Package: citeme (via r-universe)

June 7, 2026

Title Manage Person and Organisation Information

Version 0.1.1

Description Manage person and organisation information with validation and formatting capabilities. Provides R6 classes for managing organisations and their members, with support for multiple languages, ORCID identifiers, ROR identifiers, licensing requirements, publisher information, and integration with citation management systems.

License GPL-3

URL https://inbo.github.io/citeme/

BugReports https://github.com/inbo/citeme/issues

Depends R (>= 4.1.0)

Imports assertthat, desc, gert, jsonlite, knitr, R6, tools, utils, yaml

Suggests mockery, quarto, testthat (>= 3.0.0)

VignetteBuilder quarto

Config/citeme/communities inbo

Config/citeme/keywords organisation; author; standardisation; inbo_version

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

Roxygen list(markdown = TRUE)

Config/pak/sysreqs git libgit2-dev libssl-dev

Repository https://inbo.r-universe.dev

Date/Publication 2026-05-08 09:50:02 UTC

RemoteUrl https://github.com/inbo/citeme

RemoteRef HEAD

RemoteSha c9e2896d883fe5db5876a63809a935090ea677f3

Contents

add_badges *add badges to a README*

Description

- doi: add a DOI badge
- language: add a language badge
- license: add a license badge
- url: add a website badge
- version: add a version badge

Usage

```
add_badges(readme_path = ".", ...)
```

Arguments

readme_path	Directory containing the README.md or README.Rmd file.
...	Additional arguments

See Also

Other utils: coalesce()

Examples

```
## Not run:
  add_badges(url = "https://www.inbo.be")
  add_badges(doi = "10.5281/zenodo.8063503")
  add_badges(version = "v0.1.2")
  add_badges(url = "https://www.inbo.be", doi = "10.5281/zenodo.8063503")

## End(Not run)
```

add_individual *Add an individual to a file*

Description

Add an individual to a file

Usage

```
add_individual(
  path = ".",
  role = c("aut", "cre", "ctb", "rev", "cph", "fnd", "pbl")
)
```

Arguments

path	A path to a file or directory. Supported file types: • Quarto files (`_quarto.yml` or `*.qmd`) • R package DESCRIPTION files • README files (`README.md` or `README.Rmd`) • Bookdown index files (`index.md` or `index.Rmd`)
role	The role of the person to add. One or multiple of `"aut"` (author), `"cre"` (creator/maintainer), `"ctb"` (contributor), `"rev"` (reviewer), `"cph"` (copyright holder), `"fnd"` (funder), or `"pbl"` (publisher). Unless you want to an individual to a Quarto or bookdown document. Then you can only specify one role, and `"aut"` will be added to the `author` field, #' `"rev"` to the `reviewer` field, `"cph"` to the `rightsholder` field, `"fnd"` to the `funder` field, and `"pbl"` to the `publisher` field.

See Also

Other individual: `has_person_role()`, `individual2badge()`, `individual2df()`, `individual2person()`, `select_individual()`, `select_person_role()`, `store_individuals()`

---

ask_email *Ask for an e-mail address*

Description

Ask for an e-mail address

Usage

```
ask_email(prompt)
```

Arguments

prompt	the string printed when prompting the user for input. Should usually end with a space `" "`.

See Also

Other question: `ask_language()`, `ask_new_license()`, `ask_orcid()`, `ask_ror()`, `ask_url()`, `ask_yes_no()`, `menu_first()`

ask_language *Ask for a language*

Description

Ask for a language

Usage

```
ask_language(org, prompt = "Which language?")
```

Arguments

org	An org_list object containing the available languages.
prompt	the string printed when prompting the user for input. Should usually end with a space " ".

See Also

Other question: ask_email(), ask_new_license(), ask_orcid(), ask_ror(), ask_url(), ask_yes_no(), menu_first()

ask_new_license *Ask one or more licenses.*

Description

Ask one or more licenses.

Usage

```
ask_new_license(licenses, type = c("package", "project", "data"))
```

Arguments

licenses	A named vector of available licenses. The vector contains the URL to the markdown version of the license. Use the abbreviation of the license name as names.
type	The type of license. Must be one of "package", "project" or "data".

See Also

Other question: ask_email(), ask_language(), ask_orcid(), ask_ror(), ask_url(), ask_yes_no(), menu_first()

ask_orcid *Ask for an* ORCID iD

Description

The ORCID iD is a unique, persistent identifier free of charge to researchers. This function prompts the user to enter an ORCID iD and validates its format. The ORCID iD must be in the format 0000-0000-0000-0000 where the last digit can be a number or "X". Empty strings are considered valid to allow optional ORCID iD.

Usage

```
ask_orcid(prompt = "orcid: ")
```

Arguments

prompt	A character string to display as a prompt to the user. The default is "orcid: ".

Value

A character string containing the ORCID iD entered by the user.

See Also

Other question: ask_email(), ask_language(), ask_new_license(), ask_ror(), ask_url(), ask_yes_no(), menu_first()

ask_ror *Ask for an ROR*

Description

The Research Organization Registry (ROR) is a global, community-led registry of open persistent identifiers for research organizations. ROR makes it easy for anyone or any system to disambiguate institution names and connect research organizations to researchers and research outputs.

Usage

```
ask_ror(prompt)
```

Arguments

prompt	the string printed when prompting the user for input. Should usually end with a space " ".

See Also

Other question: ask_email(), ask_language(), ask_new_license(), ask_orcid(), ask_url(), ask_yes_no(), menu_first()

---

ask_url *Ask an URL*

Description

Ask an URL

Usage

```
ask_url(prompt)
```

Arguments

prompt	the string printed when prompting the user for input. Should usually end with a space " ".

See Also

Other question: ask_email(), ask_language(), ask_new_license(), ask_orcid(), ask_ror(), ask_yes_no(), menu_first()

---

ask_yes_no *Function to ask a simple yes no question*

Description

Provides a simple wrapper around utils::askYesNo(). This function is used to ask yes no questions in an interactive way. It repeats the question until a valid answer is given.

Usage

```
ask_yes_no(msg, default = TRUE, prompts = c("Yes", "No", "Cancel"), ...)
```

Arguments

msg	The prompt message for the user.
default	The default response.
prompts	Any of: a character vector containing 3 prompts corresponding to return values of TRUE, FALSE, or NA, or a single character value containing the prompts separated by / characters, or a function to call.
...	Additional parameters, ignored by the default function.

See Also

Other question: `ask_email()`, `ask_language()`, `ask_new_license()`, `ask_orcid()`, `ask_ror()`, `ask_url()`, `menu_first()`

`cache_org` *Cache organisation list from URL*

Description

This function retrieves the organisation list from a given URL and caches it locally. If the URL is `"https://github.com/inbo"`, it uses the default INBO organisation list. For other URLs, it checks if a public `citeme` repository exists and clones it to retrieve the organisation list. The cached organisation list is stored in a specified configuration folder.

Usage

```
cache_org(url, config_folder = R_user_dir("citeme", "config"))
```

Arguments

`url`	The URL of the organisation list to retrieve.
`config_folder`	The folder where the cached organisation list should be stored. Defaults to the user's R configuration directory for `citeme`.

Value

The retrieved organisation list, or `NULL` if the URL is invalid or the repository.

See Also

Other organisation: `get_available_organisations()`, `get_default_org_list()`, `inbo_org_list()`, `new_org_item()`, `new_org_list()`, `org_list_from_url()`, `organisation2df()`, `select_license()`, `store_organisations()`, `stored_organisations()`

`citation_meta` *The* `citation_meta` *R6 class*

Description

A class which contains citation information.

Active bindings

`get_errors` Return the errors

`get_meta` Return the meta data as a list

`get_notes` Return the notes

`get_person` Return the individuals and organisations as a list of `person` objects.

`get_type` A string indicating the type of source.

`get_path` The path to the project.

`get_warnings` Return the warnings

Methods

Public methods:

- `citation_meta$new()`
- `citation_meta$print()`
- `citation_meta$clone()`

`citation_meta$new()`: Initialize a new `citation_meta` object.

Usage:

```
citation_meta$new(path = ".")
```

Arguments:

`path` The path to the root of the project or the file from which to extract citation metadata.

`citation_meta$print()`: Print the `citation_meta` object.

Usage:

```
citation_meta$print(...)
```

Arguments:

`...` currently ignored.

`citation_meta$clone()`: The objects of this class are cloneable with this method.

Usage:

```
citation_meta$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other class: `org_item`, `org_list`

coalesce *coalesce*

Description

Return the first non-NULL argument.

Usage

```
coalesce(...)
```

Arguments

...	Arguments to check for non-NULL values.

Value

The first non-NULL argument, or NULL if all arguments are NULL.

See Also

Other utils: `add_badges()`

get_available_organisations *Get the list of available organisations*

Description

This function retrieves the list of available organisations from the local configuration files and locally stored organisations. It returns a list containing the names, languages, licenses, ORCID, Zenodo, ROR, website, and logo of the available organisations.

Usage

```
get_available_organisations()
```

Value

A list containing the names, languages, licenses, ORCID, Zenodo, ROR, website, and logo of the available organisations.

See Also

Other organisation: `cache_org()`, `get_default_org_list()`, `inbo_org_list()`, `new_org_item()`, `new_org_list()`, `org_list_from_url()`, `organisation2df()`, `select_license()`, `store_organisations()`, `stored_organisations()`

get_default_org_list *Get the default organization list*

Description

This function retrieves the default organisation list from the `organisation.yml` file in the organisations `citeme` repository. The `origin` of the repository is used to determine the root URL of the organisation.

Usage

```
get_default_org_list(x = ".")
```

Arguments

`x`	The path to the repository. Defaults to the current working directory.

Value

An `org_list` object containing the organisation list. The function also stores the information in the user's R configuration.

See Also

Other organisation: `cache_org()`, `get_available_organisations()`, `inbo_org_list()`, `new_org_item()`, `new_org_list()`, `org_list_from_url()`, `organisation2df()`, `select_license()`, `store_organisations()`, `stored_organisations()`

has_person_role *Does a person object has a given role?*

Description

Does a person object has a given role?

Usage

```
has_person_role(individual, role)
```

Arguments

`individual`	A `person` object.
`role`	A character vector of roles to check for.

Value

A logical vector indicating whether the individual has any of the specified roles.

See Also

Other individual: add_individual(), individual2badge(), individual2df(), individual2person(), select_individual(), select_person_role(), store_individuals()

---

inbo_org_list *The INBO organisation list*

Description

The INBO organisation list

Usage

```
inbo_org_list()
```

See Also

Other organisation: cache_org(), get_available_organisations(), get_default_org_list(), new_org_item(), new_org_list(), org_list_from_url(), organisation2df(), select_license(), store_organisations(), stored_organisations()

---

individual2badge *Create a markdown badge for an individual*

Description

This function creates a markdown badge for an individual, including their name, ORCID, email, and affiliation if available. It also adds a footnote with the individual's role and affiliation.

Usage

```
individual2badge(
  individual,
  role = c("aut", "cre", "cph", "ctb", "fnd", "rev")
)
```

Arguments

individual	A data frame with columns given, family, orcid, email, and affiliation.
role	A character string indicating the individual's role. Must be one or more of "aut" (author), "cre" (contact person), "cph" (copyright holder), "ctb" (contributor), "fnd" (funder), "rev" (reviewer). Default is '"aut"'.

Value

A character string containing the markdown badge for the individual.

See Also

Other individual: add_individual(), has_person_role(), individual2df(), individual2person(), select_individual(), select_person_role(), store_individuals()

`individual2df` *Convert person object in a data.frame.*

Description

Results in a data.frame with the given name, family name, e-mail, ORCID, affiliation and role. Missing elements result in an empty string (""). Persons with multiple roles will have the roles as a comma separated string.

Usage

```
individual2df(person)
```

Arguments

person	The person object or a list of person objects, NA or NULL. Any "character" is converted to a person object using as.person() with a warning.

See Also

Other individual: add_individual(), has_person_role(), individual2badge(), individual2person(), select_individual(), select_person_role(), store_individuals()

`individual2person` *Select the individual information to use as a* `person` *object*

Description

This function retrieves the individual information using select_individual() and converts it to a person object. If the individual argument is not provided, it will prompt the user to select an individual. The person object will include the given name, family name, email, ORCID, and affiliation (if available) of the selected individual.

Usage

```
individual2person(individual, role = "aut", lang)
```

Arguments

individual	An optional individual object to convert to a person object. If not provided, the function will prompt the user to select an individual.
role	The role to use for the person object. Defaults to ‘"aut"‘ (author).
lang	The language to use for the affiliation.

See Also

Other individual: add_individual(), has_person_role(), individual2badge(), individual2df(), select_individual(), select_person_role(), store_individuals()

`is_repository` *Determine if a directory is in a git repository*

Description

The path arguments specifies the directory at which to start the search for a git repository. If it is not a git repository itself, then its parent directory is consulted, then the parent's parent, and so on.

Usage

```
is_repository(path = ".")
```

Arguments

path	the location of the git repository, see details.

Value

TRUE if directory is in a git repository else FALSE

See Also

Other git: is_tracked_not_modified(), ssh_http()

`is_tracked_not_modified` *Check if a file is tracked and not modified*

Description

Check if a file is tracked and not modified

Usage

```
is_tracked_not_modified(file, repo = ".")
```

Arguments

file	path relative to the git root directory.
repo	path to the repository

See Also

Other git: is_repository(), ssh_http()

license_local_remote *Create a data frame mapping local license file names to remote license URLs*

Description

This function creates a data frame that maps local license file names to their corresponding remote license URLs. The local file names are generated by converting the license names to lowercase, replacing spaces and hyphens with underscores, and appending the `.md` extension. The remote file URLs are taken directly from the input `license` vector.

Usage

```
license_local_remote(license)
```

Arguments

`license`	A named character vector where the names are the license names and the values are the corresponding remote license URLs.

Value

A data frame with two columns: `local_file` and `remote_file`. The `local_file` column contains the generated local file names, and the `remote_file` column contains the corresponding remote license URLs.

menu_first *Improved version of* `utils::menu()`

Description

This function is a wrapper around `utils::menu()` that returns the index of the first choice instead of the index of the selected choice. This is useful when you want to ask a question with only one choice and you want to return a logical value.

Usage

```
menu_first(choices, graphics = FALSE, title = NULL)
```

Arguments

`choices`	a character vector of choices
`graphics`	a logical indicating whether a graphics menu should be used if available.
`title`	a character string to be used as the title of the menu. `NULL` is also accepted.

See Also

Other question: `ask_email()`, `ask_language()`, `ask_new_license()`, `ask_orcid()`, `ask_ror()`, `ask_url()`, `ask_yes_no()`

`new_org_item` *Interactively create a new* `org_item`

Description

Interactively create a new `org_item`

Usage

```
new_org_item(languages, licenses)
```

Arguments

`languages`	A character vector of language codes.
`licenses`	A list of license items.

Value

An `org_item` object.

See Also

Other organisation: `cache_org()`, `get_available_organisations()`, `get_default_org_list()`, `inbo_org_list()`, `new_org_list()`, `org_list_from_url()`, `organisation2df()`, `select_license()`, `store_organisations()`, `stored_organisations()`

`new_org_list` *Interactively create a new organisation list*

Description

An interactive alternative for `org_list$new()`. Reuses available organisations where possible.

Usage

```
new_org_list(git)
```

Arguments

`git`	An optional string with the absolute path to a git organisation. E.g. `"https://github.com/inbo"`

See Also

org_list, org_item

Other organisation: cache_org(), get_available_organisations(), get_default_org_list(), inbo_org_list(), new_org_item(), org_list_from_url(), organisation2df(), select_license(), store_organisations(), stored_organisations()

---

org_item *The* org_item *R6 class*

Description

A class containing a single organisation

Active bindings

as_list The organisation as a list.

get_zenodo The organisation Zenodo community.

get_default_name The organisation default name.

get_email The organisation email.

get_funder The funder rules.

get_publisher The publisher rules.

get_name The organisation names.

get_orcid The ORCID rules.

get_rightsholder The rightsholder rules.

Methods

Public methods:

- org_item$new()
- org_item$as_person()
- org_item$compare_by_name()
- org_item$get_license()
- org_item$get_pkgdown()
- org_item$print()
- org_item$clone()

org_item$new(): Initialize a new org_item object.

Usage:

```
org_item$new(
  name,
  email,
  orcid = FALSE,
  rightsholder = c("optional", "single", "shared", "when no other"),
  funder = c("optional", "single", "shared", "when no other"),
  publisher = c("optional", "single", "shared", "when no other"),
  license = list(package = c(`GPL-3.0` =
    paste("https://raw.githubusercontent.com/inbo/citeme/refs/heads/main",
    "inst/licenses/gplv3.md", sep = "/"), MIT =
    paste("https://raw.githubusercontent.com/inbo/citeme/refs/heads/main",
    "inst/licenses/mit.md", sep = "/")), project = c(`CC BY 4.0` =
    paste("https://raw.githubusercontent.com/inbo/citeme/refs/heads/main",
    "inst/licenses/cc_by_4_0.md", sep = "/")), data = c(CC0 =
    paste("https://raw.githubusercontent.com/inbo/citeme/refs/heads/main",
    "inst/licenses/cc0.md",
     sep = "/"))),
  ror = "",
  zenodo = "",
  website = "",
  logo = ""
)
```

Arguments:

`name` A named vector with the organisation names in one or more languages. The first item in the vector is the default language. The names of the vector must match the language code.

`email` An email address for the organisation. Used to contact the organisation. And used to detect if a person is affiliated with the organisation.

`orcid` Whether the organisation requires an ORCID for every person that uses this organisation as affiliation.

`rightsholder` The required copyright holder status for the organisation. `"optional"` means that the organisation is not required as the copyright holder. `"single"` means that the organisation must be the only copyright holder. `"shared"` means that the organisation must be one of the copyright holders. `"when no other"` means that if no other copyright holder is specified, the organisation must be the copyright holder.

`funder` The required funder status for the organisation. The categories are the same as for `rightsholder`.

`publisher` The required publisher status for the organisation. The categories are the same as for `rightsholder`.

`license` A list with the allowed licenses by the organisation. The list may contain the following items: `package`, `project`, and `data`. Every item must be a named character vector with the allowed licenses. The names must match the license name. The values must either match the path to a license template in the `checklist` package or an absolute URL to publicly available markdown file with the license text. Use `character(0)` to indicate that the organisation does not require a specific license for that item. `package` defaults to `c("GPL-3.0", "MIT")`. `project` defaults to `"CC BY 4.0"`. `data` defaults to `"CC0 1.0"`.

`ror` The optional ROR ID of the organisation.

`zenodo` The optional Zenodo community ID of the organisation.

`website` The optional website URL of the organisation.

`logo` The optional logo URL of the organisation.

`org_item$as_person()`: as_person The organisation as a person.

Usage:

```
org_item$as_person(
  lang = names(private$name)[1],
  role = c("cph", "fnd", "pbl")
)
```

Arguments:

`lang` The language to use for the organisation name. Defaults to the first language in the `name` vector.

`role` The role of the person in the organisation.

`org_item$compare_by_name()`: Compares the number of matching words with the organisation name. Either `Inf` when there is a perfect match. Otherwise a number between 0 and 1 indicating the ratio of the matching words with the total number of words in `name`. A value of 1 means that all words in `name` are present in one of the organisation names but in a different order.

Usage:

```
org_item$compare_by_name(name)
```

Arguments:

`name` The name to match.

`org_item$get_license()`: Get the organisation license.

Usage:

```
org_item$get_license(type = c("package", "project", "data", "all"))
```

Arguments:

`type` The type of license to get. Can be one of `"package"`, `"project"`, or `"data"`. Defaults to `"package"`.

Returns: A named character vector with the allowed licenses.

`org_item$get_pkgdown()`: The pkgdown author field.

Usage:

```
org_item$get_pkgdown(lang)
```

Arguments:

`lang` The language to use for the organisation name.

`org_item$print()`: Print the `org_item` object.

Usage:

```
org_item$print()
```

`org_item$clone()`: The objects of this class are cloneable with this method.

Usage:

```
org_item$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other class: citation_meta, org_list

`org_list` *The* `org_list` *R6 class*

Description

A class containing a list of organisations

Active bindings

`as_list` The list of `org_item` objects.

`get_default_name` The organisations default name.

`get_default_funder` The default funder.

`get_default_publisher` The default publisher.

`get_default_rightsholder` The default rightsholder.

`get_email` The organisations email.

`get_git` The git organisation URL.

`get_listed_licenses` Return the available licenses.

`get_languages` The different languages of the organisations.

`which_funder` The required funders.

`which_publisher` The required publishers.

`which_rightsholder` The required rightsholders.

Methods

Public methods:

- `org_list$add_item()`
- `org_list$check()`
- `org_list$get_allowed_licenses()`
- `org_list$get_person()`
- `org_list$get_zenodo_by_email()`
- `org_list$get_match()`
- `org_list$get_name_by_domain()`
- `org_list$get_pkgdown()`
- `org_list$new()`
- `org_list$print()`
- `org_list$read()`
- `org_list$validate_person()`
- `org_list$validate_rules()`

- org_list$write()
- org_list$clone()

`org_list$add_item()`: Add one or more `org_item` objects to the list.

Usage:

```
org_list$add_item(...)
```

Arguments:

`...` One or more `org_item` objects.

`org_list$check()`: Check if the organisation list is compatible with the default as set in the organisations git repository.

Usage:

```
org_list$check(x = ".")
```

Arguments:

`x` The path to the project directory

`org_list$get_allowed_licenses()`: Return the allowed licenses.

Usage:

```
org_list$get_allowed_licenses(
  email = character(0),
  type = c("package", "project", "data", "all")
)
```

Arguments:

`email` The email addresses of the organisations. Returns all available licenses if missing.

`type` The type of license to return. Can be one of `"package"`, `"project"`, `"data"` or `"all"`.

`org_list$get_person()`: Return the organisation with matching email as a `person()`.

Usage:

```
org_list$get_person(email, role = c("cph", "fnd"), lang)
```

Arguments:

`email` The email address of the organisation.

`role` The role of the person to return.

`lang` The language to return the organisation name in.

`org_list$get_zenodo_by_email()`: Return a vector of Zenodo communities associated with the organisations with matching email.

Usage:

```
org_list$get_zenodo_by_email(email)
```

Arguments:

`email` The email addresses to match against.

Returns: A character vector with the communities.

`org_list$get_match()`: Return the organisation with a matching name.

Usage:

```
org_list$get_match(name)
```

Arguments:

`name` The name of the organisation to match.

Returns: A list with the organisation name, email and match ratio.

`org_list$get_name_by_domain()`: Return the organisation names with a matching email domain.

Usage:

```
org_list$get_name_by_domain(email, lang)
```

Arguments:

`email` The email address to match the domain against.

`lang` The language to return the organisation name in.

Details: The function extracts the domain from the email address and matches it against the organisation email addresses. If multiple organisations have the same domain, the function returns all matching names. If the language is not available for a specific organisation, it will return the first available name.

Returns: A character vector with the organisation names.

`org_list$get_pkgdown()`: get_pkgdown The pkgdown author field

Usage:

```
org_list$get_pkgdown(lang)
```

Arguments:

`lang` The language to use for affiliation.

`org_list$new()`: Initialize a new `org_list` object.

Usage:

```
org_list$new(..., git = character(0))
```

Arguments:

`...` One or more `org_item` objects.

`git` An optional string with the absolute path to a git organisation. E.g. `"https://github.com/inbo"`

`org_list$print()`: Print the `org_list` object.

Usage:

```
org_list$print()
```

`org_list$read()`: Read the `org_list` object from an `organisation.yml` file.

Usage:

```
org_list$read(x = ".")
```

Arguments:

`x` The path to the directory where the `organisation.yml` file is stored.

`org_list$validate_person()`: Validate a `person` object given the `org_list` object.

Usage:

```
org_list$validate_person(person, lang)
```

Arguments:

`person` The `person` object to validate.

`lang` The language to use for affiliation.

`org_list$validate_rules()`: Validate the rules for the rightsholder, funder and publisher.

Usage:

```
org_list$validate_rules(
  rightsholder = person(),
  funder = person(),
  publisher = person()
)
```

Arguments:

`rightsholder` The rightsholders as a `person` object.

`funder` The funders as a `person` object.

`publisher` The publishers as a `person` object.

`org_list$write()`: Write the `org_list` object to an `organisation.yml` file.

Usage:

```
org_list$write(x = ".", license = FALSE)
```

Arguments:

`x` The path to the directory where the `organisation.yml` file should be written.

`license` Whether to include license information.

Returns: The path to the written `organisation.yml` file.

`org_list$clone()`: The objects of this class are cloneable with this method.

Usage:

```
org_list$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other class: `citation_meta`, `org_item`

`org_list_from_url` *Get the default organisation list from a git URL This function retrieves the default organisation list from a git URL. It checks if the organisation list is already cached in the user's R configuration. If it is, it returns the cached version. If not, it attempts to retrieve the organisation list from the specified git URL and caches it for future use.*

Description

Get the default organisation list from a git URL This function retrieves the default organisation list from a git URL. It checks if the organisation list is already cached in the user's R configuration. If it is, it returns the cached version. If not, it attempts to retrieve the organisation list from the specified git URL and caches it for future use.

Usage

```
org_list_from_url(git)
```

Arguments

`git`	The git URL to retrieve the organisation list from.

Value

An `org_list` object containing the organisation list.

See Also

Other organisation: `cache_org()`, `get_available_organisations()`, `get_default_org_list()`, `inbo_org_list()`, `new_org_item()`, `new_org_list()`, `organisation2df()`, `select_license()`, `store_organisations()`, `stored_organisations()`

`organisation2df` *Convert organisation object to a data.frame*

Description

Results in a data.frame with the email, default name, ROR, ORCID requirement, Zenodo community, website, and logo of the organisation.

Usage

```
organisation2df(x)
```

Arguments

`x`	An `org_list` or `org_item` object.

Value

A data.frame with the organisation information.

See Also

Other organisation: cache_org(), get_available_organisations(), get_default_org_list(), inbo_org_list(), new_org_item(), new_org_list(), org_list_from_url(), select_license(), store_organisations(), stored_organisations()

---

select_individual *Select an individual from the local database or add a new individual.*

Description

Reuse existing individual information or add a new individual. Allows to update existing individual information.

Usage

```
select_individual(email, lang)
```

Arguments

email	An optional email address. When given and it matches with a single person, the function immediately returns the information of that person.
lang	The language to use for the affiliation. Defaults to the first language in the name vector of the org_list object. When the affiliation is not available in that language, it will use the first available language.

Value

A data.frame with individual information.

See Also

Other individual: add_individual(), has_person_role(), individual2badge(), individual2df(), individual2person(), select_person_role(), store_individuals()

select_license *Select a license for a project, package or dataset*

Description

This function allows you to select a license for a project, package or dataset from the list of allowed licenses for the organisation. If there is only one license available, it will be selected automatically.

Usage

```
select_license(org, type = c("package", "project", "data"))
```

Arguments

org	An object of class org_list created with new_org_list() or org_list$new().
type	The type of license to select. One of "package", "project", or "data". Default is "package".

Value

The name of the selected license.

See Also

Other organisation: cache_org(), get_available_organisations(), get_default_org_list(), inbo_org_list(), new_org_item(), new_org_list(), org_list_from_url(), organisation2df(), store_organisations(), stored_organisations()

select_person_role *Select the person object with a given role*

Description

Select the person object with a given role

Usage

```
select_person_role(individual, role)
```

Arguments

individual	A person object.
role	A character vector of roles to check for.

Value

A logical vector indicating whether the individual has any of the specified roles.

See Also

Other individual: `add_individual()`, `has_person_role()`, `individual2badge()`, `individual2df()`, `individual2person()`, `select_individual()`, `store_individuals()`

---

ssh_http *Convert SSH URL to HTTP URL*

Description

This function converts a git SSH URL to an HTTP URL. It also removes any `OAuth2` tokens from the URL. The resulting URL is used to determine the root URL of the organisation for retrieving the organisation list.

Usage

```
ssh_http(url)
```

Arguments

`url`	The git URL to convert.

Value

The converted HTTP URL.

See Also

Other git: `is_repository()`, `is_tracked_not_modified()`

---

store_individuals *Store individual details for later usage*

Description

Store individual details for later usage

Usage

```
store_individuals(x = ".")
```

Arguments

`x`	Path to a project

See Also

Other individual: add_individual(), has_person_role(), individual2badge(), individual2df(), individual2person(), select_individual(), select_person_role()

---

store_organisations *Store organisation details for later usage*

Description

This function stores organisation information locally for later retrieval. The organisations are stored in a tab-separated file in the user's data directory.

Usage

```
store_organisations(x)
```

Arguments

x	An org_list or org_item object to store.

Value

NULL invisibly.

See Also

Other organisation: cache_org(), get_available_organisations(), get_default_org_list(), inbo_org_list(), new_org_item(), new_org_list(), org_list_from_url(), organisation2df(), select_license(), stored_organisations()

---

stored_organisations *Retrieve stored organisation information*

Description

This function retrieves organisation information that was previously stored using store_organisations().

Usage

```
stored_organisations()
```

Value

A data.frame with the stored organisation information. The data.frame contains the columns: email, default_name, names, ror, orcid, zenodo, website, and logo. Returns an empty data.frame if no organisations are stored.

See Also

Other organisation: cache_org(), get_available_organisations(), get_default_org_list(), inbo_org_list(), new_org_item(), new_org_list(), org_list_from_url(), organisation2df(), select_license(), store_organisations()

`validate_citation` *Validate a citation metadata object*

Description

Validate a citation metadata object

Usage

```
validate_citation(meta)
```

Arguments

`meta`	A `citation_meta` object.

Value

A character vector of validation errors, or an empty character vector if the metadata is valid.

See Also

Other validation: validate_email(), validate_language(), validate_license_list(), validate_orcid(), validate_ror(), validate_url()

`validate_email` *Check if a vector contains valid email*

Description

It only checks the format of the text, not if the email address exists.

Usage

```
validate_email(email)
```

Arguments

`email`	A vector with email addresses.

Value

A logical vector.

See Also

Other validation: validate_citation(), validate_language(), validate_license_list(), validate_orcid(), validate_ror(), validate_url()

---

validate_language *Validate language code*

Description

A valid language code is a string in the format of xx-YY, where xx is a two-letter lowercase language code and YY is a two-letter uppercase country code. For example, en-GB for English (United Kingdom) and nl-BE for Dutch (Belgium).

Usage

```
validate_language(language)
```

Arguments

language	A string to validate

Value

The input language code if it is valid, otherwise an error message.

See Also

Other validation: validate_citation(), validate_email(), validate_license_list(), validate_orcid(), validate_ror(), validate_url()

---

validate_license_list *Validate a license list*

Description

Validate a license list

Usage

```
validate_license_list(license)
```

Arguments

license	A list with three named character vectors: package, project, and data. Each vector should contain unique license names as values and unique package, project, or data names as names.

Value

Invisibly returns NULL if the license list is valid, otherwise throws an error.

See Also

Other validation: validate_citation(), validate_email(), validate_language(), validate_orcid(), validate_ror(), validate_url()

---

validate_orcid *Validate the structure of an* ORCID iD

Description

The ORCID iD is a unique, persistent identifier free of charge to researchers. Checks whether the ORCID iD has the proper format and a correct checksum. The ORCID iD must be in the format 0000-0000-0000-0000 where the last digit can be a number or "X". Empty strings are considered valid to allow optional ORCID iD.

Usage

```
validate_orcid(orcid)
```

Arguments

orcid	A vector of ORCID

Value

A logical vector with the same length as the input vector.

See Also

Other validation: validate_citation(), validate_email(), validate_language(), validate_license_list(), validate_ror(), validate_url()

validate_ror *Validate ROR*

Description

The Research Organization Registry (ROR) is a global, community-led registry of open persistent identifiers for research organizations. ROR makes it easy for anyone or any system to disambiguate institution names and connect research organizations to researchers and research outputs. Validate that a ROR is a string of 9 characters starting with 0, followed by 6 characters which can be a letter (except i, l, o) or a digit, and ending with 2 digits.

Usage

```
validate_ror(ror)
```

Arguments

ror	A ROR to validate.

Value

A logical value indicating whether the ROR is valid.

See Also

Other validation: validate_citation(), validate_email(), validate_language(), validate_license_list(), validate_orcid(), validate_url()

validate_url *Validate URL*

Description

Validate that a URL is a string of length 1 which is not NA and matches the pattern of a valid URL.

Usage

```
validate_url(url)
```

Arguments

url	A URL to validate.

Value

A logical value indicating whether the URL is valid.

See Also

Other validation: validate_citation(), validate_email(), validate_language(), validate_license_list(), validate_orcid(), validate_ror()

Index