

Package: dataspice (via r-universe)

November 27, 2024

Version 1.1.0

Title Create Lightweight Schema.org Descriptions of Data

Description The goal of 'dataspice' is to make it easier for researchers to create basic, lightweight, and concise metadata files for their datasets. These basic files can then be used to make useful information available during analysis, create a helpful dataset ``README" webpage, and produce more complex metadata formats to aid dataset discovery. Metadata fields are based on the 'Schema.org' and 'Ecological Metadata Language' standards.

License MIT + file LICENSE

URL <https://github.com/ropensci/dataspice>

BugReports <https://github.com/ropensci/dataspice/issues>

Encoding UTF-8

Language en-US

ByteCompile true

RoxygenNote 7.1.1

Imports purrr, EML, fs, jsonlite, whisker, readr, stringr, tools, tibble, shiny, rhandsontable, dplyr, tidyr, ggplot2, magrittr

Suggests testthat, kableExtra, knitr, rmarkdown, servr, listviewer, maps

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Config/pak/sysreqs make libicu-dev libxml2-dev libssl-dev libnode-dev libx11-dev zlib1g-dev

Repository <https://ropensci.r-universe.dev>

RemoteUrl <https://github.com/ropensci/dataspice>

RemoteRef main

RemoteSha f1e459a31fb2657df18839bb9c4ff5a90bc2ad39

Contents

as_jsonld	2
build_site	3
create_spice	4
crosswalk	4
crosswalk_creator	5
crosswalk_datetime	5
crosswalk_distribution	6
crosswalk_Organization	6
crosswalk_Person	6
crosswalk_variables	7
edit_access	8
edit_attributes	8
edit_biblio	9
edit_creators	10
eml_to_spice	10
es_access	11
es_attributes	12
es_biblio	12
es_creators	12
jsonld_to_mustache	13
parse_GeoShape_box	13
parse_spatialCoverage	14
prep_access	14
prep_attributes	15
serve_site	16
spice_to_eml	17
validate_access	18
validate_attributes	18
validate_biblio	19
validate_creators	19
validate_file_paths	20
write_jsonld	20
write_spice	21

Index	23
--------------	-----------

as_jsonld	<i>Convert a list object to JSON-LD</i>
-----------	---

Description

Convert a list object to JSON-LD

Usage

```
as_jsonld(
  x,
  context = "http://schema.org",
  pretty = TRUE,
  auto_unbox = TRUE,
  ...
)
```

Arguments

x	the object to be encoded.
context	JSON-LD context; "http://schema.org".
pretty	Whether or not to prettify output. See toJSON .
auto_unbox	Whether or not to automatically unbox output. See toJSON .
...	Other arguments to be passed to toJSON .

build_site	<i>Build a dataspice site</i>
------------	-------------------------------

Description

Build a dataspice site

Usage

```
build_site(
  path = file.path("data", "metadata", "dataspice.json"),
  template_path = system.file("template.html5", package = "dataspice"),
  out_path = file.path("docs", "index.html")
)
```

Arguments

path	(character) Path to a JSON+LD file with dataspice metadata
template_path	(character) Optional. Path to a template for whisker.render
out_path	(character) Optional. Path to write the site's index.html to. Defaults to docs/index.html.

Value

Nothing. Creates/overwrites docs/index.html

Examples

```
## Not run:
# Create JSON+LD from a set of metadata templates
json <- write_json(biblio, access, attributes, creators)
build_site(json)

## End(Not run)
```

create_spice	<i>Put metadata templates within a metadata subdirectory</i>
--------------	--

Description

Put metadata templates within a metadata subdirectory

Usage

```
create_spice(dir = "data")
```

Arguments

dir	Directory containing data, within which a metadata subdirectory will be created. Defaults to data.
-----	--

Examples

```
## Not run:
create_spice()

# Create templates from the data in a folder other than `data`
create_spice("my_data")

## End(Not run)
```

crosswalk	<i>Crosswalk a term</i>
-----------	-------------------------

Description

Crosswalk a term

Usage

```
crosswalk(doc, term)
```

Arguments

doc (list) A dataspice document as a list
 term (character) The term to crosswalk.

Value

(list) The result of the crosswalk. May be an empty list on failure.

crosswalk_creator *Crosswalk a Schema.org/creator*

Description

Crosswalk a Schema.org/creator

Usage

crosswalk_creator(creator)

Arguments

creator (list) A creator

crosswalk_datetime *Convert a date(time) of unknown format into EML*

Description

A quick and dirty crosswalk of an unknown date(time) input to EML that really only works for ISO8601 input. All other formats will fail and be returned as-is as a calendarDate. From there the user will need to do a conversion themselves.

Usage

crosswalk_datetime(input)

Arguments

input (character) Some unknown date(time) input

Value

(list) A list with members calendarDate and time. time will be NULL if parsing fails or if the time string inside input isn't ISO8601

crosswalk_distribution

Crosswalk a Schema.org/distribution

Description

Crosswalk a Schema.org/distribution

Usage

crosswalk_distribution(distribution)

Arguments

distribution (list) A distribution

crosswalk_Organization

Crosswalk a Schema.org/Organization

Description

Crosswalk a Schema.org/Organization

Usage

crosswalk_Organization(creator)

Arguments

creator (list) A creator

crosswalk_Person

Crosswalk functions for as_eml Crosswalk a Schema.org/Person

Description

Crosswalk functions for as_eml Crosswalk a Schema.org/Person

Usage

crosswalk_Person(creator)

Arguments

creator (list) A creator

crosswalk_variables *Crosswalk dataspice variables to EML*

Description

See [set_attributes](#) for more information on what must be filled out after this is run in order to get a valid EML `attributeList`.

Usage

```
crosswalk_variables(spice)
```

Arguments

spice (list) Your dataspice metadata

Value

(data.frame) A partial EML attributes table

Examples

```
## Not run:
# Load an example dataspice JSON that comes installed with the package
spice <- system.file(
  "examples", "annual-escapement.json",
  package = "dataspice")

# Convert it to EML (notice the warning)
eml_doc <- suppressWarnings({spice_to_eml(spice)})
attributes <- crosswalk_variables(spice)

# Now fill in the attributes data.frame. See `EML::set_attributes`.

# And last, set the attributes on our EML document
eml_doc$dataset$dataTable[[1]]$attributeList <-
  EML::set_attributes(attributes)

## End(Not run)
```

edit_access	<i>Shiny apps for editing dataspice metadata tables</i>
-------------	---

Description

Launch Shiny app for editing individual dataspice metadata tables. Use `edit_*`() where `*` is one of the four dataspice metadata tables `attributes`, `biblio`, `access` **or** `creators`.

Usage

```
edit_access(metadata_dir = file.path("data", "metadata"))
```

Arguments

`metadata_dir` the directory containing the dataspice metadata .csv files. Defaults to `data/metadata/` directory in **current project root**.

Examples

```
## Not run:
edit_attributes()
edit_biblio()
edit_access()
edit_creators()

# Specifying a different dataspice metadata directory
edit_attributes(metadata_dir = "analysis/data/metadata/")

## End(Not run)
```

edit_attributes	<i>Shiny apps for editing dataspice metadata tables</i>
-----------------	---

Description

Launch Shiny app for editing individual dataspice metadata tables. Use `edit_*`() where `*` is one of the four dataspice metadata tables `attributes`, `biblio`, `access` **or** `creators`.

Usage

```
edit_attributes(metadata_dir = file.path("data", "metadata"))
```

Arguments

`metadata_dir` the directory containing the dataspice metadata .csv files. Defaults to `data/metadata/` directory in **current project root**.

Examples

```
## Not run:
edit_attributes()
edit_biblio()
edit_access()
edit_creators()

# Specifying a different dataspice metadata directory
edit_attributes(metadata_dir = "analysis/data/metadata/")

## End(Not run)
```

edit_biblio

Shiny apps for editing dataspice metadata tables

Description

Launch Shiny app for editing individual dataspice metadata tables. Use `edit_*`() where `*` is one of the four dataspice metadata tables `attributes`, `biblio`, `access` or `creators`.

Usage

```
edit_biblio(metadata_dir = file.path("data", "metadata"))
```

Arguments

`metadata_dir` the directory containing the dataspice metadata .csv files. Defaults to `data/metadata/` directory in **current project root**.

Examples

```
## Not run:
edit_attributes()
edit_biblio()
edit_access()
edit_creators()

# Specifying a different dataspice metadata directory
edit_attributes(metadata_dir = "analysis/data/metadata/")

## End(Not run)
```

edit_creators	<i>Shiny apps for editing dataspice metadata tables</i>
---------------	---

Description

Launch Shiny app for editing individual dataspice metadata tables. Use `edit_*`() where `*` is one of the four dataspice metadata tables `attributes`, `biblio`, `access` or `creators`.

Usage

```
edit_creators(metadata_dir = file.path("data", "metadata"))
```

Arguments

`metadata_dir` the directory containing the dataspice metadata .csv files. Defaults to `data/metadata/` directory in **current project root**.

Examples

```
## Not run:
edit_attributes()
edit_biblio()
edit_access()
edit_creators()

# Specifying a different dataspice metadata directory
edit_attributes(metadata_dir = "analysis/data/metadata/")

## End(Not run)
```

eml_to_spice	<i>Create dataspice metadata tables from EML</i>
--------------	--

Description

Create dataspice metadata tables from EML

Usage

```
eml_to_spice(eml, path = NULL)
```

Arguments

`eml` (emld) An EML object
`path` (character) Folder path for saving the tables to disk

Value

A list with names attributes, access, biblio, and creators. Optionally, if path is specified, saves the four tables as CSV files.

Examples

```
## Not run:
# First, load up an example EML record
library(EML)

eml_path <- system.file(
  file.path("example-dataset", "broodTable_metadata.xml"),
  package = "dataspice")
eml <- read_eml(eml_path)

# Generate the four dataspice tables
my_spice <- eml_to_spice(eml)

# Or save them as a file
# Generate the four dataspice tables
eml_to_spice(eml, ".")

## End(Not run)
```

es_access

Get access from EML

Description

Return EML access in the dataspice access.csv format.

Usage

```
es_access(eml, path = NULL)
```

Arguments

eml (emld) an EML object
path (character) folder path for saving the table to disk

es_attributes	<i>Get attributes from EML</i>
---------------	--------------------------------

Description

Return EML attributes in the dataspice attributes.csv format.

Usage

```
es_attributes(eml, path = NULL)
```

Arguments

eml	(emld) an EML object
path	(character) folder path for saving the table to disk

es_biblio	<i>Get biblio from EML</i>
-----------	----------------------------

Description

Return EML biblio in the dataspice biblio.csv format.

Usage

```
es_biblio(eml, path = NULL)
```

Arguments

eml	(emld) an EML object
path	(character) folder path for saving the table to disk

es_creators	<i>Get creators from EML</i>
-------------	------------------------------

Description

Return EML creators in the dataspice creators.csv format.

Usage

```
es_creators(eml, path = NULL)
```

Arguments

eml	(emld) an EML object
path	(character) folder path for saving the table to disk

jsonld_to_mustache *Convert JSONLD to a list suitable for Mustache templating*

Description

Convert JSONLD to a list suitable for Mustache templating

Usage

jsonld_to_mustache(path)

Arguments

path (character) Path to file on disk to convert

Value

(list) Mustache-appropriate list

parse_GeoShape_box *Parse spatialCoverage\$geo\$box section for use in a Leaflet map*

Description

Parse spatialCoverage\$geo\$box section for use in a Leaflet map

Usage

parse_GeoShape_box(box)

Arguments

box (list) spatialCoverage\$geo\$box section of the JSONLD

Value

(list) Template-specific variables for Leaflet

parse_spatialCoverage *Parse spatialCoverage section for use in a Leaflet map*

Description

Parse spatialCoverage section for use in a Leaflet map

Usage

```
parse_spatialCoverage(spatialCoverage)
```

Arguments

spatialCoverage
(list) spatialCoverage section of the JSONLD

Value

(list) Template-specific variables for Leaflet

prep_access *Prepare access*

Description

Extract fileNames from data file(s) and add them to access.csv. The helper [validate_file_paths](#) can be used to create vectors of valid file paths that can be checked and then passed as data_path argument to [prep_access](#).

Usage

```
prep_access(  
  data_path = "data",  
  access_path = file.path("data", "metadata", "access.csv"),  
  ...  
)
```

Arguments

data_path character vector of either:

1. path(s) to the data file(s).
2. single path to directory containing data file(s). Currently only tabular .csv and .tsv or .rds files are supported.

access_path path to the access.csv file. Defaults to data/metadata/access.csv.

... parameters passed to list.files(). For example, use recursive = TRUE to list files in a folder recursively or use pattern to filter files for patterns.

Value

Updates access.csv and writes to access_path.

Examples

```
## Not run:
# First create the metadata tempaltes
create_spice()

# Then begin filling them in from your data files
prep_access()

## End(Not run)
```

prep_attributes	<i>Prepare attributes</i>
-----------------	---------------------------

Description

Extract variableNames from data file(s) and add them to attributes.csv. The helper [validate_file_paths](#) can be used to create vectors of valid file paths that can be checked and then passed as data_path argument to [prep_attributes](#).

Usage

```
prep_attributes(
  data_path = "data",
  attributes_path = file.path("data", "metadata", "attributes.csv"),
  ...
)
```

Arguments

data_path	character vector of either: <ol style="list-style-type: none"> 1. path(s) to the data file(s). 2. single path to directory containing data file(s). Currently only tabular .csv and .tsv files are supported. Alternatively attributes returned using names() can be extracted from r object, stored as .rds files.
attributes_path	path to the attributes.csv`` file. Defaults to data/metadata/attributes.csv`.
...	parameters passed to list.files(). For example, use recursive = TRUE to list files in a folder recursively or use pattern to filter files for patterns.

Value

prep_attributes() updates the attributes.csv and writes to attributes_path.

Examples

```
## Not run:
create_spice()
# extract attributes from all `csv`, `tsv`, `rds` files in the data folder
# (non recursive)
prep_attributes()
# recursive
prep_attributes(recursive = TRUE)
# extract attributes from a single file using file path
data_path <- system.file("example-dataset", "BroodTables.csv",
  package = "dataspice")
prep_attributes(data_path)
# extract attributes from a single file by file path pattern matching
data_path <- system.file("example-dataset", package = "dataspice")
prep_attributes(data_path, pattern = "StockInfo")
# extract from a folder using folder path
data_path <- system.file("example-dataset", package = "dataspice")
prep_attributes(data_path)

## End(Not run)
```

 serve_site

Serve site

Description

Serve site

Usage

```
serve_site(path = "docs")
```

Arguments

path (character) Optional. Directory to serve. Defaults to docs.

Value

Nothing.

Examples

```
## Not run:
# Build your site
json <- write_json(biblio, access, attributes, creators)
build_site(json)

# Serve it
serve_site()

## End(Not run)
```

spice_to_eml	<i>Convert dataspice metadata to EML</i>
--------------	--

Description

Performs an (imperfect) conversion of dataspice metadata to EML. It's very likely you will get validation errors and need to fix them afterwards but `spice_to_eml` is a good way to a richer metadata schema (EML) when you're already using dataspice but need a richer metadata schema.

Usage

```
spice_to_eml(spice = file.path("data", "metadata", "dataspice.json"))
```

Arguments

`spice` (list) Your dataspice metadata. Uses `data/metadata/dataspice.json` by default.

Value

(emld) The crosswalked emld object

Examples

```
# Load an example dataspice JSON that comes installed with the package
spice <- system.file(
  "examples", "annual-escapement.json",
  package = "dataspice"
)

# And crosswalk it to EML
spice_to_eml(spice)

# We can also create dataspice metadata from scratch and crosswalk it to EML
myspice <- list(
  name = "My example spice",
  creator = "Me",
  contact = "Me"
)
spice_to_eml(myspice)
```

validate_access	<i>Validate access.csv</i>
-----------------	----------------------------

Description

Validate access.csv

Usage

```
validate_access(access)
```

Arguments

access (data.frame) A data.frame read in from access.csv

Value

Nothing. Side-effect: Can stop execution if validation fails.

validate_attributes	<i>Validate attributes.csv</i>
---------------------	--------------------------------

Description

Validate attributes.csv

Usage

```
validate_attributes(attributes)
```

Arguments

attributes (data.frame) A data.frame read in from attributes.csv

Value

Nothing. Side-effect: Can stop execution if validation fails.

validate_biblio	<i>Validate biblio.csv</i>
-----------------	----------------------------

Description

Validate biblio.csv

Usage

```
validate_biblio(biblio)
```

Arguments

biblio (data.frame) A data.frame read in from biblio.csv

Value

Nothing. Side-effect: Can stop execution if validation fails.

validate_creators	<i>Validate creators.csv</i>
-------------------	------------------------------

Description

Validate creators.csv

Usage

```
validate_creators(creators)
```

Arguments

creators (data.frame) A data.frame read in from creators.csv

Value

Nothing. Side-effect: Can stop execution if validation fails.

validate_file_paths *Validate file paths*

Description

Helper function to return a set of file paths for use in other functions

Usage

```
validate_file_paths(data_path = "data", ...)
```

Arguments

data_path	character vector of either: <ol style="list-style-type: none">1. path(s) to the data file(s).2. single path to directory containing data file(s). Currently only tabular .csv and .tsv files are supported. Alternatively attributes returned using names() can be extracted from r object, stored as .rds files.
...	parameters passed to list.files(). For example, use recursive = TRUE to list files in a folder recursively or use pattern to filter files for patterns.

Value

One or more data file paths

Examples

```
## Not run:  
# Assuming some data files in "./data"  
my_files <- validate_file_paths()  
  
# If your data files are in `another_folder`  
my_files <- validate_file_paths("another_folder")  
  
## End(Not run)
```

write_jsonld *Write a list out as object to JSON-LD*

Description

Write a list out as object to JSON-LD

Usage

```
write_jsonld(
  x,
  path,
  context = "http://schema.org",
  pretty = TRUE,
  auto_unbox = TRUE,
  ...
)
```

Arguments

x	an object to be serialized to JSON
path	file on disk
context	JSON-LD context; "http://schema.org"
pretty	adds indentation whitespace to JSON output. Can be TRUE/FALSE or a number specifying the number of spaces to indent. See prettify
auto_unbox	automatically unbox all atomic vectors of length 1. It is usually safer to avoid this and instead use the unbox function to unbox individual elements. An exception is that objects of class <code>AsIs</code> (i.e. wrapped in <code>I()</code>) are not automatically unboxed. This is a way to mark single values as length-1 arrays.
...	additional conversion arguments, see also toJSON or fromJSON

write_spice

Write spice

Description

Write out your metadata as a dataspice JSON-LD document

Usage

```
write_spice(path = file.path("data", "metadata"), ...)
```

Arguments

path	location of metadata files
...	additional arguments to jsonlite::toJSON()

Value

A JSON-LD file at the path specified

Examples

```
## Not run:  
# First create your metadata templates  
create_spice()  
  
# Then fill in the template files however you like  
  
# Then write out your dataspice file  
write_spice()  
  
## End(Not run)
```

Index

as_jsonld, [2](#)

build_site, [3](#)

create_spice, [4](#)

crosswalk, [4](#)

crosswalk_creator, [5](#)

crosswalk_datetime, [5](#)

crosswalk_distribution, [6](#)

crosswalk_Organization, [6](#)

crosswalk_Person, [6](#)

crosswalk_variables, [7](#)

edit_access, [8](#)

edit_attributes, [8](#)

edit_biblio, [9](#)

edit_creators, [10](#)

eml_to_spice, [10](#)

es_access, [11](#)

es_attributes, [12](#)

es_biblio, [12](#)

es_creators, [12](#)

fromJSON, [21](#)

jsonld_to_mustache, [13](#)

jsonlite::toJSON(), [21](#)

parse_GeoShape_box, [13](#)

parse_spatialCoverage, [14](#)

prep_access, [14](#), [14](#)

prep_attributes, [15](#), [15](#)

prettify, [21](#)

serve_site, [16](#)

set_attributes, [7](#)

spice_to_eml, [17](#)

toJSON, [3](#), [21](#)

unbox, [21](#)

validate_access, [18](#)

validate_attributes, [18](#)

validate_biblio, [19](#)

validate_creators, [19](#)

validate_file_paths, [14](#), [15](#), [20](#)

whisker.render, [3](#)

write_jsonld, [20](#)

write_spice, [21](#)