Benthos_greater_North_Sea

Peter M.J. Herman, Willem Stolte, Luuk van der Heijden

20-6-2020

Introduction

The large databases of EMODNET Biology only store confirmed presences of species. However, when mapping species distribution, it is also important where the species did not occur: there is at least as much information in absences as in presences. Inferring absences from presence-only databases is difficult and always involves some guesswork. In this product we have used as much meta-information as possible to guide us in inferring absences. There is important meta-information at two different levels: the level of the data set, and the level of the species. Datasets can contain implicit information on absences when they have uniformly searched for the same species over a number of sample locations. Normally, if the species would have been present there, it would have been recorded. Other datasets, however, are not informative at all about absences. Typical examples are museum collections. The fact that a specimen is found at a particular place confirms that it lived there, but does not give information on any other species being present or absent in the same spot. A difficulty is that some datasets have searched for a restricted part of the total community, e.g. only sampled shellfish but no worms. In this case, absence of a shellfish species is relevant, but absence of a worm is not. The dataset can only be used to infer absence for the species it has targeted. Here we implicitly assume that a dataset inventoring the endomacrobenthos, is targeting all species belonging to this functional group. Usually, the distinction can be made on the basis of the metadata. It is also helpful to plot the total number of species versus the total number of samples. Incomplete datasets have far less species than expected for their size, compared to 'complete' datasets. At the species level, taxonomic databases such as WoRMS give information on the functional group the species belongs to. This information is present for many species, but it is incomplete. However, we can use it in a step to select the most useful datasets. We could use it to restrict the species to be used in mapping, but since the data are incomplete that would cause some loss of interesting information.

Data selection and retrieval

The retrieval of data goes in three 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. This yields a large dataset with benthic data, but many of these data 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 in the sea floor.

From the dataset resulting from this action we harvest 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. We also qualify the datasets as either 'complete' or 'incomplete'. The result of this manual procedure is the file ./data/derived_data/allDatasets_selection.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")
```

```r
regions <- sf::st_read(layerurl)

# read selected geographic layers for downloading
roi <- read_delim("data/derived_data/regions.csv", delim = ",")

# check by plotting
regions %>% filter(mrgid %in% roi$mrgid) %>%
  ggplot() +
  geom_sf(fill = "blue", color = "white") +
  geom_sf_text(aes(label = mrgid), size = 2, color = "white") +
  theme(axis.title = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank())
ggsave("data/derived_data/regionsOfInterest.png", width = 3, height =  4, )
#== download data by geographic location and trait ====================================
beginDate<- "1900-01-01"
endDate <- "2020-05-31"
attributeID1 <- "benthos"
attributeID2 <- NULL
attributeID3 <- NULL
# Full occurrence (selected columns)
for(ii in 1:length(roi$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%5C%2C%27",attributeID2,
     "%27%29%29%3Bcontext%3A0100&propertyName=datasetid%2Cdatecollected%2Cdecimallatitude",
     "%2Cdecimallongitude%2Ccoordinateuncertaintyinmeters%2Cscientificname%2Caphiaid%2C",
     "scientificnameaccepted%2Cinstitutioncode%2Ccollectioncode%2Coccurrenceid%2C",
     "scientificnameauthorship%2Cscientificnameid%2Ckingdom%2Cphylum%2Cclass%2Corder%2C",
     "family%2Cgenus%2Csubgenus%2Caphiaidaccepted%2Cbasisofrecord%2Ceventid&outputFormat=csv")
  filename = paste0("region", roi$mrgid[ii], ".csv")
  data <- read_csv(downloadURL)
  write_delim(data, file.path(downloadDir, "byTrait", filename), delim = ";")
}
filelist <- list.files("data/raw_data/byTrait")
allDataExtra <- lapply(filelist, function(x)
  read_delim(file.path("data", "raw_data/byTrait", x),
             delim = ";",
             col_types = "ccccccTnnlccccccccccccccc")) %>%
  set_names(sub(".csv", "", filelist)) %>%
  bind_rows(.id = "mrgid") %>%
  mutate(mrgid = sub("region", "", mrgid))
#write_delim(allDataExtra, file.path(dataDir, "allDataExtra.csv"), delim = ";")

#=== from downloaded data ============================
#
#allDataExtra <- read_delim(file.path(dataDir, "allDataExtra.csv"), delim = ";")
datasetidsoi <- allDataExtra %>% distinct(datasetid) %>%
```

```
  mutate(datasetid = sub('http://www.emodnet-biology.eu/data-catalog?module=dataset&dasid=',
                         "", datasetid, fixed = T))
#==== retrieve data by dataset ==============
#
source("read_dasid_features.R")
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="./data/derived_data/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"
```

Step 2. Download by dataset.

In this step, we download the part of all these useful datasets that occur in the region of interest. For practical reasons this region is subdivided in many subregions - in that way the downloaded files are not too big and there is less risk of interruptions of the process. After download, all these files will be recombined into one big datafile.

```
# content of script requestData_step2.R

# read selected geographic layers for downloading
roi <- read_delim("data/derived_data/regions.csv", delim = ",")
getDatasets <- read_csv("./data/derived_data/allDatasets_selection.csv")
getDatasets <- getDatasets %>% filter(include)
for(ii in 1:length(roi$mrgid)){
  for(jj in 1:length(getDatasets$datasetid)){
    datasetid <- getDatasets$datasetid[jj]
    mrgid <- roi$mrgid[ii]
    print(paste("downloading data for ", roi$marregion[ii], "and dataset nr: ", datasetid))
    downloadURL <- paste0("https://geo.vliz.be/geoserver/wfs/ows?service=WFS&version=1.1.0",
      "&request=GetFeature&typeName=Dataportal%3Aeurobis-obisenv_full&resultType=results&",
      "viewParams=where%3A%28%28up.geoobjectsids+%26%26+ARRAY%5B", mrgid,
      "%5D%29%29+AND+datasetid+IN+(",datasetid,");context%3A0100&propertyName=datasetid%2C",
      "datecollected%2Cdecimallatitude%2Cdecimallongitude%2Ccoordinateuncertaintyinmeters%2C",
      "scientificname%2Caphiaid%2Cscientificnameaccepted%2Cinstitutioncode%2Ccollectioncode%2C",
      "occurrenceid%2Cscientificnameauthorship%2Cscientificnameid%2Ckingdom%2Cphylum%2Cclass",
      "%2Corder%2Cfamily%2Cgenus%2Csubgenus%2Caphiaidaccepted%2Cbasisofrecord%2Ceventid&",
      "outputFormat=csv")
    data <- read_csv(downloadURL, col_types = "ccccccTnnnccccccccccccccc")
    filename = paste0("region", roi$mrgid[ii], "_datasetid", datasetid,  ".csv")
    if(nrow(data) != 0){
      write_delim(data, file.path(downloadDir, "byDataset", filename), delim = ",")
    }
  }
}
```

step 3. Combine all downloaded datasets into one big dataset

In this step, we read in all the files written during the previous step, and combine the data into one big dataset to be used for further analysis and production of the maps. The procedure is contained in the script 'requestData_step3.R'. It is kept separate from the previous step because the downloading is very time-consuming and it can better be completed before this step takes place, including checks on errors, timeouts etc.

The procedure for this step is simple, and documented in the following code.

```
filelist <- list.files("data/raw_data/byDataset")
all2Data <- lapply(filelist, function(x)
  read_delim(file.path("data", "raw_data/byDataset", x),
             delim = ",",
             col_types = "ccccccTnnnccccccccccccccc"
            )
) %>%
  set_names(filelist) %>%
  bind_rows(.id = "fileID") %>%
  separate(fileID, c("mrgid", "datasetID"), "_") %>%
  mutate(mrgid = sub("[[:alpha:]]+", "", mrgid)) %>%
  mutate(datasetID = sub("[[:alpha:]]+", "", datasetID))
# mutate(mrgid = sub("region", "", mrgid))
all2Data<- all2Data %>%
  mutate(AphiaID=as.numeric(substr(aphiaidaccepted,52,65)))%>%
  filter(!is.na(AphiaID)) %>%
  filter(!is.na(decimallongitude)) %>%
  filter(!is.na(decimallatitude)) %>%
  filter(!is.na(datecollected))
write_delim(all2Data, file.path(dataDir, "all2Data.csv"), delim = ",")
```

Analysis of the species 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. For example, one of the results of the present exercise was that it appeared not a single Amphipod is called 'benthos' in WoRMS. This has been flagged as a potential problem to the taxonomic editors of WoRMS.

The checks are illustrated in the script "select_species_4_maps.R". Note that the actual downloading of the trait information of over 6000 species is commented out. Results have been written to a file after performing this step once.

```
# 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
bns<-read_delim(file.path(dataDir,"all2Data.csv"),
                col_types = "cccccccTnnnccccccccccccccccn",
                delim=",")
splst <- bns %>%
         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)
# #nsp_attr<-data.frame(AphiaID=NA,measurementTypeID=NA,measurementType=NA,
# #                     measurementValue=NA,source_id=NA,reference=NA,qualitystatus=NA,
# #                     AphiaID_Inherited=NA,CategoryID=NA)
# nsp_attr<-tibble()
# 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$AphiaID[i],include_inherited = T),silent = T)
#   if(! is.null(ttt)) nsp_attr<-rbind(nsp_attr,ttt[,1:9])
# }
#
# nsp_attr <- nsp_attr %>%
#                   mutate(AphiaID=as.numeric(AphiaID)) %>%
#                   left_join(splst,by="AphiaID")
# write_delim(nsp_attr,file.path(dataDir,"nsp_attr.Rdata"),delim=",")
nsp_attr <- read_delim(file.path(dataDir,"nsp_attr.csv"),delim=",")
# what Functional groups are there?
fg <- nsp_attr %>% filter(measurementType=="Functional group") %>%
                   select(measurementValue) %>%
                   distinct
print(fg)
# what Paraphyletic groups are there?
pfg <- nsp_attr %>% filter(measurementType=="Paraphyletic group") %>%
                    select(measurementValue) %>%
                    distinct
print(pfg)
# fill in attributes columns of splst based on the attributes downloaded from WoRMS
set_attr<-function(attr){
  tt <- nsp_attr                                  %>%
          filter(grepl(attr,measurementValue)) %>%
          select(AphiaID)                         %>%
          distinct()
  splst <- splst %>%
             mutate(!!attr:=ifelse(AphiaID %in% tt$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$Pisces_tax <- splst$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"
# 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=",")
```

Production of maps

For the production of maps, 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 intention is to use this information to produce interpolated maps covering also the non-sampled space. As a first step in visualisation, we rasterize this information and show it in a map per species.

```
#############################################################
#### load data
#############################################################
sbnsc<- read_delim("./data/derived_data/all2Data.csv",
                   col_types = "ccccccccTnnnccccccccccccccccn",
                   delim=",")
trdi<-read.csv("./data/derived_data/allDatasets_selection.csv",
               stringsAsFactors = FALSE)
usedds<- trdi %>% filter (include) %>% dplyr::select(datasetid)
splst<-read_delim(file.path(dataDir,"sp2use.csv"),
                  col_types = "dccccccclllllllllllllllll",
                  delim=",")
#############################################################
##### select few columns to work with
##### filter to only the used datasets
##### and filter to true benthic species only
```

```
#############################################################
trec<- sbnsc %>% dplyr::select(eventDate=datecollected,
                               decimalLongitude=decimallongitude,
                               decimalLatitude=decimallatitude,
                               scientificName=scientificnameaccepted,
                               aphiaID=AphiaID,
                               datasetid=datasetid) %>%
  mutate(datasetid=as.numeric(substr(datasetid,65,90))) %>%
  filter(datasetid %in% usedds$datasetid)
trec<- trec %>%  filter(aphiaID %in% splst$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) %>%
  distinct() %>%
  mutate(eventNummer=row_number())
trec <- trec %>% left_join(events,by=c('eventDate','decimalLongitude','decimalLatitude'))
########### 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)
#
# #notes. BIS (599) and Voordelta (4662) rather poor in species for the effort. is OK
# #       similar 5701 Belgian coast and 1794 French coast
# #       In the end we only keep four incomplete datasets
#
#
# manage the incomplete datasets
#
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))
}
#################################################################
# 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>200))
############ end of the generic part. What follows is a loop over the species ##
spmin<-1
spmax<-nsptoplot
pdf("./product/species_maps.pdf",width=7,height=9)

for(ss in spmin:spmax){
  spAphId<-spfr$aphiaID[ss]
  specname<-spfr$scientificName[ss]
  spcolumn<-paste0("pa",spAphId)
  progress(value=ss,max.value=spmax,init=(ss=spmin))
  # from the list of incomplete datasets, check if they have our species.
  # Only keep these, drop the others
  tt_ds<- ic_sp                          %>%
    filter(aphiaID==spAphId) %>%
    distinct(datasetid)      %>%
    bind_rows(trdi_ct        %>%
                dplyr::select(datasetid))
  # The dataset to be used consists 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= as.numeric(any(aphiaID==spAphId)))  %>%
    left_join(events,by='eventNummer')
  spesh <- spe                                  %>%
    mutate (spcolumn=(pres_abs==1)) %>%
    dplyr::select (- pres_abs)
  names(spesh)[5]<-spcolumn
  spesh <- spesh %>% dplyr::select('eventNummer',spcolumn)
  if(ss==spmin) allspe <- spesh else {
    allspe <- allspe %>% full_join(spesh,by='eventNummer')
  }
  coordinates(spe)<- ~decimalLongitude+decimalLatitude
  projection(spe)<-proWG
  r1<-rasterize(spe,r,field="pres_abs",fun=mean)
  #
  #plotting
  par(bg="lightblue")
  yor<-brewer.pal(7,"YlOrRd")
  plot(0, 0, type="n", ann=FALSE, axes=FALSE)
  par(new=TRUE)
  plot(r1,breaks=c(-0.01,0,0.2,0.4,0.6,0.8,1),
```

```
       col=yor,
       main=paste(specname,"all targeting datasets"),
       legend=FALSE)
  plot(rs,add=T,col=lcol,legend=FALSE)
  legend("bottomright",col=yor[1:6],pch=15,
         legend=c("0",">0-0.2",">0.2-0.4",">0.4-0.6",">0.6-0.8",">0.8-1"),
         bg=lcol)
}
par(bg="white")
dev.off()
evs<-tibble(eventNummer=allspe$eventNummer)
evs<- evs %>% left_join(events,by='eventNummer')
spe<- cbind(evs,allspe[,2:ncol(allspe)])
save(spe,file=file.path(mapsDir,"spe.Rdata"))
write_delim(spfr,path=file.path(mapsDir,"specieslist.csv"),delim=",")
```

Analysis of rare species

There are an astonishing number of rare species in this dataset. Around 1800 taxa have been sampled 8 times or less. In this added analysis we look where these rare taxa have been found. For this we partly recycle code from the previous chunks, to arrive finally at a shapefile with the positions of samples with rare species (including the number of rare taxa found), as well as a shapefile with all events. This is combined in GIS with base layers from Emodnet habitats.

```
############################################################
#### load data
############################################################
sbnsc<- read_delim(file.path(dataDir,"all2Data.csv"),
                   col_types = "ccccccccTnnncccccccccccccccn",
                   delim=",")
trdi<-read.csv(file.path(dataDir,"allDatasets_selection.csv"),
               stringsAsFactors = FALSE)
usedds<- trdi %>% filter (include & complete) %>% dplyr::select(datasetid)
# higher taxa than genus removed from species list, to avoid
# unidentified specimens to be called 'rare'
splst<-read_delim(file.path(dataDir,"sp2use.csv"),
                  col_types = "dccccccclllllllllllllllll",
                  delim=",") %>% filter(!is.na(genus))
############################################################
##### select few columns to work with
##### filter to only the used datasets
##### and filter to true benthic species only
############################################################
trec<- sbnsc %>% dplyr::select(eventDate=datecollected,
                         decimalLongitude=decimallongitude,
                                    decimalLatitude=decimallatitude,
                                    scientificName=scientificnameaccepted,
                                    aphiaID=AphiaID,
                                    datasetid=datasetid) %>%
                 mutate(datasetid=as.numeric(substr(datasetid,65,90))) %>%
                 filter(datasetid %in% usedds$datasetid)
trec<- trec %>%  filter(aphiaID %in% splst$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) %>%
                  distinct() %>%
                          mutate(eventNummer=row_number())
trec <- trec %>% left_join(events,by=c('eventDate','decimalLongitude','decimalLatitude'))

#################################################################
# find rare species, discard any higher taxa from the list
#
raresp<- trec                                  %>%
           group_by(aphiaID,scientificName) %>%
           summarize(n_events=n())          %>%
             arrange(desc(n_events))          %>%
           filter(n_events<9)

spe<- trec %>% filter (aphiaID %in% raresp$aphiaID)
sev<-spe %>% group_by(eventNummer) %>% summarise(nrares=n()) %>% arrange(desc(nrares))
spe<- spe %>% left_join(sev, by="eventNummer")
coordinates(spe)<- ~decimalLongitude+decimalLatitude
projection(spe)<-proWG
# par(bg="lightblue")
# plot(spe,pch=19,cex=0.3*log(spe$nrares),col="red")
# plot(rs,add=T,col=lcol,legend=FALSE)
# par(bg="white")

shapefile(spe,file.path(rareDir,"rarespecs.shp"),overwrite=TRUE)
coordinates(events) <- ~decimalLongitude+decimalLatitude
projection(events)<-proWG
shapefile(events,file.path(rareDir,"events.shp"),overwrite=TRUE)
```

Comparison of presence/absence data with numerical abundance data

Within EMODnet Biology, a separate data product has been made containing numerical abundance data of benthic species in the NOrth Sea and Baltic. This data set has been prepared based on a small selection of (large) datasets that have comparable boxcore or grab methods, and that do provide the numerical abundance estimate. In the following script the results of the presence/absence analysis is compared with the results of the dataset with numerical abundance estimates. For a random selection of 100 species, side-by-side maps using the two different approaches are produced. The code is in the script "comp_pa_num.R". Output is produced in the pdf file "compPAdens.pdf" in the product/maps directory.

```
######### read in stored data
load(file.path(mapsDir,"spe.Rdata"))
splst<-read_delim(file.path(mapsDir,"specieslist.csv"),delim=",")
names(spe)[5:ncol(spe)]<-splst$scientificName[1:(ncol(spe)-4)]
spe<-spe %>% mutate_at(5:ncol(spe),as.numeric)
# read in numerical density data
numdts<-read_delim(file.path(dataDir,"df_ab.csv"),delim=",")
evts<-numdts %>% select(data,sta,x,y) %>% distinct()


pdf(file.path(mapsDir,"compPAdens.pdf"),width=8,height=5.5)
# select at random 100 species among the first 500
```

```
spslct<-unique(floor(runif(100)*500)+1)
par(mfrow=c(1,2))
for(i in spslct){
  spp<-cbind(spe[,1:4],spe[,i+4])
  spp<-spp[!is.na(spp[,5]),]
  specname<-names(spe)[i+4]
  names(spp)[5]<-"pres_abs"
  coordinates(spp)<- ~decimalLongitude+decimalLatitude
  projection(spp)<-proWG
  r1<-rasterize(spp,r,field="pres_abs",fun=mean)
  #
  #plotting
  par(bg="lightblue")
  yor<-brewer.pal(7,"YlOrRd")
  plot(0, 0, type="n", ann=FALSE, axes=FALSE)
  par(new=TRUE)
  plot(r1,breaks=c(-0.01,0,0.2,0.4,0.6,0.8,1),
       col=yor,
       main=paste(specname,"P/A"),
       legend=FALSE)
  plot(rs,add=T,col=lcol,legend=FALSE)
  legend("bottomright",col=yor[1:6],pch=15,
         legend=c("0",">0-0.2",">0.2-0.4",">0.4-0.6",">0.6-0.8",">0.8-1"),
         bg=lcol,cex=0.6)
  ######## plot the numerical data of the same species ###############
  spp<- numdts %>% filter(tx==specname)
  spp<- spp %>% full_join(evts,by=c("data","sta","x","y")) %>%
    mutate(dens=ifelse(is.na(dens),0,dens))
  if(nrow(spp)>0){
    coordinates(spp)<- ~x+y
    projection(spp)<-proWG
    r1<-rasterize(spp,r,field="dens",fun=mean)
    md<-max(log(values(r1)+1),na.rm=T)
    r1<-log(r1+1)/md
    #
    #plotting
    par(bg="lightblue")
    yor<-brewer.pal(7,"YlOrRd")
    plot(0, 0, type="n", ann=FALSE, axes=FALSE)
    par(new=TRUE)
    plot(r1,breaks=c(-0.01,0,0.2,0.4,0.6,0.8,1),
         col=yor,
         main=paste(specname,"density"),
         legend=FALSE)
    plot(rs,add=T,col=lcol,legend=FALSE)
    legend("bottomright",col=yor[1:6],pch=15,
           legend=c("0",paste0(">0-",floor(exp(0.2*md)-1)),
                    paste0(">",floor(exp(0.2*md)-1),"-",floor(exp(0.4*md)-1)),
                    paste0(">",floor(exp(0.4*md)-1),"-",floor(exp(0.6*md)-1)),
                    paste0(">",floor(exp(0.6*md)-1),"-",floor(exp(0.8*md)-1)),
                    paste0(">",floor(exp(0.8*md)-1),"-",floor(exp(1.0*md)-1))),
           bg=lcol,cex=0.6)
  }else{
```

```
    plot(0, 0, type="n", ann=FALSE, axes=FALSE)
    par(new=TRUE)

  }

}
dev.off()
```