1_Preparation.R

Iris Van Paemel

2024-05-17

Workflow introduction

This workflow (RGBIFanimals_IVP) is a revision of the original workflow rgbifanimals. (link to original workflow: https://github.com/DvBMolEc/rgbifanimals)

The goal of the RGBIFanimals_IVP workflow is to analyse data obtained from the Autonomous Reef Monitoring Structures - Marine Biodiversity Observation Network (ARMS-MBON) project. However, the aim is to make this workflow interoperable and shareable to marine biologists or anyone working in marine biology => in need of a method for detecting alien species.

The focus of this workflow is the detection of marine alien species or Non-Indigenous Species (NIS). These are species that travel great distances to a point that they are no longer in their natural habitat. These species are being detected with the ARMS project. There are three types of potential alien species: cryptogenic, hitchhiking and rande expanding species.

The most important step in this pipeline is fetching occurrence data from the Global Biodiversity Information Facility (GBIF) website/database and calculating distances with samplelocations from ARMS. This step is performed in the [new main script] and this is computationally very expensive, this step takes hours relative to the amount of data being analysed.

1_Preparation.R script

WARNING: This script focuses on outputfiles from the BOLDigger pipeline. Make sure to check this script before running. Some ARMS locations and taxonomy are hardcoded. The headers in the BOLDigger file consist of sample names that are not consistent and sometimes have year only or full dates. Some of them have ARMS fraction names (MF100, MF500, SF40).

Input

The input data consists of csv files (these csv files are saved in the directory Inputs/ which is located in the same directory as the scripts are in):

- BOLDigger_output.csv => Header consists of sequence, Phylum, Class, Order, Family, Genus, Species, Similarity and all ARMS filenames.
- MetaData.csv => Header consists of Sample_region_country, Country, Observartory-ID, ARMS-ID, Latitude, Longitude, Depth_m, Monitoring_area, Deployment_date, Retrieval_date, Days_deployed, Fraction and Filename Filename is also present in the BOLDigger output as columnnames.
- Synonyms.csv => Header consists of Old_name and New_name

Output

For the output of this script, a new directory "Output" is made. All of the csv files below are created:

- UniqueSpecies.csv => Header consists of Specieslist and Similarity
- ASV_Count.csv => Header consists of Specieslist and 'n' (n = count ASV)
- Read_Count_ASVs_ARMS.csv => Header consists of sequence, Phylum, Class, Order, Family, Genus, Species, Specieslist, Similarity and ARMS filenames in format: [Location].JJ.CD78.[dates].[ARMS fraction] without 'MF100', 'MF500' or 'SF40' (fractions)
- Read_Count_Species_ARMS.csv => Header consists of Specieslist and ARMS filenames
- Read_Count_Species_Loc_Year.csv => Header consists of Phylum, Class, Order, Family, Specieslist and ARMS filenames
- Read_Count_Species_Fraction.csv => Header consists of Specieslist and ARMS filenames with fractions.
- Data_LOI.csv => Header consists of sequence, Phylum, Class, Order, Family, Genus, Species, Specieslist, Similarity
- Species_Location.csv => Header consists of Specieslist, BelgianCoast, Getxo, Koster, Laeso, Limfjord, Plymouth, Roscoff, SwedishWestCoast, TZS, Vigo (Number of species detected with ARMS on specific location) #### next two files are made in directory Inputs/
- Inputs/Coordinates.csv => Header consists of Observatory.ID, Longitude and Latitude (latitudes and longitudes of ARMS)
- Inputs/MetaData_Adjusted.csv => Sample_region_country, Country, Observatory.ID, ARMS.ID, Latitude, Longitude, Depth_m, Monitoring_area, Deployment_date, Retrieval_date, Days_Deployed, Fraction, Filename, Year, Location_Year, No_Fraction

script + descriptions

- load dplyr and tidyverse packages (if not yet installed, install first)
- set working directory to the directory where this script is located (active document => opened)
- save the working directory in variable 'dir'
- load the BOLDigger output into variable 'Data'

```
#Load all necessary packages
library("dplyr")
library("tidyverse")

#Set wd to wherever the script files are stored
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))
dir <- getwd()

#Load BOLDigger output
Data <- read.csv("Inputs/BOLDigger_output.csv", sep = ";", row.names = NULL, header = TRUE)
```

FILTERING STEP (line 16 in 1_Preparation script)

- save the column names from Data in the variable 'Locations_BOLD'
- split the column names and place them in a dataframe
- save the ARMS-MBON location names in variable 'LOI' /! Replace with your own data
- Filter Locations_BOLD V1 column to locations that are present in LOI
- Locations_BOLD$V3 is made by pasting columns V1 and V2 together separated by a '.'

```
#############################################################################################
#FILTERING STEP
#filter columnnames of BOLDigger file on locations of interest: make dataframe Locations_BOLD
#in V3 of this dataframe: result of filtering and combining V2 and V1 again.
#############################################################################################

#Extract locations of interest from main Data frame
Locations_BOLD <- colnames(Data)
Locations_BOLD <- as.data.frame(stringr::str_split_fixed(string = Locations_BOLD, pattern = "\\.", 2)) 
LOI <- c("Toralla", "Getxo", "Vigo", "Roscoff", "Plymouth", "Galway", "BelgianCoast", "Bjorko", "Gbg",
         "Helsingborg", "Hjuvik", "Koster", "Laesoe1", "Laesoe2", "Laesoe3", "Limfjord", "Marstrand", "
         "Varberg", "Gdynia", "TZS")
Locations_BOLD <- Locations_BOLD %>% # %>% = pipe operator, joins filter function in next line
  filter(V1 %in% LOI)  # %in% checks if LOI is present in column V1 (for filtering of df)
#Paste colnames back together for later filtering steps
Locations_BOLD$V3 <- paste0(Locations_BOLD$V1, ".", Locations_BOLD$V2)
```

METADATA STEP (line 32 in 1_Preparation script)

- reading in the metadata from MetaData.csv
- joins MetaData and Locations_BOLD by the common columns 'Filename' and 'V3'
- remove V1 and V2 from MetaData and values that start with 'X'
- replacing blank values with 'NA' and remove NA values
- remove rows with missing values using complete.cases() function
- make a new column MetaData$Year with only the years from MetaData$Deployment_date
- add another column 'Location_Year' with Observatory.ID and Year split by '_'
- add another column 'No_Fraction' with filenames without ARMS fraction in it /! Replace with your own data

```
#############################################################################################
#METADATA STEP
#replace missing values to NA and remove NA values. Join V1 and V3 into MetaData
#############################################################################################

#### Metadata ####
MetaData <- read.csv("Inputs/MetaData.csv")
#Select applicable ARMS deployments
MetaData <- left_join(MetaData, Locations_BOLD, by = c("Filename" = "V3"))
# here you add V1 and V2 from Locations_BOLD to the table of metadata
#Clean-up of data
MetaData <- MetaData %>% dplyr::select(-starts_with("X"), -V1, -V2) # remove column V1, V2 and column s

MetaData[MetaData == ""] <- NA # Replace blank by NA

MetaData <- na.omit(MetaData)  # remove missing values
MetaData <- MetaData[complete.cases(MetaData), ]  # remove rows with missing values
MetaData$Year <- format(as.Date(MetaData$Deployment_date), "%Y") # add column Year with only years
### adding extra column with combination of Observatory.ID and year
MetaData$Location_Year <- paste0(MetaData$Observatory.ID, "_", MetaData$Year)
#Add column in which ARMS fraction is removed
MetaData$No_Fraction <- MetaData$Filename
MetaData$No_Fraction <- sub("\\.MF.00", "", MetaData$No_Fraction)
```

```
MetaData$No_Fraction <- sub("\\.MT.00", "", MetaData$No_Fraction)
MetaData$No_Fraction <- sub("\\.SF40", "", MetaData$No_Fraction)
```

METADATA CONTINUED (line 54 in 1_Preparation script)

- Make a variable with taxonomy
- Merge the genus and species together into specieslist, seperated by a space
- replace , with . in Data$Similarity
- substitute species names with synonyms using the synonyms.csv /! Replace with your own data
- First it selects in Data$Specieslist the old names that are present in Synonyms$Old_name and if there's a match, it replaces it with the new name
- Data_LOI is the filtered version of Data with "sequence", Taxonomy variable, "Specieslist", "Similarity" and MetaData$Filename variable
- contaminants, groups that are not of interest and sequences without observations are removed /! Replace with your own data
- reset row numbers

```
#############################################################################
#Determine taxonomy order
Taxonomy <- c("Phylum", "Class", "Order", "Family", "Genus", "Species")

#### Solve species name issues main df ####

#Merge Genus and species into Specieslist
Data$Specieslist <- paste(Data$Genus, Data$Species, sep = " ")

#Replace , with . in Similarity, to be able to assign numeric variable
Data$Similarity <- gsub(",", ".", x = Data$Similarity, fixed = T)
#Substitute species names with synonyms
Synonyms <- read.csv("Inputs/Synonyms.csv")
Old_name <- Synonyms$Old_name
New_name <- Synonyms$New_name
Data$Specieslist[Data$Specieslist %in% Old_name] <- New_name[match(Data$Specieslist, Old_name, nomatch =
rm(Old_name, New_name, Synonyms)

#Filter dataset based on locations of interest
Data_LOI <- Data[, c("sequence", Taxonomy, "Specieslist", "Similarity", MetaData$Filename)]
rm(Data)
# Remove contaminants ...
RemoveSpecies <- c("sapiens", "lupus", "scrofa")
Data_LOI <- Data_LOI[!(Data_LOI$Species %in% RemoveSpecies), ]
# ... and groups which are not of interest
RemovePhyla <- c("Amoebozoa", "Ascomycota", "Bacillariophyta", "Chlorophyta", "Heterokontophyta", "Ochr
Data_LOI <- Data_LOI[!(Data_LOI$Phylum %in% RemovePhyla), ]
# Remove sequences with no observations in our chosen subset
Data_LOI <- Data_LOI[rowSums(Data_LOI[, 10:ncol(Data_LOI)])>0, ]
#Reset rownumbers for easier subsetting
rownames(Data_LOI) <- c(1:nrow(Data_LOI))
rm(RemovePhyla, RemoveSpecies)
```

SET OF ADDITIONAL DATAFRAMES FOR ADDITIONAL INSIGHT (line 89 in 1_Preparation script)

- Extract unique species with their corresponding lowest similarity

  - select species & similarity from Data_LOI
  - order alphabetically
  - select distinct = remove duplicates
  - group data on specieslist
  - select rows with minimal similarity

- calculation of amplicon sequence variant count without duplicates
- make a new dataframe 'df' with column names, save column names in 'Column_Order'
- apply a function on the column names from df that are duplicated, but take the column name only 1 time. Then perform the function; merge columns with a '$' as a separator and sum up the rows. save this in df2
- Read_Count_ASVs_ARMS withholds df2 and not duplicated colnames of df
- Read_Count_Species_ARMS variable:
  - remove column names from sequence to similarity from Read_Count_ASVs_ARMS
  - select Specieslist and sort column names
  - group by Specieslist and summarise
- order columns by Column_Order
- Filtering Data_LOI in variable RC_Species_Loc_Year with specific columns from Taxonomy to Meta-Data
- change columnnames from this variable and save it into df
- merge duplicate columns again in the same way as last time in variable 'df2'
- group by Specieslist and sum read counts location
  - relocate columns phylum up until Specieslist to before 'BelgianCoast_2018'
  - turn columns 6 until the last column to numeric
  - group by taxonomy columns
  - summarise the numeric columns from BelgianCoast to Vigo
- making a matrix with presence and absence from the Read_Count_Species_ARMS dataframe. If the columns from the second one to the last one are higher than 0, then give them the value 1
- Read_Count_Species_Fraction
  - select Specieslist and sort colnames from Data_LOI
  - group by specieslist and summarise
- Export data and make all of the files.
- Build dataframe to be used in AlienIdentification.R => this script doesn't exist?
- Make a df with Specieslist and Observatory.ID
- Merge columns that are duplicate, same as before
- Export Coordinates to use in Main_script and metadata in Visualisation script

```
#########################################################################
#### Set of additional dataframes for additional insight ####
#########################################################################
#Extract unique species with their corresponding lowest Similarity
Uniques <- Data_LOI %>%
  dplyr::select(Specieslist, Similarity) %>%
  arrange(Specieslist) %>%   # order alphabetically on specieslist
  distinct() %>%    # select distinct, remove duplicates
  group_by(Specieslist) %>%   # group data on specieslist
  slice_min(order_by = Similarity, with_ties = T)  # select rows with minimal similarity
# in the groups of specieslist. Ties => if more than 1 row with same similarity, keep them

#ASV count for each species
ASV_Count <- Data_LOI[!duplicated(Data_LOI$sequence), ] %>% dplyr::count(Specieslist)

#Read count per ASV, per ARMS, per year (sum three fractions MF100, MF500, SF40)
#Create duplicate column names
df <- Data_LOI
colnames(df) <- c("sequence", Taxonomy, "Specieslist", "Similarity", MetaData$No_Fraction)
Column_Order <- unique(colnames(df))

#Merge columns with duplicate names and sum contents
df2 <- sapply(unique(colnames(df)[duplicated(colnames(df))]),
              function(x) rowSums(df[,grepl(paste(x, "$", sep=""), colnames(df))]))
Read_Count_ASVs_ARMS <- cbind(df2, df[,!duplicated(colnames(df)) &
                                      !duplicated(colnames(df), fromLast = TRUE)])
rm(df, df2)

#Read count per Species, per ARMS, per year
Read_Count_Species_ARMS <- Read_Count_ASVs_ARMS %>%
  dplyr::select(-sequence, -Phylum, -Order, -Class, -Family, -Genus, -Species, -Similarity) %>%
  dplyr::select(Specieslist, sort(colnames(.))) %>%
  group_by(Specieslist) %>%
  summarise_all(sum)
#Readable format
Read_Count_ASVs_ARMS <- Read_Count_ASVs_ARMS[, Column_Order]

#Read count per Location per Year
RC_Species_Loc_Year <- Data_LOI[, c(Taxonomy[1:4], "Specieslist", MetaData$Filename)]
#Create duplicate column names
colnames(RC_Species_Loc_Year) <- c(Taxonomy[1:4], "Specieslist", MetaData$Location_Year)
df <- RC_Species_Loc_Year
#Merge columns with duplicate names and sum contents
df2 <- sapply(unique(colnames(df)[duplicated(colnames(df))]),
              function(x) rowSums(df[,grepl(paste(x, "$", sep=""), colnames(df))]))
RC_Species_Loc_Year <- as.data.frame(cbind(df2, df[,!duplicated(colnames(df)) & !duplicated(colnames(df)
rm(df, df2)
#group by Specieslist and sum read counts locations
RC_Species_Loc_Year <- RC_Species_Loc_Year %>%
  relocate(Phylum:Specieslist, .before = BelgianCoast_2018) %>% # relocates columns phylum -> specieslis
  mutate_at(c(6:ncol(RC_Species_Loc_Year)), as.numeric) %>% #turn columns 6->end to numeric
  group_by(Phylum, Class, Order, Family, Specieslist,) %>%  # group by columns
  summarise(across(BelgianCoast_2018:Vigo_2019, sum))  #summarise columns belgiancoast->vigo
```

```r
#Presence/Absence matrix for Species per ARMS
Pres_Abs <- Read_Count_Species_ARMS
Pres_Abs[2:ncol(Pres_Abs)][Pres_Abs[2:ncol(Pres_Abs)] > 0] <- 1
# if nr. in columns 2-> last column are > 0, enter 1

#Read count per species per fraction
Read_Count_Species_Fraction <- Data_LOI  %>%
  dplyr::select(-sequence, -Phylum, -Order, -Class, -Family, -Genus, -Species, -Similarity) %>%
  dplyr::select(Specieslist, sort(colnames(.))) %>%
  group_by(Specieslist) %>%
  summarise_all(sum)

#Export
dir.create(file.path(dir, "Output"))
setwd(file.path(dir,"Output"))
write.csv(Uniques, "UniqueSpecies.csv", row.names = FALSE)
write.csv(ASV_Count, "ASV_Count.csv", row.names = FALSE)
write.csv(Read_Count_ASVs_ARMS, "Read_Count_ASVs_ARMS.csv", row.names = FALSE)
write.csv(Read_Count_Species_ARMS, "Read_Count_Species_ARMS.csv", row.names = FALSE)
write.csv(RC_Species_Loc_Year, "Read_Count_Species_Loc_Year.csv", row.names = FALSE)
write.csv(Read_Count_Species_Fraction, "Read_Count_Species_Fraction.csv", row.names = FALSE)
write.csv(Data_LOI, "Data_LOI.csv", row.names = FALSE)

#### Build dataframe to be used in AlienIdentification.R ####
Species_Location <- Read_Count_Species_Fraction
### THIS IS THE SAME AS Read_Count_Species_Fraction => removed duplicate code

colnames(Species_Location) <- c("Specieslist", MetaData$Observatory.ID) #make df Species_Location with
df <- Species_Location
#Merge columns with duplicate names and sum contents
df2 <- sapply(unique(colnames(df)[duplicated(colnames(df))]),
              function(x) rowSums(df[,grepl(paste(x, "$", sep=""), colnames(df))]))
Species_Location <- as.data.frame(cbind(df2, df[,!duplicated(colnames(df)) & !duplicated(colnames(df),
Species_Location <- Species_Location %>% dplyr::select(Specieslist, everything())
rm(df, df2)
write.csv(Species_Location, "Species_Location.csv", row.names = F)

#Export Coordinates, which are used in 3_Main_script and MetaData, which is used in 4_Visualisation
setwd(dir)
Coordinates <- MetaData %>% group_by(Observatory.ID) %>% dplyr::select(Observatory.ID, Longitude, Latitu
write.csv(Coordinates, "Inputs/Coordinates.csv", row.names = F)
write.csv(MetaData, "Inputs/MetaData_Adjusted.csv", row.names = F)

#### Clean R environment ####
rm(list = ls())
```