# Experiences in dowloading and parsing large files with abundance and biomass data

*Peter M.J. Herman*

*23-9-2020*

## Introduction

In this report I document the operations on the EMODNET Biology databases, needed to reconstruct a large-scale overview of abundance of benthic animals, and combining several independent data sets. In particular, I will highlight the problems and pecularities to take into account when processing these data. This experience can hopefully be used to evaluate the database structure and the data download and parsing tools available.

## Basic data download

Using the online download tool, a URL can be constructed that downloads a specific data set. In this case, I have downloaded whatever was available for the dataset. A specific point of attention is that the availability of measusrments and facts multiplies the number of records in the downloaded data file. A single species occurrence results in 10 records, if 10 different paramters (e.g. abundance, dry weight, area sampled etc..) are stored in the database. For that reason, one easily exceeds the built-in limit of 1,000,000 records in the files to be downloaded. In order to avoid this problem, I have downloaded the data per year and per dataset, and afterwards recompiled the datasets by stacking all years on top of one another.

The code also shows the datasets concerned. Note that, apart from these data, also the datasets Macrobel (Belgian macrofauna data mainly from Gent University) and NSBS (North Sea Benthos Survey) were included. Both had serious problems and should be re-introduced into the database of Emodnet Biology. For NSBS, based on old files I still had on my computer, I have made a restoration that was added as a file NSBS_TOTAL.csv. The code for that is separate from the code discussed here.

```
getDatasets <- data.frame(name=c("Macrobel","Mareano","NSBS","MWTL","ODAM","POHJE",
                                 "Polish","Puck","Rebent","RSMP","SHARK"),
                          datasetid=c(145,4539,67,5759,4494,5725,2467,611,4412,5922,2454),
                          include=c(T,T,T,T,T,T,T,T,T,T,T))
getDatasets <- getDatasets %>% filter(include)
for(jj in 1:length(getDatasets$datasetid)){
  for(year in 1980:2019){
    datasetid <- getDatasets$datasetid[jj]
    datasetname<-getDatasets$name[jj]
    decadename<-decades$nams[ii]
    begindate <- paste0(year,"-01-01")
    enddate   <- paste0(year,"-12-31")
    print(paste("downloading data for dataset",datasetname,"ID nr: ", datasetid,"year",year))
    downloadURL <- paste0(
      "http://geo.vliz.be/geoserver/wfs/ows?service=WFS&version=1.1.0&",
      "request=GetFeature&typeName=Dataportal%3Aeurobis-obisenv&",
      "resultType=results&viewParams=where%3Adatasetid+IN+%28",
      datasetid,
      "%29+AND+%28%28observationdate+BETWEEN+%27",
      begindate,
      "%27+AND+%27",
      enddate,
      "%27+%29%29%3Bcontext%3A0100&propertyName=",
      "datasetid%2C",
      "datecollected%2C",
      "decimallatitude%2C",
      "decimallongitude%2C",
      "scientificname%2C",
      "aphiaid%2C",
      "scientificnameaccepted%2C",
      "institutioncode%2C",
      "collectioncode%2C",
      "yearcollected%2C",
      "monthcollected%2C",
      "daycollected%2C",
      "recordnumber%2C",
      "fieldnumber%2C",
      "minimumdepthinmeters%2C",
      "occurrenceid%2C",
      "scientificnameauthorship%2C",
      "scientificnameid%2C",
      "taxonrank%2C",
      "kingdom%2C",
      "phylum%2C",
      "class%2C",
      "order%2C",
      "family%2C",
      "genus%2C",
      "subgenus%2C",
      "specificepithet%2C",
      "infraspecificepithet%2C",
      "aphiaidaccepted%2C",
      "samplingeffort%2C",
      "samplingprotocol%2C",
      "qc%2C",
      "eventid%2C",
      "parameter%2C",
      "parameter_value%2C",
      "parameter_group_id%2C",
      "parameter_measurementtypeid%2C",
      "parameter_bodcterm%2C",
      "parameter_bodcterm_definition%2C",
      "parameter_standardunit%2C",
      "parameter_standardunitid%2C",
      "parameter_original_measurement_type%2C",
      "parameter_original_measurement_unit%2C",
      "parameter_conversion_factor_to_standard_unit%2C",
      "event%2C",
      "event_type%2C",
      "event_type_id",
      "&outputFormat=csv"
    )
    data <- read_csv(downloadURL)
    filename = paste0(datasetname,"_",datasetid,"_",year,".csv")
    if(nrow(data) != 0){
```

```r
      write_delim(data, file.path(downloadDir, filename), delim = ",")
    }
  }
}



filnams<-list.files(downloadDir)

for (ds in getDatasets$name){
  filelist<-filnams[grep(ds,filnams)]

  all2Data <- lapply(filelist, function(x)
    read_delim(file.path(downloadDir, x),
             delim = ",",
             col_types = "cnccccTnnnccnnnccccncccccccccccccccnccncccccccnccc"
    )
  ) %>%
  set_names(filelist) %>%
  bind_rows(.id = "fileID")


  filename = paste0(ds,"_TOTAL.csv")
  write_delim(all2Data,file.path(totalsDir,filename),delim=",")
  rm(all2Data)
}
```

## Working up the data files

The code contains a small section to read in the files written in the previous step, before the calculations and further treatment starts. The function retrieve_data will be called in the main loop of the program.

```r
datfils<-c("SHARK_TOTAL.csv",
           "RSMP_TOTAL.csv",
           "REBENT_TOTAL.csv",
           "PUCK_TOTAL.csv",
           "POLISH_TOTAL.csv",
           "POHJE_TOTAL.csv",
           "ODAM_TOTAL.csv",
           "MWTL_TOTAL.csv",
           "MAREANO_TOTAL.csv")

retrieve_data<-function(fil){
  data<-read_delim(file.path(totalsDir,fil),delim=",",
                   col_types = "ccnccccTnnnccnnnccccncccccccccccccccnccncccccccnccc")
  return(data)
}
```

In working up the data files, a number of problems were encountered that required special adaptations in the script. These will be discussed here, in the order they appear in the script. They will be illustrated by the code used.

### Inconsistent indication of the area sampled

There are four different ways, across these 9 files, in which the area sampled by the corer or grab is indicated. In some cases, the field "samplingeffort" is used for this. We come back on that later. In most other files, the parameter "AreaBedSamp (mˆ2)" is used, except for one file where "InstrumentSurfaceArea (mˆ2)" indicates exactly the same parameter. I therefore added a translation table for this pair of parameters. This problem is limited to this parameter only, so the introduction of a translation table was a bit of an overkill.

The code gives the translation array. It further illustrates how the basic casting of the data is done, including the code where translation is performed where needed. In the casting, care was taken to include all variables that could be of importance. This included occurrenceid, which made sure that no summarizing function for the parameter_value needed to be given, as it never occurred that the same parameter was found twice for the same value of all the left-hand side variables. Not giving a summarizing function is a good method to implicitly check for this condition, as a warning is produced during casting when the need for such a function is encountered.

```
stand_par_nam<-tibble(oldnam=c("InstrumentSurfaceArea (m^2)"),
                      newnam=c("AreaBedSamp (m^2)"))

do_cast<-function(data,fil){
  datarr<-dcast(data,
               datasetid+institutioncode+collectioncode+eventid+
                 datecollected+yearcollected+
                 monthcollected+daycollected+recordnumber+fieldnumber+decimallongitude+
                 decimallatitude+minimumdepthinmeters+occurrenceid+scientificname+
                 scientificnameid+aphiaid+taxonrank+scientificnameaccepted+
                 scientificnameauthorship+aphiaidaccepted+kingdom+phylum+class+order+family+
                 genus+subgenus+specificepithet+infraspecificepithet+samplingeffort+samplingprotocol+
                 qc
               ~ parameter,
               value.var="parameter_value"
  )
  parnams<-names(datarr)[34:length(names(datarr))]
  for(i in 1:nrow(stand_par_nam)){
    parnams[parnams==stand_par_nam$oldnam[i]]<-stand_par_nam$newnam[i]
  }
  names(datarr)[34:length(names(datarr))]<-parnams
  return(datarr)
}
```

### No indication of the nature of the parameter_value field

The parameter_value can be a string, a numeric value, a date, time, or whatever. This depends on the nature of parameter. However, although we are given plenty of details on parameter, including the BODC definition and the reference to the BODC site, a simple indication of the nature of the value is not given. That creates a problem. After casting the long file in square format, we have columns with the values of the different parameters. If these values are numbers, we want to summarize, sum, average them etc., but in order to do so they need to be transformed into numeric columns. For now, I have introduced a list of numeric parameters in my code to solve this problem, but a small addition in the data file could easily solve this problem in a much more elegant way. For now, the function make_num does the job.

```
numfields<-c("AreaBedSamp (m^2)",
             "BedAbund (#/m^2)",
             "BedAshFreeBiom (g/m^2)",
             "BedCoverage (%)",
             "BedDryWtBiom (g/m^2)",
```

```r
             "BedWetWtBiom (g/m^2)",
             "Count (Dmnless)",
             "DWBiom_Samp (kg)",
             "Length (mm)",
             "LowSieveMesh (mm)",
             "NetMesh (mm)",
             "Salinity (ppt)",
             "SubSamplingCoefficient (Dmnless)",
             "UpSieveMesh (mm)",
             "VolWBodySamp (m^3)",
             "WC_Temp (degC)",
             "WWBiom_Samp (kg)"
             )

make_num<-function(datarr,fil){
  parnams<-names(datarr)
  datarr <- datarr %>% as_tibble() %>%
    mutate(across(parnams[parnams %in% numfields],as.numeric))

}
```

**No clear and systematic indication of the sampling, replication and subsampling structure**

Throughout the data analysis, we used as an operational definition of a "sample" all recorded occurrences that share date, time, latitude, longitude and depth. In many cases such a sample consists of several replicate grabs, box corers or other units of sampling. The data structure lacks a clearly identified field to introduce this unit. In several datasets, the field "eventid" is effectively used for this, but there are several exceptions. In SHARK, e.g., we need the field "fielnumber" to find back what is most probably the replication unit. In POHJE, most samples are sieved over two different sieves, and animals are recorded from both sieves. The sieve information is added to the field "eventid", which makes this field unfit to be used as an indicator of replicate sample units. The structure of the POHJE file is so complicated that a dedicated clean-up function had to be written that performs a separate cast to get rid of non-understandable and non-informative subsampling information that clutters the file.

We propose that a clearer structure is adopted, and that one makes sure there is a field in the database that will always indicate unity of the basic sampling unit (e.g. one grab). "Sample" can be derived from place and time if need be, although it would be very convenient to also have a field for that. Subsampling is a separate problem, especially if the subsampling is selective as in the case of the different sieves. If one wants to restrict data to one sieve size only, it is important to know what sieve has been used for every occurrence. Sieve size is a characteristic of the sample that therefore should be incorporated alongside sampling methodology and sampling surface. It is not well possible to include this in the Measurements or Facts, as it is an attribute that needs to be known for the occurrence.

The following code gives the clean-up function for POHJE, and further shows the code used to determine the number of replicates per sampling event. It assumes that eventid has a unique value for each basic sampling unit (e.g. grab) that constitutes a replicate. In some datasets, information found elsewhere was transferred to this field beforehand. Further the function calculates the total area sampled by all replicates together in a sample. It assumes the surface sampled by a replicate can be found in the field "AreaBedSamp (m^2)". In some datasets the information had to be retrieved and stored there beforehand.

```r
cleanPOHJE<-function(data){
  datar<-dcast(data,
               id+datasetid+institutioncode+collectioncode+eventid+datecollected+yearcollected
               +monthcollected+daycollected+recordnumber+fieldnumber+decimallongitude+
                 decimallatitude+minimumdepthinmeters+occurrenceid+scientificname+
                 scientificnameid+aphiaid+taxonrank+scientificnameaccepted+
                 scientificnameauthorship+aphiaidaccepted+kingdom+phylum+class+order+family+
                 genus+subgenus+specificepithet+infraspecificepithet+samplingeffort+samplingprotocol
               +qc+parameter+parameter_value+parameter_group_id+parameter_measurementtypeid+
                 parameter_bodcterm+parameter_bodcterm_definition+parameter_standardunit+
                 parameter_standardunitid+parameter_original_measurement_type+
                 parameter_original_measurement_unit+
                 parameter_conversion_factor_to_standard_unit ~ event_type,
               fun.aggregate=length,
               value.var="event_type_id")
  # correct missing parameter field when it should be description of the bed
  datar <- datar %>%
    mutate(parameter=ifelse(is.na(parameter) & parameter_bodcterm=="Description of the bed",
                            "BedDescrip",
                            parameter)
    )
  # remove Sieve information from eventid, so that events and not subsamples are indicated
  datar<- datar %>%
    mutate(dd=regexpr("-Sie",eventid)) %>%
    mutate(eventid=ifelse(dd>0,
                          substr(eventid,1,dd-1),
                          eventid
                          )
           )
  return(datar)
}



add_nrep<-function(datarr,fil){
  nrep<- datarr %>%
    group_by(datecollected,decimallatitude,decimallongitude,minimumdepthinmeters) %>%
    summarize(nrep=length(unique(eventid))) %>%
    ungroup() %>%
    mutate(sampid=row_number())

  datarr <- datarr %>%
    left_join(nrep,by=c("datecollected","decimallatitude","decimallongitude","minimumdepthinmeters"))

  arsamps<-datarr %>%
    group_by(sampid,eventid)%>%
    summarise(arsamp=mean(`AreaBedSamp (m^2)`)) %>%
    ungroup() %>%
    group_by(sampid) %>%
    summarise(sum_areasamp=sum(arsamp))

  data_w_rep<-datarr %>%
    left_join(arsamps,by='sampid')

  return(data_w_rep)
}
```

## Difficulties to calculate abundances and biomasses per unit surface

The following code shows how abundance (number per m2), biomass density (different weight systems, g per m2), counts (individuals per sample) and mass (different weight units per sample) are calculated based on the variety of information that may be present in the different files. This calculation is quite error-prone. One of the problems is that it is very difficult to obtain an overview of these very large files, and one never knows what is happening somewhere in the middle of the records.

It would be highly advisable that all data originators provide their data not only in the original format (e.g. counts per sample, together with area of the sample), but also provide abundance and aerial biomass density data directly to the database. That puts the responsibility for checking these calculations with the people who know the data best. Only rarely will a user of the data need the real original counts, it is far more common that abundance or biomass density is used. Besides, in the rare case where one needs the original counts, they can always be reconstructed from abundance and size of the sampling unit.

In doing the calculations, one also bumps into strange unit problems. If biomass density is expressed as g/m2, and area sampled is expressed as m2, why is mass per sample than expressed in kg and not in g? Consistency would avoid many errors here.

```
sum_avg_param<-function(datarr,fil){
  pp<-names(datarr)
  data_sap<- datarr %>%
    group_by(datasetid,collectioncode,
             datecollected,yearcollected,monthcollected,daycollected,
             decimallongitude,decimallatitude,minimumdepthinmeters,
             sampid,nrep,scientificnameaccepted,
             scientificnameauthorship,aphiaidaccepted,
             kingdom,phylum,class,order,family,genus,subgenus,specificepithet,
             sum_areasamp) %>%
    summarise(avg_bedabund=
                ifelse("BedAbund (#/m^2)" %in% pp,
                  ifelse(!is.na(sum(`AreaBedSamp (m^2)`)),
                    sum(`BedAbund (#/m^2)`*`AreaBedSamp (m^2)`)/mean(sum_areasamp),
                    sum(`BedAbund (#/m^2)`)/mean(nrep)
                  ),
                  ifelse("Count (Dmnless)" %in% pp & !is.na(mean(sum_areasamp)),
                    sum(`Count (Dmnless)`)/mean(sum_areasamp),
                    NA/mean(nrep)
                  )
                ),
              avg_bedwetwtbiom=
                ifelse("BedWetWtBiom (g/m^2)" %in% pp,
                  ifelse(!is.na(sum(`AreaBedSamp (m^2)`)),
                    sum(`BedWetWtBiom (g/m^2)`*`AreaBedSamp (m^2)`)/mean(sum_areasamp),
                    sum(`BedWetWtBiom (g/m^2)`)/mean(nrep)
                  ),
                  ifelse("WWBiom_Samp (kg)" %in% pp & !is.na(mean(sum_areasamp)),
                         sum(`WWBiom_Samp (kg)`)*1000/mean(sum_areasamp),
                         NA/mean(nrep)
                  )
                ),
              avg_beddrywtbiom=
                ifelse("BedDryWtBiom (g/m^2)" %in% pp,
                  ifelse(!is.na(sum(`AreaBedSamp (m^2)`)),
                    sum(`BedDryWtBiom (g/m^2)`*`AreaBedSamp (m^2)`)/mean(sum_areasamp),
                    sum(`BedDryWtBiom (g/m^2)`)/mean(nrep)
                  ),
                  ifelse("DWBiom_Samp (kg)" %in% pp & !is.na(mean(sum_areasamp)),
                         sum(`DWBiom_Samp (kg)`)*1000/mean(sum_areasamp),
                         NA/mean(nrep)
                  )
                ),
              avg_bedafdwtbiom=
                ifelse("BedAshFreeBiom (g/m^2)" %in% pp,
                  ifelse(!is.na(`AreaBedSamp (m^2)`),
                    sum(`BedAshFreeBiom (g/m^2)`*`AreaBedSamp (m^2)`)/mean(sum_areasamp),
                    sum(`BedAshFreeBiom (g/m^2)`)/mean(nrep)
                  ),
                  NA/mean(nrep)
                ),
              sum_count=
                ifelse("Count (Dmnless)"%in% pp,
                  sum(`Count (Dmnless)`),
                  NA/mean(nrep)
                ),
              sum_drywtbiomsamp=
                ifelse("DWBiom_Samp (kg)" %in% pp,
                  sum(`DWBiom_Samp (kg)`),
                  NA/mean(nrep)
                ),
              sum_wetwtbiomsamp=
                ifelse("WWBiom_Samp (kg)" %in% pp,
                  sum(`WWBiom_Samp (kg)`),
                  NA/mean(nrep)
                )
    )
  return(data_sap)
}
```

## A main loop with more exceptions than loop

The main loop is, in principle, very simple. A data file is read, the data are cast in square format, numerical fields are made numeric, the number of replicates per sample is calculated alongside (if that information exists) with the total area sampled, the sum and/or average of the relevant abundance and biomass variables is calculated, and the results are written to a file.

However, the majority of the following code consists of all the changes that were needed for the different data sets. These exceptions and small tweaks illustrate e.g. where information on area sampled could be found (in a variable that is, strange enough, a character variable that contains the unit in its value field instead of in the header, and that uses decimal commas in some data sets instead of decimal points). It illustrates that for some parameters the infilling of the field 'parameter' has failed, so that one does not have this essential information prior to the parameter_value. Most often, however, it can be derived from other information in the file when this occurs. Comments in the code give information on what the tweaks were about.

```
# main loop over the datasets
nds<-length(datfils)
nams<-substr(datfils,1,regexpr("_",datfils)-1)

for(ds in 1:nds){
  fil<-datfils[ds]
```

```
# read the data
data<-retrieve_data(fil)

      # for POHJE, some initial cleaning of the subsampling mess is needed
        if(fil=="POHJE_TOTAL.csv")data<-cleanPOHJE(data)
      # for MWTL some parameter fields are empty. We fill them
        if (fil=="MWTL_TOTAL.csv"){
          data<- data %>%
            mutate(parameter_new=ifelse(parameter_measurementtypeid==
                                          "http://vocab.nerc.ac.uk/collection/P01/current/SACFOR01/",
                                        "AbundCat (Dmnless)",parameter))%>%
            mutate(parameter_new=ifelse(parameter_measurementtypeid==
                                          "http://vocab.nerc.ac.uk/collection/P01/current/UKMH0405/",
                                        "JNCC_Class (Dmnless)",parameter_new))%>%
            dplyr::select(-parameter)%>%
            mutate(parameter=parameter_new,.keep="unused")
        }
      # for MAREANO, some records have parameter= NA. We drop them
        if (fil=="MAREANO_TOTAL.csv") data<-data %>% filter (!is.na(parameter))

# cast the parameters
data1<-do_cast(data,fil)

      # derive the area sampled from the character field samplingeffort in PUCK and ODAM
        if (fil=="PUCK_TOTAL.csv" | fil=="ODAM_TOTAL.csv"){
          data1<-data1 %>%
            mutate(samplingeffort=gsub(" m²","",samplingeffort)) %>%
            mutate(`AreaBedSamp (m^2)`=gsub(",",".",samplingeffort))
        }
      # and just that slight little difference for POLISH
        if(fil=="POLISH_TOTAL.csv"){
          data1<-data1 %>%
            mutate(`AreaBedSamp (m^2)`=gsub(" m2","",samplingeffort))
        }
      # select only core samples from MWTL, and restrict to records with abundance
        if (fil=="MWTL_TOTAL.csv"){
          data1 <- data1 %>%
            filter(`Instrument (Dmnless)`=="unconsolidated sediment corers" |
                     `Instrument (Dmnless)`== "Hamon happer") %>%
            filter(!is.na(`BedAbund (#/m^2)`))
        }

# make all numerical parameters numeric
data2<-make_num(data1,fil)

      # for POHJE,calculate abundance where it is lacking but count and areasampled are given
        if(fil=="POHJE_TOTAL.csv"){
          data2 <- data2 %>%
            mutate(`BedAbund (#/m^2)`=
                     ifelse(
                       is.na(`BedAbund (#/m^2)`)&!is.na(`Count (Dmnless)`)&!is.na(`AreaBedSamp (m^2)`
                       `Count (Dmnless)`/`AreaBedSamp (m^2)`,
                       `BedAbund (#/m^2)`
                       )
              ) %>%
              filter(!is.na(`BedAbund (#/m^2)`))
          }

        # remove records without area sampled in RSMP, as we only have counts
          if(fil=="RSMP_TOTAL.csv")data2 <- data2 %>% filter (!is.na(`AreaBedSamp (m^2)`))

        # In REBENT, there are missing areabedsample values. all of these have been taken with a Smith-M
        # and all the Smith-McIntyre grab samples that do have an area sampled, have 0.1 for this area
        # we replace the NA with 0.1
        # In addition, in REBENT there are records without a count (mostly algae in quadrats, presence o
          if(fil=="REBENT_TOTAL.csv"){
            data2 <- data2 %>%
              mutate (`AreaBedSamp (m^2)`=ifelse(is.na(`AreaBedSamp (m^2)`),0.1,`AreaBedSamp (m^2)`)) %>
              filter (!is.na(`Count (Dmnless)`))
          }

        # use fieldnumber instead of eventid to identify events in SHARK
          if(fil=="SHARK_TOTAL.csv") data2$eventid<-data2$fieldnumber

        # no information on area samples in MAREANO. Rename samplingeffort (only NA)
          if (fil=="MAREANO_TOTAL.csv") data2$`AreaBedSamp (m^2)`<-as.numeric(data2$samplingeffort)

  # determine number of replicates per sampling occasion (date, place), and add column to the data frame
  data3<-add_nrep(data2,fil)

  # Average/sum parameters over samples
  data4<-sum_avg_param(data3,fil)

  # write results
  #pth<-file.path(dataDir,paste0(nams[ds],"_cross_CA.csv"))
  #write_delim(data4,pth,delim=",")

  pth<-file.path(dataDir,paste0(nams[ds],"_cross_CA.Rdata"))
  save(data4,file=pth)

  rm(data,data1,data2,data3,data4)
}
```

**Conclusions on parsing the data**

It is clear from this code that the data files, although already standardized to a fair degree, still have a number of differences that frustrate the machine-based extraction of information. Where it is justified, some difference in structure or content of the files is allowable provided this is internally documented. However, much of the variation presently found is only clutter and does not serve a purpose.

Some clean-up actions are purely technical. It is unclear why the area sampled by a grab is given in one dataset in the field samplingeffort as "0,1 m2", in another dataset as "0,1 m²" or "0.1 m2", while in most other datasets it is in MoF under parameter "AreaBedSamp (m^2)". MoF with parameter_values given, but the field parameter as "NA" can often be corrected, but if not can better be removed as one does not know what the measurement is.

Other clean-up actions can consist of the removal of some records from the datasets unless they are completed. As an example, there is no point in storing counts per sample if it cannot be retrieved what the size of the samples was.

Clean-up can also consist of rethinking the organisation of the data in datasets. It is questionable if it is a good idea to put all information from one data provider into a single dataset. Why are camera observations without numerical abundance information in MWTL mixed with box core information that provides abundance and (ash-free-dwt) biomass for all species? Would it not be much more useful to have both datasets separate, as no one is probably going to use both parts of the data set in one analysis (and even if that would happen, what is the problem of downloading two datasets instead of one?).

Finally, a rethinking of the representation of the basic sampling, replication and subsampling structure is needed. This is currently a large source of uncertainty relying on the correct interpretation of what is in eventid or in fieldnumber etc.

## Collecting all data

After casting each of the datasets, all data are collected into a single file.

```
# collect all records and save them

fl<-tibble(fn=list.files(file.path(dataDir)))
fl<- fl %>% filter (grepl("_CA.Rdata",fn))

for(i in (1:nrow(fl))){
  f<-fl$fn[i]
  load(file.path(dataDir,f))
  if(i==1)all_recs<-data4 else all_recs<-rbind(all_recs,data4)
}

write_delim(all_recs,file.path(dataDir,"all_recs.csv"),delim=",")
save(all_recs,file=file.path(dataDir,"all_recs.Rdata"))
```

## Selecting the species to be plotted

From this raw composed data file, the species suitable for further treatment are selected. As this dataset concentrates on macrozoobenthos, the selection of species attempts to concentrate on this group only. The procedures have been used and described in the product on deriving presence/absence data, and are identical to the procedures used there.

```
#### SPECIES SELECTION ############
load(file.path(dataDir,"all_recs.Rdata"))

# 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
### extracting numeric aphiaid: mutate(AphiaID=as.numeric(substr(aphiaidaccepted,52,65)))


splst <- all_recs %>%
  ungroup() %>%
  filter(!is.na(aphiaidaccepted))%>%
  mutate(tmp=substr(aphiaidaccepted,52,65))%>%
  filter(tmp!="NA")%>%
  mutate(AphiaID=as.numeric(tmp)) %>%
  dplyr::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<-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.csv"),delim=",")
# save(nsp_attr,file=file.path(dataDir,"nsp_attr.Rdata"))
#


nsp_attr <- read_delim(file.path(dataDir,"nsp_attr.csv"),delim=",")
# what Functional groups are there?
fg <- nsp_attr %>% filter(measurementType=="Functional group") %>%
  dplyr::select(measurementValue) %>%
  distinct
print(fg)
# what Paraphyletic groups are there?
pfg <- nsp_attr %>% filter(measurementType=="Paraphyletic group") %>%
  dplyr::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)) %>%
    dplyr::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")
```

```r
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=",")
```

## Prepare the data for the production of maps

After loading the data and the list of species to use, the dataset is filtered to contain only the true macrobenthic species information. Species are ordered in the order of their frequency (number of records with presence) in the data set. Events are defined, across the dataset, as samples that have a unique position in space (coordinates, depth) and time. A square matrix is prepared that contains, in the first columns, information on location, time and event, and than has one column per species with the numerical abundance. Note that there is also quite some information on biomass in the basic data file. In principle, a similar file could also be made for biomass, but some conversion between wet weight, dry weight and ash-free dry weight would be needed for that. The product of this step is a file with the numerical abundance (including all zeroes) of all species in all samples. This is the basic data product. It is written as a csv file and as a binary file. It will be used in preparing the maps and the interpolations.

```
##########################################################
#### load data
##########################################################
sbnsc<- read_delim(file.path(dataDir,"all_recs.csv"),
                   col_types = "ccTnnnnnnnnccccccccccnnnnnnnn",
                   delim=",")
splst<-read_delim(file.path(dataDir,"sp2use.csv"),
                  col_types = "dcccccccllllllllllllllll",
                  delim=",")
##########################################################
##### filter to true benthic species only
##########################################################
trec<- sbnsc %>% as_tibble() %>%
  mutate(datasetid=as.numeric(substr(datasetid,65,90))) %>%
  filter(!is.na(aphiaidaccepted))%>%
  mutate(tmp=substr(aphiaidaccepted,52,65))%>%
  filter(tmp!="NA")%>%
  mutate(AphiaID=as.numeric(tmp)) %>%
  dplyr::select(-tmp)%>%
  filter(AphiaID %in% splst$AphiaID)
##############################################################
# find occurrence frequency of all species, and rank the species accordingly
#
spfr<- trec %>%
  group_by(AphiaID,scientificnameaccepted) %>%
  summarize(n_events=n()) %>%
  arrange(desc(n_events))
nsptoplot<-length(which(spfr$n_events>0))
###############################################################
# make a list of all sampling events
events<- trec %>%
  dplyr::select(datasetid,datecollected,decimallongitude,decimallatitude,minimumdepthinmeters,sampid)%>
  distinct()%>%
  mutate(eventNummer=row_number())
trec <- trec %>%
  left_join(events,by=c("datasetid","datecollected","decimallongitude","decimallatitude",
                        "minimumdepthinmeters","sampid"))
############ end of the generic part. What follows is a loop over the species ##

spmin<-1
spmax<-nsptoplot
spesh<-events %>% arrange(across(eventNummer))
offs<-ncol(spesh)

for(ss in spmin:spmax){
  progress(value=ss,max.value=spmax,init=(ss=spmin))
  spAphId<-spfr$AphiaID[ss]
  specname<-spfr$scientificnameaccepted[ss]
  spcolumn<-paste0("ab_",spAphId)

  spe<- trec%>%
    filter(AphiaID==spAphId)  %>%
    dplyr::select(avg_bedabund,eventNummer)
  intmd<-events %>%
    left_join(spe,by="eventNummer")%>%
    mutate(avg_bedabund=ifelse(is.na(avg_bedabund),0,avg_bedabund))%>%
    arrange(across(eventNummer))  %>%
    dplyr::select(avg_bedabund)
  spesh<- spesh %>%
    bind_cols(intmd)
  names(spesh)[offs+ss]<-spcolumn
}

save(spesh,file=file.path(mapsDir,"spesh.Rdata"))
write_delim(spfr,path=file.path(mapsDir,"specieslist.csv"),delim=",")
write_delim(spesh,path=file.path(mapsDir,"spesh.csv"),delim=",")
```

## Producing the maps

The procedure for rasterizing the data and drawing the maps is, again, very similar to the procedure described for the presence/absence data on macrobenthos. Basically, a ggplot2 map is drawn based on the species data prepared in the previous step. In this procedure, species are selected for plotting based on a minimum frequency (of 100 samples, out of 80000). For the maps, the log10-transformed abundance is used. This choice is made in this code chunk. If other choices for transformation (including no transformation) are made, this can be done here as the basic data are stored without transformation.

```
proWG<-CRS("+proj=longlat +datum=WGS84")

##########################################################
# define a raster covering the grid. Set resolution of the raster here
##########################################################
r<-raster(ext=extent(-16,36,40,74),ncol=156,nrow=238,crs=proWG,vals=0)
##########################################################
# load some information needed for the maps
##########################################################
# Show countries
world <- ne_countries(scale = "medium", returnclass = "sf")
# EMODnet colors
emodnetColor <- list(
  # First palette
  blue = "#0A71B4",
  yellow = "#F8B334",
  darkgrey = "#555555",
  # Secondary palette,
  darkblue = "#012E58",
  lightblue = "#61AADF",
  white = "#FFFFFF",
  lightgrey = "#D9D9D9"
)
# EMODnet logo
logo_raw <- image_read("https://www.emodnet-biology.eu/sites/emodnet-biology.eu/files/public/logos/logo
logo <- logo_raw %>% image_scale("150")
#
#########################################################
# load data
#
load(file.path(mapsDir,"spesh.Rdata"))
spfr<-read_delim(file.path(mapsDir,"specieslist.csv"),delim=",")

# set minimum species frequency for plot
nsptoplot<-length(which(spfr$n_events>100))

#################################################################3
# map production. Subsetting, e.g. by time, can occur here

spmin<-1
spmax<-nsptoplot

for(ss in spmin:spmax){
  spAphId<-spfr$AphiaID[ss]
  specname<-spfr$scientificnameaccepted[ss]
  spcolumn<-paste0("ab_",spAphId)
```

```r
progress(value=ss,max.value=spmax,init=(ss=spmin))

spe <- spesh %>%
  dplyr::select(decimallongitude,decimallatitude,all_of(spcolumn))
names(spe)[3]<-"abund"

# introduce a transformation of the data here
spe <- spe %>%
  mutate(abund=log(abund+1)/log(10))
r1nam<-"log10_abund"

coordinates(spe)<- ~decimallongitude+decimallatitude
projection(spe)<-proWG
r1<-rasterize(spe,r,field="abund",fun=mean)
names(r1)<-r1nam

# Export rasters as tif
raster::writeRaster(
  r1,
  file.path(
    rasterDir, paste0(
      sprintf("%04d",ss), "_",
      spAphId, "_",
      gsub(" ", "-", specname),
      ".tif"
    )
  ),
  overwrite=TRUE
)
#
# Transform raster to vector

grid <- sf::st_as_sf(raster::rasterToPolygons(r1))
grid_bbox <- sf::st_bbox(sf::st_transform(grid, 3035))

# Plot the grid
plot_grid <- ggplot() +
  geom_sf(data = world,
          fill = emodnetColor$darkgrey,
          color = emodnetColor$lightgrey,
          size = 0.1) +
  geom_sf(data = grid, aes(fill = log10_abund), color=NA) +
  coord_sf(crs = 3035, xlim = c(grid_bbox$xmin, grid_bbox$xmax), ylim = c(grid_bbox$ymin, grid_bbox$y
  scale_fill_viridis_c(alpha = 1, begin = 0, end = 1, direction = 1) +
  ggtitle(specname,
          subtitle = paste0('AphiaID ', spAphId)) +
  theme(
    panel.background = element_rect(fill = emodnetColor$lightgrey),
    plot.title = element_text(color= emodnetColor$darkgrey, size = 14, face="bold.italic", hjust = 0.
    plot.subtitle = element_text(color= emodnetColor$darkgrey, face="bold", size=10, hjust = 0.5)
  )

# Inspect plot
```

```
  plot_grid

  # Save plot
  filnam<-file.path(plotsDir,
                    paste0(sprintf("%04d",ss), "_",spAphId, "_",gsub(" ", "-", specname),".png"))
  ggsave(filename = filnam,width = 198.4375, height = 121.70833333, dpi = 300, units = "mm")

  # # Add emodnet logo
  # plot <- image_read(filnam)
  # final_plot <- image_composite(plot, logo, gravity = "southeast", offset = "-680-220")
  # image_write(final_plot, filnam)

}
```