Package ‘cffr’

March 13, 2026

Title Generate Citation File Format ('cff') Metadata for R Packages

Version 1.3.0

Description The Citation File Format version 1.2.0
<doi:10.5281/zenodo.5171937> is a human and machine readable file
format which provides citation metadata for software. This package
provides core utilities to generate and validate this metadata.

License GPL (>= 3)

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

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

Depends R (>= 4.1.0)

Imports cli (>= 2.0.0), desc (>= 1.3.0), jsonlite (>= 1.7.2),
jsonvalidate (>= 1.1.0), tools, utils, yaml (>= 2.2.1)

Suggests bibtex (>= 0.5.0), knitr, lifecycle, quarto, rmarkdown,
testthat (>= 3.0.0), usethis

VignetteBuilder knitr, quarto

Config/Needs/website devtools

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

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

X-schema.org-keywords attribution, citation, credit, citation-files,
cff, metadata, citation-file-format, cran, r, r-package,
ropensci, rstats, r-cran

NeedsCompilation no

Author Diego Hernangómez [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-8457-4658>),
João Martins [rev] (ORCID: <https://orcid.org/0000-0001-7961-4280>),
Scott Chamberlain [rev] (ORCID:
<https://orcid.org/0000-0003-1444-9135>)

Maintainer Diego Hernangómez <diego.hernangomezherrero@gmail.com>

Repository CRAN

Date/Publication 2026-03-13 08:10:02 UTC

Contents

---

as_bibentry *Create* bibentry *objects from several sources*

Description

This function creates bibentry objects from different metadata sources (cff objects, DESCRIPTION files, etc.). The inverse transformation (bibentry object to cff_ref_lst) can be done with the corresponding as_cff.bibentry() method.

With toBibtex() it is possible to convert cff objects to BibTeX markup on the fly, see **Examples**.

Usage

```
as_bibentry(x, ...)

## Default S3 method:
as_bibentry(x, ...)

## S3 method for class 'character'
as_bibentry(x, ..., what = c("preferred", "references", "all"))

## S3 method for class '`NULL`'
as_bibentry(x, ...)
```

```
## S3 method for class 'list'
as_bibentry(x, ...)

## S3 method for class 'cff'
as_bibentry(x, ..., what = c("preferred", "references", "all"))

## S3 method for class 'cff_ref_lst'
as_bibentry(x, ...)

## S3 method for class 'cff_ref'
as_bibentry(x, ...)
```

Arguments

`x`	The source used to generate the `bibentry` object via **cffr**. It can be: • A missing value, which retrieves the `DESCRIPTION` file from your in-development package. • An existing `cff` object created with `cff()`, `cff_create()`, or `as_cff()`. • Path to a CITATION.cff file (`"CITATION.cff"`). • The name of an installed package (`"jsonlite"`). • Path to a DESCRIPTION file (`"DESCRIPTION"`).
`...`	Additional arguments to be passed to or from methods.
`what`	Fields to extract from a full `cff` object. The value could be: • `preferred`: This would create a single entry with the main citation info of the package (key `preferred-citation`). • `references`: Extract all the entries of `references` key. • `all`: A combination of the previous two options. This would extract both the `preferred-citation` and the `references` key. See `vignette("crosswalk", package = "cffr")`.

Details

A **R** `bibentry` object is the representation of a BibTeX entry. These objects can be converted to BibTeX markup with `toBibtex()`, that creates an object of class `Bibtex` and can be printed and exported as a valid BibTeX entry.

`as_bibtex()` tries to map the information of the source `x` into a `cff` object and performs a mapping of the metadata to BibTeX, according to `vignette("bibtex_cff", "cffr")`.

Value

`as_bibentry()` returns a `bibentry` object with one or more entries.

References

- Patashnik, Oren. "BIBTEXTING" February 1988. https://osl.ugr.es/CTAN/biblio/bibtex/base/btxdoc.pdf.
- Haines, R., & The Ruby Citation File Format Developers. (2021). *Ruby CFF Library (Version 0.9.0)* (Computer software). doi:10.5281/zenodo.1184077.
- Hernangomez D (2022). "BibTeX and CFF, a potential crosswalk." *The cffr package, Vignettes*. doi:10.21105/joss.03900, https://docs.ropensci.org/cffr/articles/bibtex-cff.html.

See Also

`utils::bibentry()` to understand more about the `bibentry` class.

- `vignette("crosswalk", package = "cffr")` provides details on how the metadata of a package is mapped to produce a `cff` object.
- `vignette("bibtex_cff", "cffr")` provides detailed information about the internal mapping performed between `cff` objects and BibTeX markup (both `cff` to BibTeX and BibTeX to `cff`).

Other related functions:

- `utils::toBibtex()`.

Other functions for working with BibTeX format: `cff_read()`, `cff_read_bib_text()`, `cff_write_bib()`, `encoded_utf_to_latex()`

Coercing between **R** classes with **S3 Methods**: `as_cff()`, `as_cff_person()`, `cff_class`

Examples

```
# From a cff object ----
cff_object <- cff()

cff_object

# bibentry object
bib <- as_bibentry(cff_object)

class(bib)

bib

# Print as bibtex
toBibtex(bib)

# Thanks to the S3 Method we can also do
toBibtex(cff_object)

# Other sources ----
# From a CITATION.cff
```

```
path <- system.file("examples/CITATION_complete.cff", package = "cffr")
cff_file <- as_bibentry(path)

cff_file

# For an installed package with options
installed_package <- as_bibentry("jsonvalidate", what = "all")

installed_package

# Use a DESCRIPTION file
path2 <- system.file("examples/DESCRIPTION_gitlab", package = "cffr")
desc_file <- as_bibentry(path2)

toBibtex(desc_file)
```

`as_cff` *Coerce lists,* `person` *and* `bibentry` *objects to* `cff`

Description

`as_cff()` turns an existing list-like **R** object into a so-called `cff`, a list with class `cff`, with the corresponding sub-class if applicable.

`as_cff` is an S3 generic, with methods for:

- `person` objects as produced by `utils::person()`.
- `bibentry` objects as produced by `utils::bibentry()`.
- `Bibtex` objects as produced by `toBibtex()`.
- Default: Other inputs are first coerced with `as.list()`.

Usage

```
as_cff(x, ...)

## Default S3 method:
as_cff(x, ...)

## S3 method for class 'list'
as_cff(x, ...)

## S3 method for class 'person'
as_cff(x, ...)

## S3 method for class 'bibentry'
as_cff(x, ...)
```

```
## S3 method for class 'Bibtex'
as_cff(x, ...)
```

Arguments

`x`	A `person`, `bibentry` or other object that could be coerced to a list.
`...`	Additional arguments to be passed on to other methods.

Details

For `as_cff.bibentry()` / `as_cff.Bibtex()` see `vignette("bibtex_cff", "cffr")` to understand how the mapping is performed.

`as_cff_person()` is preferred over `as_cff.person()`, since it can handle `character` person such as `"Davis, Jr., Sammy"`. For `person` objects both functions are similar.

Value

- `as_cff.person()` returns an object with classes `cff_pers_lst, cff`.
- `as_cff.bibentry()` and `as_cff.Bibtex()` return an object with classes `cff_ref_lst, cff`.
- The remaining methods return an object of class `cff`. However, if `x` has a structure compatible with `definitions.person`, `definitions.entity` or `definitions.reference`, the object has the corresponding subclass.

Learn more about the **cffr** class system in cff_class.

See Also

- `cff()`: Create a full `cff` object from scratch.
- `cff_modify()`: Modify a `cff` object.
- `cff_create()`: Create a `cff` object of a **R** package.
- `cff_read()`: Create a `cff` object from a external file.
- `as_cff_person()`: Recommended way for creating persons in CFF format.

Learn more about the **cffr** class system in cff_class.

Coercing between **R** classes with **S3 Methods**: `as_bibentry()`, `as_cff_person()`, `cff_class`

Examples

```
# Convert a list to "cff" object
cffobj <- as_cff(list(
  "cff-version" = "1.2.0",
  title = "Manipulating files"
))

class(cffobj)

# Nice display thanks to yaml package
cffobj
```

```
# bibentry method
a_cit <- citation("cffr")[[1]]

a_cit

as_cff(a_cit)

# Bibtex method
a_bib <- toBibtex(a_cit)

a_bib

as_cff(a_cit)
```

as_cff_person *Coerce **R** objects to `cff_pers_lst` objects (`cff` persons)*

Description

`as_cff_person()` turns an existing list-like **R** object into a `cff_pers_lst` object representing a list of `definitions.person` or `definitions.entity`, as defined by the Citation File Format schema.

`as_cff_person` is an S3 generic, with methods for:

- `person`: Objects created with `person()`.
- `character`: Strings with the definition of an author or multiple authors, using the standard BibTeX notation (see Markey, 2007) and others, such as the output of `format()` for person (see `format.person()`).
- Default: Other inputs are first coerced with `as.character()`.

The inverse transformation (`cff_pers_lst` to `person`) can be done using the methods `as.person.cff_pers()` and `as.person.cff_pers_lst()`.

Usage

```
as_cff_person(x, ...)

## Default S3 method:
as_cff_person(x, ...)

## S3 method for class 'person'
as_cff_person(x, ...)

## S3 method for class 'character'
as_cff_person(x, ...)
```

Arguments

`x`	Any **R** object.
`...`	Ignored by this method.

Details

`as_cff_person()` recognizes whether the input should be converted using the CFF reference for `definitions.person` or `definitions.entity`.

`as_cff_person()` uses a custom algorithm that breaks a name as explained in Section 11 of "Tame the BeaST" (Markey, 2007) (see also Decoret, 2007):

- `First von Last`.
- `von Last, First`.
- `von Last, Jr, First`.

Mapping is performed as follows:

- `First` is mapped to the CFF field `given-names`.
- `von` is mapped to the CFF field `name-particle`.
- `Last` is mapped to the CFF field `family-names`.
- `Jr` is mapped to the CFF field `name-suffix`.

For entities, the entire `character` is mapped to `name`. It is recommended to "protect" entity names with `{}`:

```
# Don't do
entity <- "Elephant and Castle"
as_cff_person(entity)
- name: Elephant
- name: Castle

# Do
entity_protect <- "{Elephant and Castle}"
as_cff_person(entity_protect)
- name: Elephant and Castle
```

`as_cff_person()` attempts to extract as much information as possible. For `character` strings from `format.person()`, the email and ORCID are also extracted.

Value

`as_cff_person()` returns an object of classes `cff_pers_lst, cff` according to the `definitions.person` or `definitions.entity` specified in the Citation File Format schema. Each element of the `cff_pers_lst` object has classes `cff_pers, cff`.

References

- Patashnik, Oren. "BIBTEXTING" February 1988. https://osl.ugr.es/CTAN/biblio/bibtex/base/btxdoc.pdf.
- Markey, Nicolas. "Tame the BeaST" *The B to X of BibTeX, Version 1.4* (October 2007). https://osl.ugr.es/CTAN/info/bibtex/tamethebeast/ttb_en.pdf.
- Decoret X (2007). "A summary of BibTex."https://maverick.inria.fr/~Xavier.Decoret/resources/xdkbibtex/bibtex_summary.html#names

See **Examples** for more information.

See Also

Examples in `vignette("cffr", "cffr")` and `utils::person()`.

Learn more about the classes `cff_pers_lst, cff_pers` classes in cff_class.

Coercing between **R** classes with **S3 Methods**: `as_bibentry()`, `as_cff()`, `cff_class`

Examples

```
# Create a person object
a_person <- person(
  given = "First", family = "Author",
  role = c("aut", "cre"),
  email = "first.last@example.com", comment = c(
    ORCID = "0000-0001-8457-4658",
    affiliation = "An affiliation"
  )
)

a_person

cff_person <- as_cff_person(a_person)

# Class cff_pers_lst / cff
class(cff_person)

# With each element with class cff_pers / cff
class(cff_person[[1]])

# Print
cff_person

# Back to person object with S3 Method
as.person(cff_person)

# Coerce a string
a_str <- paste0(
  "Julio Iglesias <fake@email.com> ",
  "(city: Miami, region: California, country: US)"
)
as_cff_person(a_str)
```

```
# Several persons
persons <- c(
  person("Clark", "Kent", comment = c(affiliation = "Daily Planet")),
  person("Lois", "Lane"), person("Oscorp Inc.")
)

a_cff <- as_cff_person(persons)

a_cff

# Printed as Bibtex thanks to the method
toBibtex(a_cff)

# Or as person object
as.person(a_cff)

# Or you can use BibTeX style as input if you prefer

x <- "Frank Sinatra and Dean Martin and Davis, Jr., Sammy and Joey Bishop"

as_cff_person(x)

as_cff_person("Herbert von Karajan")

toBibtex(as_cff_person("Herbert von Karajan"))
```

`cff` *Create `cff` objects from direct inputs*

Description

A class and utility methods for reading, creating and holding CFF information. See cff_class to learn more about `cff` objects.

Usage

```
cff(path, ...)
```

Arguments

`path`	**[Deprecated]** `path` is no longer supported, use `cff_read_cff_citation()` instead.
`...`	Named arguments to be used for creating a `cff` object. If no arguments are supplied (the default behavior), a minimal valid `cff` object is created.

Details

`cff()` converts `_` in the argument name to `-`. For example, `cff_version = "1.2.0"` is converted to `cff-version = "1.2.0"`.

Valid arguments are those specified on `cff_schema_keys()`:

- `cff-version`
- `message`
- `type`
- `license`
- `title`
- `version`
- `doi`
- `identifiers`
- `abstract`
- `authors`
- `preferred-citation`
- `repository`
- `repository-artifact`
- `repository-code`
- `commit`
- `url`
- `date-released`
- `contact`
- `keywords`
- `references`
- `license-url`

Value

A `cff` object. Under the hood, a `cff` object is a regular `list` object with a special `print` method.

See Also

Other core functions of **cffr**: `cff_create()`, `cff_modify()`, `cff_validate()`

Examples

```
# Blank cff
cff()

# Use custom params
test <- cff(
  title = "Manipulating files",
  keywords = c("A", "new", "list", "of", "keywords"),
  authors = as_cff_person("New author")
)
test

# Would fail
cff_validate(test)


# Modify with cff_create
new <- cff_create(test, keys = list(
  "cff_version" = "1.2.0",
  message = "A blank file"
))
new

# Would pass
cff_validate(new)
```

`cff_create` *Create a `cff` object from several sources*

Description

Create a full and possibly valid `cff` object from a given source. This object can be written to a `*.cff` file with `cff_write()`, see **Examples**.

Most of the heavy lifting of **cffr** is done by this function.

Usage

```
cff_create(
  x,
  keys = list(),
  cff_version = "1.2.0",
  gh_keywords = TRUE,
  dependencies = TRUE,
  authors_roles = c("aut", "cre")
)
```

Arguments

`x` The source used to generate the `cff` object. It can be:

- A missing value, which retrieves the `DESCRIPTION` file from your in-development **R** package.
- An existing `cff` object.
- The name of an installed package (`"jsonlite"`).
- Path to a `DESCRIPTION` file (`"./DESCRIPTION"`).

`keys`	List of additional keys to add to the `cff` object. See `cff_modify()`.
`cff_version`	The Citation File Format schema version that the `CITATION.cff` file adheres to for providing the citation metadata.
`gh_keywords`	Logical `TRUE/FALSE`. If the package is hosted on GitHub, would you like to add the repo topics as keywords?
`dependencies`	Logical `TRUE/FALSE`. Should the dependencies of your package be added to the `references` CFF key?
`authors_roles`	Roles to be considered as authors of the package when generating the `CITATION.cff` file. See **Details**.

Details

If `x` is a path to a `DESCRIPTION` file or if `inst/CITATION` is not present on your package, **cffr** would auto-generate a `preferred-citation` key using the information provided on that file.

By default, only persons whose role in the `DESCRIPTION` file of the package is author (`"aut"`) or maintainer (`"cre"`) are considered to be authors of the package. The default setting can be controlled via the `authors_roles` argument. See **Details** on `person()` to get additional insights on person roles.

Value

A `cff` object.

See Also

Guide to Citation File Format schema version 1.2.0.

- `cff_modify()` as the recommended way to modify a `cff` object.
- `cff_write()` for creating a CFF file.
- `vignette("cffr", "cffr")` shows an introduction on how manipulate `cff` objects.
- `vignette("crosswalk", package = "cffr")` provides details on how the metadata of a package is mapped to produce a `cff` object.

Other core functions of **cffr**: `cff()`, `cff_modify()`, `cff_validate()`

Examples

```
# Installed package
cff_create("jsonlite")

# Demo file
demo_file <- system.file("examples/DESCRIPTION_basic", package = "cffr")
cff_create(demo_file)

# Add additional keys

newkeys <- list(
  message = "This overwrites fields",
  abstract = "New abstract",
  keywords = c("A", "new", "list", "of", "keywords"),
  authors = as_cff_person("New author")
)

cff_create(demo_file, keys = newkeys)

# Update a field on a list - i.e., authors, contacts, etc.
# We are adding a new contact here

old <- cff_create(demo_file)

new_contact <- append(
  old$contact,
  as_cff_person(person(
    given = "I am",
    family = "New Contact"
  ))
)

cff_create(demo_file, keys = list("contact" = new_contact))
```

`cff_gha_update` *Install a* R*hrefhttps://CRAN.R-project.org/package=cffr***cffr** *GitHub Action*

Description

This function installs a GitHub Action on your repository. The action updates your `CITATION.cff` when any of these events occur:

- You publish a new release of the package.
- Your `DESCRIPTION` or `inst/CITATION` file is modified.
- The action can be run manually.

Usage

```
cff_gha_update(path = ".", overwrite = FALSE)
```

Arguments

`path`	Project root directory.
`overwrite`	Logical. If the action already exists, should it be overwritten?

Details

Triggers on your action can be modified, see Events that trigger workflows.

Value

Invisible, this function is called by its side effects.

See Also

Other Git/GitHub helpers provided by **cffr**: `cff_git_hook`

Examples

```
## Not run:
cff_gha_update()

## End(Not run)
```

`cff_git_hook` *Use a git pre-commit hook* **[Experimental]**

Description

Install a pre-commit hook that reminds you to update your `CITATION.cff` file. This is a wrapper of `usethis::use_git_hook()`.

Usage

```
cff_git_hook_install()

cff_git_hook_remove()
```

Details

This function installs a pre-commit hook using `usethis::use_git_hook()`.

A pre-commit hook is a script that identifies simple issues before submission to code review. This pre-commit hook warns you if any of the following conditions are met:

- You included your `DESCRIPTION` or `inst/CITATION` file in a commit but did not include your `CITATION.cff`, and the `CITATION.cff` file is "older" than your `DESCRIPTION` or `inst/CITATION` file.
- You updated your `CITATION.cff` but did not include it in your commit.

Value

Invisible. This function is called for its side effects.

A word of caution

The pre-commit hook may prevent you from committing if you are not updating your `CITATION.cff`. However, the detection mechanism is not perfect and may be triggered even if you have attempted to update your `CITATION.cff` file.

This typically occurs when you have updated your `DESCRIPTION` or `inst/CITATION` files, but those changes do not affect your `CITATION.cff` file (i.e., you are adding new dependencies).

In those cases, you can override the check by running `git commit --no-verify` in the terminal.

If you are using **RStudio**, you can also run this command from an **R** script by selecting that line and sending it to the terminal using:

- Windows & Linux: `Ctrl+Alt+Enter`.
- Mac: `Cmd+Option+Return`.

Removing the git pre-commit hook

You can remove the pre-commit hook using `cff_git_hook_remove()`.

See Also

- `usethis::use_git_hook()`, that is the underlying function used by `cff_git_hook_install()`.
- `usethis::use_git()` and related function of **usethis** for using Git with **R** packages.

Other Git/GitHub helpers provided by **cffr**: `cff_gha_update()`

Examples

```
## Not run:
cff_git_hook_install()

## End(Not run)
```

`cff_modify` *Modify a `cff` object*

Description

Add new keys or modify existing ones in a `cff` object.

Usage

```
cff_modify(x, ...)
```

Arguments

`x`	A `cff` object.
`...`	Named arguments used to modify `x`. See also the `...` argument in `cff()`.

Details

Keys provided in ... override the corresponding key in x.

You can add additional keys not detected by cff_create() using the keys argument. A list of valid keys can be retrieved with cff_schema_keys(). For details, refer to Guide to Citation File Format schema version 1.2.0. for additional details.

Value

A cff object.

See Also

This function is a wrapper of utils::modifyList().

See cff() for creating cff objects from scratch.

Other core functions of **cffr**: cff(), cff_create(), cff_validate()

Examples

```
x <- cff()
x

cff_validate(x)

x_mod <- cff_modify(x,
  contact = as_cff_person("A contact"),
  message = "This overwrites fields",
  title = "New Title",
  abstract = "New abstract",
  doi = "10.21105/joss.03900"
)

x_mod

cff_validate(x_mod)
```

---

cff_read *Read an external file as a cff object*

Description

Read files and convert them to cff objects. Files supported are:

- CITATION.cff files.
- DESCRIPTION files.
- **R** citation files (usually located in inst/CITATION).
- BibTeX files (with extension *.bib).

`cff_read()` attempts to guess the type of file provided in `path`. However, we provide aliases for each specific file type:

- `cff_read_cff_citation()`, which uses `yaml::read_yaml()`.
- `cff_read_description()`, which uses `desc::desc()`.
- `cff_read_citation()`, which uses `utils::readCitationFile()`.
- `cff_read_bib()`, which requires **bibtex** (>= 0.5.0) and uses `bibtex::read.bib()`.

Usage

```
cff_read(path, ...)

cff_read_cff_citation(path, ...)

cff_read_description(
  path,
  cff_version = "1.2.0",
  gh_keywords = TRUE,
  authors_roles = c("aut", "cre"),
  ...
)

cff_read_citation(path, meta = NULL, ...)

cff_read_bib(path, encoding = "UTF-8", ...)
```

Arguments

`path`	Path to a file.
`...`	Arguments to be passed to other functions (i.e., to `yaml::read_yaml()`, `bibtex::read.bib()`, etc.).
`cff_version`	The Citation File Format schema version that the `CITATION.cff` file adheres to for providing the citation metadata.
`gh_keywords`	Logical `TRUE/FALSE`. If the package is hosted on GitHub, would you like to add the repo topics as keywords?
`authors_roles`	Roles to be considered as authors of the package when generating the `CITATION.cff` file. See **Details**.
`meta`	A list of package metadata as obtained by `utils::packageDescription()` or `NULL` (the default). See **Details**.
`encoding`	Encoding to be assumed for `path`. See `readLines()`.

Details

For details of `cff_read_description()` see `cff_create()`.

The `meta` object:
Section 1.9 CITATION files of *Writing R Extensions* (R Core Team 2023) specifies how to create dynamic `CITATION` files using `meta` object, hence the `meta` argument in `cff_read_citation()` may be needed for reading some files correctly.

Value

- `cff_read_cff_citation()` and `cff_read_description()` return an object with class `cff`.
- `cff_read_citation()` and `cff_read_bib()` return an object of classes `cff_ref_lst, cff` according to the `definitions.references` specified in the Citation File Format schema.

Learn more about the **cffr** class system in cff_class.

References

- R Core Team (2023). *Writing R Extensions*. `https://cran.r-project.org/doc/manuals/r-release/R-exts.html`
- Hernangomez D (2022). "BibTeX and CFF, a potential crosswalk." *The cffr package, Vignettes*. doi:10.21105/joss.03900, `https://docs.ropensci.org/cffr/articles/bibtex-cff.html`.

See Also

The underlying functions used for reading external files:

- `yaml::read_yaml()` for `CITATION.cff` files.
- `desc::desc()` for `DESCRIPTION` files.
- `utils::readCitationFile()` for **R** citation files.
- `bibtex::read.bib()` for BibTeX files (extension `*.bib`).

Other functions for reading external files: `cff_read_bib_text()`

Other functions for working with BibTeX format: `as_bibentry()`, `cff_read_bib_text()`, `cff_write_bib()`, `encoded_utf_to_latex()`

Examples

```
# Create cff object from cff file

from_cff_file <- cff_read(system.file("examples/CITATION_basic.cff",
  package = "cffr"
))

head(from_cff_file, 7)

# Create cff object from DESCRIPTION
from_desc <- cff_read(system.file("examples/DESCRIPTION_basic",
  package = "cffr"
))

from_desc

# Create cff object from BibTex

if (requireNamespace("bibtex", quietly = TRUE)) {
  from_bib <- cff_read(system.file("examples/example.bib",
    package = "cffr"
  ))

  # First item only
  from_bib[[1]]
}
# Create cff object from CITATION
from_citation <- cff_read(system.file("CITATION", package = "cffr"))

# First item only
from_citation[[1]]
```

cff_read_bib_text *Read BibTeX markup as a cff_ref_lst object*

Description

Convert a character string representing a BibTeX entry into a cff_ref_lst object.

Usage

```
cff_read_bib_text(x, encoding = "UTF-8", ...)
```

Arguments

x	A character vector with one or more complete BibTeX entries.
encoding	Encoding to be assumed for x. See readLines().
...	Arguments passed to cff_read_bib().

Details

This function writes x to a temporary *.bib file and reads it using cff_read_bib().

This function requires **bibtex** (>= 0.5.0) and uses bibtex::read.bib() for parsing.

Value

An object of classes cff_ref_lst, cff according to the definitions.references specified in the Citation File Format schema. Each element of the cff_ref_lst object has classes cff_ref, cff.

See Also

cff_read_bib() for reading *.bib files.

Other functions for working with BibTeX format: as_bibentry(), cff_read(), cff_write_bib(), encoded_utf_to_latex()

Other functions for reading external files: cff_read()

Examples

```
if (requireNamespace("bibtex", quietly = TRUE)) {
  x <- c(
    "@book{einstein1921,
      title        = {Relativity: The Special and the General Theory},
      author       = {Einstein, Albert},
      year         = 1920,
      publisher    = {Henry Holt and Company},
      address      = {London, United Kingdom},
      isbn         = 9781587340925
  }",
    "@misc{misc-full,
      title        = {Handing out random pamphlets in airports},
      author       = {Joe-Bob Missilany},
      year         = 1984,
      month        = oct,
      note         = {This is a full MISC entry},
      howpublished = {Handed out at O'Hare}
  }"
  )

  cff_read_bib_text(x)
}
```

`cff_schema` *Schema utils*

Description

Helper functions with the valid values of different fields, according to the Citation File Format schema version 1.2.0.

- `cff_schema_keys()` provides the valid high-level keys of the Citation File Format.
- `cff_schema_keys_license()` provides the valid SPDX license identifier(s) to be used on the `CITATION.cff` file.
- `cff_schema_definitions_person()` and `cff_schema_definitions_entity()` returns the valid fields to be included when defining a person or entity.
- `cff_schema_definitions_refs()` provides the valid keys to be used on the `preferred-citation` and `references` keys.

Usage

```
cff_schema_keys(sorted = FALSE)

cff_schema_keys_license()

cff_schema_definitions_person()
```

```
cff_schema_definitions_entity()

cff_schema_definitions_refs()
```

Arguments

`sorted`	Logical `TRUE/FALSE`. Should the keys be arranged alphabetically?

Value

A vector of characters with the names of the valid keys to be used on a Citation File Format version 1.2.0

Source

Guide to Citation File Format schema version 1.2.0.

Examples

```
cff_schema_keys(sorted = TRUE)

# Valid Licenses keys
head(cff_schema_keys_license(), 20)

cff_schema_definitions_person()

cff_schema_definitions_entity()

cff_schema_definitions_refs()
```

`cff_validate` *Validate a* `CITATION.cff` *file or a* `cff` *object*

Description

Validate a `CITATION.cff` file or a `cff` object using the corresponding validation schema.

Usage

```
cff_validate(x = "CITATION.cff", verbose = TRUE)
```

Arguments

`x`	This is expected to be either a full `cff` object created with `cff_create()` or the path to a `CITATION.cff` file to be validated. In the case of a `*.cff` file it would read with `cff_read_cff_citation()`.
`verbose`	Logical `TRUE/FALSE`. When `TRUE`, the function displays informative messages.

Value

A message indicating the result of the validation and an invisible value `TRUE/FALSE`. On error, the result has an attribute `"errors"` containing the error summary (see **Examples** and `attr()`).

See Also

Guide to Citation File Format schema version 1.2.0.

`jsonvalidate::json_validate()`, that is the function that performs the validation.

Other core functions of **cffr**: `cff()`, `cff_create()`, `cff_modify()`

Examples

```
# Full .cff example
cff_validate(system.file("examples/CITATION_complete.cff", package = "cffr"))

# Validate a cffr object
cffr <- cff_create("jsonlite")
class(cffr)
cff_validate(cffr)


# .cff with errors
err_f <- system.file("examples/CITATION_error.cff", package = "cffr")
# Can manipulate the errors as data frame
res <- try(cff_validate(err_f))

isTRUE(res)
isFALSE(res)

attr(res, "errors")

# If a CITATION file (note that is not .cff) it throws an error
try(cff_validate(system.file("CITATION", package = "cffr")))
```

`cff_write` *Write a* `CITATION.cff` *file*

Description

This is the core function of the package and likely to be the only one you would need when developing a package.

This function writes out a `CITATION.cff` file for a given package. This function is basically a wrapper around `cff_create()` to both create the `cff` object and write it out to a YAML-formatted file in one command.

Usage

```
cff_write(
  x,
  outfile = "CITATION.cff",
  keys = list(),
  cff_version = "1.2.0",
  gh_keywords = TRUE,
  r_citation = FALSE,
  dependencies = TRUE,
  validate = TRUE,
  verbose = TRUE,
  authors_roles = c("aut", "cre"),
  encoding = "UTF-8"
)
```

Arguments

`x`	The source used to generate the `cff` object. It can be: • A missing value, which retrieves the `DESCRIPTION` file from your in-development **R** package. • An existing `cff` object. • The name of an installed package (`"jsonlite"`). • Path to a `DESCRIPTION` file (`"./DESCRIPTION"`).
`outfile`	The name and path of the `CITATION.cff` to be created.
`keys`	List of additional keys to add to the `cff` object. See `cff_modify()`.
`cff_version`	The Citation File Format schema version that the `CITATION.cff` file adheres to for providing the citation metadata.
`gh_keywords`	Logical `TRUE/FALSE`. If the package is hosted on GitHub, would you like to add the repo topics as keywords?
`r_citation`	Logical `TRUE/FALSE`. When `TRUE`, the **R** package citation (i.e., `inst/CITATION`) is created or updated. **No backup copy is created**. For more control, use `cff_write_citation()`.
`dependencies`	Logical `TRUE/FALSE`. Should the dependencies of your package be added to the `references` CFF key?
`validate`	Logical `TRUE/FALSE`. Should the new file be validated using `cff_validate()`?
`verbose`	Logical `TRUE/FALSE`. When `TRUE`, the function displays informative messages.
`authors_roles`	Roles to be considered as authors of the package when generating the `CITATION.cff` file. See **Details**.
`encoding`	The name of the encoding to be assumed. Default is `"UTF-8"`, but it can be any other value as accepted by `iconv()`, such as `"ASCII//TRANSLIT"`.

Details

For details of `authors_roles` see `cff_create()`.

When creating and writing a `CITATION.cff` for the first time, the function adds the pattern `"^CITATION\.cff$"` to your `.Rbuildignore` file to avoid `NOTE`s and `WARNING`s in `R CMD CHECK`.

Value

A `CITATION.cff` file and an (invisible) `cff` object.

See Also

Guide to Citation File Format schema version 1.2.0. This function unifies the workflow `cff_create()` + `cff_validate()` + write a file.

Other functions for creating external files: `cff_write_bib()`

Examples

```
tmpfile <- tempfile(fileext = ".cff")
cff_obj <- cff_write("jsonlite", outfile = tmpfile)

cff_obj

# Force clean-up
file.remove(tmpfile)
```

`cff_write_bib` *Export **R** objects to different file types*

Description

Export **R** objects representing citations to specific file formats:

- `cff_write_bib()` creates a `.bib` file.
- `cff_write_citation()` creates an **R** citation file as described in Section 1.9 of *Writing R Extensions* (R Core Team 2023).

Usage

```
cff_write_bib(
  x,
  file = tempfile(fileext = ".bib"),
  append = FALSE,
  verbose = TRUE,
  ascii = FALSE,
  ...
)

cff_write_citation(
  x,
  file = tempfile("CITATION_"),
  append = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`x`	A `bibentry` or a `cff` object.
`file`	Name of the file to be created. If `NULL`, the lines are displayed instead.
`append`	Logical. Should entries be appended to an existing file?
`verbose`	Logical. Display informative messages.
`ascii`	Logical. Should entries be written using ASCII characters only?
`...`	Arguments passed on to `as_bibentry.cff`, `as_bibentry.cff_ref`, `as_bibentry.cff_ref_lst`

`what` Fields to extract from a full `cff` object. The value could be:

- `preferred`: This would create a single entry with the main citation info of the package (key `preferred-citation`).
- `references`: Extract all the entries of `references` key.
- `all`: A combination of the previous two options. This would extract both the `preferred-citation` and the `references` key.

See `vignette("crosswalk", package = "cffr")`.

Details

When `x` is a `cff` object, it is converted to BibTeX using `toBibtex.cff()`.

For security reasons, if the file already exists, the function creates a backup copy in the same directory.

Value

Writes the corresponding file specified on the `file` argument.

References

- R Core Team (2023). *Writing R Extensions*. `https://cran.r-project.org/doc/manuals/r-release/R-exts.html`

See Also

`vignette("bibtex_cff", "cffr")`, `knitr::write_bib()` and the following packages:

- **bibtex**.
- **RefManageR**
- **rbibutils**

Other functions for working with BibTeX format: `as_bibentry()`, `cff_read()`, `cff_read_bib_text()`, `encoded_utf_to_latex()`

Other functions for creating external files: `cff_write()`

Examples

```
bib <- bibentry("Misc",
  title = "My title",
  author = "Fran Pérez"
)

my_temp_bib <- tempfile(fileext = ".bib")

cff_write_bib(bib, file = my_temp_bib)

cat(readLines(my_temp_bib), sep = "\n")

cff_write_bib(bib, file = my_temp_bib, ascii = TRUE, append = TRUE)

cat(readLines(my_temp_bib), sep = "\n")

# Create a CITATION file

# Use a system file
f <- system.file("examples/preferred-citation-book.cff", package = "cffr")
a_cff <- cff_read(f)

out <- file.path(tempdir(), "CITATION")
cff_write_citation(a_cff, file = out)

# Check by reading, use meta object
meta <- packageDescription("cffr")
meta$Encoding <- "UTF-8"

utils::readCitationFile(out, meta)
```

`cran_to_spdx` *Mapping between* `License` *fields and SPDX*

Description

A dataset containing the mapping between the `License` strings observed on CRAN packages and its (approximate) match on the SPDX License List.

Usage

```
cran_to_spdx
```

Format

A data frame with 86 rows and 2 variables:

- LICENSE: A valid `License` string on CRAN.
- SPDX. A valid SPDX License Identifier.

Source

https://spdx.org/licenses/

See Also

Writing R Extensions, Licensing section.

Examples

```
data("cran_to_spdx")

head(cran_to_spdx, 20)
```

Index