

Standardizing-Marine-Biological-Data

Brett Johnson

2020-08-20

Contents

1	Introduction	5
1.1	Data Structures	5
1.2	Ontologies & Controlled Vocabularies	5
1.3	Technologies	7
1.4	Notes on Integrating OBIS, Darwin Core as it relates to OOS's .	7
1.5	Metadata	7
1.6	Data QC	7
2	Methods	9
3	Applications	11
3.1	Salmon Ocean Ecology Data	11
3.2	Example two	17
4	Final Words	19

Biological data structures, definitions, measurements, and linkages are necessarily 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 1

Introduction

The world of standardizing marine biological data is complex and fraught with uncertainty for naive oceanographer, biologist, scientist, or programmer. This is about stacking 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 links necessary to make this possible and will permit broader access to better ecosystem based models. This phenomena is not unique to specific scientific domains, but is rather pervasive as many scientific domains are currently being reshaped in light of recent advances in computing power, technology, and data science.

1.1 Data Structures

The OBIS-ENV Darwin Core Archive Data Structure.

OBIS manual

1.2 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 Modeling 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

TODO: Improve this paragraph There are numerous ways to investigate which controlled vocabulary to use and this can be fairly overwhelming. For a simplified overview see here.

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.

1.2.1 Collections

1.2.2 Oceanography

Biological and Chemical Oceanography Data Management Office

Marine metadata interoperability vocab resources

1.2.3 Biology

BioPortal Ecosystem Ontology

1.2.4 NERC Search Interfaces

- SeaDataNet Common Vocab Search Interface:
- SeaDataNet Common Vocabularies:
- SeaDataNet Vocab Library

1.2.5 Geosciences

UDUNITs are more common unit measurements in geosciences

1.2.6 Eco/EnvO

Environment Ontology including genomics.

1.2.7 Wild Cards

Question: Not sure use case for this.

P01 Biological Entity Parameter Code Builder

1.3 Technologies

1.3.1 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 arrays (‘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.

1.4 Notes on Integrating OBIS, Darwin Core as it relates to OOS’s

1.5 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 OOS metadata profiles.

1.6 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 Data tools shows some great R packages for this.

1.6.1 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/>

1.6.2 Semantic Web and Darwin Core

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

1.6.3 Resource Description Framework

Darwin Core Resource Description Framework Guide

Chapter 2

Methods

We describe our methods in this chapter.

Chapter 3

Applications

Some *significant* applications are demonstrated in this chapter.

3.1 Salmon Ocean Ecology Data

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 Arbor, 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.

3.1.1 event

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

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

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

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

3.1.2 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.

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

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

#

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

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

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

```

For each occurrence 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 website (http://www.marinespecies.org/) to download the
species_matched <- readxl::read_excel(here::here("datasets", "hakai_salmon_data", "raw.
occurrence <- left_join(occurrence, species_matched, by = c("scientificName" = "Scienti
  select(occurrenceID, basisOfRecord, scientificName, eventID, occurrenceStatus = occur
write_csv(occurrence, here::here("datasets", "hakai_salmon_data", "occurrence.csv"))
```

3.1.3 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 `measurementTypes` listed on the NERC database that should be used where possible. I found it difficult to navigate this page to find the correct `measurementType`.

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

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

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

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

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

rm(fish_length, fish_weight)
```

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

rm()
```

3.1.4 Combine into DwC-A

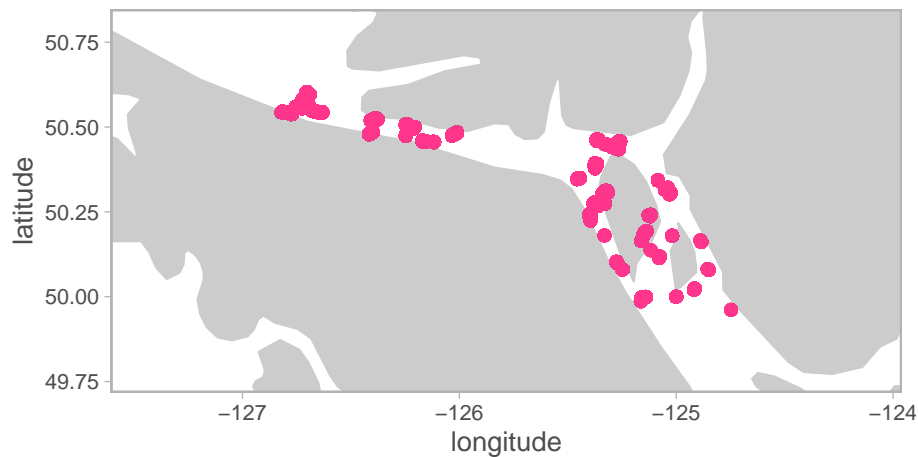
```
DwCA <- left_join(occurrence, event) %>%
  mutate(scientificNameID = 126140) %>%
  drop_na(eventDate) # loses 1388 rows. TODO: Ensure dropped data is as expected
```

3.1.5 Data QC

```
library(obistools)
# Unit tests
check_fields(DwCA)
```

```
## # A tibble: 0 x 0
```

```
plot_map(DwCA, zoom = TRUE)
```



```
ggsave(here("figs", "basic_map.png"))
```

```
check_onland(DwCA)
```

```
## # A tibble: 8,050 x 20
```

```
##      occurrenceID basisOfRecord scientificName eventID occurrenceStatus Kingdom
##      <chr>          <chr>          <chr>          <chr>    <chr>          <chr>
##  1 hakai-jsp-8~ HumanObserva~ Oncorhynchus ~ DE104N1 present      Animal~
##  2 hakai-jsp-8~ HumanObserva~ Oncorhynchus ~ DE104N1 present      Animal~
##  3 hakai-jsp-8~ HumanObserva~ Oncorhynchus ~ DE104N1 present      Animal~
##  4 hakai-jsp-8~ HumanObserva~ Oncorhynchus ~ DE104N1 present      Animal~
##  5 hakai-jsp-8~ HumanObserva~ Oncorhynchus ~ DE104N1 present      Animal~
##  6 hakai-jsp-8~ HumanObserva~ Oncorhynchus ~ DE104N1 present      Animal~
##  7 hakai-jsp-8~ HumanObserva~ Oncorhynchus ~ DE104N1 present      Animal~
##  8 hakai-jsp-8~ HumanObserva~ Oncorhynchus ~ DE104N1 present      Animal~
##  9 hakai-jsp-8~ HumanObserva~ Oncorhynchus ~ DE104N1 present      Animal~
## 10 hakai-jsp-8~ HumanObserva~ Oncorhynchus ~ DE104N1 present      Animal~
## # ... with 8,040 more rows, and 14 more variables: Phylum <chr>, Class <chr>,
## #   Order <chr>, Family <chr>, Genus <chr>, Species <chr>, eventDate <date>,
## #   decimalLatitude <dbl>, decimalLongitude <dbl>, geodeticDatum <chr>,
## #   minimumDepthInMeters <dbl>, maximumDepthInMeters <dbl>,
## #   samplingProtocol <chr>, scientificNameID <dbl>
```

```
check_depth(DwCA)
```

```
## # A tibble: 262,455 x 20
##      occurrenceID basisOfRecord scientificName eventID occurrenceStatus Kingdom
##      <chr>          <chr>          <chr>          <chr>    <chr>          <chr>
##  1 hakai-jsp-1 HumanObserva~ Oncorhynchus ~ DE101N1 present      Animal~
##  2 hakai-jsp-2 HumanObserva~ Oncorhynchus ~ DE101N1 present      Animal~
##  3 hakai-jsp-3 HumanObserva~ Oncorhynchus ~ DE101N1 present      Animal~
##  4 hakai-jsp-4 HumanObserva~ Oncorhynchus ~ DE101N1 present      Animal~
##  5 hakai-jsp-5 HumanObserva~ Oncorhynchus ~ DE101N1 present      Animal~
##  6 hakai-jsp-6 HumanObserva~ Oncorhynchus ~ DE101N1 present      Animal~
##  7 hakai-jsp-7 HumanObserva~ Oncorhynchus ~ DE101N1 present      Animal~
##  8 hakai-jsp-8 HumanObserva~ Oncorhynchus ~ DE101N1 present      Animal~
##  9 hakai-jsp-9 HumanObserva~ Oncorhynchus ~ DE101N1 present      Animal~
## 10 hakai-jsp-10 HumanObserva~ Oncorhynchus ~ DE101N1 present      Animal~
## # ... with 262,445 more rows, and 14 more variables: Phylum <chr>, Class <chr>,
## #   Order <chr>, Family <chr>, Genus <chr>, Species <chr>, eventDate <date>,
## #   decimalLatitude <dbl>, decimalLongitude <dbl>, geodeticDatum <chr>,
## #   minimumDepthInMeters <dbl>, maximumDepthInMeters <dbl>,
## #   samplingProtocol <chr>, scientificNameID <dbl>
```

```
##(report <- check_outliers_species(DwCA)) # Need to lsid from marinespecies.org
check_eventdate(DwCA)
```

```
## # A tibble: 0 x 0
```

```
tree <- treeStructure(event, occurrence)
exportTree(tree, "tree.html")
```



```
#report(DwCA) # Currently I get warnings on depths. it looks like depths are supposed to be nega
```

3.2 Example two

126140

Chapter 4

Final Words

We have finished a nice book.