Data compilation for sandy beach species red list assessments: an R script

AUTHOR
Linda Harris

AFFILIATION
Nelson Mandela University

PUBLISHED
2025-07-04

Attribution

This code is the supplementary material to:
Harris, L.R., Holness, S.D., Skowno, A.L., Sink, K.J., Raimondo, D. 2025. Sandy beach ecosystem and species red listing highlight priorities for beach conservation and restoration. Estuarine, Coastal and Shelf Science.

The code is also available at: https://github.com/lrharris/species-occurrences-for-red-listing

Background

Red list assessments require a spatial dataset of the species' distributions because many of the criteria rely on spatially explicit measures, e.g., number of locations, Area of Occupancy (**AOO**), Extent of Occurrence (**EOO**), whether it has a restricted range, etc. For this study, the sandy beach species distributions are compiled as point data from from GBIF, iNaturalist, field sampling and published records in the literature that are not on GBIF or iNaturalist.

AOO: The area occupied by a species, calculated using a standard 2x2 km^2 grid
EOO: The total area within which a species occurs, measured as a minimum convex polygon around all localities

For repeatability, transparency, and efficiency, the data were compiled in R into a dataset that is in the format required by the IUCN. The code used to compile and output the data is provided below.

Get setup

Load packages

The following packages are required to run this script.

▼ Code

```
library(rgbif)
library(rinat)
library(naniar)
library(here)
library(data.table)
library(sf)
library(redlistr)
library(tidyverse)
```

Load shore data

Data in online repositories like GBIF and iNaturalist can be spatially inaccurate. Because these data are used to calculate the AOO, which informs the species' threat status, it is important to exclude erroneous or inaccurate records. In this case, all data were filtered spatially to include only those records that are on the shore, and to exclude records that are inland or in the sea where the species are known not to occur.

In this study, we used a 600 m buffer around the ESRI World Countries shapefile, subset to the Western Indian Ocean (WIO). The buffer was set to 600 m because we wanted to include as many records as possible, also accounting for the accuracy of the

shoreline in the ESRI World Countries shapefile, but without artificially increasing the AOO. The global shapefile was cut to the WIO to make the analyses run quicker.

Load the spatial data of the shores.

▼ Code

```
shore <- st_read(here(
  "data/WorldCountries_ESRI_11052015_SWIO_pl_600m_d.shp")) %>%
  st_cast(., "MULTIPOLYGON")
```

List species to be assessed

The next step is to decide which species will be assessed. For this first analysis, we chose a selection of intertidal and supralittoral species; direct developers and those with pelagic larvae, focusing on species where there is currently no taxonomic uncertainty.

Create the species list.

▼ Code

```
species <- c("Acanthoscelis ruficornis",
             "Africorchestia quadrispinosa",
             "Bullia digitalis",
             "Bullia mozambicensis",
             "Bullia natalensis",
             "Bullia rhodostoma",
             "Capeorchestia capensis",
             "Donax serra",
             "Emerita austroafricana",
             "Excirolana latipes",
             "Excirolana natalensis",
             "Griffithsius latipes",
             "Latona madagascariensis",
             "Latona sordida",
             "Ocypode madagascariensis",
             "Ocypode ryderi",
             "Pachyphaleria capensis",
             "Tylos capensis",
             "Tylos granulatus")

# species <- "Ocypode ceratophthalmus"
```

Ocypode ceratophthalmus is listed separately because it was treated slightly differently as a result of its much larger distribution. Therefore, the code was run twice - once for the first 19 species, and once for *O. ceratophthalmus* with a few tweaks to the code (explained below).

Verify data in iNaturalist

We could have simply used the data from iNaturalist that are **Research Grade**. However, that would have meant losing lots of records of species (many of which are our own observations) where we are confident of the ID but there hasn't been confirmation of the ID by the iNaturalist community. Therefore, because we wanted to use all confirmed records of species, even if they were not Research Grade, we used the 'Alive or Dead' field as the tag for records that we wanted to include in the analysis. (Fortunately - in this case - people almost never fill in these annotations on iNaturalist).

Research Grade data: Observations in iNaturalist where the community agrees on the ID of the species, and at least 2/3 of the identifiers agree on the species name. The observation must also have a date, photograph, and a GPS location, and the observation must be of a wild specimen (i.e., not captive or cultivated). Depending on the user-assigned photo-sharing licence, Research Grade data get shared to GBIF. See more details on Research Grade data in iNaturalist and on GBIF.

Most of the data verification is done in iNaturalist itself, and came primarily from the Sandy Beaches (s Afr) project. All records that we wanted to include for a species were tagged as 'Alive'. We chose to exclude dead individuals because these animals or their shells can be washed up on beaches where they don't occur, and therefore we tagged 'Dead' individuals as well so that

they could be filtered out. Records where the species had been identified but we weren't confident of the identification were left unannotated in the 'Alive or Dead' field.

Because there are so many records of some of the species, it is important to make sure that none was overlooked. There isn't a way to filter observations in iNaturalist by records that have no annotation for the 'Alive or Dead' field, so we had to check in R to make sure each record had been verified, and that the records with no annotation are genuinely records that we did not want to include in the analysis. The process was as follows.

For each species, get the records and filter by both the ones that have been tagged as 'Alive' and 'Dead'. Sometimes there are no records of dead individuals; in that case, it's only necessary to filter out the observations of 'Alive' individuals. Check the records that are returned in the resulting tables (either inat_data1 or inat_data2) using the url field for each record, and annotate the 'Alive or Dead' field where necessary. Then rerun the code chunk to check all the records have been verified, either leaving no records in inat_data1 or inat_data2, or leaving only records that should not be included in the analysis.

Repeat this step for all species.

▼ Code

```
sp <- "..." #species name, e.g., "Acanthoscelis ruficornis"

inat_data <- get_inat_obs(taxon_name = sp, maxresults=10000)        #all
inat_data_a <- get_inat_obs(taxon_name = sp, annotation = c(17,18), #alive
                            maxresults=10000)
inat_data_d <- get_inat_obs(taxon_name = sp, annotation = c(17,19), #dead
                            maxresults=10000)
inat_data1 <- inat_data %>% filter(!(id %in% inat_data_a$id))   #filter out alive
inat_data2 <- inat_data1 %>% filter(!(id %in% inat_data_d$id))  #filter out dead
rm(sp, inat_data, inat_data_a, inat_data_d, inat_data1, inat_data2) #remove these objects and rerun for the next
         species
```

For *O. ceratophthlamus*, there were thousands of records (n=3801 on the date of analysis), which would have been challenging to review individually. Because 3192 observations were research grade (84% of all records), this was considered a sufficient sample of the data to use in the red list assessment. Research Grade data from iNaturalist get shared with GBIF only if the photo-sharing credits allow this. Therefore, we still downloaded the research grade data separately from iNaturalist, removed iNaturalist records from the GBIF dataset, and combined them.

Set up the download from GBIF

To pull GBIF data from the website, set up your GBIF login credentials (first create an account on GBIF if you don't already have one):

▼ Code

```
user = "..."
pwd = "..."
email = "..."
```

Then get the best name match from GBIF for each of the species and compile these into a list:

▼ Code

```
species_gbif_list <- list()

for (i in 1:length(species)){
sp_name = name_suggest(q = species[i], rank = "species")
species_gbif_list[[i]] <- sp_name$data
}
species_gbif_list
```

```
[[1]]
# A tibble: 1 × 3
      key canonicalName            rank
    <int> <chr>                    <chr>
```

```
1 7973496 Acanthoscelis ruficornis SPECIES

[[2]]
# A tibble: 1 × 3
      key canonicalName                rank
    <int> <chr>                        <chr>
1 7546054 Africorchestia quadrispinosa SPECIES

[[3]]
# A tibble: 1 × 3
      key canonicalName    rank
    <int> <chr>            <chr>
1 4368218 Bullia digitalis SPECIES

[[4]]
# A tibble: 1 × 3
      key canonicalName        rank
    <int> <chr>                <chr>
1 4368225 Bullia mozambicensis SPECIES

[[5]]
# A tibble: 1 × 3
      key canonicalName     rank
    <int> <chr>             <chr>
1 4368221 Bullia natalensis SPECIES

[[6]]
# A tibble: 1 × 3
      key canonicalName     rank
    <int> <chr>             <chr>
1 4368223 Bullia rhodostoma SPECIES

[[7]]
# A tibble: 1 × 3
      key canonicalName          rank
    <int> <chr>                  <chr>
1 9131682 Capeorchestia capensis SPECIES

[[8]]
# A tibble: 1 × 3
      key canonicalName rank
    <int> <chr>         <chr>
1 5729018 Donax serra   SPECIES

[[9]]
# A tibble: 1 × 3
      key canonicalName          rank
    <int> <chr>                  <chr>
1 4312457 Emerita austroafricana SPECIES

[[10]]
# A tibble: 1 × 3
      key canonicalName      rank
    <int> <chr>              <chr>
1 2212915 Excirolana latipes SPECIES

[[11]]
# A tibble: 1 × 3
      key canonicalName         rank
    <int> <chr>                 <chr>
1 2212921 Excirolana natalensis SPECIES

[[12]]
# A tibble: 1 × 3
      key canonicalName        rank
    <int> <chr>                <chr>
```

```
1 6462797 Griffithsius latipes SPECIES

[[13]]
# A tibble: 1 × 3
       key canonicalName           rank
     <int> <chr>                   <chr>
1 12266465 Latona madagascariensis SPECIES

[[14]]
# A tibble: 1 × 3
       key canonicalName  rank
     <int> <chr>          <chr>
1 12249129 Latona sordida SPECIES

[[15]]
# A tibble: 1 × 3
      key canonicalName            rank
    <int> <chr>                    <chr>
1 5971893 Ocypode madagascariensis SPECIES

[[16]]
# A tibble: 1 × 3
      key canonicalName  rank
    <int> <chr>          <chr>
1 4382445 Ocypode ryderi SPECIES

[[17]]
# A tibble: 1 × 3
      key canonicalName          rank
    <int> <chr>                  <chr>
1 7418745 Pachyphaleria capensis SPECIES

[[18]]
# A tibble: 2 × 3
       key canonicalName       rank
     <int> <chr>               <chr>
1  2205117 Tylos capensis     SPECIES
2 11168401 Tylosaurus capensis SPECIES

[[19]]
# A tibble: 2 × 3
      key canonicalName    rank
    <int> <chr>            <chr>
1 2205118 Tylos granulatus SPECIES
2 9020582 Tylos granulatus SPECIES
```

After checking these, in all cases, the first (generally only) record is the name match we need to use.

Create a table to record the metadata

Create an empty table with the fields that we want to record for each species. This is particularly important for keeping track of the GBIF doi and citation per species' dataset so that they can be appropriately reported in the red list assessments. The iNaturalist observations are cited according to their guidelines. There are too many observations from iNaturalist to cite each observation separately, so a collective citation is provided per species, but the url to each observation and the person who shared the record are included in the species' dataset.

▼ Code

```
meta_table <- tibble(download_key = character(), #GBIF download key
                     species = character(),      #species name
                     doi = character(),          #GBIF download doi
                     citation = character(),     #GBIF data citation
                     eoo_km2 = numeric(),        #initial EOO
                     aoo_km2 = numeric())        #initial AOO
```

Compile a dataset per species

Now that everything is set up, for each species, download the GBIF and iNaturalist data, and combine these with separate csv files of species occurrences from any field sampling or records from the literature that are not in GBIF or iNaturalist. Then filter the records to make sure that only those records that are within reasonable range of the shore are included - those far inland and in the sea are too inaccurate to include, and need to be excluded so that they don't artificially increase the EOO and AOO. (For example, sometimes the location of GBIF records are the museum in which they are housed rather than the place where the specimen was observed or collected.)

The data are saved and exported in the point data format required by the IUCN (both before and after the data are filtered spatially in case the records that were filtered out need to be checked). The only deviation is that the GPS locations are saved in fields called 'latitude' and 'longitude' rather than 'dec_lat' and 'dec-long', respectively, otherwise they cannot be read into GeoCAT (see below).

An initial EOO and AOO are calculated. This isn't really necessary and just serves as a guide because the data are later imported into GeoCAT to calculate the EOO and AOO. In GeoCAT, any erroneous records are removed (e.g., GBIF records beyond the known distribution of the animal) and records where species are no longer found (possibly extinct or extinct) or where species presence is uncertain are removed before calculating the EOO and AOO, as per the IUCN guidelines.

Look out for the code edit under the iNaturalist data to show how it was done for *O ceratophthalmus*.

▼ Code

```
# And off we go ------------------------------------------------------------------
for (i in 1:length(species_gbif_list)){
species_name <- species_gbif_list[[i]]$canonicalName[1]


# Download GBIF data -------------------------------------------------------------
## Set up the download ----
sp_download = occ_download(
  pred_in("taxonKey", species_gbif_list[[i]]$key[1]),
  pred("hasCoordinate", TRUE), # Select only records with coordinates
  format = "SIMPLE_CSV",
  user = user, pwd = pwd, email = email)

## Get the download key and request the data ----
sp_download
d_key=sp_download[[1]]

## Wait for the dataset to be compiled ----
still_running <- TRUE
status_ping <- 3
while (still_running) {
  meta <- occ_download_meta(key = d_key)
  status <- meta$status
  still_running <- status %in% c("PREPARING", "RUNNING")
  Sys.sleep(status_ping) # sleep between pings
}

## Save the download metadata ----
meta_table[i,1] <- d_key
meta_table[i,2] <- species_name
meta_table[i,3] <- occ_download_meta(key = d_key)$doi
meta_table[i,4] <- paste0("GBIF.org (", Sys.Date(),
                          ") GBIF Occurrence Download https://doi.org/",
                          occ_download_meta(key = d_key)$doi)

## Download the dataset ----
sp_download = occ_download_get(key = d_key,
                               path = here("data"),
                               overwrite = TRUE) %>%
```

```
  occ_download_import(sp_download, na.strings = c("", NA))
unzip(zipfile=here("data", paste0(d_key, ".zip")), exdir="data")

## Read the data into R and remove data from iNaturalist ----
gbif_data = fread(here("data", paste0(d_key, ".csv")),
                  data.table = FALSE, fill = F, encoding ="UTF-8", quote="") %>%
  mutate(latitude = decimalLatitude, longitude=decimalLongitude,
         source=paste0("GBIF.org ", Sys.Date(), "; https://gbif.org/occurrence/", gbifID),
         quality_grade="research") %>% #GBIF citation date = date of download
  dplyr::rename(recordedby=recordedBy, basisofrec=basisOfRecord) %>%
  filter(!(institutionCode=="iNaturalist")) %>%
  dplyr::select(basisofrec, catalogNumber, latitude, longitude, year,
                scientificName, source, recordedby, quality_grade)

# Download iNaturalist data ------------------------------------------------
  try(
inat_data <- get_inat_obs(taxon_name = species_name,
                          annotation = c(17,18),  #see note below
                          #quality = "research",  #see note below
                          maxresults=10000)  %>%
  replace_with_na(., replace = list(user_name="")) %>%
  mutate(recordedby=coalesce(user_name, user_login),
         observed_on=as.POSIXlt(observed_on, format="%Y-%m-%d"),
         observed_on=format(observed_on, format="%Y")) %>%
  dplyr::rename(scientificName=scientific_name, year=observed_on,
                catalogNumber=id) %>%
  mutate(basisofrec="HUMAN_OBSERVATION",
         source=paste0("iNaturalist community 2024; ", url)) %>%
  dplyr::select(basisofrec,catalogNumber, latitude, longitude, year,
                scientificName, source, recordedby, quality_grade) %>%
  filter(!(is.na(latitude))))

  if(exists("inat_data")=="FALSE") {
    inat_data<-as_tibble(matrix(nrow = 0, ncol = length(names(gbif_data))),
                                    .name_repair = ~ names(gbif_data))
  }

#note: To download all research grade data instead of downloading records of alive individuals e.g., for O
         ceratophthalmus, exclude the line:  annotation = c(17,18),  and uncomment the line below it:  quality =
         "research",

# Read in field sampling data ----------------------------------------------
other_data <- read_csv(here("data",
                            paste0(species_name, "_distribution-records.csv")),
                       show_col_types = FALSE) %>%
  rename_with(tolower) %>%
  mutate(basisofrec="HUMAN_OBSERVATION",
         catalogNumber="NA",
         source=collector,
         quality_grade = "research") %>%
  dplyr::rename(scientificName=identification, recordedby=collector) %>%
  select(-origin)


# Combine datasets ---------------------------------------------------------
all_data <- rbind(gbif_data, inat_data, other_data) %>%
  mutate(spatialref="WGS84",
         yrcompiled=substr(Sys.Date(),1,4),
         presence=1,
         compiler="Linda R. Harris",
         citation="Linda R. Harris", #citation for the dataset
         seasonal=1,
         data_sens=0,
         origin=1) %>%
  mutate(quality_grade=factor(quality_grade)) %>%
  rename(sci_name=scientificName, catalog_no=catalogNumber, event_year=year) %>%
```

```
  mutate(sci_name=species_name) %>%
  select(sci_name, presence, origin, seasonal, compiler, yrcompiled, citation,
         latitude, longitude, spatialref, data_sens, event_year, source,
         basisofrec, catalog_no, recordedby)

write_csv(all_data, here("outputs/unfiltered_data",
                         paste0(species_name, "_unfiltered.csv")))


# Filter data based on presence in the shore polygon ---------------------------
#note: skip this section for O ceratophthalmus - the distribution is larger than the shore polygon created for the
         WIO. This step was performed in ArcGIS 10.6 because it was substantially faster than in R.
spdf <- st_as_sf(x = all_data,
                 coords = c("longitude", "latitude"),
                 crs = "+proj=longlat +datum=WGS84") %>%
  st_transform(., st_crs(shore))

sp_dat_filter <-  spdf %>%
  mutate(include = as.numeric(st_intersects(spdf, shore))) %>%
  filter(include>0) %>%
  mutate(id=seq(1:nrow(.)))


# Write output ------------------------------------------------------------------
st_write(sp_dat_filter, paste0(species_name, "_temp.csv"),
         layer_options = "GEOMETRY=AS_XY")
dat <- read_csv(paste0(species_name, "_temp.csv"),show_col_types = FALSE) %>%
  dplyr::rename(latitude=Y, longitude=X) %>%
  select(-include) #add -id if you want to exclude the id field too)
write_csv(dat, here("outputs/final_data", paste0(species_name, "_distribution.csv")))
file.remove(paste0(species_name, "_temp.csv"))


# Calculate EOO and AOO (just an initial estimate, actual calculations done in geocat) ---
sp_projected <- sp_dat_filter %>%
  st_transform(., 9822) %>%
  as(., "Spatial")

EOO <- makeEOO(sp_projected)
AOO <- makeAOOGrid(sp_projected, grid.size = 2000)

meta_table[i,5] <- getAreaEOO(EOO)
meta_table[i,6] <- length(AOO)*4

rm(gbif_data, inat_data, other_data)
}
```

Export the metadata

The final step is to save the metadata table as a csv file.

▼ Code

```
# Save the metadata file ---------------------------------------------------
write_csv(meta_table,
          here(paste0("outputs/metadata_table_", Sys.Date(), ".csv")))
```