

Package ‘cfr’

May 8, 2026

Title Generate Citation File Format (‘cfr’) Metadata for R Packages

Version 1.4.0

Description The Citation File Format version 1.2.0
<[doi:10.5281/zenodo.5171937](https://doi.org/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/cfr/>, <https://github.com/ropensci/cfr>

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

Depends R (>= 4.1.0)

Imports cli (>= 2.0.0), desc (>= 1.3.0), jsonlite (>= 1.7.2),
jsonvalidate (>= 1.5.0), tools, utils, yaml (>= 2.2.1)

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

VignetteBuilder quarto

Config/Needs/website devtools

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

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

X-schema.org-keywords attribution, citation, credit, citation-files,
cfr, 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 2026-05-04 15:40:02 UTC

Contents

as_bibentry	2
as_cff	5
as_cff_person	7
cff	10
cff_create	12
cff_gha_update	14
cff_git_hook	15
cff_modify	16
cff_read	17
cff_read_bib_text	20
cff_schema	21
cff_validate	22
cff_write	23
cff_write_bib	25
cran_to_spdx	27

Index **29**

as_bibentry *Create [bibentry](#) objects from several sources*

Description

This function creates [bibentry](#) objects from different metadata sources ([cff](#) objects, DESCRIPTION files and other sources). 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 cff . It can be: <ul style="list-style-type: none"> • A missing value, which retrieves the DESCRIPTION file from your in-development package. • An existing cff object created with <code>cff()</code>, <code>cff_create()</code>, or <code>as_cff()</code>. • 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: <ul style="list-style-type: none"> • 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. <p>See <code>vignette("crosswalk", package = "cffr")</code>.</p>

Details

An **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()` and `as_cff.Bibtex()`, see `vignette("bibtex_cff", "cfr")` to understand how the mapping is performed.

`as_cff_person()` is preferred over `as_cff.person()`, since it can handle character inputs 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 **cfr** 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 for an **R** package.
- `cff_read()`: Create a `cff` object from an external file.
- `as_cff_person()`: Recommended way to create persons in CFF format.

Learn more about the **cfr** 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("cfr", "cfr")` and `utils::person()`.

Learn more about the `cff_pers_lst` and `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

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

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

- `cffi-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 `cffi` object. Under the hood, a `cffi` object is a regular `list` object with a special `print` method.

See Also

Other core functions of **cffi**: `cffi_create()`, `cffi_modify()`, `cffi_validate()`

Examples

```
# Blank cffi
cffi()

# Use custom params
test <- cffi(
  title = "Manipulating files",
```

```

    keywords = c("A", "new", "list", "of", "keywords"),
    authors = as_cffi_person("New author")
  )
  test

# Would fail
cffi_validate(test)

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

# Would pass
cffi_validate(new)

```

cffi_create

Create a cffi object from several sources

Description

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

Most of the heavy lifting of `cffi` is done by this function.

Usage

```

cffi_create(
  x,
  keys = list(),
  cffi_version = "1.2.0",
  gh_keywords = TRUE,
  dependencies = TRUE,
  authors_roles = c("aut", "cre")
)

```

Arguments

- `x` The source used to generate the `cffi` object. It can be:
- A missing value, which retrieves the DESCRIPTION file from your in-development **R** package.
 - An existing `cffi` object.
 - The name of an installed package (`"jsonlite"`).
 - Path to a DESCRIPTION file (`"/DESCRIPTION"`).

keys	List of additional keys to add to the <code>cffi</code> object. See <code>cffi_modify()</code> .
cffi_version	The Citation File Format schema version that the CITATION.cffi 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.cffi file. See Details .

Details

If `x` is a path to a DESCRIPTION file or if `inst/CITATION` is not present on your package, `cffi` 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 `cffi` object.

See Also

[Guide to Citation File Format schema version 1.2.0.](#)

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

Other core functions of `cffi`: `cffi()`, `cffi_modify()`, `cffi_validate()`

Examples

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

# Demo file
demo_file <- system.file("examples/DESCRIPTION_basic", package = "cffi")
cffi_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, for example authors or contacts.
# 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	<i>Install a Rhrefhttps://CRAN.R-project.org/package=cffrcffr GitHub Action</i>
----------------	--

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 **cffi**: [cffi_git_hook](#)

Examples

```
## Not run:  
cffi_gha_update()  
  
## End(Not run)
```

cffi_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 around `usethis::use_git_hook()`.

Usage

```
cffi_git_hook_install()
```

```
cffi_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 (for example, 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()`, which 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 **cfr**: `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 `cffi_create()` using the `keys` argument. A list of valid keys can be retrieved with `cffi_schema_keys()`. For details, refer to [Guide to Citation File Format schema version 1.2.0](#). for additional details.

Value

A `cffi` object.

See Also

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

See `cffi()` for creating `cffi` objects from scratch.

Other core functions of **cffi**: `cffi()`, `cffi_create()`, `cffi_validate()`

Examples

```
x <- cffi()
x

cffi_validate(x)

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

x_mod

cffi_validate(x_mod)
```

cffi_read

Read an external file as a cffi object

Description

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

- CITATION.cffi 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** ($\geq 0.5.0$) 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

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

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

- `cfr_read_cfr_citation()` and `cfr_read_description()` return an object with class `cfr`.
- `cfr_read_citation()` and `cfr_read_bib()` return an object of classes `cfr_ref_lst`, `cfr` according to the definitions.references specified in the [Citation File Format schema](#).

Learn more about the `cfr` class system in [cfr_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 cfr package, Vignettes*. doi:10.21105/joss.03900, <https://docs.ropensci.org/cfr/articles/bibtex-cff.html>.

See Also

The underlying functions used for reading external files:

- `yaml::read_yaml()` for CITATION.cfr 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: `cfr_read_bib_text()`

Other functions for working with BibTeX format: `as_bibentry()`, `cfr_read_bib_text()`, `cfr_write_bib()`, `encoded_utf_to_latex()`

Examples

```
# Create cfr object from cfr file

from_cfr_file <- cfr_read(system.file("examples/CITATION_basic.cfr",
  package = "cfr")
))

head(from_cfr_file, 7)

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

from_desc

# Create cfr object from BibTeX

if (requireNamespace("bibtex", quietly = TRUE)) {
  from_bib <- cfr_read(system.file("examples/example.bib",
    package = "cfr")
  )
}
```

```

))

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

# 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** ($\geq 0.5.0$) 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("bibtext", 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}
    }"
  )

  cffi_read_bib_text(x)
}

```

cffi_schema

Schema utils

Description

Helper functions with the valid values of different fields, according to the [Citation File Format schema version 1.2.0](#).

- `cffi_schema_keys()` provides the valid high-level keys of the Citation File Format.
- `cffi_schema_keys_license()` provides the valid [SPDX license identifier\(s\)](#) to be used on the CITATION.cff file.
- `cffi_schema_definitions_person()` and `cffi_schema_definitions_entity()` return the valid fields to include when defining a person or entity.
- `cffi_schema_definitions_refs()` provides the valid keys to be used on the preferred-citation and references keys.

Usage

```
cffi_schema_keys(sorted = FALSE)
```

```
cffi_schema_keys_license()
```

```
cffi_schema_definitions_person()
```

```
cff_schema_definitions_entity()
```

```
cff_schema_definitions_refs()
```

Arguments

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

Value

A character vector with the names of valid keys for 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	<i>Validate a CITATION.cff file or a cff object</i>
--------------	---

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\(\)](#), which is the function that performs the validation.

Other core functions of **cffi**: [cffi\(\)](#), [cffi_create\(\)](#), [cffi_modify\(\)](#)

Examples

```
# Full .cffi example
cffi_validate(system.file("examples/CITATION_complete.cffi", package = "cffi"))

# Validate a cffi object
cffi <- cffi_create("jsonlite")
class(cffi)
cffi_validate(cffi)

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

isTRUE(res)
isFALSE(res)

# Detailed results
df_errors <- attr(res, "errors")

df_errors[, c("field", "message")]

# If a CITATION file is supplied (note that it is not a .cffi file), it
# throws an error
try(cffi_validate(system.file("CITATION", package = "cffi")))
```

cffi_write

Write a CITATION.cffi 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.cffi file for a given package. This function is basically a wrapper around [cffi_create\(\)](#) to both create the *cffi* 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 <code>cff</code> object. It can be: <ul style="list-style-type: none"> • A missing value, which retrieves the DESCRIPTION file from your in-development R package. • An existing <code>cff</code> object. • The name of an installed package (<code>"jsonlite"</code>). • Path to a DESCRIPTION file (<code>"/DESCRIPTION"</code>).
outfile	The name and path of the CITATION.cff to be created.
keys	List of additional keys to add to the <code>cff</code> object. See <code>cff_modify()</code> .
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 (for example, inst/CITATION) is created or updated. No backup copy is created. For more control, use <code>cff_write_citation()</code> .
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 <code>cff_validate()</code> ?
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 <code>"UTF-8"</code> , but it can be any other value as accepted by <code>iconv()</code> , such as <code>"ASCII//TRANSLIT"</code> .

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.

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 cffi 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.cffi , as_bibentry.cffi_ref , as_bibentry.cffi_ref_lst what Fields to extract from a full cffi object. The value could be: <ul style="list-style-type: none"> • 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 <code>vignette("crosswalk", package = "cffi")</code> .

Details

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

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_cffi", "cffi")`, `knitr::write_bib()` and the following packages:

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

Other functions for working with BibTeX format: `as_bibentry()`, `cffi_read()`, `cffi_read_bib_text()`, `encoded_utf_to_latex()`

Other functions for creating external files: `cffi_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](#).

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

- * **bibtex**
 - as_bibentry, 2
 - cff_read, 17
 - cff_read_bib_text, 20
 - cff_write_bib, 25
 - * **core**
 - cff, 10
 - cff_create, 12
 - cff_modify, 16
 - cff_validate, 22
 - * **datasets**
 - cran_to_spdx, 27
 - * **git**
 - cff_gha_update, 14
 - cff_git_hook, 15
 - * **reading**
 - cff_read, 17
 - cff_read_bib_text, 20
 - * **s3method**
 - as_bibentry, 2
 - as_cff, 5
 - as_cff_person, 7
 - * **schemas**
 - cff_schema, 21
 - * **writing**
 - cff_write, 23
 - cff_write_bib, 25
- as.cff(as_cff), 5
- as.character(), 7
- as.list(), 5
- as.person.cff_pers(), 7
- as.person.cff_pers_lst(), 7
- as_bibentry, 2
- as_bibentry(), 6, 9, 19, 20, 26
- as_bibentry.cff, 26
- as_bibentry.cff_ref, 26
- as_bibentry.cff_ref_lst, 26
- as_cff, 5
- as_cff(), 3, 4, 9
- as_cff.bibentry(), 2
- as_cff_person, 7
- as_cff_person(), 4, 6
- attr(), 23
- bibentry, 2, 26
- bibtex::read.bib(), 18–20
- cff, 2, 3, 5, 10, 10, 11–13, 16, 17, 22–24, 26
- cff(), 3, 6, 13, 16, 17, 23
- cff_class, 4, 6, 9, 10, 19
- cff_create, 12
- cff_create(), 3, 6, 11, 17, 18, 22–25
- cff_gha_update, 14
- cff_gha_update(), 16
- cff_git_hook, 15, 15
- cff_git_hook_install(cff_git_hook), 15
- cff_git_hook_remove(cff_git_hook), 15
- cff_modify, 16
- cff_modify(), 6, 11, 13, 23, 24
- cff_pers_lst, 7
- cff_read, 17
- cff_read(), 4, 6, 18, 20, 26
- cff_read_bib(cff_read), 17
- cff_read_bib(), 18, 20
- cff_read_bib_text, 20
- cff_read_bib_text(), 4, 19, 26
- cff_read_cff_citation(cff_read), 17
- cff_read_cff_citation(), 10, 18, 22
- cff_read_citation(cff_read), 17
- cff_read_citation(), 18
- cff_read_description(cff_read), 17
- cff_read_description(), 18
- cff_ref_lst, 2, 20
- cff_schema, 21
- cff_schema_definitions_entity(cff_schema), 21
- cff_schema_definitions_entity(), 21
- cff_schema_definitions_person(cff_schema), 21

`cff_schema_definitions_person()`, 21
`cff_schema_definitions_refs`
 (`cff_schema`), 21
`cff_schema_definitions_refs()`, 21
`cff_schema_keys` (`cff_schema`), 21
`cff_schema_keys()`, 11, 17, 21
`cff_schema_keys_license` (`cff_schema`), 21
`cff_schema_keys_license()`, 21
`cff_validate`, 22
`cff_validate()`, 11, 13, 17, 25
`cff_write`, 23
`cff_write()`, 12, 13, 26
`cff_write_bib`, 25
`cff_write_bib()`, 4, 19, 20, 25
`cff_write_citation` (`cff_write_bib`), 25
`cff_write_citation()`, 24, 25
`cran_to_spdx`, 27

`desc::desc()`, 18, 19

`encoded_utf_to_latex()`, 4, 19, 20, 26

`format()`, 7
`format.person()`, 7, 8

`iconv()`, 24

`jsonvalidate::json_validate()`, 23

`knitr::write_bib()`, 26

`list`, 11

`person()`, 7, 13
`print`, 11

`readLines()`, 18, 20

`sub-class`, 5

`toBibtex()`, 2, 3, 5
`toBibtex.cff()`, 26

`usethis::use_git()`, 16
`usethis::use_git_hook()`, 15, 16
`utils::bibentry()`, 4, 5
`utils::modifyList()`, 17
`utils::packageDescription()`, 18
`utils::person()`, 5, 9
`utils::readCitationFile()`, 18, 19
`utils::toBibtex()`, 4

`yaml::read_yaml()`, 18, 19