Macrozoobenthos presence/absence in European Seas

Peter M.J. Herman

2022-03-07

Introduction

This is a continuation and generalisation of the presence/absence data product on benthos of the Greater North Sea and of the Med Sea and Black Sea. In this product, all data of all European seas are analysed. The aim of this analysis is to derive information, not only on presences of taxa that are recorded in the dataset, but also (by inference) on absence. It is assumed that datasets which have analysed the complete benthic community, would have recorded all benthic species if they would have been present. Not finding a taxon in such a sample, is considered a recorded absence. For datasets that are known to have targeted only part of the community, absence can be inferred for those species that have been found elsewhere in the dataset, but not in a particular sample. The implementation of this principle is documented in this description.

Because of the size of the dataset (around 250,000 samples with over 10,000 taxa found), the analysis has been split up in different building blocks. Each block independently reads the necessary information from previous steps from files, and cleans up the memory after having completed its task. The large datafiles cannot be documented in GitHub, and will need to be rebuilt if this software is to be implemented again.

Preliminaries

The following code opens all packages used, sets some paths and defines the basic lon-lat projection used in this project. In addition, it reads some auxiliary functions that are stored in the script ./scripts/auxiliary__functions.R

Packages etc.

```
require(raster)
require(sp)
require(rgdal)
require(svMisc)
require(tidyverse)
require(sf)
require(worrms)
library(ggplot2)
library('rnaturalearth')
library(magick)
library(rgeos)
library(EMODnetBiologyMaps)
require(imis)
require(progress)
require(ncdf4)

#############
downloadDir <- "data/raw_data"
dataDir     <- "data/derived_data"
```

```r
mapsDir     <- "product/maps"
rasterDir   <- "product/species_rasters"
plotsDir    <- "product/species_plots"
#############
proWG<-CRS("+proj=longlat +datum=WGS84")
source("./scripts/auxiliary_functions.R")
```

Auxiliary functions

```r
emodnet_map_plot_2<-function (data, fill = NULL, title = NULL, subtitle = NULL,
                             legend = NULL, crs = 3035,
                             xlim = c(2426378.0132,7093974.6215),
                             ylim = c(1308101.2618, 5446513.5222),
                             direction = 1, option = "viridis",zoom = FALSE, ...)
{
  stopifnot(class(data)[1] %in% c("sf","RasterLayer",
                                  "SpatialPolygonsDataFrame",
                                  "SpatialPointsDataFrame",
                                  "SpatialLinesDataFrame"))
  emodnet_map_basic <- emodnet_map_basic(...) +
    ggplot2::ggtitle(label = title, subtitle = subtitle)
  if (class(data)[1] == "RasterLayer") {
    message("Transforming RasterLayer to sf vector data")
    data <- sf::st_as_sf(raster::rasterToPolygons(data))
    fill <- sf::st_drop_geometry(data)[, 1]
  }
  if (class(data)[1] %in% c("SpatialPolygonsDataFrame",
                            "SpatialPointsDataFrame", "SpatialLinesDataFrame")){
    message("Transforming sp to sf")
    data <- sf::st_as_sf(data)
  }
  if (sf::st_geometry_type(data, FALSE) == "POINT") {
    emodnet_map_plot <- emodnet_map_basic + ggplot2::geom_sf(data = data,
                                 color = emodnet_colors()$yellow)
  }
  if (sf::st_geometry_type(data, FALSE) == "POLYGON") {
    emodnet_map_plot <- emodnet_map_basic +
      ggplot2::geom_sf(data = data,ggplot2::aes(fill = fill),
                       size = 0.05,color=NA) +
      ggplot2::scale_fill_viridis_c(alpha = 0.8,
                                    name = legend, direction = direction,
                                    option = option)
  }
  if (zoom == TRUE) {
    bbox <- sf::st_bbox(sf::st_transform(data, crs))
    xlim <- c(bbox$xmin, bbox$xmax)
    ylim <- c(bbox$ymin, bbox$ymax)
  }
  emodnet_map_plot <- emodnet_map_plot + ggplot2::coord_sf(crs = crs,
                                                           xlim = xlim, ylim = ylim)
  return(emodnet_map_plot)
}
########## end of modified plot function - eventually to be moved into the package
```

```
########## function to read dataset characteristics

fdr2<-function(dasid){
  datasetrecords <- datasets(dasid)
  dascitations <- getdascitations(datasetrecords)
  if(nrow(dascitations)==0)dascitations<-tibble(dasid=as.character(dasid),
                                                title="",citation="")
  if(nrow(dascitations)==1) if(is.na(dascitations$citation)) dascitations$citation<-""
  daskeywords <- getdaskeywords(datasetrecords)
  if(nrow(daskeywords)==0)daskeywords<-tibble(dasid=as.character(dasid),
                                              title="",keyword="")
  if(nrow(daskeywords)==1) if(is.na(daskeywords$keyword))daskeywords$keyword<-""
  dascontacts <- getdascontacts(datasetrecords)
  if(nrow(dascontacts)==0)dascontacts<-tibble(dasid=as.character(dasid),
                                              title="",contact="")
  if(nrow(dascontacts)==1) if(is.na(dascontacts$contact))dascontacts$contact<-""
  dastheme <- getdasthemes(datasetrecords)
  if(nrow(dastheme)==0)dastheme<-tibble(dasid=as.character(dasid),
                                        title="",theme="")
  if(nrow(dastheme)==1) if(is.na(dastheme$theme))dastheme$theme<-""
  dastheme2 <- aggregate(theme ~ dasid, data = dastheme, paste,
                         collapse = " , ")
  daskeywords2 <- aggregate(keyword ~ dasid, data = daskeywords,
                            paste, collapse = " , ")
  dascontacts2 <- aggregate(contact ~ dasid, data = dascontacts,
                            paste, collapse = " , ")
  output <- dascitations %>% left_join(dascontacts2, by = "dasid") %>%
    left_join(dastheme2, by = "dasid") %>% left_join(daskeywords2,
                                                     by = "dasid")
  return(output)
}
```

Data selection and retrieval

The retrieval of data goes in two steps.

Step 1. Use functional group information to harvest potentially interesting datasets.

A query is performed for data on species known to be benthic (in WoRMS) and to occur in a number of different sea regions. The sea regions of interest are set in the file ./data/derived_data/regions.csv. The download yields a large dataset with benthic data. It is downloaded in chunks to minimize the risk of timeouts and other problems in downloading too big chunks of data at once, and recompiled afterwards into the very large (several gigabytes) file allDataExtra.csv.

```
# read geographic layers for plotting
layerurl <- paste0("http://geo.vliz.be/geoserver/MarineRegions/ows?",
                   "service=WFS&version=1.0.0&",
                   "request=GetFeature&typeName=MarineRegions:eez_iho_union_v2&",
                   "outputFormat=application/json")
regions <- sf::st_read(layerurl)

# read selected geographic layers for downloading
roi <- read_delim(file.path(dataDir,"regions.csv"), delim = ",")
```

```r
# check by plotting
regions %>% filter(mrgid %in% roi$mrgid) %>%
  ggplot() +
  geom_sf(fill = "blue", color = "white") +
  geom_sf_text(aes(label = mrgid), size = 2, color = "white") +
  theme(axis.title = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank())
ggsave(file.path(dataDir,"regionsOfInterest.png"), width = 3, height =  4, )

#== download data by geographic location and trait ===================
beginDate<- "1900-01-01"
endDate <- "2022-03-01"
attributeID1 <- "Benthos"
# Full occurrence (selected columns)
for(ii in 1:length(roi$mrgid)){
  mrgid <- roi$mrgid[ii]
  print(paste("downloading data for", roi$marregion[ii]))
  downloadURL <- paste0(
    "http://geo.vliz.be/geoserver/wfs/ows?service=WFS&version=1.1.0&",
    "request=GetFeature&",
    "typeName=Dataportal%3Aeurobis-obisenv_full&",
    "resultType=results&",
    "viewParams=where%3A%28%28up.geoobjectsids+%26%26+ARRAY%5B",mrgid,
    "%5D%29%29+AND+%28%28observationdate+BETWEEN+%27",beginDate,
    "%27+AND+%27",endDate,
    "%27+%29%29+AND+aphiaid+IN+%28+SELECT+aphiaid+FROM+",
    "eurobis.taxa_attributes+WHERE+selectid+IN+%28%27",
    attributeID1,
    "%27%29%29%3Bcontext%3A0100&",
    "propertyName=datasetid%2C",
    "datecollected%2C",
    "decimallatitude%2C",
    "decimallongitude%2C",
    "scientificname%2C",
    "aphiaid%2C",
    "scientificnameaccepted%2C",
    "yearcollected%2C",
    "waterbody%2C",
    "country%2C",
    "recordnumber%2C",
    "fieldnumber%2C",
    "minimumdepthinmeters%2C",
    "maximumdepthinmeters%2C",
    "aphiaidaccepted%2C",
    "catalognumber%2C",
    "qc%2C",
    "eventid&",
    "outputFormat=csv")


  filename = paste0("region", roi$mrgid[ii], ".csv")
  data <- read_csv(downloadURL,col_types = "cnccTnccccnnnncccccn")
  write_delim(data, file.path(downloadDir, "byTrait", filename), delim = ",")
}
filelist <- list.files(file.path(downloadDir,"byTrait"))
allDataExtra <- lapply(filelist, function(x)
  read_delim(file.path(downloadDir, "byTrait", x),
             delim = ",",
             col_types = "cnccTnccccnnnncccccn")) %>%
  set_names(sub(".csv", "", filelist)) %>%
  bind_rows(.id = "mrgid") %>%
  mutate(mrgid = sub("region", "", mrgid))
write_delim(allDataExtra, file.path(dataDir, "allDataExtra.csv"), delim = ",")
rm(layerurl,regions,roi,beginDate,endDate,attributeID1,mrgid,downloadURL,
   filename,data,allDataExtra)
gc()
```

Step 2. Download by dataset.

Many of the data from the first step come from datasets that are not useful for our purpose. As an example, planktonic datasets contain many benthic animals, because larvae of benthic animals occur in the zooplankton (the so-called meroplankton). We cannot use the plankton datasets to infer anything about absence of benthos on the sea floor.

The dataset resulting from the first action is used for a single purpose: identify all potentially interesting datasets, that contain at least one benthic animal in the region of interest. We subsequently use the imis database with meta-information on the datasets, to list the meta-data of all these datasets. This results in the file ./data/derived_data/allDatasets.csv.

In this file we perform the (manual) selection of datasets to be used. The steps used were: (1) only select datasets that had macrozoobenthos as their primary target. We also excluded datasets focusing on meiobenthos, hyperbenthos, epibenthos. (2) estimate from the dataset description if complete communities were targeted. We qualify the datasets as 'complete' if this is the case, and as 'incomplete' if not. The result of this manual procedure is the file ./data/derived_data/allDatasets_selection.csv. In comparison with the file allDatasets.csv, we have added two logical fields: "include" and "complete". The file is included into the github repository for inspection.

Subsequently, a file is produced listing per marine region which datasets need to be downloaded. This was necessary because the server produces an error if one tries to download a dataset for a region where that dataset has no data.

```r
### input
# allDataExtra
#
### output
#
# file allDatasets.csv : a list of all potentially interesting datasets
# file allDatasets_selection.csv: a manual edit of the previous file, where two
#                  fields are added: include (TRUE/FALSE): use the data
#                                    complete (TRUE/FALSE): dataset looks for all species
# file invent.csv: a list of regions and datasets to be used
#
allDataExtra <- read_delim(file.path(dataDir, "allDataExtra.csv"), delim = ",",
                           col_types = "ncnccTnccccnnnncccccc")

allDataExtra <- allDataExtra %>%
  mutate(datasetid = as.numeric(
    sub('http://www.emodnet-biology.eu/data-catalog?module=dataset&dasid=',
                                    "", datasetid, fixed = T)))
datasetidsoi <- allDataExtra %>% distinct(datasetid)

#==== retrieve data by dataset =use function fdr from auxiliary functions=========

  all_info <- data.frame()
  for (i in datasetidsoi$datasetid){
    dataset_info <- fdr2(i)
    all_info <- rbind(all_info, dataset_info)
  }
  names(all_info)[1]<-"datasetid"
  write.csv(all_info,file=file.path(dataDir,"allDatasets.csv"),row.names = F)
  # Note
  # this step is followed by manual inspection of data sets, and selection
  # results in file "./data/derived_data/allDatasets_selection.csv"

if(!file.exists(file.path(dataDir,"allDatasets_selection.csv"))){
  stop("perform manual selection of datasets in allDatasets_selection.csv")
}else{
  getDatasets <- read_csv(file.path(dataDir,"allDatasets_selection.csv"))
  getDatasets <- getDatasets %>% filter(include)

  allDataExtra <- allDataExtra %>%
    filter(datasetid %in% getDatasets$datasetid)
  invent <- allDataExtra %>% count(mrgid,datasetid)
  write_delim(invent,file.path(dataDir,"invent.csv"),delim=",")
  rm(allDataExtra,datasetidsoi,all_info,getDatasets,invent)
  gc()
}
```

With these preliminaries, we can proceed to downloading all relevant datasets in all regions, and fusing these into a single (very large) file.

```
## input
# file regions.csv
# file invent.csv
#
### output
# downoadfiles (many!)
# file all2Data.csv with the 'final' downloads joined
#

roi <- read_delim(file.path(dataDir,"regions.csv"), delim = ",")
invent<-read_delim(file.path(dataDir,"invent.csv"),delim=',')


for(ii in 1:nrow(invent)){
  di <- invent$datasetid[ii]
  mg <- invent$mrgid[ii]
  region <- roi$marregion[roi$mrgid==mg]
  print(paste("downloading data for ", region, "and dataset nr: ", di))
  downloadURL <-
    paste0("https://geo.vliz.be/geoserver/wfs/ows?service=WFS&version=1.1.0",
           "&request=GetFeature&typeName=Dataportal%3Aeurobis-obisenv_full&",
           "resultType=results&viewParams=where%3A%28%28up.geoobjectsids+",
           "%26%26+ARRAY%5B", mg,"%5D%29%29+AND+datasetid+IN+(",di,");",
           "context%3A0100&propertyName=datasetid%2C",
           "datecollected%2Cdecimallatitude%2Cdecimallongitude%2C",
           "coordinateuncertaintyinmeters%2C",
           "scientificname%2Caphiaid%2Cscientificnameaccepted%2C",
           "institutioncode%2Ccollectioncode%2C",
           "occurrenceid%2Cscientificnameauthorship%2Cscientificnameid%2C",
           "kingdom%2Cphylum%2Cclass",
           "%2Corder%2Cfamily%2Cgenus%2Csubgenus%2Caphiaidaccepted%2C",
           "basisofrecord%2Ceventid&",
           "outputFormat=csv")
  data <- read_csv(downloadURL, col_types = "ccccccTnnncccccccccccccccc")
  filename = paste0("region", mg, "_datasetid", di,  ".csv")
  write_delim(data, file.path(downloadDir, "byDataset", filename), delim = ",")
}

## Combine all downloaded datasets into one big dataset

filelist <- list.files(file.path(downloadDir,"byDataset"))
for(i in 1:length(filelist)){
  ininf <- read_delim(file.path(downloadDir,"byDataset", filelist[i]),
                      delim = ",",
                      col_types = "ccccccTnnncccccccccccccccc")%>%
    mutate(fileID=filelist[i])  %>%
    separate(fileID,c("mrgid","datasetID","_"))%>%
    mutate(mrgid = sub("[[:alpha:]]+", "", mrgid)) %>%
    mutate(datasetID = sub("[[:alpha:]]+", "", datasetID))%>%
    mutate(AphiaID=as.numeric(substr(aphiaidaccepted,52,65)))%>%
    filter(!is.na(AphiaID)) %>%
    filter(!is.na(decimallongitude)) %>%
    filter(!is.na(decimallatitude)) %>%
    filter(!is.na(datecollected)) %>%
    select(-aphiaid) %>%
    select(-aphiaidaccepted)%>%
    select(-`_`)%>%
    select(-datasetid) %>%
    select(-FID)%>%
    select(-scientificnameid)

  if(i==1)app<-FALSE else app<-TRUE
  write_delim(ininf,file.path(dataDir,"all2Data.csv"),append=app,delim=',')
  rm(roi,invent,all2Data,filelist,ininf,filename,data,downloadURL,di,mg,region,i)
  gc()
}
```

Analysis of the taxa represented in the dataset

The selection of data in the first step makes use of the traits stored in WoRMS, the World Register of Marine Species. For many species in this database, the functional group to which the species belongs is recorded. However, this is not yet complete. We can help the development of these traits databases from the compilation of data performed here. Since we selected benthic data sets, we can assume that most species in our database will be benthic, although it appears this is not absolutely the case everywhere. Here we try to use as much information as possible, either from the traits database or from the taxonomic position of the taxa, to derive what functional group they belong to. That is used to narrow down the list of taxa to the benthic species we are targeting, but also to report back to WoRMS with suggestions to improve the traits database.

The checks are illustrated in the following code, in two steps. First, the downloading of the trait information of over 15000 species is only done once. Results have been written to a file after performing this step, and as long as the file exists, it is read in.

```r
# Download attributes of all taxa in the file all2Data.csv
# input: all2Data.csv
#
all2Data<-read_delim(file.path(dataDir, "all2Data.csv"), delim = ",",
                     col_types = "ncccTnnncccccccccccccnnn")

nsp_attr<-tibble()
splst <- all2Data %>%
  select(AphiaID,scientificnameaccepted,phylum,class,order,family,genus,
         subgenus) %>%
  distinct() %>%
  mutate(benthos=FALSE,endobenthos=FALSE,macrobenthos=FALSE,epibenthos=FALSE,
         meiobenthos=FALSE,phytobenthos=FALSE,
         plankton=FALSE,nekton=FALSE,Pisces=FALSE,Algae=FALSE,
         Aves_tax=FALSE,Pisces_tax=FALSE,Algae_tax=FALSE,Plants_tax=FALSE,
         meio_tax=FALSE,micro_tax=FALSE,misc_tax=FALSE)

for(i in 1:nrow(splst)){
  print(paste(i,"out of",nrow(splst),"downloading attributes of species",
              splst$scientificnameaccepted[i],"AphiaID",splst$AphiaID[i]))
  ttt<-NULL
  try(ttt<-wm_attr_data(id=splst$AphiaID[i],include_inherited = T),silent = T)
  if(! is.null(ttt)) nsp_attr<-rbind(nsp_attr,ttt[,1:9])
}
nsp_attr <- nsp_attr %>%
  mutate(AphiaID=as.numeric(AphiaID)) %>%
  left_join(splst,by="AphiaID")
write_delim(nsp_attr,file.path(dataDir,"nsp_attr.csv"),delim=",")

# clean up
rm(all2Data,nsp_attr,ttt,splst)
gc()
```

Once the basic attributes are retrieved (a very time-consuming step), the species to be used in the analysis are selected. This selection is documented in the following code. We make use as much as possible of stored traits information, to eliminate species known to be something else than macrozoobenthos (e.g. meiobenthos, hyperbenthos, plants, birds, fish etc.). Then we use taxonomic information to eliminate taxa in these groups as much as possible. All Nematodes, as an example, are classified as meiobenthos even if WoRMS does not do it. After the selection, we produce lists of doubtful classifications as input to WoRMS. We also produce the file sp2use.csv, with the names and AphiaIDs of species to be used in subsequent analysis.

```r
## Analysis of the species represented in the dataset
#
# input: all2Data, nsp_attr
# output: specieslist files for Worms editors, sp2use.csv

all2Data<-read_delim(file.path(dataDir, "all2Data.csv"), delim = ",",
                      col_types = "ncccTnnnccccccccccccnnn")
nsp_attr <- read_delim(file.path(dataDir,"nsp_attr.csv"),delim=",",
                        col_types="nnccnccnncccccccllllllllllllllll")

# we build the species list, keeping the taxonomic information we have in the
# total data set
# we foresee logical columns in the species list to group the species by in the
# rest of this script
splst <- all2Data%>%
  select(AphiaID,scientificnameaccepted,phylum,class,order,family,genus,
         subgenus) %>%
  distinct() %>%
  mutate(benthos=FALSE,endobenthos=FALSE,macrobenthos=FALSE,epibenthos=FALSE,
         meiobenthos=FALSE,phytobenthos=FALSE,
         plankton=FALSE,nekton=FALSE,Pisces=FALSE,Algae=FALSE,
         Aves_tax=FALSE,Pisces_tax=FALSE,Algae_tax=FALSE,Plants_tax=FALSE,
         meio_tax=FALSE,micro_tax=FALSE,misc_tax=FALSE)
# ###### determine, using attributes, which species are benthos #######
# ###### again, several hours download ##########
# (done once, result stored as delimited file)
# what Functional groups are there?
fg <- nsp_attr %>% filter(measurementType=="Functional group") %>%
  select(measurementValue) %>%
  distinct
print(fg)
# what Paraphyletic groups are there?
pfg <- nsp_attr %>% filter(measurementType=="Paraphyletic group") %>%
  select(measurementValue) %>%
  distinct
print(pfg)
# fill in attributes columns of splst based on attributes downloaded from WoRMS
set_attr<-function(attr){
  tt <- nsp_attr                               %>%
    filter(grepl(attr,measurementValue)) %>%
    select(AphiaID)                      %>%
    distinct()
  splst <- splst %>%
    mutate(!!attr:=ifelse(AphiaID %in% tt$AphiaID,TRUE,FALSE))
  return(splst)
}
splst<-set_attr("benthos")
splst<-set_attr("endobenthos")
splst<-set_attr("macrobenthos")
splst<-set_attr("epibenthos")
splst<-set_attr("meiobenthos")
splst<-set_attr("phytobenthos")
splst<-set_attr("Pisces")
splst<-set_attr("Algae")
splst<-set_attr("plankton")
splst<-set_attr("nekton")
# fill in attributes columns based on taxonomic information
splst <- splst %>%
  mutate(Pisces_tax=FALSE) %>%
  mutate(Aves_tax=FALSE) %>%
  mutate(Algae_tax=FALSE) %>%
  mutate(Plants_tax=FALSE) %>%
  mutate(micro_tax=FALSE) %>%
  mutate(meio_tax=FALSE) %>%
  mutate(misc_tax=FALSE)

splst$Pisces_tax <- splst$Pisces_tax | splst$class  == "Actinopterygii"
splst$Pisces_tax <- splst$Pisces_tax | splst$class  == "Elasmobranchii"
splst$Aves_tax   <- splst$Aves_tax   | splst$class  == "Aves"
splst$Algae_tax  <- splst$Algae_tax  | splst$phylum == "Chlorophyta"
splst$Algae_tax  <- splst$Algae_tax  | splst$phylum == "Rhodophyta"
splst$Algae_tax  <- splst$Algae_tax  | splst$phylum == "Ochrophyta"
splst$Algae_tax  <- splst$Algae_tax  | splst$phylum == "Charophyta"
splst$Algae_tax  <- splst$Algae_tax  | splst$phylum == "Cyanobacteria"
splst$Algae_tax  <- splst$Algae_tax  | splst$phylum == "Haptophyta"
splst$Plants_tax <- splst$Plants_tax | splst$Algae_tax
splst$Plants_tax <- splst$Plants_tax | splst$phylum == "Tracheophyta"
splst$Plants_tax <- splst$Plants_tax | splst$phylum == "Bryophyta"
splst$micro_tax  <- splst$micro_tax | splst$phylum == "Ascomycota"
splst$micro_tax  <- splst$micro_tax | splst$phylum == "Proteobacteria"
splst$meio_tax   <- splst$meio_tax  | splst$phylum == "Nematoda"
splst$meio_tax   <- splst$meio_tax   | splst$phylum == "Foraminifera"
splst$meio_tax   <- splst$meio_tax   | splst$phylum == "Tardigrada"
splst$meio_tax   <- splst$meio_tax   | splst$phylum == "Gastrotricha"
splst$meio_tax   <- splst$meio_tax   | splst$phylum == "Kinorhyncha"
splst$meio_tax   <- splst$meio_tax   | splst$phylum == "Ciliophora"
splst$meio_tax   <- splst$meio_tax   | splst$class  == "Ostracoda"
splst$meio_tax   <- splst$meio_tax   | splst$order  == "Harpacticoida"
splst$misc_tax   <- splst$misc_tax   | splst$class  == "Arachnida"
splst$misc_tax   <- splst$misc_tax   | splst$class  == "Mammalia"
splst$misc_tax   <- splst$misc_tax   | splst$class  == "Insecta"
splst$misc_tax   <- splst$misc_tax   | splst$class  == "Ichthyostraca"
splst$misc_tax   <- splst$misc_tax   | splst$class  == "Diplopoda"
splst$misc_tax   <- splst$misc_tax   | splst$class  == "Collembola"
splst$misc_tax   <- splst$misc_tax   | splst$class  == "Chilopoda"
splst$misc_tax   <- splst$misc_tax   | splst$class  == "Clitellata"

splst <- splst %>%
  mutate(Pisces_tax=ifelse(is.na(Pisces_tax),FALSE,Pisces_tax)) %>%
  mutate(Aves_tax=ifelse(is.na(Aves_tax),FALSE,Aves_tax)) %>%
  mutate(Algae_tax=ifelse(is.na(Algae_tax),FALSE,Algae_tax)) %>%
  mutate(Plants_tax=ifelse(is.na(Plants_tax),FALSE,Plants_tax)) %>%
  mutate(micro_tax=ifelse(is.na(micro_tax),FALSE,micro_tax)) %>%
  mutate(meio_tax=ifelse(is.na(meio_tax),FALSE,meio_tax)) %>%
  mutate(misc_tax=ifelse(is.na(misc_tax),FALSE,misc_tax))

# write splst to output
write_delim(splst,file.path(dataDir,"splst.csv"),delim=",")
#
#
# lists to be produced for WoRMS people
# list of fish species that do not have Paraphyletic group == Pisces
```

```
prob1 <- splst %>% filter (Pisces_tax & !Pisces)
write_delim(prob1,file.path(dataDir,"specieslist1.csv"),delim=",")
# list of algae species that do not have Paraphyletic group == Algae
prob2 <- splst %>% filter (Algae_tax & !Algae)
write_delim(prob2,file.path(dataDir,"specieslist2.csv"),delim=",")
# list of species that should have a paraphyletic group 'plants' or something
prob3 <- splst %>% filter (Plants_tax)
write_delim(prob3,file.path(dataDir,"specieslist3.csv"),delim=",")
# list of species that are likely meiobenthos (based on taxonomy) but no
# attribute meiobenthos
prob4 <- splst %>% filter (meio_tax & !meiobenthos)
write_delim(prob4,file.path(dataDir,"specieslist4.csv"),delim=",")
# list of bird species that maybe should get a Paraphyletic group 'Aves'
prob5 <- splst %>% filter (Aves_tax)
write_delim(prob5,file.path(dataDir,"specieslist5.csv"),delim=",")
# list of species that are classified as 'nekton' but are sometimes considered
# benthic
prob6 <- splst %>% filter (nekton)
write_delim(prob6,file.path(dataDir,"specieslist6.csv"),delim=",")
# list of species of odd taxa that do not really belong in benthos studies
prob7 <- splst %>% filter (misc_tax & !benthos)
write_delim(prob7,file.path(dataDir,"specieslist7.csv"),delim=",")
# list of species found in benthic datasets, but that are not benthos, not fish,
# not birds, not plants, not micro-organisms, not meiofauna, not plankton and
# not nekton
prob8 <- splst %>% filter (!benthos&!Pisces&!Pisces_tax&!Aves_tax&!Plants_tax&
                             !Algae&!micro_tax&!meio_tax&!meiobenthos&!plankton&
                             !nekton) %>%
  arrange(phylum,class,order,family,genus,subgenus,scientificnameaccepted)
write_delim(prob8,file.path(dataDir,"specieslist8.csv"),delim=",")
####### So, what species to use for the maps? ########################
# species should be:
# * not meiobenthos or meio_tax
# * not phytobenthos
# * not Pisces or Pisces_tax
# * not Plants_tax (which includes Algae_tax)
# * not Algae
# * not micro_tax
# * not Aves_tax
# * not misc_tax
# * not plankton (if they are not either benthos or nekton too)
sp2use <- splst %>%
  filter (!meiobenthos & !meio_tax & !phytobenthos & !Pisces & !Pisces_tax &
            !Plants_tax & !Algae & !micro_tax & ! Aves_tax &
            !(plankton & !(benthos|nekton)) &
            !(misc_tax & !benthos))
write_delim(sp2use,file.path(dataDir,"sp2use.csv"),delim=",")

# clean up
rm(all2Data,nsp_attr,splst,fg,pfg,set_attr,prob1,prob2,prob3,prob4,prob5,prob6,
   prob7,prob8,sp2use)
gc()
```

Derivation of presence/absence information

For the derivation of presence/absence information, we define ‘sampling events’ as the ensembles of records that share time and place. We consider such events as one sample. For the incomplete datasets, we inventory what species they have targeted. Finally, for every species we determine whether or not it was present in all sampling events of all relevant datasets. This presence/absence information is written to the output file, together with the spatial location and the sampling date. The output file is a CF compliant netcdf file, that is prepared using a function that writes all dimensions, variables and attributes to the file, but not yet the actual presence-absence data. That information is calculated in a loop over species, en for each species it is added to the netcdf file right after the calculation. This procedure limits the size of files to be kept in memory.

The function used to prepare the netcdf file is documented in this code chunk. Data are stored in the so-called ‘point’ format of CF conventions. In this case, presence (1), absence (0) or no observation (NA) of a taxon in a specific sample is stored in a large sample * taxa two-dimensional matrix. The samples are further specified by their latitute, longitude and date of sampling, which all three are one-dimensional arrays with the number of samples as length. Taxa are specified by the Aphia ID in Worms and by the scientific name of the taxon, which both have the dimension number of taxa. Further information is stored on the crs. This is an empty variable that is added because of its attributes, which store the real information on the coordinate stystem used. In this case geographic coordinates and the WGS84 ellipsoid have been used. Finally, a large number of metadata general attributes are added to the file.

```r
create_netcdf_output <- function(events,spfr,netcdf_fil){
  nobs <- nrow(events)
  nsp <- nrow(spfr)

  # transform the dates to days since 1970-1-1
  events$eventDate<-as.numeric(as.Date(events$eventDate))

  # make vectors with AphiaIds and Specnames
  AphiaIDs <- paste0("urn:lsid:marinespecies.org:taxname:",spfr$aphiaID)
  spnams <- as.vector(spfr$scientificName)[1:nsp]


  ## ADD DIMENSIONS
  points_dim <- ncdim_def("points", "", 1:nobs, unlim=FALSE, create_dimvar=FALSE)
  tax_dim <- ncdim_def("taxon","",1:nsp,unlim=FALSE,create_dimvar=FALSE)
  nchar_dim <- ncdim_def("string45", "", 1:45, create_dimvar=FALSE )

  ## ADD VARIABLES
  time_var <- ncvar_def(name="Date",
                      units="days since 1970-01-01 00:00:00",
                      dim=list(points_dim),
                      missval = as.integer(-99999),
                      longname="time",
                      prec='integer')
  lat_var <- ncvar_def(name="lat",
                     units="degree_north",
                     dim=list(points_dim),
                     missval=NA,
                     longname="latitude",
                     prec='float')
  lon_var <- ncvar_def(name="lon",
                     units="degree_east",
                     dim=list(points_dim),
                     missval=NA,
                     longname="longitude",
                     prec='float')
AphiaID_var <- ncvar_def(name = "AphiaID",
                         units = "",
                         dim = list(nchar_dim,tax_dim),
                         longname="biological_taxon_lsid",
                         prec='char')
TaxName_var <- ncvar_def(name="Taxon_Name",
                         units = "",
                         dim=list(nchar_dim,tax_dim),
                         longname="biological_taxon_name",
                         prec='char')
PresAbs_var <- ncvar_def (name="Pres_abs",units = "", dim=list(points_dim,tax_dim),
                          missval = 2,
                          longname="Presence(1)Absence(0)of species at sampling event",
                          prec='byte')
CRS_var <- ncvar_def(name="crs",
                     dim=list(),
                     units="",
                     longname="coordinate reference system")

## CREATE A NEW FILE
ncnew <- nc_create(netcdf_fil, list(time_var, lat_var, lon_var,CRS_var,
                                    AphiaID_var, TaxName_var),
                   force_v4=TRUE)

## ADD GLOBAL ATTRIBUTES
# see http://www.unidata.ucar.edu/software/thredds/current/netcdf-java/
#            formats/DataDiscoveryAttConvention.html
ncatt_put(ncnew, 0, "ncei_template_version", "NCEI_NetCDF_Point_Template_v2.0")
ncatt_put(ncnew, 0, "featureType", "point")
ncatt_put(ncnew, 0, "title", "Presence/absence of macrobenthos in European Seas")
ncatt_put(ncnew, 0, "summary",
          paste0("This dataset compiles all available macrozoobenthos data in Emodnet ",
                 "biology. Presence is recorded when a species is in the database. ",
                 "Absence is recorded when the species was",
                 " not found in a sample that was (in principle) looking for the species.",
                 " NA is given when the species was not looked for at the sampling event. ",
                 "The data are recorded in an event*species matrix.",
                 "  Every event is characterised by x,y,time. z is always the sea bottom. ",
                 "Taxa (i.e. species or higher taxa such as genus, family etc.) are",
                 "characterised by their AphiaID, a sequential number in the World ",
                 "Record of Marine Species ",
                 " (WoRMS - https://marinespecies.org) and by their taxon name"))
ncatt_put(ncnew, 0, "keywords", "Waiting For Keywords")
ncatt_put(ncnew, 0, "Conventions", "CF-1.8, ACDD-1.3")
ncatt_put(ncnew, 0, "id", "Presence_absence_macrobenthos_European_Seas")
ncatt_put(ncnew, 0, "naming_authority", "deltares.nl")
ncatt_put(ncnew, 0, "history",
          paste0("This product is an extension of a previously published product",
                 "doing this analysis on the Greater North Sea. See ",
                 "https://www.emodnet-biology.eu/blog/summary-presenceabsence-maps",
                    "-macro-endobenthos-greater-north-sea. The present product has",
                    "used the same methodology with some adjustments, and covers all",
                    "European Seas. It is also more recent, so has some additional",
                    "data for the North Sea")
)
ncatt_put(ncnew, 0, "source",
          paste0("See the Emodnet biology portal for all data sources available. ",
                 "The workflow of this data product is described on the Emodnet github ",
                 "site (subsite Benthos_European_Seas)",
                 "Still waiting for links to be updated"))
ncatt_put(ncnew, 0, "processing_level",
          "Interoperable data collation - minimally interpreted")
ncatt_put(ncnew, 0, "comment", "")
ncatt_put(ncnew, 0, "acknowledgment",
          paste0("Data originators are acknowledged for the data. Full list of ",
                 "datasets used, including reference to responsible people, can",
                 "be found in the underlying files on github - see file",
                 "./data/derived_data/allDatasets_selection.csv.",
                 "European Marine Observation Data Network (EMODnet) Biology project",
                 "(EASME/EMFF/2017/1.3.1.2/02/SI2.789013), funded by the European Union",
                 "under Regulation (EU) No 508/2014 of the European Parliament and of",
                 "the Council of 15 May 2014 on the European Maritime and Fisheries Fund")
)
ncatt_put(ncnew, 0, "license", "CC-BY")
ncatt_put(ncnew, 0, "standard_name_vocabulary", "CF Standard Name Table vNN")
ncatt_put(ncnew, 0, "date_created", "2022-03-04 13:21:00")
ncatt_put(ncnew, 0, "creator_name", "Peter Herman")
ncatt_put(ncnew, 0, "creator_email", "peter.herman@deltares.nl")
ncatt_put(ncnew, 0, "creator_url", "www.deltares.nl")
ncatt_put(ncnew, 0, "institution", "Deltares")
ncatt_put(ncnew, 0, "project", "Emodnet_Biology")
ncatt_put(ncnew, 0, "publisher_name", "Emodnet Biology Project")
ncatt_put(ncnew, 0, "publisher_email", "bio@emodnet.eu")
ncatt_put(ncnew, 0, "publisher_url", "Wait for central portal URL")
ncatt_put(ncnew, 0, "geospatial_lat_min", "26")
ncatt_put(ncnew, 0, "geospatial_lat_max", "82")
ncatt_put(ncnew, 0, "geospatial_lon_min", "-36")
ncatt_put(ncnew, 0, "geospatial_lon_max", "61")
ncatt_put(ncnew, 0, "time_coverage_start", "-70492")
ncatt_put(ncnew, 0, "time_coverage_end", "18992")
ncatt_put(ncnew, 0, "time_coverage_duration", "89484")
ncatt_put(ncnew, 0, "time_coverage_resolution", "day")
ncatt_put(ncnew, 0, "time_coverage_units", "days since 1970-01-01 00:00:00")
ncatt_put(ncnew, 0, "sea_name", paste0("Arctic Sea, Baltic Sea, Black Sea, ",
                                       "Mediterranean, North Atlantic Ocean, North Sea")
)
ncatt_put(ncnew, 0, "publisher_institution", "VLIZ")
ncatt_put(ncnew, 0, "geospatial_lat_units", "degrees_north")
ncatt_put(ncnew, 0, "geospatial_lon_units", "degrees_east")
ncatt_put(ncnew, 0, "date_modified", "2022-03-04 13:21:00")
ncatt_put(ncnew, 0, "date_issued", "2022-03-04 13:21:00")
ncatt_put(ncnew, 0, "date_metadata_modified", "2022-03-04 14:21:00")
ncatt_put(ncnew, 0, "product_version", "0.3")
```

```
  ncatt_put(ncnew, 0, "cdm_data_type", "Point")
  ncatt_put(ncnew, 0, "metadata_link", "Wait for link")
  ncatt_put(ncnew, 0, "references", "Wait for link")
  #
  # add attributes of crs
  #
  ncatt_put(ncnew,CRS_var,'geographic_crs_name','WGS 84')
  ncatt_put(ncnew,CRS_var,'grid_mapping_name','latitude_longitude')
  ncatt_put(ncnew,CRS_var,'inverse_flattening', '298.257223563')
  ncatt_put(ncnew,CRS_var,'longitude_of_prime_meridian', '0.0')
  ncatt_put(ncnew,CRS_var,'prime_meridian_name','Greenwich')
  ncatt_put(ncnew,CRS_var,'reference_ellipsoid_name','WGS 84')
  ncatt_put(ncnew,CRS_var,'semi_major_axis','6378137.0')
  ncatt_put(ncnew,CRS_var,'semi_minor_axis','6356752.314245179')

  # write values to the file
  # list(time_var, lat_var, lon_var, z_var, AphiaID, TaxName, PresAbs, crs_var)
  ncvar_put(nc=ncnew,varid=time_var,vals=events$eventDate)
  ncvar_put(nc=ncnew,varid=lat_var,vals=events$decimalLatitude)
  ncvar_put(nc=ncnew,varid=lon_var,vals=events$decimalLongitude)
  ncvar_put(nc=ncnew,varid=AphiaID_var,vals=as.vector(AphiaIDs))
  ncvar_put(nc=ncnew,varid=TaxName_var,vals=spnams)

  ncvar_add(nc=ncnew,PresAbs_var,indefine=TRUE)

  nc_close(ncnew)
}
```

The code used to calculate presence/absence information and store it in the netcdf output file is documented here.

```
# needed files #
# all2Data.csv
# sp2use.csv
# allDatasets_selection.csv

#########################
# output file: netcdf file with samples, species and presence/absence information
#
#### read input #############
trec<-read_delim(file.path(dataDir, "all2Data.csv"), delim = ",",
                     col_types = "ncccTnnncccccccccccccnnn")
sp2use<-read_delim(file.path(dataDir,"sp2use.csv"),delim=",",
                   col_types = "dcccccccllllllllllllllll")
trdi<-read_delim(file.path(dataDir,"allDatasets_selection.csv"),delim=',',
                 col_types="nccccccccllcc")

##########################################################
##### select few columns to work with
##### and filter to true benthic species only
##########################################################
trec<- trec %>%
  dplyr::select(eventDate=datecollected,
                decimalLongitude=decimallongitude,
```

```r
                decimalLatitude=decimallatitude,
                scientificName=scientificnameaccepted,
                aphiaID=AphiaID,
                datasetid=datasetID) %>%
  filter(aphiaID %in% sp2use$AphiaID)
##############################################################
# Define 'sampling events' as all records that share time and place, give
# ID numbers to all events (eventNummer), and store the eventNummer in each
# record of trec
##############################################################
events<- trec %>% dplyr::select(eventDate,decimalLongitude,
                                decimalLatitude,datasetid) %>%
  distinct() %>%
  mutate(eventNummer=row_number())

trec <- trec %>%
  left_join(events,by=c('eventDate','decimalLongitude',
                        'decimalLatitude','datasetid')) %>%
  distinct(eventDate,decimalLongitude,decimalLatitude,
           scientificName,aphiaID,datasetid,eventNummer)


########### work on datasets
#
#### check on completeness
#
# nsp<-trec %>% group_by(datasetid) %>%
#   distinct(aphiaID)    %>%
#   mutate(nspec=n())    %>%
#   dplyr::select(datasetid,nspec) %>%
#   distinct()
# nev<-trec %>% group_by(datasetid)     %>%
#   distinct(eventNummer)    %>%
#   mutate(nev=n())          %>%
#   dplyr::select(datasetid,nev) %>%
#   distinct()                    %>%
#   left_join(nsp,by='datasetid')%>%
#   left_join(trdi,by='datasetid')
# #
#plot(nev$nev,nev$nspec,log="xy",col=ifelse(nev$complete,"blue","red"),pch=19,
#     xlab="number of events in dataset",ylab="number of species in dataset")
#text(nev$nev*1.2,nev$nspec*(1+(runif(nrow(nev))-0.5)*0.4),nev$datasetid,cex=0.5)
##############################################################
# find occurrence frequency of all species, and rank the species accordingly
#
spfr<- trec %>%
  group_by(aphiaID,scientificName) %>%
  summarize(n_events=n()) %>%
  arrange(desc(n_events))
nsptoplot<-length(which(spfr$n_events>0))
############
#
# manage the incomplete datasets
```

```r
#
trdi_ct<-trdi %>% filter (complete)
trdi_ic<-trdi %>% filter (!complete)
# make a species list for each incomplete dataset
ic_sp<-data.frame(datasetid=NULL,aphiaID=NULL)
for(i in 1:nrow(trdi_ic)){
  ds<-trdi_ic$datasetid[i]
  specs<-unique(trec$aphiaID[trec$datasetid==ds])
  ic_sp<-rbind(ic_sp,data.frame(datasetid=rep(ds,length(specs)),aphiaID=specs))
}
# and add the complete datasets
for(i in 1:nrow(trdi_ct)){
  ds<-trdi_ct$datasetid[i]
  specs<-spfr$aphiaID
  ic_sp<-rbind(ic_sp,data.frame(datasetid=rep(ds,length(specs)),aphiaID=specs))
}

# write events and species list
write_delim(events,file=file.path(mapsDir,"events.csv"),delim=",")
write_delim(spfr,file=file.path(mapsDir,"spfr.csv"),delim=",")

#create output netcdf file if it does not yet exist
# This is a precaution against overwriting a file that takes hours to build
netcdf_fil <- file.path(mapsDir,"Macrobenthos_Eur_Seas_Pres_Abs_v0-3.nc")
if(file.exists(netcdf_fil)){
  stop("netcdf file already exists. Delete it if you want to create new one")
}else{
  source("./scripts/create_netcdf_output.R")
  create_netcdf_output(events,spfr,netcdf_fil)
}

# now produce the output and write it into the netcdf file
ncout<-nc_open(netcdf_fil,write=TRUE)

spmin<-1
spmax<-nsptoplot
#nams <- as.vector(spfr$scientificName)
pb <- progress_bar$new(total=nsptoplot)

for(ss in spmin:spmax){
    spAphId<-spfr$aphiaID[ss]
    pb$tick()
    # from the list of datasets, check if they have our species.
    # Only keep these, drop the others
    tt_ds<- ic_sp                      %>%
      filter(aphiaID==spAphId) %>%
      distinct(datasetid)      #      %>%
    # The datasets to be used consist of all complete datasets, and all
    # incomplete datasets that targeted our species
    spe<- trec                                                         %>%
      filter(datasetid %in% tt_ds$datasetid)                       %>%
      group_by(eventNummer)                                        %>%
      summarize(pres_abs= any(aphiaID==spAphId),.groups = 'drop')  %>%
      full_join(events,by='eventNummer') %>%
      arrange(eventNummer)
    ncvar_put(ncout,"Pres_abs",as.numeric(spe$pres_abs),
              start=c(1,ss),count=c(-1,1))
}
nc_close(ncout)
```

Making rasters and maps per species

The intention is to use this information to produce interpolated maps covering also the non-sampled space. As a first step in visualisation, we can rasterize this information and show it in a map per species. The following code block documents a function that calculates the raster and produces a map for a species, based on the sequence number of the species in the list of species. Note that this list of species is ordered (in descending order) by the number of occurrences of the species in the total data set.

We use the EMODnet mapping package EMODnetBiologyMaps to produce maps of the rasters.

```r
plotraster <- function(specnr,netcdf_fil){
  if(!file.exists(netcdf_fil)){
    stop("first make the netcdf output file")
  }
  # define a raster covering the grid. Set resolution of the raster here
  r<-raster(ext=extent(-36,61,26,82),ncol=480,nrow=380,crs=proWG,vals=0)
  # read input from netcdf file
  ncfil<-nc_open(netcdf_fil)
  lats<-ncvar_get(ncfil,"lat")
  lons<-ncvar_get(ncfil,"lon")
  spec_names <- ncvar_get(ncfil,"Taxon_Name")
  aphiaids <- ncvar_get(ncfil,"AphiaID")
  presabs<-ncvar_get(ncfil,"Pres_abs",start=c(1,specnr),count=c(-1,1))
  nc_close(ncfil)

  aphiaids <- sub("urn:lsid:marinespecies.org:taxname:","",aphiaids)
  spe<-data.frame(lat=lats,lon=lons,pres_abs=NA)
  coordinates(spe)<- ~lon+lat
  projection(spe)<-proWG

  spe_sp<-spe
  spe_sp$pres_abs<-presabs
  spe_spe<-spe_sp[!is.na(spe_sp$pres_abs),]
  r1<-rasterize(spe_sp,r,field="pres_abs",fun=mean)

  legend="P(pres)"
  specname <- spec_names[specnr]
  spAphId <- aphiaids[specnr]
  ec<-emodnet_colors()
  plot_grid <- emodnet_map_plot_2(data=r1,title=specname,
                                  subtitle=paste0('AphiaID ', spAphId),
                                  zoom=TRUE,seaColor=ec$darkgrey,
                                  landColor=ec$lightgrey,legend=legend)
  print(plot_grid)
  filnam<-file.path(plotsDir,
                    paste0(sprintf("%04d",specnr), "_",spAphId, "_",
                           gsub(" ", "-", specname),".png"))
  emodnet_map_logo(plot_grid,path=filnam,width=120,height=160,dpi=300,
                   units="mm",offset="+0+0")
}
```