Download_GBIF-Data

Max

GBIF-Daten mit R herunterladen

Geht das auch ohne Login?

Erstmal mit Login:

```
#Libraries:
library(rgbif) # für download/datenbeschaffung
library(dplyr) #für df und aufräumen
library(sf) # für umwandlung in räumliche daten und export (geojson)

#Working directory:
setwd("~/CAU-Zeug/UGM/B-Bereich/Umweltmodellierung/Projekt_Hobby/UW_Mod_R")
```

usageKey und andere Infos zu den gewünschten Daten abgreifen:

```
#Das soll die GBIF interne ID der Art beschaffen
usageKey_Species <- rgbif::name_backbone(name = "Amanita phalloides")
usageKey_Species$usageKey
```

```
[1] 5240325
```

```
# Fragt die Datensätze nach den folgenden Kriterien ab
data_info <- occ_search(taxonKey = usageKey_Species$usageKey,
                        year = '2020, 2025',
                        hasCoordinate = TRUE,
                        country = 'DE',
                        hasGeospatialIssue = FALSE,
                        basisOfRecord = 'HUMAN_OBSERVATION',
                        )

data_info$meta$count #Gibt die Menge der "verfügbaren" Datensätze nach den Kriterien wieder.
```

```
[1] 1343
```

Download der Daten:

```
# Laut Copilot heißt das "Paging", also die Datenbank, Seite für Seite durchgehen.
# Ich würde einen for-Loop nehmen. Geht aber bestimmt auch anders

limit <- 500 # maximale Menge Daten, die in einem Rutsch abgegriffen werden können
total <- data_info$meta$count # Die Menge der abgefragten Daten
pages <- ceiling(total/limit) # Die Anzahl der nötigen Zugriffe, bis alle Daten abgesammelt w

data_gesammelt <- list()  # Noch leere Liste, die später mit den Seiten gefüllt wird

for (seite in seq_len(pages)) {
  start <- (seite -1)* limit
  cat("Sammle Seite ", seite, "von", pages, "...\n")

  data_abfrage <- rgbif::occ_search(taxonKey = usageKey_Species$usageKey,
                        year = '2020, 2025',
                        hasCoordinate = TRUE,
                        country = 'DE',
                        hasGeospatialIssue = FALSE,
                        basisOfRecord = 'HUMAN_OBSERVATION',
                        limit = limit,
                        start = start
                        )

  data_gesammelt[[seite]] <- data_abfrage$data #die einzelnen Seiten liegen jetzt als separat
  data_gesammelt[[seite]] <- dplyr::select(data_abfrage$data,
  occurrenceID, scientificName, decimalLatitude, decimalLongitude, eventDate, year, institut

}
```

```
Sammle Seite  1 von 3 ...
Sammle Seite  2 von 3 ...
Sammle Seite  3 von 3 ...
```

```
# Spalten können variieren - kp warum - deshalb mit dplyr::bind_rows das Ganze vereinheitlic
df_data <- dplyr::bind_rows(data_gesammelt)

# Mit rbind ein ganzes DF erstellen
```

```r
df_data <- do.call(rbind, data_gesammelt)

# Zahl der Zeilen des finalen DF = Anzahl der tatsächlich geladenen Datensätze (sollte = data
nrow(df_data)
```

```
[1] 1343
```

Jetzt wo die Daten heruntergeladen sind, müssen sie noch in eine GIS-handlebare Form übertragen werden -> GeoJSON

```r
# mit dem sf-Paket wird das df zu einem geo-df, reprojeziert und dann als geojson exportiert

geo_df_data <- sf::st_as_sf(df_data,
                            coords = c("decimalLongitude", "decimalLatitude"),
                            crs = 4326 # standart bei GBIF
                            )

geo_df_data_reproj <- sf::st_transform(geo_df_data, crs = 25832) # weil Standart für fast al

## Speichern als GeoJSON:
dir.create("Data_saved") #erzeugt das Directory für die geojson
st_write(geo_df_data_reproj, "./Data_saved/Species_Data.geojson", driver = "GeoJSON", delete
```

```
Deleting source `./Data_saved/Species_Data.geojson' using driver `GeoJSON'
Writing layer `Species_Data' to data source
  `./Data_saved/Species_Data.geojson' using driver `GeoJSON'
Writing 1343 features with 6 fields and geometry type Point.
```