Data Mobilisation from GBIF to the EBV Data Portal for IAS of Union Concern

Notebook 3 - Metrics computation for the IAS occurrences cube

true

2024-08-23

Introduction

In this notebook, we calculate simple metrics for the invasive alien species (IAS) of union concern in Europe. To do this, an IAS occurrence cube has been created previously (see Notebook 01) using the occurrence cube software developed by GBIF—the Global Biodiversity Information Facility— under the Biodiversity Building Blocks for Policy (B3) project. Details of the data query in GBIF are available at DOI 10.15468/dl.gxk3vh. Here, we present five metrics:

- Total number of occurrences
- Earliest date of occurrence
- Latest date of occurrence
- Basis of record of the earliest occurrence
- Basis of record of the latest date of occurrence

Note: This series of notebooks is part of the results of Task 3.3 of the Biodiversity Building Blocks for Policy project funded by the European Union's Horizon Europe Research and Innovation Programme (ID No 101059592). Additional notebooks exploring the results and calculating simple metrics are also available in the same repository.

Load Library and Input Data

We start by loading all the libraries needed in this notebook.

```
rm(list=ls())
gc()
```

```
##           used (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells 1861038 99.4    6147488 328.4 12006812 641.3
## Vcells 4390227 33.5   47616720 363.3 74401122 567.7
```

```
# Load requiered libraries
library(b3gbi) # for csv occurrence cubes
library(purrr) # for data summary and grouping
library(here)
```

```
library(dplyr)
library(lubridate) # for dates
library(terra) # for raster
library(ncdf4)
library(ggplot2)
library(sf)
library(viridis) # for colour palettes
```

Now we load all input data obtained in the previous notebooks. These are:

- The occurrence IAS cube obtained previously through GBIF API.
- Inputs related to the EEA grid at 10 km for rasterisation.
- Taxonomic information of all analysed species.

```
# Load occurrence cube using b3gbi
occcube <- "0077925-240506114902167"
cin <- process_cube(here(paste0("output/datacubes/csv/ias/", occcube, ".csv")))

# Import taxonomy of all IAS based on gbif backbone taxonomy
gbif_tax <- read.csv(here("input/data/ias/taxonomy/List87IAS_EU_match_gbif_synonyms_acceptedUsageKeys.cs

# Load precomputed centroids for EEA 10 km grid
coorin <- read.csv(here("input/grid/centroids/eeagrid_centroids_10K.csv"))

# Import and convert the EU borders to a data frame
borders_eu <- st_read("C:/data/grid/eea_v_3035_100_k_adm-boundaries-eea38-plus_i_2018-2020_v01_r00/NUTS
```

```
## Reading layer 'NUTS2021_3035' from data source
##   'C:\data\grid\eea_v_3035_100_k_adm-boundaries-eea38-plus_i_2018-2020_v01_r00\NUTS2021_3035.shp'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 44 features and 27 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 943758.8 ymin: 941658.1 xmax: 7316569 ymax: 6405005
## Projected CRS: ETRS89-extended / LAEA Europe
```

```
# Load reference EEA grid in raster format
gridin <- rast(here("input/grid/eeagrid_10K.tif"))
res <- res(gridin)[1] #resolution of reference raster

# Replace by corresponding column name of input dataset
colnames(coorin)[colnames(coorin) == "eeacellcode"] <- "cellCode"
```

We now create an empty raster with three dimensions: latitute, longitude and species. In this empty raster we will map our metrics.

```
# Find number of species
spskey <- unique(cin[["data"]][["taxonKey"]])
```

Metrics Computation

Metric 1: Total Number of Occurrences We calculate the total number of occurrences for each species using the function `total_occ`.

```r
# Calculate total number of occurrences per pixel
total_occ <- function(cin, gridin, coorin){

  # Find number of species
  spskey <- unique(cin[["data"]][["taxonKey"]])

  # Create empty raster
  r <- rast(ext(gridin), resolution=res(gridin), nlyrs=length(spskey), crs=crs(gridin))
  values(r) <- NA

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

    # Subset one species
    spsi <- cin[["data"]][cin[["data"]]$taxonKey == spskey[i], ]

    # Sum up the total number of occurrences per cellCode
    metricx <- spsi %>%
      group_by(cellCode) %>%
       summarize(total_occurrences = sum(obs), cellCode = first(cellCode))

    # Merge pixel coordinates with the corresponing EEA grid ID
    metriccoor <- merge(metricx[,c("total_occurrences", "cellCode")], coorin, by="cellCode")

    # Check for a few occurrences. Only 1 occurrence cannot be rasterised, and error extente when they
    if(length(unique(metriccoor$x)) < 5){
      #add second empty point
      x <- metriccoor$x[1] + res
      y <- metriccoor$y[1] + res
      metriccoor <- rbind(metriccoor, c(NA, NA ,NA, x, y))
    }

  # Rasterize data
  r[[i]]  <- rast(metriccoor[,c("x", "y", "total_occurrences")], type="xyz", crs=crs(gridin), extent=ex
    }
  names(r) <- spskey
  return(r)
}
```

The next step is to run the previous funcion and plot an example.

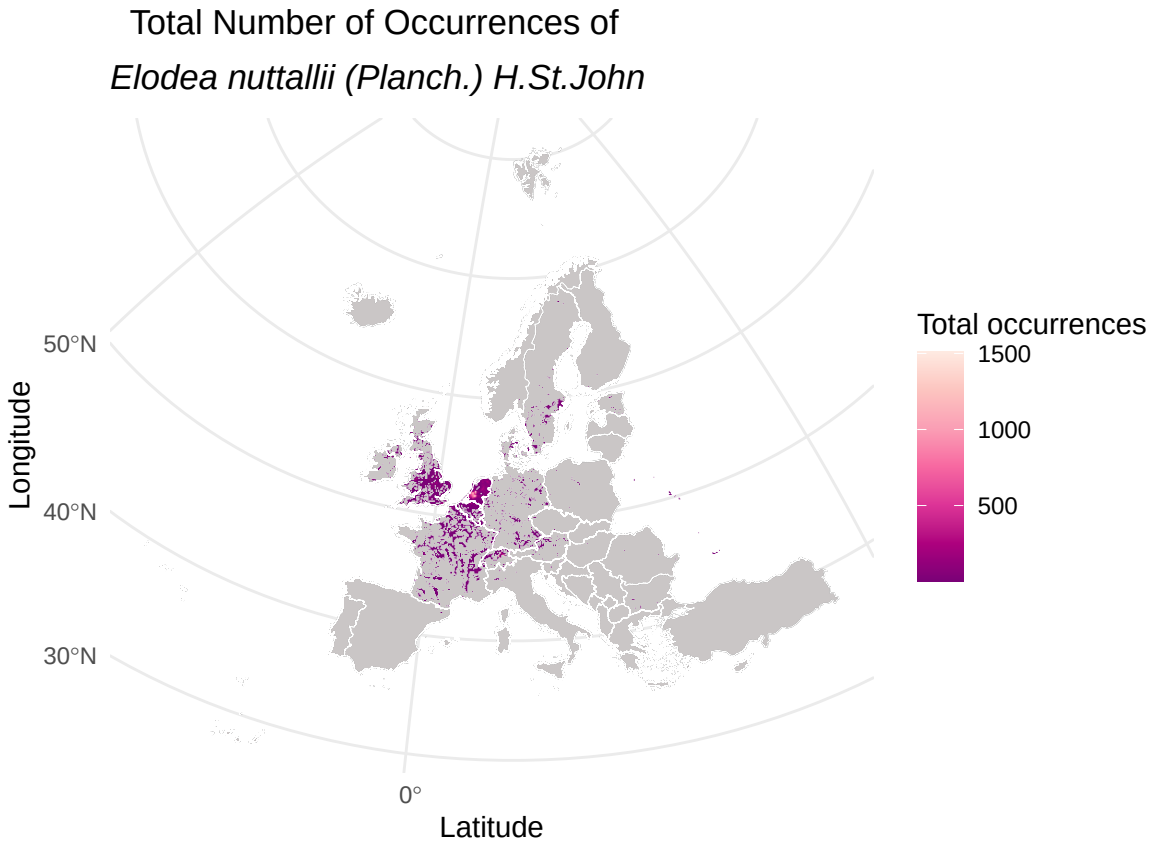
```r
total_occ_sps <- total_occ(cin, gridin, coorin)
```

Prepare data for plotting

```r
# Select one specie
spi <- 24 # 6

# Convert data to a data frame
df <- as.data.frame(total_occ_sps[[spi]], xy=TRUE)
spid <- as.numeric(colnames(df)[3])
colnames(df)[colnames(df) == colnames(df)[3]] <- "Value"

# Find the specie scientific name
```

```r
spi_name <- gbif_tax %>%
  filter(key %in% spid)
```

```r
ggplot() +
  geom_sf(data = borders_eu, fill = "#cac6c6") +
  geom_raster(data = df, aes(x = x, y = y, fill = Value)) +
  geom_sf(data = borders_eu, fill = NA, color = "white") +
  scale_fill_distiller(name="Total occurrences", palette="RdPu") +
  theme_minimal() +
  labs(
    title = bquote(atop("Total Number of Occurrences of ",
     italic(.(spi_name$scientificName)), "in GBIF")),
    x = "Latitude",
    y = "Longitude",
    fill = "Total <br> occurrences")
```




```r
# save figure if needed
ggsave(here("output/figures/nb_ias/fig2_ias_gray3_id83.png"))
```

Metric 2 and 3: Earliest Date of Occurrence and Latest Date of Occurrence at GBIF We calculate the earliest (latest) date of occurrence for each species. The function `min_max_dates` finds the earliest (or latest) dates of records per species at the pixel level across the entire dataset.

```r
min_max_dates <- function(cin, gridin, coorin, fxdates){
  # Find number of species
  spskey <- unique(cin[["data"]][["taxonKey"]])

  # Create empty raster
  r <- rast(ext(gridin), resolution=res(gridin), nlyrs=length(spskey), crs=crs(gridin))
  values(r) <- NA

  # Create empty dataframe for subsequent metrics
  metricout <- data.frame()

  for (i in 1:length(spskey)){  # length(spskey)

    # Subset one species
    spsi <- cin[["data"]][cin[["data"]]$taxonKey == spskey[i], ]

    # Add a new column for decimal dates
    spsi$decimalDate <- format(ym(spsi$yearMonth), "%Y%m")
    spsi$decimalDate <- as.numeric(spsi$decimalDate)

    if (fxdates == "earliest dates"){
    # Find metric. In this example earliest date of record
    metricx <- spsi %>%
      group_by(cellCode) %>%
      slice_min(decimalDate, with_ties = FALSE) %>%
      ungroup()
    metricx
    } else if (fxdates == "latest dates") {
    # Find metric. In this example latest date of record
    metricx <- spsi %>%
      group_by(cellCode) %>%
      slice_max(decimalDate, with_ties = FALSE) %>%
      ungroup()
    metricx
    }

    # Merge pixel coordinates with the corresponing EEA grid ID
    metriccoor <- merge(metricx[,c("decimalDate", "cellCode")], coorin, by="cellCode")

    # Check for a few occurrences. Only 1 occurrence cannot be rasterised, and an 'extent' error is gen
    if(length(unique(metriccoor$x)) < 5){
      # Add second empty point
      x <- metriccoor$x[1] + res
      y <- metriccoor$y[1] + res
      metriccoor <- rbind(metriccoor, c(NA, NA ,NA, x, y))
    }

  metricout <- rbind(metricout, metricx)

  # Rasterize data
  r[[i]]  <- rast(metriccoor[,c("x", "y", "decimalDate")], type="xyz", crs=crs(gridin), extent=ext(grid
  }
```

```
  # Assign name of species key to the corresponding raster 'layer'
  names(r) <- spskey

  results_list <- list(raster_metric = r, metrics = metricout)
  return(results_list)

}
```

In the following code chunk, we compute metrics 2 and 3. Note that the last function argument specifies whether we calculate the earliest or latest date of record in GBIF.

```
min_dates_all <- min_max_dates(cin, gridin, coorin, "earliest dates")
max_dates_all <- min_max_dates(cin, gridin, coorin, "latest dates")
```

Note that the min_max_dates function returns a list. The first element of the list is a raster containing the earliest (or latest) date of the records. The second element is a table that will be used in the next section to calculate metrics 4 and 5.

```
# Extract raster of earliest dates of records
min_dates <- min_dates_all[[1]]

# Extract raster of latest dates of records
max_dates <- max_dates_all[[1]]
```

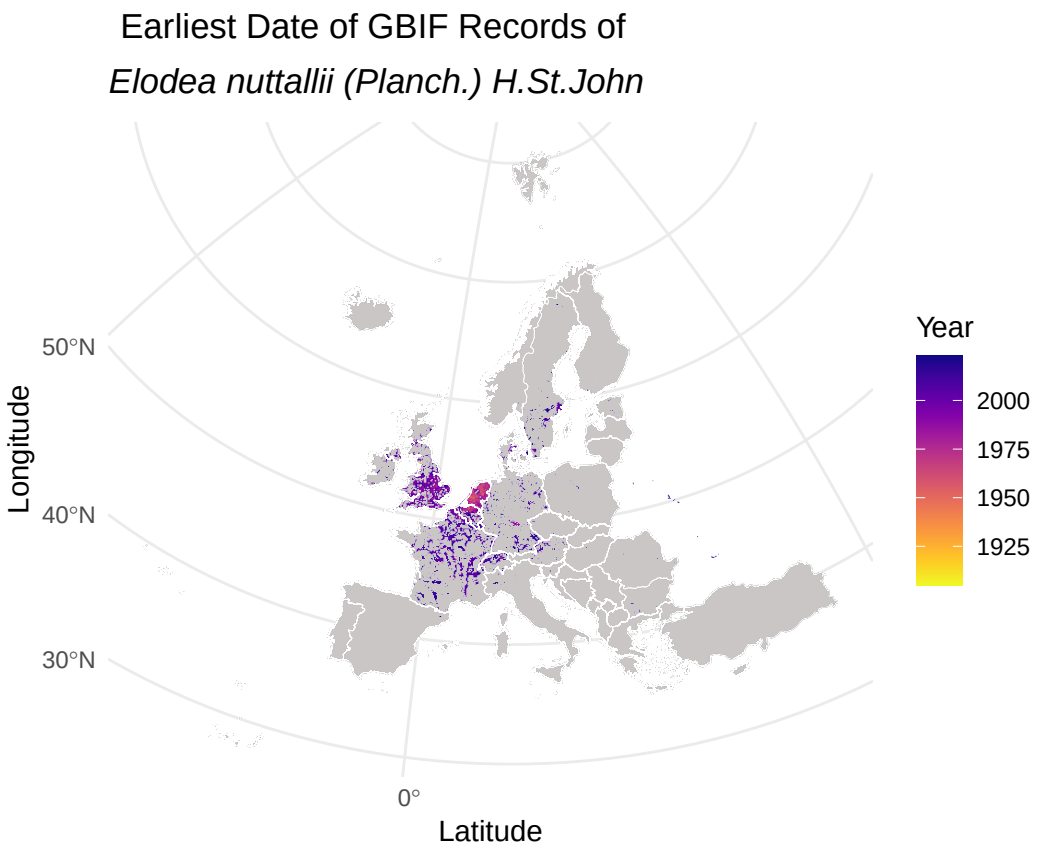
Prepare data for plotting.

```
# Convert raster layers to data frames
spi <- 24  # 6
df1 <- as.data.frame(min_dates[[spi]], xy=TRUE)
spid <- as.numeric(colnames(df1)[3])
# spid <- 8909595 # for plotting
colnames(df1)[colnames(df1) == colnames(df1)[3]] <- "Value"
df1[["Value"]] <- df1[["Value"]]/100 # to only show years

df2 <- as.data.frame(max_dates[[spi]], xy=TRUE)
colnames(df2)[colnames(df2) == colnames(df2)[3]] <- "Value"
df2[["Value"]] <- df2[["Value"]]/100 # to only show years

# Find the specie scientific name
spi_name <- gbif_tax %>%
  filter(key %in% spid)
```

This code chunk is for plotting the 'Earliest Date of Records at GBIF' for one species.

```
ggplot() +
  geom_sf(data = borders_eu, fill = "#cac6c6") +
  geom_raster(data = df1, aes(x = x, y = y, fill = Value)) +
  geom_sf(data = borders_eu, fill = NA, color = "white") +
  # scale_fill_binned(name="Year", type="viridis") + # Optional: for better color scale
  # scale_fill_distiller(name="Year", palette="RdPu") +
  scale_fill_viridis(name="Year", option = "C", direction = -1) +
  theme_minimal() +
```

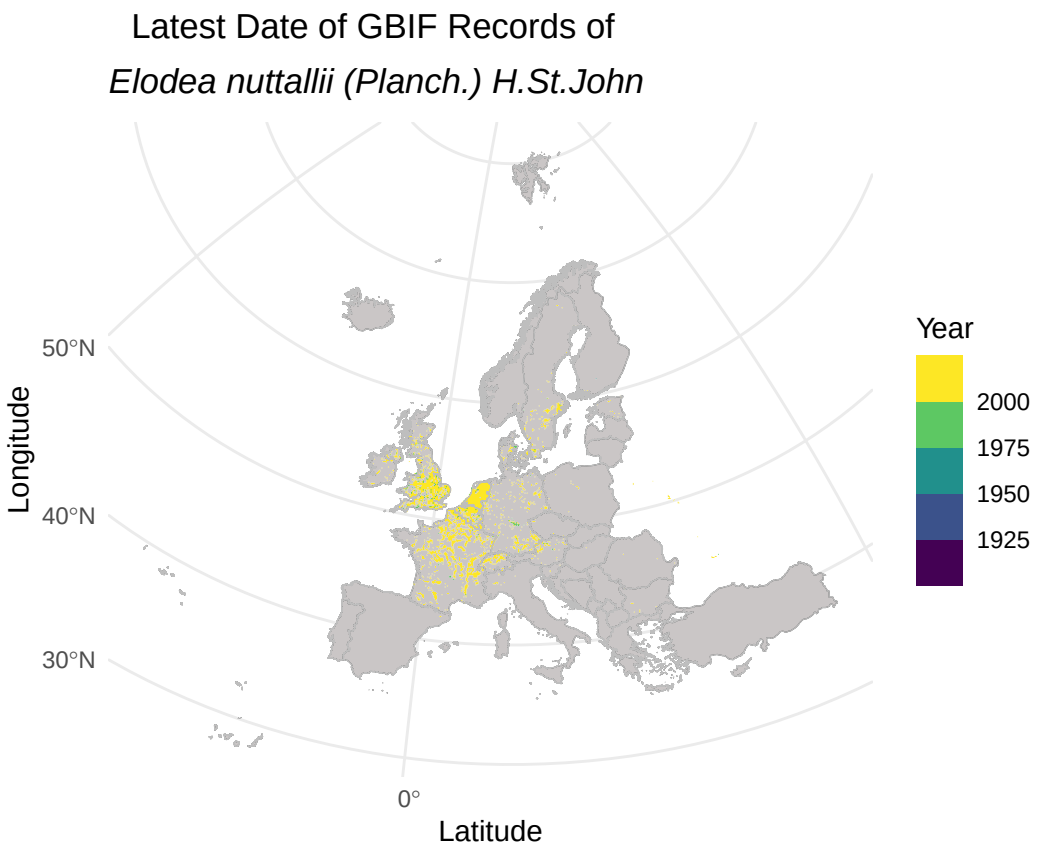
```
  ggtitle(bquote(atop("Earliest Date of GBIF Records of ",
     italic(.(spi_name$scientificName), "per pixel")))) +
  labs(x = "Latitude",
    y = "Longitude",
    fill = "Year")
```




```
# save figure if needed
ggsave(here("output/figures/nb_ias/fig1_ias_grey_id83.png"))
```

And the following code chunck is for plotting the "Latest date of records at GBIF" for the same specie.

```
ggplot() +
  geom_sf(data = borders_eu, fill = "#cac6c6") +
  geom_raster(data = df2, aes(x = x, y = y, fill = Value)) +
  geom_sf(data = borders_eu, fill = NA, color = "gray") +
  scale_fill_binned(name="Year", type="viridis") +
  theme_minimal() +
  labs(
    title = bquote(atop("Latest Date of GBIF Records of ",
     italic(.(spi_name$scientificName)), "per pixel")),
    x = "Latitude",
    y = "Longitude",
    fill = "Year")
```



Metrics 4 and 5: Basis of Record We compute the basis of record for the earliest and latest occurrences at GBIF.

First, we identify which are the unique "basis of record" in our IAS occurrence cube and add an identifier.

```
# Select unique 'basis of record' from cin
basis_rec <- data.frame(unique((cin$data)[3]))
basis_rec <- basis_rec[order((basis_rec$basisofrecord)),]
```

Next, we use a data frame obtained previously when computing metrics 2 and 3. Specifically, we refer to the second element of the min_max_dates function, which is a table containing the earliest and latest dates of species occurrences in GBIF.

```
# Extract dataframe of earliest dates of records
min_dates_df <- min_dates_all[[2]]

# Add numeric identifiers based on the custom order of basisofrecord
min_dates_df <- min_dates_df %>%
  mutate(basisofrecordID = match(basisofrecord, basis_rec))

# Extract dataframe of latest dates of records
max_dates_df <- max_dates_all[[2]]

# Add numeric identifiers based on the custom order of basisofrecord
max_dates_df <- max_dates_df %>%
  mutate(basisofrecordID = match(basisofrecord, basis_rec))
```

```
datasource_bydate <- function(df, gridin, coorin, spskey){

   # Create empty raster
  r <- rast(ext(gridin), resolution=res(gridin), nlyrs=length(spskey), crs=crs(gridin))
  values(r) <- NA

  for (i in 1:length(spskey)){
    # Subset one species
     spsi <- df[df$taxonKey == spskey[i], ]

    # Calculate the mean of the data source identifier per cellCode. It should be an integer number.
    metricx <- spsi %>%
      group_by(cellCode) %>%
       summarize(basisofrecord_id = mean(basisofrecordID))

    # Merge pixel coordinates with the corresponing EEA grid ID
    metriccoor <- merge(metricx[,c("basisofrecord_id", "cellCode")], coorin, by="cellCode")

    # Check for a few occurrences. Only 1 occurrence cannot be rasterised, and an 'extent' error is gen
    if(length(unique(metriccoor$x)) < 5){
      # Add second empty point
      x <- metriccoor$x[1] + res
      y <- metriccoor$y[1] + res
      metriccoor <- rbind(metriccoor, c(NA, NA ,NA, x, y))
    }

  # Rasterize data
  r[[i]]  <- rast(metriccoor[,c("x", "y", "basisofrecord_id")], type="xyz", crs=crs(gridin), extent=ext
  }
  # Assign name of species key to the corresponding raster 'layer'
  names(r) <- spskey

  return(r)
}
```

In the following code chunk, we compute metrics 4 and 5.

```
# Calculate metrics 4 and 5
min_dates_datasource <- datasource_bydate(min_dates_df, gridin, coorin, spskey)
max_dates_datasource <- datasource_bydate(max_dates_df, gridin, coorin, spskey)
```

```
## |---------|---------|---------|---------|=======================================
```

Saving data sets as individual tiffs

```
writeRaster(total_occ_sps, here("output/datacubes/tif_metrics/ias/01_ias_total_occurrences.tif"), datat
writeRaster(min_dates, here("output/datacubes/tif_metrics/ias/02_ias_earliest_date_records.tif"), datat
writeRaster(max_dates, here("output/datacubes/tif_metrics/ias/03_ias_latest_date_records.tif"), datatyp
writeRaster(min_dates_datasource, here("output/datacubes/tif_metrics/ias/04_ias_earliest_date_records_s
writeRaster(max_dates_datasource, here("output/datacubes/tif_metrics/ias/05_ias_latest_date_records_sou
```

Prepare taxonomy for the EBV Data Portal Due to the absence of data for 10 species, we prepared a new file including only taxonomic information of 77 species for our results.

```
# Filter taxonomy file for only species with data
# Import file for all IAS taxonomy and following the ebvcube format
# tax <- read.csv(here("input/data/ias/taxonomy/List87IAS_EU_match_gbif_synonyms_acceptedUsageKeys.csv"

# Subset only for species with data occurrences in GBIF
gbif_spx <- gbif_tax %>%
  filter(acceptedUsageKey %in% spskey)

# Sort the right taxonomic order for filling the EBV Data Portal requierement before saving
gbif_spx2 <- gbif_spx[,c("kingdom", "phylum", "class", "order", "family", "genus", "species", "usageKey
write.csv(gbif_spx2, here("input/data/ias/taxonomy/List77IAS_EU_gbif_acceptedUsageKey_ebvcube+key.csv")

gbif_spx2 <- gbif_spx[,c("kingdom", "phylum", "class", "order", "family", "genus", "species")]
write.csv(gbif_spx2, here("input/data/ias/taxonomy/List77IAS_EU_gbif_acceptedUsageKey_ebvcube.csv"), qu
```