Darwin Core Guide

Standardizing Marine Biological Data Working Group

2022-02-17

Contents

Preface

Biological data structures, definitions, measurements, and linkages are neccessarily as diverse as the systems they represent. This presents a real challenge when integrating data across biological research domains such as ecology, oceanography, fisheries, and climate sciences.

Lots of standards exist for use with biological data but navigating them can be difficult for data managers who are new to them. The Earth Science Information Partners (ESIP) Biological Data Standards Cluster developed this primer for managers of biological data to provide a quick, easy resource for navigating a selection of the standards that exist. The goal of the primer is to spread awareness about existing standards and is intended to be shared online and at conferences to increase the adoption of standards for biological data and make them FAIR.

Benson, Abigail; LaScala-Gruenewald, Diana; McGuinn, Robert; Satterthwaite, Erin; Beaulieu, Stace; Biddle, Mathew; et al. (2021): Biological Observation Data Standardization - A Primer for Data Managers. ESIP. Online resource. https://doi.org/10.6084/m9.figshare.16806712.v1

Chapter 1

Introduction

The world of standardizing marine biological data can seem complex for the naive oceanographer, biologist, scientist, or programmer. Transforming and integrating data is about combining the right standards for your desired interoperability with other data types. For example, interoperating fish biology measurements with climate level variables. There are a few concepts necessary to make this possible such as standard data structures, controlled vocabularies and knowledge representations, along with metadata standards to facilitate data discovery. This will permit the inclusion of more data and broader access to better ecosystem based models. Many scientific domains data handling practices are currently being reshaped in light of recent advances in computing power, technology, and data science.

Chapter 2

Applications

Some applications are demonstrated in this chapter.

2.1 Aligning Data to Darwin Core - Sampling Event with Measurement or Fact

Abby Benson
October 8,2019

2.1.1 General information about this notebook

This notebook was created for the IOOS DMAC Code Sprint Biological Data Session The data in this notebook were created specifically as an example and meant solely to be illustrative of the process for aligning data to the biological data standard - Darwin Core. These data should not be considered actually occurrences of species and any measurements are also contrived. This notebook is meant to provide a step by step process for taking original data and aligning it to Darwin Core.

```
library(readr)
library(uuid)
library(dplyr)

MadeUpDataForBiologicalDataTraining <- read_csv("~/OBIS/Reference Documentation/Presentations/IOO
```

```
## Parsed with column specification:
## cols(
##   date = col_character(),
##   lat = col_double(),
##   lon = col_double(),
##   region = col_character(),
##   station = col_double(),
##   transect = col_double(),
##   `scientific name` = col_character(),
##   `percent cover` = col_double(),
##   depth = col_double(),
##   `bottom type` = col_character(),
##   rugosity = col_double(),
##   temperature = col_double()
## )
```

First we need to to decide if we will provide an occurrence only version of the data or a sampling event with measurement or facts version of the data. Occurrence only is easier to create. It's only one file to produce. However, several pieces of information will be left out if we choose that option. If we choose to do sampling event with measurement or fact we'll be able to capture all of the data in the file creating a lossless version. Here we decide to use the sampling event option to include as much information as we can. First let's create the eventID and occurrenceID in the original file so that information can be reused for all necessary files down the line.

```
MadeUpDataForBiologicalDataTraining$eventID <- paste(MadeUpDataForBiologicalDataTraini
                                                     MadeUpDataForBiologicalDataTraini
                                                     MadeUpDataForBiologicalDataTraini
MadeUpDataForBiologicalDataTraining$occurrenceID <- ""
MadeUpDataForBiologicalDataTraining$occurrenceID <- sapply(MadeUpDataForBiologicalData
                                       function(x) UUIDgenerate(use.time = TRUE))
```

We will need to create three separate files to comply with the sampling event format. We'll start with the event file but we only need to include the columns that are relevant to the event file.

```
event <- MadeUpDataForBiologicalDataTraining[c("date", "lat", "lon", "region", "statio
                                               "transect", "depth", "bottom type", "ev
```

Next we need to rename any columns of data that match directly to Darwin Core. We know this based on our crosswalk spreadsheet CrosswalkToDarwin-Core.csv

```
event$decimalLatitude <- event$lat
event$decimalLongitude <- event$lon
```

```
event$minimumDepthInMeters <- event$depth
event$maximumDepthInMeters <- event$depth
event$habitat <- event$`bottom type`
event$island <- event$region
```

Let's see how it looks:

```
head(event, n = 10)
```

```
## # A tibble: 10 x 15
##    date    lat   lon region station transect depth `bottom type` eventID
##    <chr> <dbl> <dbl> <chr>    <dbl>    <dbl> <dbl> <chr>         <chr>
##  1 7/16~  18.3 -64.8 St. J~     250        1    25 shallow reef~ St. Jo~
##  2 7/16~  18.3 -64.8 St. J~     250        1    25 shallow reef~ St. Jo~
##  3 7/16~  18.3 -64.8 St. J~     250        1    25 shallow reef~ St. Jo~
##  4 7/16~  18.3 -64.8 St. J~     250        1    25 shallow reef~ St. Jo~
##  5 7/16~  18.3 -64.8 St. J~     250        2    35 complex back~ St. Jo~
##  6 7/16~  18.3 -64.8 St. J~     250        2    35 complex back~ St. Jo~
##  7 7/16~  18.3 -64.8 St. J~     250        2    35 complex back~ St. Jo~
##  8 7/16~  18.3 -64.8 St. J~     250        2    35 complex back~ St. Jo~
##  9 7/16~  18.3 -64.8 St. J~     250        3    85 deep reef     St. Jo~
## 10 7/16~  18.3 -64.8 St. J~     250        3    85 deep reef     St. Jo~
## # ... with 6 more variables: decimalLatitude <dbl>,
## #   decimalLongitude <dbl>, minimumDepthInMeters <dbl>,
## #   maximumDepthInMeters <dbl>, habitat <chr>, island <chr>
```

We need to convert the date to ISO format

```
event$eventDate <- as.Date(event$date, format = "%m/%d/%Y")
```

We will also have to add any missing required fields

```
event$basisOfRecord <- "HumanObservation"
event$geodeticDatum <- "EPSG:4326 WGS84"
```

Then we'll remove any columns that we no longer need to clean things up a bit.

```
event$date <- NULL
event$lat <- NULL
event$lon <- NULL
event$region <- NULL
event$station <- NULL
event$transect <- NULL
event$depth <- NULL
event$`bottom type` <- NULL
```

We have too many repeating rows of information. We can pare this down using eventID which is a unique identifier for each sampling event in the data- which is six, three transects per site.

```
event <- event[which(!duplicated(event$eventID)),]

head(event, n = 6)
```

```
## # A tibble: 6 x 10
##   eventID decimalLatitude decimalLongitude minimumDepthInM~
##   <chr>             <dbl>            <dbl>            <dbl>
## 1 St. Jo~            18.3            -64.8               25
## 2 St. Jo~            18.3            -64.8               35
## 3 St. Jo~            18.3            -64.8               85
## 4 St. Jo~            18.3            -64.8               28
## 5 St. Jo~            18.3            -64.8               16
## 6 St. Jo~            18.3            -64.8               90
## # ... with 6 more variables: maximumDepthInMeters <dbl>, habitat <chr>,
## #   island <chr>, eventDate <date>, basisOfRecord <chr>,
## #   geodeticDatum <chr>
```

Finally we write out the event file

```
write.csv(event, file="MadeUpData_event.csv", row.names=FALSE, fileEncoding="UTF-8", qu
```

Next we need to create the occurrence file. We start by creating the dataframe.

```
occurrence <- MadeUpDataForBiologicalDataTraining[c("scientific name", "eventID", "occu
```

Then we'll rename the columns that align directly with Darwin Core.

```
occurrence$scientificName <- occurrence$`scientific name`
```

Finally we'll add required information that's missing.

```
occurrence$occurrenceStatus <-  ifelse (occurrence$`percent cover` == 0, "absent", "pre
```

2.1.2 Taxonomic Name Matching

A requirement for OBIS is that all scientific names match to the World Register of Marine Species (WoRMS) and a scientificNameID is included. A scientificNameID looks like this "urn:lsid:marinespecies.org:taxname:275730" with the last digits after the colon being the WoRMS aphia ID. We'll need to go out to WoRMS to grab this information. Create a lookup table of unique scientific names

```
lut_worms <- as.data.frame(unique(occurrence_only$scientificName))
lut_worms$scientificName <- as.character(lut_worms$`unique(occurrence_only$scientificName)`)
lut_worms$`unique(occurrence_only$scientificName)` <- NULL
lut_worms$scientificName <- as.character(lut_worms$scientificName)
```

Add the columns that we can grab information from WoRMS including the required scientificNameID.

```
lut_worms$acceptedname <- ""
lut_worms$acceptedID <- ""
lut_worms$scientificNameID <- ""
lut_worms$kingdom <- ""
lut_worms$phylum <- ""
lut_worms$class <- ""
lut_worms$order <- ""
lut_worms$family <- ""
lut_worms$genus <- ""
lut_worms$scientificNameAuthorship <- ""
lut_worms$taxonRank <- ""
```

Taxonomic lookup using the library taxizesoap

```
for (i in 1:nrow(lut_worms)){
  df <- worms_records(scientific = lut_worms$scientificName[i])
  lut_worms[i,]$scientificNameID <- df$lsid[1]
  lut_worms[i,]$acceptedname <- df$valid_name[1]
  lut_worms[i,]$acceptedID <- df$valid_AphiaID[1]
  lut_worms[i,]$kingdom <- df$kingdom[1]
  lut_worms[i,]$phylum <- df$phylum[1]
  lut_worms[i,]$class <- df$class[1]
  lut_worms[i,]$order <- df$order[1]
  lut_worms[i,]$family <- df$family[1]
  lut_worms[i,]$genus <- df$genus[1]
  lut_worms[i,]$scientificNameAuthorship <- df$authority[1]
  lut_worms[i,]$taxonRank <- df$rank[1]
  message(paste("Looking up information for species:", lut_worms[i,]$scientificName))
}
```

```
## Looking up information for species: Acropora cervicornis
```

```
## Looking up information for species: Madracis auretenra
## Looking up information for species: Mussa angulosa

## Looking up information for species: Siderastrea radians
```

Merge the lookup table of unique scientific names back with the occurrence data.

```
occurrence <- merge(occurrence, lut_worms, by = "scientificName")
```

We're going to remove any unnecessary columns to clean up the file

```
occurrence$`scientific name` <- NULL
occurrence$`percent cover` <- NULL
```

Quick look at what we have before we write out the file

```
head(occurrence, n = 10)
```

```
##          scientificName        eventID
## 1  Acropora cervicornis St. John_250_1
## 2  Acropora cervicornis St. John_250_2
## 3  Acropora cervicornis St. John_250_3
## 4  Acropora cervicornis St. John_356_1
## 5  Acropora cervicornis St. John_356_2
## 6  Acropora cervicornis St. John_356_3
## 7    Madracis auretenra St. John_250_1
## 8    Madracis auretenra St. John_250_2
## 9    Madracis auretenra St. John_250_3
## 10   Madracis auretenra St. John_356_1
##                              occurrenceID occurrenceStatus
## 1  63be8f7e-ea9a-11e9-8649-49c6324f3a06           absent
## 2  63beb687-ea9a-11e9-8649-49c6324f3a06           absent
## 3  63beb68b-ea9a-11e9-8649-49c6324f3a06          present
## 4  63bedd76-ea9a-11e9-8649-49c6324f3a06          present
## 5  63bedd7a-ea9a-11e9-8649-49c6324f3a06          present
## 6  63bedd7e-ea9a-11e9-8649-49c6324f3a06          present
## 7  63beb684-ea9a-11e9-8649-49c6324f3a06          present
## 8  63beb688-ea9a-11e9-8649-49c6324f3a06          present
## 9  63beb68c-ea9a-11e9-8649-49c6324f3a06           absent
## 10 63bedd77-ea9a-11e9-8649-49c6324f3a06          present
##            acceptedname acceptedID
## 1  Acropora cervicornis     206989
## 2  Acropora cervicornis     206989
## 3  Acropora cervicornis     206989
## 4  Acropora cervicornis     206989
## 5  Acropora cervicornis     206989
## 6  Acropora cervicornis     206989
## 7    Madracis auretenra     430664
## 8    Madracis auretenra     430664
## 9    Madracis auretenra     430664
## 10   Madracis auretenra     430664
##                             scientificNameID  kingdom   phylum    class
## 1  urn:lsid:marinespecies.org:taxname:206989 Animalia Cnidaria Anthozoa
## 2  urn:lsid:marinespecies.org:taxname:206989 Animalia Cnidaria Anthozoa
## 3  urn:lsid:marinespecies.org:taxname:206989 Animalia Cnidaria Anthozoa
## 4  urn:lsid:marinespecies.org:taxname:206989 Animalia Cnidaria Anthozoa
## 5  urn:lsid:marinespecies.org:taxname:206989 Animalia Cnidaria Anthozoa
## 6  urn:lsid:marinespecies.org:taxname:206989 Animalia Cnidaria Anthozoa
## 7  urn:lsid:marinespecies.org:taxname:430664 Animalia Cnidaria Anthozoa
## 8  urn:lsid:marinespecies.org:taxname:430664 Animalia Cnidaria Anthozoa
## 9  urn:lsid:marinespecies.org:taxname:430664 Animalia Cnidaria Anthozoa
## 10 urn:lsid:marinespecies.org:taxname:430664 Animalia Cnidaria Anthozoa
##           order         family    genus   scientificNameAuthorship
## 1  Scleractinia    Acroporidae Acropora            (Lamarck, 1816)
## 2  Scleractinia    Acroporidae Acropora            (Lamarck, 1816)
## 3  Scleractinia    Acroporidae Acropora            (Lamarck, 1816)
## 4  Scleractinia    Acroporidae Acropora            (Lamarck, 1816)
## 5  Scleractinia    Acroporidae Acropora            (Lamarck, 1816)
## 6  Scleractinia    Acroporidae Acropora            (Lamarck, 1816)
## 7  Scleractinia Pocilloporidae Madracis Locke, Weil & Coates, 2007
## 8  Scleractinia Pocilloporidae Madracis Locke, Weil & Coates, 2007
## 9  Scleractinia Pocilloporidae Madracis Locke, Weil & Coates, 2007
## 10 Scleractinia Pocilloporidae Madracis Locke, Weil & Coates, 2007
##    taxonRank
## 1    Species
## 2    Species
## 3    Species
## 4    Species
## 5    Species
## 6    Species
## 7    Species
## 8    Species
## 9    Species
## 10   Species
```

Write out the file. All done with occurrence!

```
write.csv(occurrence, file="MadeUpData_Occurrence.csv", row.names=FALSE, fileEncoding="UTF-8", qu
```

The last file we need to create is the measurement or fact file. For this we need to combine all of the measurements or facts that we want to include making sure to include IDs from the BODC NERC vocabulary where possible.

```
temperature <- MadeUpDataForBiologicalDataTraining[c("eventID", "temperature", "date")]
temperature$occurrenceID <- ""
temperature$measurementType <- "temperature"
temperature$measurementTypeID <- "http://vocab.nerc.ac.uk/collection/P25/current/WTEMP/
temperature$measurementValue <- temperature$temperature
temperature$measurementUnit <- "Celsius"
temperature$measurementUnitID <- "http://vocab.nerc.ac.uk/collection/P06/current/UPAA/
temperature$measurementAccuracy <- 3
temperature$measurementDeterminedDate <- as.Date(temperature$date, format = "%m/%d/%Y")
temperature$measurementMethod <- ""
temperature$temperature <- NULL
temperature$date <- NULL
rugosity <- MadeUpDataForBiologicalDataTraining[c("eventID", "rugosity", "date")]
rugosity$occurrenceID <- ""
rugosity$measurementType <- "rugosity"
rugosity$measurementTypeID <- ""
rugosity$measurementValue <- rugosity$rugosity
rugosity$measurementUnit <- ""
rugosity$measurementUnitID <- ""
rugosity$measuremntAccuracy <- ""
rugosity$measurementDeterminedDate <- as.Date(rugosity$date, format = "%m/%d/%Y")
rugosity$measurementMethod <- ""
rugosity$rugosity <- NULL
rugosity$date <- NULL
percentcover <- MadeUpDataForBiologicalDataTraining[c("eventID", "occurrenceID", "perce
percentcover$measurementType <- "Percent Cover"
percentcover$measurementTypeID <- "http://vocab.nerc.ac.uk/collection/P01/current/SDBI
percentcover$measurementValue <- percentcover$`percent cover`
percentcover$measurementUnit <- "Percent/100m^2"
percentcover$measurementUnitID <- ""
percentcover$measuremntAccuracy <- 5
percentcover$measurementDeterminedDate <- as.Date(percentcover$date, format = "%m/%d/%
percentcover$measurementMethod <- ""
percentcover$`percent cover` <- NULL
percentcover$date <- NULL
measurementOrFact <- rbind(temperature, rugosity, percentcover)
```

```
## Error in match.names(clabs, names(xi)): names do not match previous names
```

Let's check to see what it looks like

```
head(measurementOrFact, n = 50)
```

```
## # A tibble: 50 x 9
##    eventID occurrenceID measurementType measurementType~ measurementValue
##    <chr>   <chr>        <chr>           <chr>                       <dbl>
##  1 St. Jo~ ""           temperature     ""                           25.2
##  2 St. Jo~ ""           temperature     ""                           25.2
##  3 St. Jo~ ""           temperature     ""                           25.2
##  4 St. Jo~ ""           temperature     ""                           25.2
##  5 St. Jo~ ""           temperature     ""                           24.8
##  6 St. Jo~ ""           temperature     ""                           24.8
##  7 St. Jo~ ""           temperature     ""                           24.8
##  8 St. Jo~ ""           temperature     ""                           24.8
##  9 St. Jo~ ""           temperature     ""                           23.1
## 10 St. Jo~ ""           temperature     ""                           23.1
## # ... with 40 more rows, and 4 more variables: measurementUnit <chr>,
## #   measurementUnitID <chr>, measurementDeterminedDate <date>,
## #   measurementMethod <chr>
```

```
write.csv(measurementOrFact, file="MadeUpData_mof.csv", row.names=FALSE, fileEncoding="UTF-8", qu
```

2.2 Salmon Ocean Ecology Data

2.2.1 Intro

One of the goals of the Hakai Institute and the Canadian Integrated Ocean Observing System (CIOOS) is to facilitate Open Science and FAIR (findable, accessible, interoperable, reusable) ecological and oceanographic data. In a concerted effort to adopt or establish how best to do that, several Hakai and CIOOS staff attended an International Ocean Observing System (IOOS) Code Sprint in Ann Arbour, Michigan between October 7–11, 2019, to discuss how to implement FAIR data principles for biological data collected in the marine environment.

The Darwin Core is a highly structured data format that standardizes data table relations, vocabularies, and defines field names. The Darwin Core defines three table types: `event`, `occurrence`, and `measurementOrFact`. This intuitively captures the way most ecologists conduct their research. Typically, a survey (event) is conducted and measurements, counts, or observations (collectively measurementOrFacts) are made regarding a specific habitat or species (occurrence).

In the following script I demonstrate how I go about converting a subset of the data collected from the Hakai Institute Juvenile Salmon Program and discuss challenges, solutions, pros and cons, and when and what's worthwhile to convert to Darwin Core.

The conversion of a dataset to Darwin Core is much easier if your data are already tidy (normalized) in which you represent your data in separate tables that reflect the hierarchical and related nature of your observations. If your data are not already in a consistent and structured format, the conversion would likely be very arduos and not intuitive.

2.2.2 event

The first step is to consider what you will define as an event in your data set. I defined the capture of fish using a purse seine net as the `event`. Therefore, each row in the `event` table is one deployment of a seine net and is assigned a unique `eventID`.

My process for conversion was to make a new table called `event` and map the standard Darwin Core column names to pre-existing columns that serve the same purpose in my original `seine_data` table and populate the other required fields.

```
event <- tibble(eventID = survey_seines$seine_id,
                eventDate = date(survey_seines$survey_date),
                decimalLatitude = survey_seines$lat,
                decimalLongitude = survey_seines$long,
                geodeticDatum = "EPSG:4326 WGS84",
                minimumDepthInMeters = 0,
                maximumDepthInMeters = 9, # seine depth is 9 m
                samplingProtocol = "http://dx.doi.org/10.21966/1.566666" # This is the
               )

write_csv(event, here::here("datasets", "hakai_salmon_data", "event.csv"))
```

2.2.3 occurrence

Next you'll want to determine what constitutes an occurrence for your data set. Because each event caputers fish, I consider each fish to be an occurrence. Therefore, the unit of observation (each row) in the occurrence table is a fish. To link each occurence to an event you need to include the `eventID` column for every occurrence so that you know what seine (event) each fish (occurrence) came from. You must also provide a globally unique identifier for each occurrence. I already have a locally unique identifier for each fish in the original `fish_data` table called `ufn`. To make it globally unique I pre-pend the organization and research program metadata to the `ufn` column.

```
#TODO: Include bycatch data as well

## make table long first
seines_total_long <- survey_seines %>%
  select(seine_id, so_total, pi_total, cu_total, co_total, he_total, ck_total) %>%
  pivot_longer(-seine_id, names_to = "scientificName", values_to = "n")

seines_total_long$scientificName <- recode(seines_total_long$scientificName, so_total = "Oncorhyn

seines_taken_long <- survey_seines %>%
  select(seine_id, so_taken, pi_taken, cu_taken, co_taken, he_taken, ck_taken) %>%
  pivot_longer(-seine_id, names_to = "scientificName", values_to = "n_taken")

seines_taken_long$scientificName <- recode(seines_taken_long$scientificName, so_taken = "Oncorhyn

## remove records that have already been assigned an ID
seines_long <-  full_join(seines_total_long, seines_taken_long, by = c("seine_id", "scientificNam
  drop_na() %>%
  mutate(n_not_taken = n - n_taken) %>% #so_total includes the number taken so I subtract n_taken
  select(-n_taken, -n) %>%
  filter(n_not_taken > 0)

all_fish_caught <-
  seines_long[rep(seq.int(1, nrow(seines_long)), seines_long$n_not_taken), 1:3] %>%
  select(-n_not_taken) %>%
  mutate(prefix = "hakai-jsp-",
         suffix = 1:nrow(.),
         occurrenceID = paste0(prefix, suffix)
  ) %>%
  select(-prefix, -suffix)

#

# Change species names to full Scientific names
latin <- fct_recode(fish_data$species, "Oncorhynchus nerka" = "SO", "Oncorhynchus gorbuscha" = "P
  as.character()

fish_retained_data <- fish_data %>%
  mutate(scientificName = latin) %>%
  select(-species) %>%
  mutate(prefix = "hakai-jsp-",
         occurrenceID = paste0(prefix, ufn)) %>%
  select(-semsp_id, -prefix, -ufn, -fork_length_field, -fork_length, -weight, -weight_field)

occurrence <- bind_rows(all_fish_caught, fish_retained_data) %>%
  mutate(basisOfRecord = "HumanObservation",
        occurenceStatus = "present") %>%
  rename(eventID = seine_id)
```

For each occuerence of the six different fish species that I caught I need to match the species name that I provide with the official `scientificName` that is part of the World Register of Marine Species database http://www.marinespecies.org/

```
# I went directly to the WoRMS webite (http://www.marinespecies.org/) to download the 

species_matched <- readxl::read_excel(here::here("datasets", "hakai_salmon_data", "raw

occurrence <- left_join(occurrence, species_matched, by = c("scientificName" = "Scienti
  select(occurrenceID, basisOfRecord, scientificName, eventID, occurrenceStatus = occu

write_csv(occurrence, here::here("datasets", "hakai_salmon_data", "occurrence.csv"))
```

2.2.4 measurementOrFact

To convert all your measurements or facts from your normal format to Darwin Core you essentially need to put all your measurements into one column called measurementType and a corresponding column called MeasurementValue. This standardizes the column names are in the `measurementOrFact` table. There are a number of predefined `measurementType`s listed on the NERC database that should be used where possible. I found it difficult to navigate this page to find the correct `measurementType`.

Here I convert length, and weight measurements that relate to an event and an occurrence and call those `measurementTypes` as `length` and `weight`.

```
fish_data$weight <- coalesce(fish_data$weight, fish_data$weight_field)
fish_data$fork_length <- coalesce(fish_data$fork_length, fish_data$fork_length_field)

fish_length <- fish_data %>%
  mutate(occurrenceID = paste0("hakai-jsp-", ufn)) %>%
  select(occurrenceID, eventID = seine_id, fork_length, weight) %>%
  mutate(measurementType = "fork length", measurementValue = fork_length) %>%
  select(eventID, occurrenceID, measurementType, measurementValue) %>%
  mutate(measurementUnit = "millimeters",
         measurementUnitID = "http://vocab.nerc.ac.uk/collection/P06/current/UXMM/")

fish_weight <- fish_data %>%
  mutate(occurrenceID = paste0("hakai-jsp-", ufn)) %>%
  select(occurrenceID, eventID = seine_id, fork_length, weight) %>%
  mutate(measurementType = "mass", measurementValue = weight) %>%
  select(eventID, occurrenceID, measurementType, measurementValue) %>%
  mutate(measurementUnit = "grams",
         measurementUnitID = "http://vocab.nerc.ac.uk/collection/P06/current/UGRM/")

measurementOrFact <- bind_rows(fish_length, fish_weight) %>%
  drop_na(measurementValue)

rm(fish_length, fish_weight)

write_csv(measurementOrFact, here::here("datasets", "hakai_salmon_data", "measurementOrFact.csv")
```

2.3 Hakai Seagrass

2.3.1 Setup

This section clears the workspace, checks the working directory, and installs packages (if required) and loads packages, and loads necessary datasets

```
library("knitr")
# Knitr global chunk options
opts_chunk$set(message = FALSE,
               warning = FALSE,
               error   = FALSE)
```

2.3.1.1 Load Data

First load the seagrass density survey data, set variable classes, and have a quick look

```
# Load density data
seagrassDensity <-
  read.csv("raw_data/seagrass_density_survey.csv",
           colClass = "character") %>%
  mutate(date             = ymd(date),
         depth            = as.numeric(depth),
         transect_dist    = factor(transect_dist),
         collected_start  = ymd_hms(collected_start),
         collected_end    = ymd_hms(collected_end),
         density          = as.numeric(density),
         density_msq      = as.numeric(density_msq),
         canopy_height_cm = as.numeric(canopy_height_cm),
         flowering_shoots = as.numeric(flowering_shoots)) %T>%
  glimpse()
```

```
## Rows: 3,031
## Columns: 22
## $ X                 <chr> "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "1~
## $ organization      <chr> "HAKAI", "HAKAI", "HAKAI", "HAKAI", "HAKAI", "HAKAI",~
## $ work_area         <chr> "CALVERT", "CALVERT", "CALVERT", "CALVERT", "CALVERT"~
## $ project           <chr> "MARINEGEO", "MARINEGEO", "MARINEGEO", "MARINEGEO", "~
## $ survey            <chr> "PRUTH_BAY", "PRUTH_BAY", "PRUTH_BAY", "PRUTH_BAY", "~
## $ site_id           <chr> "PRUTH_BAY_INTERIOR4", "PRUTH_BAY_INTERIOR4", "PRUTH_~
## $ date              <date> 2016-05-13, 2016-05-13, 2016-05-13, 2016-05-13, 2016~
## $ sampling_bout     <chr> "4", "4", "4", "4", "4", "4", "4", "6", "6", "6", "6"~
## $ dive_supervisor   <chr> "Zach", "Zach", "Zach", "Zach", "Zach", "Zach", "Zach~
## $ collector         <chr> "Derek", "Derek", "Derek", "Derek", "Derek", "Derek",~
## $ hakai_id          <chr> "2016-05-13_PRUTH_BAY_INTERIOR4_0", "2016-05-13_PRUTH~
## $ sample_type       <chr> "seagrass_density", "seagrass_density", "seagrass_den~
## $ depth             <dbl> 6.0, 6.0, 6.0, 6.0, 5.0, 6.0, 6.0, 9.1, 9.0, 8.9, 9.0~
## $ transect_dist     <fct> 0, 5, 10, 15, 20, 25, 30, 10, 15, 20, 25, 30, 0, 5, 1~
## $ collected_start   <dttm> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ collected_end     <dttm> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ density           <dbl> 13, 10, 18, 22, 16, 31, 9, 5, 6, 6, 6, 3, 13, 30, 23,~
## $ density_msq       <dbl> 208, 160, 288, 352, 256, 496, 144, 80, 96, 96, 96, 48~
## $ canopy_height_cm  <dbl> 60, 63, 80, 54, 55, 50, 63, 85, 80, 90, 95, 75, 60, 6~
## $ flowering_shoots  <dbl> NA, NA, NA, NA, NA, NA, NA, 0, 0, 0, 0, 0, NA, NA, NA~
## $ comments          <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ quality_log       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
```

Next, load the habitat survey data, and same as above, set variable classes as necessary, and have a quick look.

```
# load habitat data, set variable classes, have a quick look
seagrassHabitat <-
  read.csv("raw_data/seagrass_habitat_survey.csv",
           colClasses = "character") %>%
  mutate(date            = ymd(date),
         depth           = as.numeric(depth),
         hakai_id        = str_pad(hakai_id, 5, pad = "0"),
         transect_dist   = factor(transect_dist),
         collected_start = ymd_hms(collected_start),
         collected_end   = ymd_hms(collected_end)) %T>%
  glimpse()
```

```
## Rows: 2,052
## Columns: 28
## $ X                 <chr> "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "1~
## $ organization      <chr> "HAKAI", "HAKAI", "HAKAI", "HAKAI", "HAKAI", "HAKAI",~
## $ work_area         <chr> "CALVERT", "CALVERT", "CALVERT", "CALVERT", "CALVERT"~
## $ project           <chr> "MARINEGEO", "MARINEGEO", "MARINEGEO", "MARINEGEO", "~
## $ survey            <chr> "CHOKED_PASS", "CHOKED_PASS", "CHOKED_PASS", "CHOKED_~
## $ site_id           <chr> "CHOKED_PASS_INTERIOR6", "CHOKED_PASS_INTERIOR6", "CH~
## $ date              <date> 2017-11-22, 2017-11-22, 2017-11-22, 2017-11-22, 2017~
## $ sampling_bout     <chr> "6", "6", "6", "6", "6", "6", "1", "1", "1", "1", "1"~
## $ dive_supervisor   <chr> "gillian", "gillian", "gillian", "gillian", "gillian"~
## $ collector         <chr> "zach", "zach", "zach", "zach", "zach", "zach", "kyle~
## $ hakai_id          <chr> "10883", "2017-11-22_CHOKED_PASS_INTERIOR6_5 - 10", "~
## $ sample_type       <chr> "seagrass_habitat", "seagrass_habitat", "seagrass_hab~
## $ depth             <dbl> 9.2, 9.4, 9.3, 9.0, 9.2, 9.2, 3.4, 3.4, 3.4, 3.4, 3.4~
## $ transect_dist     <fct> 0 - 5, 10-May, 15-Oct, 15 - 20, 20 - 25, 25 - 30, 0 -~
## $ collected_start   <dttm> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ collected_end     <dttm> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ bag_uid           <chr> "10883", NA, NA, "11094", NA, "11182", "7119", NA, "7~
## $ bag_number        <chr> "3557", NA, NA, "3520", NA, "903", "800", NA, "318", ~
## $ density_range     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ substrate         <chr> "sand,shell hash", "sand,shell hash", "sand,shell has~
## $ patchiness        <chr> "< 1", "< 1", "02-Jan", "< 1", "< 1", "< 1", "< 1", "~
## $ adj_habitat_1     <chr> "seagrass", "seagrass", "seagrass", "seagrass", "seag~
## $ adj_habitat_2     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ sample_collected <chr> "TRUE", "FALSE", "FALSE", "TRUE", "FALSE", "TRUE", "T~
## $ vegetation_1      <chr> NA, NA, NA, NA, NA, NA, "des", NA, "des", NA, NA, NA,~
## $ vegetation_2      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ comments          <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ quality_log       <chr> "1: Flowering shoots 0 for entire transects", NA, NA,~
```

Finally, load coordinate data for surveys, and subset necessary variables

```
coordinates <-
  read.csv("raw_data/seagrassCoordinates.csv",
           colClass = c("Point.Name" = "character")) %>%
  select(Point.Name, Decimal.Lat, Decimal.Long) %T>%
  glimpse()
```

```
## Rows: 70
## Columns: 3
## $ Point.Name   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ Decimal.Lat  <dbl> 52.06200, 52.05200, 51.92270, 51.92500, 51.80900, 51.8090~
## $ Decimal.Long <dbl> -128.4120, -128.4030, -128.4648, -128.4540, -128.2360, -1~
```

2.3.1.2 Merge Datasets

Now all the datasets have been loaded, and briefly formatted, we'll join together the habitat and density surveys, and the coordinates for these.

The seagrass density surveys collect data at discrete points (ie. 5 metres) along the transects, while the habitat surveys collect data over sections (ie. 0 - 5 metres) along the transects. In order to fit these two surveys together, we'll narrow the habitat surveys from a range to a point so the locations will match. Based on how the habitat data is collected, the point the habitat survey is applied to will be the distance at the end of the swath (ie. 10-15m will become 15m). To account for no preceeding distance, the 0m distance will use the 0-5m section of the survey.

First, well make the necessary transformations to the habitat dataset.

```r
# Reformat seagrassHabitat to merge with seagrassDensity
## replicate 0 - 5m transect dist to match with 0m in density survey;
## rest of habitat bins can map one to one with density (ie. 5 - 10m -> 10m)
seagrass0tmp <-
  seagrassHabitat %>%
  filter(transect_dist %in% c("0 - 5", "0 - 2.5")) %>%
  mutate(transect_dist = factor(0))

## collapse various levels to match with seagrassDensity transect_dist
seagrassHabitat$transect_dist <-
  fct_collapse(seagrassHabitat$transect_dist,
               "5" = c("0 - 5", "2.5 - 7.5"),
               "10" = c("5 - 10", "7.5 - 12.5"),
               "15" = c("10 - 15", "12.5 - 17.5"),
               "20" = c("15 - 20", "17.5 - 22.5"),
               "25" = c("20 - 25", "22.5 - 27.5"),
               "30" = c("25 - 30", "27.5 - 30"))

## merge seagrass0tmp into seagrassHabitat to account for 0m samples,
## set class for date, datetime variables
seagrassHabitatFull <-
  rbind(seagrass0tmp, seagrassHabitat) %>%
  filter(transect_dist != "0 - 2.5")  %>% # already captured in seagrass0tmp
  droplevels(.)  # remove now unused factor levels
```

With the distances of habitat and density surveys now corresponding, we can now merge these two datasets plus there coordinates together, combine redundant fields, and remove unnecessary fields.

```
# Merge seagrassHabitatFull with seagrassDensity, then coordinates
seagrass <-
  full_join(seagrassHabitatFull, seagrassDensity,
            by = c("organization",
                   "work_area",
                   "project",
                   "survey",
                   "site_id",
                   "date",
                   "transect_dist")) %>%
  # merge hakai_id.x and hakai_id.y into single variable field;
  # use combination of date, site_id, transect_dist, and field uid (hakai_id
  # when present)
  mutate(field_uid = ifelse(sample_collected == TRUE, hakai_id.x, "NA"),
         hakai_id = paste(date, "HAKAI:CALVERT", site_id, transect_dist, sep = ":"),
         # below, aggregate metadata that didn't merge naturally (ie. due to minor
         # differences in watch time or depth gauges)
         dive_supervisor = dive_supervisor.x,
         collected_start = ymd_hms(ifelse(is.na(collected_start.x),
                                          collected_start.y,
                                          collected_start.x)),
         collected_end   = ymd_hms(ifelse(is.na(collected_start.x),
                                          collected_start.y,
                                          collected_start.x)),
         depth_m         = ifelse(is.na(depth.x), depth.y, depth.x),
         sampling_bout   = sampling_bout.x) %>%
  left_join(., coordinates,  # add coordinates
            by = c("site_id" = "Point.Name")) %>%
  select( - c(X.x, X.y, hakai_id.x, hakai_id.y,  # remove unnecessary variables
              dive_supervisor.x, dive_supervisor.y,
              collected_start.x, collected_start.y,
              collected_end.x, collected_end.y,
              depth.x, depth.y,
              sampling_bout.x, sampling_bout.y)) %>%
  mutate(density_msq = as.character(density_msq),
         canopy_height_cm = as.character(canopy_height_cm),
         flowering_shoots = as.character(flowering_shoots),
         depth_m = as.character(depth_m)) %T>%
  glimpse()
```

```
## Rows: 3,743
## Columns: 38
## $ organization     <chr> "HAKAI", "HAKAI", "HAKAI", "HAKAI", "HAKAI", "HAKAI",~
## $ work_area        <chr> "CALVERT", "CALVERT", "CALVERT", "CALVERT", "CALVERT"~
## $ project          <chr> "MARINEGEO", "MARINEGEO", "MARINEGEO", "MARINEGEO", "~
## $ survey          <chr> "CHOKED_PASS", "CHOKED_PASS", "CHOKED_PASS", "PRUTH_B~
## $ site_id         <chr> "CHOKED_PASS_INTERIOR6", "CHOKED_PASS_EDGE1", "CHOKED~
## $ date            <date> 2017-11-22, 2017-05-19, 2017-05-19, 2017-07-03, 2017~
## $ collector.x     <chr> "zach", "kyle", NA, "tanya", "zach", "zach", "zach", ~
## $ sample_type.x   <chr> "seagrass_habitat", "seagrass_habitat", "seagrass_hab~
## $ transect_dist   <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ bag_uid         <chr> "10883", "7119", "7031", "2352", "10255", "10023", "1~
## $ bag_number      <chr> "3557", "800", "301", "324", "3506", "3555", "3534", ~
## $ density_range   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ substrate       <chr> "sand,shell hash", "sand,shell hash", "sand,shell has~
## $ patchiness      <chr> "< 1", "< 1", "< 1", "< 1", "< 1", "05-Apr", "04-Mar"~
## $ adj_habitat_1   <chr> "seagrass", "sand", "standing kelp", "seagrass", "sea~
## $ adj_habitat_2   <chr> NA, NA, NA, NA, NA, NA, "standing kelp", NA, NA, NA, ~
## $ sample_collected <chr> "TRUE", "TRUE", "TRUE", "TRUE", "TRUE", "TRUE", "TRUE~
## $ vegetation_1    <chr> NA, "des", "des", "zm", "des", NA, NA, NA, NA, NA, NA~
## $ vegetation_2    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, "~
## $ comments.x      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ quality_log.x   <chr> "1: Flowering shoots 0 for entire transects", NA, NA,~
## $ collector.y     <chr> "derek", "ondine", "ondine", "derek", "derek", "derek~
## $ sample_type.y   <chr> "seagrass_density", "seagrass_density", "seagrass_den~
## $ density         <dbl> 4, 10, 6, 13, 6, 1, 2, 6, 21, 3, 7, 4, 3, 14, 17, 11,~
## $ density_msq     <chr> "64", "160", "96", "208", "96", "16", "32", "96", "33~
## $ canopy_height_cm <chr> "80", "80", "110", "60", "125", "100", "100", "125", ~
## $ flowering_shoots <chr> "0", NA, NA, NA, NA, NA, NA, "0", NA, NA, NA, "0", NA~
## $ comments.y      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ quality_log.y   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, "~
## $ field_uid       <chr> "10883", "07119", "07031", "02352", "10255", "10023",~
## $ hakai_id        <chr> "2017-11-22:HAKAI:CALVERT:CHOKED_PASS_INTERIOR6:0", "~
## $ dive_supervisor <chr> "gillian", "gillian,gillian.sadlierbrown", "gillian,g~
## $ collected_start <dttm> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ collected_end   <dttm> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ depth_m         <chr> "9.2", "3.4", "4.8", "2.4", "5.3", "5.6", "4.4", "2.5~
## $ sampling_bout   <chr> "6", "1", "3", "5", "5", "3", "5", "2", "1", "2", "6"~
## $ Decimal.Lat     <dbl> 51.67482, 51.67882, 51.67493, 51.64532, 51.67349, 51.~
## $ Decimal.Long    <dbl> -128.1195, -128.1148, -128.1237, -128.1193, -128.1180~
```

2.3.2 Convert Data to Darwin Core - Extended Measurement or Fact format

The Darwin Core ExtendedMeasurementOrFact (eMoF) extension bases records around a core event (rather than occurrence as in standard Darwin Core), allowing for additional measurement variables to be associated with occurrence data.

2.3.2.1 Add Event ID and Occurrence ID variables to dataset

As this dataset will be annually updated, rather than using natural keys (ie. using package::uuid to autogenerate) for event and occurence IDs, here we will use surrogate keys made up of a concatenation of date survey, transect location, observation distance, and sample ID (for occurrenceID, when a sample is present).

```
# create and populate eventID variable
## currently only event is used, but additional surveys and abiotic data
## are associated with parent events that may be included at a later date
seagrass$eventID <- seagrass$hakai_id

# create and populate occurrenceID; combine eventID with transect_dist
# and field_uid
## in the event of <NA> field_uid, no sample was collected, but
## measurements and occurrence are still taken; no further subsamples
## are associated with <NA> field_uids
seagrass$occurrenceID <-
  with(seagrass,
       paste(eventID, transect_dist, field_uid, sep = ":"))
```

2.3.2.2 Create Event, Occurrence, and eMoF tables

Now that we've created eventIDs and occurrenceIDs to connect all the variables together, we can begin to create the Event, Occurrence, and extended Measurement or Fact table necessary for DarwinCore compliant datasets

```
# subset seagrass to create event table
seagrassEvent <-
  seagrass %>%
  distinct %>%  # some duplicates in data stemming from database conflicts
  select(date,
         Decimal.Lat, Decimal.Long, transect_dist,
         depth_m, eventID) %>%
  rename(eventDate                     = date,
         decimalLatitude               = Decimal.Lat,
         decimalLongitude              = Decimal.Long,
         coordinateUncertaintyInMeters = transect_dist,
         minimumDepthInMeters          = depth_m,
         maximumDepthInMeters          = depth_m) %>%
  mutate(geodeticDatum  = "WGS84",
         samplingEffort = "30 metre transect") %T>% glimpse
```

2.3.2.2.1 Event Table

```
## Rows: 3,659
## Columns: 8
## $ eventDate                     <date> 2017-11-22, 2017-05-19, 2017-05-19, 201~
## $ decimalLatitude               <dbl> 51.67482, 51.67882, 51.67493, 51.64532, ~
## $ decimalLongitude              <dbl> -128.1195, -128.1148, -128.1237, -128.11~
## $ coordinateUncertaintyInMeters <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ maximumDepthInMeters          <chr> "9.2", "3.4", "4.8", "2.4", "5.3", "5.6"~
## $ eventID                       <chr> "2017-11-22:HAKAI:CALVERT:CHOKED_PASS_IN~
## $ geodeticDatum                 <chr> "WGS84", "WGS84", "WGS84", "WGS84", "WGS~
## $ samplingEffort                <chr> "30 metre transect", "30 metre transect"~
```

```
# save event table to csv
write.csv(seagrassEvent, "processed_data/hakaiSeagrassDwcEvent.csv")
```

```


# subset seagrass to create occurrence table
seagrassOccurrence <-
  seagrass %>%
  distinct %>%  # some duplicates in data stemming from database conflicts
  select(eventID, occurrenceID) %>%
  mutate(basisOfRecord = "HumanObservation",
         scientificName   = "Zostera subg. Zostera marina",
         occurrenceStatus = "present")

# Taxonomic name matching
# in addition to the above metadata, DarwinCore format requires further
# taxonomic data that can be acquired through the WoRMS register.
## Load taxonomic info, downloaded via WoRMS tool
# zmWorms <-
#   read.delim("raw_data/zmworms_matched.txt",
#              header = TRUE,
#              nrows  = 1)

zmWorms <- wm_record(id = 145795)

# join WoRMS name with seagrassOccurrence create above
seagrassOccurrence <-
  full_join(seagrassOccurrence, zmWorms,
          by = c("scientificName" = "scientificname")) %>%
  select(eventID, occurrenceID, basisOfRecord, scientificName, occurrenceStatus, AphiaID,
         url, authority, status, unacceptreason, taxonRankID, rank,
         valid_AphiaID, valid_name, valid_authority, parentNameUsageID,
         kingdom, phylum, class, order, family, genus, citation, lsid,
         isMarine, match_type, modified) %T>%
  glimpse
```

2.3.2.2.2 Occurrence Table

```
## Rows: 3,659
## Columns: 27
## $ eventID           <chr> "2017-11-22:HAKAI:CALVERT:CHOKED_PASS_INTERIOR6:0", ~
## $ occurrenceID      <chr> "2017-11-22:HAKAI:CALVERT:CHOKED_PASS_INTERIOR6:0:0:~
## $ basisOfRecord     <chr> "HumanObservation", "HumanObservation", "HumanObserv~
## $ scientificName    <chr> "Zostera subg. Zostera marina", "Zostera subg. Zoste~
## $ occurrenceStatus  <chr> "present", "present", "present", "present", "present~
## $ AphiaID           <int> 145795, 145795, 145795, 145795, 145795, 145795, 1457~
## $ url               <chr> "https://www.marinespecies.org/aphia.php?p=taxdetail~
## $ authority         <chr> "Linnaeus, 1753", "Linnaeus, 1753", "Linnaeus, 1753"~
## $ status            <chr> "accepted", "accepted", "accepted", "accepted", "acc~
## $ unacceptreason    <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ taxonRankID       <int> 220, 220, 220, 220, 220, 220, 220, 220, 220, 220, 22~
## $ rank              <chr> "Species", "Species", "Species", "Species", "Species~
## $ valid_AphiaID     <int> 145795, 145795, 145795, 145795, 145795, 145795, 1457~
## $ valid_name        <chr> "Zostera subg. Zostera marina", "Zostera subg. Zoste~
## $ valid_authority   <chr> "Linnaeus, 1753", "Linnaeus, 1753", "Linnaeus, 1753"~
## $ parentNameUsageID <int> 370435, 370435, 370435, 370435, 370435, 370435, 3704~
## $ kingdom           <chr> "Plantae", "Plantae", "Plantae", "Plantae", "Plantae~
## $ phylum            <chr> "Tracheophyta", "Tracheophyta", "Tracheophyta", "Tra~
## $ class             <chr> "Magnoliopsida", "Magnoliopsida", "Magnoliopsida", "~
## $ order             <chr> "Alismatales", "Alismatales", "Alismatales", "Alisma~
## $ family            <chr> "Zosteraceae", "Zosteraceae", "Zosteraceae", "Zoster~
## $ genus             <chr> "Zostera", "Zostera", "Zostera", "Zostera", "Zostera~
## $ citation          <chr> "WoRMS (2022). Zostera subg. Zostera marina Linnaeus~
## $ lsid              <chr> "urn:lsid:marinespecies.org:taxname:145795", "urn:ls~
## $ isMarine          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ match_type        <chr> "exact", "exact", "exact", "exact", "exact", "exact"~
## $ modified          <chr> "2008-12-09T10:03:16.140Z", "2008-12-09T10:03:16.140~
```

```
# save occurrence table to csv
write.csv(seagrassOccurrence, "processed_data/hakaiSeagrassDwcOccurrence.csv")
```

```
seagrassMof <-
  seagrass %>%
  # select variables for eMoF table
  select(date,
         eventID, survey, site_id, transect_dist,
         substrate, patchiness, adj_habitat_1, adj_habitat_2,
         vegetation_1, vegetation_2,
         density_msq, canopy_height_cm, flowering_shoots) %>%
  # split substrate into two variables (currently holds two substrate type in same var
  separate(substrate, sep = ",", into = c("substrate_1", "substrate_2")) %>%
  # change variables names to match NERC database (or to be more descriptive where non
  rename(measurementDeterminedDate   = date,
         SubstrateTypeA              = substrate_1,
         SubstrateTypeB              = substrate_2,
         BarePatchLengthWithinSeagrass = patchiness,
         PrimaryAdjacentHabitat      = adj_habitat_1,
         SecondaryAdjacentHabitat    = adj_habitat_2,
         PrimaryAlgaeSp              = vegetation_1,
         SecondaryAlgaeSp            = vegetation_2,
         BedAbund                    = density_msq,
         CanopyHeight                = canopy_height_cm,
         FloweringBedAbund           = flowering_shoots) %>%
  # reformat variables into DwC MeasurementOrFact format
  # (single values variable, with measurement type, unit, etc. variables)
  pivot_longer( - c(measurementDeterminedDate, eventID, survey, site_id, transect_dist
               names_to = "measurementType",
               values_to = "measurementValue",
               values_ptypes = list(measurementValue = "character")) %>%
  # use measurement type to fill in remainder of variables relating to
  # NERC vocabulary and metadata fields
  mutate(
    measurementTypeID = case_when(
      measurementType == "BedAbund" ~ "http://vocab.nerc.ac.uk/collection/P01/current/
      measurementType == "CanopyHeight" ~ "http://vocab.nerc.ac.uk/collection/P01/curre
      # measurementType == "BarePatchWithinSeagrass" ~ "",
      measurementType == "FloweringBedAbund" ~ "http://vocab.nerc.ac.uk/collection/P01
    measurementUnit = case_when(
      measurementType == "BedAbund" ~ "Number per square metre",
      measurementType == "CanopyHeight" ~ "Centimetres",
      measurementType == "BarePatchhLengthWithinSeagrass" ~ "Metres",
      measurementType == "FloweringBedAbund" ~ "Number per square metre"),
    measurementUnitID = case_when(
      measurementType == "BedAbund" ~ "http://vocab.nerc.ac.uk/collection/P06/current/
      measurementType == "CanopyHeight" ~ "http://vocab.nerc.ac.uk/collection/P06/curre
      measurementType == "BarePatchhLengthWithinSeagrass" ~ "http://vocab.nerc.ac.uk/collection/P
      measurementType == "FloweringBedAbund" ~ "http://vocab.nerc.ac.uk/collection/P06/current/UP
    measurementAccuracy = case_when(
      measurementType == "CanopyHeight" ~ 5),
    measurementMethod = case_when(
      measurementType == "BedAbund" ~ "25cmx25cm quadrat count",
      measurementType == "CanopyHeight" ~ "in situ with ruler",
      measurementType == "BarePatchhLengthWithinSeagrass" ~ "estimated along transect line",
      measurementType == "FloweringBedAbund" ~ "25cmx25cm quadrat count")) %>%
  select(eventID, measurementDeterminedDate, measurementType, measurementValue,
         measurementTypeID, measurementUnit, measurementUnitID, measurementAccuracy,
         measurementMethod) %T>%
#  select(!c(survey, site_id, transect_dist)) %T>%
  glimpse()
```

2.3.2.2.3 Extended MeasurementOrFact table

```
## Rows: 37,430
## Columns: 9
## $ eventID                   <chr> "2017-11-22:HAKAI:CALVERT:CHOKED_PASS_INTERI~
## $ measurementDeterminedDate <date> 2017-11-22, 2017-11-22, 2017-11-22, 2017-11~
## $ measurementType           <chr> "SubstrateTypeA", "SubstrateTypeB", "BarePat~
## $ measurementValue          <chr> "sand", "shell hash", "< 1", "seagrass", NA,~
## $ measurementTypeID         <chr> NA, NA, NA, NA, NA, NA, NA, "http://vocab.ne~
## $ measurementUnit           <chr> NA, NA, NA, NA, NA, NA, NA, "Number per squa~
## $ measurementUnitID         <chr> NA, NA, NA, NA, NA, NA, NA, "http://vocab.ne~
## $ measurementAccuracy       <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 5, NA, NA, N~
## $ measurementMethod         <chr> NA, NA, NA, NA, NA, NA, NA, "25cmx25cm quadr~
```

```
# save eMoF table to csv
write.csv(seagrassMof, "processed_data/hakaiSeagrassDwcEmof.csv")
```

2.3.3 Session Info

Print session information below in case necessary for future reference

```
# Print Session Info for future reference
sessionInfo()
```

```
## R version 4.1.1 (2021-08-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
```

```
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] worrms_0.4.2    magrittr_2.0.2  knitr_1.37      lubridate_1.8.0
##  [5] here_1.0.1      forcats_0.5.1   stringr_1.4.0   dplyr_1.0.8
##  [9] purrr_0.3.4     readr_2.1.2     tidyr_1.2.0     tibble_3.1.6
## [13] ggplot2_3.3.5   tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.8       assertthat_0.2.1 rprojroot_2.0.2  digest_0.6.29
##  [5] utf8_1.2.2       R6_2.5.1         cellranger_1.1.0 backports_1.4.1
##  [9] reprex_2.0.1     evaluate_0.14    httr_1.4.2       pillar_1.7.0
## [13] rlang_1.0.1      curl_4.3.2       readxl_1.3.1     rstudioapi_0.13
## [17] jquerylib_0.1.4  rmarkdown_2.11   urltools_1.7.3   triebeard_0.3.0
## [21] bit_4.0.4        munsell_0.5.0    broom_0.7.12     compiler_4.1.1
## [25] modelr_0.1.8     xfun_0.29        pkgconfig_2.0.3  htmltools_0.5.2
## [29] tidyselect_1.1.1 httpcode_0.3.0   bookdown_0.24    fansi_1.0.2
## [33] crayon_1.5.0     tzdb_0.2.0       dbplyr_2.1.1     withr_2.4.3
## [37] crul_1.2.0       grid_4.1.1       jsonlite_1.7.3   gtable_0.3.0
## [41] lifecycle_1.0.1  DBI_1.1.2        scales_1.1.1     cli_3.2.0
## [45] stringi_1.7.6    vroom_1.5.7      fs_1.5.2         xml2_1.3.3
## [49] ellipsis_0.3.2   generics_0.1.2   vctrs_0.3.8      tools_4.1.1
## [53] bit64_4.0.5      glue_1.6.1       hms_1.1.1        parallel_4.1.1
## [57] fastmap_1.1.0    yaml_2.2.2       colorspace_2.0-2 rvest_1.0.2
## [61] haven_2.4.3
```

2.4 Trawl Data

One of the more common datasets that can be standardized to Darwin Core and integrated within OBIS is catch data from e.g. a trawl sampling event, or a zooplankton net tow. Of special concern here are datasets that include both a total (species-specific) catch weight, in addition to individual measurements (for a subset of the overall data). In this case, through our standardization to Darwin Core, we want to ensure that data users understand that the individual measurements are a part of, or subset of, the overall (species-specific) record, whilst at the same time ensure that data providers are not duplicating occurrence records to OBIS.

The GitHub issue related to application is can be found here

2.4.1 Workflow Overview

In our current setup, this relationship between the overall catch data and subsetted information is provided in the resourceRelationship extension. This extension *cannot* currently be harvested by GBIF. The required terms for this extension are `resourceID`, `relatedResourceID`, `resourceRelationshipID` and `relationshipOfResource`. The `relatedResourceID` here refers to the *object* of the relationship, whereas the `resourceID` refers to the *subject* of the relationship:

- resourceRelationshipID: a unique identifier for the relationship between one resource (the subject) and another (relatedResource, object).
- resourceID: a unique identifier for the resource that is the subject of the relationship.
- relatedResourceID: a unique identifier for the resource that is the object of the relationship.
- relationshipOfResource: The relationship of the subject (identified by the resourceID) to the object (relatedResourceID). The relationshipOfResource is a free text field.

A few resources have been published to OBIS that contain the resourceRelationship extension (examples). Here, I'll lay out the process and coding used for the Trawl Catch and Species Abundance from the 2019 Gulf of Alaska International Year of the Salmon Expedition. In the following code chunks some details are omitted to improve the readability - the overall code to standardize the catch data can be found here. This dataset includes species-specific total catch data at multiple stations (sampling events). From each catch, individual measurements were also taken. Depending on the number of individual caught in the trawl, this was either the total number of species individuals caught, or only a subset (in case of large numbers of individuals caught).

In this specific data record, we created a single Event Core with three extensions: an *occurrence* extension, *measurement or fact* extension, and the *resourceRelationship* extension. However, in this walk-through I'll only touch on the Event Core, occurrence extension and resourceRelationship extension.

The trawl data is part of a larger project collecting various data types related to salmon ocean ecology. Therefore, in our Event Core we nested information related to the sampling event in the specific layer. (include a visual representation of the schema). Prior to creating the Event Core, we ensured that e.g. dates and times followed the correct ISO-8601 standards, and converted to the correct time zone.

```
# Time is recorded numerically (1037 instead of 10:37), so need to change these column
trawl2019$END_DEPLOYMENT_TIME <- substr(as.POSIXct(sprintf("%04.0f", trawl2019$END_DEP
trawl2019$BEGIN_RETRIEVAL_TIME <- substr(as.POSIXct(sprintf("%04.0f", trawl2019$BEGIN_
# Additionally, the vessel time is recorded in 'Vladivostok' according to the metadata
trawl2019 <- trawl2019 %>%
  mutate(eventDate_start = format_iso_8601(as.POSIXct(paste(EVENT_DATE_START, END_DEPL
                                                     tz = "Asia/Vladivostok")),
         eventDate_start = str_replace(eventDate_start, "\\+00:00", "Z"),
         eventDate_finish = format_iso_8601(as.POSIXct(paste(EVENT_DATE_FINISH, BEGIN_
                                                      tz = "Asia/Vladivostok")),
         eventDate_finish = str_replace(eventDate_finish, "\\+00:00", "Z"),
         eventDate = paste(eventDate_start, eventDate_finish, sep = "/"),
         project = "IYS",
         cruise = paste(project, "GoA2019", sep = ":"),
         station = paste(cruise, TOW_NUMBER, sep=":Stn"),
         trawl = paste(station, "trawl", sep=":"))
```

Then we created the various layers of our Event Core. We created these layers/data frames from two separate datasets that data are pulled from - one dataset that contains the *overall* catch data, and one dataset that contains the *specimen* data:

```
trawl2019_allCatch <- read_excel(here("Trawl", "2019", "raw_data",
                                      "2019_GoA_Fish_Trawl_catchdata.xlsx"), sheet = "
  mutate(project = "IYS",
         cruise = paste(project, "GoA2019", sep = ":"),
         station = paste(cruise, `TOW_NUMBER (number)`, sep = ":Stn"),
         trawl = paste(station, "trawl", sep = ":"))

trawl2019_specimen <- read_excel(here("Trawl", "2019", "raw_data", "2019_GoA_Fish_Spec
                                 sheet = "SPECIMEN_FINAL") %>%
  mutate(project = "IYS",
         cruise = paste(project, "GoA2019", sep = ":"),
         station = paste(cruise, TOW_NUMBER, sep = ":Stn"),
         trawl = paste(station, "trawl", sep = ":"),
         sample = paste(trawl, "sample", sep = ":"),
         sample = paste(sample, row_number(), sep = ""))
```

Next we created the Event Core, ensuring that we connect the data to the right layer (i.e. date and time should be connected to the layer associated with the sampling event). Please note that because we are creating multiple layers and nesting information, and then at a later stage combining different tables, this results in cells being populated with `NA`. These have to be removed prior to publishing the Event Core through the IPT.

```
trawl2019_project <- trawl2019 %>%
  select(eventID = project) %>%
  distinct(eventID) %>%
  mutate(type = "project")

trawl2019_cruise <- trawl2019 %>%
  select(eventID = cruise,
         parentEventID = project) %>%
  distinct(eventID, .keep_all = TRUE) %>%
  mutate(type = "cruise")

trawl2019_station <- trawl2019 %>%
  select(eventID = station,
         parentEventID = cruise) %>%
  distinct(eventID, .keep_all = TRUE) %>%
  mutate(type = "station")

# The coordinates associated to the trawl need to be presented in a LINESTRING.
# END_LONGITUDE_DD needs to be inverted (has to be between -180 and 180, inclusive).
trawl2019_coordinates <- trawl2019 %>%
  select(eventID = trawl,
         START_LATITUDE_DD,
         longitude,
         END_LATITUDE_DD,
         END_LONGITUDE_DD) %>%
  mutate(END_LONGITUDE_DD = END_LONGITUDE_DD * -1,
         footprintWKT = paste("LINESTRING (", longitude, START_LATITUDE_DD, ",",
                              END_LONGITUDE_DD, END_LATITUDE_DD, ")"))
trawl2019_linestring <- obistools::calculate_centroid(trawl2019_coordinates$footprintWKT)
trawl2019_linestring <- cbind(trawl2019_coordinates, trawl2019_linestring) %>%
  select(eventID, footprintWKT, decimalLatitude, decimalLongitude, coordinateUncertaintyInMeters)

trawl2019_trawl <- trawl2019 %>%
  select(eventID = trawl,
         parentEventID = station,
         eventDate,
         year,
         month,
         day) %>%
  mutate(minimumDepthInMeters = 0, # headrope was at the surface
         maximumDepthInMeters = trawl2019$MOUTH_OPENING_HEIGHT,
         samplingProtocol = "midwater trawl", # when available add DOI to paper here
         locality = case_when(
           trawl2019$EVENT_SUB_TYPE == "Can EEZ" ~ "Canadian EEZ"),
         locationID = case_when(
           trawl2019$EVENT_SUB_TYPE == "Can EEZ" ~ "http://marineregions.org/mrgid/849
  left_join(trawl2019_linestring, by = "eventID") %>%
  distinct(eventID, .keep_all = TRUE) %>%
    mutate(type = "midwater trawl")

trawl2019_sample <- trawl2019_specimen %>%
  select(eventID = sample,
         parentEventID = trawl) %>%
  distinct(eventID, .keep_all = TRUE) %>%
  mutate(type = "individual sample")

trawl2019_event <- bind_rows(trawl2019_project,
                             trawl2019_cruise,
                             trawl2019_station,
                             trawl2019_trawl,
                             trawl2019_sample)

# Remove NAs from the Event Core:
trawl2019_event <- sapply(trawl2019_event, as.character)
trawl2019_event[is.na(trawl2019_event)] <- ""
trawl2019_event <- as.data.frame(trawl2019_event)
```

TO DO: Add visual of e.g. the top 10 rows of the Event Core.

Now that we created the Event Core, we create the occurrence extension. To do this, we create two separate occurrence data tables: one that includes the occurrence data for the *total* catch, and one data table for the *specimen* data. Finally, the Occurrence extension is created by combining these two data frames. Personally, I prefer to re-order it so it makes visual sense to me (nest the specimen occurrence records under their respective overall catch data).

```
trawl2019_allCatch_worms <- worrms::wm_records_names(unique(trawl2019_allCatch$scienti
trawl2019_occ <- left_join(trawl2019_allCatch, trawl2019_allCatch_worms, by = "scienti
  rename(eventID = trawl,
         specificEpithet = species,
         scientificNameAuthorship = authority,
         taxonomicStatus = status,
         taxonRank = rank,
         scientificName = scientificname,
         scientificNameID = lsid,
         individualCount = `CATCH_COUNT (pieces)(**includes Russian expansion for some
         occurrenceRemarks = COMMENTS) %>%
  mutate(occurrenceID = paste(eventID, "occ", sep = ":"),
         occurrenceID = paste(occurrenceID, row_number(), sep = ":"),
         occurrenceStatus = "present",
        sex = "")

trawl2019_catch_ind_worms <- worrms::wm_records_names(unique(trawl2019_catch_ind$scientificname))
trawl2019_catch_ind_occ <- left_join(trawl2019_catch_ind, trawl2019_catch_ind_worms, by = "scient
  rename(scientificNameAuthorship = authority,
         taxonomicStatus = status,
         taxonRank = rank,
         scientificName = scientificname,
         scientificNameID = lsid) %>%
  mutate(occurrenceID = paste(eventID, "occ", sep = ":"),
         occurrenceStatus = "present",
         individualCount = 1)

# Combine the two occurrence data frames:
trawl2019_occ_ext <- dplyr::bind_rows(trawl2019_occ_fnl, trawl2019_catch_ind_fnl)

# To re-order the occurrenceID, use following code:
order <- stringr::str_sort(trawl2019_occ_ext$occurrenceID, numeric=TRUE)
trawl2019_occ_ext <- trawl2019_occ_ext[match(order, trawl2019_occ_ext$occurrenceID),] %>%
  mutate(basisOfRecord = "HumanObservation")
```

TO DO: Add visual of e.g. the top 10 rows of the Occurrence extension.

Please note that in the *overall* species-specific occurrence data frame, *individualCount* was not included. This term should not be used for abundance studies, but to avoid confusion and the appearance that the specimen records are an additional observation on top of the overall catch record, the *individualCount* term was left blank for the overall catch data.

A resource relationship extension is created to further highlight that the individual samples in the occurrence extension are part of a larger overall catch that was also listed in the occurrence extension. In this extension, we wanted to make sure to highlight that the *specimen* occurrence records are *a subset of* the *overall* catch data through the field `relationshipOfResource1`. Each of these relationships gets a unique `resourceRelationshipID`.

```
trawl_resourceRelationship <- trawl2019_occ_ext %>%
  select(eventID, occurrenceID, scientificName) %>%
  mutate(resourceID = ifelse(grepl("sample", trawl2019_occ_ext$occurrenceID), trawl2019_occ_ext$o
  mutate(eventID = gsub(":sample.*", "", trawl2019_occ_ext$eventID)) %>%
  group_by(eventID, scientificName) %>%
  filter(n() != 1) %>%
  ungroup()

trawl_resourceRelationship <- trawl_resourceRelationship %>%
  mutate(relatedResourceID = ifelse(grepl("sample", trawl_resourceRelationship$occurren
  mutate(relationshipOfResource = ifelse(!is.na(resourceID), "is a subset of", NA)) %>%
  dplyr::arrange(eventID, scientificName) %>%
  fill(relatedResourceID) %>%
  filter(!is.na(resourceID))

order <- stringr::str_sort(trawl_resourceRelationship$resourceID, numeric = TRUE)
trawl_resourceRelationship <- trawl_resourceRelationship[match(order, trawl_resourceRel

trawl_resourceRelationship <- trawl_resourceRelationship %>%
  mutate(resourceRelationshipID = paste(relatedResourceID, "rr", sep = ":"),
         ID = sprintf("%03d", row_number()),
         resourceRelationshipID = paste(resourceRelationshipID, ID, sep = ":")) %>%
  select(eventID, resourceRelationshipID, resourceID, relationshipOfResource, relatedRe
```

TO DO: Add visual of e.g. the top 10 rows of the ResourceRelationship extension.

2.4.2 FAQ

Q1. Why not use the terms *associatedOccurrence* or *associatedTaxa*? **A**. There seems to be a movement away from the term *associatedOccurrence* as the `resourceRelationship` extension has a much broader use case. Some issues that were raised on GitHub exemplify this, see e.g. here. *associatedTaxa* is used to provide identifiers or names of taxa and the associations of an Occurrence with them. This term is not apt for establishing relationships between taxa, only between specific Occurrences of an organism with other taxa. As noted on the TDWG website, *[...] Note that the ResourceRelationship class is an alternative means of representing associations, and with more detail.* See also e.g. this issue.

Chapter 3

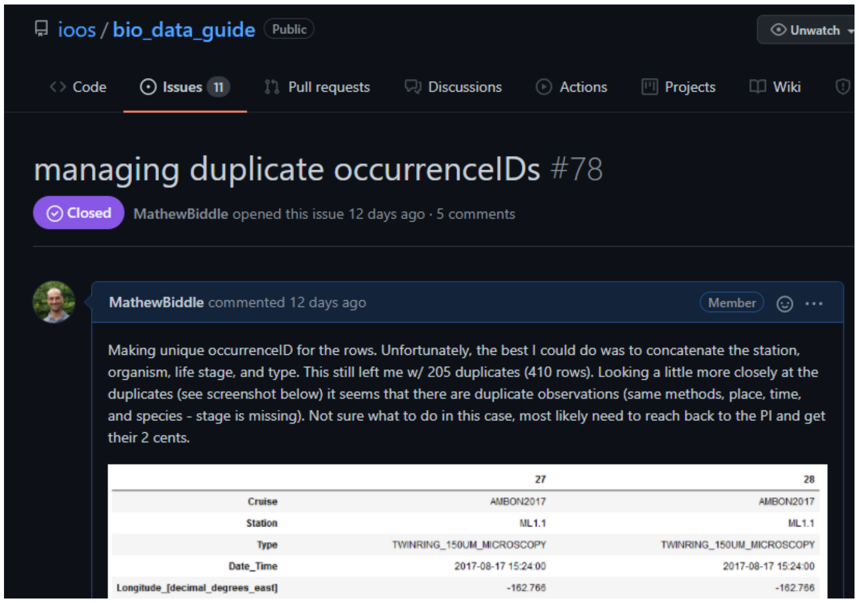
Dealing with errors

Datasets can have a wide variety of errors that pop up during the darwin core alignment process. This chapter details ways in which a data manager can identify, discuss, and resolve potential errors in the data.

It should be noted that, in most cases, the data manager/scientist aligning the data to darwin core should reach out to the data originator to ensure the actions taken are not incorrectly representing the observations.

3.1 Example using GitHub to resolve errors

1. Dataset sent to OBIS-USA via email.
2. OBIS-USA uploaded to IPT.
3. Once the data were uploaded, the IPT identified there was an issue with the `occurrenceID` field. The issue was then presented and discussed in a

GitHub ticket: 

managing duplicate occurrenceIDs #78

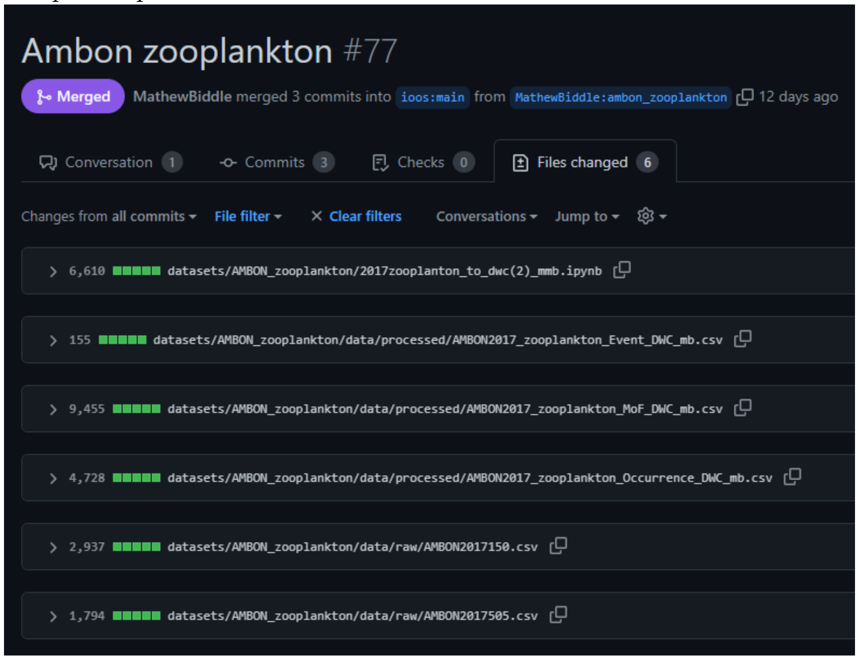
MathewBiddle opened this issue 12 days ago · 5 comments

MathewBiddle commented 12 days ago

Making unique occurrenceID for the rows. Unfortunately, the best I could do was to concatenate the station, organism, life stage, and type. This still left me w/ 205 duplicates (410 rows). Looking a little more closely at the duplicates (see screenshot below) it seems that there are duplicate observations (same methods, place, time, and species - stage is missing). Not sure what to do in this case, most likely need to reach back to the PI and get their 2 cents.

	27	28
Cruise	AMBON2017	AMBON2017
Station	ML1.1	ML1.1
Type	TWINRING_150UM_MICROSCOPY	TWINRING_150UM_MICROSCOPY
Date_Time	2017-08-17 15:24:00	2017-08-17 15:24:00
Longitude_[decimal_degrees_east]	-162.766	-162.766

4. The data manager uploaded the raw data and code to GitHub through the pull request below. This included a fix for the `occurrenceID` issue.

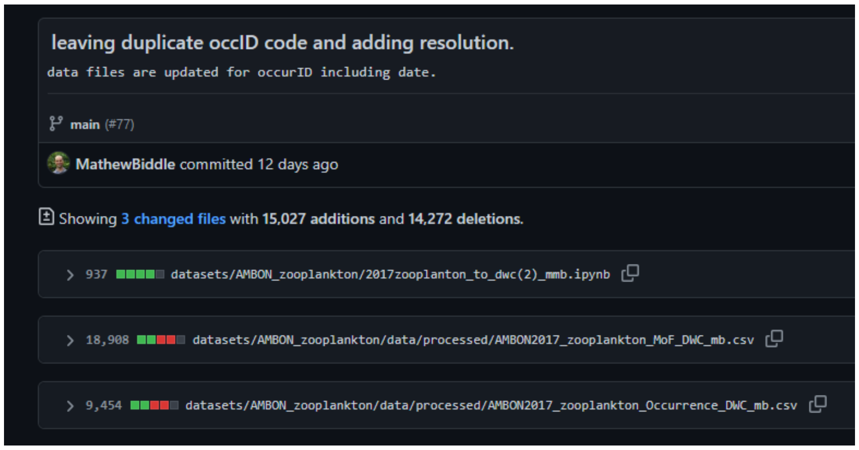


Ambon zooplankton #77

MathewBiddle merged 3 commits into `ioos:main` from `MathewBiddle:ambon_zooplankton` 12 days ago

```
6,610 datasets/AMBON_zooplankton/2017zooplanton_to_dwc(2)_mmb.ipynb
155 datasets/AMBON_zooplankton/data/processed/AMBON2017_zooplankton_Event_DWC_mb.csv
9,455 datasets/AMBON_zooplankton/data/processed/AMBON2017_zooplankton_MoF_DWC_mb.csv
4,728 datasets/AMBON_zooplankton/data/processed/AMBON2017_zooplankton_Occurrence_DWC_mb.csv
2,937 datasets/AMBON_zooplankton/data/raw/AMBON2017150.csv
1,794 datasets/AMBON_zooplankton/data/raw/AMBON2017505.csv
```

5. The OBIS node manager was notified of the availability of a revised dataset by pointing directly to the appropriate commit in GitHub:



leaving duplicate occID code and adding resolution.

```
data files are updated for occurID including date.
```

main (#77)

MathewBiddle committed 12 days ago

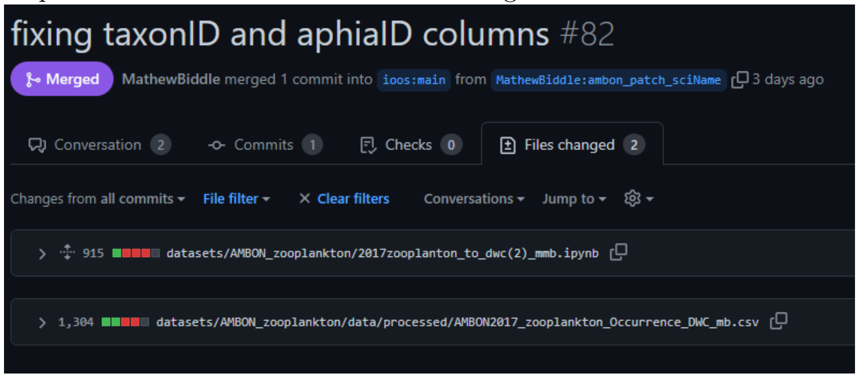
Showing 3 changed files with 15,027 additions and 14,272 deletions.

```
937 datasets/AMBON_zooplankton/2017zooplanton_to_dwc(2)_mmb.ipynb
18,908 datasets/AMBON_zooplankton/data/processed/AMBON2017_zooplankton_MoF_DWC_mb.csv
9,454 datasets/AMBON_zooplankton/data/processed/AMBON2017_zooplankton_Occurrence_DWC_mb.csv
```

6. The OBIS node manager downloaded the data from the commit above and uploaded them to the IPT.
7. The IPT returned a summary of the dataset including that 434 records had invalid `scientificNameID` records in the occurrence file.
8. After some data sleuthing, the data manager noticed that the code accidentally removed trailing zeros from `scientificNameID` that ended in `0`:

```
#taxonid needs to not have trailing .0
taxons = df[['taxonID']].astype('string', errors='ignore')
t=taxons['taxonID'].convert_dtypes()
t=t.str.strip('.0')
t
df['taxonID']=t
df.head()
```

9. So, the data manager updated the code to resolve the issue and generate a new occurrence file.

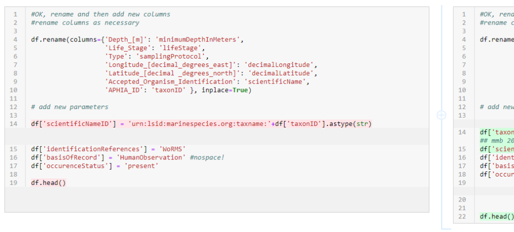


fixing taxonID and aphiaID columns #82

Merged MathewBiddle merged 1 commit into `ioos:main` from `MathewBiddle:ambon_patch_sciName` 3 days ago

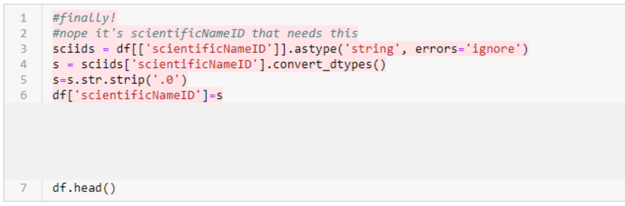
```
915 datasets/AMBON_zooplankton/2017zooplanton_to_dwc(2)_mmb.ipynb
1,304 datasets/AMBON_zooplankton/data/processed/AMBON2017_zooplankton_Occurrence_DWC_mb.csv
```

1. Here is fixing the `scientificNameID` generation:



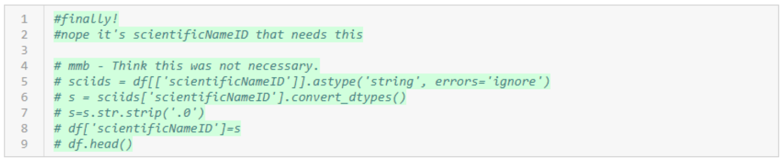
```
#OK, rename and then add new columns
#rename columns as necessary

df.rename(columns={'Depth_[m]': 'minimumDepthInMeters',
                   'Life_Stage': 'lifeStage',
                   'Type': 'samplingProtocol',
                   'Longitude_[decimal_degrees_east]': 'decimalLongitude',
                   'Latitude_[decimal _degrees_north]': 'decimalLatitude',
                   'Accepted_Organism_Identification': 'scientificName',
                   'APHIA_ID': 'taxonID' }, inplace=True)

# add new parameters

df['scientificNameID'] = 'urn:lsid:marinespecies.org:taxname:'+df['taxonID'].astype(str)

df['identificationReferences'] = 'WoRMS'
df['basisOfRecord'] = 'HumanObservation' #nospace!
df['occurenceStatus'] = 'present'

df.head()
```

```
#OK, rena
#rename c

df.rename






# add new

df['taxon
## mmb 20
df['scien
df['ident
df['basis
df['occur



df.head()
```



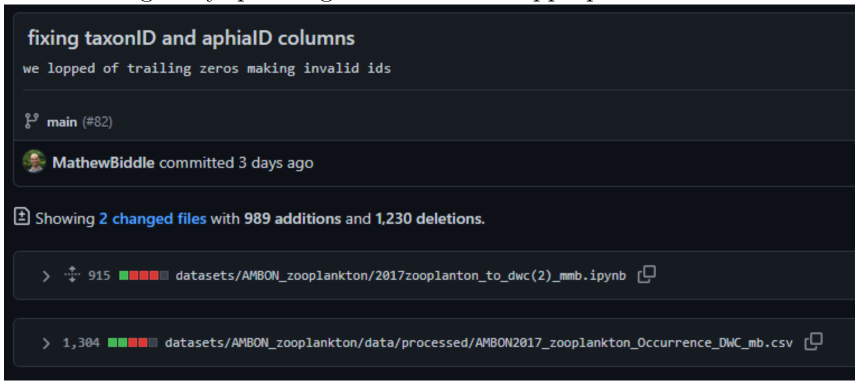
```
#finally!
#nope it's scientificNameID that needs this
sciids = df[['scientificNameID']].astype('string', errors='ignore')
s = sciids['scientificNameID'].convert_dtypes()
s=s.str.strip('.0')
df['scientificNameID']=s

df.head()
```

2. Here is removing the problematic code:



```
#finally!
#nope it's scientificNameID that needs this

# mmb - Think this was not necessary.
# sciids = df[['scientificNameID']].astype('string', errors='ignore')
# s = sciids['scientificNameID'].convert_dtypes()
# s=s.str.strip('.0')
# df['scientificNameID']=s
# df.head()
```

10. The revised occurrence file was then resubmitted to the OBIS node manager by pointing them at the appropriate commit record:



fixing taxonID and aphiaID columns

we lopped of trailing zeros making invalid ids

main (#82)

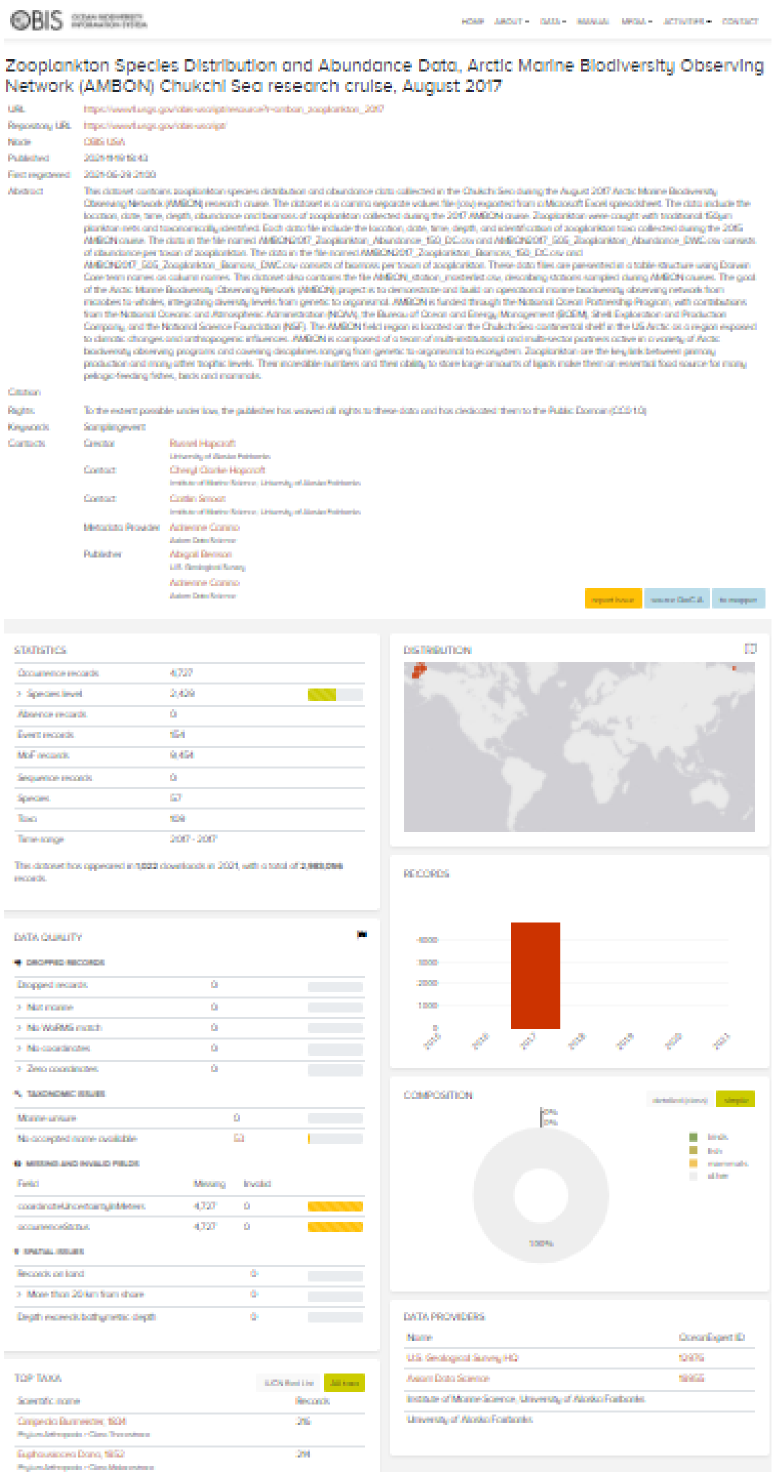
MathewBiddle committed 3 days ago

Showing 2 changed files with 989 additions and 1,230 deletions.

915 🟩🟥🟥🟥▨ datasets/AMBON_zooplankton/2017zooplanton_to_dwc(2)_mmb.ipynb

1,304 🟩🟩🟥🟥▨ datasets/AMBON_zooplankton/data/processed/AMBON2017_zooplankton_Occurrence_DWC_mb.csv

11. The OBIS node manager downloaded the data from the commit above and uploaded them to the IPT.
12. The IPT and OBIS landing page now indicated that no more issues with


OBIS OCEAN BIODIVERSITY INFORMATION SYSTEM
HOME ABOUT DATA MANUAL MEDIA ACTIVITIES CONTACT

Zooplankton Species Distribution and Abundance Data, Arctic Marine Biodiversity Observing Network (AMBON) Chukchi Sea research cruise, August 2017

URL	https://www1.usgs.gov/obis-usa/ipt/resource?r=ambon_zooplankton_2017
Repository URL	https://www1.usgs.gov/obis-usa/ipt/
Node	OBIS USA
Published	2021-11-19 18:43
First registered	2021-06-29 21:30
Abstract	This dataset contains zooplankton species distribution and abundance data collected in the Chukchi Sea during the August 2017 Arctic Marine Biodiversity Observing Network (AMBON) research cruise. The dataset is a comma separate values file (csv) exported from a Microsoft Excel spreadsheet. The data include the location, date, time, depth, abundance and biomass of zooplankton collected during the 2017 AMBON cruise. Zooplankton were caught with traditional 150μm plankton nets and taxonomically identified. Each data file include the location, date, time, depth, and identification of zooplankton taxa collected during the 2015 AMBON cruise. The data in the file named AMBON2017_Zooplankton_Abundance_150_DC.csv and AMBON2017_505_Zooplankton_Abundance_DWC.csv consists of abundance per taxon of zooplankton. The data in the file named AMBON2017_Zooplankton_Biomass_150_DC.csv and AMBON2017_505_Zooplankton_Biomass_DWC.csv consists of biomass per taxon of zooplankton. These data files are presented in a table structure using Darwin Core term names as column names. This dataset also contains the file AMBON_station_masterlist.csv, describing stations sampled during AMBON cruises. The goal of the Arctic Marine Biodiversity Observing Network (AMBON) project is to demonstrate and build an operational marine biodiversity observing network from microbes to whales, integrating diversity levels from genetic to organismal. AMBON is funded through the National Ocean Partnership Program, with contributions from the National Oceanic and Atmospheric Administration (NOAA), the Bureau of Ocean and Energy Management (BOEM), Shell Exploration and Production Company, and the National Science Foundation (NSF). The AMBON field region is located on the Chukchi Sea continental shelf in the US Arctic as a region exposed to climate changes and anthropogenic influences. AMBON is composed of a team of multi-institutional and multi-sector partners active in a variety of Arctic biodiversity observing programs and covering disciplines ranging from genetic to organismal to ecosystem. Zooplankton are the key link between primary production and many other trophic levels. Their incredible numbers and their ability to store large amounts of lipids make them an essential food source for many pelagic-feeding fishes, birds and mammals.
Citation	
Rights	To the extent possible under law, the publisher has waived all rights to these data and has dedicated them to the Public Domain (CC0 1.0)
Keywords	Samplingevent
Contacts	Creator: Russell Hopcroft, University of Alaska Fairbanks
	Contact: Cheryl Clarke-Hopcroft, Institute of Marine Science, University of Alaska Fairbanks
	Contact: Caitlin Smoot, Institute of Marine Science, University of Alaska Fairbanks
	Metadata Provider: Adrienne Canino, Axiom Data Science
	Publisher: Abigail Benson, U.S. Geological Survey
	Adrienne Canino, Axiom Data Science

STATISTICS

Occurrence records	4,727
› Species level	2,409
Absence records	0
Event records	164
MoF records	9,454
Sequence records	0
Species	57
Taxa	109
Time range	2017 - 2017

This dataset has appeared in 1,022 downloads in 2021, with a total of 2,983,096 records.

DISTRIBUTION

RECORDS

DATA QUALITY

DROPPED RECORDS

Dropped records	0
› Not marine	0
› No WoRMS match	0
› No coordinates	0
› Zero coordinates	0

TAXONOMIC ISSUES

Marine unsure	0
No accepted name available	53

MISSING AND INVALID FIELDS

Field	Missing	Invalid
coordinateUncertaintyInMeters	4,727	0
occurrenceStatus	4,727	0

SPATIAL ISSUES

Records on land	0
› More than 20 km from shore	0
Depth exceeds bathymetric depth	0

COMPOSITION

DATA PROVIDERS

Name	OceanExpert ID
U.S. Geological Survey HQ	12975
Axiom Data Science	18955
Institute of Marine Science, University of Alaska Fairbanks	
University of Alaska Fairbanks	

TOP TAXA

Scientific name	Records
Copepoda Burmeister, 1834 Phylum Arthropoda > Class Thecostraca	216
Euphausiacea Dana, 1852 Phylum Arthropoda > Class Malacostraca	214

these data are present:

Chapter 4

Frequently Asked Questions

Q. What data structure does OBIS recommend? A. The OBIS-ENV Darwin Core Archive Data Structure. OBIS manual

Q. What is a controlled vocabulary, why use them? A. There are a number of controlled vocabularies that are used to describe parameters commonly used in specific research domains. Using terms defined in a controlled vocabulary allows for greater interoperability of data sets within the domain, and ideally between domains by ensuring that variables that are the same can be identified.

Q. What controlled vocabularies does OBIS rely on? A. WoRMS, NERC Vocabulary Server inlcuding: * Device categories using the SeaDataNet device categories

- Device make/model using the SeaVoX Device Catalogue
- Platform categories using SeaVoX Platform Categories
- Platform instances using the ICES Platform Codes
- Unit of measure

Q. How can I find out which common measurementTypes are used in measurement or facts tables in existing OBIS datasets? A. See Measurement Types in OBIS

Q. What is an ontology? A. An ontology is a classification system for establishing a hierarchically related set of concepts. Concepts are often terms from controlled vocabularies. Ontologies can include all of the following, but are not required to include them. Classes (general things, types of things) Instances (individual things) Relationships among things Properties of things Functions, processes, constraints, and rules relating to things

Q. What is ERDDAP? A. ERDDAP is a data server. It provides 'easier access to scientific data' by providing a consistent interface that aggregates many disparate data sources. It does this by providing translation services between many common file types for gridded arrarys ('net CDF' files) and tabular data (spreadsheets). Data access is also made easier because it unifies different types of data servers and access protocols.

Q. What metadata profile does OBIS use? A. OBIS uses the GBIF EML profile (version 1.1)

Q. Can Darwin Core be used in the Semantic Web/Resrouce Description Framework? A. See Darwin Core Resource Description Framework Guide and Lessons learned from adapting the Darwin Core vocabulary standard for use in RDF

Chapter 5

Tools

Below are some of the tools and packages used in workflows. R and Python package "Type" is BIO for packages specifically for biological applications, and GEN for generic packages.

5.1 R

Package	Type	Description
bdveRse	BIO	A family of R packages for biodiversity data.
ecocomDP	BIO	Work with the Ecological Community Data Design Pattern. 'ecocomDP' is a flexible data model for harmonizing ecological community surveys, in a research question agnostic format, from source data published across repositories, and with methods that keep the derived data up-to-date as the underlying sources change.

Package	Type	Description
EDIorg/EMLasseblyline	BIO	For scientists and data managers to create high quality EML metadata for dataset publication.
finch	BIO	Parse Darwin Core Files
iobis/obistools	BIO	Tools for data enhancement and quality control.
robis	BIO	R client for the OBIS API
ropensci/EML	BIO	Provides support for the serializing and parsing of all low-level EML concepts
taxize	BIO	Interacts with a suite of web 'APIs' for taxonomic tasks, such as getting database specific taxonomic identifiers, verifying species names, getting taxonomic hierarchies, fetching downstream and upstream taxonomic names, getting taxonomic synonyms, converting scientific to common names and vice versa, and more.
worrms	BIO	Client for World Register of Marine Species. Includes functions for each of the API methods, including searching for names by name, date and common names, searching using external identifiers, fetching synonyms, as well as fetching taxonomic children and taxonomic classification.

Package	Type	Description
Hmisc	GEN	Contains many functions useful for data analysis, high-level graphics, utility operations, functions for computing sample size and power, simulation, importing and annotating datasets, imputing missing values, advanced table making, variable clustering, character string manipulation, conversion of R objects to LaTeX and html code, and recoding variables. Particularly check out the describe() function.
lubridate	GEN	Functions to work with date-times and time-spans: fast and user friendly parsing of date-time data, extraction and updating of components of a date-time (years, months, days, hours, minutes, and seconds), algebraic manipulation on date-time and time-span objects.
stringr	GEN	Simple, Consistent Wrappers for Common String Operations

Package	Type	Description
tidyverse	GEN	The ‘tidyverse’ is a set of packages that work in harmony because they share common data representations and ‘API’ design. This package is designed to make it easy to install and load multiple ‘tidyverse’ packages in a single step.
uuid	GEN	Tools for generating and handling of UUIDs (Universally Unique Identifiers).

5.2 Python

Package	Type	Description
metapype	BIO	A lightweight Python 3 library for generating EML metadata
python-dwca-reader	BIO	A simple Python package to read and parse Darwin Core Archive (DwC-A) files, as produced by the GBIF website, the IPT and many other biodiversity informatics tools.
pyworms	BIO	Python client for the World Register of Marine Species (WoRMS) REST service.

Package	Type	Description
numpy	GEN	NumPy (Numerical Python) is an open source Python library that's used in almost every field of science and engineering. It's the universal standard for working with numerical data in Python, and it's at the core of the scientific Python and PyData ecosystems.
pandas	GEN	pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. Super helpful when manipulating tabular data!
uuid	GEN	This module provides immutable UUID objects (class UUID) and the functions uuid1(), uuid3(), uuid4(), uuid5() for generating version 1, 3, 4, and 5 UUIDs as specified in RFC 4122. Built in – part of the Python standard library.

Package	Type	Description
obis-qc	BIO	Quality checks on occurrence records. Checks `occurrenceStatus`, `individualCount`, `eventDate`, `decimalLatitude`, `decimalLongitude`, `coordinateUncertaintyInMeters`, `minimumDepthInMeters`, `maximumDepthInMeters`, `scientificName`, `scientificNameID`. Checks from Vandepitte et al. flags not implemented: 3, 9, 14, 15, 16, 10, 17, 21-30.
biopython	BIO	Biopython is a set of freely available tools for biological computation written in Python by an international team of developers. It is a distributed collaborative effort to develop Python libraries and applications which address the needs of current and future work in bioinformatics.

5.3 Google Sheets

Package	Description
Google Sheet DarwinCore Archive Assistant add-on	Google Sheet add-on which assists the creation of Darwin Core Archives (DwCA) and publising to Zenodo. DwCA's are stored into user's Google Drive and can be downloaded for upload into IPT installations or other software which is able to read DwC-archives.

5.4 Validators

Name	Description
Darwin Core Archive Validator	This validator verifies the structural integrity of a Darwin Core Archive. It does not check the data values, such as coordinates, dates or scientific names.
GBIF DATA VALIDATOR	The GBIF data validator is a service that allows anyone with a GBIF-relevant dataset to receive a report on the syntactical correctness and the validity of the content contained within the dataset.
LifeWatch Belgium	Through this interactive section of the LifeWatch.be portal users can upload their own data using a standard data format, and choose from several web services, models and applications to process the data.

References

Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2021. *Rmarkdown: Dynamic Documents for r*. https://CRAN.R-project.org/package=rmarkdown.

Bache, Stefan Milton, and Hadley Wickham. 2022. *Magrittr: A Forward-Pipe Operator for r*. https://CRAN.R-project.org/package=magrittr.

Barbier, Edward B. 2017. "Marine Ecosystem Services." *Current Biology*. Cell Press. https://doi.org/10.1016/j.cub.2017.03.020.

Benson, Abigail, Cassandra M. Brooks, Gabrielle Canonico, Emmett Duffy, Frank Muller-Karger, Heidi M. Sosik, Patricia Miloslavich, and Eduardo Klein. 2018. "Integrated Observations and Informatics Improve Understanding of Changing Marine Ecosystems." *Frontiers in Marine Science*. Frontiers Media S.A. https://doi.org/10.3389/fmars.2018.00428.

Benson, Abigail, Diana LaScala-Gruenewald, Robert McGuinn, Erin Satterthwaite, Stace Beaulieu, Mathew Biddle, Lynn deWitt, et al. 2021. "Biological Observation Data Standardization - a Primer for Data Managers." ESIP. https://doi.org/10.6084/m9.figshare.16806712.v1.

Benson, Abigail, Tylar Murray, Gabrielle Canonico, Enrique Montes, Frank Muller-Karger, Maria T. Kavanaugh, Joaquin Trinanes, and Lynn M. deWitt. 2021. "Data Management and Interactive Visualizations for the Evolving Marine Biodiversity Observation Network." *Oceanography*. https://doi.org/10.5670/oceanog.2021.220.

Borer, Elizabeth T., Eric W. Seabloom, Matthew B. Jones, and Mark Schildhauer. 2009. "Some Simple Guidelines for Effective Data Management." *Bulletin of the Ecological Society of America* 90 (April). https://doi.org/10.1890/0012-9623-90.2.205.

Canonico, Gabrielle, Pier Luigi Buttigieg, Enrique Montes, Frank E. Muller-Karger, Carol Stepien, Dawn Wright, Abigail Benson, et al. 2019. "Global Observational Needs and Resources for Marine Biodiversity." *Frontiers in Marine Science*. Frontiers Media S.A. https://doi.org/10.3389/fmars.2019.00367.

Chamberlain, Scott. 2020. *Worrms: World Register of Marine Species (WoRMS) Client*. https://CRAN.R-project.org/package=worrms.

Crystal-Ornelas, Robert, Charuleka Varadharajan, Ben Bond-Lamberty, Kristin

Boye, Madison Burrus, Shreyas Cholia, Michael Crow, et al. 2021. “A Guide to Using GitHub for Developing and Versioning Data Standards and Reporting Formats.” *Earth and Space Science*, July, e2021EA001797. https://doi.org/10.1029/2021EA001797.

Davies, Neil, John Deck, Eric C Kansa, Sarah Whitcher Kansa, John Kunze, Christopher Meyer, Thomas Orrell, et al. 2021. “Internet of Samples (iSamples): Toward an Interdisciplinary Cyberinfrastructure for Material Samples.” *GigaScience* 10: 1–5. https://doi.org/10.1093/gigascience/giab028.

Djurhuus, Anni, Collin J. Closek, Ryan P. Kelly, Kathleen J. Pitz, Reiko P. Michisaki, Hilary A. Starks, Kristine R. Walz, et al. 2020. “Environmental DNA Reveals Seasonal Shifts and Potential Interactions in a Marine Community.” *Nature Communications* 11 (December): 1–9. https://doi.org/10.1038/s41467-019-14105-1.

Duffy, J. Emmett, Linda A. Amaral-Zettler, Daphne G. Fautin, Gustav Paulay, Tatiana A. Rynearson, Heidi M. Sosik, and John J. Stachowicz. 2013. “Envisioning a Marine Biodiversity Observation Network.” *BioScience* 63 (May): 350–61. https://doi.org/10.1525/bio.2013.63.5.8.

Fornwall, M, R Gisiner, S E Simmons, H Moustahfid, G Canonico, P Halpin, P Goldstein, et al. 2012. “Expanding Biological Data Standards Development Processes for US IOOS: Visual Line Transect Observing Community for Mammal, Bird, and Turtle Data.” IOOS. https://www.researchgate.net/publication/255681522.

Grolemund, Garrett, and Hadley Wickham. 2011. “Dates and Times Made Easy with lubridate.” *Journal of Statistical Software* 40 (3): 1–25. https://www.jstatsoft.org/v40/i03/.

Hardisty, Alex R., William K. Michener, Donat Agosti, Enrique Alonso García, Lucy Bastin, Lee Belbin, Anne Bowser, et al. 2019. “The Bari Manifesto: An Interoperability Framework for Essential Biodiversity Variables.” *Ecological Informatics* 49 (January): 22–31. https://doi.org/10.1016/j.ecoinf.2018.11.003.

Heberling, J Mason, Joseph T Miller, Daniel Noesgaard, Scott B Weingart C, Dmitry Schigel, and Douglas E Soltis. 2021. “Data Integration Enables Global Biodiversity Synthesis.” *Proceedings of the National Academy of Sciences of the United States of America*. https://doi.org/10.1073/pnas.2018093118/-/DCSupplemental.

Henry, Lionel, and Hadley Wickham. 2020. *Purrr: Functional Programming Tools*. https://CRAN.R-project.org/package=purrr.

Jonathan A. Hare, Mark W. Nelson, Wendy E. Morrison. n.d. “A Vulnerability Assessment of Fish and Invertebrates to Climate Change on the Northeast u.s. Continental Shelf.” *PLoS ONE* 11 (2): e0146756. https://doi.org/10.1371/journal.pone.0146756.

Jones, Matthew B., Mark P. Schildhauer, O. J. Reichman, and Shawn Bowers. 2006. “The New Bioinformatics: Integrating Ecological Data from the Gene to the Biosphere.” *Annual Review of Ecology, Evolution, and Systematics*. https://doi.org/10.1146/annurev.ecolsys.37.091305.110031.

Kavanaugh, Maria T., Matthew J. Oliver, Francisco P. Chavez, Ricardo M.

Letelier, Frank E. Muller-Karger, and Scott C. Doney. 2016. “Seascapes as a New Vernacular for Pelagic Ocean Monitoring, Management and Conservation.” *ICES Journal of Marine Science* 73 (July): 1839–50. https://doi.org/10.1093/icesjms/fsw086.

Kot, Connie Y., Ei Fujioka, Lucie J. Hazen, Benjamin D. Best, Andrew J. Read, and Patrick N. Halpin. 2010. “Spatio-Temporal Gap Analysis of OBIS-SEAMAP Project Data: Assessment and Way Forward.” *PLoS ONE* 5 (September): 12990. https://doi.org/10.1371/journal.pone.0012990.

Lab, Malin Pinksy. n.d. “OceanAdapt.” *GitHub*. https://oceanadapt.rutgers.edu/.

Lamprecht, Anna-Lena, Leyla Garcia, Mateusz Kuzak, Carlos Martinez, Ricardo Arcila, Eva Martin Del Pico, Victoria Dominguez Del Angel, et al. 2019. “Towards FAIR Principles for Research Software.” Edited by Paul Groth. *Data Science*, 1–23. https://doi.org/10.3233/DS-190026.

Maria Dornelas, Brian McGill, Nicholas J. Gotelli. 2014. “Assemblage Time Series Reveal Biodiversity Change but Not Systematic Loss.” *Science* 344 (6181): 296–99. https://doi.org/10.1126/science.1248484.

McKenna, Megan F., Simone Baumann-Pickering, Annebelle C. M. Kok, William K. Oestreich, Jeffrey D. Adams, Jack Barkowski, Kurt M. Fristrup, et al. 2021. “Advancing the Interpretation of Shallow Water Marine Soundscapes.” *Frontiers in Marine Science* 0 (September): 1426. https://doi.org/10.3389/FMARS.2021.719258.

Miloslavich, Patricia, Nicholas J. Bax, Samantha E. Simmons, Eduardo Klein, Ward Appeltans, Octavio Aburto-Oropeza, Melissa Andersen Garcia, et al. 2018. “Essential Ocean Variables for Global Sustained Observations of Biodiversity and Ecosystem Changes.” *Global Change Biology* 24 (June): 2416–33. https://doi.org/10.1111/gcb.14108.

Montes, Enrique, Anni Djurhuus, Frank E. Muller-Karger, Daniel Otis, Christopher R. Kelble, and Maria T. Kavanaugh. 2020. “Dynamic Satellite Seascapes as a Biogeographic Framework for Understanding Phytoplankton Assemblages in the Florida Keys National Marine Sanctuary, United States.” *Frontiers in Marine Science* 7 (July): 575. https://doi.org/10.3389/fmars.2020.00575.

Moustahfid, Hassan, and Philip Goldstein. 2014. “IOOS Biological Data Services Enrollment Procedures.”

Moustahfid, Hassan, Jim Potemra, Philip Goldstein, Roy Mendelssohn, and Annette Desrochers. 2011. “Making United States Integrated Ocean Observing System (u.s. IOOS) Inclusive of Marine Biological Resources.” https://www.researchgate.net/publication/254013004.

Müller, Kirill. 2020. *Here: A Simpler Way to Find Your Files*. https://CRAN.R-project.org/package=here.

Müller, Kirill, and Hadley Wickham. 2021. *Tibble: Simple Data Frames*. https://CRAN.R-project.org/package=tibble.

Muller-Karger, Frank E., Erin Hestir, Christiana Ade, Kevin Turpie, Dar A. Roberts, David Siegel, Robert J. Miller, et al. 2018. “Satellite Sensor Requirements for Monitoring Essential Biodiversity Variables of Coastal

Ecosystems.” *Ecological Applications* 28 (April): 749–60. https://doi.org/10.1002/eap.1682.

Muller-Karger, Frank E., Patricia Miloslavich, Nicholas J. Bax, Samantha Simmons, Mark J. Costello, Isabel Sousa Pinto, Gabrielle Canonico, et al. 2018. “Advancing Marine Biological Observations and Data Requirements of the Complementary Essential Ocean Variables (EOVs) and Essential Biodiversity Variables (EBVs) Frameworks.” *Frontiers in Marine Science*. Frontiers Media S.A. https://doi.org/10.3389/fmars.2018.00211.

Neeley, Aimee, Stace E. Beaulieu, Chris Proctor, Ivona Cetinić, Joe Futrelle, Inia Soto Ramos, Heidi M. Sosik, et al. 2021. “Standards and Practices for Reporting Plankton and Other Particle Observations from Images.” https://doi.org/10.1575/1912/27377.

O’Brien, Margaret, Colin A. Smith, Eric R. Sokol, Corinna Gries, Nina Lany, Sydne Record, and Max C. N. Castorani. 2021. “ecocomDP: A Flexible Data Design Pattern for Ecological Community Survey Data.” *Ecological Informatics* 64 (September): 101374. https://doi.org/10.1016/J.ECOINF.2021.101374.

Pooter, Daphnis De, Ward Appeltans, Nicolas Bailly, Sky Bristol, Klaas Deneudt, Menashè Eliezer, Ei Fujioka, et al. 2017. “Toward a New Data Standard for Combined Marine Biological and Environmental Datasets - Expanding OBIS Beyond Species Occurrences.” *Biodiversity Data Journal* 5 (January): 10989. https://doi.org/10.3897/BDJ.5.e10989.

Provoost, Pieter. n.d. “Iobis/Ebsa.” *GitHub*. https://github.com/iobis/ebsa.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Rücknagel, J., P. Vierkant, R. Ulrich, G. Kloska, E. Schnepf, D. Fichtmüller, E. Reuter, et al. 2015. “Metadata Schema for the Description of Research Data Repositories. Version 3.0.” https://doi.org/10.2312/re3.008.

Rule, Adam, Amanda Birmingham, Cristal Zuniga, Ilkay Altintas, Shih-Cheng Huang, Rob Knight, Niema Moshiri, et al. 2019. “Ten Simple Rules for Writing and Sharing Computational Analyses in Jupyter Notebooks.” *PLOS Computational Biology* 15 (July): e1007007. https://doi.org/10.1371/JOURNAL.PCBI.1007007.

Santora, Jarrod A., Elliott L. Hazen, Isaac D. Schroeder, Steven J. Bograd, Keith M. Sakuma, and John C. Field. 2017. “Impacts of Ocean Climate Variability on Biodiversity of Pelagic Forage Species in an Upwelling Ecosystem.” *Marine Ecology Progress Series* 580 (September): 205–20. https://doi.org/10.3354/meps12278.

Schmid, Moritz S, Dominic Daprano, Kyler M Jacobson, Christopher Sullivan, Christian Briseño-Avena, Jessica Y Luo, and Robert K Cowen. 2021. *A Convolutional Neural Network Based High-Throughput Image Classification Pipeline - Code and Documentation to Process Plankton Underwater Imagery Using Local HPC Infrastructure and NSF’s XSEDE*. Zenodo. https://doi.org/10.5281/ZENODO.4641158.

Spinu, Vitalie, Garrett Grolemund, and Hadley Wickham. 2021. *Lubridate:*

Make Dealing with Dates a Little Easier. https://CRAN.R-project.org/package=lubridate.

Taylor, Gordon T., Frank E. Muller-Karger, Robert C. Thunell, Mary I. Scranton, Yrene Astor, Ramon Varela, Luis Troccoli Ghinaglia, et al. 2012. "Ecosystem Responses in the Southern Caribbean Sea to Global Climate Change." *Proceedings of the National Academy of Sciences of the United States of America* 109 (November): 19315–20. https://doi.org/10.1073/pnas.1207514109.

Tittensor, Derek P, Camilo Mora, Walter Jetz, Heike K Lotze, Daniel Ricard, Edward Vanden Berghe, and Boris Worm. 2010. "Global Patterns and Predictors of Marine Biodiversity Across Taxa." *Nature* 466 (7310): 1098.

Warren, R., J. Price, E. Graham, N. Forstenhaeusler, and J. VanDerWal. 2018. "The Projected Effect on Insects, Vertebrates, and Plants of Limiting Global Warming to 1.5°c Rather Than 2°c." *Science* 360 (May): 791–95. https://doi.org/10.1126/science.aar3646.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

———. 2019. *Stringr: Simple, Consistent Wrappers for Common String Operations*. https://CRAN.R-project.org/package=stringr.

———. 2021a. *Forcats: Tools for Working with Categorical Variables (Factors)*. https://CRAN.R-project.org/package=forcats.

———. 2021b. *Tidyverse: Easily Install and Load the Tidyverse*. https://CRAN.R-project.org/package=tidyverse.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, and Dewey Dunnington. 2021. *Ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. https://CRAN.R-project.org/package=ggplot2.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2022. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.

Wickham, Hadley, and Maximilian Girlich. 2022. *Tidyr: Tidy Messy Data*. https://CRAN.R-project.org/package=tidyr.

Wickham, Hadley, Jim Hester, and Jennifer Bryan. 2022. *Readr: Read Rectangular Text Data*. https://CRAN.R-project.org/package=readr.

Wieczorek, John, David Bloom, Robert Guralnick, Stan Blum, Markus Döring, Renato Giovanni, Tim Robertson, and David Vieglais. 2012. "Darwin Core: An Evolving Community-Developed Biodiversity Data Standard." *PLoS ONE* 7 (January): 29715. https://doi.org/10.1371/journal.pone.0029715.

Wilkinson, Mark D., Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, et al. 2016. "The FAIR Guiding Principles for Scientific Data Management and Stewardship." *Scientific Data* 3 (March): 1–9. https://doi.org/10.1038/sdata.2016.18.

Xie, Yihui. 2014. "Knitr: A Comprehensive Tool for Reproducible Research in R." In *Implementing Reproducible Computational Research*, edited by Victoria Stodden, Friedrich Leisch, and Roger D. Peng. Chapman; Hall/CRC. http://www.crcpress.com/product/isbn/9781466561595.

———. 2015b. *Dynamic Documents with R and Knitr.* 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. http://yihui.name/knitr/.

———. 2015a. *Dynamic Documents with R and Knitr.* 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.org/knitr/.

———. 2016. *Bookdown: Authoring Books and Technical Documents with R Markdown.* Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/bookdown.

———. 2021a. *Bookdown: Authoring Books and Technical Documents with r Markdown.* https://CRAN.R-project.org/package=bookdown.

———. 2021b. *Knitr: A General-Purpose Package for Dynamic Report Generation in r.* https://yihui.org/knitr/.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide.* Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.

Xie, Yihui, Christophe Dervieux, and Emily Riederer. 2020. *R Markdown Cookbook.* Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown-cookbook.