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, 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 (recapitulate)

- We want to add metadata to the csv from Karl!
- 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)

```
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

- Undescribed raw dataset

```
undescribed_data
```

```
##   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
```

```
described_dataset = data.set(undescribed_data,
                             col.defs = col_defs,
                             unit.defs = unit_defs)
```

- Metadata from variables just prepared
- `col_defs`
- `unit_def`

Assemble (inspect)

```
described_dataset
```

```
## 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 `described_data`)

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)

```
getSlots("creator")
```

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

```
getSlots("individualName")
```

```
##  salutation   givenName     surName
## "character" "character" "character"
```

Add creator (name, mail)

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

- 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"
```

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)

- Instantiate an address
- Fill the slots

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

- Assign the address to the creator

```
claas_creator@address = address
```

- And/Or everything put together ...

Add creator (single step)

- All of the crator information together

```
claas_creator = new("creator",
             individualName = new("individualName",
                                  givenName = "Claas-Thido",
                                  surName = "Pfaff"),
             electronicMailAddress = "fake@test.com",
             address = new("address",
                           deliveryPoint = "Universität Leipzig, Johannisallee 21",
                           city          = "Leipzig",
                           postalCode    = "04103",
                           country       = "GER")
            )
```

- So why do we need all this
- objects/classes/instances/nesting (EML is XML)

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>")`
- coerce `as(you, "contact")`
- 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>")
myname_contact = as(myname, "contact")
myaddress = new("address",
                deliveryPoint = "University Hohenheim, Schloss Hohenheim 1",
                city          = "Stuttgart",
                postalCode    = "70599",
                country       = "GER")
myname_contact@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 files

```
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 license

```
title = "Count of life fish in traps of the sacramento and american river"
intellectualRights = "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 = "yourfilename.xml")`
- Find it in your current WD

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

```
metadata_online = eml_read("http://bit.ly/1viuNDZ")
```

```
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()`
- Extract contact (hint: `eml_get()`)
- Get the core data and metadata (hint: `eml_get()`)
- extract the data.set
- Extract the data.frame
- If you are in the mood
- Find out the title (hint: use subset `@`)

— bg:#EEE

Your turn (Read/Import)

```
eml_from_url = eml_read("http://bit.ly/1yhi1b3")
```

```
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")
```

```
eml_from_url@dataset@title
```

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

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
- Get the slides
- Get this presentation

- Find EML package:
- https://github.com/ropensci/EML