Data mobilisation from GBIF to the EBV Data Portal for the Birds Directive Annex I

Notebook 03 - Calculation of Metrics for the Birds Directive Annex I Using Occurrence Cubes (Part I)

true

2024-08-21

Introduction

In this notebook, we calculate simple metrics for the EU Birds Directive Annex I based on species occurrence in GBIF. For this, a bird occurrence cube (see Notebook 01 in this repository) was previously created using the occurrence cube software developed by GBIF under the Biodiversity Building Blocks for Policy (B3) Project. Details of the data query in GBIF are available at doi: 10.15468/dl.uh84tp. We present three metrics below:

- Total number of occurrences
- Earliest month with occurrences across all years
- Latest month with occurrences across all years

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  917229 49.0    1866132 99.7  1189554 63.6
## Vcells 2174778 16.6    8388608 64.0  2927771 22.4
```

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

```r
library(lubridate) # for dates
library(terra) # for raster
library(ncdf4)
library(ggplot2)
library(sf)
library(stringr)
library(viridis)
```

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

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

```r
occcube <- "0062979-240626123714530"

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

# Import the taxonomy of all species listed in the Birds Directive Annex 1 based on gbif backbone taxon
gbif_tax <- read.csv(here("input/data/birds/taxonomy/list193birds_directive_annexi_allaccepted_usagekey

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

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

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

# Import and convert the EU borders to a data frame
borders_eu <- st_read(here("input/grid/shp/NUTS2021_3035.shp"))
```

```
## Reading layer 'NUTS2021_3035' from data source
##   'H:\B3\B-Cubed_data_mobilization\input\grid\shp\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
```

We now identify which species of the Birds Directive Annex I have data in our species occurrence cubes.

```r
# Find number of species
spskey <- unique(cin[["data"]][["taxonKey"]]) # "taxonKey" is the name of the species key in b3bgi
```

Metrics Computation

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

```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

  # Calculate separately the total occurrence of each species
  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))

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

    # print(length(unique(metriccoor$x)))  # only for checking purposes

    # Check if there are too few occurrences in the dataset
    # Only 1 occurrence cannot be rasterized, and an `extent` error might occur
    if(length(unique(metriccoor$x)) < 10){
      # Add second empty point
      x <- metriccoor$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
    }

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

  return(r)
}
```

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

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

# Save file
writeRaster(total_occ_sps, here("output/datacubes/tif_metrics/birds/01_birds_total_occurrences.tif"), da
```

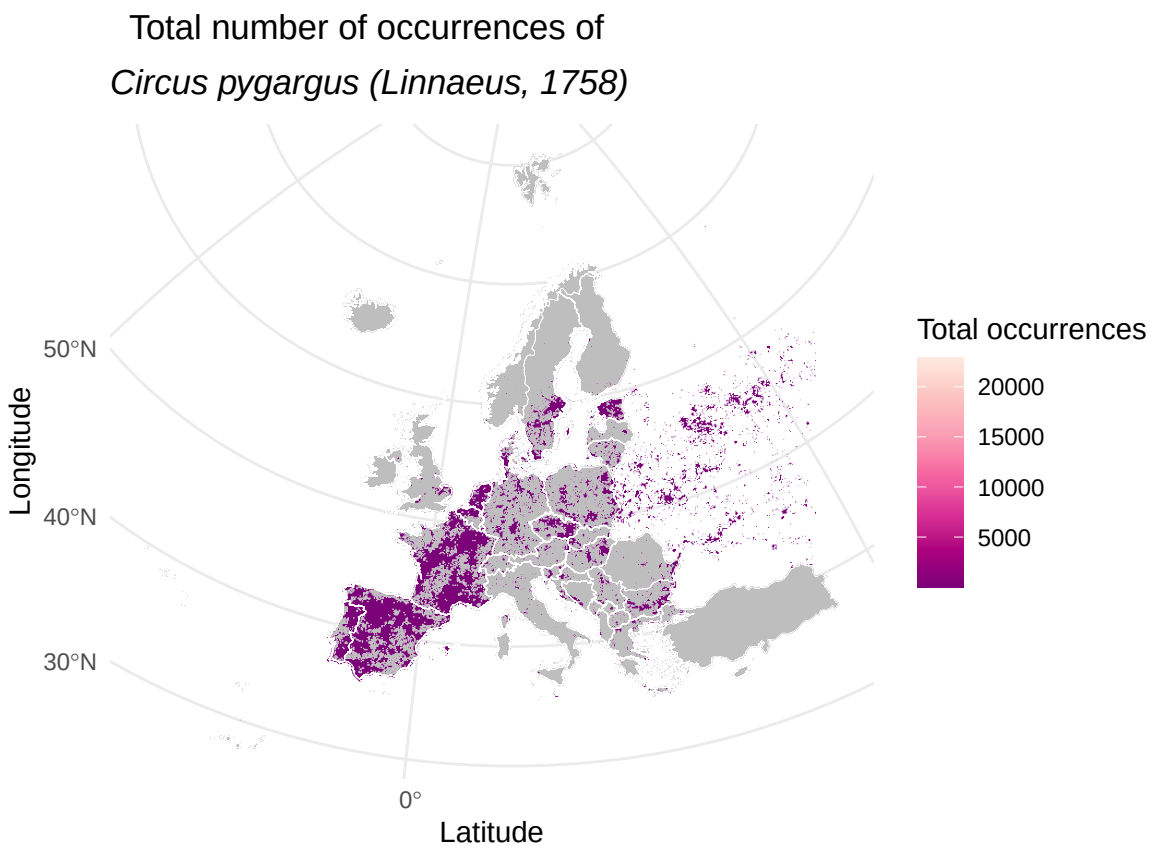
Prepare data for plotting

```r
# Select one specie
spi <- 57

# 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
spi_name <- gbif_tax %>%
  filter(acceptedUsageKey %in% spid)
```

```r
ggplot() +
  geom_sf(data = borders_eu, fill = "gray") +
  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), "per pixel"))),
    x = "Latitude",
    y = "Longitude",
    fill = "Total <br> occurrences")
```



```
# save figure if needed
ggsave(here("output/figures/birds_ias/fig1_ias_grey_id82.png"))
```

x #### **Metric 2 and 3:** Earliest month with occurrences across all years at GBIF

We calculate the earliest (latest) month of occurrence across all years for each species. The function `min_max_month` finds the earliest (latest) month of records per species at the pixel level across the entire dataset.

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

  # Create empty raster for metric 2
  rmin <- rast(ext(gridin), resolution=res(gridin), nlyrs=length(spskey), crs=crs(gridin))
  values(rmin) <- NA

  # Create empty raster for metric 3
  rmax <- rast(ext(gridin), resolution=res(gridin), nlyrs=length(spskey), crs=crs(gridin))
  values(rmax) <- NA


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

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

  # Convert date from character to numeric
  todates <- as.data.frame(str_split(spsi$yearMonth, "-", simplify = TRUE))
  colnames(todates) <- c("year", "month")

  # Add year and month columns separate to the initial data
  spsi$year <- todates$year
  spsi$month <- todates$month

 # Calculate metric 2. In this example earliest month of record
    metric_min <- spsi %>%
      group_by(cellCode) %>%
      slice_min(month, with_ties = FALSE) %>%
      ungroup()

# Calculate metric 3. In this example latest month of record
    metric_max <- spsi %>%
      group_by(cellCode) %>%
      slice_max(month, with_ties = FALSE) %>%
      ungroup()

    metricx_coor <- function(metricx, coorin, res){

      # Join pixel coordinates with the corresponing EEA grid ID
      metriccoorx <- merge(metricx[,c("month", "cellCode")], coorin, by="cellCode")
      # print(length(unique(metriccoorx$x)))

      if(length(unique(metriccoorx$x)) < 10){
```

```
        #add second empty point
        x <- metriccoorx$x[1] + res
        y <- metriccoorx$y[1] + res
        metriccoorx <- rbind(metriccoorx, c(NA, NA ,NA, x, y))
      }

      return(metriccoorx)
    }
  metriccoor_min <- metricx_coor(metric_min, coorin, res)
  metriccoor_max <- metricx_coor(metric_max, coorin, res)

  # Rasterize data
  rmin[[i]]  <- rast(metriccoor_min[,c("x", "y", "month")], type="xyz", crs=crs(gridin), extent=ext(gri

  # Rasterize data
  rmax[[i]]  <- rast(metriccoor_max[,c("x", "y", "month")], type="xyz", crs=crs(gridin), extent=ext(gri

  }

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


  results_list <- list(earliest_month = rmin, latest_month = rmax)
  return(results_list)

}
```

In the following code chunk, we compute metrics 2 and 3.

```
earliest_latest_months <- min_max_month(cin, gridin, coorin)
```

Note that the min_max_month function returns a list. The first element of the list is a raster containing the earliest month of the records. The second element is a raster containg the latest month of records.

```
# Extract raster of earliest month of records
earliest_month <- earliest_latest_months[[1]]

# Extract raster of latest month of records
latest_month <- earliest_latest_months[[2]]
```

```
# Save fille
writeRaster(earliest_month, here("output/datacubes/tif_metrics/birds/02_birds_earliest_month_records.ti
writeRaster(latest_month, here("output/datacubes/tif_metrics/birds/03_birds_latest_month_records.tif"),
```

Prepare data for plotting.

```
# Convert raster layers to data frames
spi <- 57

df1 <- as.data.frame(earliest_month[[spi]], xy=TRUE)
```

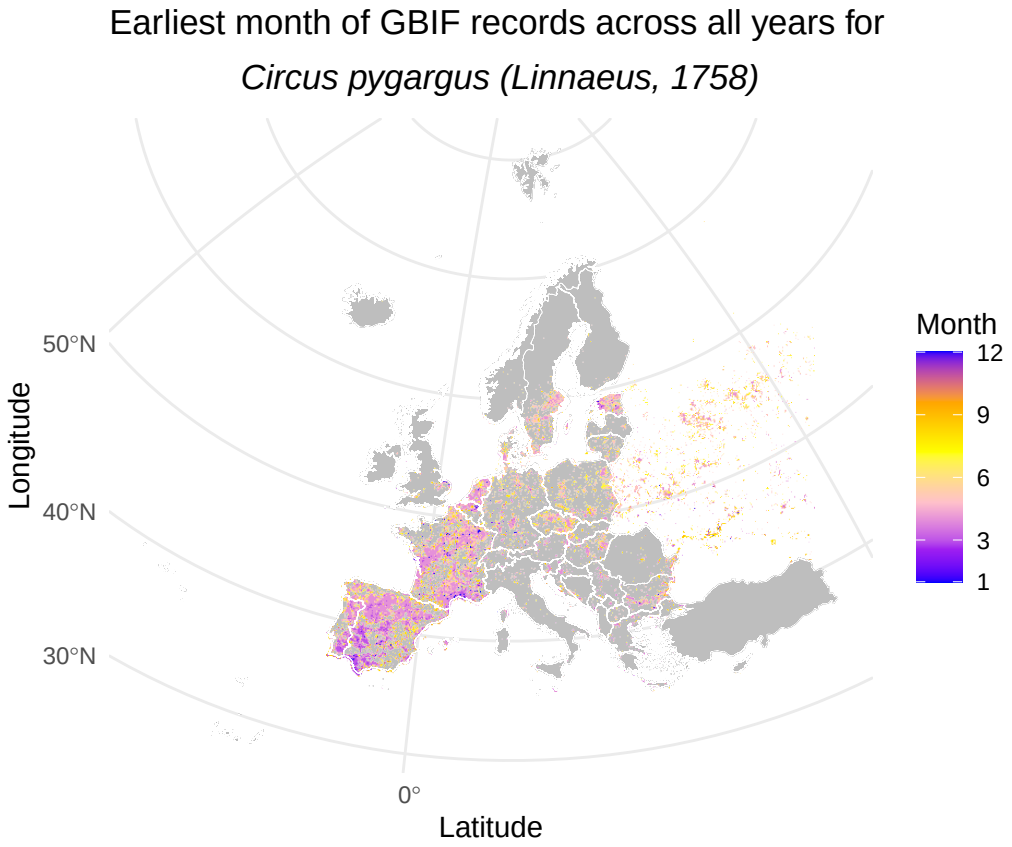
```r
spid <- as.numeric(colnames(df1)[3])
colnames(df1)[colnames(df1) == colnames(df1)[3]] <- "Value"

df2 <- as.data.frame(latest_month[[spi]], xy=TRUE)
colnames(df2)[colnames(df2) == colnames(df2)[3]] <- "Value"
df2[["Value"]] <- df2[["Value"]]

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

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

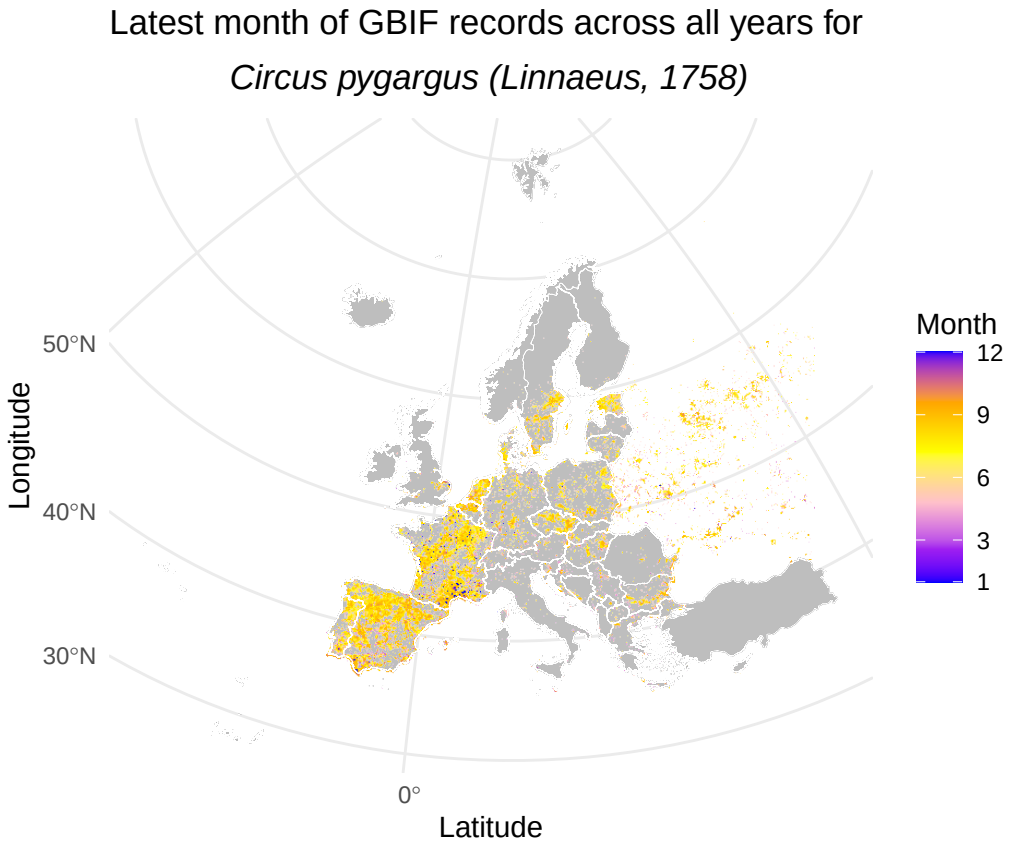
```r
ggplot() +
  geom_sf(data = borders_eu, fill = "gray") +
  geom_raster(data = df1, aes(x = x, y = y, fill = Value)) +
  geom_sf(data = borders_eu, fill = NA, color = "white") +
  scale_fill_gradientn(
    name = "Month",
    colors = c("blue", "purple", "pink", "yellow", "orange", "blue"), # Cyclic colors
    values = scales::rescale(c(1, 3, 6, 9, 12)), # Rescale for cyclic effect
    breaks = c(1, 3, 6, 9, 12)
  ) +
  theme_minimal() +
  labs(
    title = bquote(atop("Earliest month of GBIF records across all years for ",
                        italic(.(spi_name$scientificName), "per pixel"))),
    x = "Latitude",
    y = "Longitude",
    fill = "Month")
```




```
# save figure if needed
ggsave(here("output/figures/birds_ias/fig2_ias_grey_id82.png"))
```

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

```
ggplot() +
  geom_sf(data = borders_eu, fill = "gray") +
  geom_raster(data = df2, aes(x = x, y = y, fill = Value)) +
  geom_sf(data = borders_eu, fill = NA, color = "white") +
  scale_fill_gradientn(
    name = "Month",
    colors = c("blue", "purple", "pink", "yellow", "orange", "blue"), # Cyclic colors
    values = scales::rescale(c(1, 3, 6, 9, 12)), # Rescale for cyclic effect
    breaks = c(1, 3, 6, 9, 12)
  ) +
  theme_minimal() +
  labs(
     title = bquote(atop("Latest month of GBIF records across all years for ",
                         italic(.(spi_name$scientificName), "per pixel"))),
    x = "Latitude",
    y = "Longitude",
    fill = "Month")
```




```
# save figure if needed
ggsave(here("output/figures/birds_ias/fig3_ias_grey_id82.png"))
```