Introducing LivingNorwayR

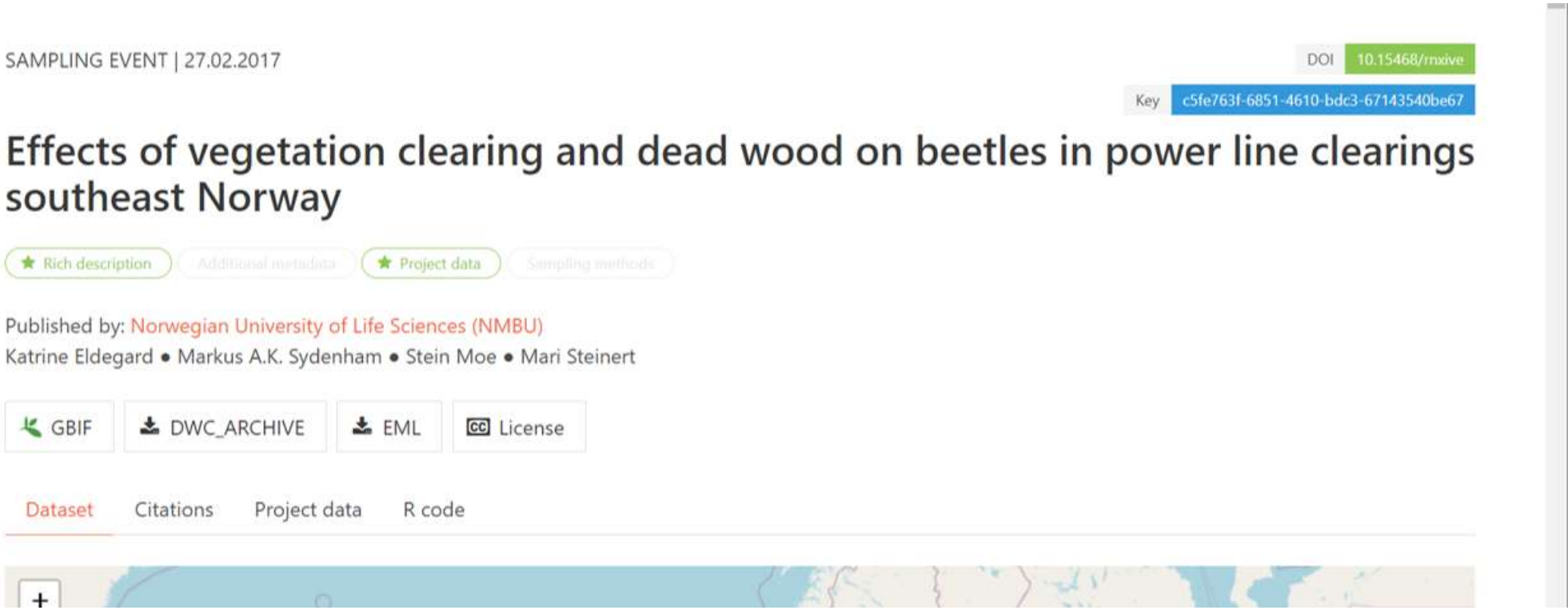
Matt

LivingNorwayR

Motivation:

- Provide a workflow for creating a Darwin Core Standard-compliant data archive ("a data package")
- Facilitate FAIR data/metadata sharing
- Make it easy to interact with the Living Norway Portal

Getting data from the Portal



Living Norway Data Portal

getLNPortalData

```
1 library(LivingNorwayR)
2 suppressMessages(library(tidyverse))
3 beetle_data<-LivingNorwayR::getLNportalData("c5fe763f-6851-4610-bdc3-671435
4
5 class(beetle_data)
```

```
[1] "DwCArchive" "R6"
```

An R6 Class of type DWCArchive is downloaded and imported in to your R session.

Using the data

```
# DWCA consists of a core, extension(s), metadata, &
# metafile (metadata about the DWCA)

beetle_data$getCoreTable()
beetle_data$getExtensionTables()
beetle_data$getMetadata()
```


getCoreTable

```
coreTable<-beetle_data$getCoreTable()
class(coreTable)
```

```
[1] "GBIFEvent"  "DwCGeneric" "R6"
```

```
coreTable_df<-coreTable$exportAsDataFrame()
names(coreTable_df)
```

```
 [1] "id"                                "datasetName"
 [3] "eventID"                           "locationID"
 [5] "country"                           "municipality"
 [7] "decimalLatitude"                   "decimalLongitude"
 [9] "geodeticDatum"                     "coordinateUncertaintyInMeters"
```


Visualisation example

Code | Output

```
library(leaflet)
map<-leaflet(data = coreTable_df) %>%
  addProviderTiles(providers$Esri.NatGeoWorldMap) %>%
  addMarkers(~decimalLongitude, ~decimalLatitude,
             popup = ~as.character(municipality),
             label = ~as.character(municipality))
```


getExtensionTables

```
occ<-beetle_data$getExtensionTables()
occ<-occ[[1]]$exportAsDataFrame()
names(occ)
```

```
 [1] "id"                   "institutionCode"      "basisOfRecord"
 [4] "occurrenceID"         "recordedBy"           "organismQuantity"
 [7] "organismQuantityType" "occurrenceStatus"     "eventID"
[10] "eventDate"            "locationID"           "identifiedBy"
[13] "scientificName"       "kingdom"              "order"
[16] "taxonRank"
```


Visualisation examples

Code | Output

```
Meligethes<-occ %>%
  select(scientificName) %>%
  filter(grepl("Meligethes",scientificName))
p=Meligethes %>%
  group_by(scientificName) %>%
  tally() %>%
  ggplot(aes(n,reorder(scientificName,n)))+
    geom_bar(stat="identity", fill="darkblue")+
  labs(x="Records", y="Scientific name")+
  theme_classic()
library(rphylopic)
img<- name_search(text = "Meligethes", options = "namebankID")[[1]] # find
img_id_all <- name_images(uuid = img$uid[1])  # list images

img_pic<-image_data(img_id_all$supertaxa[[1]]$uid, size = 256)[[1]] # get a
```


getMetadata

We can access the metadata object as well

```
metadata=beetle_data$getMetadata()
Title<-metadata$getTitle()
Creator<-metadata$getCreatorInfo()

paste0("The project, ","'", Title, "'", " was created by ", Creator[[1]]$in
      )
```

```
[1] "The project, 'Effects of vegetation clearing and dead wood on beetles in
power line clearings southeast Norway' was created by Katrine Eldegard"
```


Converting your own data to Living Norway/DWCA format

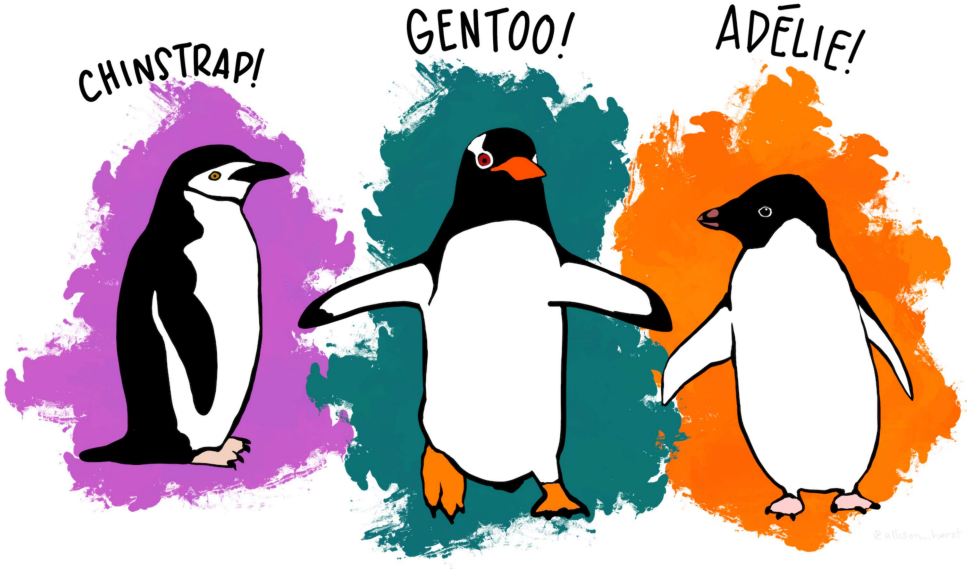
A dataset

An ecological dataset generally has “similar” elements. An **Event** (e.g. some sampling method), a set of **Occurrences** (e.g. a list of species/taxa observed) and some **Measurement or fact** (e.g. measurements of size, environmental covariates etc.). **Metadata** (data about the data) describes the dataset for other people.

Penguins

Let’s use a well-known and openly available dataset from R; The Palmer Penguins dataset.

This dataset consists of observations and measurements of three different species of penguin in Antarctica.



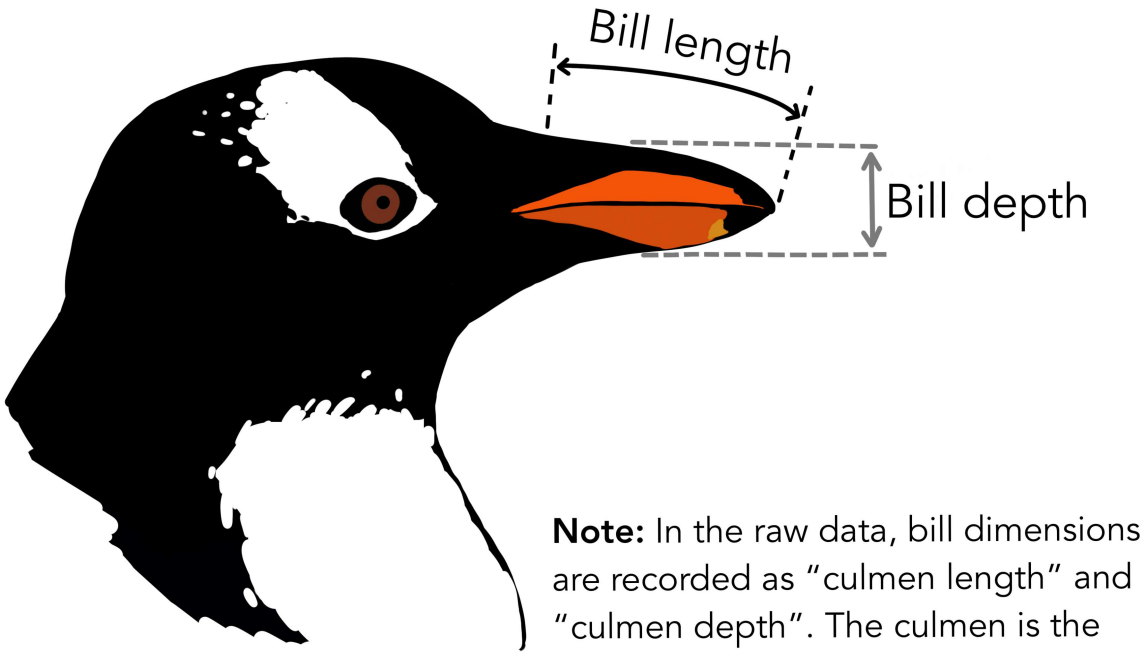
The three species of penguin. Artwork by Allison Horst

Explore the data

```
library(palmerpenguins)
penguin_data<-penguins_raw
penguin_data %>%
  head() %>%
   kableExtra::kable() %>%
   kableExtra::kable_styling("striped", full_width = F) %>%
 kableExtra::scroll_box(width = "1200px", height = "250px")
```

studyName	Sample Number	Species	Region	Island	S
PAL0708	1	Adelie	Anvers	Torgersen	A

Each row is a single individual penguin of one of the three species. There are measures of body size (bill and flipper lengths), sex (male or female), as well as information on egg laying date and stable isotope analysis from blood samples.



Bill length and depth measurements for each penguin. Artwork by Allison Horst

Mapping to Darwin Core

Deciding on the Core

The first task is to decide how our data will be structured.

```
1 LivingNorwayR::getGBIFCoreClasses()[1:6]#Truncated
```

```
$GBIFEvent
http://rs.tdwg.org/dwc/terms/Event - Event
An action that occurs at some location during some time.

    Defined in: https://dwc.tdwg.org/
    IRI: http://rs.tdwg.org/dwc/terms/Event
    Version IRI: http://rs.tdwg.org/dwc/terms/version/Event-2018-09-06
    Type: Class
    Date modified: 2018-09-06
    Executive committee decisions:
        http://rs.tdwg.org/decisions/decision-2014-10-26_15
    Examples:
        A specimen collection process. A camera trap image capture.  A marine
trawl.
    Miscellaneous information:
```


Deciding on the extensions

We have species data for each Event and we can include an Occurrence table as an extension.

```
1 LivingNorwayR::getGBIFExtensionClasses()[1:6] # Truncated
```

```
$GBIFMultimedia
http://rs.tdwg.org/ac/terms/Multimedia - Multimedia
The Audubon Core is a set of vocabularies designed to represent metadata for
biodiversity multimedia resources and collections. These vocabularies aim to
represent information that will help to determine whether a particular
resource or collection will be fit for some particular biodiversity science
application before acquiring the media. Among others, the vocabularies address
such concerns as the management of the media and collections, descriptions of
their content, their taxonomic, geographic, and temporal coverage, and the
appropriate ways to retrieve, attribute and reproduce them.

    IRI: http://rs.tdwg.org/ac/terms/Multimedia
    Version IRI: http://rs.tdwg.org/ac/terms/Multimedia
    Type: class
    Date modified: 2015-03-19
    Notes:
```


The Event Core

We can get a list of the terms associated with an Event by using this code:

```
1 getGBIFEventMembers()[1:6] # Truncated
```

```
$`http://purl.org/dc/terms/type`
http://purl.org/dc/terms/type - Type (DEPRECATED)
The nature or genre of the resource.

    Defined in: https://dwc.tdwg.org/
    IRI: http://purl.org/dc/terms/type
    Type: Property
    Date modified: 2008-01-14
    Is replaced by: http://purl.org/dc/elements/1.1/type
    Notes:
        To provide a string literal value for type, use dc:type rather than
this term. In accordance with the Darwin Core RDF guide, rdf:type should be
used instead of this term to indicate an IRI value for type.
    Executive committee decisions:
        http://rs.tdwg.org/decisions/decision-2009-12-07_1
```


The Event Core

GBIF recommends some required and suggested terms for Events here. These include **eventID**, **eventDate**, **samplingProtocol**, **samplingSizeValue** and **samplingSizeUnit** as required. Some of the strongly recommended elements that it makes sense for us to include are **parentEventID**, **countryCode**, **locationID decimalLatitude**, **decimalLongitude**, **geodeticDatum** and **coordinateUncertaintyInMeters**. We can also add **type**,**datasetName**, **ownerInstitutionCode**, **country**, **year**, **month** and **day**.

Parent Events

Each event is a part of a higher level Event which is referred to as a Parent Event. The Parent Event in our case is represented by the “studyName” column. This represents a unique expedition carried out at a separate time. We can include this information in the Event table.

Each Parent Event needs a unique persistent identifier, **parentEventID**, which we can obtain from the uuid package.

```
penguin_data<-penguin_data %>% janitor::clean_names()
penguin_data<-penguin_data %>%
  group_by(study_name) %>%
  mutate(
    parentEventID = uuid::UUIDgenerate(use.time = FALSE)
  )
```


There are different date ranges for each parent event and these need to be added as an **eventDate.**

```
# Event Date for parentIDs

parent_penguinEvent=penguin_data %>%
  group_by(parentEventID) %>%
  summarise(min=min(date_egg), max=max(date_egg)) %>%
  mutate(eventDate=paste0(min,"/", max)) %>% mutate(eventID=parentEventID)
```


We can also add some wider scale geographic information to the parent events. Such as **continent** and **islandGroup**.

```
# Event continent and islandGroup for parentIDs
parent_penguinEvent=parent_penguinEvent %>%
  mutate(continent="Antarctica") %>%
  mutate(islandGroup="Palmer Archipelago") %>%
  select(!c(min,max))
```


Event

Let's start with the **type**, **datasetName** and **ownerInstitutionCode**. The **type** is "Event", the **datasetName** is "Palmer-penguins" and the Palmer Station Antarctica LTER **ownerInstitutionCode** is "PAL".

```
penguin_data=penguin_data %>%
  mutate(ownerInstitutionCode="PAL",
         type="Event",
         datasetName="Palmer-penguins")
```


Each Event also needs a unique identifier (**eventID**) and we can use the same approach as above. This time as each row is an Event we need to make sure the dataframe is ungrouped.

```
penguin_data=penguin_data %>%
  ungroup() %>%
  rowwise()%>%
    mutate(
    eventID = uuid::UUIDgenerate(use.time = FALSE)
  )
```


Again we can include an **eventDate** for each event.

```
penguin_data=penguin_data %>%
    mutate(
    eventDate = date_egg) %>%
  mutate(day=lubridate::day(date_egg),
         month=lubridate::month(date_egg),
         year=lubridate::year(date_egg))
```

As each event is a sample in the penguins dataset where they measured a individual penguin we can set the **sampleSizeValue** as 1. The **sampleSizeUnit** can be “Adult penguin”.

```
penguin_data=penguin_data %>%
  mutate(sampleSizeValue=1) %>%
  mutate(sampleSizeUnit="Adult penguin")
```


We can get the **samplingProtocol** for each event by looking at the original data package (links can be found when you type ?? palmerpenguins::penguins_raw in to the R Console). All three parent events have the same protocol.

```
1 penguin_data=penguin_data %>%
2   mutate(samplingProtocol= "Each season, study nests, where pairs of adults
```


countryCode is the two letter standard (using ISO 3166-1-alpha-2) code for a country. Antarctica, defined as the territories south of 60°S is given the code AQ.

```
penguin_data =penguin_data %>%
  mutate(countryCode="AQ",
         country="Antarctica")
```


The **locationID** can be a global unique identifier or an identifier specific to the data set. We have the region and island for the penguins data so we can use these to develop a data specific identifier.

```
penguin_data=penguin_data %>%
  mutate(locationID=paste0(region, "_", island))
```


We are not provided with precise coordinates (**decimalLatitude**, **decimalLongitude**) for the samples in the penguin data. However, we can get the centroid for each island.

```
penguin_data=penguin_data %>%
  mutate(decimalLatitude=
           case_when(
             island=="Torgersen"~-64.77308,
             island=="Biscoe"~-65.4333316,
             island=="Dream"~-64.7333333,
             TRUE~as.numeric(NA))
  ) %>%
  mutate(decimalLongitude=
           case_when(
             island=="Torgersen"~-64.07413,
             island=="Biscoe"~-65.499998,
             island=="Dream"~-64.2333333,
             TRUE~as.numeric(NA))

  ) %>%
  mutate(geodeticDatum="WGS 84") #default geodeticDatum is WGS 84
```


As the coordinates are just the centroid for the island we need to include some measure of uncertainty (**coordinateUncertaintyInMeters**).

```
penguin_data=penguin_data %>%
  mutate(coordinateUncertaintyInMeters=
           case_when(
             island=="Torgersen"~400,
             island=="Biscoe"~500,
             island=="Dream"~400,
             TRUE~as.numeric(NA))
  )
```


Finally, we can create the Event core by selecting those elements that we have listed above.

```
eventDF=penguin_data %>%
  select(c("datasetName",
           "ownerInstitutionCode",
           "parentEventID",
           "eventID",
           "samplingProtocol",
           "sampleSizeValue",
           "sampleSizeUnit",
           "eventDate",
           "year",
           "month",
           "day",
           "locationID",
           "country",
           "countryCode",
           "decimalLatitude",
           "decimalLongitude",
           "geodeticDatum",
           "coordinateUncertaintyInMeters"))
```


Then we need to add in the Parent Events in to the Event dataframe.

```
eventDF=eventDF %>%
  mutate(continent=NA) %>%
  mutate(islandGroup=NA) %>%
  mutate(eventDate=as.character(eventDate)) %>% bind_rows(parent_penguinEve
```


The final stage is to initialise an event object in the livingNorwayR

```
GBIF_Event=initializeGBIFEvent(eventDF, idColumnInfo = "eventID", nameAutoM
```


The Occurrence extension

We can find the supported terms for the Occurrence extension by using the following function.

```
1 getGBIFOccurrenceMembers()[1:6]#Truncated
```

```
$`http://purl.org/dc/terms/type`
http://purl.org/dc/terms/type - Type (DEPRECATED)
The nature or genre of the resource.

    Defined in: https://dwc.tdwg.org/
    IRI: http://purl.org/dc/terms/type
    Type: Property
    Date modified: 2008-01-14
    Is replaced by: http://purl.org/dc/elements/1.1/type
    Notes:
        To provide a string literal value for type, use dc:type rather than
this term. In accordance with the Darwin Core RDF guide, rdf:type should be
used instead of this term to indicate an IRI value for type.
    Executive committee decisions:
        http://rs.tdwg.org/decisions/decision-2009-12-07_1
```


GBIF also has a list of required and recommended terms for Occurrence data. The required terms are **occurrenceID**, **basisOfRecord**,**scientificName**, **eventDate** (included in the event core). The recommended terms include **countryCode**, **taxonRank**,**kingdom**, **decimalLatitude**, **decimalLongitude**, **geodeticDatum**, **coordinateUncertaintyInMeters**,**individualCount**, **organismQuantity** and **organismQuantityType**, some of which are included in the event core.

The **type** is an “Occurrence”. The **collectionCode** can be “Palmer Station Antarctica LTER” and the **occurrenceID** should be a globally unique identifier.

```
penguin_data=penguin_data %>%
  mutate(type="Occurrence",
    collectionCode="Palmer Station Antarctica LTER") %>%
  mutate(occurrenceID=uuid::UUIDgenerate(use.time = FALSE))
```


The **basisOfRecord** records how the observation was made.

```
penguin_data=penguin_data %>%
  mutate(basisOfRecord="HumanObservation")
```


organismQuantity and **organismQuantityType**, are 1 individual penguin.

```
penguin_data=penguin_data %>%
  mutate(organismQuantity= 1,
         organismQuantityType= "individual")
```


For the **scientificName** and **vernacularName** we can use the species name from the original data set.

```
penguin_data=penguin_data %>%
  mutate(scientificName=gsub("[\\(\\)]", "", regmatches(species, gregexpr("
         vernacularName=gsub("\\s*\\([^\\)]+\\)","",species))
```


taxonRank, **kingdom**, **phylum**, **class**, **order**, **family** and **genus** all relate to the scientific name of the species. The **taxonRank** will be **species** because we have identified the penguins to the species level.

```
penguin_data=penguin_data %>%
  mutate(kingdom="Animalia",
         phylum="Chordata",
         class="Aves",
         order= "Sphenisciformes",
         family= "Spheniscidae",
         genus="Pygoscelis",
         ) %>%
  mutate(taxonRank="species")
```


Finally, as we did with the Event core, we can create the occurrence extension by selecting those elements that we have listed above and create a livingNorwayR object.

```
occ_ext=penguin_data %>%
  select(type,collectionCode,basisOfRecord,occurrenceID, organismQuantity,
         eventID, eventDate, scientificName,kingdom, phylum, class, order,

GBIF_Occ=initializeGBIFOccurrence(occ_ext, idColumnInfo = "occurrenceID",na
```


The Measurement or Fact extension

Our final extension is the “Measurement or Fact extension”. We can look at the definition of this extension using the following code:

```
1 getGBIFExtensionClasses()$GBIFMeasurementOrFact
```

```
http://rs.tdwg.org/dwc/terms/MeasurementOrFact - Measurement or Fact
A measurement of or fact about an rdfs:Resource
(http://www.w3.org/2000/01/rdf-schema#Resource).

    Defined in: https://dwc.tdwg.org/
    IRI: http://rs.tdwg.org/dwc/terms/MeasurementOrFact
    Version IRI: http://rs.tdwg.org/dwc/terms/version/MeasurementOrFact-2018-
09-06
    Type: Class
    Date modified: 2018-09-06
    Notes:
        Resources can be thought of as identifiable records or instances of
classes and may include, but need not be limited to dwc:Occurrence,
dwc:Organism, dwc:MaterialSample, dwc:Event, dwc:Location,
dwc:GeologicalContext, dwc:Identification, or dwc:Taxon.
    Executive committee decisions:
```


GBIF has a list of properties for this extension. These include **measurementID measurementType**, **measurementValue**, **measurementAccuracy**, **measurementUnit**, **measurementDeterminedDate**, **measurementDeterminedBy**, **measurementMethod** and **measurementRemarks**, none of which are required.

Let's start with the **measurementID**. We should also include the **occurrenceID** and **eventID** so that we know which individual and in which sampling event the measurements were taken.

```
M_or_f=penguin_data %>%
  mutate(measurementID=uuid::UUIDgenerate(use.time = FALSE))
```


There are a number of measurements that we can include in this extension.

```
M_or_f=M_or_f %>%
  select(measurementID,parentEventID,occurrenceID,eventID,
         culmen_length_mm, culmen_depth_mm, delta_13_c_o_oo, delta_15_n_o_o
```


We need to pivot the data so that all the measurement types go in to a single column called **measurementType**. All the measurements go in to a column called **measurementValue**.

```
M_or_f=M_or_f %>%
  pivot_longer(cols = c(culmen_length_mm, culmen_depth_mm, delta_13_c_o_oo,
                values_to = "measurementValue"
  ) %>%
  drop_na(measurementValue)
```


Finally we create a measurement or fact object using livingNorwayR.

```
GBIF_Measure=initializeGBIFMeasurementOrFact(M_or_f, idColumnInfo = "measur
```


Metadata

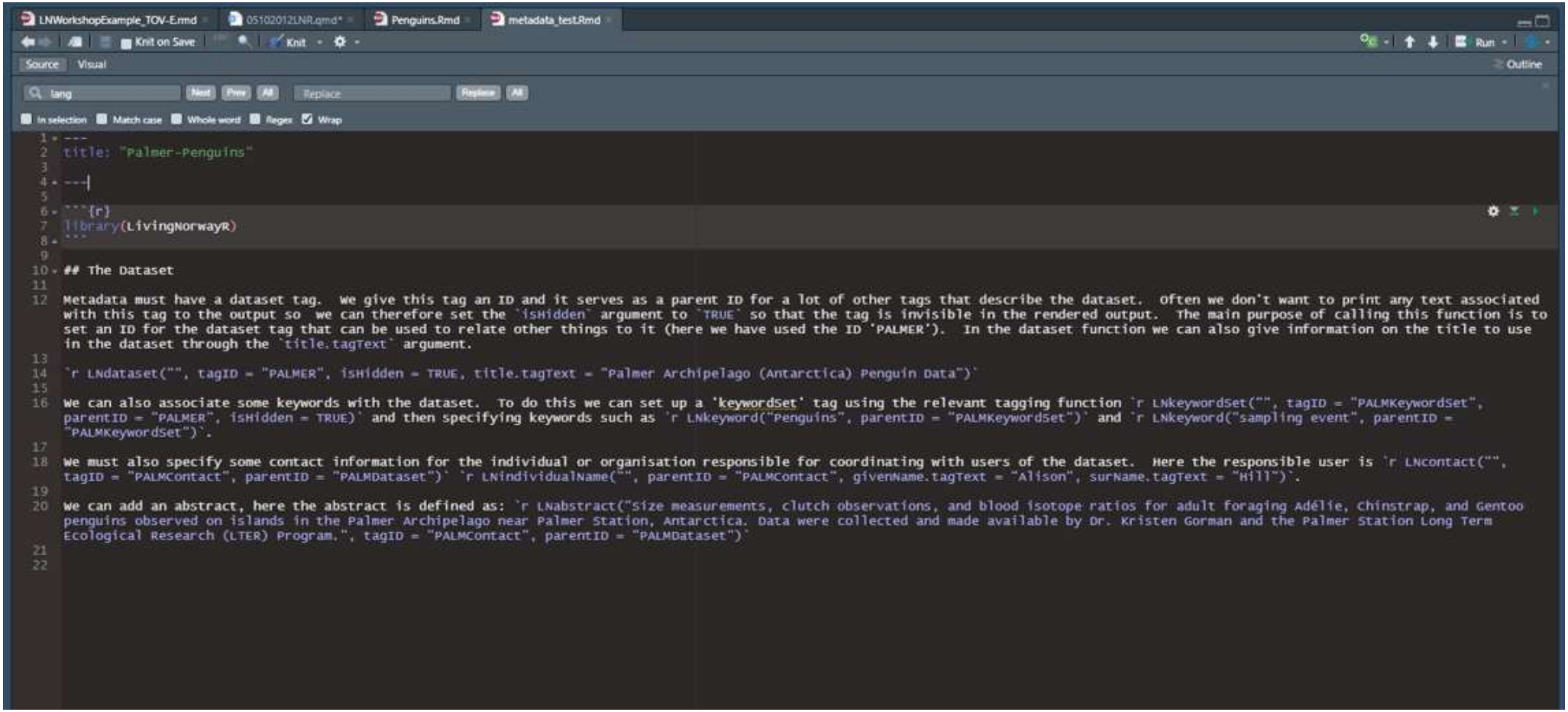
EML Metadata

In Darwin Core archive files the standard format for handling metadata is EML, a flavour of XML specifically designed for the handling of ecological metadata.

Writing EML in R is hard!!!!!! There are a few options available.

Metadata in LivingNorwayR

- The vision is to have a RMarkdown interface that allows one to render to a data report and EML at the same time
- The current version is a little buggy, but we have a development version that should be released by the end of this year


LNWorkshopExample_TOV-E.rmd
05102012LNR.qmd*
Penguins.Rmd
metadata_test.Rmd
Knit on Save
Knit
Run
Source
Visual
Outline
lang
Next
Prev
All
Replace
Replace
All
In selection
Match case
Whole word
Regex
Wrap
---
title: "Palmer-Penguins"

```{r}
library(LivingNorwayR)
```
The Dataset
Metadata must have a dataset tag. We give this tag an ID and it serves as a parent ID for a lot of other tags that describe the dataset. Often we don't want to print any text associated with this tag to the output so we can therefore set the `isHidden` argument to `TRUE` so that the tag is invisible in the rendered output. The main purpose of calling this function is to set an ID for the dataset tag that can be used to relate other things to it (here we have used the ID 'PALMER'). In the dataset function we can also give information on the title to use in the dataset through the `title.tagText` argument.
`r LNdataset("", tagID = "PALMER", isHidden = TRUE, title.tagText = "Palmer Archipelago (Antarctica) Penguin Data")`
We can also associate some keywords with the dataset. To do this we can set up a 'keywordSet' tag using the relevant tagging function `r LNkeywordSet("", tagID = "PALMKeywordSet", parentID = "PALMER", isHidden = TRUE)` and then specifying keywords such as `r LNkeyword("Penguins", parentID = "PALMKeywordSet")` and `r LNkeyword("sampling event", parentID = "PALMKeywordSet")`.
We must also specify some contact information for the individual or organisation responsible for coordinating with users of the dataset. Here the responsible user is `r LNcontact("", tagID = "PALMContact", parentID = "PALMDataset")` `r LNindividualName("", parentID = "PALMContact", givenName.tagText = "Alison", surName.tagText = "Hill")`.
We can add an abstract, here the abstract is defined as: `r LNabstract("Size measurements, clutch observations, and blood isotope ratios for adult foraging Adélie, Chinstrap, and Gentoo penguins observed on islands in the Palmer Archipelago near Palmer Station, Antarctica. Data were collected and made available by Dr. Kristen Gorman and the Palmer Station Long Term Ecological Research (LTER) Program.", tagID = "PALMContact", parentID = "PALMDataset")`

Palmer-Penguins

```
library(LivingNorwayR)
```

```
## Welcome to LivingNorwayR for more information
##   about the project see https://livingnorway.no/.
##   If you have any problems please submit an issue
##                          at https://github.com/LivingNorway/LivingNorwayR
```

The Dataset

Metadata must have a dataset tag. We give this tag an ID and it serves as a parent ID for a lot of other tags that describe the dataset. Often we don't want to print any text associated with this tag to the output so we can therefore set the `isHidden` argument to `TRUE` so that the tag is invisible in the rendered output. The main purpose of calling this function is to set an ID for the dataset tag that can be used to relate other things to it (here we have used the ID 'PALMER'). In the dataset function we can also give information on the title to use in the dataset through the `title.tagText` argument.

We can also associate some keywords with the dataset. To do this we can set up a 'keywordSet' tag using the relevant tagging function and then specifying keywords such as Penguins and sampling event.

We must also specify some contact information for the individual or organisation responsible for coordinating with users of the dataset. Here the responsible user is Alison Hill.

We can add an abstract, here the abstract is defined as: Size measurements, clutch observations, and blood isotope ratios for adult foraging Adélie, Chinstrap, and Gentoo penguins observed on islands in the Palmer Archipelago near Palmer Station, Antarctica. Data were collected and made available by Dr. Kristen Gorman and the Palmer Station Long Term Ecological Research (LTER) Program.

```
GBIF_metadata<-metadata$initialize("C:/Users/matthew.grainger/Documents/Pro
```


Create a DWC-a file

```
coreDWC<-GBIF_Event
extDWC<-c(GBIF_Occ, GBIF_Measure)
metadata<- GBIF_metadata


initializeDwCArchive(coreDwC, extDwC, metadata)
```


Where to get help with the Package

GitHub please submit an issue

If you want to contribute please do so through a pull request (this can be code, use-cases, documentation etc.)