Package ‘emld’

October 13, 2022

Title Ecological Metadata as Linked Data

Version 0.5.1

Description This is a utility for transforming Ecological Metadata Language ('EML') files into 'JSON-LD' and back into 'EML.' Doing so creates a list-based representation of 'EML' in R, so that 'EML' data can easily be manipulated using standard 'R' tools. This makes this package an effective backend for other 'R'-based tools working with 'EML.' By abstracting away the complexity of 'XML' Schema, developers can build around native 'R' list objects and not have to worry about satisfying many of the additional constraints of set by the schema (such as element ordering, which is handled automatically). Additionally, the 'JSON-LD' representation enables the use of developer-friendly 'JSON' parsing and serialization that may facilitate the use of 'EML' in contexts outside of 'R,' as well as the informatics-friendly serializations such as 'RDF' and 'SPARQL' queries.

URL https://docs.ropensci.org/emld/, https://github.com/ropensci/emld

BugReports https://github.com/ropensci/emld/issues

Depends R (>= 3.1.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests spelling, testthat, magrittr, rmarkdown, covr, knitr, rdflib, jqr

Imports xml2, jsonlite, jsonld, methods, yaml

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Carl Boettiger [aut, cre, cph] (<https://orcid.org/0000-0002-1642-628X>), Matthew B. Jones [aut, cph] (<https://orcid.org/0000-0003-0077-4738>), Bryce Mecum [aut, cph] (<https://orcid.org/0000-0002-0381-3766>)

Maintainer Carl Boettiger <cboettig@gmail.com>

Repository CRAN

Date/Publication 2020-09-27 10:50:10 UTC

R topics documented:

`emld-package` *emld: Ecological Metadata as Linked Data*

Description

The goal of emld is to provide a way to work with EML metadata in the JSON-LD format. At it's heart, the package is simply a way to translate an EML XML document into JSON-LD and be able to reverse this so that any semantically equivalent JSON-LD file can be serialized into EML-schema valid XML.

Details

The package has only three core functions:

- `as_emld()` Convert EML's `xml` files (or the `json` version created by this package) into a native R object (an S3 class called `emld`, essentially just a `list`).
- `as_xml()` Convert the native R format, `emld`, back into XML-schema valid EML.
- `as_json()` Convert the native R format, `emld`, into `json`(LD).

Author(s)

Maintainer: Carl Boettiger <cboettig@gmail.com> (ORCID) [copyright holder]

Authors:

- Matthew B. Jones <jones@nceas.ucsb.edu> (ORCID) [copyright holder]
- Bryce Mecum <mecum@nceas.ucsb.edu> (ORCID) [copyright holder]

See Also

Useful links:

- https://docs.ropensci.org/emld/
- https://github.com/ropensci/emld
- Report bugs at https://github.com/ropensci/emld/issues

`as_emld` *Coerce an EML file or object into an emld object.*

Description

Coerce an EML file or object into an emld object.

Usage

```
as_emld(x, from = c("guess", "xml", "json", "list"))
```

Arguments

`x`	path to an EML file
`from`	explicit type for the input format. By default, will attempt to guess the format, but it always safer to specify the input format. This is essential for literal text strings or raw vectors where the type cannot be guessed by the R object class or file extension of the input.

Value

an emld object

Examples

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

as_json *Coerce an emld object into JSON*

Description

Coerce an emld object into JSON

Usage

```
as_json(x, file = NULL)
```

Arguments

x	an emld object
file	optional path to write out to file. Otherwise, defaults to NULL and will return a json object.

Details

Note: since emld list object maintains a 1:1 correspondence with JSON, following the conventions of jsonlite, this function is basically trivial. The only purpose is to default to auto_unbox = TRUE in serializing lists to JSON.

Value

a json object. Or if a file path is provided, the metadata is written out in JSON file and the function returns NULL invisibly.

Examples

```
f <- system.file("extdata/example.xml", package = "emld")
emld <- as_emld(f)
json <- as_json(emld)
## can also write a json file to disk:
json_file <- tempfile()
as_json(emld, json_file)
```

as_xml *Coerce an emld object into XML (EML's standard format)*

Description

Coerce an emld object into XML (EML's standard format)

Usage

```
as_xml(x, file = NULL, root = "eml", ns = "eml", schemaLocation = TRUE)
```

Arguments

`x`	an emld object
`file`	optional path to write out to file. Otherwise, defaults to NULL and will return an xml_document object.
`root`	name for the root node; default to 'eml'
`ns`	namespace abbreviation on root node, default 'eml'
`schemaLocation`	If not explicitly set on `x`, automatically set `xsi:schemaLocation` based upon the root namespace (`TRUE`, default), do not set a `xsi:schemaLocation` (`FALSE`), or set a specific `xsi:schemaLocation` value (`"Your value here..."`). See Examples.

Details

Unlike as_json, this function cannot rely on the existing convention of serializing a list to xml, eg, as defined by xml2::as_xml_document() Instead, this relies on a modified version, as_eml_document. In addition further steps must be taken when working with JSON-LD to deal with different possible framings and namespaces from the JSON-LD context element. Thus this `as_xml` function is particular to EML and `emld` objects alone.

Value

a xml_document object. Or if a file path is provided, the metadata is written out in XML file and the function returns `NULL` invisibly.

Examples

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

## can also write directly to a file:
xml_file <- tempfile()
as_xml(emld, xml_file)

## if you don't want the `xsi:schemaLocation` attribute set
as_xml(emld, schemaLocation = FALSE)

## or if you want to set your own value
as_xml(emld, schemaLocation = "https://eml.ecoinformatics.org/eml-2.2.0
http://example.com/eml-2.2.0/eml.xsd")
```

eml_ns *Get the XML namespace for a version of EML*

Description

Utility function for use when filling in `xmlns`, `schemaLocation`, or `vocab` in various representations of EML. This is a little more future-proof than keeping a dictionary for each version since this won't break on the next release.

Usage

```
eml_ns(version = eml_version())
```

Arguments

`version`	EML version, currently either eml-2.2.0 (current version) or eml-2.1.1. Defaults to current version.

Value

returns the full XML namespace URI for the specified version of the schema

eml_validate *eml_validate*

Description

eml_validate processes an EML document using the XSD schema for the appropriate version of EML and determines if the document is schema-valid as defined by the XSD specification

Usage

```
eml_validate(eml, encoding = "UTF-8", schema = NULL)
```

Arguments

`eml`	file path, xml_document,
`encoding`	optional encoding for files, default UTF-8.
`schema`	path to schema

Value

Whether the document is valid (logical)

Examples

```


f <- system.file("extdata", "example.xml", package = "emld")

## validate file directly from disk:
eml_validate(f)

## validate an eml object:
eml <- as_emld(f)
eml_validate(eml)
```

`eml_version` *Set or check the EML version default*

Description

Set or check the EML version default

Usage

```
eml_version(version = getOption("emld_db", "eml-2.2.0"))
```

Arguments

`version`	EML version, currently either eml-2.2.0 (current version), or eml-2.1.1. The 'eml-' prefix can be omitted.

Value

returns the EML version string. As a side-effect, sets the requested version as the default version by setting the `emld_db` variable in `options()`.

Examples

```
eml_version()
eml_version("2.1.1")
eml_version("eml-2.1.1")
```

`find_real_root_name` *Get the real* `QName` *for the root element, including its prefix*

Description

Note that if a default namespace is used, the prefix will be `d1`.

Usage

```
find_real_root_name(doc)
```

Arguments

`doc`	An `xml_document`

Value

A `list` with elements `prefix` and `name`. `prefix` will be `NULL` if the element has no namespace prefix but `name` will always be a `character`.

`guess_root_schema` *Find the root schema module and version*

Description

Find the root schema module and version

Usage

```
guess_root_schema(doc)
```

Arguments

`doc`	An `xml_document`

Value

If found, a list with names 'version', 'module', and ‘namespace. If not found, throws an error.

guess_schema_location *Guess an appropriate* `schemaLocation` *value for a given version of the schema*

Description

This is a simple helper to make filling in the `schemaLocation` attribute on documents this package creates. Supports EML 2.1.1 and newer.

Usage

```
guess_schema_location(version = eml_version())
```

Arguments

`version`	Optional. Override the version of the schema. Defaults to the current version returned by `eml_version`. See `eml_version` for information on how to change the current version.

Value

Returns a string suitable as a value for `schemaLocation` or `NULL` if a value wasn't found.

Examples

```
## Not run:
# Get an appropriate schemaLocation value for the current version fo EML
guess_schema_location()

# Get an appropriate value for EML 2.1.1
guess_schema_location("eml-2.1.1")

## End(Not run)
```

template *Create a template for an EML object*

Description

Create a template for an EML object

Usage

```
template(object)
```

Arguments

`object`	the name of an eml object to create

Details

Note: while this function can be called in recursions, doing so may be a bad idea.

Value

a list with elements named according to the properties of the object. This can be coerced into EML, see vignettes. NULL-valued elements (~) can take a data entry directly, while empty list()-valued elements () indicate properties that take other eml objects as values.

Examples

```
template("creator")
```

Index