Darwin Core Marine Example Compendium

By: Standardizing Marine Biological Data Working Group

Updated: 2024-09-04

Contents

Preface

This book contains a collection of examples and resources related to mobilizing marine biological data to the Darwin Core standard for sharing though OBIS. This book has been developed by the Standardizing Marine Biological Data Working Group (SMBD). The working group is an open community of practitioners, experts, and scientists looking to learn and educate the community on standardizing and sharing marine biological data.

If you would like to join the SMBD or learn more, checkout this README.

Chapter 1

Introduction

The world of standardizing marine biological data can seem complex for the naive oceanographer, biologist, scientist, or programmer. This book intends to ease the burden of learning about the Darwin Core standard by compiling a list of example applications and tools for translating source data into Darwin Core. This collection of resources does not replace the Darwin Core standards documentation (https://dwc.tdwg.org/) or the OBIS Manual (https://manual.obis.org/), but instead it supplements those resources with examples of real world applications.

In this book we cover:

- Applications - These are the real world examples of aligning data to Darwin Core.
- Frequently Asked Questions - A collection of Frequently Asked Questions.
- Tools - A collection of useful tools, packages, and programs for working with marine biological data.
- Extras - Some useful extra tidbits about metadata and using GitHub to debug data issues.

If you would like to learn more about standardizing biological data (not only marine), 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.

Chapter 2

Applications

This chapter contains a series of example applications to convert source data to the Darwin Core standard. You can find these examples (and more!) in the GitHub repository under the `datasets/` directory.

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

Abby Benson
January 9, 2022

2.1.1 General information about this notebook

Script to process the Texas Parks and Wildlife Department (TPWD) Aransas Bay bag seine data from the format used by the Houston Advanced Research Center (HARC) for bays in Texas. Taxonomy was processed using a separate script (TPWD_Taxonomy.R) using a taxa list pulled from the pdf "2009 Resource Monitoring Operations Manual". All original data, processed data and scripts are stored on an item in USGS ScienceBase.

```
# Load some of the libraries
library(reshape2)
library(tidyverse)
library(readr)
```

```
# Load the data
BagSeine <- read.csv("https://www.sciencebase.gov/catalog/file/get/53a887f4e4b075096c60cfdd?f=__d
```

Note that if not already done you'll need to run the TPWD_Taxonomy.R script to get the taxaList file squared away or load the taxonomy file to the World Register of Marine Species Taxon Match Tool https://www.marinespecies.org/aphia.php?p=match

2.1.2   Event file

To start we will create the Darwin Core **Event** file. This is the file that will have all the information about the sampling event such as date, location, depth, sampling protocol. Basically anything about the cruise or the way the sampling was done will go in this file. You can see all the Darwin Core terms that are part of the event file here http://tools.gbif.org/dwca-validator/extension.do?id=dwc:Event.

The original format for these TPWD HARC files has all of the information associated as the event in the first approximately 50 columns and then all of the information about the occurrence (species) as columns for each species. We will need to start by limiting to the event information only.

```r
event <- BagSeine[,1:47]
```

Next there are several pieces of information that need 1) to be added like the geodeticDatum 2) to be pieced together from multiple columns like datasetID or 3) minor changes like the minimum and maximum depth.

```r
event <- event %>%
  mutate(type = "Event",
         modified = lubridate::today(),
         language = "en",
         license = "http://creativecommons.org/publicdomain/zero/1.0/legalcode",
         institutionCode = "TPWD",
         ownerInstitutionCode = "HARC",
         coordinateUncertaintyInMeters = "100",
         geodeticDatum = "WGS84",
         georeferenceProtocol = "Handheld GPS",
         country = "United States",
         countryCode = "US",
         stateProvince = "Texas",
         datasetID = gsub(" ", "_", paste("TPWD_HARC_Texas", event$Bay, event$Gear_Type
         eventID = paste("Station", event$station_code, "Date", event$completion_dttm,
         sampleSizeUnit = "hectares",
         CompDate = lubridate::mdy_hms(event$CompDate, tz="America/Chicago"),
         StartDate = lubridate::mdy_hms(event$StartDate, tz="America/Chicago"),
         minimumDepthInMeters = ifelse(start_shallow_water_depth_num < start_deep_water
                                         start_shallow_water_depth_num, start_deep_water_
```

```
        maximumDepthInMeters = ifelse(start_deep_water_depth_num > start_shallow_water_depth_num
                                      start_deep_water_depth_num, start_shallow_water_depth_num)
```

```
head(event[,48:64], n = 10)
     type    modified language                                              license instituti
1   Event 2022-01-09       en http://creativecommons.org/publicdomain/zero/1.0/legalcode
2   Event 2022-01-09       en http://creativecommons.org/publicdomain/zero/1.0/legalcode
3   Event 2022-01-09       en http://creativecommons.org/publicdomain/zero/1.0/legalcode
4   Event 2022-01-09       en http://creativecommons.org/publicdomain/zero/1.0/legalcode
5   Event 2022-01-09       en http://creativecommons.org/publicdomain/zero/1.0/legalcode
6   Event 2022-01-09       en http://creativecommons.org/publicdomain/zero/1.0/legalcode
7   Event 2022-01-09       en http://creativecommons.org/publicdomain/zero/1.0/legalcode
8   Event 2022-01-09       en http://creativecommons.org/publicdomain/zero/1.0/legalcode
9   Event 2022-01-09       en http://creativecommons.org/publicdomain/zero/1.0/legalcode
10  Event 2022-01-09       en http://creativecommons.org/publicdomain/zero/1.0/legalcode
   ownerInstitutionCode coordinateUncertaintyInMeters geodeticDatum georeferenceProtocol       co
1                  HARC                           100         WGS84         Handheld GPS United S
2                  HARC                           100         WGS84         Handheld GPS United S
3                  HARC                           100         WGS84         Handheld GPS United S
4                  HARC                           100         WGS84         Handheld GPS United S
5                  HARC                           100         WGS84         Handheld GPS United S
6                  HARC                           100         WGS84         Handheld GPS United S
7                  HARC                           100         WGS84         Handheld GPS United S
8                  HARC                           100         WGS84         Handheld GPS United S
9                  HARC                           100         WGS84         Handheld GPS United S
10                 HARC                           100         WGS84         Handheld GPS United S
   countryCode stateProvince                                datasetID
1           US         Texas TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_95_Date_09JAN1997:14:3
2           US         Texas TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_95_Date_18AUG2000:11:0
3           US         Texas TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_96_Date_28JUN2005:08:4
4           US         Texas TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_96_Date_23AUG2006:11:4
5           US         Texas TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_96_Date_17OCT2006:14:2
6           US         Texas TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_96_Date_19FEB1996:10:2
7           US         Texas TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_96_Date_11JUN2001:14:1
8           US         Texas TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_96_Date_16MAR1992:09:4
9           US         Texas TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_96_Date_25SEP1996:11:2
10          US         Texas TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_96_Date_08MAY1997:13:2
   sampleSizeUnit minimumDepthInMeters maximumDepthInMeters
1        hectares                  0.0                  0.6
2        hectares                  0.1                  0.5
3        hectares                  0.4                  0.6
4        hectares                  0.2                  0.4
5        hectares                  0.7                  0.8
6        hectares                  0.1                  0.3
7        hectares                  0.4                  0.5
```

8	hectares	0.0	0.4
9	hectares	0.3	0.7
10	hectares	0.4	0.6

For this dataset there was a start timestamp and end timestamp that we can use to identify the sampling effort which can be really valuable information for downstream users when trying to reuse data from multiple projects.

```
## Calculate duration of bag seine event
event$samplingEffort <- ""
for (i in 1:nrow(event)){
  event[i,]$samplingEffort <- abs(lubridate::as.duration(event[i,]$CompDate - event[i,]
}
event$samplingEffort <- paste(event$samplingEffort, "seconds", sep = " ")
```

Finally there were a few columns that were a direct match to a Darwin Core term and therefore just need to be renamed to follow the standard.

```
event <- event %>%
  rename(samplingProtocol = Gear_Type,
         locality = Estuary,
         waterBody = SubBay,
         decimalLatitude = Latitude,
         decimalLongitude = Longitude,
         sampleSizeValue = surface_area_num,
         eventDate = CompDate)
```

2.1.3   Occurrence file

The next file we need to create is the **Occurrence** file. This file includes all the information about the species that were observed. An occurrence in Darwin Core is the intersection of an organism at a time and a place. We have already done the work to identify the time and place in the event file so we don't need to do that again here. What we do need to is identify all the information about the organisms. Another piece of information that goes in here is basisOfRecord which is a required field and has a controlled vocabulary. For the data we work with you'll usually put `HumanObservation` or `MachineObservation`. If it's eDNA data you'll use `MaterialSample`. If your data are part of a museum collection you'll use `PreservedSpecimen`.

Important to note that there is overlap in the Darwin Core terms that "allowed" to be in the event file and in the occurrence file. This is because data can be submitted as "Occurrence Only" where you don't have a separate event file. In that case, the location and date information will need to be included in the

occurrence file. Since we are formatting this dataset as a sampling event we will not include location and date information in the occurrence file. To see all the Darwin Core terms that can go in the occurrence file go here https://tools.gbif.org/dwca-validator/extension.do?id=dwc:occurrence.

This dataset in its original format is in "wide format". All that means is that data that we would expect to be encoded as values in the rows are instead column headers. We have to pull all the scientific names out of the column headers and turn them into actual values in the data.

```
occurrence <- melt(BagSeine, id=1:47, measure=48:109, variable.name="vernacularName", value.name=
```

You'll notice when we did that step we went from 5481 obs (or rows) in the data to 334341 obs. We went from wide to long.

```
dim(BagSeine)
[1] 5481   109
dim(occurrence)
[1] 334341     49
```

Now as with the event file we have several pieces of information that need to be added or changed to make sure the data are following Darwin Core. We always want to include as much information as possible to make the data as reusable as possible.

```
occurrence <- occurrence %>%
  mutate(vernacularName = gsub("\\.",' ', vernacularName),
         eventID = paste("Station", station_code, "Date", completion_dttm, sep = "_"),
         occurrenceStatus = ifelse(relativeAbundance == 0, "Absent", "Present"),
         basisOfRecord = "HumanObservation",
         organismQuantityType = "Relative Abundance",
         collectionCode = paste(Bay, Gear_Type, sep = " "))
```

We will match the taxa list with our occurrence file data to bring in the taxonomic information that we pulled from WoRMS. To save time you'll just import the processed taxa list which includes the taxonomic hierarchy and the required term scientificNameID which is one of the most important pieces of information to include for OBIS.

```
taxaList <- read.csv("https://www.sciencebase.gov/catalog/file/get/53a887f4e4b075096c60cfdd?f=__d
## Merge taxaList with occurrence
occurrence <- merge(occurrence, taxaList, by = "vernacularName", all.x = T)
## Test that all the vernacularNames found a match in taxaList_updated
Hmisc::describe(occurrence$scientificNameID)
      n  missing distinct
```

```
   334341          0          61
```

```
lowest : urn:lsid:marinespecies.org:taxname:105792 urn:lsid:marinespecies.org:taxname:
highest: urn:lsid:marinespecies.org:taxname:367528 urn:lsid:marinespecies.org:taxname:
```

For that last line of code we are expecting to see no missing values for scientificNameID. Every row in the file should have a value in scientificNameID which should be a WoRMS LSID that look like this `urn:lsid:marinespecies.org:taxname:144531`

We need to create a unique ID for each row in the occurrence file. This is known as the `occurrenceID` and is a required term. The `occurrenceID` needs to be globally unique and needs to be permanent and kept in place if any updates to the dataset are made. You should not create brand new occurrenceIDs when you update a dataset. To facilitate this I like to build the `occurrenceID` from pieces of information available in the dataset to create a unique ID for each row in the occurrence file. For this dataset I used the `eventID` (Station + Date) plus the scientific name. This only works if there is only one scientific name per station per date so if you have different ages or sexes of species at the same station and date this method of creating the occurrenceID won't work for you.

```
occurrence$occurrenceID <- paste(occurrence$eventID, gsub(" ", "_",occurrence$scientif
occurrence[1,]$occurrenceID
[1] "Station_95_Date_09JAN1997:14:35:00.000_Atractosteus_spatula"
```

For the occurrence file we only have one column to rename. We could have avoided this step if we had named it `organismQuantity` up above but I kept this to remind me what the data providers had called this.

```
occurrence <- occurrence %>%
  rename(organismQuantity = relativeAbundance)
```

2.1.4   Extended Measurement or Fact extension file

The final file we are going to create is the **Extended Measurement or Fact extension (emof)**. This is a bit like a catch all for any measurements or facts that are not captured in Darwin Core. Darwin Core does not have terms for things like temperature, salinity, gear type, cruise number, length, weight, etc. We are going to create a long format file where each of these is a set of rows in the extended measurement or fact file. You can find all the terms in this extension here https://tools.gbif.org/dwca-validator/extension.do?id= http://rs.iobis.org/obis/terms/ExtendedMeasurementOrFact.

OBIS uses the BODC NERC Vocabulary Server to provide explicit definitions for each of the measurements https://vocab.nerc.ac.uk/search_nvs/.

For this dataset I was only able to find code definitions provided by the data providers for some of the measurements. I included the ones that I was able to find code definitions and left out any that I couldn't find those for. The ones I was able to find code definitions for were `Total.Of.Samples_Count`, `gear_size`, `start_wind_speed_num`, `start_barometric_pressure_num`, `start_temperature_num`, `start_salinity_num`, `start_dissolved_oxygen_num`. All the others I left out.

```r
totalOfSamples <- event[c("Total.Of.Samples_Count", "eventID")]
totalOfSamples <- totalOfSamples[which(!is.na(totalOfSamples$Total.Of.Samples_Count)),]
totalOfSamples <- totalOfSamples %>%
  mutate(measurementType = "Total number of samples used to calculate relative abundance",
         measurementUnit = "",
         measurementTypeID = "",
         measurementUnitID = "",
         occurrenceID = "") %>%
  rename(measurementValue = Total.Of.Samples_Count)

gear_size <- event[c("gear_size", "eventID")]
gear_size <- gear_size[which(!is.na(gear_size$gear_size)),]
gear_size <- gear_size %>%
  mutate(measurementType = "gear size",
         measurementUnit = "meters",
         measurementTypeID = "http://vocab.nerc.ac.uk/collection/P01/current/MTHAREA1/",
         measurementUnitID = "http://vocab.nerc.ac.uk/collection/P06/current/ULAA/",
         occurrenceID = "") %>%
  rename(measurementValue = gear_size)

start_wind_speed_num <- event[c("start_wind_speed_num", "eventID")]
start_wind_speed_num <- start_wind_speed_num[which(!is.na(start_wind_speed_num$start_wind_speed_n
start_wind_speed_num <- start_wind_speed_num %>%
  mutate(measurementType = "wind speed",
         measurementUnit = "not provided",
         measurementTypeID = "http://vocab.nerc.ac.uk/collection/P01/current/EWSBZZ01/",
         measurementUnitID = "",
         occurrenceID = "") %>%
  rename(measurementValue = start_wind_speed_num)

start_barometric_pressure_num <- event[c("start_barometric_pressure_num", "eventID")]
start_barometric_pressure_num <- start_barometric_pressure_num[which(!is.na(start_barometric_pres
start_barometric_pressure_num <- start_barometric_pressure_num %>%
  mutate(measurementType = "barometric pressure",
         measurementUnit = "not provided",
         measurementTypeID = "http://vocab.nerc.ac.uk/collection/P07/current/CFSN0015/",
         measurementUnitID = "",
         occurrenceID = "") %>%
```

```r
  rename(measurementValue = start_barometric_pressure_num)

start_temperature_num <- event[c("start_temperature_num", "eventID")]
start_temperature_num <- start_temperature_num[which(!is.na(start_temperature_num$start
start_temperature_num <- start_temperature_num %>%
  mutate(measurementType = "water temperature",
         measurementUnit = "Celsius",
         measurementTypeID = "http://vocab.nerc.ac.uk/collection/P01/current/TEMPPR01/
         measurementUnitID = "http://vocab.nerc.ac.uk/collection/P06/current/UPAA/",
         occurrenceID = "") %>%
  rename(measurementValue = start_temperature_num)

start_salinity_num <- event[c("start_salinity_num", "eventID")]
start_salinity_num <- start_salinity_num[which(!is.na(start_salinity_num$start_salinity
start_salinity_num <- start_salinity_num %>%
  mutate(measurementType = "salinity",
         measurementUnit = "ppt",
         measurementTypeID = "http://vocab.nerc.ac.uk/collection/P01/current/ODSDM021/
         measurementUnitID = "http://vocab.nerc.ac.uk/collection/P06/current/UPPT/",
         occurrenceID = "") %>%
  rename(measurementValue = start_salinity_num)

start_dissolved_oxygen_num <- event[c("start_dissolved_oxygen_num", "eventID")]
start_dissolved_oxygen_num <- start_dissolved_oxygen_num[which(!is.na(start_dissolved_o
start_dissolved_oxygen_num <- start_dissolved_oxygen_num %>%
  mutate(measurementType = "dissolved oxygen",
         measurementUnit = "ppm",
         measurementTypeID = "http://vocab.nerc.ac.uk/collection/P09/current/DOX2/",
         measurementUnitID = "http://vocab.nerc.ac.uk/collection/P06/current/UPPM/",
         occurrenceID = "") %>%
  rename(measurementValue = start_dissolved_oxygen_num)

alternate_station_code <- event[c("alternate_station_code", "eventID")]
alternate_station_code <- alternate_station_code[which(!is.na(alternate_station_code$al
alternate_station_code <- alternate_station_code %>%
  mutate(measurementType = "alternate station code",
         measurementUnit = "",
         measurementTypeID = "",
         measurementUnitID = "",
         occurrenceID = "") %>%
  rename(measurementValue = alternate_station_code)

organismQuantity <- occurrence[c("organismQuantity", "eventID", "occurrenceID")]
organismQuantity <- organismQuantity[which(!is.na(organismQuantity$organismQuantity)),]
organismQuantity <- organismQuantity %>%
```

```r
  mutate(measurementType = "relative abundance",
         measurementUnit = "",
         measurementTypeID = "http://vocab.nerc.ac.uk/collection/S06/current/S0600020/",
         measurementUnitID = "") %>%
  rename(measurementValue = organismQuantity)

# Bind the separate measurements together into one file
mof <- rbind(totalOfSamples, start_barometric_pressure_num, start_dissolved_oxygen_num,
             start_salinity_num, start_temperature_num, start_wind_speed_num, gear_size,
             alternate_station_code, organismQuantity)
head(mof)
 measurementValue                                eventID
1               18 Station_95_Date_09JAN1997:14:35:00.000
2              103 Station_95_Date_18AUG2000:11:02:00.000
3              401 Station_96_Date_28JUN2005:08:41:00.000
4               35 Station_96_Date_23AUG2006:11:47:00.000
5               57 Station_96_Date_17OCT2006:14:23:00.000
6                5 Station_96_Date_19FEB1996:10:27:00.000
                                                    measurementType measurementUnit measurementTypeID
1 Total number of samples used to calculate relative abundance
2 Total number of samples used to calculate relative abundance
3 Total number of samples used to calculate relative abundance
4 Total number of samples used to calculate relative abundance
5 Total number of samples used to calculate relative abundance
6 Total number of samples used to calculate relative abundance
  measurementUnitID occurrenceID
1
2
3
4
5
6
tail(mof)
       measurementValue                                eventID      measurementType measurementUni
334336        0.0000000 Station_217_Date_03APR2003:13:28:00.000 relative abundance
334337        0.0000000 Station_217_Date_24FEB2006:10:12:00.000 relative abundance
334338        0.1428571 Station_217_Date_23JUN2001:12:28:00.000 relative abundance
334339        0.0000000 Station_212_Date_23MAY1990:10:43:00.000 relative abundance
334340        0.1224490 Station_212_Date_24JUL1990:09:34:00.000 relative abundance
334341        0.0000000 Station_212_Date_21MAR2001:11:52:00.000 relative abundance
                                                    measurementTypeID measurementUnitID
334336 http://vocab.nerc.ac.uk/collection/S06/current/S0600020/
334337 http://vocab.nerc.ac.uk/collection/S06/current/S0600020/
334338 http://vocab.nerc.ac.uk/collection/S06/current/S0600020/
334339 http://vocab.nerc.ac.uk/collection/S06/current/S0600020/
```

```
334340 http://vocab.nerc.ac.uk/collection/S06/current/S0600020/
334341 http://vocab.nerc.ac.uk/collection/S06/current/S0600020/
                                                        occurrenceID
334336 Station_217_Date_03APR2003:13:28:00.000_Litopenaeus_setiferus
334337 Station_217_Date_24FEB2006:10:12:00.000_Litopenaeus_setiferus
334338 Station_217_Date_23JUN2001:12:28:00.000_Litopenaeus_setiferus
334339 Station_212_Date_23MAY1990:10:43:00.000_Litopenaeus_setiferus
334340 Station_212_Date_24JUL1990:09:34:00.000_Litopenaeus_setiferus
334341 Station_212_Date_21MAR2001:11:52:00.000_Litopenaeus_setiferus

# Write out the file
write.csv(mof, file = (paste0(event[1,]$datasetID, "_mof_", lubridate::today(),".csv")
```

2.1.5 Cleaning up Event and Occurrence files

Now that we have all of our files created we can clean up the **Event** and **Oc-
currence** files to remove the columns that are not following Darwin Core. We
had to leave the extra bits in before because we needed them to create the **emof**
file above.

```
event <- event[c("samplingProtocol","locality","waterBody","decimalLatitude","decimalLo
                 "eventDate","sampleSizeValue","minimumDepthInMeters",
                 "maximumDepthInMeters","type","modified","language","license","institu
                 "ownerInstitutionCode","coordinateUncertaintyInMeters",
                 "geodeticDatum", "georeferenceProtocol","country","countryCode","state
                 "datasetID","eventID","sampleSizeUnit","samplingEffort")]
head(event)
  samplingProtocol                 locality   waterBody decimalLatitude decimalLongitude
1       Bag Seine Mission-Aransas Estuary Aransas Bay        28.13472         -97.00833
2       Bag Seine Mission-Aransas Estuary Aransas Bay        28.13528         -97.00722
3       Bag Seine Mission-Aransas Estuary Aransas Bay        28.13444         -96.99611
4       Bag Seine Mission-Aransas Estuary Aransas Bay        28.13444         -96.99611
5       Bag Seine Mission-Aransas Estuary Aransas Bay        28.13444         -96.99611
6       Bag Seine Mission-Aransas Estuary Aransas Bay        28.13472         -96.99583
            eventDate sampleSizeValue minimumDepthInMeters maximumDepthInMeters  type
1 1997-01-09 14:35:00            0.03                  0.0                  0.6 Event
2 2000-08-18 11:02:00            0.03                  0.1                  0.5 Event
3 2005-06-28 08:41:00            0.03                  0.4                  0.6 Event
4 2006-08-23 11:47:00            0.03                  0.2                  0.4 Event
5 2006-10-17 14:23:00            0.03                  0.7                  0.8 Event
6 1996-02-19 10:27:00            0.03                  0.1                  0.3 Event
                                       license institutionCode ownerInsti
1 http://creativecommons.org/publicdomain/zero/1.0/legalcode            TPWD
2 http://creativecommons.org/publicdomain/zero/1.0/legalcode            TPWD
```

```
3 http://creativecommons.org/publicdomain/zero/1.0/legalcode                    TPWD                    HAR
4 http://creativecommons.org/publicdomain/zero/1.0/legalcode                    TPWD                    HAR
5 http://creativecommons.org/publicdomain/zero/1.0/legalcode                    TPWD                    HAR
6 http://creativecommons.org/publicdomain/zero/1.0/legalcode                    TPWD                    HAR
  coordinateUncertaintyInMeters geodeticDatum georeferenceProtocol       country countryCode stat
1                           100         WGS84       Handheld GPS United States           US
2                           100         WGS84       Handheld GPS United States           US
3                           100         WGS84       Handheld GPS United States           US
4                           100         WGS84       Handheld GPS United States           US
5                           100         WGS84       Handheld GPS United States           US
6                           100         WGS84       Handheld GPS United States           US
                        datasetID                                eventID sampleSizeUnit
1 TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_95_Date_09JAN1997:14:35:00.000       hectares
2 TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_95_Date_18AUG2000:11:02:00.000       hectares
3 TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_96_Date_28JUN2005:08:41:00.000       hectares
4 TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_96_Date_23AUG2006:11:47:00.000       hectares
5 TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_96_Date_17OCT2006:14:23:00.000       hectares
6 TPWD_HARC_Texas_Aransas_Bay_Bag_Seine Station_96_Date_19FEB1996:10:27:00.000       hectares
  samplingEffort
1     120 seconds
2     120 seconds
3     120 seconds
4     120 seconds
5     120 seconds
6     120 seconds

write.csv(event, file = paste0(event[1,]$datasetID, "_event_", lubridate::today(),".csv"), fileEn

occurrence <- occurrence[c("vernacularName","eventID","occurrenceStatus","basisOfRecord",
                   "scientificName","scientificNameID","kingdom","phylum","class",
                   "order","family","genus",
                   "scientificNameAuthorship","taxonRank", "organismQuantity",
                   "organismQuantityType", "occurrenceID","collectionCode")]
head(occurrence)
  vernacularName                                eventID occurrenceStatus    basisOfRecord
1  Alligator gar Station_95_Date_09JAN1997:14:35:00.000           Absent HumanObservation
2  Alligator gar Station_95_Date_18AUG2000:11:02:00.000           Absent HumanObservation
3  Alligator gar Station_96_Date_28JUN2005:08:41:00.000           Absent HumanObservation
4  Alligator gar Station_96_Date_23AUG2006:11:47:00.000           Absent HumanObservation
5  Alligator gar Station_96_Date_17OCT2006:14:23:00.000           Absent HumanObservation
6  Alligator gar Station_96_Date_19FEB1996:10:27:00.000           Absent HumanObservation
       scientificName                        scientificNameID  kingdom   phylum        class
1 Atractosteus spatula urn:lsid:marinespecies.org:taxname:279822 Animalia Chordata Actinopteri
2 Atractosteus spatula urn:lsid:marinespecies.org:taxname:279822 Animalia Chordata Actinopteri
3 Atractosteus spatula urn:lsid:marinespecies.org:taxname:279822 Animalia Chordata Actinopteri
```

```
4 Atractosteus spatula urn:lsid:marinespecies.org:taxname:279822 Animalia Chordata Act
5 Atractosteus spatula urn:lsid:marinespecies.org:taxname:279822 Animalia Chordata Act
6 Atractosteus spatula urn:lsid:marinespecies.org:taxname:279822 Animalia Chordata Act
              order         family        genus scientificNameAuthorship taxonRank organ
1 Lepisosteiformes Lepisosteidae Atractosteus          (Lacepède, 1803)   Species
2 Lepisosteiformes Lepisosteidae Atractosteus          (Lacepède, 1803)   Species
3 Lepisosteiformes Lepisosteidae Atractosteus          (Lacepède, 1803)   Species
4 Lepisosteiformes Lepisosteidae Atractosteus          (Lacepède, 1803)   Species
5 Lepisosteiformes Lepisosteidae Atractosteus          (Lacepède, 1803)   Species
6 Lepisosteiformes Lepisosteidae Atractosteus          (Lacepède, 1803)   Species
  organismQuantityType                                                    occurrenceID
1   Relative Abundance Station_95_Date_09JAN1997:14:35:00.000_Atractosteus_spatula Aran
2   Relative Abundance Station_95_Date_18AUG2000:11:02:00.000_Atractosteus_spatula Aran
3   Relative Abundance Station_96_Date_28JUN2005:08:41:00.000_Atractosteus_spatula Aran
4   Relative Abundance Station_96_Date_23AUG2006:11:47:00.000_Atractosteus_spatula Aran
5   Relative Abundance Station_96_Date_17OCT2006:14:23:00.000_Atractosteus_spatula Aran
6   Relative Abundance Station_96_Date_19FEB1996:10:27:00.000_Atractosteus_spatula Aran

write.csv(occurrence, file = paste0(event[1,]$datasetID, "_occurrence_",lubridate::tod
```

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 ta-
ble relations, vocabularies, and defines field names. The Darwin Core defines
three table types: event, occurrence, and measurementOrFact. This intu-
itively captures the way most ecologists conduct their research. Typically, a
survey (event) is conducted and measurements, counts, or observations (collec-
tively 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 DOI for th
                )

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.

```r
#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 =

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 =

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

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 full taxono

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

occurrence <- left_join(occurrence, species_matched, by = c("scientificName" = "ScientificName"))
  select(occurrenceID, basisOfRecord, scientificName, eventID, occurrenceStatus = occurenceStatus

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 `measurementType`s 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) %>%
```

```r
  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", "measurementO
```

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

```r
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

```r
# 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.

```r
# 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).

```r
# 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

```r
# 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, Aphia
          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 (2024). 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")
```

```r
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 variable)
  separate(substrate, sep = ",", into = c("substrate_1", "substrate_2")) %>%
  # change variables names to match NERC database (or to be more descriptive where none exist)
  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/SDBIOL02/",
      measurementType == "CanopyHeight" ~ "http://vocab.nerc.ac.uk/collection/P01/current/OBSMAXI
      # measurementType == "BarePatchWithinSeagrass" ~ "",
      measurementType == "FloweringBedAbund" ~ "http://vocab.nerc.ac.uk/collection/P01/current/SD
    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/UPMS/",
      measurementType == "CanopyHeight" ~ "http://vocab.nerc.ac.uk/collection/P06/current/ULCM/",
```

```
      measurementType == "BarePatchhLengthWithinSeagrass" ~ "http://vocab.nerc.ac.uk/cc
      measurementType == "FloweringBedAbund" ~ "http://vocab.nerc.ac.uk/collection/P06,
    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 l
      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_0.4.3    magrittr_2.0.3  knitr_1.42     here_1.0.1
##  [5] lubridate_1.9.3 forcats_1.0.0   stringr_1.5.0   dplyr_1.1.2
##  [9] purrr_1.0.1     readr_2.1.5     tidyr_1.3.0     tibble_3.2.1
## [13] ggplot2_3.4.2   tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.2.1 xfun_0.39         colorspace_2.1-0 vctrs_0.6.5
##  [5] generics_0.1.3   htmltools_0.5.5  yaml_2.3.7        utf8_1.2.3
##  [9] rlang_1.1.1      pillar_1.9.0     glue_1.6.2        httpcode_0.3.0
## [13] withr_2.5.0      bit64_4.0.5      readxl_1.4.3      lifecycle_1.0.3
## [17] munsell_0.5.0    gtable_0.3.3     cellranger_1.1.0  evaluate_0.21
## [21] tzdb_0.3.0       fastmap_1.1.1    parallel_4.1.1    curl_5.0.0
## [25] fansi_1.0.4      triebeard_0.4.1  urltools_1.7.3    Rcpp_1.0.10
## [29] scales_1.2.1     jsonlite_1.8.4   vroom_1.6.3       bit_4.0.5
## [33] hms_1.1.3        digest_0.6.31    stringi_1.7.12    bookdown_0.40
## [37] grid_4.1.1       rprojroot_2.0.4  cli_3.6.1         tools_4.1.1
## [41] crul_1.5.0       crayon_1.5.2     pkgconfig_2.0.3   timechange_0.3.0
## [45] rmarkdown_2.21   rstudioapi_0.16.0 R6_2.5.1         compiler_4.1.1
```

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 sub-setted 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 tab. This
trawl2019 <- trawl2019 %>%
  mutate(eventDate_start = format_iso_8601(as.POSIXct(paste(EVENT_DATE_START, END_DEPLOYMENT_TIME
                                                tz = "Asia/Vladivostok")),
         eventDate_start = str_replace(eventDate_start, "\\+00:00", "Z"),
         eventDate_finish = format_iso_8601(as.POSIXct(paste(EVENT_DATE_FINISH, BEGIN_RETRIEVAL_T
                                                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 = "CATCH_FINAL
  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_Specimen_data.x
                                 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) %>%
```

```r
  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$footprintWK
trawl2019_linestring <- cbind(trawl2019_coordinates, trawl2019_linestring) %>%
  select(eventID, footprintWKT, decimalLatitude, decimalLongitude, coordinateUncertain

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

```r
trawl2019_catch_ind_worms <- worrms::wm_records_names(unique(trawl2019_catch_ind$scient
trawl2019_catch_ind_occ <- left_join(trawl2019_catch_ind, trawl2019_catch_ind_worms, by
  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 exten-
sion.**

Please note that in the *overall* species-specific occurrence data frame, *individu-
alCount* 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 indi-
vidual 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`.

```r
trawl_resourceRelationship <- trawl2019_occ_ext %>%
  select(eventID, occurrenceID, scientificName) %>%
  mutate(resourceID = ifelse(grepl("sample", trawl2019_occ_ext$occurrenceID), trawl2019
  mutate(eventID = gsub(":sample.*", "", trawl2019_occ_ext$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$resourceID, numeric = TRUE)
trawl_resourceRelationship <- trawl_resourceRelationship[match(order, trawl_resourceRelationship$

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, relatedResourceID)
```

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.

2.5   dataset-edna

By Diana LaScala-Gruenewald

2.5.1   Introduction

Rationale:

DNA derived data are increasingly being used to document taxon occurrences. To ensure these data are useful to the broadest possible community, GBIF published a guide entitled "Publishing DNA-derived data through biodiversity data platforms." This guide is supported by the DNA derived data extension for Darwin Core, which incorporates MIxS terms into the Darwin Core standard.

This use case draws on both the guide and the extension to illustrate how to incorporate a DNA derived data extension file into a Darwin Core archive.

For further information on this use case and the DNA Derived data extension in general, see the recording of the OBIS Webinar on Genetic Data.

Project abstract:

The example data employed in this use case are from marine filtered seawater samples collected at a nearshore station in Monterey Bay, California, USA. They were collected by CTD rosette and filtered by a peristaltic pump system. Subsequently, they underwent metabarcoding for the 18S V9 region. The resulting ASVs, their assigned taxonomy, and the metadata associated with their collection are the input data for the conversion scripts presented here.

A selection of samples from this collection were included in the publication "Environmental DNA reveals seasonal shifts and potential interactions in a marine community" which was published with open access in *Nature Communications* in 2020.

Contacts: - Francisco Chavez - Principle Investigator (chfr@mbari.org) - Kathleen Pitz - Research Associate (kpitz@mbari.org) - Diana LaScala-Gruenewald - Point of Contact (dianalg@mbari.org)

2.5.2   Published data

- GBIF
- OBIS

2.5.3   Repo structure

```
.
+-- README.md                 :Description of this repository
+-- LICENSE                   :Repository license
+-- .gitignore                :Files and directories to be ignored by git
|
+-- raw
|   +-- asv_table.csv          :Source data containing ASV sequences and number of rea
|   +-- taxa_table.csv         :Source data containing taxon matches for each ASV
|   +-- metadata_table.csv     :Source data containing metadata about samples (e.g. co
|
+-- src
|   +-- conversion_code.py     :Darwin Core mapping script
|   +-- conversion_code.ipynb  :Darwin Core mapping Jupyter Notebook
|   +-- WoRMS.py               :Functions for querying the World Register of Marine Sp
|
+-- processed
```

```
|   +-- occurrence.csv         :Occurrence file, generated by conversion_code
|   +-- dna_extension.csv      :DNA Derived Data Extension file, generated by conversion_code
```

2.6 Converting ATN netCDF file to Dawrin Core

Created: 2022-03-23 Updated: 2023-11-16

Credit: Stephen Formel, Mathew Biddle

This notebook walks through downloading an example netCDF file from the an Archive package at NCEI and translating it to a Darwin Core Archive compliant package for easy loading and publishing via the Integrated Publishing Toolkit (IPT). The example file follows a specific specification for ATN satellite trajectory observations as documented here. More information about the ATN netCDF specification can be found in the repository https://github.com/ioos/ioos-atn-data.

This example uses the tidync package to work with netCDF data.

Data used in this notebook are available from NCEI at the following link https://www.ncei.noaa.gov/archive/accession/0282699.

```
#Load libraries

library(tidync)
library(obistools)
library(ncdf4)
library(tidyverse) #includes stringr
library(lubridate)
library(maps)
library(mapdata)
```

2.6.1 Downloading and preprocessing the source data

See https://www.ncei.noaa.gov/archive/accession/0282699

```
url = 'https://www.nodc.noaa.gov/archive/arc0217/0282699/1.1/data/0-data/atn_45866_great-white-sh

#fname = str_split_i(url, "/", i=-1)
fname = paste('data/src/',basename(url))

download.file(url, fname, mode = "wb")
```

2.6.1.1   Open the netCDF file

Once the file is opened, we print out the details of what the netCDF file contains.

```
atn <- nc_open(fname)
atn
```

```
## File data/src/ atn_45866_great-white-shark_trajectory_20090923-20091123.nc (NC_FORM
##
##      36 variables (excluding dimension variables):
##          string deploy_id[]   (Contiguous storage)
##              long_name: id for this deployment. This is typically the tag ptt
##              comment: Friendly name given to the tag by the user. If no specific fri
##              coordinates: time z lon lat
##              instrument: instrument_location
##              platform: animal
##              coverage_content_type: referenceInformation
##              _FillValue: -9999
##          double time[obs]   (Contiguous storage)
##              units: seconds since 1990-01-01 00:00:00Z
##              standard_name: time
##              axis: T
##              _CoordinateAxisType: Time
##              calendar: standard
##              long_name: Time of the measurement, in seconds since 1990-01-01
##              actual_min: 2009-09-23T00:00:00Z
##              actual_max: 2009-11-23T05:12:00Z
##              ancillary_variables: qartod_time_flag qartod_rollup_flag qartod_speed_f
##              instrument: instrument_location
##              platform: animal
##              coverage_content_type: coordinate
##              _FillValue: NaN
##          int z[obs]   (Contiguous storage)
##              _FillValue: -9999
##              axis: Z
##              long_name: depth of measurement
##              positive: down
##              standard_name: depth
##              units: m
##              actual_min: 0
##              actual_max: 0
##              instrument:
##              platform: animal
##              comment: This variable is synthetically generated to represent the dept
##              coverage_content_type: coordinate
```

```
##          double lat[obs]    (Contiguous storage)
##              axis: Y
##              _CoordinateAxisType: Lat
##              long_name: Latitude portion of location in decimal degrees North
##              standard_name: latitude
##              units: degrees_north
##              valid_max: 90
##              valid_min: -90
##              actual_min: 23.59
##              actual_max: 34.045
##              ancillary_variables: qartod_location_flag qartod_rollup_flag qartod_speed_flag err
##              instrument: instrument_location
##              platform: animal
##              coverage_content_type: coordinate
##              _FillValue: NaN
##          double lon[obs]    (Contiguous storage)
##              axis: X
##              _CoordinateAxisType: Lon
##              long_name: Longitude portion of location in decimal degrees East
##              standard_name: longitude
##              units: degrees_east
##              valid_max: 180
##              valid_min: -180
##              actual_min: -166.18
##              actual_max: -118.504
##              ancillary_variables: qartod_location_flag qartod_rollup_flag qartod_speed_flag err
##              instrument: instrument_location
##              platform: animal
##              coverage_content_type: coordinate
##              _FillValue: NaN
##          int ptt[obs]    (Chunking: [29])  (Compression: shuffle,level 1)
##              _FillValue: -9999
##              coordinates: time z lon lat
##              long_name: Platform Transmitter Terminal (PTT) id used for Argos transmissions
##              comment: PTT id for this deployment. PTT ids may be used on multiple deployments,
##              coverage_content_type: referenceInformation
##              instrument: instrument_location
##              platform: animal
##          string instrument[obs]    (Contiguous storage)
##              coordinates: time z lon lat
##              comment: Wildlife Computers instrument family. Variable may report manufacturer de
##              long_name: Instrument family
##              instrument: instrument_location
##              platform: animal
##              coverage_content_type: referenceInformation
##          string type[obs]    (Contiguous storage)
```

```
##              coordinates: time z lon lat
##              comment: Type of location: Argos, FastGPS or User
##              long_name: Type of location information - Argos, GPS satellite or user
##              instrument: instrument_location
##              platform: animal
##              coverage_content_type: referenceInformation
##          string location_class[obs]    (Contiguous storage)
##              coordinates: time z lon lat
##              standard_name: quality_flag
##              comment: Quality codes from the ARGOS satellite (in meters): G,3,2,1,0,
##              long_name: Location Quality Code from ARGOS satellite system
##              code_values: G,3,2,1,0,A,B,Z
##              code_meanings: estimated error less than 100m and 1+ messages received
##              instrument: instrument_location
##              platform: animal
##              ancillary_variables: lat lon
##              coverage_content_type: qualityInformation
##          int error_radius[obs]    (Chunking: [29])  (Compression: shuffle,level 1)
##              _FillValue: -9999
##              coordinates: time z lon lat
##              long_name: Error radius
##              units: m
##              comment: If the position is best represented as a circle, this field giv
##              instrument: instrument_location
##              platform: animal
##              ancillary_variables: lat lon offset offset_orientation
##              coverage_content_type: qualityInformation
##          int semi_major_axis[obs]    (Chunking: [29])  (Compression: shuffle,level 1)
##              _FillValue: -9999
##              coordinates: time z lon lat
##              long_name: Error - ellipse semi-major axis
##              units: m
##              comment: If the estimated position error is best expressed as an ellipse
##              instrument: instrument_location
##              platform: animal
##              ancillary_variables: lat lon ellipse_orientation offset offset_orientati
##              coverage_content_type: qualityInformation
##          int semi_minor_axis[obs]    (Chunking: [29])  (Compression: shuffle,level 1)
##              _FillValue: -9999
##              coordinates: time z lon lat
##              long_name: Error - ellipse semi-minor axis
##              units: m
##              comment: If the estimated position error is best expressed as an ellipse
##              instrument: instrument_location
##              platform: animal
##              ancillary_variables: lat lon ellipse_orientation offset offset_orientati
```

```
##              coverage_content_type: qualityInformation
##         int ellipse_orientation[obs]   (Chunking: [29])  (Compression: shuffle,level 1)
##             _FillValue: -9999
##             coordinates: time z lon lat
##             long_name: Error - ellipse orientation in degrees clockwise from true north
##             units: degrees
##             comment: The angle in degrees of the ellipse from true north, proceeding clockwise
##             instrument: instrument_location
##             platform: animal
##             ancillary_variables: lat lon semi_major_axis semi_minor_axis offset offset_orienta
##             coverage_content_type: qualityInformation
##         int offset[obs]   (Chunking: [29])  (Compression: shuffle,level 1)
##             _FillValue: -9999
##             coordinates: time z lon lat
##             long_name: Error - offset in meters to center of error ellipse or circle
##             units: m
##             comment: This field is non-zero if the circle or ellipse are not centered on the (
##             instrument: instrument_location
##             platform: animal
##             ancillary_variables: lat lon error_radius semi_major_axis semi_minor_axis offset_c
##             coverage_content_type: qualityInformation
##         int offset_orientation[obs]   (Chunking: [29])  (Compression: shuffle,level 1)
##             _FillValue: -9999
##             coordinates: time z lon lat
##             long_name: Error - offset orientation angle to ellipse center
##             units: degrees
##             comment: If the "Offset" field is non-zero, this field is the angle in degrees fro
##             instrument: instrument_location
##             platform: animal
##             ancillary_variables: lat lon error_radius semi_major_axis semi_minor_axis offset
##             coverage_content_type: qualityInformation
##         double gpe_msd[obs]   (Chunking: [29])  (Compression: shuffle,level 1)
##             coordinates: time z lon lat
##             comment: Historical. No longer applicable.
##             long_name:
##             units:
##             instrument: instrument_location
##             platform: animal
##             coverage_content_type: auxillaryInformation
##             _FillValue: NaN
##         double gpe_u[obs]   (Chunking: [29])  (Compression: shuffle,level 1)
##             coordinates: time z lon lat
##             comment: Historical. No longer applicable.
##             long_name:
##             units:
##             instrument: instrument_location
```

```
##                    platform: animal
##                    coverage_content_type: auxillaryInformation
##                    _FillValue: NaN
##            int count[obs]    (Chunking: [29])   (Compression: shuffle,level 1)
##                    _FillValue: -9999
##                    coordinates: time z lon lat
##                    comment: Total number of times a particular data item was received, ver:
##                    long_name: Count
##                    units: count
##                    instrument: instrument_location
##                    platform: animal
##                    coverage_content_type: auxillaryInformation
##            unsigned byte qartod_time_flag[obs]    (Chunking: [29])   (Compression: shuffl
##                    _FillValue: 241
##                    coordinates: time z lon lat
##                    standard_name: gross_range_test_quality_flag
##                    long_name: Time QC test - gross range test
##                    implementation: https://github.com/ioos/ioos_qc/
##                    flag_meanings: PASS NOT_EVALUATED SUSPECT FAIL MISSING
##                    flag_values: 1
##                     flag_values: 2
##                     flag_values: 3
##                     flag_values: 4
##                     flag_values: 9
##                    references: https://cdn.ioos.noaa.gov/media/2020/03/QARTOD_TS_Manual_Up
##                    coverage_content_type: qualityInformation
##            unsigned byte qartod_speed_flag[obs]    (Chunking: [29])   (Compression: shuf
##                    _FillValue: 241
##                    coordinates: time z lon lat
##                    standard_name: gross_range_test_quality_flag
##                    long_name: Speed QC test - gross range test
##                    references: https://cdn.ioos.noaa.gov/media/2020/03/QARTOD_TS_Manual_Up
##                    implementation: https://github.com/ioos/ioos_qc/
##                    flag_meanings: PASS NOT_EVALUATED SUSPECT FAIL MISSING
##                    flag_values: 1
##                     flag_values: 2
##                     flag_values: 3
##                     flag_values: 4
##                     flag_values: 9
##                    coverage_content_type: qualityInformation
##            unsigned byte qartod_location_flag[obs]    (Chunking: [29])   (Compression: s
##                    _FillValue: 241
##                    coordinates: time z lon lat
##                    standard_name: location_test_quality_flag
##                    long_name: Location QC test - Location test
##                    implementation: https://github.com/ioos/ioos_qc/
```

```
##                 flag_meanings: PASS NOT_EVALUATED SUSPECT FAIL MISSING
##                 flag_values: 1
##                  flag_values: 2
##                  flag_values: 3
##                  flag_values: 4
##                  flag_values: 9
##                 references: https://cdn.ioos.noaa.gov/media/2020/03/QARTOD_TS_Manual_Update2_20032
##                 coverage_content_type: qualityInformation
##             unsigned byte qartod_rollup_flag[obs]   (Chunking: [29])  (Compression: shuffle,level
##                 _FillValue: 241
##                 coordinates: time z lon lat
##                 standard_name: aggregate_quality_flag
##                 long_name: Aggregate QC value
##                 implementation: https://github.com/ioos/ioos_qc/
##                 flag_meanings: PASS NOT_EVALUATED SUSPECT FAIL MISSING
##                 flag_values: 1
##                  flag_values: 2
##                  flag_values: 3
##                  flag_values: 4
##                  flag_values: 9
##                 references: https://cdn.ioos.noaa.gov/media/2020/03/QARTOD_TS_Manual_Update2_20032
##                 coverage_content_type: qualityInformation
##             int crs[]   (Contiguous storage)
##                 epsg_code: EPSG:4326
##                 grid_mapping_name: latitude_longitude
##                 inverse_flattening: 298.257223563
##                 long_name: Coordinate Reference System - http://www.opengis.net/def/crs/EPSG/0/432
##                 semi_major_axis: 6378137
##                 coverage_content_type: referenceInformation
##             string trajectory[]   (Contiguous storage)
##                 cf_role: trajectory_id
##                 long_name: trajectory identifier
##             int animal_age[]   (Contiguous storage)
##                 _FillValue: -9999
##                 units:
##                 long_name: age of the animal as measured or estimated at deployment
##                 coverage_content_type: referenceInformation
##                 animal_age: Not provided
##             string animal_life_stage[]   (Contiguous storage)
##                 animal_life_stage: juvenile
##                 long_name: Lifestage of the animal at time of deployment
##                 coverage_content_type: referenceInformation
##             string animal_sex[]   (Contiguous storage)
##                 animal_sex: male
##                 long_name: sex of the animal at time of tag deployment
##                 coverage_content_type: referenceInformation
```

```
##          float animal_weight[]   (Contiguous storage)
##              _FillValue: NaN
##              units: kg
##              long_name: mass of the animal as measured or estimated at deployment
##              animal_weight: Not provided
##              coverage_content_type: referenceInformation
##          float animal_length[]   (Contiguous storage)
##              _FillValue: NaN
##              animal_length_type: total length
##              units: cm
##              animal_length: 213.0 (cm) total length
##              long_name: length of the animal as measured or estimated at deployment
##              coverage_content_type: referenceInformation
##          float animal_length_2[]   (Contiguous storage)
##              _FillValue: NaN
##              animal_length_2_type: Not provided
##              units:
##              animal_length_2: Not provided
##              long_name: length of the animal as measured or estimated at deployment
##              coverage_content_type: referenceInformation
##          string animal[]   (Contiguous storage)
##              rank: Species
##              infraorder:
##              scientificname: Carcharodon carcharias
##              long_name: tagged animal id
##              superdomain: Biota
##              order: Lamniformes
##              authority: (Linnaeus, 1758)
##              kingdom: Animalia
##              species: Carcharodon carcharias
##              genus: Carcharodon
##              megaclass:
##              family: Lamnidae
##              taxonRankID: 220
##              class: Elasmobranchii
##              cf_role: trajectory_id
##              coverage_content_type: referenceInformation
##              subphylum: Vertebrata
##              phylum: Chordata
##              AphiaID: 105838
##              valid_name: Carcharodon carcharias
##              infraphylum: Gnathostomata
##              subclass: Neoselachii
##              suborder:
##          string instrument_tag[]   (Contiguous storage)
##              manufacturer: Wildlife Computers
```

```
##              make_model: SPOT5
##              serial_number: 07S0230
##              long_name: telemetry tag applied to animal
##              coverage_content_type: referenceInformation
##              calibration_date: Not Provided
##         string instrument_location[]   (Contiguous storage)
##              manufacturer: Wildlife Computers
##              make_model: SPOT5
##              serial_number: 07S0230
##              long_name: Wildlife Computers SPOT5
##              location_type: argos / modeled
##              comment: Location
##              coverage_content_type: referenceInformation
##              calibration_date: Not Provided
##         string taxon_name[]   (Contiguous storage)
##              standard_name: biological_taxon_name
##              long_name: most precise taxonomic classification for the tagged animal
##              coverage_content_type: referenceInformation
##              source: Froese, R. and D. Pauly. Editors. (2023). FishBase. Carcharodon carcharias
##              url: https://www.marinespecies.org/aphia.php?p=taxdetails&id=105838
##         string taxon_lsid[]   (Contiguous storage)
##              standard_name: biological_taxon_lsid
##              long_name: Namespaced Taxon Identifier for the tagged animal
##              coverage_content_type: referenceInformation
##              source: Froese, R. and D. Pauly. Editors. (2023). FishBase. Carcharodon carcharias
##              url: https://www.marinespecies.org/aphia.php?p=taxdetails&id=105838
##         string comment[obs]   (Contiguous storage)
##              long_name: Comment
##              comment: Optional text field
##              coordinates: time z lon lat
##              instrument: instrument_location
##              platform: animal
##              coverage_content_type: auxillaryInformation
##
##     1 dimensions:
##         obs   Size:29 (no dimvar)
##
##     89 global attributes:
##         date_created: 2023-08-16T20:00:00Z
##         featureType: trajectory
##         cdm_data_type: Trajectory
##         Conventions: CF-1.10, ACDD-1.3, IOOS-1.2
##         argos_program_number: 2414
##         creator_email: chris.lowe@csulb.edu
##         id: 5f0668a86321be13bc7ef628
##         tag_type: SPOT5
```

```
##          source: Service Argos
##          acknowledgement: NOAA IOOS, Axiom Data Science, Navy ONR, NOAA NMFS, Wildli
##          creator_name: Chris G. Lowe
##          creator_url:
##          geospatial_lat_units: degrees_north
##          geospatial_lon_units: degrees_east
##          infoUrl: https://portal.atn.ioos.us/#metadata/6e2ba85c-2f61-4bc5-8c2b-34d67
##          institution: California State University Long Beach
##          keywords: EARTH SCIENCE > AGRICULTURE > ANIMAL SCIENCE > ANIMAL ECOLOGY AND
##          license: These data may be used and redistributed for free, but are not inte
##          metadata_link:
##          naming_authority: com.wildlifecomputers
##          platform_category: animal
##          platform: fish
##          platform_vocabulary: https://vocab.nerc.ac.uk/collection/L06/current/
##          processing_level: NetCDF file created from position data obtained from Wildl
##          project: Project White Shark: Juvenile Satellite Biotelemetry, 2001-2020
##          publisher_email: atndata@ioos.us
##          publisher_institution: US Integrated Ocean Observing System Office
##          publisher_name: US Integrated Ocean Observing System (IOOS) Animal Telemetry
##          publisher_url: https://atn.ioos.us
##          publisher_country: USA
##          standard_name_vocabulary: CF-v78
##          vendor: Wildlife Computers
##          geospatial_lat_min: 23.59
##          geospatial_lat_max: 34.045
##          geospatial_lon_min: -166.18
##          geospatial_lon_max: -118.504
##          geospatial_bbox: POLYGON ((-118.504 23.59, -118.504 34.045, -166.18 34.045,
##          geospatial_bounds: POLYGON ((-166.18 23.59, -118.581 34.038, -118.53 34.045
##          geospatial_bounds_crs: EPSG:4326
##          time_coverage_start: 2009-09-23T00:00:00Z
##          time_coverage_end: 2009-11-23T05:12:00Z
##          time_coverage_duration: P61DT5H12M0S
##          time_coverage_resolution: P2DT2H39M43S
##          date_issued: 2023-08-16T20:00:00Z
##          date_modified: 2023-08-16T20:00:00Z
##          history: 2023-08-07T20:24:04Z - Created by the IOOS ATN DAC from the Wildli
##          summary: Wildlife Computers SPOT5 tag (ptt id 45866) deployed on a great wh
##          title: Great white shark (Carcharodon carcharias) location data from a sate
##          uuid: ff554ebf-bf4b-5a82-8a90-9c0ceb799d96
##          platform_name: Carcharodon carcharias
##          platform_id: 105838
##          vendor_id: 5f0668a86321be13bc7ef628
##          sea_name: North Pacific Ocean
##          arbitrary_keywords: ATN, Animal Telemetry Network, IOOS, Integrated Ocean OU
```

```
##          contributor_role_vocabulary: https://vocab.nerc.ac.uk/collection/G04/current/
##          creator_role_vocabulary: https://vocab.nerc.ac.uk/collection/G04/current/
##          creator_sector_vocabulary: https://mmisw.org/ont/ioos/sector
##          creator_type: person
##          date_metadata_modified: 20230816
##          instrument: Satellite telemetry tag
##          instrument_vocabulary:
##          keywords_vocabulary: GCMD Science Keywords v15.1
##          ncei_template_version: NCEI_NetCDF_Trajectory_Template_v2.0
##          product_version:
##          program: IOOS Animal Telemetry Network
##          publisher_type: institution
##          references:
##          animal_common_name: great white shark
##          animal_id: 09_13
##          animal_scientific_name: Carcharodon carcharias
##          deployment_id: 5f0668a86321be13bc7ef628
##          deployment_start_datetime: 2009-09-23T00:00:00Z
##          deployment_end_datetime: 2009-11-23T00:00:00Z
##          wmo_platform_code:
##          comment: 09_13-45866
##          ptt_id: 45866
##          deployment_start_lat: 34.03
##          deployment_start_lon: -118.56
##          contributor_name: Thomas Farrugia
##          contributor_email: tjfarrugia@alaska.edu
##          contributor_role: collaborator
##          contributor_institution: California State University Long Beach
##          contributor_url:
##          creator_role: principalInvestigator
##          creator_sector: academic
##          creator_country: USA
##          creator_institution: California State University Long Beach
##          creator_institution_url: https://www.csulb.edu/shark-lab
##          citation: Lowe, Chris G.; Farrugia, Thomas. (2023) great white shark (Carcharodon carc
```

2.6.1.2 Collect all the metadata from the netCDF file.

This gathers not only the global attributes, but the variable level attributes
as well. As you can see in the **variable** column the term `NC_GLOBAL` refers to
global attributes.

```
metadata <- ncmeta::nc_atts(fname)
metadata
```

```
## # A tibble: 381 x 4
##       id name                   variable  value
##    <int> <chr>                  <chr>     <named list>
## 1      0 long_name              deploy_id <chr [1]>
## 2      1 comment                deploy_id <chr [1]>
## 3      2 coordinates            deploy_id <chr [1]>
## 4      3 instrument             deploy_id <chr [1]>
## 5      4 platform               deploy_id <chr [1]>
## 6      5 coverage_content_type  deploy_id <chr [1]>
## 7      6 _FillValue             deploy_id <dbl [1]>
## 8      0 units                  time      <chr [1]>
## 9      1 standard_name          time      <chr [1]>
## 10     2 axis                   time      <chr [1]>
## # i 371 more rows
```

2.6.1.3   Store the data as a tibble

Collect the data dimensioned by `time` from the netCDF file as a tibble. Then,
print the first ten rows.

```
atn <- tidync(fname)

atn_tbl <- atn %>% hyper_tibble(force=TRUE)

head(atn_tbl, n=4)
```

```
## # A tibble: 4 x 23
##         time     z   lat   lon   ptt instrument type  location_class error_radius
##        <dbl> <int> <dbl> <dbl> <int> <chr>      <chr> <chr>                 <int>
## 1 622512000     0  34.0 -119. 45866 SPOT        User  nan                      NA
## 2 622708920     0  23.6 -166. 45866 SPOT        Argos A                        NA
## 3 622724940     0  34.0 -119. 45866 SPOT        Argos 1                        NA
## 4 622725060     0  34.0 -119. 45866 SPOT        Argos 0                        NA
## # i 14 more variables: semi_major_axis <int>, semi_minor_axis <int>,
## #   ellipse_orientation <int>, offset <int>, offset_orientation <int>,
## #   gpe_msd <dbl>, gpe_u <dbl>, count <int>, qartod_time_flag <int>,
## #   qartod_speed_flag <int>, qartod_location_flag <int>,
## #   qartod_rollup_flag <int>, comment <chr>, obs <int>
```

2.6.1.4   Dealing with time

Notice the data in the **time** column aren't formatted as times. We need to read
the metadata associated with the time variable to understand what the units
are. Below, we print a tibble of all the attributes from the **time** variable.

Notice the *units* attribute and it's value of `seconds since 1990-01-01 00:00:00Z`. We need to use that information to convert the time variable to something useful that `ggplot` can handle.

```
time_attrs <- metadata %>% dplyr::filter(variable == "time")
time_attrs
```

```
## # A tibble: 13 x 4
##       id name                    variable value
##    <int> <chr>                   <chr>    <named list>
## 1      0 units                   time     <chr [1]>
## 2      1 standard_name           time     <chr [1]>
## 3      2 axis                    time     <chr [1]>
## 4      3 _CoordinateAxisType     time     <chr [1]>
## 5      4 calendar                time     <chr [1]>
## 6      5 long_name               time     <chr [1]>
## 7      6 actual_min              time     <chr [1]>
## 8      7 actual_max              time     <chr [1]>
## 9      8 ancillary_variables     time     <chr [1]>
## 10     9 instrument              time     <chr [1]>
## 11    10 platform                time     <chr [1]>
## 12    11 coverage_content_type   time     <chr [1]>
## 13    12 _FillValue              time     <dbl [1]>
```

So, we grab the value from the `units` attribute, split the string to collect the date information, and apply that to a time conversion function `as.POSIXct`.

```
#library(stringr) - loaded with tidyverse
# grab origin date from time variable units attribute
tunit <- time_attrs %>% dplyr::filter(name == "units")
lunit <- str_split(tunit$value,' ')[[1]]
atn_tbl$time <- as.POSIXct(atn_tbl$time, origin=lunit[3], tz="GMT")

str(atn_tbl)
```

```
## tibble [29 x 23] (S3: tbl_df/tbl/data.frame)
##  $ time           : POSIXct[1:29], format: "2009-09-23 00:00:00" "2009-09-25 06:42:00" ..
##  $ z              : int [1:29] 0 0 0 0 0 0 0 0 0 0 ...
##  $ lat            : num [1:29] 34 23.6 34 34 34 ...
##  $ lon            : num [1:29] -119 -166 -119 -119 -119 ...
##  $ ptt            : int [1:29] 45866 45866 45866 45866 45866 45866 45866 45866 45866 4586
##  $ instrument     : chr [1:29] "SPOT" "SPOT" "SPOT" "SPOT" ...
##  $ type           : chr [1:29] "User" "Argos" "Argos" "Argos" ...
##  $ location_class : chr [1:29] "nan" "A" "1" "0" ...
```

```
##  $ error_radius       : int [1:29] NA NA NA NA NA NA NA NA NA NA ...
##  $ semi_major_axis    : int [1:29] NA NA NA NA NA NA NA NA NA NA ...
##  $ semi_minor_axis    : int [1:29] NA NA NA NA NA NA NA NA NA NA ...
##  $ ellipse_orientation : int [1:29] NA NA NA NA NA NA NA NA NA NA ...
##  $ offset             : int [1:29] NA NA NA NA NA NA NA NA NA NA ...
##  $ offset_orientation : int [1:29] NA NA NA NA NA NA NA NA NA NA ...
##  $ gpe_msd            : num [1:29] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##  $ gpe_u              : num [1:29] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##  $ count              : int [1:29] NA NA NA NA NA NA NA NA NA NA ...
##  $ qartod_time_flag   : int [1:29] 1 1 1 1 1 1 1 1 1 1 ...
##  $ qartod_speed_flag  : int [1:29] 2 4 4 4 1 1 1 1 1 1 ...
##  $ qartod_location_flag: int [1:29] 1 1 1 1 1 1 1 1 1 1 ...
##  $ qartod_rollup_flag : int [1:29] 1 4 4 4 1 1 1 1 1 1 ...
##  $ comment            : chr [1:29] "" "" "" "" ...
##  $ obs                : int [1:29] 1 2 3 4 5 6 7 8 9 10 ...
```

2.6.2   Converting to Darwin Core

Now let's work through converting this netCDF file to Darwin Core. Following the guidance published at https://github.com/tdwg/dwc-for-biologging/wiki/Data-guidelines and https://github.com/ocean-tracking-network/biologging_standardization/tree/master/examples/braun-blueshark/darwincore-example

2.6.2.1   Occurrence Core

Below is the mapping table from DarwinCore to the netCDF file.

DarwinCore Term	Status	netCDF source
occurrenceStatus	Required	hardcoded to `present`.
basisOfRecord	Required	data contained in the `type` variable where `type` of `User` = `HumanObservation` and `Argos` = `MachineObservation`.
occurrenceID	Required	`eventDate`, plus data contained in `z` variable, plus `animal_common_name` global attribute.

DarwinCore Term	Status	netCDF source
organismID	Required	`platform_id` global attribute plus the `animal_common_name` global attribute.
eventDate	Required	data contained in `time` variable. Converted to ISO8601.
decimalLatitude & decimalLongitude	Required	data in `lat` and `lon` variable, respectively.
geodeticDatum	Required	attribute `epsg_code` in the `crs` variable.
scientificName	Required	data from the variable `taxon_name`.
scientificNameID		data from the variable `taxon_lsid`.
eventID	Strongly recommended	`animal_common_name` global attribute plus the `eventDate`.
samplingProtocol	Strongly recommended	
kingdom	Strongly recommended	`kingdom` attribute in the `animal` variable.
taxonRank	Strongly recommended	`rank` attribute in the `animal` variable.
coordinateUncertaintyInMeters	Share if available	maximum value of the data from the variables `error_radius`, `semi_major_axis`, and `offset`.
lifeStage	Share if available	data from the variable `animal_life_stage`.
sex	Share if available	data from the variable `animal_sex`.

Now start working through the crosswalk. A few thoughts about some of the functions we use:

1. `case_when` is a function from dplyr that is essentially a 'vectorized' ifelse function. The take-home is that it plays nice with other `tidyverse` functions, like `mutate` and IMO is a bit more readable than a complex ifelse statement.

2. `rename` is another nice dplyr function for renaming columns. It workes well following `mutate` because you can see the mutation applied to a column

and then the column renamed, rather than a complex creation of a new
column and dropping of the old column.

```r
# Defined to grab attributes in subsequent code
nc <- nc_open(fname)

occurrencedf <- atn_tbl %>%
    select( # Select desired columns

        time,
        lat,
        lon,
        type,
        location_class,
        qartod_time_flag,
        qartod_speed_flag,
        qartod_location_flag,
        qartod_rollup_flag

        ) %>%
    mutate( # add and mutate columns.

        type = case_when(type == 'User' ~ 'HumanObservation',
                         type == 'Argos' ~ 'MachineObservation'),

        time = format(time, '%Y-%m-%dT%H:%M:%SZ'),

        kingdom = metadata %>% dplyr::filter(variable == "animal" & name == "kingdom")

        taxonRank = metadata %>% dplyr::filter(variable == "animal" & name == "rank") 

        occurrenceStatus = "present",

        sex = ncvar_get( nc, 'animal_sex'),

        lifeStage = ncvar_get( nc, 'animal_life_stage'),

        scientificName = ncvar_get( nc, 'taxon_name'),

        scientificNameID = ncvar_get( nc, "taxon_lsid")

        ) %>%

    rename(  # rename columns to Darwin Core terms
```

```
        basisOfRecord = type,
        eventDate = time,
        decimalLatitude = lat,
        decimalLongitude = lon) %>%

    arrange(eventDate) #arrange by increasing date

# minimumDepthInMeters = z,
occurrencedf$minimumDepthInMeters = atn_tbl$z

# maximumDepthInMeters = z,
occurrencedf$maximumDepthInMeters = atn_tbl$z

# organismID - {platformID}_{common_name}
common_name_tbl <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name ==
common_name <- chartr(" ", "_", common_name_tbl$value)
platform_id_tbl <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name ==
platform_id <- chartr(" ", "_", platform_id_tbl$value)
occurrencedf$organismID <- paste(platform_id , common_name, sep = "_")

# occurrenceID - {eventDate}_{depth}_{common_name}
occurrencedf$occurrenceID <- sub(" ", "_", paste(occurrencedf$eventDate, atn_tbl$z, common_name,

# geodeticDatum
gd_tbl <- metadata %>% dplyr::filter(variable == "crs") %>% dplyr::filter(name == "epsg_code")
occurrencedf$geodeticDatum <- paste(gd_tbl$value)

# eventID
#eventID - {common_name}_{dateTime}
cname = metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name == "animal_com
occurrencedf$eventID <- sub(" ", "_", paste0(cname$value, "_", occurrencedf$eventDate))
```

```
str(occurrencedf)
```

```
## tibble [29 x 22] (S3: tbl_df/tbl/data.frame)
##  $ eventDate          : chr [1:29] "2009-09-23T00:00:00Z" "2009-09-25T06:42:00Z" "2009-09-25T
##  $ decimalLatitude    : num [1:29] 34 23.6 34 34 34 ...
##  $ decimalLongitude   : num [1:29] -119 -166 -119 -119 -119 ...
##  $ basisOfRecord      : chr [1:29] "HumanObservation" "MachineObservation" "MachineObservatio
##  $ location_class     : chr [1:29] "nan" "A" "1" "0" ...
##  $ qartod_time_flag   : int [1:29] 1 1 1 1 1 1 1 1 1 1 ...
##  $ qartod_speed_flag  : int [1:29] 2 4 4 4 1 1 1 1 1 1 ...
##  $ qartod_location_flag: int [1:29] 1 1 1 1 1 1 1 1 1 1 ...
##  $ qartod_rollup_flag : int [1:29] 1 4 4 4 1 1 1 1 1 1 ...
##  $ kingdom            : chr [1:29] "Animalia" "Animalia" "Animalia" "Animalia" ...
```

```
## $ taxonRank          : chr [1:29] "Species" "Species" "Species" "Species" ...
## $ occurrenceStatus   : chr [1:29] "present" "present" "present" "present" ...
## $ sex                : chr [1:29] "male" "male" "male" "male" ...
## $ lifeStage          : chr [1:29] "juvenile" "juvenile" "juvenile" "juvenile" ...
## $ scientificName     : chr [1:29] "Carcharodon carcharias" "Carcharodon carcharias
## $ scientificNameID   : chr [1:29] "urn:lsid:marinespecies.org:taxname:105838" "ur
## $ minimumDepthInMeters: int [1:29] 0 0 0 0 0 0 0 0 0 0 ...
## $ maximumDepthInMeters: int [1:29] 0 0 0 0 0 0 0 0 0 0 ...
## $ organismID         : chr [1:29] "105838_great_white_shark" "105838_great_white_s
## $ occurrenceID       : chr [1:29] "2009-09-23T00:00:00Z_0_great_white_shark" "2009
## $ geodeticDatum      : chr [1:29] "EPSG:4326" "EPSG:4326" "EPSG:4326" "EPSG:4326"
## $ eventID            : chr [1:29] "great_white shark_2009-09-23T00:00:00Z" "great_
```

2.6.2.1.1 Add coordinateUncertaintyInMeters AND filter by location_class

When we add `coordinateUncertaintyInMeters` we are also filtering out where `location_class` == `A`,`B`,or `Z`.

In these data we also have additional information about the Location Quality Code from ARGOS satellite system. Below are the codes and those meanings.

code_values	code meanings
G	estimated error less than 100m and 1+ messages received per satellite pass
3	estimated error less than 250m and 4+ messages received per satellite pass
2	estimated error between 250m and 500m and 4+ messages per satellite pass
1	estimated error between 500m and 1500m and 4+ messages per satellite pass
0	estimated error greater than 1500m and 4+ messages received per satellite pass
A	no least squares estimated error or unbounded kalman filter estimated error and 3 messages received per satellite pass
B	no least squares estimated error or unbounded kalman filter estimated error and 1 or 2 messages received per satellite pass
Z	invalid location (available for Service Plus or Auxilliary Location Processing)

Since codes `A`, `B`, and `Z` are essentially bad values, I propose that we filter those out.

Also, create a mapping table for `coordinateUncertaintyInMeters` that corresponds to the ARGOS code maximum error as shown in the table below:

code	coordinateUncertaintyInMeters
G	100
3	250
2	500
1	1500
0	10000 (ref)

Below we create a lookup table for the `location_class` values we agree are good, which contains the `coordinateUncertaintyInMeters` for the appropriate location class. When we merge that table with our raw data, the observations that don't match the location_classes in our lookup table will not be transfered over (ie. they will be filtered out).

```r
occurrencedf <- occurrencedf %>%
    filter(location_class %in% c('nan','G','3','2','1','0')) %>%
    mutate(  # This returns NA for any other values than those defined below
        coordinateUncertaintyInMeters = case_when(location_class == 'nan' ~ 0,
                                                  location_class == 'G' ~ 200,
                                                  location_class == '3' ~ 250,
                                                  location_class == '2' ~ 500,
                                                  location_class == '1' ~ 1500,
                                                  location_class == '0' ~ 10000) # https://git
        ) %>%
    arrange(eventDate) # arrange by increasing date


occurrencedf
```

```
## # A tibble: 19 x 23
##    eventDate      decimalLatitude decimalLongitude basisOfRecord location_class
##    <chr>                    <dbl>            <dbl> <chr>         <chr>
##  1 2009-09-23T00:~           34.0            -119. HumanObserva~ nan
##  2 2009-09-25T11:~           34.0            -119. MachineObser~ 1
##  3 2009-09-25T11:~           34.0            -119. MachineObser~ 0
##  4 2009-09-27T17:~           34.0            -119. MachineObser~ 1
##  5 2009-10-08T20:~           34.0            -119. MachineObser~ 2
##  6 2009-10-15T11:~           34.0            -119. MachineObser~ 0
##  7 2009-10-17T06:~           34.0            -119. MachineObser~ 0
##  8 2009-10-17T09:~           34.0            -119. MachineObser~ 2
##  9 2009-10-17T10:~           34.0            -119. MachineObser~ 3
## 10 2009-10-18T08:~           34.0            -119. MachineObser~ 1
```

```
## 11 2009-10-18T10:~          34.0               -119. MachineObser~ 2
## 12 2009-10-18T11:~          34.0               -119. MachineObser~ 0
## 13 2009-10-23T23:~          34.0               -119. MachineObser~ 2
## 14 2009-10-24T00:~          34.0               -119. MachineObser~ 0
## 15 2009-10-26T10:~          34.0               -119. MachineObser~ 3
## 16 2009-10-27T16:~          34.0               -119. MachineObser~ 1
## 17 2009-10-27T16:~          34.0               -119. MachineObser~ 2
## 18 2009-10-29T11:~          34.0               -119. MachineObser~ 2
## 19 2009-10-31T21:~          34.0               -119. MachineObser~ 0
## # i 18 more variables: qartod_time_flag <int>, qartod_speed_flag <int>,
## #   qartod_location_flag <int>, qartod_rollup_flag <int>, kingdom <chr>,
## #   taxonRank <chr>, occurrenceStatus <chr>, sex <chr>, lifeStage <chr>,
## #   scientificName <chr>, scientificNameID <chr>, minimumDepthInMeters <int>,
## #   maximumDepthInMeters <int>, organismID <chr>, occurrenceID <chr>,
## #   geodeticDatum <chr>, eventID <chr>, coordinateUncertaintyInMeters <dbl>
```

Notice how we went from 29 rows down to 19 rows by only selecting specific the location_class.

2.6.2.1.2 Create a dataGeneralizations column to describe how many duplicates were found for each deprecation series Add a dataGeneralizations column containing a string like 'first of # records' to indicate there are more records in the raw dataset to be discovered by the super-curious.

The dataGeneralizations string is compiled by counting the number of consecutive duplicates and inserting that into a standard string. That string is "first of [n] records" which will make more sense once we've filtered down to keep the first occurrence of the hour.

The next step below this, we filter out only the first observation of the hour.

```
# sort by date
occurrencedf <- occurrencedf %>% arrange(eventDate)

occurrencedf <- occurrencedf %>%
    mutate(eventDateHrs = format(as.POSIXct(eventDate, format="%Y-%m-%dT%H:%M:%SZ"),"%
            ) %>%
    add_count(eventDateHrs) %>%
    mutate(dataGeneralizations = case_when(n == 1 ~ "",
                                            TRUE ~ paste("first of ", n ,"records")
                                            )
            ) %>%
    select(-n)

occurrencedf
```

```
## # A tibble: 19 x 25
##    eventDate        decimalLatitude decimalLongitude basisOfRecord location_class
##    <chr>                     <dbl>            <dbl> <chr>         <chr>
##  1 2009-09-23T00:~            34.0            -119. HumanObserva~ nan
##  2 2009-09-25T11:~            34.0            -119. MachineObser~ 1
##  3 2009-09-25T11:~            34.0            -119. MachineObser~ 0
##  4 2009-09-27T17:~            34.0            -119. MachineObser~ 1
##  5 2009-10-08T20:~            34.0            -119. MachineObser~ 2
##  6 2009-10-15T11:~            34.0            -119. MachineObser~ 0
##  7 2009-10-17T06:~            34.0            -119. MachineObser~ 0
##  8 2009-10-17T09:~            34.0            -119. MachineObser~ 2
##  9 2009-10-17T10:~            34.0            -119. MachineObser~ 3
## 10 2009-10-18T08:~            34.0            -119. MachineObser~ 1
## 11 2009-10-18T10:~            34.0            -119. MachineObser~ 2
## 12 2009-10-18T11:~            34.0            -119. MachineObser~ 0
## 13 2009-10-23T23:~            34.0            -119. MachineObser~ 2
## 14 2009-10-24T00:~            34.0            -119. MachineObser~ 0
## 15 2009-10-26T10:~            34.0            -119. MachineObser~ 3
## 16 2009-10-27T16:~            34.0            -119. MachineObser~ 1
## 17 2009-10-27T16:~            34.0            -119. MachineObser~ 2
## 18 2009-10-29T11:~            34.0            -119. MachineObser~ 2
## 19 2009-10-31T21:~            34.0            -119. MachineObser~ 0
## # i 20 more variables: qartod_time_flag <int>, qartod_speed_flag <int>,
## #   qartod_location_flag <int>, qartod_rollup_flag <int>, kingdom <chr>,
## #   taxonRank <chr>, occurrenceStatus <chr>, sex <chr>, lifeStage <chr>,
## #   scientificName <chr>, scientificNameID <chr>, minimumDepthInMeters <int>,
## #   maximumDepthInMeters <int>, organismID <chr>, occurrenceID <chr>,
## #   geodeticDatum <chr>, eventID <chr>, coordinateUncertaintyInMeters <dbl>,
## #   eventDateHrs <chr>, dataGeneralizations <chr>
```

2.6.2.1.3 Decimate occurrences down to the first detection/location per hour   Here we've done the decimation in Python: https://gist.github.com/MathewBiddle/d434ac2b538b2728aa80c6a7945f94be

Essentially we build a new colum that is the date plus the two digit hour. Then we find where that column has duplicates and keep the first entry.

In R, we do something slightly different as we only keep the distinct (ie. unique) rows and if there are duplicates, pick the first row of the duplicate.

```r
# sort by date
occurrencedf_dec <- occurrencedf %>% arrange(eventDate)

# filter table to only unique date + hour and pick the first row.
occurrencedf_dec <- distinct(occurrencedf_dec,eventDateHrs,.keep_all = TRUE) %>%
    select(-eventDateHrs)
```

`occurrencedf_dec`

```
## # A tibble: 17 x 24
##    eventDate      decimalLatitude decimalLongitude basisOfRecord location_class
##    <chr>                    <dbl>            <dbl> <chr>         <chr>
##  1 2009-09-23T00:~           34.0           -119. HumanObserva~ nan
##  2 2009-09-25T11:~           34.0           -119. MachineObser~ 1
##  3 2009-09-27T17:~           34.0           -119. MachineObser~ 1
##  4 2009-10-08T20:~           34.0           -119. MachineObser~ 2
##  5 2009-10-15T11:~           34.0           -119. MachineObser~ 0
##  6 2009-10-17T06:~           34.0           -119. MachineObser~ 0
##  7 2009-10-17T09:~           34.0           -119. MachineObser~ 2
##  8 2009-10-17T10:~           34.0           -119. MachineObser~ 3
##  9 2009-10-18T08:~           34.0           -119. MachineObser~ 1
## 10 2009-10-18T10:~           34.0           -119. MachineObser~ 2
## 11 2009-10-18T11:~           34.0           -119. MachineObser~ 0
## 12 2009-10-23T23:~           34.0           -119. MachineObser~ 2
## 13 2009-10-24T00:~           34.0           -119. MachineObser~ 0
## 14 2009-10-26T10:~           34.0           -119. MachineObser~ 3
## 15 2009-10-27T16:~           34.0           -119. MachineObser~ 1
## 16 2009-10-29T11:~           34.0           -119. MachineObser~ 2
## 17 2009-10-31T21:~           34.0           -119. MachineObser~ 0
## # i 19 more variables: qartod_time_flag <int>, qartod_speed_flag <int>,
## #   qartod_location_flag <int>, qartod_rollup_flag <int>, kingdom <chr>,
## #   taxonRank <chr>, occurrenceStatus <chr>, sex <chr>, lifeStage <chr>,
## #   scientificName <chr>, scientificNameID <chr>, minimumDepthInMeters <int>,
## #   maximumDepthInMeters <int>, organismID <chr>, occurrenceID <chr>,
## #   geodeticDatum <chr>, eventID <chr>, coordinateUncertaintyInMeters <dbl>,
## #   dataGeneralizations <chr>
```

Notice that we have gone from 19 rows to 17 rows. Removing rows observed on
`2009-09-25T11:11:00Z` and `2009-10-27T16:22:00Z` as they were the second
points within that specifc hour.

2.6.2.1.4 Filter on QARTOD flags?   We also have QARTOD flags and
they are as follows:

value	meaning
1	PASS
2	NOT_EVALUATED
3	SUSPECT
4	FAIL

value	meaning
9	MISSING

The QARTOD tests are:

variable	long_name
qartod_time_flag	Time QC test - gross range test
qartod_speed_flag	Speed QC test - gross range test
qartod_location_flag	Location QC test - Location test
qartod_rollup_flag	Aggregate QC value

I'm not sure what to do here. My preference would be to include all rows where
`qartod_rollup_flag == 1` and drop the rest. But I'm open to suggestions.

```
# perform filter but don't save it.
filter(occurrencedf_dec, qartod_rollup_flag == 1)
```

```
## # A tibble: 16 x 24
##     eventDate      decimalLatitude decimalLongitude basisOfRecord location_class
##     <chr>                   <dbl>            <dbl> <chr>          <chr>
##  1 2009-09-23T00:~          34.0           -119. HumanObserva~ nan
##  2 2009-09-27T17:~          34.0           -119. MachineObser~ 1
##  3 2009-10-08T20:~          34.0           -119. MachineObser~ 2
##  4 2009-10-15T11:~          34.0           -119. MachineObser~ 0
##  5 2009-10-17T06:~          34.0           -119. MachineObser~ 0
##  6 2009-10-17T09:~          34.0           -119. MachineObser~ 2
##  7 2009-10-17T10:~          34.0           -119. MachineObser~ 3
##  8 2009-10-18T08:~          34.0           -119. MachineObser~ 1
##  9 2009-10-18T10:~          34.0           -119. MachineObser~ 2
## 10 2009-10-18T11:~          34.0           -119. MachineObser~ 0
## 11 2009-10-23T23:~          34.0           -119. MachineObser~ 2
## 12 2009-10-24T00:~          34.0           -119. MachineObser~ 0
## 13 2009-10-26T10:~          34.0           -119. MachineObser~ 3
## 14 2009-10-27T16:~          34.0           -119. MachineObser~ 1
## 15 2009-10-29T11:~          34.0           -119. MachineObser~ 2
## 16 2009-10-31T21:~          34.0           -119. MachineObser~ 0
## # i 19 more variables: qartod_time_flag <int>, qartod_speed_flag <int>,
## #   qartod_location_flag <int>, qartod_rollup_flag <int>, kingdom <chr>,
## #   taxonRank <chr>, occurrenceStatus <chr>, sex <chr>, lifeStage <chr>,
## #   scientificName <chr>, scientificNameID <chr>, minimumDepthInMeters <int>,
## #   maximumDepthInMeters <int>, organismID <chr>, occurrenceID <chr>,
## #   geodeticDatum <chr>, eventID <chr>, coordinateUncertaintyInMeters <dbl>,
## #   dataGeneralizations <chr>
```

Drop the quality flag columns to align with DarwinCore standard.

```
occurrencedf_dec <- occurrencedf_dec %>%
    select(
        -c(location_class,
            qartod_time_flag,
            qartod_speed_flag,
            qartod_location_flag,
            qartod_rollup_flag
            ))

names(occurrencedf_dec)
```

```
##  [1] "eventDate"                "decimalLatitude"
##  [3] "decimalLongitude"         "basisOfRecord"
##  [5] "kingdom"                  "taxonRank"
##  [7] "occurrenceStatus"         "sex"
##  [9] "lifeStage"                "scientificName"
## [11] "scientificNameID"         "minimumDepthInMeters"
## [13] "maximumDepthInMeters"     "organismID"
## [15] "occurrenceID"             "geodeticDatum"
## [17] "eventID"                  "coordinateUncertaintyInMeters"
## [19] "dataGeneralizations"
```

2.6.2.2   Add NCEI URL to references

For full transparency, we will add the Darwin Core term references to provide
the URL to the NCEI landing page, where the full dataset can be accessed by
curious users.

```
occurrencedf_dec$references <- "https://www.ncei.noaa.gov/archive/accession/0282699"

occurrencedf_dec
```

```
## # A tibble: 17 x 20
##    eventDate      decimalLatitude decimalLongitude basisOfRecord kingdom taxonRank
##    <chr>                    <dbl>            <dbl> <chr>         <chr>   <chr>
##  1 2009-09-23T~              34.0            -119. HumanObserva~ Animal~ Species
##  2 2009-09-25T~              34.0            -119. MachineObser~ Animal~ Species
##  3 2009-09-27T~              34.0            -119. MachineObser~ Animal~ Species
##  4 2009-10-08T~              34.0            -119. MachineObser~ Animal~ Species
##  5 2009-10-15T~              34.0            -119. MachineObser~ Animal~ Species
##  6 2009-10-17T~              34.0            -119. MachineObser~ Animal~ Species
##  7 2009-10-17T~              34.0            -119. MachineObser~ Animal~ Species
```

```
##  8 2009-10-17T~              34.0              -119. MachineObser~ Animal~ Species
##  9 2009-10-18T~              34.0              -119. MachineObser~ Animal~ Species
## 10 2009-10-18T~              34.0              -119. MachineObser~ Animal~ Species
## 11 2009-10-18T~              34.0              -119. MachineObser~ Animal~ Species
## 12 2009-10-23T~              34.0              -119. MachineObser~ Animal~ Species
## 13 2009-10-24T~              34.0              -119. MachineObser~ Animal~ Species
## 14 2009-10-26T~              34.0              -119. MachineObser~ Animal~ Species
## 15 2009-10-27T~              34.0              -119. MachineObser~ Animal~ Species
## 16 2009-10-29T~              34.0              -119. MachineObser~ Animal~ Species
## 17 2009-10-31T~              34.0              -119. MachineObser~ Animal~ Species
## # i 14 more variables: occurrenceStatus <chr>, sex <chr>, lifeStage <chr>,
## #   scientificName <chr>, scientificNameID <chr>, minimumDepthInMeters <int>,
## #   maximumDepthInMeters <int>, organismID <chr>, occurrenceID <chr>,
## #   geodeticDatum <chr>, eventID <chr>, coordinateUncertaintyInMeters <dbl>,
## #   dataGeneralizations <chr>, references <chr>
```

```r
tag_id <- metadata %>% dplyr::filter(variable == "NC_GLOBAL" & name == "ptt_id")

file_name_occur <- paste('data/dwc/atn_',tag_id$value,'_occurrence.csv', sep = "")

write.csv(occurrencedf_dec, file=file_name_occur, row.names=FALSE, fileEncoding="UTF-8", quote=TR
```

2.6.2.2.1 Write decimated occurrence file as csv

2.6.2.3 Measurement or Fact

Since we do have any additional observations, we can create a measurement or fact file to include those data. Might be worthwhile to include tag/device metadata, some of the animal measurements, and the detachment information. Each term should have a definition URI.

The measurementOrFact file will only contain information referencing the `basisOfRecord = HumanObservation` as these observations were made when the animal was directly tagged, in person (ie. when `basisOfRecord == HumanObservation`).

DarwinCore Term	Status	netCDF
organismID		The `platform_id` global attribute plus the `animal_common_name` global attribute.
occurrenceID	Required	`eventDate`, plus data contained in `z` variable, plus `animal_common_name` global attribute.
measurementType	Required	`long_name` attribute of the `animal_weight`, `animal_length`, `animal_length_2` variables.
measurementValue	Required	The data from the `animal_weight`, `animal_length`, `animal_length_2` variables.
eventID	Strongly Recommended	`animal_common_name` global attribute plus the `eventDate`.
measurementUnit	Strongly Recommended	`unit` attribute of the `animal_weight`, `animal_length`, `animal_length_2` variables.
measurementMethod	Strongly Recommended	`animal_weight`, `animal_length`, `animal_length_2` attributes of their respective variables.
measurementTypeID	Strongly Recommended	mapping table somewhere?
measurementMethodID	Strongly Recommended	mapping table somewhere?

DarwinCore Term	Status	netCDF
measurementUnitID	Strongly Recommended	mapping table somewhere?
measurementAccuracy	Share if available	
measurementDeterminedDate	Share if available	
measurementDeterminedBy	Share if available	
measurementRemarks	Share if available	
measurementValueID	Share if available	

2.6.2.3.1 Extracting variables for Extended Measurement Or Fact (eMOF) Here there are two approaches to transforming a variable to the eMOF Darwin Core extension. The goal is to collapse the measurement name, value, unit, related identifiers and remarks into a generalized long format that can be linked to occurrences and events. For more info see:

1. The OBIS manual
2. The Marine Biological Data Mobilization Workshop 2023 (SF:Not sure if it's cool to reference the workshop like this)

The first several lines of the below code show an example of pulling out the variable attributes and individually mapping them to the eMOF terms. However, this can be done more efficiently (although less readable) via this chunk of code:

```
# Supply vector of variable names
c("animal_length",
  "animal_length_2",
  "animal_weight") %>%

    # Create a named list of the variable attributes and convert it into a data frame, for each
    purrr::map_df(function(x) {
      Olist(measurementValue = ncvar_get( nc, x),
            measurementType = ncatt_get( nc, x)$long_name,
            measurementUnit = ncatt_get( nc, x)$units,
            measurementMethod = ncatt_get( nc, x)[[paste0(x,'_type')]])
    })
```

```
# # Measurement or Fact extension
# # Need to find the occurrence where basisOfRecord == HumanObservation, then pull the organism.

emof_data <- #var_names %>%
    #filter(str_starts(name, pattern = "animal_[lw]e")) %>% #example using regex to parse names
    #    pull(name) %>%
```

```r
    # Example using vector of variables
    c("animal_length",
      "animal_length_2",
      "animal_weight") %>%
    purrr::map_df(function(x) {
        list(measurementValue = ncvar_get( nc, x),
             measurementType = ncatt_get( nc, x)$long_name,
             measurementUnit = ncatt_get( nc, x)$units,
             measurementMethod = ncatt_get( nc, x)[[paste0(x,'_type')]]])
    }) %>%

    filter(measurementValue != "NaN")


emofdf <- occurrencedf %>%
    filter(basisOfRecord == 'HumanObservation') %>%
    select(organismID, eventID, occurrenceID) %>%
    cbind(emof_data)

str(emofdf)
```

```
## 'data.frame':    1 obs. of  7 variables:
##  $ organismID      : chr "105838_great_white_shark"
##  $ eventID         : chr "great_white shark_2009-09-23T00:00:00Z"
##  $ occurrenceID    : chr "2009-09-23T00:00:00Z_0_great_white_shark"
##  $ measurementValue : num 213
##  $ measurementType  : chr "length of the animal as measured or estimated at deployme
##  $ measurementUnit  : chr "cm"
##  $ measurementMethod: chr "total length"
```

```r
tag_id <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name ==

file_name_emof <- paste('data/dwc/atn_',tag_id$value,'_emof.csv', sep = "")

write.csv(emofdf, file=file_name_emof, row.names=FALSE, fileEncoding="UTF-8", quote=TRU
```

2.6.2.3.2 Write emof file as csv

2.6.2.4 Metadata creation

Now that we know our data are aligned to Darwin Core, we can start collecting
metadata. Using the R package EML we can create the EML metadata to

associate with the data above.

Some good sources to help identify what requirements we need in the EML
metadata can be found at:

- https://github.com/gbif/ipt/wiki/GMPHowToGuide

- https://github.com/gbif/ipt/wiki/GMPHowToGuide#dataset-resource

```
# library(EML)
```

The first thing we need to do is collect all of the relevant pieces of metadata for
our EML record.

```
# me <- list(individualName = list(givenName = "Matt", surName = "Biddle"))

# my_eml <- list(dataset = list(

#                          title = "A Minimal Valid EML Dataset",

#                          creator = me,

#                          contact = me

#                            )

#                  )
```

```
# geographicDescription <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter

# west <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name == "geospat

# east <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name == "geospat

# north <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name == "geospa

# south <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name == "geospa

# altitudeMin <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name == "g

# altitudeMax <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name == "g

# altitudeUnits <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name ==
```

```r
# coverage <-

#   set_coverage(begin = format(min(atn_tbl$time),'%Y-%m-%d'), end = format(max(atn_tb

#                sci_names = RNetCDF::var.get.nc(RNetCDF::open.nc("atn_trajectory_temp

#                geographicDescription = paste(geographicDescription$value),

#                west = paste(west$value),

#                east = paste(east$value) ,

#                north = paste(north$value) ,

#                south = paste(south$value) ,

#                altitudeMin = paste(altitudeMin$value),

#                altitudeMaximum = paste(altitudeMax$value),

#                altitudeUnits = ifelse (paste(altitudeUnits$value) == 'm', "meter", "


# creator_name <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filte

# creator_email <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filt

# creator_sector <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::fil

# creator <- eml$creator(

#          eml$individualName(

#               givenName = paste(creator_name$value),

#               surName = paste(creator_name$value)

#               ),

#          position = paste(creator_sector$value),

#          electronicMailAddress = paste(creator_email$value)

#          )
```

```r
# #contact_name = metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name ==

# contact <- eml$contact(

#           eml$individualName(

#           givenName = paste(creator_name$value),

#           surName = paste(creator_name$value)),

#           position = paste(creator_sector$value),

#           electronicMailAddress = paste(creator_email$value)

#           )

# #metadata_name

# metadataProvider <- eml$metadataProvider(

#           eml$individualName(

#               givenName = paste(creator_name$value),

#               surName = paste(creator_name$value)),

#           position = paste(creator_sector$value),

#           electronicMailAddress = paste(creator_email$value)

#           )

# ## these are the entries in contributor, need to iterate since comma separated list.

# contrib_name <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name ==

# contrib_position <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name

# contrib_email <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name ==

# associatedParty <- eml$associatedParty(

#                   eml$individualName(

#                   givenName = paste(contrib_name$value),
```

```
#                             surName = paste(contrib_name$value)),

#                             position = paste(contrib_position$value),

#                             electronicMailAddress = paste(contrib_email$value)

#                             )

# abstract <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(na

# # keywords

# keywords <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(na

# kw_vocab <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(na

# keywordSet <- list(

#     list(

#         keywordThesaurus = kw_vocab$value$keywords_vocabulary,

#         keyword = as.list(strsplit(keywords$value$keywords, ", "))

#         ))

# title <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(name

# methods <- "NEED TO MAP FROM NCFILE"

# license <- metadata %>% dplyr::filter(variable == "NC_GLOBAL") %>% dplyr::filter(nam
```

Now build the eml file.

```
# library(uuid)

# physical <- set_physical(file_name_occur)

# # attributeList <-
# #   set_attributes(attributes,
```

```
# #                    factors,

# #                    col_classes = c("character",

# #                                    "Date",

# #                                    "Date",

# #                                    "Date",

# #                                    "factor",

# #                                    "factor",

# #                                    "factor",

# #                                    "numeric"))


# my_eml <- eml$eml(

#           packageId = paste(uuid_tbl$value),

#           system = "uuid",

#           dataset = eml$dataset(

#                alternateIdentifier = UUIDgenerate(use.time = TRUE),

#                title = title$value,

#                creator = creator,

#                metadataProvider = metadataProvider,

#                #associatedParty = associatedParty,

#                contact = contact,

#                pubDate = format(Sys.time(),'%Y-%m-%d'),

#                language = "English",

#                intellectualRights = eml$intellectualRights(
```

```
#                                          para = "To the extent possible under law, the pu

#                                          #para = paste(license$value),

#                                                                    ),

#                       abstract = eml$abstract(

#                                  para = abstract$value$summary,

#                                               ),

#                       keywordSet = keywordSet,

#                       coverage = coverage,

# #                        license = eml$license(

# #                                  licenseName = "CC0 1.0",

# #                                  #licenseName = paste(license$value),

# #                                               ),

#                       #dataTable = eml$dataTable(

#                       #   entityName = file_name_occur,

#                       #   entityDescription = "Occurrences",

#                       #   physical = physical)

#                       ))
```

Validate EML

```
# val <- eml_validate(my_eml)

# attr(val,"errors")
```

Write eml to file.

```
# file_name_eml <- 'eml.xml'

# write_eml(my_eml, file_name_eml)
```

Raw EML

```
# my_eml
```

2.6.2.4.1 Create meta.xml Below is an example of the contents of meta.xml:

```
<archive xmlns="http://rs.tdwg.org/dwc/text/" metadata="eml.xml">

  <core encoding="UTF-8" fieldsTerminatedBy="\t" linesTerminatedBy="\n" fieldsEnclosedBy="" ignor

    <files>

      <location>occurrence.txt</location>

    </files>

    <id index="0" />

    <field index="1" term="http://rs.tdwg.org/dwc/terms/datasetID"/>

    <field index="2" term="http://rs.tdwg.org/dwc/terms/institutionCode"/>

    <field index="3" term="http://rs.tdwg.org/dwc/terms/collectionCode"/>

    <field index="4" term="http://rs.tdwg.org/dwc/terms/basisOfRecord"/>

    <field index="5" term="http://rs.tdwg.org/dwc/terms/occurrenceID"/>

    <field index="6" term="http://rs.tdwg.org/dwc/terms/catalogNumber"/>

    <field index="7" term="http://rs.tdwg.org/dwc/terms/occurrenceRemarks"/>

    <field index="8" term="http://rs.tdwg.org/dwc/terms/individualCount"/>

    <field index="9" term="http://rs.tdwg.org/dwc/terms/sex"/>

    <field index="10" term="http://rs.tdwg.org/dwc/terms/occurrenceStatus"/>

    <field index="11" term="http://rs.tdwg.org/dwc/terms/eventDate"/>

    <field index="12" term="http://rs.tdwg.org/dwc/terms/year"/>

    <field index="13" term="http://rs.tdwg.org/dwc/terms/decimalLatitude"/>
```

```xml
    <field index="14" term="http://rs.tdwg.org/dwc/terms/decimalLongitude"/>

    <field index="15" term="http://rs.tdwg.org/dwc/terms/coordinateUncertaintyInMeters"

    <field index="16" term="http://rs.tdwg.org/dwc/terms/scientificNameID"/>

    <field index="17" term="http://rs.tdwg.org/dwc/terms/scientificName"/>

  </core>

  <extension encoding="UTF-8" fieldsTerminatedBy="\t" linesTerminatedBy="\n" fieldsEncl

    <files>

      <location>extendedmeasurementorfact.txt</location>

    </files>

    <coreid index="0" />

    <field index="1" term="http://rs.tdwg.org/dwc/terms/occurrenceID"/>

    <field index="2" term="http://rs.tdwg.org/dwc/terms/measurementType"/>

    <field index="3" term="http://rs.tdwg.org/dwc/terms/measurementValue"/>

    <field index="4" term="http://rs.tdwg.org/dwc/terms/measurementUnit"/>

    <field index="5" term="http://rs.iobis.org/obis/terms/measurementUnitID"/>

    <field index="6" term="http://rs.tdwg.org/dwc/terms/measurementDeterminedDate"/>

  </extension>

</archive>
```

Checkout XML package for R.

`conda install -c conda-forge r-xml`

Another example in this github repository.

Or use the gui here to create meta.xml.

```r
# library(XML)
```

```r
# doc = newXMLDoc()

# archiveNode = newXMLNode("archive", attrs = c(metadata=file_name_eml), namespaceDefinitions=c("

# ## For the core occurrence

# coreNode = newXMLNode("core", attrs = c(encoding="UTF-8", linesTerminatedBy="\\r\\n", fieldsTe

# filesNode = newXMLNode("files", parent = coreNode)

# locationNode = newXMLNode("location", file_name_occur, parent = filesNode)

# idnode = newXMLNode("id", attrs = c(index="9"), parent = coreNode)

# # iterate over the columns in occurrence file to create field elements

# i=0

# for (col in colnames(occurrencedf))

#     {

#     termstr = paste("http://rs.tdwg.org/dwc/terms/",col, sep="")

#     i=i+1

#     fieldnode = newXMLNode("field", attrs = c(index=i, term=termstr), parent=coreNode)

# }

# ## for the extensions

# extensionNode = newXMLNode("extension", attrs = c(encoding="UTF-8", linesTerminatedBy="\\r\\n",

# filesNode = newXMLNode("files", parent = extensionNode)

# locationNode = newXMLNode("location", file_name_event, parent = filesNode)

# idnode = newXMLNode("id", attrs = c(index="0"), parent = extensionNode)
```

```r
# # iterate over the columns in occurrence file to create field elements

# i=0

# for (col in colnames(eventdf))

#     {

#     if (col == 'modified'){

#         termstr = paste("http://purl.org/dc/terms/", col, sep="")

#     } else {

#         termstr = paste("http://rs.tdwg.org/dwc/terms/",col, sep="")

#         }


#     i=i+1


#     fieldnode = newXMLNode("field", attrs = c(index=i, term=termstr), parent=extensi

# }



# print(doc)



# saveXML(doc, file="meta.xml")
```

2.6.2.5   Build the DarwinCore-Archive zip package

```r
# library(zip)


# files = c(file_name_occur, file_name_event, file_name_eml, "meta.xml")

# zip::zip(
```

```
#     "atn.zip",

#     files,

#     root = ".",

#     mode = "mirror",

# )


# zip_list("atn.zip")
```

2.6.3 sessionInfo()

```
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] mapdata_2.3.1   maps_3.4.1      ncdf4_1.21      obistools_0.1.0
##  [5] tidync_0.3.0    worrms_0.4.3    magrittr_2.0.3  knitr_1.42
##  [9] here_1.0.1      lubridate_1.9.3 forcats_1.0.0   stringr_1.5.0
## [13] dplyr_1.1.2     purrr_1.0.1     readr_2.1.5     tidyr_1.3.0
## [17] tibble_3.2.1    ggplot2_3.4.2   tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.10       rprojroot_2.0.4  digest_0.6.31    utf8_1.2.3
##  [5] R6_2.5.1          cellranger_1.1.0 evaluate_0.21    pillar_1.9.0
##  [9] rlang_1.1.1       curl_5.0.0       readxl_1.4.3     ncmeta_0.3.5
## [13] rstudioapi_0.16.0 rmarkdown_2.21   urltools_1.7.3   htmlwidgets_1.6.2
```

```
## [17] bit_4.0.5         triebeard_0.4.1   munsell_0.5.0     compiler_4.1.1
## [21] xfun_0.39         pkgconfig_2.0.3   htmltools_0.5.5   tidyselect_1.2.1
## [25] httpcode_0.3.0    bookdown_0.40     fansi_1.0.4       crayon_1.5.2
## [29] tzdb_0.3.0        withr_2.5.0       rappdirs_0.3.3    crul_1.5.0
## [33] grid_4.1.1        jsonlite_1.8.4    gtable_0.3.3      lifecycle_1.0.3
## [37] scales_1.2.1      cli_3.6.1         stringi_1.7.12    vroom_1.6.3
## [41] leaflet_2.1.2     xml2_1.3.6        generics_0.1.3    vctrs_0.6.5
## [45] data.tree_1.1.0   tools_4.1.1       bit64_4.0.5       glue_1.6.2
## [49] RNetCDF_2.6-2     crosstalk_1.2.0   hms_1.1.3         parallel_4.1.1
## [53] fastmap_1.1.1     yaml_2.3.7        timechange_0.3.0  colorspace_2.1-0
```

Appendix A

Frequently Asked Questions

1. **Q.** What data structure does OBIS recommend?

   A. The OBIS-ENV Darwin Core Archive Data Structure. OBIS manual

2. **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.

3. **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

4. **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

5. **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

6. **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.

7. **Q.** What metadata profile does OBIS use?

   A. OBIS uses the GBIF EML profile (version 1.1)

8. **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

Appendix B

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.

B.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).

B.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.
pyobis	BIO	Pyobis is an interesting python package that helps users fetch data from OBIS API which harvests occurrence records from thousands of datasets and makes them available as a single integrated dataset.

Package	Type	Description
pyworms	BIO	Python client for the World Register of Marine Species (WoRMS) REST service.
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.

B.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.

B.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.

Appendix C

Extras

Below is some more in-depth information into specific areas associated with standardizing your data to Darwin Core and uploading it through the IPT.

C.1 Ecological Metadata Language (EML)

The Ecological Metadata Language (EML) is a community developed and maintained metadata standard that is typically associated with ecological-, and earth and environmental data. The purpose of EML is to provide the ecological community with an extensible, flexible, metadata standard used in data analysis and archiving, which will allow automated machine processing, searching and retrieval. EML has been around since 2003, and can be considered a "dialect" or specification to XML to describe tables and other data objects. The XML Schema provides a framework for the metadata, with defined "rules" on how to organize the metadata without any stipulations (another way to put it, *XML is the language that defines the rules that govern the EML syntax*). The XML Schema defines the structure of some information in a document (e.g. elements and attributes names and relationships), but does not provide any specific details the information included within. An EML document or file (eml.xml) is used to provide detailed description of metadata related to data objects, including tables (and other data objects), their columns, typing etc, and how data tables are linked or grouped. EML is widely used for datasets about ecosystem level observations, and can be used to detail data table information to a high granularity, which allows data users to arrange data tables in any way they need to. EML is particularly useful for wide data tables as table level details are entirely contained within the metadata document, meaning that there is not necessarily a need for external definitions such as a code list. However, if you have a long table arrangement, like the Darwin Core Archive (DwC-A), you can define the allowable values for the column in the metadata as well.

EML is excellent at describing the details of a column of data so that the data values in the tables can be read into analysis systems or an analysis environment using the metadata, or even into a relational database. EML allows for tables to be easily reusable, and read into workflows, translated or reformatted. A drawback to EML is that, compared to the ISO standard, EML is a community standard, adopting a more 'bottom-up' approach. This is contrary to the ISO standard, which are internationally agreed upon standards by experts ('top-down'). However, at the time of EML development in the early 2000's there was a gap in metadata options to describe ecological data tables, with ISO standards typically being more applicable to geographic data. EML can cover almost anything and is particularly good at tabular data. But at the same time, due to the self-contained nature, there can be little control from outside lists, which means that the description is left to the EML constructor (data provider/manager) and consequently, individual datasets can look quite different from each other, even when they contain similar measurements. As such, it will be important to document best practices and clear mapping of fields between different metadata schemas (e.g. cross-walks between ISO.xml and EML.xml). As of version 2.2, EML can link to external ontologies, and there is capacity for annotation with external terms (e.g. through their URIs). Code lists and external dictionaries can help as they sometimes contain additional information that might not fit into EML (e.g. protocols, or code lists stored in ontologies). Having these external code lists and exporting them as EML snippets could go a long way in reducing that heterogeneity, because the constructors can then select measurements from lists when developing EML documents.

EML is implemented as a series of XML document types (modules) that can be used in an extensible manner to document ecological data. Each EML module is designed to describe one logical part of the total metadata that should be included with any ecological dataset. The architecture of EML was designed to serve the needs of the ecological community, and has benefitted from previous work in other related metadata standards. Using this format can facilitate future growth of the metadata language, and EML supports an active developer community (see e.g. NCEAS EML GitHub). EML adopts much of its syntax from the other metadata standards that have evolved from the expertise of groups in other disciplines. Whenever possible, EML adopted entire trees of information in order to facilitate conversion of EML documents into other metadata languages. The GBIF IPT is a tool used to create a single eml.xml file format inside the DwC-A data package. However, the IPT does not use any of the EML's built-in table description modules, and perhaps primarily uses one EML module (resource module) for high-level metadata.

However, it is important to know how both OBIS and GBIF use EML, as often a higher granularity of the metadata can be found in the original data tables. An example of this is spatial coverage. The IPT only allows for either a bounding box to be documented (populating North, South, East, and West coordinates), or a single polygon. The EML document however would be able to capture multiple polygons worth of spatial coverage (i.e. a polygon for each transect

surveyed). This more detailed information however is often captured in the data (in an OBIS record). Additionally, not all fields that can be populated in an EML document can be translated to the IPT, or harvested by OBIS and GBIF. The GBIF IPT only produces a select number of fields or attributes available in EML.

Important: When reading the EML section in the OBIS manual, you'll notice that it reads that OBIS uses the GBIF EML profile (version 1.1). However, the current EML version is 2.2.0, as per EcoInformatics. This does not mean that these versions are not compatible, rather, it means that the GBIF IPT currently uses a subset of available EML 2.2.0 fields and attributes, the subset of which they have versioned 1.1.

If you are interested in creating an EML metadata file, it is possible to upload those into the IPT. There are R packages that can help in developing an EML.xml file. These packages are e.g. EML, emld or EMLassemblyline.

C.2   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:

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.
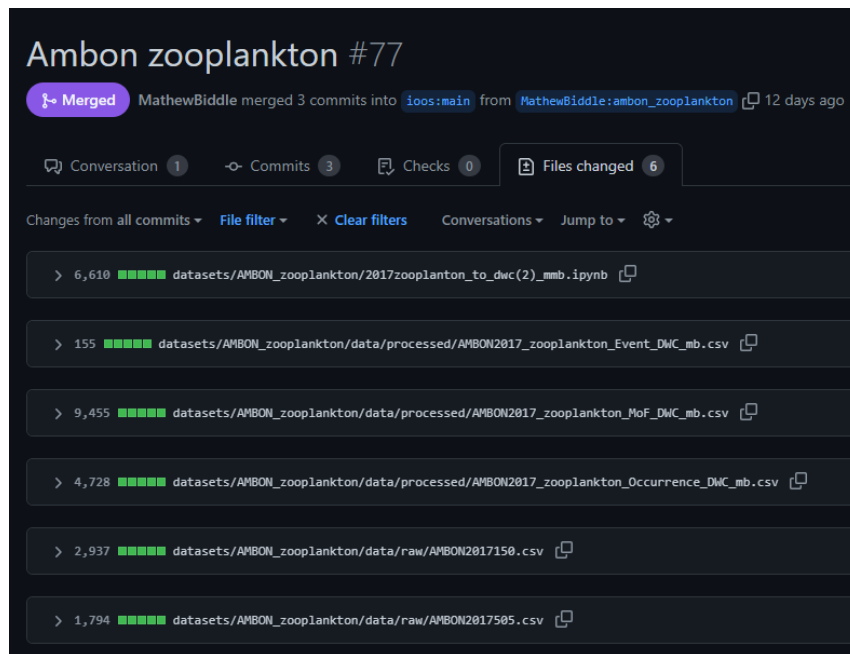
5. The OBIS node manager was notified of the availability of a revised dataset by pointing directly to the appropriate commit in GitHub:
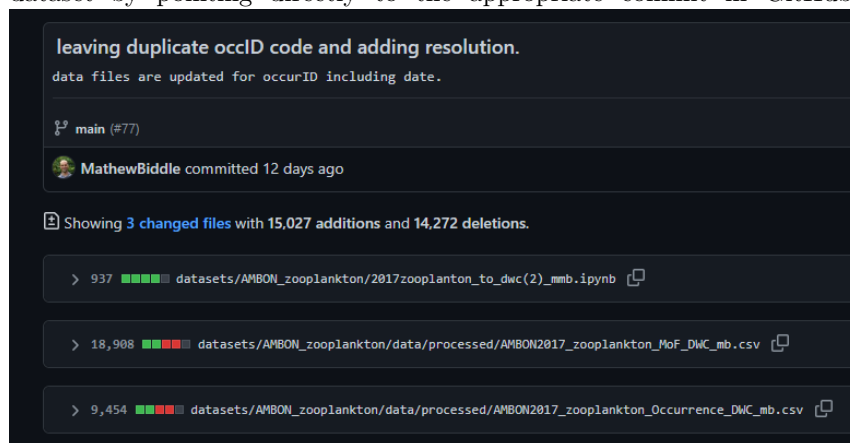
6. The OBIS node manager downloaded the data from the commit above and uploaded them to the IPT.

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.

1. Here is fixing the `scientificNameID` generation:

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

7   df.head()
```

2. Here is removing the problematic code:

```
1   #finally!
2   #nope it's scientificNameID that needs this
3
4   # mmb - Think this was not necessary.
5   # sciids = df[['scientificNameID']].astype('string', errors='ignore')
6   # s = sciids['scientificNameID'].convert_dtypes()
7   # s=s.str.strip('.0')
8   # df['scientificNameID']=s
9   # df.head()
```

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

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
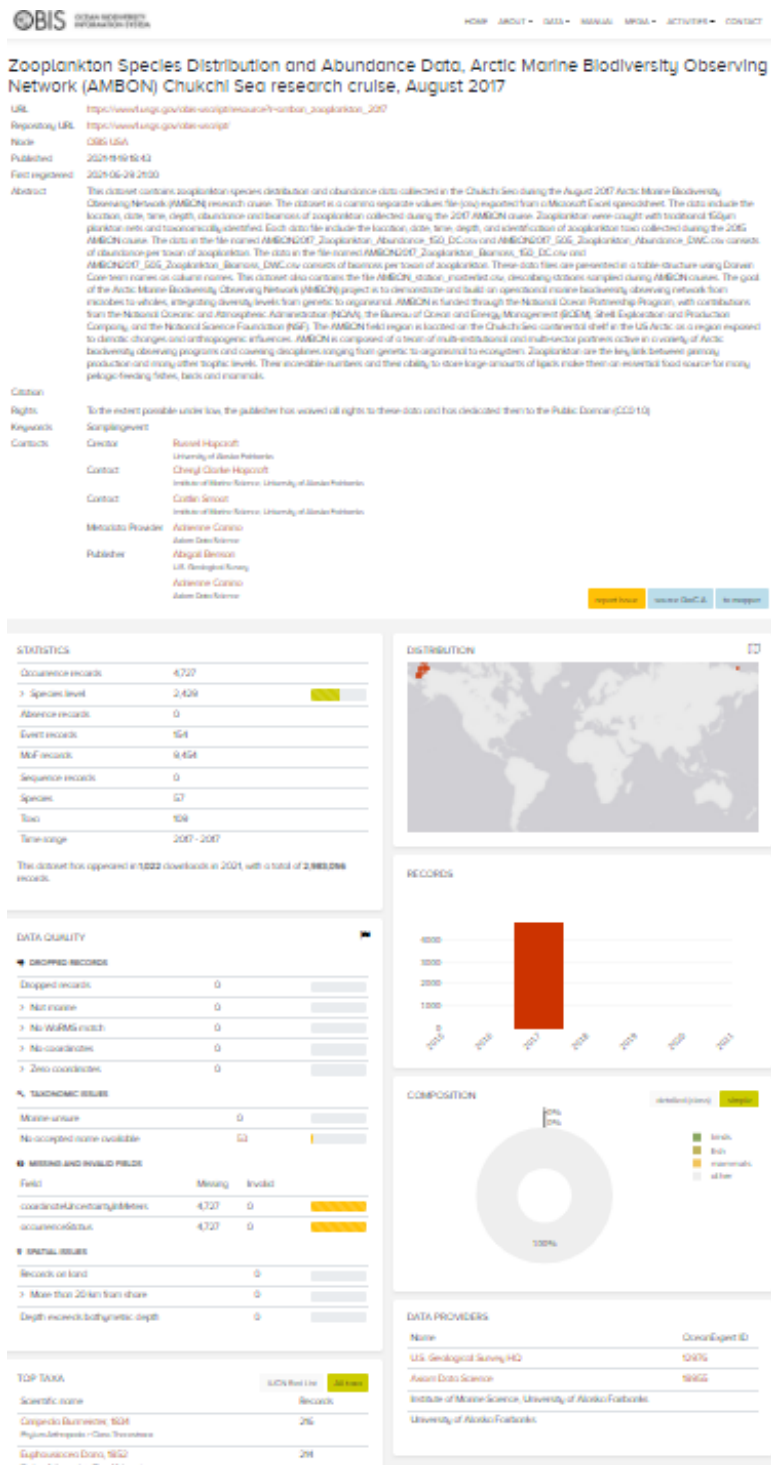
these data are present:

References

Allaire, JJ, Yihui Xie, Christophe Dervieux, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, et al. 2023. *Rmarkdown: Dynamic Documents for r.* https://CRAN.R-project.org/package=rmarkdown.

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.

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. "Envi-
    ronmental 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. Stachow-
    icz. 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 Develop-
    ment Processes for US IOOS: Visual Line Transect Observing Community
    for Mammal, Bird, and Turtle Data." IOOS. https://www.researchgate.
    net/publication/255681522.

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 Biodiver-
    sity 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 Wein-
    gart C, Dmitry Schigel, and Douglas E Soltis. 2021. "Data Integra-
    tion Enables Global Biodiversity Synthesis." *Proceedings of the Na-
    tional Academy of Sciences of the United States of America.* https:
    //doi.org/10.1073/pnas.2018093118/-/DCSupplemental.

Jonathan A. Hare, Mark W. Nelson, Wendy E. Morrison. n.d. "A Vul-
    nerability 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 Bow-
    ers. 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.

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.

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.

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.

———. 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.

———. 2023. *Knitr: A General-Purpose Package for Dynamic Report Generation in r.* https://yihui.org/knitr/.

———. 2024. *Bookdown: Authoring Books and Technical Documents with r Markdown.* https://CRAN.R-project.org/package=bookdown.

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.