species and traits

Tom Webb

17/06/2020

A dataset of substrate preferences for European benthic invertebrates

This document describes the steps taken to create a single dataset of substrate preferences for a selection of European benthic invertebrates, using data from MarLIN's BIOTIC database, supplemented with taxonomic and species attribute data from WoRMS and occurrence data from OBIS.

First step is to load required packages

```
library(tidyverse)
library(worrms)
library(robis)
library(here)
```

Now read in datasets - these are datasets including all information on individual traits, supplied from BIOTIC by Dan Lear in June 2020:

```
env_pos <- read_csv(here::here("data", "raw_data/biotic/bioticEnvPos.csv"))
feed_method <- read_csv(here::here("data", "raw_data/biotic/bioticFeedingMethod.csv"))
habit <- read_csv(here::here("data", "raw_data/biotic/bioticHabit.csv"))
substratum <- read_csv(here::here("data", "raw_data/biotic/bioticSubstratum.csv"))
dev_mech <- read_csv(here::here("data", "raw_data/biotic/bioticDevMech.csv"))
```

The main trait of interest is `substratum`, which includes species-level substrate preferences. To faciliate working with this, we create a lookup table with full substrate names and abbreviated names:

```
substrate_values <- tibble(
  substratum = unique(substratum$substratum)
  ) %>%
  mutate(substratum_code = case_when(
    substratum == "Large to very large boulders" ~ "Large_VLarge_Boulders",
    substratum == "Small boulders" ~ "Small_Boulders",
    substratum == "Muddy gravel" ~ "Muddy_gravel",
    substratum == "Muddy sand" ~ "Muddy_sand",
    substratum == "Sandy mud" ~ "Sandy_mud",
    substratum == "Coarse clean sand" ~ "Coarse_sand",
    substratum == "Fine clean sand" ~ "Fine_sand",
    substratum == "Other species (see additional information)" ~ "Other_species",
    substratum == "Artificial (e.g. metal/wood/concrete)" ~ "Artificial",
    substratum == "Insufficient information" ~ "No_Info",
    substratum == "Gravel / shingle" ~ "Gravel_shingle",
    substratum == "Salt marsh" ~ "Salt_marsh",
    substratum == "Biogenic reef" ~ "Biogenic_reef",
    substratum == "Under boulders" ~ "Under_boulders",
    substratum == "Crevices / fissures" ~ "Crevices",
```

```
    substratum == "Water column (pelagic)" ~ "Pelagic",
    substratum == "No preference" ~ "No_preference",
    substratum == "Muddy gravelly sand" ~ "Mud_grav_sand",
    substratum == "Sandy gravelly mud" ~ "Sand_grav_mud",
    substratum == "See additional information" ~ "No_Info",
    substratum == "Gravelley sand" ~ "Gravel_sand",
    substratum == "Muddy sandy gravel" ~ "Mud_sand_gravel",
    TRUE ~ substratum
  ))
```

Then replace long values with abbreviated values in substratum dataframe:

```
substratum <- left_join(substratum, substrate_values, by = "substratum") %>%
  select(-substratum)
```

We can remove uninformative records (no info or no preference):

```
substratum <- substratum %>% filter(!(substratum_code %in% c("No_Info", "No_preference")))
```

Now create a wide version of this data (one species per row, each substrate forms a column) - do this for each trait:

```
substratum_wide <- substratum %>%
  count(SpeciesName, substratum_code) %>%
  pivot_wider(names_from = substratum_code, values_from = n)

env_pos_wide <- env_pos %>%
  count(SpeciesName, envpos) %>%
  pivot_wider(names_from = envpos, values_from = n)

feed_method_wide <- feed_method %>%
  count(SpeciesName, feedingmethod) %>%
  pivot_wider(names_from = feedingmethod, values_from = n)

habit_wide <- habit %>%
  count(SpeciesName, Habit) %>%
  pivot_wider(names_from = Habit, values_from = n)
```

Use these to get a full species list:

```
biotic_species <- unique(c(
  pull(env_pos_wide, SpeciesName),
  pull(feed_method_wide, SpeciesName),
  pull(habit_wide, SpeciesName),
  pull(substratum_wide, SpeciesName)
))
```

To match the species names to the WoRMS taxonomy, we could use the `worrms` package:

```
biotic_aphias <- wm_name2id_(biotic_species)
biotic_aphias <- biotic_aphias %>% enframe() %>% mutate(value = unlist(value))
```

However, a significant proportion do not result in a match, so instead we use the WoRMS taxon match tool at http://www.marinespecies.org/aphia.php?p=match - first writing the species list to file,

```
write_csv(tibble(name = biotic_species), here::here("data", "derived_data/biotic_aphias_to_check.txt"))
```

Then reading in the matched records:

```
biotic_aphias_matched <- read_tsv(here::here("data", "derived_data/biotic_aphias_to_check_matched.txt"))
```

Next remove unmatched species, plus any at rank above species

```
biotic_aphias_matched <- biotic_aphias_matched %>%
  filter(!is.na(AphiaID_accepted) & !is.na(Species)) %>%
  select(-c(Subgenus, Species, Subspecies))
```

Now add functional group, using functional group matching function in `get_worms_fgrp.R` - load this function first (also available from https://github.com/tomjwebb/WoRMS-functional-groups):

```
source(here::here("scripts", "get_worms_fgrp.R"))
```

Then run it:

```
spp_attr <- biotic_aphias_matched %>%
  mutate(aphia = AphiaID_accepted) %>%
  group_by(AphiaID_accepted) %>%
  group_map(~ get_worms_fgrp(AphiaID = .x$aphia)) %>%
  bind_rows()
```

Do a quick check of species with two adult classifications:

```
spp_attr %>% filter(!is.na(adult_2))
```

```
## # A tibble: 32 x 12
##    AphiaID adult      larva       adult_2      zoea  zoea_2 polyp medusa polyp_2
##      <dbl> <chr>      <chr>       <chr>        <chr> <chr>  <chr> <chr>  <chr>
##  1  102202 epibenthos <NA>        macrobenthos <NA>  <NA>   <NA>  <NA>   <NA>
##  2  103552 benthos    zooplankton macrobenthos <NA>  <NA>   <NA>  <NA>   <NA>
##  3  103579 benthos    zooplankton macrobenthos <NA>  <NA>   <NA>  <NA>   <NA>
##  4  103719 benthos    zooplankton macrobenthos <NA>  <NA>   <NA>  <NA>   <NA>
##  5  103732 benthos    zooplankton macrobenthos <NA>  <NA>   <NA>  <NA>   <NA>
##  6  103743 benthos    zooplankton macrobenthos <NA>  <NA>   <NA>  <NA>   <NA>
##  7  103788 benthos    zooplankton macrobenthos <NA>  <NA>   <NA>  <NA>   <NA>
##  8  103862 benthos    zooplankton macrobenthos <NA>  <NA>   <NA>  <NA>   <NA>
##  9  103882 benthos    zooplankton macrobenthos <NA>  <NA>   <NA>  <NA>   <NA>
## 10  103890 benthos    zooplankton macrobenthos <NA>  <NA>   <NA>  <NA>   <NA>
## # ... with 22 more rows, and 3 more variables: larva > planula <chr>,
## #   larva > planula_2 <chr>, <NA> <chr>
```

And any that have two larval classifications:

```
spp_attr %>% filter(!is.na(`larva > planula_2`))
```

```
## # A tibble: 2 x 12
##   AphiaID adult larva adult_2 zoea  zoea_2 polyp medusa polyp_2 `larva > planul~
##     <dbl> <chr> <chr> <chr>   <chr> <chr>  <chr> <chr>  <chr>   <chr>
## 1  135144 macr~ <NA>  <NA>    <NA>  <NA>   <NA>  <NA>   <NA>    zooplankton
## 2  135193 macr~ <NA>  <NA>    <NA>  <NA>   <NA>  <NA>   <NA>    zooplankton
## # ... with 2 more variables: larva > planula_2 <chr>, <NA> <chr>
```

Or both larva and `larva > planula`:

```
spp_attr %>% filter(!is.na(larva) & !is.na(`larva > planula`))
```

```
## # A tibble: 0 x 12
## # ... with 12 variables: AphiaID <dbl>, adult <chr>, larva <chr>,
## #   adult_2 <chr>, zoea <chr>, zoea_2 <chr>, polyp <chr>, medusa <chr>,
```

```
## #   polyp_2 <chr>, larva > planula <chr>, larva > planula_2 <chr>, <NA> <chr>
```

Or both polyp and adult:

```
spp_attr %>% filter(!is.na(polyp) & !is.na(adult))
```

```
## # A tibble: 0 x 12
## # ... with 12 variables: AphiaID <dbl>, adult <chr>, larva <chr>,
## #   adult_2 <chr>, zoea <chr>, zoea_2 <chr>, polyp <chr>, medusa <chr>,
## #   polyp_2 <chr>, larva > planula <chr>, larva > planula_2 <chr>, <NA> <chr>
```

How many species (if any) have juvenile data?

```
if("juvenile" %in% names(spp_attr)){spp_attr %>% filter(!is.na(juvenile))}
```

Given this we can create a simplified version of just adult and larval functional group:

```
spp_attr_simple <- spp_attr %>%
  mutate(larva = ifelse(is.na(larva), `larva > planula`, larva),
         adult = ifelse(is.na(adult), polyp, adult)) %>%
  select(AphiaID, adult, larva)
```

Next step, get an OBIS checklist for each species - this will show how many (global) occurrence records there are for species:

```
spp_obis <- checklist(taxonid = biotic_aphias_matched$AphiaID_accepted) %>%
  as_tibble()
```

Note - this returns AphiaIDs that are not in our biotic Aphias dataset:

```
spp_obis %>% filter(!(taxonID %in% biotic_aphias_matched$AphiaID_accepted))
```

```
## # A tibble: 50 x 72
##    scientificName         scientificNameAutho~ taxonID bold_id ncbi_id taxonRank
##    <chr>                  <chr>                  <int>   <int>   <int> <chr>
##  1 Varicorbula gibba      (Olivi, 1792)         378492      NA  228466 Species
##  2 Pseudosyllis brevipen~ Grube, 1863           131373      NA 1986320 Species
##  3 Cylista elegans        (Dalyell, 1848)      1471945      NA      NA Species
##  4 Cylista undata         (Müller, 1778)        855675      NA      NA Species
##  5 Fucus distichus subsp~ (C.Agardh) H.T.Powe~  292672      NA   87147 Subspeci~
##  6 Luidia sarsii sarsii   Duben & Koren in Du~  752125      NA      NA Subspeci~
##  7 Corallina officinalis~ (Decaisne) Kützing,~  550769      NA 1141996 Variety
##  8 Hyalinoecia tubicola ~ McIntosh, 1885        335486      NA 1261052 Subspeci~
##  9 Astropecten irregular~ (Delle Chiaje, 1827)  125200      NA      NA Subspeci~
## 10 <NA>                   <NA>                   423870      NA      NA <NA>
## # ... with 40 more rows, and 66 more variables: taxonomicStatus <chr>,
## #   acceptedNameUsage <chr>, acceptedNameUsageID <int>, is_marine <lgl>,
## #   kingdom <chr>, phylum <chr>, class <chr>, subclass <chr>, order <chr>,
## #   suborder <chr>, superfamily <chr>, family <chr>, genus <chr>,
## #   species <chr>, kingdomid <int>, phylumid <int>, classid <int>,
## #   subclassid <int>, orderid <int>, suborderid <int>, superfamilyid <int>,
## #   familyid <int>, genusid <int>, speciesid <int>, records <int>,
## #   is_brackish <lgl>, is_terrestrial <lgl>, wrims <lgl>, category <chr>,
## #   is_freshwater <lgl>, subphylum <chr>, superclass <chr>, superorder <chr>,
## #   infraorder <chr>, parvorder <chr>, subfamily <chr>, subphylumid <int>,
## #   superclassid <int>, superorderid <int>, infraorderid <int>,
## #   parvorderid <int>, subfamilyid <int>, tribe <chr>, tribeid <int>,
## #   subspecies <chr>, subspeciesid <int>, infraclass <chr>, infraclassid <int>,
## #   section <chr>, subsection <chr>, sectionid <int>, subsectionid <int>,
```

```
## #   subgenus <chr>, subgenusid <int>, subterclass <chr>, subterclassid <int>,
## #   subkingdom <chr>, subkingdomid <int>, infrakingdom <chr>,
## #   infraphylum <chr>, infrakingdomid <int>, infraphylumid <int>,
## #   variety <chr>, varietyid <int>, forma <chr>, formaid <int>
```

All of these are at ranks below species level:

```
spp_obis %>% filter(!(taxonID %in% biotic_aphias_matched$AphiaID_accepted)) %>% count(taxonRank)
```

```
## # A tibble: 5 x 2
##   taxonRank      n
## * <chr>      <int>
## 1 Forma          2
## 2 Species        4
## 3 Subspecies    39
## 4 Variety        4
## 5 <NA>           1
```

They also include rather few total OBIS records, and can just be removed here in the next step, which simplifies to just AphiaID and OBIS records:

```
spp_obis_simple <- spp_obis %>%
  filter(taxonID %in% biotic_aphias_matched$AphiaID_accepted) %>%
  select(taxonID, records) %>%
  rename(AphiaID = taxonID, obis_records = records)
```

Now join the OBIS and the Functional Group data:

```
spp_fg_obis <- full_join(spp_attr_simple, spp_obis_simple, by = "AphiaID")
```

Now add taxonomic info back into substratum data, then add the Functional Group and OBIS info:

```
substratum_full <- substratum_wide %>% left_join(
  select(biotic_aphias_matched, name, AphiaID, AphiaID_accepted, ScientificName_accepted, Kingdom:Family
  by = c("SpeciesName" = "name")) %>%
  filter(Kingdom == "Animalia") %>%
  select(-c(Kingdom)) %>%
  left_join(spp_fg_obis, by = c("AphiaID_accepted" = "AphiaID")) %>%
  select(AphiaID_accepted, ScientificName_accepted, adult, larva, obis_records, everything())
```

Restrict to benthos (broad sense):

```
substratum_full <- substratum_full %>%
  filter(adult %in% c("benthos", "endobenthos", "epibenthos", "macrobenthos"))
```

Add in other traits - first Environmental Position (infanua / epifauna). Create a simple version of `env_pos` first - there are quite a few classifications here:

```
env_pos %>% count(envpos)
```

```
## # A tibble: 13 x 2
##    envpos                      n
##  * <chr>                   <int>
##  1 Demersal                   21
##  2 Epibenthic                 80
##  3 Epifaunal                 105
##  4 Epifloral                  25
##  5 Epilithic                  61
##  6 Epiphytic                  16
```

```
##  7 Epizoic                       10
##  8 Infaunal                     104
##  9 Interstitial                   3
## 10 Lithotomous                    1
## 11 Not researched                 5
## 12 Pelagic                        3
## 13 See additional information     1
```

But most can be simplified to inf/epi. So:

```
env_pos_simple <- env_pos %>%
  left_join(select(biotic_aphias_matched, name, AphiaID_accepted), by = c("SpeciesName" = "name")) %>%
  filter(AphiaID_accepted %in% substratum_full$AphiaID_accepted) %>%
  mutate(envpos_simple = ifelse(str_detect(envpos, "Epi"), "Epifaunal", envpos)) %>%
  count(AphiaID_accepted, envpos_simple) %>%
  mutate(n = 1) %>%
  pivot_wider(names_from = envpos_simple, values_from = n) %>%
  rowwise() %>%
  mutate(n_pos = sum(Epifaunal, Infaunal, Demersal, Interstitial, Lithotomous, na.rm = TRUE)) %>%
  select(AphiaID_accepted, n_pos, everything()) %>%
  ungroup()
```

Check species with multiple classifications:

```
env_pos_simple %>% filter(n_pos > 1)
```

```
## # A tibble: 11 x 8
##    AphiaID_accepted n_pos Epifaunal Infaunal Demersal Pelagic Interstitial
##               <dbl> <dbl>     <dbl>    <dbl>    <dbl>   <dbl>        <dbl>
##  1           100906     2         1        1       NA      NA           NA
##  2           101891     2         1        1       NA      NA           NA
##  3           103220     2         1        1       NA      NA           NA
##  4           107254     2        NA        1        1      NA           NA
##  5           107682     2         1        1       NA      NA           NA
##  6           129914     2         1        1       NA      NA           NA
##  7           130387     2         1        1       NA      NA           NA
##  8           130987     2         1        1       NA      NA           NA
##  9           140266     2         1       NA       NA      NA            1
## 10           140467     2         1        1       NA      NA           NA
## 11           140712     2         1       NA        1      NA           NA
## # ... with 1 more variable: Lithotomous <dbl>
```

For all of these we can use infaunal / epifaunal or 'both'. Quick check of species with only one value, in one of Demersal, Interstitial, Lithotomous:

```
env_pos_simple %>% filter(n_pos == 1 & (Demersal == 1 | Interstitial == 1 | Lithotomous == 1))
```

```
## # A tibble: 3 x 8
##   AphiaID_accepted n_pos Epifaunal Infaunal Demersal Pelagic Interstitial
##              <dbl> <dbl>     <dbl>    <dbl>    <dbl>   <dbl>        <dbl>
## 1           107387     1        NA       NA        1      NA           NA
## 2           138945     1        NA       NA       NA      NA            1
## 3           140770     1        NA       NA       NA      NA           NA
## # ... with 1 more variable: Lithotomous <dbl>
```

107387 is *Liocarcinus depurator*, swimming crab, listed in Biotic as swimmer/crawler/burrower so inf/epi seems appropriate. 138945 is *Caecum armoricum*, a gastropod, interstitial which we will class as infauna. 140770 is

Pholas dactylus, a burrowing mollusc (common piddock), which burrows into soft rock / artificial structures. Hard substrate (bedrock) so classing here as epifauna. Update these, and simplify to infauna/epifauna/both:

```
env_pos_simple <- env_pos_simple %>%
  mutate(Epifaunal = ifelse(AphiaID_accepted %in% c(107387, 138945), 1, Epifaunal),
                            Infaunal = ifelse(AphiaID_accepted %in% c(107387, 140770), 1, Infaunal),
                            inf_epi = case_when(
                              Epifaunal == 1 & Infaunal == 1 ~ "Both",
                              Epifaunal == 1 & is.na(Infaunal) ~ "Epifaunal",
                              Infaunal == 1 & is.na(Epifaunal) ~ "Infaunal"
                            )
                            ) %>%
  select(AphiaID_accepted, inf_epi)
```

Add to substratum data:

```
substratum_full <- substratum_full %>%
  left_join(env_pos_simple, by = "AphiaID_accepted") %>%
  select(AphiaID_accepted:obis_records, inf_epi, everything())
```

Create a simple version of dev_mech (larval development mode). There are quite a few classifications here

```
dev_mech %>% count(devmech)
```

```
## # A tibble: 14 x 2
##    devmech                       n
##  * <chr>                     <int>
##  1 Brooding                     28
##  2 Direct Development           50
##  3 Insufficient information     20
##  4 Lecithotrophic               87
##  5 Not relevant                  4
##  6 Not researched                9
##  7 Oviparous                    37
##  8 Ovoviviparous                 8
##  9 Planktotrophic              168
## 10 Schizotomous                  4
## 11 See additional information    3
## 12 Spores (sexual / asexual)    22
## 13 Viviparous (No Care)          5
## 14 Viviparous (Parental Care)    5
```

Restrict to species in the substratum dataset, and simplify dev mech where possible. NB - viviparity / oviparity tells us nothing about larval dispersal (e.g. *Bugula turbinata* is listed as viviparous, but also has lecithotrophic larvae - see http://www.marlin.ac.uk/biotic/browse.php?sp=4418).

```
dev_mech_simple <- dev_mech %>%
  left_join(select(biotic_aphias_matched, name, AphiaID_accepted), by = c("SpeciesName" = "name")) %>%
  filter(AphiaID_accepted %in% substratum_full$AphiaID_accepted) %>%
  mutate(devmech_simple = case_when(
    devmech %in% c("Brooding", "Direct Development", "Schizotomous") ~ "Direct",
    devmech %in% c("Lecithotrophic", "Planktotrophic") ~ "Planktonic",
    TRUE ~ "Other"
  )) %>%
  count(AphiaID_accepted, devmech_simple) %>%
  mutate(n = 1) %>%
  pivot_wider(names_from = devmech_simple, values_from = n) %>%
```

```
  rowwise() %>%
  mutate(n_devmech = sum(Planktonic, Direct, Other, na.rm = TRUE)) %>%
  select(AphiaID_accepted, n_devmech, everything()) %>%
  ungroup()
```

Check species with planktonic AND direct classifications:

```
dev_mech_simple %>% filter(Planktonic == 1 & Direct == 1)
```

```
## # A tibble: 5 x 5
##   AphiaID_accepted n_devmech Planktonic Direct Other
##              <dbl>     <dbl>      <dbl>  <dbl> <dbl>
## 1           117428         2          1      1    NA
## 2           123970         2          1      1    NA
## 3           124201         2          1      1    NA
## 4           130537         2          1      1    NA
## 5           141639         2          1      1    NA
```

This includes things like *Cordylophora caspia* which have both sexual (planktonic) and asexual (direct) reproduction. For our purposes - what's important is if larvae *might* turn up in a plankton survey, so we want a planktonic yes/no/unknown variable only, and can just join that to substratum data.

```
dev_mech_simple <- dev_mech_simple %>%
  mutate(plank_biotic = case_when(
  Planktonic == 1 ~ "Yes",
  Direct == 1 & is.na(Planktonic) ~ "No",
  TRUE ~ "Unknown"))
```

Add to substratum data:

```
substratum_full <- substratum_full %>%
  left_join(select(dev_mech_simple, AphiaID_accepted, plank_biotic), by = "AphiaID_accepted") %>%
  mutate(plank_larv = case_when(
    larva == "zooplankton" | plank_biotic == "Yes" ~ "Yes",
    plank_biotic == "No" ~ "No",
    TRUE ~ "Unknown")) %>%
  select(AphiaID_accepted, ScientificName_accepted, obis_records, inf_epi, plank_larv,
         Bedrock:Strandline, everything())
```

Some final tidying to remove/rename columns:

```
substratum_full <- substratum_full %>% select(-c(adult, larva, plank_biotic)) %>%
  rename(biotic_SpeciesName = SpeciesName, biotic_AphiaID = AphiaID)
```

Now export this derived dataset, together with the substrate values table:

```
write_csv(substratum_full, here::here("data", "derived_data/benthic_species_substratum_prefs.csv"))
write_csv(substrate_values, here::here("data", "derived_data/substrate_values_key.csv"))
```