The Journal of Open Source Software

Ecological Metadata as Linked Data

Carl Boettiger[1]

1 University of California, Berkeley

DOI: 10.21105/joss.01276

Software

- Review
- Repository
- Archive

Submitted: 23 February 2019
Published: 26 February 2019

The Ecological Metadata Language, (EML), is a widely used metadata standard in the ecological and environmental sciences [Michener, Brunt, Helly, Kirchner, & Stafford (1997); Jones, Schildhauer, Reichman, & Bowers (2006); Leinfelder et al. (2010)]. Efforts such as the Long Term Ecological Research (LTER) and the National Center for Ecological Analysis and Synthesis (NCEAS) have used and driven the development of EML over the past two decades, and now major efforts such as the NSF-sponsored National Ecological Observatory Network (NEON), the DataONE Network, are making an unprecedented amount of ecological data available with rich, machine-readable metadata descriptions using EML [Battelle (2016); Keller, Schimel, Hargrove, & Hoffman (2008); Overpeck, Meehl, Bony, & Easterling (2011)]. EML defines a strict XML schema that ensures metadata are machine readable and inter-operable through an XML-schema validation process.

While this provides a predictable but extensible format, it also poses a significant technical barrier to many researchers seeking to analyze existing EML documents or create their own. The creation of EML in the late 1990s also pre-dated the advent of more developer-friendly and user-friendly technologies such as the popular JSON serialization, as well as the advent of semantic or linked data model and it's application in ecoinformatics [Michener & Jones (2012)]. More recently still, the creation of the W3C JSON-LD [Sporny, Longley, Kellogg, Lanthaler, & Lindström (2017)] has brought combined the developer-friendly JSON format with the powerful semantics of the Resource Description Format (RDF) popular in informatics [Schreiber & Raimond (2014)]. This package seeks to bring both of these advantages to bear on the rich-but-clunky XML structure used by EML. By automatically mapping EML into JSON-LD, we are able to realize the benefits easier creation and manipulation of EML structures as JSON objects or R list objects, while relying on the power of JSON-LD semantics (in particular, context, compaction and framing, [Ooms (2017)]) to transform that data to and from EML-schema-compliant XML. This approach opens numerous research applications, ranging from the simple parsing or creation of EML files as R list objects to complex queries and semantic reasoning over large numbers of EML documents with semantic SPARQL queries [Prud'hommeaux & Seaborne (2008)]. The package vignette provides scripted examples of each of these.

```
library(emld)
library(jsonlite)
library(magrittr) # for pipes
library(jqr)      # for JQ examples only
library(rdflib)   # for RDf examples only
```

Reading EML

The `EML` package can get particularly cumbersome when it comes to extracting and manipulating existing metadata in highly nested EML files. The `emld` approach can leverage a rich array of tools for reading, extracting, and manipulating existing EML files.

We can parse a simple example and manipulate is as a familiar list object (S3 object):

```r
f <- system.file("extdata/example.xml", package="emld")
eml <- as_emld(f)

cat(eml$dataset$title)
```

```
## Data from Cedar Creek LTER on productivity and species richness
##   for use in a workshop titled "An Analysis of the Relationship between
##   Productivity and Diversity using Experimental Results from the Long-Term
##   Ecological Research Network" held at NCEAS in September 1996.
```

Writing EML

Because `emld` objects are just nested lists, we can create EML just by writing lists:

```r
me <- list(individualName = list(givenName = "Carl", surName = "Boettiger"))

eml <- list(dataset = list(
              title = "dataset title",
              contact = me,
              creator = me),
              system = "doi",
              packageId = "10.xxx")

as_xml(eml, "ex.xml")
testthat::expect_true(eml_validate("ex.xml") )
```

Note that we don't have to worry about the order of the elements here, `as_xml` will re-order if necessary to validate. (For instance, in valid EML the `creator` becomes listed before `contact`. Of course this is a very low-level interface that does not help the user know what an EML looks like. Creating EML from scratch without knowledge of the schema is a job for the `EML` package and beyond the scope of the lightweight `emld`.

Working with EML as JSON-LD

For many applications, it is useful to merely treat EML as a list object, as seen above, allowing the R user to leverage a standard tools and intuition in working with these files. However, `emld` also opens the door to new possible directions by thinking of EML data in terms of a JSON-LD serialization rather than an XML serialization. First, owing to it's comparative simplicity and native data typing (e.g. of Boolean/string/numeric data), JSON is often easier for many developers to work with than EML's native XML format.

As JSON: Query with JQ

For example, JSON can be queried with with JQ, a simple and powerful query language that also gives us a lot of flexibility over the return structure of our results. JQ syntax is both intuitive and well documented, and often easier than the typical munging of JSON/list data using `purrr`. Here's an example query that turns EML to JSON and then extracts the north and south bounding coordinates:

```r
hf205 <- system.file("extdata/hf205.xml", package="emld")

as_emld(hf205) %>%
  as_json() %>%
  jq('.dataset.coverage.geographicCoverage.boundingCoordinates |
       { northLat: .northBoundingCoordinate,
         southLat: .southBoundingCoordinate }') %>%
  fromJSON()
```

```
## $northLat
## [1] "+42.55"
##
## $southLat
## [1] "+42.42"
```

Nice features of JQ include the ability to do recursive descent (common to XPATH but not possible in `purrr`) and specify the shape and names of the return object (e.g. as a list with elements named `northLat` and `southLat` in this case.)

As semantic data: SPARQL queries

Another side-effect of the JSON-LD representation is that we can treat EML as *semantic* data. This can provide a way to integrate EML records with other data sources, and means we can query the EML using semantic SPARQL queries. One nice thing about SPARQL queries is that, in contrast to XPATH, JQ, or other graph queries, SPARQL always returns a `data.frame` – a particularly convenient and familiar format for R users.

First, we render the EML XML document into JSON-LD file:

```r
f <- system.file("extdata/hf205.xml", package="emld")
as_emld(f) %>%
  as_json("hf205.json")
```

We can now construct a SPARQL query. SPARQL queries look like SQL queries in that we name the columns we want with a `SELECT` command. Unlike SQL, SPARQL allows us to walk the graph by treating these names as variables, indicated by prefixing a `?` to the variable name. We then use a WHERE block to define how these variables relate to each other. In this case, we ask for the genus and species name and bounding box found in the EML file.

```r
sparql <-
  'PREFIX eml: <eml://ecoinformatics.org/eml-2.2.0/>

  SELECT ?genus ?species ?northLat ?southLat ?eastLong ?westLong

  WHERE {
    ?y eml:taxonRankName "genus" .
    ?y eml:taxonRankValue ?genus .
    ?y eml:taxonomicClassification ?s .
    ?s eml:taxonRankName "species" .
    ?s eml:taxonRankValue ?species .
    ?x eml:northBoundingCoordinate ?northLat .
    ?x eml:southBoundingCoordinate ?southLat .
    ?x eml:eastBoundingCoordinate ?eastLong .
    ?x eml:westBoundingCoordinate ?westLong .
  }
'
```

We can now use the `rdflib` library to execute this SPARQL query on the EML document and display the resulting `data.frame`:

```
rdf <- rdf_parse("hf205.json", "jsonld")
df <- rdf_query(rdf, sparql)
df
```

```
## # A tibble: 1 x 6
##   genus      species  northLat southLat eastLong westLong
##   <chr>      <chr>       <dbl>    <dbl>    <dbl>    <dbl>
## 1 Sarracenia purpurea     42.6     42.4    -72.1    -72.3
```

References

Battelle. (2016). National Ecological Observatory Network. Retrieved from http://data.neonscience.org/

Jones, M. B., Schildhauer, M. P., Reichman, O., & Bowers, S. (2006). The New Bioinformatics: Integrating Ecological Data from the Gene to the Biosphere. *Annual Review of Ecology, Evolution, and Systematics*, *37*(1), 519–544. doi:10.1146/annurev.ecolsys.37.091305.110031

Keller, M., Schimel, D. S., Hargrove, W. W., & Hoffman, F. M. (2008). A continental strategy for the National Ecological Observatory Network. *Frontiers in Ecology and the Environment*, *6*(5), 282–284. doi:10.1890/1540-9295(2008)6[282:ACSFTN]2.0.CO;2

Leinfelder, B., Tao, J., Costa, D., Jones, M. B., Servilla, M., O'Brien, M., & Burt, C. (2010). A metadata-driven approach to loading and querying heterogeneous scientific data. *Ecological Informatics*, *5*(1), 3–8. doi:10.1016/j.ecoinf.2009.08.006

Michener, W. K., & Jones, M. B. (2012). Ecoinformatics: supporting ecology as a data-intensive science. *Trends in Ecology & Evolution*, *27*(2), 85–93. doi:10.1016/j.tree.2011.11.016

Michener, W. K., Brunt, J., Helly, J. J., Kirchner, T. B., & Stafford, S. G. (1997). Nongeospatial metadata for the ecological sciences. *Ecological ...*. doi:10.1890/1051-0761%281997%29007%5B0330%3ANMFTES%5D2.0.CO%3B2

Ooms, J. (2017). *jsonld: JSON for Linking Data*. Retrieved from https://CRAN.R-project.org/package=jsonld

Overpeck, J. T., Meehl, G. A., Bony, S., & Easterling, D. R. (2011). Climate Data Challenges in the 21st Century. *Science (New York, N.Y.)*, *331*(6018), 700–702. doi:10.1126/science.1197869

Prud'hommeaux, E., & Seaborne, A. (2008). SPARQL Query Language for RDF. Retrieved from https://www.w3.org/TR/rdf-sparql-query/

Schreiber, G., & Raimond, Y. (2014). RDF 1.1 Primer. Retrieved from https://www.w3.org/TR/rdf11-primer/

Sporny, M., Longley, D., Kellogg, G., Lanthaler, M., & Lindström, N. (2017). JSON-LD 1.0: A JSON-based Serialization for Linked Data. Retrieved from https://www.w3.org/TR/json-ld/