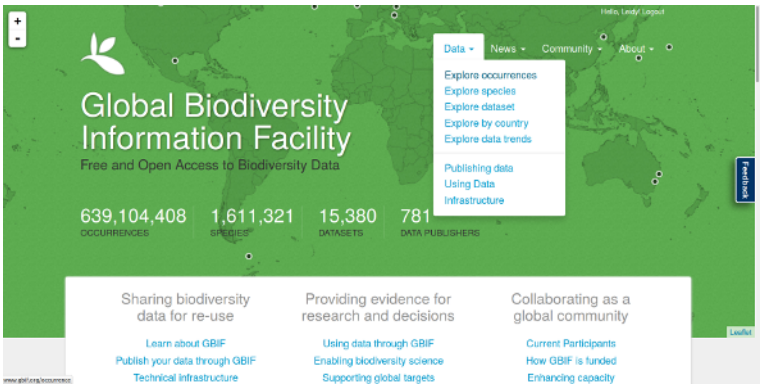


Figure 1. Accessing to explore occurences in GBIF

> Georeferenced records
> Add a filter
> Location (Fig. 2)

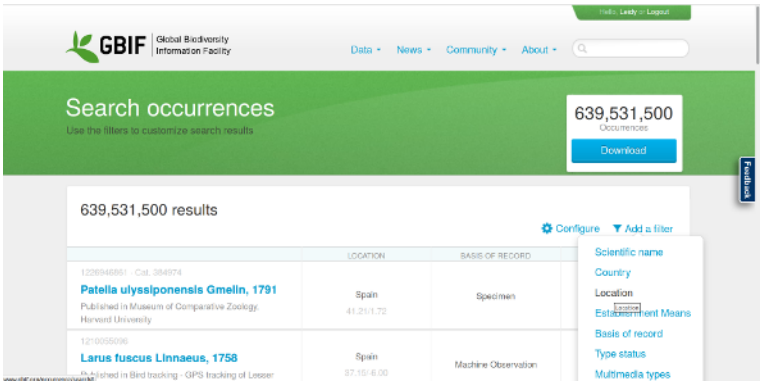


Figure 2. Add the Location Filter

> Location Box
> Draw tools: Undefined (Fig. 3)

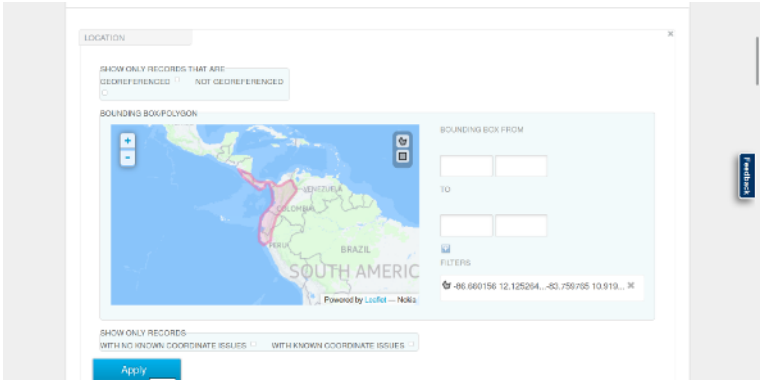


Figure 3. Drawing the polygon.

Then, we draw the area of interest (in our case, The Biogeographic Choco form (*sensu lato:* Hernandez 1992), see Fig. 3), and click on:

> Apply

We will download a specific taxon using again the 'add filter' tool. For example, we will download all occurrences of mammals reported by GBIF for The Biogeographic Choco (*sensu lato:* Hernandez, 1992) (Fig. 3-4), as was used in the example in R-Alarcon and Miranda-Esquivel (submitted).

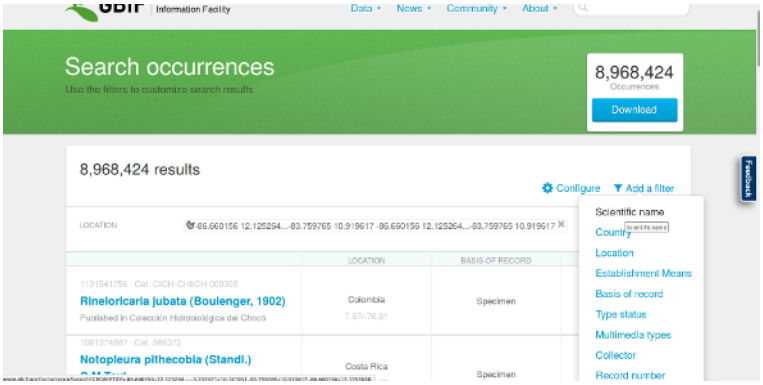


Figure 4. Add the scientific name filter.

For this example, we will use a selection of 35 species from the example in R-Alarcon and Miranda-Esquivel (submitted). These 35 species are the best candidates because they present all possible issues we can face in a standard dataset.

```
> Add Filter
> Scientific name (Fig. 5)
```

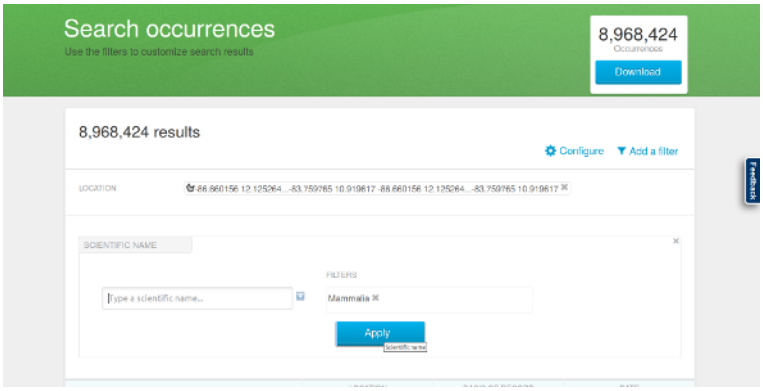


Figure 5. Applying the filter.

```
> Download (Fig. 6)
```

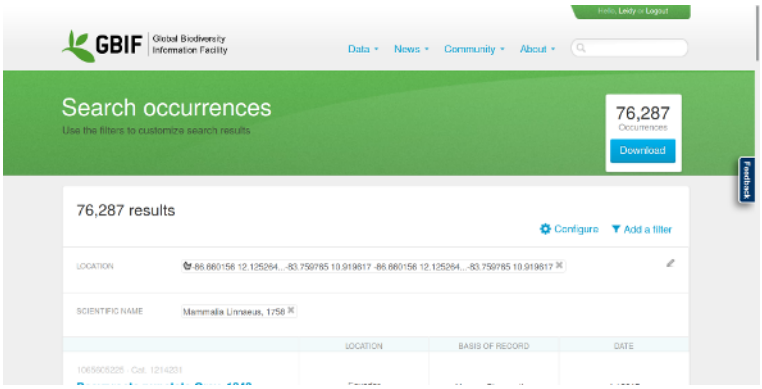


Figure 6. Click on Download.

The best file format to download is as a zip file, because it brings the complete information in the Darwin Core Standard (Wieczorek et al. 2012). Keep in mind that you must be registered at GBIF website.

After download the zip file, we have a folder with some information files. The file which have the occurrences data is the called 'occurrences.text'. If you want to read it, note that the separator is TAB.

1.2 The process

All the files imported or exported from ***geocleaMT*** follow the headers proposed by the Darwin Core standard (Wieczorek et al. 2012), as it is used by GBIF.

There are some functions that need to know the header names or the position of the column in the header. The code data('ID_DarwinCore') has the headers and the order as these are imported from GBIF.

```
data('ID_DarwinCore')

str(ID_DarwinCore)
```

```
## 'data.frame':    224 obs. of  2 variables:
##  $ ID        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ HeaderName: Factor w/ 224 levels "abstract","acceptedNameUsage",..: 76 1 4 5 6 7 8 13 14 18 ...
```

```
knitr::kable(head(ID_DarwinCore, 4L), format = 'markdown')
```

ID	HeaderName
1	gbifID
2	abstract
3	accessRights
4	accrualMethod

2.0 Get 35 Mammalian species distributed in the Biogeograpic Choco: Pipeline 1

We can get the example file from the GitHub repository in the example folder, or we can write it from the data(example) object.

```
# we will get the example file from data(Example).
# If you downloaded it from GBIF, the file will in txt format.

data(Example)

dir.create(path = 'data/', showWarnings = FALSE, recursive = FALSE, mode = '7777')

readAndWrite(action = 'write', path = 'data/', frmt = 'saveTXT',
             name = 'Example.txt', object = Example)
```

We can see the structure, and some basic statistics of the input data, including the species' names.

```
str(Example)
```

```
## 'data.frame':    8397 obs. of  14 variables:
##  $ gbifID           : int  665729802 665785591 665796068 665818285 665818376 665818972 665831992 665
##  $ spatial          : logi  NA NA NA NA NA NA ...
##  $ type             : Factor w/ 8 levels "","Evento","Imagen estática",..: 6 6 6 6 6 6 6 6 6 6 ...
##  $ decimalLatitude  : num  10.58 8.37 10.3 -3.43 -0.15 ...
##  $ decimalLongitude : num  -83.5 -72.3 -84.7 -79.2 -80.1 ...
##  $ disposition      : Factor w/ 4 levels "","En colección",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ elevation        : num  3 NA 1400 1500 NA NA 1200 130 1350 2200 ...
##  $ elevationAccuracy: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ issue            : Factor w/ 31 levels "","COORDINATE_REPROJECTED",..: 29 9 21 21 16 9 21 9 9 9 .
##  $ familyKey        : int  9366 9621 9366 5510 5510 5510 9621 9366 5311 5510 ...
##  $ speciesKey       : int  2433270 2436636 2433148 2438780 2439270 2439169 5219591 2433178 2433573 24
##  $ species          : Factor w/ 34 levels "Aotus lemurinus",..: 2 7 9 20 25 31 1 5 24 31 ...
##  $ genericName      : Factor w/ 31 levels "Aotus","Artibeus",..: 2 5 6 17 22 28 1 4 21 28 ...
##  $ lastParsed       : Factor w/ 350 levels "","2014-05-29T13:40Z",..: 61 60 61 60 60 60 60 60 60 60
```

```
head(Example, 1L)
```

```
##      gbifID spatial           type decimalLatitude decimalLongitude
## 1 665729802      NA PhysicalObject        10.58333        -83.51667
##   disposition elevation elevationAccuracy
## 1                   3               NA
##                                           issue familyKey speciesKey
## 1 TAXON_MATCH_HIGHERRANK;GEODETIC_DATUM_ASSUMED_WGS84      9366    2433270
##              species genericName        lastParsed
## 1 Artibeus lituratus    Artibeus 2015-05-21T14:03Z
```

```
levels(Example$species)
```

```
##  [1] "Aotus lemurinus"          "Artibeus lituratus"
##  [3] "Artibeus watsoni"         "Ateles geoffroyi"
##  [5] "Carollia castanea"        "Carollia colombiana"
##  [7] "Cebus albifrons"          "Cebus capucinus"
##  [9] "Centurio senex"           "Diplomys labilis"
## [11] "Echinoprocta rufescens"   "Euryoryzomys nitidus"
## [13] "Felis silvestris"         "Heteromys teleus"
## [15] "Lasiurus cinereus"        "Liomys adspersus"
## [17] "Micoureus alstoni"        "Molossus pretiosus"
## [19] "Monodelphis adusta"       "Mus musculus"
## [21] "Myotis riparius"          "Nasua narica"
## [23] "Neonycteris pusilla"      "Potos flavus"
## [25] "Rattus rattus"            "Saccopteryx antioquensis"
## [27] "Saguinus oedipus"         "Sigmodon peruanus"
## [29] "Sturnira luisi"           "Tamandua mexicana"
## [31] "Thomasomys baeops"        "Tonatia saurophila"
## [33] "Tylomys mirae"            "Urocyon cinereoargenteus"
```

Now, we read the initial data set. As the file to load has fewer than 100.000 occurrences we use the ***readDbR*** function that is executed into the R platform. The function can use the columns names or the number of the ID assigned in data('ID_DarwinCore').

```
# Load the txt file

readDbR <- readDbR(data          = 'Example.txt',
                   path.data     = 'data/' ,
                   cut.col       = c('gbifID', 'decimalLatitude', 'decimalLongitude',
                                     'elevation', 'speciesKey', 'species'),
                   delt.undeterm = TRUE,
                   save.name     = 'out.readDbR',
                   wrt.frmt      = 'saveRDS',
                   save.in       = 'readDBR/Mammalia/')
```

We get a table from the ***readDbR*** function with the basic information as: initial occurrences, final occurrences and total species. When the **delt.undeterm** parameter is 'TRUE', the records without species taxonomy will be deleted, then the initial occurrences and final occurrences will be different, in this case all occurrences have species assigned.

```
knitr::kable(readDbR, format = 'markdown')
```

	Initial.Occurr	Final.Occurr	Total.sp
Initial.tab	8397	8397	34

The output file will be saved in 'readDBR/Mammalia/' path:

```
out.readDbR <- readAndWrite(action = 'read', path = 'readDBR/Mammalia/',
                            frmt = 'readRDS', name = 'out.readDbR')


# The structure and format of out.readDbBash will be the same than the input
# file 'Example.txt'

str(out.readDbR)
```

```
## 'data.frame':    8397 obs. of  6 variables:
##  $ gbifID          : chr  "665729802" "665785591" "665796068" "665818285" ...
##  $ decimalLatitude : chr  "10.58333" "8.36667" "10.3" "-3.433" ...
##  $ decimalLongitude: chr  "-83.51667" "-72.3" "-84.7" "-79.183" ...
##  $ elevation       : chr  "3" NA "1400" "1500" ...
##  $ speciesKey      : chr  "2433270" "2436636" "2433148" "2438780" ...
##  $ species         : chr  "Artibeus lituratus" "Cebus albifrons" "Centurio senex" "Mus musculus" ...
```

```
knitr::kable(head(out.readDbR), format = 'markdown')
```

gbifID	decimalLatitude	decimalLongitude	elevation	speciesKey	species
665729802	10.58333	-83.51667	3	2433270	Artibeus lituratus
665785591	8.36667	-72.3	NA	2436636	Cebus albifrons
665796068	10.3	-84.7	1400	2433148	Centurio senex
665818285	-3.433	-79.183	1500	2438780	Mus musculus
665818376	-0.15	-80.06667	NA	2439270	Rattus rattus
665818972	-0.3	-78.46667	NA	2439169	Thomasomys baeops

We will save the information as a table to record the cleaning process. The ***pathStructure*** function created all the folders to follow the protocol, among those, the folder *'tables'* to save this table of information.

```
readAndWrite(action = 'write', frmt = 'saveTXT', path = 'tables/',
             name = 'readDbR_example.txt', object = readDbR)
```

When the file is larger than 100.000 occurrences, it could be difficult to read using ***readDbR***. If this is the case, we can use the ***readDbBash*** function to read large files. This process will be four times faster than with ***readDbR***.

NOTE: As a bash process, the **cut.col** argument works with the ID number of the headers of the Darwin Core standard, see the number assigned in data('ID_DarwinCore').

In the case of a file downloaded from GBIF, the ID of Darwin core headers c(1, 78, 79, 200, 218, 219) correspond to: 'gbifID', 'decimalLatitude', 'decimalLongitude', 'elevation', 'speciesKey', 'species'. In this example the columns to use are c(1, 4, 5, 7, 11, 12).

```
system <- Sys.info()["sysname"]
if (!system == "Windows") {
readDbBash <- readDbBash(data          = 'Example.txt',
                         path.data     = 'data/',
                         cut.col       = c(1, 4, 5, 7, 11, 12),
                         delt.undeterm = T,
                         save.name     = 'out.readDbBash',
                         wrt.frmt      = 'saveRDS',
                         save.in       = 'readDBBash/Mammalia/')
}
```

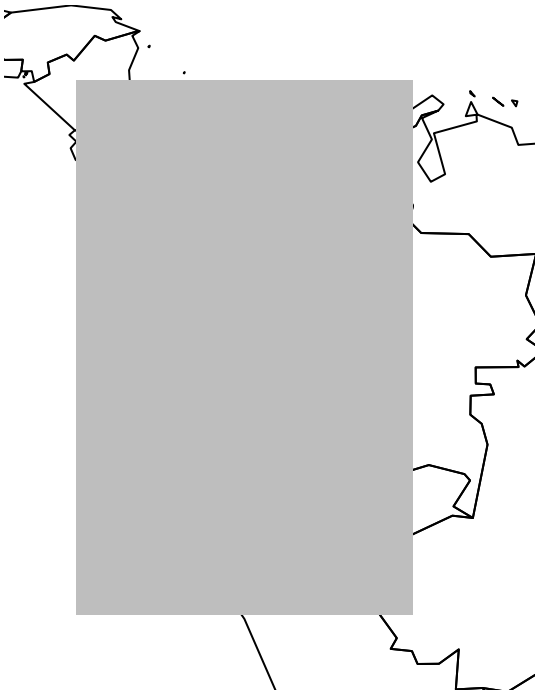
```
## Your OS is Linux. Please, be sure you have R version 3.2.1 or upper
```

Again, we write the output as a table to track the whole process.

```
system <- Sys.info()["sysname"]
if (!system == "Windows") {
readAndWrite(action = 'write', frmt = 'saveTXT', path = 'tables/',
             name = 'readDbBash_example.txt', object = readDbBash)
}
```

As our objective is to get species with occurrences into the elevation range 0 - 1000 MASL, then we can assign the elevation data to each coordinate from the elevation database 'Altitudes' (The elevation data was compiled from the Google Maps Elevation API). The average error is ~ 60m for 0-1000 MASL and ~ 150m for >1000 MASL (Fig. 7).

```
data('Altitudes')

data(wrld_simpl)

coordinates(Altitudes) <- Altitudes[, c('decimalLongitude', 'decimalLatitude')]

par(oma = c(0,0,0,0), mar = c(0,0,0,0) + 0.1)

plot(wrld_simpl, ylim = c(-10,15), xlim = c(-83,-73))

points(Altitudes, pch = '.', col = 'grey', cex = 0.01 )
```



```
# Figure 7. Coordinates with elevation data in data('Altitudes')
```

We can assign the elevation using the ***assignAltitude*** function. If the resolution in the elevation database used is 0.1, we should assign 1 in the **round.coord** argument. For example if the coordinate is Lat: 5.3456, Long: -74.2345 and the database is Lat: 5.3, Long: -74.2, elev: 856 MASL. Then the original coordinates will be rounded to Lat: 5.3, Long: -74.2 and 856 MASL will be assigned as the approximate elevation.

```
data('Altitudes')

assignAlt <- assignElevation(data = 'out.readDbR',
                             path.data = 'readDBR/Mammalia/',
                             rd.frmt = 'readRDS' ,
                             elevations.db = Altitudes ,
                             round.coord = 1,
                             wrt.frmt = 'saveRDS',
                             save.assigned.in = 'assignElevation/Mammalia/alt.assig/',
                             save.unassigned.in = 'assignElevation/Mammalia/alt.unassig/')

readAndWrite(action = 'write', frmt  = 'saveTXT' , path = 'tables/',
             name = 'assignAltitude_example.txt', object = assignAlt)
```

```
knitr::kable(head(assignAlt), format = 'markdown')
```

Occurrences	Elev.Assigned	Elev.Unassigned
8397	7945	452

If there are unassigned points, we can assign the altitude using another elevation database or Google Maps Elevation API. When there are occurrences without elevation assigned from the ***assignAltitude*** function,

we can find them in the file: 'alt.unassigned', in the path of **save.unassigned.in** argument. Thus, we can use the ***elevFromGg*** function to get the altitude directly from Google Maps Elevation API. This process will be slower, and the API has a restriction, where you can only download around 2500 elevation points every 24 hours. More information about restrictions in: developers.Google.com.

```
elevFromGg(data               = 'alt.unassigned',
           rd.frmt            = 'readRDS',
           path               = 'assignElevation/Mammalia/alt.unassig/',
           API.Key            = NULL,
           starts.in          = 1 ,
           round.coord        = 1 ,
           save.name          = 'alt.assigned',
           wrt.frmt           = 'saveRDS',
           save.assigned.in   = 'elevFromGg/Mammalia/alt.assig/' ,
           save.unassigned.in = 'elevFromGg/Mammalia/alt.unassig/' ,
           save.temp.in       = 'elevFromGg/Mammalia/')
```

NOTE: If we get the message 'Error in Call.API$results[[1]] : subscript out of bounds error', you must wait for 24 hours to access again the Google Maps Elevation API.

When we have both tables with assigned elevation ('alt.assig' from ***assignAltitude*** and 'alt.assig' from ***elevFromGg***) we can bind them.

```
# Read the tables

assign1 <- readRDS('assignElevation/Mammalia/alt.assig/alt.assigned')
assign2 <- readRDS('elevFromGg/Mammalia/alt.assig/alt.assigned')

# Delete the last column called resolution to bind them

assign2 <- assign2[, -7]

# bind by columns

newassign <- rbind(assign1, assign2)

# Write  the new table as an output file
readAndWrite(action = 'write', frmt = 'saveRDS' , path = 'elevFromGg/Mammalia/alt.assig/',
             name = 'completedElevation', object = newassign)
```

We need to get species with occurrences in the elevation range from 0 to 1000 MAMSL, then we use the ***cutRange*** function.

```
cutRange <- cutRange(data            = 'completedElevation' ,
                     path            = 'elevFromGg/Mammalia/alt.assig/',
                     rd.frmt         = 'readRDS',
                     range.from      = 0,
                     range.to        = 1000,
                     wrt.frmt        = 'saveRDS',
                     save.inside.in  = 'cutRange/Mammalia/inside/',
                     save.outside.in = 'cutRange/Mammalia/outside/')

readAndWrite(action = 'write', frmt = 'saveTXT', path = 'tables/',
             name = 'cut_example.txt', object = cutRange)
```

```
knitr::kable(cutRange, format = 'markdown')
```

Total.occurrences	Total.sp	sp.inside	sp.outside
8397	34	34	27

Invasive species could bias the analysis in macroecology or biogeography, because their distribution do not correspond to their original or native distributions.

We use the ***invasiveSp*** function to reduce this bias. This function is connected to the largest alien species databases (Island Biodiversity and Invasive Species (IBIS) (Kell and Worswick 1997) and Global Invasive Species Database (GISD) (Lowe et al. 2000)).

```
invasive <- invasiveSp(data                = 'inside.range',
                       rd.frmt             = 'readRDS',
                       path                = 'cutRange/Mammalia/inside/',
                       starts.in           = 1,
                       save.Sp.list        = TRUE,
                       wrt.frmt            = 'saveRDS',
                       save.foreign.in     = 'invasiveSP/Mammalia/foreign/',
                       save.non.foreign.in = 'invasiveSP/Mammalia/non.foreign/',
                       save.temp.in        = 'invasiveSP/Mammalia/')

readAndWrite(action = 'write', frmt   = 'saveTXT', path   = 'tables/',
             name   = 'invasive_Mammalia.txt', object = invasive)
```

If the process is stopped, you can reset it using the last number reported in console. You have to put this number in starts.in parameter as argument and the process will restart in the position assigned.

NOTE: The foreign species are marked with a 1 symbol (TRUE), and the non foreign species with a 0.

```
knitr::kable(invasive, format = 'markdown')
```

Species	ForeignSp	NonForeignSp
34	2	0

Finally, we are done. Now we have the list of species to download from GBIF in the file 'input.rgbif.non.foreign', that can be found in 'invasiveSP/Mammalia/non.foreign/' path.

```
toDownload <- readAndWrite(action = 'read', frmt = 'readRDS',
                           path = 'invasiveSP/Mammalia/non.foreign/',
                           name = 'input.rgbif.non.foreign')

str(toDownload)
```

```
## 'data.frame':    32 obs. of  1 variable:
##  $ species: Factor w/ 32 levels "Aotus lemurinus",..: 2 3 8 20 5 30 7 28 23 18 ...
```

```
head(toDownload)
```

```
##              species
## 1 Artibeus lituratus
## 2   Artibeus watsoni
## 3    Cebus capucinus
## 4    Myotis riparius
## 5  Carollia castanea
## 6 Tonatia saurophila
```

3.0 Obtain species restricted to the Biogeographic Choco: Pipeline 2.

When we have the list of non-invasive species that inhabit on The Biogeographic Choco (**sensu lato**: Hernandez (1992)), we can download all occurrences for each species using the ***gbifDownSp*** function. This function will save a file by species, thus if we decide to include or exclude species/records in the analysis, it would be easier to manipulate the files and perform the task.

The ***gbifDownSp*** function works as a wrapper for the **occ_search** function from the ***rgif*** package. You can use some additional parameters inherited from **occ_search** as **limit**, **basisOfRecord**, **country**, **publishingCountry**, **lastInterpreted**, **geometry**, **collectionCode**, **institutionCode** and **year**.

```
# Read the list of species to download. It can be the output file of the
# invasiveSp function.

input <- readAndWrite(action = 'read', frmt   = 'readRDS' ,
                      path   = 'invasiveSP/Mammalia/non.foreign/',
                      name   = 'input.rgbif.non.foreign')

# Download the species

gbifDownSp(sp.name          = input,
           taxon.key        = NULL ,
           genus            = NULL,
           epithet          = NULL,
           starts.in        = 1 ,
           wrt.frmt         = 'saveRDS',
           save.download.in = 'gbifDownSp/Mammalia/' ,
           limit            = 200000)
```

The occurrences reported by GBIF could be georeferenced or not. We can separate georeferenced from non georeferenced using the ***splitGeoref*** function.

The argument in the **data** parameter is a list of files to read, all files must be in the same format ('RDS' or 'TXT'). See the ***readAndWrite*** function.

```
# create the list of files

out.gbifDownSp <- list.files('gbifDownSp/Mammalia/')

SplitGeoref <- splitGeoref(data             = out.gbifDownSp,
                           rd.frmt          = 'readRDS',
```

```
                            path             = 'gbifDownSp/Mammalia/' ,
                            min.occ          = 3,
                            round.coord      = 4,
                            wrt.frmt         = 'saveRDS',
                            save.min.occ.in  = 'splitGeoref/Mammalia/min.Occurren/',
                            save.georef.in   = 'splitGeoref/Mammalia/georref/',
                            save.ungeoref.in = 'splitGeoref/Mammalia/not.georref/')

readAndWrite(action = 'write', frmt   = 'saveTXT', path   = 'tables/',
             name   = 'splitGeoref_example.txt', object = SplitGeoref)
```

```
knitr::kable(head(SplitGeoref, 4L), format = 'markdown')
```

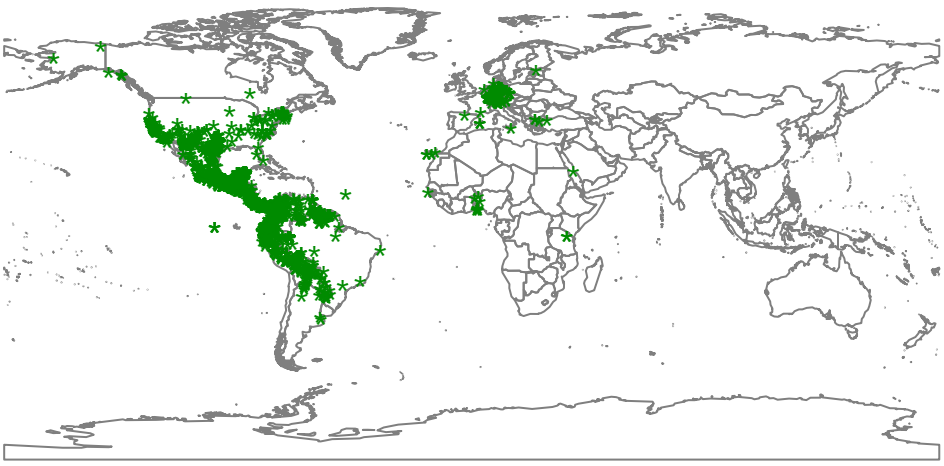
Species	georef	ungeoref	Min.occ
Aotus lemurinus	47	135	0
Artibeus lituratus	42	5	0
Artibeus watsoni	59	56	0
Ateles geoffroyi	176	55	0

The coordinates could be in different formats, but the unique valid format in this pipeline is decimal degree because we follow the Darwin Core Standard (Wieczorek 2012). Then, we test whether there are some errors with the coordinates format using the ***checkCoord*** function. Additionally, this function can check the range of the coordinates (Long: -180, 180; Lat: -90, 90).

```
out.splitGeref <- list.files('splitGeoref/Mammalia/georref/')

checkCoord(data          = out.splitGeref,
           path          = 'splitGeoref/Mammalia/georref/',
           rd.frmt       = 'readRDS',
           wrt.frmt      = 'saveRDS',
           save.right.in = 'checkCoord/Mammalia/right.coord/' ,
           save.wrong.in = 'checkCoord/Mammalia/wrogn.coord/')
```

We will check the georeferenced records downloaded from GBIF (Fig. 8).

```
data('wrld_simpl')

plot(wrld_simpl, border = 'grey50')

georefFromGBIF <- list.files('checkCoord/Mammalia/right.coord/')

for (i in 1:length(georefFromGBIF)) {
  sp <- readRDS(paste('checkCoord/Mammalia/right.coord/', georefFromGBIF[i], sep = ''))

  coordinates(sp) <- sp[,c('decimalLongitude', 'decimalLatitude')]

  plot(sp, add = TRUE, pch = '*', col = 'green4')
  }
```



```
# Figure 8. Occurrences downloaded from GBIF, with geocode assigned.
```

Sometimes the coordinates might have problems as wrong sign value, and as result we have coordinates of land species with occurrences at sea.

```
out.checkCoord <- list.files('checkCoord/Mammalia/right.coord/')

pointsAtSea <- pointsAtSea(data            = out.checkCoord ,
                           path            = 'checkCoord/Mammalia/right.coord/',
                           rd.frmt         = 'readRDS',
                           wrt.frmt        = 'saveRDS',
                           save.OnEarth.in = 'pointAtSea/Mammalia/on.earth/',
                           save.AtSea.in   = 'pointAtSea/Mammalia/at.sea/')


readAndWrite(action = 'write', frmt = 'saveTXT', path = 'tables/',
             name = 'pointsAtSea_example.txt', object = pointsAtSea)
```

Species	Total.occurrences	On.earth	At.sea
Aotus lemurinus	47	46	1
Artibeus lituratus	42	42	0
Artibeus watsoni	59	57	2
Ateles geoffroyi	176	162	14

Here, we can see the points at sea that were deleted in the last process (Fig. 9).

```
atsea <- list.files('pointAtSea/Mammalia/at.sea/')

if (length(atsea) != 0 ) {
  data('wrld_simpl')

  plot(wrld_simpl, border = 'grey50')

  for (i in 1:length(atsea )) {
    spAtSea <- readRDS(paste('pointAtSea/Mammalia/at.sea/', atsea[i], sep = ''))
```

```
    coordinates(spAtSea) <- spAtSea[,c('decimalLongitude','decimalLatitude')]

    plot(spAtSea,  add = T, pch = '*', col = 'green4')
  }

  onEarth <- list.files('pointAtSea/Mammalia/on.earth/')

  for (i in 1:length(onEarth)) {
    spOnEarth <- readRDS(paste('pointAtSea/Mammalia/on.earth/', onEarth[i], sep = ''))

    coordinates(onEarth) <- onEarth[,c('decimalLongitude','decimalLatitude')]

    plot(spOnEarth, add = T, pch = '*', col = 'orange')

    # Figure 9. Records at sea and on mainland.
    }
  }else{
    cat('You do not have records at sea, your figure 9 will be equal to figure 8. \n')
}
```

```
## You do not have records at sea, your figure 9 will be equal to figure 8.
```

We have downloaded species distributed on Choco, but some species could be widespread. We will use the function ***spOutPoly*** in two steps: first, delete species distributed in others continents, either because the species are cosmopolitan or because the species is in other continent but the coordinates were placed in the area of interest because the sign was changed, for example a coordinate of an African species was positioned on America. In this case, we will perform the process at the continent level in a restricted form because the parameters: **max.per.out** and **max.occ.out** present low values and the conditions B1 an B2 are TRUE (See: R-Alarcon and Miranda-Esquivel (submitted)).

If the argument in the **execute** parameter is 'FALSE', the process will not classified the species and only will return a table of information that can be saved in your current work directory (See: getwd()). If **execute** is 'TRUE' the process will classified the species and the table of information can be the result of the function.

For example, when the execute argument is 'FALSE', we will find the table of information in the current working directory (getwd ()), this file will be called 'Classify.sp'.

```
out.OnEarth <- list.files('pointAtSea/Mammalia/on.earth/')

data('America')

americaFilter <- spOutPoly(data            = out.OnEarth,
                           rd.frmt         = 'readRDS' ,
                           path            = 'pointAtSea/Mammalia/on.earth/',
                           shp.poly        = America,
                           max.per.out     = 10,
                           max.occ.out     = 3,
                           execute         = FALSE,
                           B1              = TRUE,
                           B2              = TRUE,
                           wrt.frmt        = 'saveRDS',
                           save.inside.in  = NULL ,
                           save.outside.in = NULL)
```

When we want to run the process we use execute = 'TRUE', then the table of information will be a vector data frame class assigned in 'americaFilter'. This vector will be saved using readAndWrite as in previous functions.

```
americaFilter <- spOutPoly(data            = out.OnEarth,
                           rd.frmt         = 'readRDS' ,
                           path            = 'pointAtSea/Mammalia/on.earth/',
                           shp.poly        = America,
                           max.per.out     = 10,
                           max.occ.out     = 3,
                           execute         = TRUE,
                           B1              = TRUE,
                           B2              = TRUE,
                           wrt.frmt        = 'saveRDS',
                           save.inside.in  = 'spOutPoly_America/Mammalia/inside/' ,
                           save.outside.in = 'spOutPoly_America/Mammalia/outside/')

readAndWrite(action = 'write', frmt = 'saveTXT', path = 'tables/',
             name   = 'americaFilterExc_example.txt', object = americaFilter)
```

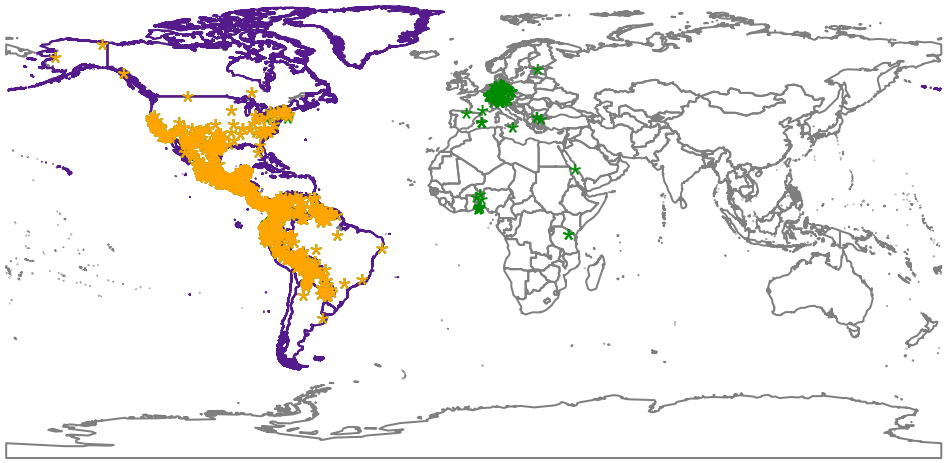
```
knitr::kable(head(americaFilter, 4L), format = 'markdown')
```

Sp	No.occurrences	No.inside	No.outside	Percent.out	Status.sp	Delete	CondicionApplied
Aotus lemurinus	46	45	1	2.17	inside	points	B1T
Artibeus lituratus	42	42	0	0	inside	None	A
Artibeus watsoni	57	56	1	1.75	inside	points	B1T
Ateles geoffroyi	162	159	3	1.85	inside	points	B1T

Now we can see in green the distributions that were deleted, and in orange the distributions restricted to America (purple polygon).

```
data("wrld_simpl")
data('America')

plot(wrld_simpl, border = 'grey50')

plot(America, border = 'purple4', add = TRUE)

onEarth <- list.files('pointAtSea/Mammalia/on.earth/')

for (i in 1:length(onEarth)) {
  spOnEarth <- readRDS(paste('pointAtSea/Mammalia/on.earth/', onEarth[i], sep = ''))

  coordinates(spOnEarth) <- spOnEarth[, c('decimalLongitude','decimalLatitude')]

  plot(spOnEarth, add = TRUE, pch = '*', col = 'green4')

}

DistriAmerica <- list.files('spOutPoly_America/Mammalia/inside/')
```

```
for (i in 1:length(DistriAmerica)) {
  spDistriAmerica <- readRDS(paste('spOutPoly_America/Mammalia/inside/',
                                   DistriAmerica[i], sep = ''))

  coordinates(spDistriAmerica) <- spDistriAmerica[, c('decimalLongitude',
                                                      'decimalLatitude')]

  plot(spDistriAmerica, add = TRUE, pch = '*', col = 'orange')
}
```



```
# Figure 10. Species distributed on the American continent.
```

The second step is performed as a relaxed process, because the parameter values are more relaxed and the condition B1 and B2 is FALSE (B1F, B2F). See: R-Alarcon and Miranda-Esquivel (submitted).

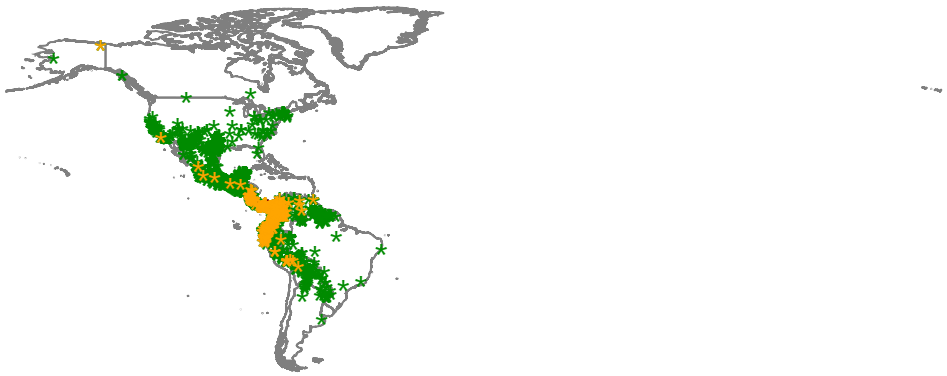
```
out.americFilter <- list.files('spOutPoly_America/Mammalia/inside/')

data('ChcBiog')
chocoFilter <- spOutPoly(data            = out.americFilter,
                         rd.frmt         = 'readRDS' ,
                         path            = 'spOutPoly_America/Mammalia/inside/',
                         shp.poly        = ChcBiog,
                         max.per.out     = 50,
                         max.occ.out     = 10,
                         execute         = TRUE,
                         B1              = FALSE,
                         B2              = FALSE,
                         wrt.frmt        = 'saveRDS',
                         save.inside.in  = 'spOutPoly_Choco/Mammalia/inside/' ,
                         save.outside.in = 'spOutPoly_Choco/Mammalia/outside/')

readAndWrite(action = 'write', frmt = 'saveTXT', path = 'tables/',
             name = 'ChocoFilterExc_example.txt', object = chocoFilter)
```

Sp	No.occurrences	No.inside	No.outside	Percent.out	Status.sp	Delete	CondicionApplied
Aotus lemurinus	45	28	17	37.78	inside	None	B1F

Sp	No.occurrences	No.inside	No.outside	Percent.out	Status.sp	Delete	CondicionApplied
Artibeus lituratus	42	30	12	28.57	inside	None	B1F
Artibeus watsoni	56	19	37	66.07	outside	sp	B3
Ateles geoffroyi	159	19	140	88.05	outside	sp	B3
Carollia castanea	95	75	20	21.05	inside	None	B1F
Carollia colombiana	1	1	0	0	inside	None	A

We can see in orange the species restricted to The Biogeographic Choco.

```
data('ChcBiog')
data('America')
plot(America, border = 'grey50')

DistriAmerica <- list.files('spOutPoly_America/Mammalia/inside/')

for (i in 1:length(DistriAmerica)) {
  spDistriAmerica <- readRDS(paste('spOutPoly_America/Mammalia/inside/',
                                   DistriAmerica[i], sep = ''))
  coordinates(spDistriAmerica) <- spDistriAmerica[, c('decimalLongitude', 'decimalLatitude')]
  plot(spDistriAmerica, add = TRUE, pch = '*', col = 'green4')
}

plot(ChcBiog, border = 'red4', add = TRUE)

DistriChoco <- list.files('spOutPoly_Choco/Mammalia/inside/')

for (i in 1:length(DistriChoco)) {
  spDistriChoco <- readRDS(paste('spOutPoly_Choco/Mammalia/inside/',
                                 DistriChoco[i], sep = ''))
  coordinates(spDistriChoco) <- spDistriChoco[, c('decimalLongitude', 'decimalLatitude')]

  plot(spDistriChoco, add = TRUE, pch = '*', col = 'orange')
}
```



```
# Figure 11. Species distributed in the biogeographic Choco.
```

However some points are outside of Choco. This happens because in the last filter when the B1 or B2 was FALSE (B1F, B2F), the anomalous points outside of Choco were not deleted. Now we need to detect these

points. We can use the ***delPointsOrSp*** function to delete the points/species outside the area, the filter is semi-automatic.

For the parameters west = -120, east = -65, south = -25, north = 25, ***delPointsOrSp*** will search coordinates less than -120 and higher than -65 on longitude, and less than -25 and higher than 25 on latitude.

In this case, for example the species *Saguinus oedipus* presents a record in USA over 25 degrees, and the function will ask whether delete this point or delete the total distribution of this species. In our case we deleted the point.

NOTE: Until here, we could use all the processes in a single run, maybe as an Rscript, but the ***delPointsOrSp*** function requires that user to decide whether delete some points or delete the total distribution of a species.

```
out.choco <- list.files('spOutPoly_Choco/Mammalia/inside/')

delPointsOrSp <- delPointsOrSp(data         = out.choco,
                               rd.frmt      = 'readRDS' ,
                               path         = 'spOutPoly_Choco/Mammalia/inside/',
                               west         = -120,
                               east         = -65,
                               south        = -25,
                               north        = 25,
                               plot.distrib = TRUE,
                               wrt.frmt     = 'saveRDS',
                               save.file    = 'delPointOrSp/Mammalia/')
```

```
## Aotus lemurinus: There are no points in the range

## [1] "Check the distribution: Artibeus lituratus"
## Delete points?
##  (yes: y or not: n)=
## Delete species?
##  (yes: y or not: n)=

## Carollia castanea: There are no points in the range
## Carollia colombiana: There are no points in the range
## Diplomys labilis: There are no points in the range
## Echinoprocta rufescens: There are no points in the range
## Heteromys teleus: There are no points in the range
## Liomys adspersus: There are no points in the range
## Micoureus alstoni: There are no points in the range
## Molossus pretiosus: There are no points in the range

## [1] "Check the distribution: Saguinus oedipus"
## Delete points?
##  (yes: y or not: n)=
## Delete species?
##  (yes: y or not: n)=

## Sigmodon peruanus: There are no points in the range
## Sturnira luisi: There are no points in the range
## Thomasomys baeops: There are no points in the range
## Tylomys mirae: There are no points in the range
```

```
readAndWrite(action = 'write', frmt = 'saveTXT', path = 'tables/',
             name = 'delPointsOrSp_example.txt', object = delPointsOrSp)
```

```
knitr::kable(head(delPointsOrSp), format = 'markdown')
```

Group	Init.Sp	Init.Occurrences	Fin.Sp	Fin.Occurrences
Aotus lemurinus	1	45	1	45
Artibeus lituratus	1	42	1	42
Carollia castanea	1	95	1	95
Carollia colombiana	1	1	1	1
Diplomys labilis	1	7	1	7
Echinoprocta rufescens	1	12	1	12

We can delete the points when are asked:

```
[1] "Check the distribution: Saguinus oedipus"
Delete points?
 (yes: y or not: n)= y
```

or, we can delete the distribution of the species:

```
[1] "Check the distribution: Saguinus oedipus"
Delete points?
 (yes: y or not: n)= n
 Delete species? y
```

or we can delete nothing:

```
[1] "Check the distribution: Saguinus oedipus"
Delete points?
 (yes: y or not: n)= n
 Delete species? n
```

We can see a point over 25 decimal degrees, this point is deleted because it did not correspond to the distribution of the species. For this example, we kept the point over 25 decimal degrees.

NOTE: In Rmarkdown we can not deleted these anomalous points, but you could run the 'WorkedExample.R' code and perform this complete process.

```
data('America')
data('ChcBiog')
par(oma = c(0,0,0,0), mar = c(0,0,0,0) + 0.1)

plot(America, ylim = c(-50,50), xlim = c(-140,-30),  border = 'grey50')

DistriChoco <- list.files('spOutPoly_Choco/Mammalia/inside/')

for (i in 1:length(DistriChoco)) {
  spDistriChoco <- readRDS(paste('spOutPoly_Choco/Mammalia/inside/',
```

```
                                DistriChoco[i], sep = ''))

  coordinates(spDistriChoco) <- spDistriChoco[, c('decimalLongitude', 'decimalLatitude')]

  plot(spDistriChoco, add = TRUE, pch = '*', col = 'green4')
}

deltPoint <- list.files('delPointOrSp/Mammalia/')

for (i in 1:length(deltPoint)) {

  spdeltPoint <- readRDS(paste('delPointOrSp/Mammalia/', deltPoint[i], sep = ''))

  coordinates(spdeltPoint) <- spdeltPoint[, c('decimalLongitude', 'decimalLatitude')]

  plot(spdeltPoint, add = TRUE,  pch = '*',  col = 'orange')

}

plot(ChcBiog, border = 'red4', add = TRUE)
abline(h = 25)
abline(h = -25)
abline(v = -120)
abline(v = -65)
```

```
# Figure 12. Occurrences that do not correspond to a chocoan species.
```

After all the filters used, some species could have very few records. Thus, to define a polygon as an area of distribution we need at least 3 occurrences, then we will get species with a minimal number of occurrences using the ***usefulSp*** function.

```
out.deltpoints <- list.files('delPointOrSp/Mammalia/')

usefulSp <- usefulSp(data = out.deltpoints,
                     path = 'delPointOrSp/Mammalia/',
                     cut.off = 3,
                     rd.frmt = 'readRDS',
                     wrt.frmt = 'saveRDS',
                     save.useful.in = 'usefulSp/Mammalia/useful/',
                     save.useless.in = 'usefulSp/Mammalia/useless/')

readAndWrite(action = 'write', frmt   = 'saveTXT', path   = 'tables/',
             name   = 'usefulSP_example.txt', object = usefulSp)
```

```
knitr::kable(head(usefulSp,4L), format = 'markdown')
```

Species	Total.occurrences	State
Aotus lemurinus	45	in
Artibeus lituratus	42	in
Carollia castanea	95	in
Carollia colombiana	1	out

Now we prepare a single file with the data cleaned, and 'That's all folks'

```
out.useful <- list.files('usefulSp/Mammalia/useful/' )

StackSp <- stackSp(data            = out.useful,
                   rd.frmt         = 'readRDS' ,
                   path            = 'usefulSp/Mammalia/useful/',
                   save.name       = 'stackSp_example' ,
                   save.staking.in = 'stackingSp/',
                   wrt.frmt        =  'saveRDS')
```

```
knitr::kable(StackSp,  format = 'markdown')
```

TotalSp	TotalOccurrences
13	373

We might want some information about the spatial distribution as a whole or by species. These metrics can be the mean propinquity, median propinquity, mode, skewness, etc.

For example, as a whole:

```
# Remember to read the final file to perform the meanPropinquity process
out.stackSp <- readAndWrite(action = 'read', frmt = 'readRDS' ,
                      path = 'stackingSp/', name = 'stackSp_example')
```

```
PropinquityWhole <- meanPropinquity(coord.table = out.stackSp,
                                    calculatedBy = 'whole',
                                    wrt.frmt    = 'saveTXT' ,
                                    save.info.in = 'meanPropinquity/Mammalia/' ,
                                    plot.dist    =  TRUE,
                                    plot.onMap   = TRUE,
                                    save.plot.in = 'meanPropinquity/Mammalia/')
```

```
knitr::kable(head(PropinquityWhole),  format = 'markdown')
```

Occurrences	MeanPropinquity	Median	SD	MinDist	MaxDist	Skewness	Mode	ModeSkewness
373	0.5993664	0.1696367	2.616306	1e-04	42.08164	0.3809524	0.004905	0.8035714

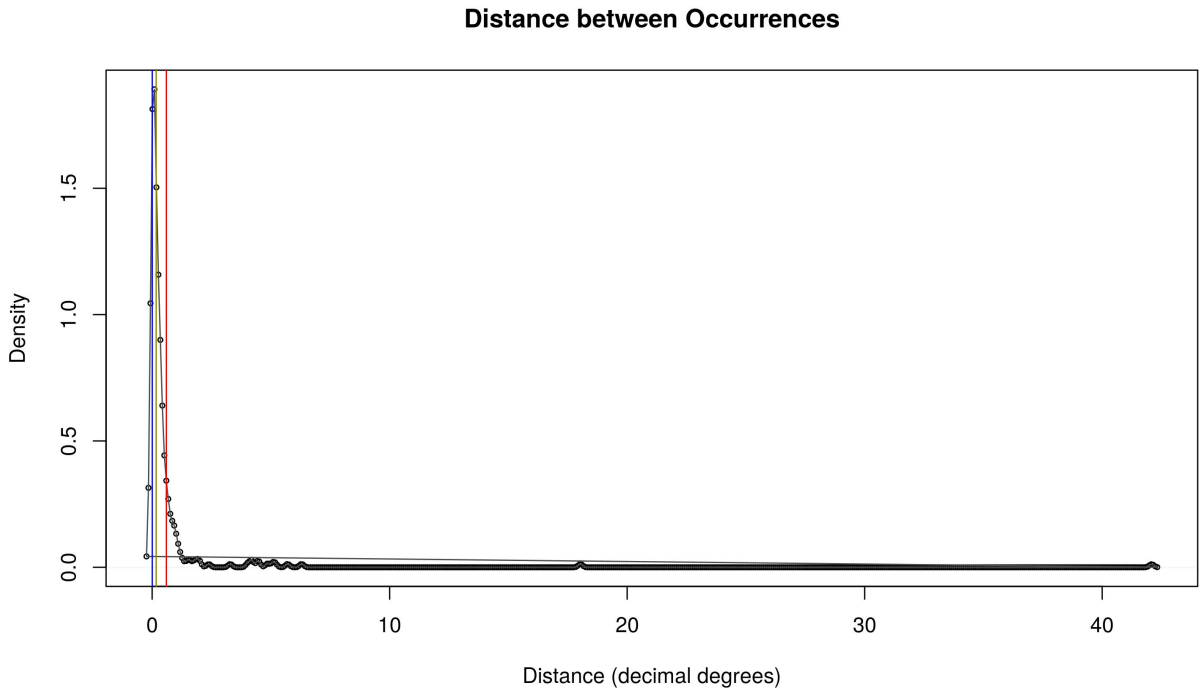
or by species:

```
PropinquitySp <- meanPropinquity(coord.table    = out.stackSp ,
                                 calculatedBy = 'species',
                                 wrt.frmt     = 'saveTXT' ,
                                 save.info.in = 'meanPropinquity/Mammalia/' ,
                                 plot.dist    =  TRUE,
                                 plot.onMap   = TRUE,
                                 save.plot.in = 'meanPropinquity/Mammalia/')
```

```
knitr::kable(head(PropinquitySp, 4L),  format = 'markdown')
```

Species	Occurrences	MeanPropinquity	Median	SD	MinDist	MaxDist	Skewness	M
Aotus lemurinus	45	1.1216761	0.4615993	2.472575	0.0001000	15.827867	0.5000000	1.1216
Artibeus lituratus	42	3.2443090	0.0942555	10.153677	0.0001000	63.536043	0.1219512	3.2443
Carollia castanea	95	0.6542852	0.1126281	1.543883	0.0001414	12.242746	0.2978723	0.6542
Diplomys labilis	7	1.5636763	0.3961985	2.157365	0.0004243	5.256586	0.6666667	1.5636

The information table and the density plot with the descriptors can be seen in 'meanPropinquity/Mammalia/' path.



DENSITY PLOT LEGEND ::::

RED: Mean Propinquity

YELLOW: Median Propinquity

BLUE: Mode Propinquity

Finally, we also could convert between formats using the ***readAndWrite*** function. For example, from an R object ('RDS') to plain text ('TXT'), so we could read this with any program.

```
finalFile <- readAndWrite(action = 'read', frmt = 'readRDS' ,
                          path = 'stackingSp/', name = 'stackSp_example')



readAndWrite(action = 'write', frmt = 'readRDS' , path = '',
             name = 'finalFile_example', object = finalFile)
```

The final file looks like:

```
head(finalFile, 1L)
```

```
##              name        key decimalLatitude decimalLongitude  issues
## 1 Aotus lemurinus 1213678503          3.5983         -76.7142 cdround
##                               datasetKey
## 1 8ac9b3f9-383e-4401-bb17-fdd802057087
##                       publishingOrgKey publishingCountry    protocol
## 1 cd9bc4b5-4375-4991-aec5-0b4443b5d7a6                CO DWC_ARCHIVE
##                     lastCrawled                   lastParsed extensions
## 1 2015-12-09T14:34:27.221+0000 2015-12-09T14:34:27.292+0000       none
```

```
##       basisOfRecord taxonKey kingdomKey phylumKey classKey orderKey
## 1 HUMAN_OBSERVATION  5219591          1        44      359      798
##   familyKey genusKey speciesKey
## 1      9621  2436667    5219591
##                                    scientificName  kingdom   phylum
## 1 Aotus lemurinus (I. Geoffroy Saint-Hilaire, 1843) Animalia Chordata
##      order  family genus         species genericName specificEpithet
## 1 Primates Cebidae Aotus Aotus lemurinus       Aotus       lemurinus
##   taxonRank elevation elevationAccuracy     continent   stateProvince year
## 1   SPECIES      1420                 0 SOUTH_AMERICA Valle del Cauca 2013
##   month day                    eventDate              lastInterpreted
## 1     8  18 2013-08-17T22:00:00.000+0000 2015-12-09T14:34:32.354+0000
##   identifiers facts relations geodeticDatum    class countryCode  country
## 1        none  none      none         WGS84 Mammalia          CO Colombia
##     rightsHolder         identifier                   habitat
## 1 Fundación GAIA GAIA:DAG:MAM:00001 Bosque Natural de Galería
##   institutionID                                          georeferencedBy
## 1 900.318.849-7 Adriana Lucía Guerrero Chacon;Sebastian Orjuela Salazar
##   samplingEffort
## 1    1 kilómetro
##                                                  locality county
## 1 Quebrada La Miquera, Hacienda Larraniaga, Vereda El Chilcal  Dagua
##   municipality     gbifID collectionCode language       occurrenceID
## 1   El Limonar 1213678503           GAIA       es GAIA:DAG:MAM:00001
##     type catalogNumber
## 1 Evento         00001
##                                              recordedBy vernacularName
## 1 Adriana Lucía Guerrero Chacon ; Sebastian Orjuela Salazar        Marteja
##   institutionCode samplingProtocol               accessRights
## 1  Fundación GAIA        Recorrido Sólo para uso no comercial
##                                        identifiedBy dateIdentified
## 1 Adriana Lucía Guerrero Chacon;Sebastian Orjuela Salazar           <NA>
##   modified references informationWithheld verbatimEventDate datasetName
## 1     <NA>       <NA>                <NA>              <NA>        <NA>
##   taxonID http...unknown.org.occurrenceDetails rights eventTime
## 1    <NA>                                 <NA>   <NA>      <NA>
##   identificationID verbatimLocality occurrenceRemarks  sex
## 1             <NA>             <NA>              <NA> <NA>
##   infraspecificEpithet higherGeography endDayOfYear startDayOfYear
## 1                 <NA>            <NA>         <NA>           <NA>
##   associatedSequences higherClassification recordNumber
## 1                <NA>                 <NA>         <NA>
##   http...unknown.org.organismID preparations verbatimElevation
## 1                          <NA>         <NA>              <NA>
##   nomenclaturalCode ownerInstitutionCode datasetID collectionID lifeStage
## 1              <NA>                 <NA>      <NA>         <NA>      <NA>
##   verbatimCoordinateSystem establishmentMeans occurrenceStatus
## 1                     <NA>              <NA>             <NA>
##   georeferenceProtocol georeferenceVerificationStatus georeferenceRemarks
## 1                 <NA>                            <NA>                <NA>
##   georeferenceSources eventID acceptedNameUsage previousIdentifications
## 1                <NA>    <NA>              <NA>                    <NA>
##   dynamicProperties identificationVerificationStatus locationAccordingTo
## 1              <NA>                              <NA>                <NA>
```

```
##   georeferencedDate otherCatalogNumbers identificationQualifier typeStatus
## 1              <NA>                <NA>                    <NA>       <NA>
##   typifiedName locationRemarks waterBody eventRemarks locationID
## 1         <NA>            <NA>      <NA>         <NA>       <NA>
##   reproductiveCondition associatedOccurrences taxonomicStatus disposition
## 1                  <NA>                  <NA>            <NA>        <NA>
##   identificationRemarks taxonRemarks created fieldNumber islandGroup
## 1                  <NA>         <NA>    <NA>        <NA>        <NA>
##   island coordinateAccuracy depth depthAccuracy behavior fieldNotes
## 1   <NA>                 NA    NA            NA     <NA>       <NA>
##   license bibliographicCitation parentNameUsage verbatimTaxonRank
## 1    <NA>                  <NA>            <NA>              <NA>
##   associatedReferences geologicalContextID latestEraOrHighestErathem
## 1                 <NA>                <NA>                      <NA>
##   latestEonOrHighestEonothem earliestEonOrLowestEonothem
## 1                       <NA>                       <NA>
##   latestPeriodOrHighestSystem earliestEraOrLowestErathem
## 1                        <NA>                       <NA>
##   earliestEpochOrLowestSeries lowestBiostratigraphicZone
## 1                        <NA>                       <NA>
##   earliestPeriodOrLowestSystem highestBiostratigraphicZone
## 1                         <NA>                        <NA>
##   latestEpochOrHighestSeries
## 1                       <NA>
```