Standardizing-Marine-Biological-Data

Brett Johnson

2021-09-17

Contents

Chapter 1

Preface

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

Chapter 2

Introduction

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

2.1 Data Structures

The OBIS-ENV Darwin Core Archive Data Structure.

OBIS manual

2.2 Ontologies & Controlled Vocabularies

An ontology is a classification system for establishing a hierarchically related set of concepts. Concepts are often terms from controlled vocabularies.

From Marine Metadata: # TODO: add link

"Ontologies can include all of the following, but are not required to include them, depending on which perspective from above you adhere to:

Classes (general things, types of things) Instances (individual things) Relationships among things Properties of things Functions, processes, constraints, and rules relating to things"

TODO: Research Unified Modelling Language?

There are a number of controlled vocabularies that are used to describe parameters commonly used in specific research domains. This allows for greater interoperability of data sets within the domain, and ideally between domains. Here, we strive to document a number of relevant examples.

- Climate and Format (CF) Standard Names are applied to sensors for application with OPeNDAP web service.
- Device categories using the SeaDataNet device categories in NERC 2.0
- Device make/model using the SeaVoX Device Catalogue in NERC 2.0,
- Platform categories using SeaVoX Platform Categories in NERC 2.0
- Platform instances using the ICES Platform Codes in NERC 2.0
- Unit of measure
- GCMD Keywords (NASA)
- Geographic Domain/Features of Interest
- GeoLink base ontology was part of the EarthCube GeoLink Project

Note: To describe a measurement or fact of a biological specimen that conforms to Darwin Core standards, it's necessary to use the 'Biological entity described elsewhere' method rather than taxon specific.

2.2.1 Resources

2.2.2 Oceanography

Biological and Chemcial Oceanography Data Management Office

Marine metadata interoperability vocab resources

2.2.3 Biology

BioPortal Ecosystem Ontology

2.2.4 NERC Search Interfaces

- SeaDatanet Common Vocab Search Interface:
- SeaDataNet Common Vocabularies:
- SeaDataNet Vocab Library
- Measurement Types in OBIS

2.2.5 Geosciences

UDUNITSare more common unit measurements in geosciences

2.2.6 Eco/EnvO

Environment Ontology including genomics.

2.2.7 Wild Cards

Question: Not sure use case for this.

P01 Biological Entity Parameter Code Builder

2.3 Technologies

2.3.1 ERDDAP

ERDDAP can be thought of as 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. Here is a basic erddap installation that walks you through how to load a data set.

2.4 Notes on Integrating OBIS, Darwin Core as it relates to OOS's

2.5 Metadata

OBIS uses the GBIF EML profile (version 1.1). In case data providers use ISO19115/ISO19139, there is a mapping available here: http://rs.gbif.org/schema/eml-gbif-profile/1.1/eml2iso19139.xsl This will be important for integrating OBIS datasets to other CIOOS and IOOS metadata profiles.

2.6 Data QC

There are a number of tools available to check the quality of data or check your data format against the expected standard.

OBIS Datatools shows some great R packages for this.

2.6.1 Compliance Checking

LifeWatch Belgium provides a number of tools to check your data against. Specifically you can test OBIS data format and see a map of your sample locations to check if they are on land. See http://www.lifewatch.be/data-services/

2.6.2 Semantic Web and Darwin Core

Lessons learned from adapting the Darwin Core vocabulary standard for use in RDF

2.6.3 Resource Description Framework

Darwin Core Resource Description Framework Guide

Chapter 3

Applications

Some *significant* applications are demonstrated in this chapter.

3.1 Salmon Ocean Ecology Data

3.1.1 Intro

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

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

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

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

3.1.2 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.

```
#TODO: Include abiotic measurements (YSI temp and salinity from 0 and 1 m) to hang off

event <- tibble(datasetName = "Hakai Institute Juvenile Salmon Program",
                eventID = survey_seines$seine_id,
                eventDate = date(survey_seines$survey_date),
                eventTime = paste0(survey_seines$set_time, "-0700"),
                eventRemarks = paste3(survey_seines$survey_comments, survey_seines$sei
                decimalLatitude = survey_seines$lat,
                decimalLongitude = survey_seines$long,
                locationID = survey_seines$site_id,
                coordinatePrecision = 0.00001,
                coordinateUncertaintyInMeters = 10,
                country = "Canada",
                countryCode = "CA",
                stateProvince = "British Columbia",
                habitat = "Nearshore marine",
                geodeticDatum = "EPSG:4326 WGS84",
                minimumDepthInMeters = 0,
                maximumDepthInMeters = 9, # seine depth is 9 m
                samplingProtocol = "http://dx.doi.org/10.21966/1.566666", # This is th
                language = "en",
                license = "http://creativecommons.org/licenses/by/4.0/legalcode",
                bibliographicCitation = "Johnson, B.T., J.C.L. Gan, S.C. Godwin, M. Kr
                references = "https://github.com/HakaiInstitute/jsp-data",
                institutionID = "https://www.gbif.org/publisher/55897143-3f69-42f1-810
                institutionCode = "Hakai"
               )
```

3.1.3 occurrence

Next you'll want to determine what constitutes an occurrence for your data set. Because each event captures 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 occurrence 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.

Not every fish is actually collected and given a Universal Fish Number (UFN) in our fish data tables, so in our field data sheets we record the total number of fish captured and the total number retained. So to get an occurrence row for every fish captured I create a row for every fish caught (minus the number taken) and create a generic numeric id (ie hakai-jsp-1) in one table and then join that to the fish table that includes a row for every fish retained that already has a UFN.

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

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

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

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

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

all_fish_not_retained <-
  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(seine_id, scientificName, occurrenceID)

occurrence <- bind_rows(all_fish_not_retained, fish_retained_data) %>%
  rename(eventID = seine_id) %>%  # rename = dplyr::rename; vs plyr::rename
  mutate(`Life stage` = "juvenile")

unique_taxa <- unique(occurrence$scientificName)
worms_names <- wm_records_names(unique_taxa) # library(worrms)
df_worms_names <- bind_rows(worms_names) %>%
  select(scientificName = scientificname,
         scientificNameAuthorship = authority,
         taxonRank = rank,
         scientificNameID = lsid
         )

#include bycatch species

unique_bycatch <- unique(bycatch$scientificName) %>%  glimpse()
```

```
##  chr [1:29] "Oncorhynchus nerka" "Oncorhynchus tshawytscha" ...
```

```
by_worms_names <- wm_records_names(unique_bycatch) %>%
  bind_rows() %>%
  select(scientificName = scientificname,
         scientificNameAuthorship = authority,
         taxonRank = rank,
         scientificNameID = lsid
         )

bycatch_occurrence <- bycatch %>%
  select(eventID = seine_id, occurrenceID, scientificName, `Life stage` = bm_ageclass)
  filter(scientificName != "unknown")

bycatch_occurrence$`Life stage`[bycatch_occurrence$`Life stage` == "J"] <- "juvenile"
bycatch_occurrence$`Life stage`[bycatch_occurrence$`Life stage` == "A"] <- "adult"
bycatch_occurrence$`Life stage`[bycatch_occurrence$`Life stage` == "Y"] <- "Young of year"

combined_worms_names <- bind_rows(by_worms_names, df_worms_names) %>%
  distinct(scientificName, .keep_all = TRUE)

occurrence <- bind_rows(bycatch_occurrence, occurrence)

occurrence <- left_join(occurrence, combined_worms_names) %>%
    mutate(basisOfRecord = "HumanObservation",
        occurrenceStatus = "present")

write_csv(occurrence,"../datasets/hakai_salmon_data/raw_data/occurrence.csv") # here::here("..",

# This removes events that didn't result in any occurrences
event <- dplyr::semi_join(event, occurrence, by = 'eventID') %>%
  mutate(coordinateUncertaintyInMeters = ifelse(is.na(decimalLatitude), 1852, coordinateUncertain

simple_sites <- sites %>%
  select(site_id, ocgy_std_lat, ocgy_std_lon)

event <- dplyr::left_join(event, simple_sites, by = c("locationID" = "site_id")) %>%
  mutate(decimalLatitude = coalesce(decimalLatitude, ocgy_std_lat),
         decimalLongitude = coalesce(decimalLongitude, ocgy_std_lon)) %>%
  select(-c(ocgy_std_lat, ocgy_std_lon))

write_csv(event,"../datasets/hakai_salmon_data/raw_data/event.csv") # here::here("..", "datasets'
```

3.1.4 measurementOrFact

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

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

```
mof_types <- read_csv("https://raw.githubusercontent.com/HakaiInstitute/jsp-data/maste

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_data$`Life stage` <- "juvenile"



measurementOrFact <- fish_data %>%
  mutate(occurrenceID = paste0("hakai-jsp-", ufn)) %>%
  select(occurrenceID, eventID = seine_id, "Length (fork length)" = fork_length,
         "Standard length" = standard_length, "Weight" = weight, `Life stage`) %>%
  pivot_longer(`Length (fork length)`:`Life stage`,
               names_to = "measurementType",
               values_to = "measurementValue",
               values_transform = list(measurementValue = as.character)) %>%
  filter(measurementValue != "NA") %>%
  left_join(mof_types,by = c("measurementType")) %>%
  mutate(measurementValueID = case_when(measurementValue == "juvenile" ~ "http://vocab
         measurementID = paste(eventID, measurementType, occurrenceID, sep = "-"))


write_csv(measurementOrFact,"../datasets/hakai_salmon_data/raw_data/extendedMeasuremen
```

```
#Check that every eventID in Occurrence occurs in event table
no_keys <- dm(event, occurrence, measurementOrFact)
only_pk <- no_keys %>%
  dm_add_pk(event, eventID) %>%
  dm_add_pk(occurrence, occurrenceID) %>%
  dm_add_pk(measurementOrFact, measurementID)
dm_examine_constraints(only_pk)

model <- only_pk %>%
  dm_add_fk(occurrence, eventID, event) %>%
  dm_add_fk(measurementOrFact, occurrenceID, occurrence)
dm_examine_constraints(model)

#TODO: Fix bookdown issues so that dm_draw shows data model html output. Perhaps add
# dm_draw(model, view_type = "all")
```

3.2 Example Two

Add another example here, perhaps zooplankton?

Chapter 4

Final Words

We have finished a nice book.