The EML R package

Claas-Thido Pfaff, Carl Boettiger, Karthik Ram, Matt Jones

## Why metadata matters?

* You are interested in salmon species
* Distribution across N.A.
* ~ past 30 years
* Only find publications (no datasets)
* You ask the authors and your network

He Claas

A former colleague of mine was working intensly with salmon  
species in North America over years. He is retired now but we  
still have his data laying around in our archive. I hope this  
is useful to you!  
-- *All the best Karl*

Attachment.csv

## Why metadata matters?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| river | spp | stg | ct | dates |
| SAC | king | smolt | 293 | 1991-10-10 |
| SAC | king | parr | 410 | 1992-11-10 |
| AM | ccho | smolt | 210 | 1993-10-10 |

* These you guess:
* river: Abbr. of collection sites (full name)
* spp: Abbr. of species names (full name)
* stg: The life stage of fish
* But what about the rest/details?
* ct: Is numeric (Measured, Statistics, Method)
* dates: Which date format? (YMD, YDM)

## Why metadata matters?

* You ask and get the answer:

He Claas

I just checked the data again and fortunately I was involved  
in that particular data collection! The information you need is:

river: sac = The sacramento river, am = The american river  
spp: king = King Salmon, ccho = Coho Salmon  
stg: par = Third life stage, smolt = Fourth life stage  
ct: It is the count of life fish caught in traps  
dates: The date format is YMD

-- *All the best Karl*

* With that information you can start use the data!

## Why metadata matters?

* We learn:
* Without proper metadata
* data unusalbe and lost
* Metadata standards (DwC, EML)
* Ecological Metadata Language ([EML](assets/files/EML_example.xml), XML)
* Allows to capture aspects of data:
  + Units and categories
  + Temporal and spatial coverage ...
  + Contact information ... and much more
* In a structured machine readable way

Morpho Data-Up Fegraus et al. 2005 BEF-Data

## The package (About)

* Metadata tools
* Morpho (DataOne, KNB)
* Metacat (DataOne, KNB)
* Data-Up (Californian Libraries)
* BEF-Data (BEF-China)
* EML for R (initial commit 24 Jun 2013)
* Motivation (R package for EML)
* Many data undescribed; Biologists in R
* Introduces a wide spread standard to R
* Read + Write metadata
* Publish (Data + Metadata)

EML

## The package (About)

* Part of the rOpenSci community
* Data-Acess, Vizualisation, Reproducibility... 30+)
* rgbif (Global Biodiversity Information Facility)
* taxize (20+ Taxonomic Databases for e.g. species name resolving)
* rBEFdata (Access to BEFdata data management platforms)
* The EML package is developed by:

rOpenSci

--- bg:#EEE

## The package (Install)

* Not yet available via CRAN
* Source code via GitHub
* <https://github.com/ropensci/EML>
* Devtools (Hadley Wickham)

install.packages("devtools")  
library("devtools")

* Install from github

install\_github("ropensci/EML", build=FALSE, dependencies=c("DEPENDS", "IMPORTS"))  
library("EML")

Install Script.R

## Typical metadata

- eml  
 - dataset  
 - creator (o)  
 - contact (o)  
 - publisher  
 - title (o)  
 - pubDate  
 - keywords  
 - abstract  
 - intellectualRights (o)  
 - methods  
 - coverage  
 - dataTable (o)  
 - physical  
 - attributeList  
 - additionalMetadata

## Typical metadata (add core)

- eml  
 - dataset  
 - creator (o)  
 - contact (o)  
 - publisher  
 - title (o)  
 - pubDate  
 - keywords  
 - abstract  
 - intellectualRights (o)  
 - methods  
 - coverage  
 - dataTable (x)  
 - physical  
 - attributeList  
 - additionalMetadata

## Create metadata

* River site used for collection
* river: sac = The sacramento river, am = The american river
* Scientific species names
* spp: king = King Salmon, ccho = Coho Salmon
* Life stage of fish
* stg: par = Third life stage, smolt = Fourth life stage
* Count of life fish in traps
* ct: numeric
* The date of data collection:
* dates: Format is Year, Month, Day

Attachment.csv

## Create metadata

* EML package adds data.set(data.frame, col.defs =, unit.defs =)
* col.defs (plain text definition)

col\_defs = c("River site used for collection",  
 "Species common name",  
 "Life Stage",  
 "Count of live fish in traps",   
 "Date of collection")

* unit.defs (factor => levels, dates => YYYY or MM-DD-YY, numeric => unit list [KNB](http://bit.ly/1vEmFnE))

unit\_defs = list(c(SAC = "The Sacramento River", AM = "The American River"),  
 c(king = "King Salmon", ccho = "Coho Salmon"),  
 c(parr = "third life stage", smolt = "fourth life stage"),  
 c(unit = "number"),  
 c(format = "YYYY-MM-DD"))

## Assemble

* Assemble
* core metadadta
* data
* with data.set()

described\_dataset = data.set(undescribed\_data,  
 col.defs = col\_defs,  
 unit.defs = unit\_defs)

* We use the variables just prepared
* col\_defs
* unit\_def

## Assemble (inspect)

## Object of class "data.set"  
## river spp stg ct dates  
## 1 SAC king smolt 293 1991-10-10  
## 2 SAC king parr 410 1992-11-10  
## 3 AM ccho smolt 210 1993-10-10  
## Slot "col.defs":  
## [1] "River site used for collection" "Species common name"   
## [3] "Life Stage" "Count of live fish in traps"   
## [5] "Date of collection"   
##   
## Slot "unit.defs":  
## [[1]]  
## SAC AM   
## "The Sacramento River" "The American River"   
##   
## [[2]]  
## king ccho   
## "King Salmon" "Coho Salmon"   
##   
## [[3]]  
## parr smolt   
## "third life stage" "fourth life stage"   
##   
## [[4]]  
## unit   
## "number"   
##   
## [[5]]  
## format   
## "YYYY-MM-DD"

--- bg:#EEE

## Your turn (core metadata)

* Get the data

undescribed\_data = read.csv("http://bit.ly/11Q4GOt")

* Create the colum definitions (character vector)
* Save to variable (e.g col\_defs)
* Create unit definitions (list)
* Save to variable (e.g unit\_defs)
* Use: unit = "number" (for the count)
* Use: format = "YYYY-MM-DD" (for the date)
* Assemble (data.set(undescribed\_data, col.defs = col\_defs, unit.defs = unit\_defs))
* save result to variable (e.g data\_with\_core\_metadata)

Failed? Your rescue!

## Typical metadata (add more "x")

- eml  
 - dataset  
 - creator (x)  
 - contact (o)  
 - publisher  
 - title (o)  
 - pubDate  
 - keywords  
 - abstract  
 - intellectualRights (o)  
 - methods  
 - coverage  
 - dataTable (done)  
 - physical  
 - attributeList  
 - additionalMetadata

## Objects (excursion)

* create an instance from an object

new\_contact\_instance = new("contact")

* show all variables (slotnames)

getSlots("contact")

## individualName organizationName positionName   
## "individualName" "character" "character"   
## address phone electronicMailAddress   
## "address" "character" "character"   
## onlineUrl userID references   
## "character" "character" "ListOfreferences"

slotNames("contact")

## Objects (excursion)

* Slots can contain var. data types:
* character
* numeric
* lists
* other objects
* Subsetting (not $ but @)

the\_instance@slotname

* coercions

as("22", "numeric")

## Add creator (name, mail)

claas\_creator = new("creator",   
 individualName = new("individualName",   
 givenName = "Claas-Thido",   
 surName = "Pfaff"),   
 electronicMailAddress = "fake@test.com")

getSlots("creator")

## individualName organizationName positionName   
## "individualName" "character" "character"   
## address phone electronicMailAddress   
## "address" "character" "character"   
## onlineUrl userID references   
## "character" "character" "ListOfreferences"

## Add creator (name, mail)

* Convenient with coercion

claas\_person = eml\_person("Claas-Thido Pfaff <fake@test.com>")

claas\_creator = as(claas\_person, "creator")

* Subsetting

claas\_creator@individualName@surName

## [1] "Pfaff"

claas\_creator@individualName@givenName

## [1] "Claas-Thido"

## Add creator (address)

getSlots("creator")

## individualName organizationName positionName   
## "individualName" "character" "character"   
## address phone electronicMailAddress   
## "address" "character" "character"   
## onlineUrl userID references   
## "character" "character" "ListOfreferences"

getSlots("address")

## deliveryPoint city administrativeArea   
## "character" "character" "character"   
## postalCode country references   
## "character" "character" "ListOfreferences"

## Add creator (address)

address = new("address",  
 deliveryPoint = "Universität Leipzig, Johannisallee 21",  
 city = "Leipzig",  
 postalCode = "04103",  
 country = "GER")

claas\_creator@address = address

* Why is this important?

## It is important because!

* Class names and slot names
* get fields in the EML!

<creator>  
 <individualName>  
 <givenName>Claas-Thido</givenName>  
 <surName>Pfaff</surName>  
 </individualName>  
 <address>  
 <deliveryPoint>Universität Leipzig, Johannisallee 21</deliveryPoint>  
 <city>Leipzig</city>  
 <postalCode>04103</postalCode>  
 <country>GER</country>  
 </address>  
 <electronicMailAddress>fake@test.com</electronicMailAddress>  
</creator>

--- bg:#EEE

## Your turn (add contact)

- eml  
 - dataset  
 - creator (done)  
 - contact (x)  
 - publisher  
 - title (o)  
 - pubDate  
 - keywords  
 - abstract  
 - intellectualRights (o)  
 - methods  
 - coverage  
 - dataTable (done)  
 - physical  
 - attributeList  
 - additionalMetadata

--- bg:#EEE

## Your turn (add contact)

* Add a contact
* Use you = eml\_person("Your Name <yourmail@provider.com>")
* Add an address
* address = new("address", deliveryPoint = "....")
* also add: city, postalCode, country
* hint: slotNames("address")
* Do not forget to assign the address to your contact!
* you@address = address

Failed? Your rescue!

--- bg:#EEE

## Your turn (add contact 1)

* Convenient
* with eml\_person()
* Create address
* Assign address

myname = eml\_person("J. Steidle <steidle@fake.com>")  
myaddress = new("address",  
 deliveryPoint = "University Hohenheim, Schloss Hohenheim 1",  
 city = "Stuttgart",  
 postalCode = "70599",  
 country = "GER")  
myname@address = myaddress

Failed? Your rescue!

--- bg:#EEE

## Your turn (add contact 2)

* More verbose
* More basic (no wrapper function)
* Everything in one block

new("contact", individualName = new("individualName",   
 givenName = "Claas-Thido Pfaff",   
 surName = "Pfaff"),   
 electronicMailAddress = "claas-thido.pfaff@uni-leipzig.de",  
 phone = "+49-341-97-38587",  
 address = new("address",  
 deliveryPoint = "Universität Leipzig, Johannisallee 21",  
 city = "Leipzig",  
 postalCode = "04103",  
 country = "GER")

Failed? Your rescue!

## Typical metadata (add more "x")

- eml  
 - dataset  
 - creator (done)  
 - contact (done)  
 - publisher  
 - title (x)  
 - pubDate  
 - keywords  
 - abstract  
 - intellectualRights (x)  
 - methods  
 - coverage  
 - dataTable (done)  
 - physical  
 - attributeList  
 - additionalMetadata

## Put all together

* The eml() command assembles

data = eml(dat = described\_dataset,  
 title = "Count of life fish in traps",  
 contact = claas\_contact,  
 creator = claas\_creator,  
 intellectualRights = "CC0, Creative commons zero"  
 )

* Write out the EML to a file

eml\_write(data, file="mymetadata.xml")

## [1] "mymetadata.xml"

* More often you use .eml

EML file CSV file

## Publish (curr. figshare, knb)

* Publish to figshare (requires rfigshare package)

eml\_publish("mymetadata.xml",   
 description="Example EML file from EML",  
 categories = "Ecology",   
 tags = "EML",   
 destination="figshare")

Your article has been created! Your id number is 1256252  
[1] 1256252

* Requires
* R-Package (rfigshare)
* Figshare account (<http://figshare.com>)

## Publish (curr. figshare, knb)

## Publish (curr. figshare, knb)

--- bg:#EEE

## Your turn (assemble/write out)

* Assemble
* The dataset + core metadata (you created)
* the contact (you created)
* Add a title and usage rights

title = "Count of life fish in traps of the sacramento and american river"

license = "CC0, http://creativecommons.org/publicdomain/zero/1.0"

* Hint: final = eml(dat = ..., title =, contact = ...)
* Write out the metadata and data
* Hint: eml\_write(final, file = "xy.xml")

Failed? Your rescue!

## Read metadata

* Read metadata from any EML formated source (File, URL, KNB-ID)

metadata\_locally = eml\_read("mymetadata.xml")

metadata\_online = eml\_read("http://bit.ly/1viuNDZ")

* Then use eml\_get(metadata, "xy")
* coverage
* contact
* unit.defs
* col.defs
* creator
* data.set ...

## Read metadata

eml\_get(metadata\_online, "contact")

## [1] "Claas-Thido Pfaff <fake@test.com>"

eml\_get(metadata\_locally, "col.defs")

## attribute attribute   
## "River site used for collection" "Species common name"   
## attribute attribute   
## "Life Stage" "Count of live fish in traps"   
## attribute   
## "Date of collection"

## Import data

eml\_get(metadata\_locally, "data.set")

## Object of class "data.set"  
## river spp stg ct dates  
## 1 SAC king smolt 293 1991-10-10  
## 2 SAC king parr 410 1992-11-10  
## 3 AM ccho smolt 210 1993-10-10  
## Slot "col.defs":  
## attribute attribute   
## "River site used for collection" "Species common name"   
## attribute attribute   
## "Life Stage" "Count of live fish in traps"   
## attribute   
## "Date of collection"   
##   
## Slot "unit.defs":  
## $attribute  
## SAC AM   
## "The Sacramento River" "The American River"   
##   
## $attribute  
## king ccho   
## "King Salmon" "Coho Salmon"   
##   
## $attribute  
## parr smolt   
## "third life stage" "fourth life stage"   
##   
## $attribute  
## [1] "number"  
##   
## $attribute  
## [1] "YYYY-MM-DD"

## Import data

* Or only the raw data

eml\_get(metadata\_locally, "data.frame")

## river spp stg ct dates  
## 1 SAC king smolt 293 1991-10-10  
## 2 SAC king parr 410 1992-11-10  
## 3 AM ccho smolt 210 1993-10-10

* Note using
* data.frame not
* data.set here

--- bg:#EEE

## Your turn (Read/Import)

* Import the metadata from here:
* <http://bit.ly/1yhi1b3>
* use eml\_read()
* Read the title (hint: use subset with @)
* Extract contact (hint: eml\_get())
* Get the core data and metadata (hint: eml\_get())
* extract the data.set
* Extract the data.frame

--- bg:#EEE

## Your turn (Read/Import)

eml\_from\_url = eml\_read("http://bit.ly/1yhi1b3")

eml\_from\_url@dataset@title

## [1] "Count of life fish in traps"

eml\_get(eml\_from\_url, "contact")

## [1] "Claas-Thido Pfaff <fake@test.com>"

eml\_get(eml\_from\_url, "data.set")  
eml\_get(eml\_from\_url, "data.frame")

## Wrap-up

* The EML package
* Read/Write metadata
* From any EML source
* Describe your own data
* Store your metadata and reuse it!
* Publication of citable data products
* This was very brief:
* Just visit GitHub for more!
* <https://github.com/ropensci/EML>

Thanks for your attention!

Any questions?

* Find this slides:
* <http://cpfaff.github.io/emlforrcourse>
* Find EML package:
* <https://github.com/ropensci/EML>