Package 'codemetar'

March 16, 2022

Type Package

Title Generate 'CodeMeta' Metadata for R Packages

Version 0.3.4

Description The 'Codemeta' Project defines a 'JSON-LD' format for describing software metadata, as detailed at <https://codemeta.github.io>. This package provides utilities to generate, parse, and modify 'codemeta.json' files automatically for R packages, as well as tools and examples for working with 'codemeta.json' 'JSON-LD' more generally.

License GPL-3

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

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

Depends R (>= 3.2.0)

Imports commonmark, crul, desc, gert, gh, jsonlite (>= 1.6), magrittr, memoise, methods, pingr, purrr, remotes, sessioninfo, stats, urltools, xml2, cli, codemeta

Suggests withr, covr, details, dplyr (>= 0.7.0), jsonld, jsonvalidate, knitr, printr, rmarkdown, testthat (>= 3.0.0), usethis

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.1.2

X-schema.org-isPartOf https://ropensci.org

X-schema.org-keywords metadata, codemeta, ropensci, citation, credit, linked-data

Config/testthat/edition 3

NeedsCompilation no

Author Carl Boettiger [aut, cre, cph] (<https://orcid.org/0000-0002-1642-628X>), Anna Krystalli [rev, ctb] (<https://orcid.org/0000-0002-2378-4915>),

Toph Allen [rev] (<https://orcid.org/0000-0003-4580-091X>),
Maëlle Salmon [ctb, aut] (<https://orcid.org/0000-0002-2815-0399>),
rOpenSci [fnd] (https://ropensci.org/),
Katrin Leinweber [ctb] (<https://orcid.org/0000-0001-5135-5758>),
Noam Ross [ctb] (<https://orcid.org/0000-0002-2136-0000>),
Arfon Smith [ctb],
Jeroen Ooms [ctb] (<https://orcid.org/0000-0002-4035-0289>),
Sebastian Meyer [ctb] (<https://orcid.org/0000-0002-1791-9449>),
Michael Rustler [ctb] (<https://orcid.org/0000-0003-0647-7726>),
Hauke Sonnenberg [ctb] (<https://orcid.org/0000-0001-9134-2871>),
Sebastian Kreutzer [ctb] (<https://orcid.org/0000-0002-0734-2199>),
Thierry Onkelinx [ctb] (<https://orcid.org/0000-0001-8804-4216>)

Maintainer Carl Boettiger <cboettig@gmail.com>

Repository CRAN

Date/Publication 2022-03-16 14:40:08 UTC

R topics documented:

`codemetar-package` *codemetar: generate codemeta metadata for R packages*

Description

The 'Codemeta' Project defines a 'JSON-LD' format for describing software metadata, as detailed at <https://codemeta.github.io>. This package provides utilities to generate, parse, and modify 'codemeta.json' files automatically for R packages, as well as tools and examples for working with 'codemeta.json' 'JSON-LD' more generally.

Details

Why bother creating a codemeta.json for your package? R packages encode lots of metadata in the `DESCRIPTION` file, `README`, and other places, telling users and developers about the package purpose, authors, license, dependencies, and other information that facilitates discovery, adoption, and credit for your software. Unfortunately, because each software language records this metadata in a different format, that information is hard for search engines, software repositories, and other developers to find and integrate.

By generating a `codemeta.json` file, you turn your metadata into a format that can easily crosswalk between metadata in many other software languages. CodeMeta is built on schema.org a

simple structured data format developed by major search engines like Google and Bing to improve discoverability in search. CodeMeta is also understood by significant software archiving efforts such as Software Heritage Project, which seeks to permanently archive all open source software.

For more general information about the CodeMeta Project for defining software metadata, see https://codemeta.github.io. In particular, new users might want to start with the User Guide, while those looking to learn more about JSON-LD and consuming existing codemeta files should see the Developer Guide.

Why codemetar? The 'Codemeta' Project defines a 'JSON-LD' format for describing software metadata, as detailed at https://codemeta.github.io. This package provides utilities to **generate, parse, and modify codemeta.jsonld files automatically for R packages**, as well as tools and examples for **working with codemeta json-ld more generally**.

It has three main goals:

- Quickly **generate a valid codemeta.json file from any valid R package**. To do so, we automatically extract as much metadata as possible using the DESCRIPTION file, as well as extracting metadata from other common best-practices such as the presence of Travis and other badges in README, etc.
- Facilitate the addition of further metadata fields into a codemeta.json file, as well as general manipulation of codemeta files.
- Support the ability to crosswalk between terms used in other metadata standards, as identified by the Codemeta Project Community, see https://codemeta.github.io/crosswalk/

Author(s)

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

Authors:

- Maëlle Salmon (ORCID) [contributor]

Other contributors:

- Anna Krystalli (ORCID) [reviewer, contributor]
- Toph Allen (ORCID) [reviewer]
- rOpenSci (https://ropensci.org/) [funder]
- Katrin Leinweber (ORCID) [contributor]
- Noam Ross (ORCID) [contributor]
- Arfon Smith [contributor]
- Jeroen Ooms (ORCID) [contributor]
- Sebastian Meyer (ORCID) [contributor]
- Michael Rustler (ORCID) [contributor]
- Hauke Sonnenberg (ORCID) [contributor]
- Sebastian Kreutzer (ORCID) [contributor]
- Thierry Onkelinx (ORCID) [contributor]

See Also

Useful links:

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

---

`create_codemeta` *create_codemeta*

Description

create a codemeta list object in R for further manipulation. Similar to `write_codemeta()`, but returns an R list object rather than writing directly to a file. See examples.

Usage

```
create_codemeta(
  pkg = ".",
  root = ".",
  id = NULL,
  use_filesize = FALSE,
  force_update = getOption("codemeta_force_update", TRUE),
  verbose = TRUE,
  ...
)
```

Arguments

`pkg`	package path to package root, or description file (character), or a codemeta object (list)
`root`	if pkg is a codemeta object, optionally give the path to package root. Default guess is current dir.
`id`	identifier for the package, e.g. a DOI (or other resolvable URL)
`use_filesize`	whether to try to estimating and adding a filesize by using `base::file.size()`. Files in `.Rbuildignore` are ignored.
`force_update`	Update guessed fields even if they are defined in an existing codemeta.json file
`verbose`	Whether to print messages indicating opinions e.g. when DESCRIPTION has no URL. – See `give_opinions`; and indicating the progress of internet downloads.
`...`	additional arguments to `write_json`

Value

a codemeta list object

Examples

```
path <- system.file("", package="codemeta")
cm <- create_codemeta(path)
cm$keywords <- list("metadata", "ropensci")
```

`extract_badges` *Extract all badges from Markdown file*

Description

Extract all badges from Markdown file

Usage

```
extract_badges(path)
```

Arguments

`path`	Path to Markdown file

Value

A data.frame with for each badge its text, link and link to its image.

Examples

```
## Not run:
extract_badges(system.file("examples/README_fakepackage.md", package="codemetar"))

## End(Not run)
```

`give_opinions` *Function giving opinions about a package*

Description

Function giving opinions about a package

Usage

```
give_opinions(pkg_path = getwd(), verbose = FALSE)
```

Arguments

pkg_path	Path to the package root
verbose	Whether to print message related to internet download progress.

Value

A data.frame of opinions

`write_codemeta` *write_codemeta*

Description

write out a codemeta.json file for a given package. This function is basically a wrapper around create_codemeta() to both create the codemeta object and write it out to a JSON-LD-formatted file in one command. It can also be used simply to write out to JSON-LD any existing object created with `create_codemeta()`.

Usage

```
write_codemeta(
  pkg = ".",
  path = "codemeta.json",
  root = ".",
  id = NULL,
  use_filesize = TRUE,
  force_update = getOption("codemeta_force_update", TRUE),
  use_git_hook = NULL,
  verbose = TRUE,
  write_minimeta = FALSE,
  ...
)
```

Arguments

pkg	package path to package root, or description file (character), or a codemeta object (list)
path	file name of the output, leave at default "codemeta.json"
root	if pkg is a codemeta object, optionally give the path to package root. Default guess is current dir.
id	identifier for the package, e.g. a DOI (or other resolvable URL)
use_filesize	whether to try to estimating and adding a filesize by using `base::file.size()`. Files in `.Rbuildignore` are ignored.
force_update	Update guessed fields even if they are defined in an existing codemeta.json file
use_git_hook	Deprecated argument.

verbose	Whether to print messages indicating opinions e.g. when DESCRIPTION has no URL. – See give_opinions; and indicating the progress of internet downloads.
write_minimeta	whether to also create the file schemaorg.json that corresponds to the metadata Google would validate, to be inserted to a webpage for SEO. It is saved as "inst/schemaorg.json" alongside path (by default, "codemeta.json").
...	additional arguments to write_json

Details

Why bother creating a codemeta.json for your package? R packages encode lots of metadata in the DESCRIPTION file, README, and other places, telling users and developers about the package purpose, authors, license, dependencies, and other information that facilitates discovery, adoption, and credit for your software. Unfortunately, because each software language records this metadata in a different format, that information is hard for search engines, software repositories, and other developers to find and integrate.

By generating a codemeta.json file, you turn your metadata into a format that can easily crosswalk between metadata in many other software languages. CodeMeta is built on schema.org a simple structured data format developed by major search engines like Google and Bing to improve discoverability in search. CodeMeta is also understood by significant software archiving efforts such as Software Heritage Project, which seeks to permanently archive all open source software.

For more general information about the CodeMeta Project for defining software metadata, see https://codemeta.github.io. In particular, new users might want to start with the User Guide, while those looking to learn more about JSON-LD and consuming existing codemeta files should see the Developer Guide.

How to keep codemeta.json up-to-date? In particular, how to keep it up to date with DESCRIPTION? codemetar itself no longer supports automatic sync, but there are quite a few methods available out there. Choose one that fits well into your workflow!

- You could rely on devtools::release() since it will ask you whether you updated codemeta.json when such a file exists.
- You could use a git pre-commit hook that prevents a commit from being done if DESCRIPTION is newer than codemeta.json.
  - You can use the precommit package in which there's a "codemeta-description-updated" hook.
  - If that's your only pre-commit hook (i.e. you don't have one created by e.g. usethis::use_readme_rmd()), then you can create it using

```
script = readLines(system.file("templates", "description-codemetajson-pre-commit.sh", package = "codem
usethis::use_git_hook("pre-commit",
                      script = script)
```

- You could use GitHub actions. Refer to GitHub actions docs https://github.com/features/actions, and to the example workflow provided in this package (type system.file("templates","codemeta-github = "codemetar")). You can use the cm-skip keyword in your commit message if you don't want this to run on a specific commit. The example workflow provided is setup to only run when a push is made to the master branch. This setup is designed for if you're using a git flow setup where the master branch is only committed and pushed to via pull requests. After each

PR merge (and the completion of this GitHub action), your master branch will always be up to date and so long as you don't make manual changes to the codemeta.json file, you won't have merge conflicts.

Alternatively, you can have GitHub actions route run `codemetar` on each commit. If you do this you should try to remember to run git pull before making any new changes on your local project. However, if you forgot to pull and already committed new changes, fret not, you can use (git pull –rebase) to rewind you local changes on top of the current upstream `HEAD`.

```
on:
  push:
    branches: master
    paths:
      - DESCRIPTION
      - .github/workflows/main.yml

name: Render codemeta
jobs:
  render:
    name: Render codemeta
    runs-on: macOS-latest
    if: "!contains(github.event.head_commit.message, 'cm-skip')"
    steps:
      - uses: actions/checkout@v1
      - uses: r-lib/actions/setup-r@v1
      - name: Install codemetar
        run: Rscript -e 'install.packages("codemetar")'
      - name: Render codemeta
        run: Rscript -e 'codemetar::write_codemeta()'
      - name: Commit results
        run: |
       git commit codemeta.json -m 'Re-build codemeta.json' || echo "No changes to commit"
       git push https://${{github.actor}}:${{secrets.GITHUB_TOKEN}}@github.com/${{github.repository}
```

Value

writes out the codemeta.json file, and schemaorg.json if `write_codemeta` is `TRUE`.

Technical details

If pkg is a codemeta object, the function will attempt to update any fields it can guess (i.e. from the DESCRIPTION file), overwriting any existing data in that block. In this case, the package root directory should be the current working directory.

When creating and writing a codemeta.json for the first time, the function adds "codemeta.json" to .Rbuildignore.

Examples

```
## Not run:
```

```
# from anywhere in the package source directory
write_codemeta()

## End(Not run)
```

Index