Package: protocolhelper (via r-universe)

September 5, 2024

Title Helper Functions to Manage Protocols

Version 0.6.1

Description Helper functions to manage INBO protocols.

License GPL-3

URL https://inbo.github.io/protocolhelper/, https://github.com/inbo/protocolhelper

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

Depends R (>= 3.6)

Imports assertthat, bookdown, checklist (>= 0.2.4), commonmark, fs, gert, jsonlite, knitr, purrr, R6, reactable, rmarkdown, rprojroot, slickR, stringr, xfun, xml2, yaml, ymlthis

Suggests dplyr, lubridate, magick, pander, png, testthat (>= 2.1.0), tinytex

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

Config/checklist/communities inbo

Config/checklist/keywords protocols; R package; Reproducible research; Rmarkdown templates

Encoding UTF-8

Language en-GB

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

SystemRequirements Pandoc (>= 2.0.0)

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

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

RemoteRef HEAD

RemoteSha 2772aa23f88dd5f89b91b1846a9560c3e5a936b4

Contents

`add_captions` *Touch up figure and table captions after using* `convert_docx_to_rmd()`

Description

After using `convert_docx_to_rmd()`, figure and table captions are part of the main text and cross references to them are just text and numbers, making it difficult to integrate these `.Rmd` files in reports: they are not automatically renumbered, these captions cannot be listed,... This function replaces the figure and table descriptions with true captions, and cross references to these figures and tables with dynamic references (that refer to the figure or table label).

Usage

```
add_captions(
  from,
  to,
  name_figure_from = "Figuur",
  name_table_from = "Tabel",
  name_figure_to = "Figuur",
```

```
  name_table_to = "Tabel"
)
```

Arguments

from	The .Rmd file to convert. Can be given as an absolute or relative path.
to	The filename to write the resulting .Rmd file. Can be given as an absolute or relative path.
name_figure_from	name that is given to figures in captions and cross references in the .Rmd provided in from (default is Figuur)
name_table_from	name that is given to tables in captions and cross references in the .Rmd provided in from (default is Tabel)
name_figure_to	name that should be given to figures in cross references in the output .Rmd (default is Figuur)
name_table_to	name that should be given to tables in cross references in the output .Rmd (default is Tabel)

Details

This function expects an input that is generated by convert_docx_to_rmd(), with

- figure captions below the figure and starting with Figuur (or other name to be defined in variable name_figure), then a unique number and finally the description in one paragraph
- table captions above the table and starting with Tabel (or other name to be defined in variable name_table), then a unique number and finally the description in one paragraph
- figures consisting of one image, not 2 figures near each other
- cross references having the same label as the figure or table they refer to, default Figuur or Tabel followed by the unique number written in exactly the same way

Please check carefully for mistakes after using this function

See Also

Other convert: convert_docx_to_rmd()

---

add_dependencies *Adds dependencies to the YAML of an* index.Rmd *file*

Description

Adds dependencies to the YAML of an index.Rmd file

Usage

```
add_dependencies(
  code_mainprotocol,
  protocol_code,
  version_number,
  params,
  appendix = !is.na(params)
)
```

Arguments

code_mainprotocol	Protocol code of the protocol for which dependencies need to be declared in the YAML of its index.Rmd file
protocol_code	Character vector of protocol codes that are dependencies to the main protocol.
version_number	Character vector of version numbers corresponding with protocol_code.
params	List of lists with protocol-specific parameters corresponding with parameters from the protocols in protocol_code. Use NA if no parameters should be set for a protocol.
appendix	Logical vector indicating whether or not a dependency needs to be included as a subprotocol (at the end of the main protocol in an appendix). Default is !is.na(params). When params is not NA, appendix will always be set to TRUE even if the user passes another value.

See Also

Other creation: add_one_subprotocol(), add_subprotocols(), create_protocol(), insert_protocolsection(), update_protocol(), update_version_number()

Examples

```
## Not run:
protocolhelper::add_dependencies(
  code_mainprotocol = "spp-999-en",
  protocol_code = c("sfp-123-en", "spp-124-en"),
  version_number = c("2020.01", "2020.02"),
  params = list(NA, list(width = 8, height = 8))
  )

## End(Not run)
```

add_label *Helper function to create labelled captions for* pander *tables*

Description

The function adds a reference label to the caption so that pander tables can be cross-referenced in bookdown using the `\@ref(tab:label)` syntax. The function should only be used in `pander(x, caption = add_label())`.

Usage

```
add_label(caption = "", tag = "tab")
```

Arguments

caption	The caption text as a string
tag	The tag to use as a prefix. Default is tab.

Value

The caption text prefixed with a reference label.

See Also

Other utility: `get_path_to_protocol()`, `get_protocol_type()`, `get_protocolnumbers()`, `get_short_titles()`, `get_version_number()`, `increment_version_number()`

Examples

```
add_label("caption text")
```

add_one_subprotocol *Helper function to add one sub-protocol to a project-specific protocol of which it is a dependency*

Description

The function renders the sub-protocol to `bookdown::markdown_document2()` and saves the resulting md file (and any associated media and data files) in a subfolder of the directory of the project-specific protocol. This function should normally not be called directly. Use `add_subprotocols()` instead.

Usage

```
add_one_subprotocol(
  code_subprotocol,
  version_number,
  params2 = NULL,
  code_mainprotocol,
  fetch_remote = TRUE
)
```

Arguments

code_subprotocol	Character string giving the protocol code from which a sub-protocol will be made (usually a sfp-type protocol)
version_number	Character string with format YYYY.NN
params2	A list of parameter key-value pairs.
code_mainprotocol	Character string giving the protocol code for the main protocol
fetch_remote	Whether or not to fetch the remote. Default TRUE.

See Also

Other creation: add_dependencies(), add_subprotocols(), create_protocol(), insert_protocolsection(), update_protocol(), update_version_number()

Examples

```
## Not run:
  add_subprotocols(code_mainprotocol = 'spp-999-en')

## End(Not run)
```

---

add_subprotocols *Render all sub-protocols belonging to a main protocol to single markdown files*

Description

The function should be called interactively (in the console) after the dependencies section in the YAML header of the index.Rmd file of the main protocol has been filled in with the aid of the protocolhelper::add_dependencies() function. For reproducibility, it is good practice to save the call in a separate R script. For each sub-protocol a single markdown file and associated media and data files will be written. Each sub-protocol will be written to a subfolder of the main protocol. The subfolder name is the same as the version number of the sub-protocol.

Usage

```
add_subprotocols(code_mainprotocol, fetch_remote = TRUE)
```

Arguments

code_mainprotocol	Character string giving the protocol code for the main protocol
fetch_remote	Whether or not to fetch the remote. Default TRUE.

See Also

Other creation: add_dependencies(), add_one_subprotocol(), create_protocol(), insert_protocolsection(), update_protocol(), update_version_number()

---

check_all *Check protocol* frontmatter *and* structure

Description

Combines check_frontmatter() and check_structure() in one function.

Usage

```
check_all(protocol_code, fail = !interactive())
```

Arguments

protocol_code	Character string giving the protocol code
fail	Should the function drop an error in case of a problem? Defaults to TRUE in a non-interactive session and FALSE in an interactive session.

Value

A report of all failed checks.

See Also

Other check: check_all_author_info(), check_frontmatter(), check_structure(), protocolcheck, validate_orcid()

check_all_author_info *Helper function to check if author information is correct*

Description

Checks the format of author names and `orcid` ids.

Usage

```
check_all_author_info(author_list, problems_vect)
```

Arguments

`author_list`	the yaml front matter part containing author info
`problems_vect`	character vector of previously encountered problems

Value

a character vector of previously encountered problems and problems identified for author names and `orcid` ids.

See Also

Other check: `check_all()`, `check_frontmatter()`, `check_structure()`, `protocolcheck`, `validate_orcid()`

check_frontmatter *Checks protocol metadata*

Description

This function reads metadata from the yaml front matter stored in the `index.Rmd` file of a protocol and checks if the metadata format is correct. This function is intended for checking if a protocol is ready to be rendered and published (for instance, it will fail if version number is `YYYY.NN.dev`).

Usage

```
check_frontmatter(protocol_code, fail = !interactive())
```

Arguments

`protocol_code`	Character string giving the protocol code
`fail`	Should the function drop an error in case of a problem? Defaults to `TRUE` in a non-interactive session and `FALSE` in an interactive session.

Value

A report of all failed checks.

See Also

Other check: check_all(), check_all_author_info(), check_structure(), protocolcheck, validate_orcid()

`check_structure` *Checks protocol document structure*

Description

This function reads the protocol and checks if the document structure is correct: chunks have head and tail, required headings are present and in the right order,... This function is intended for checking if a protocol is ready to be rendered and published.

Usage

```
check_structure(protocol_code, fail = !interactive())
```

Arguments

`protocol_code`	Character string giving the protocol code
`fail`	Should the function drop an error in case of a problem? Defaults to `TRUE` in a non-interactive session and `FALSE` in an interactive session.

Value

A report of all failed checks.

See Also

Other check: check_all(), check_all_author_info(), check_frontmatter(), protocolcheck, validate_orcid()

`convert_docx_to_rmd` *Convert a docx-file (a protocol) to an (R)markdown file*

Description

This function is derived from the `redoc::dedoc()` function and uses `pandoc` to convert between docx and markdown. Several options are preset to end-up with a markdown document that is in syntax as close as possible to Rmarkdown files in RStudio. During conversion, graphics (e.g. png, jpg) will be extracted from the docx archive and placed in a folder `./media` and named `image1`, `image2`, etcetera. Additionally, .emf files will be converted to .png.

Usage

```
convert_docx_to_rmd(
  from,
  to = sub("docx$", "Rmd", from),
  dir_media = ".",
  wrap = NA,
  overwrite = FALSE,
  verbose = FALSE,
  wd = getwd()
)
```

Arguments

`from`	The `.docx` file to convert. Can be given as an absolute or relative path.
`to`	The filename including path to write the resulting `.Rmd` file. The default is to use the same name and path as the `.docx` document.
`dir_media`	The directory to write the folder `media` with images to, relative to the folder where the `.Rmd` is written. Defaults to '.' (the path where the `.Rmd` file is written).
`wrap`	The width at which to wrap text. If `NA` (default), text is not wrapped.
`overwrite`	Whether or not to overwrite the `to` file if it already existed. Defaults to `FALSE`.
`verbose`	Whether to print `pandoc` progress text. Defaults to `FALSE`.
`wd`	Current working directory (used to handle relative paths).

Details

Metadata in the page headers and footers of the docx are ignored and will thus be lost during conversion. In case the header or footer did contain important metadata, it will need to be recovered manually. Usually header information will go inside a yaml section of an Rmarkdown.

See Also

Other convert: `add_captions()`

`create_protocol` *Create a folder with a bookdown (R markdown) template to start a new protocol and optionally render to html*

Description

This function will create a new folder based on values that are passed on via the parameters and creates a R-markdown (bookdown) skeleton based on a template file to start working on a new protocol. Optionally, the rmarkdown chapters are rendered to an html file which will be saved in a matching subfolder of the `docs` folder.

Usage

```
create_protocol(
  protocol_type = c("sfp", "spp", "sap", "sop", "sip"),
  title,
  short_title,
  authors,
  orcids,
  date = Sys.Date(),
  reviewers,
  file_manager,
  version_number = get_version_number(),
  theme = NULL,
  project_name = NULL,
  language = c("nl", "en"),
  subtitle = NULL,
  from_docx = NULL,
  protocol_number = NULL,
  template = protocol_type,
  render = FALSE
)

create_sfp(
  title,
  subtitle = NULL,
  short_title,
  authors,
  orcids,
  date = Sys.Date(),
  reviewers,
  file_manager,
  version_number = get_version_number(),
  theme = c("generic", "water", "air", "soil", "vegetation", "species"),
  language = c("nl", "en"),
  from_docx = NULL,
  protocol_number = NULL,
```

```
  template = c("sfp", "generic"),
  render = FALSE
)

create_spp(
  title,
  subtitle = NULL,
  short_title,
  authors,
  orcids,
  date = Sys.Date(),
  reviewers,
  file_manager,
  version_number = get_version_number(),
  project_name,
  language = c("nl", "en"),
  from_docx = NULL,
  protocol_number = NULL,
  template = c("spp"),
  render = FALSE
)

create_sap(
  title,
  subtitle = NULL,
  short_title,
  authors,
  orcids,
  date = Sys.Date(),
  reviewers,
  file_manager,
  version_number = get_version_number(),
  language = c("nl", "en"),
  from_docx = NULL,
  protocol_number = NULL,
  template = c("sap", "generic"),
  render = FALSE
)

create_sip(
  title,
  subtitle = NULL,
  short_title,
  authors,
  orcids,
  date = Sys.Date(),
  reviewers,
  file_manager,
```

```
  version_number = get_version_number(),
  language = c("nl", "en"),
  from_docx = NULL,
  protocol_number = NULL,
  template = c("sip", "generic"),
  render = FALSE
)

create_sop(
  title,
  subtitle = NULL,
  short_title,
  authors,
  orcids,
  date = Sys.Date(),
  reviewers,
  file_manager,
  version_number = get_version_number(),
  language = c("nl", "en"),
  from_docx = NULL,
  protocol_number = NULL,
  template = c("sop", "generic"),
  render = FALSE
)
```

Arguments

protocol_type	Either sfp (standard field protocol), spp (standard project protocol), sap (standard analytical protocol), sip (standard instrument protocol), sop (standard operating protocol)
title	A character string giving the main title of the protocol
short_title	A character string of less than 20 characters to use in folder and file names
authors	A character vector for authors of the form c("lastname1, firstname1", "lastname2, firstname2")
orcids	A character vector of orcid IDs, equal in length to authors. If one of the authors does not have an orcid ID, use NA to indicate this in the corresponding position of the character vector (or get an orcid ID).
date	A character string of the date in ISO 8601 format (YYYY-MM-DD)
reviewers	A character vector for reviewers of the form First name Last name
file_manager	A character string for the name of the document maintainer of the form First name Last name
version_number	A version number of the form YYYY.##. The default is a function which will determine this number automatically. It should normally not be changed.
theme	A character string equal to one of "generic" (default), "water", "air", "soil", "vegetation" or "species". It is used as the folder location (source/sfp/theme) where standard field protocols that belong to the same theme will be stored. Ignored if protocol_type is other than "sfp".

project_name — A character string that is used as the folder location (source/spp/project_name) where project-specific protocols that belong to the same project will be stored. Preferably a short name or acronym. If the folder does not exist, it will be created. Ignored if protocol_type is other than "spp".

language — Language of the protocol, either "nl" (Dutch), the default, or "en" (English).

subtitle — A character string for an optional subtitle

from_docx — A character string with the path (absolute or relative) to a .docx file containing a pre-existing protocol. Please make sure to copy-paste all relevant meta-data from the .docx file to the corresponding parameters of this function. If nothing is provided (i.e. default = NULL), an empty template will be used.

protocol_number — A character string giving the protocol number. This parameter should normally not be specified (i.e. NULL), unless from_docx is specified. A protocol number is a three digit string where the first digit corresponds with a theme and the last two digits identify a protocol within a theme for standard field protocols. A protocol number for other protocol types is just a three digit string. If NULL (the default), a protocol number will be determined automatically based on pre-existing protocol numbers. Note that for backwards compatibility with protocol numbers that were already in use at INBO, we made a list of reserved numbers. These reserved numbers will not be used when protocol_number is NULL. The only time you will need to explicitly pass a protocol number to the protocol_number argument is when you want to migrate a pre-existing INBO protocol to protocolsource and hence use one of the reserved numbers. Protocol numbers that are already in use in protocolsource can be retrieved with get_protocolnumbers().

template — Which template to use? Default is set equal to protocol_type. However, you can also set this to "generic" in which case a simplified template will be used that can be used as an alternative to the default templates.

render — Whether or not to render the protocol to html. Defaults to FALSE.

Details

It is assumed that the source folder is a subfolder of an RStudio project with git version control. A target folder to which files will be written will be created as subdirectories beneath source. The subfolder structure is of the form /sfp/<theme>/<sfp>_<protocolnumber>_<language>_<short_title>/ for standard field protocols. Or /spp/<project_name>/<spp>_<protocolnumber>_<language>_<short_title>/ for standard project protocols. Or /sip/<sip>_<protocolnumber>_<language>_<short_title>/ for sips (and analogous for sop and sap). The folder names are determined by the corresponding arguments of the function. A matching subfolder structure will be created beneath the docs folder (and output files needed for rendering to html output will be placed in it if render = TRUE. The template Rmarkdown files and the Rmarkdown files that result from converting a docx protocol (see from_docx argument), will be written to the target folder beneath source. Template Rmarkdown files with the same name as Rmarkdown files that result from converting a docx protocol will be overwritten by the latter. Besides Rmarkdown files, this target folder will also contain files needed to render to a Bookdown gitbook such as a _bookdown.yml and _output.yml. The NEWS.md file must be used to document the changes between revisions of the protocol. Furthermore, a data and a media folder will be created as subdirectories of the target folder. The media folder can be used to

store image files and will contain image files extracted from the docx protocol when the from_docx argument is used. The data folder can be used to store tabular data that are needed for the protocol.

See Also

Other creation: add_dependencies(), add_one_subprotocol(), add_subprotocols(), insert_protocolsection(), update_protocol(), update_version_number()

Examples

```
## Not run:
protocolhelper::create_protocol(
  protocol_type = "sfp",
  title = "Test 1", subtitle = "subtitle", short_title = "water 1",
  authors = c("Someone, Else", "Another, One"),
  orcids = c("0000-0001-2345-6789", "0000-0002-2345-6789"),
  reviewers = "me", file_manager = "who?",
  theme = "water", language = "en")

## End(Not run)
```

---

get_path_to_protocol *Function to get (or set) the full path to a protocol*

Description

A function that is used by other functions and should normally not be used directly.

For existing protocol codes, arguments theme and project_name are always ignored. The function will return the absolute path for that protocol.

For new sfp or spp protocols, also either the theme or the project_name argument and short_title are required apart from the protocol_code. The function will construct the absolute path where the source code for that new protocol will be written.

Usage

```
get_path_to_protocol(
  protocol_code,
  theme = NULL,
  project_name = NULL,
  short_title = NULL
)
```

Arguments

protocol_code	Character string giving the protocol code
theme	A character string equal to one of "generic", "water", "air", "soil", "vegetation" or "species". Defaults to NULL. Only needed if no folder with the name of the protocol code exists and the request is for a sfp protocol.

project_name	Character string giving the name of the project folder. Defaults to NULL. Only needed if no folder with the name of the protocol code exists and the request is for a spp protocol.
short_title	A character string of less than 20 characters to use in folder and filenames. Defaults to NULL. Only needed if no folder with the name of the protocol code exists.

Value

A character vector containing the full path to the protocol.

See Also

Other utility: add_label(), get_protocol_type(), get_protocolnumbers(), get_short_titles(), get_version_number(), increment_version_number()

Examples

```
## Not run:
get_path_to_protocol(protocol_code = "sfp-401-nl")

## End(Not run)
```

---

get_protocolnumbers *Function to list all occupied protocol numbers*

Description

This function will search for protocol numbers in filenames of Rmarkdown files listed underneath the source folder. The search will be restricted to files of a given protocol type and given language.

Usage

```
get_protocolnumbers(
  protocol_type = c("sfp", "sip", "sap", "sop", "spp"),
  language = c("nl", "en")
)
```

Arguments

protocol_type	A character string equal to sfp (default), sip, sap, sop or spp.
language	Language of the protocol, either "nl" (Dutch), the default, or "en" (English).

Value

A character vector with occupied protocol numbers for a specific protocol type

See Also

Other utility: add_label(), get_path_to_protocol(), get_protocol_type(), get_short_titles(), get_version_number(), increment_version_number()

Examples

```
## Not run:
get_protocolnumbers()

## End(Not run)
```

---

get_protocol_type *Get protocol type from protocol code*

Description

The protocol type corresponds to the first 3 letters of the protocol code

Usage

```
get_protocol_type(protocol_code, labels = TRUE, auto_identifier = FALSE)
```

Arguments

protocol_code	Character vector giving the protocol code(s)
labels	Logical. If TRUE return full labels, else return just the three letter abbreviation.
auto_identifier	Logical. If TRUE returns labels following Pandoc's auto-identifier rules.

Value

A factor with 5 levels corresponding to sfp, sip, sap, sop and spp. The labels depend on auto_identifier setting.

See Also

Other utility: add_label(), get_path_to_protocol(), get_protocolnumbers(), get_short_titles(), get_version_number(), increment_version_number()

get_short_titles *Function to list all short titles that are already in use.*

Description

This function will search for short titles in filenames of Rmarkdown files listed underneath the source folder. The search will be restricted to files of a given protocol type and given language.

Usage

```
get_short_titles(
  protocol_type = c("sfp", "sip", "sap", "sop", "spp"),
  language = c("nl", "en")
)
```

Arguments

protocol_type	A character string equal to sfp (default), sip, sap, sop or spp.
language	Language of the protocol, either "nl" (Dutch), the default, or "en" (English).

Value

A character vector with short titles that are in use for a given protocol type.

See Also

Other utility: add_label(), get_path_to_protocol(), get_protocol_type(), get_protocolnumbers(), get_version_number(), increment_version_number()

Examples

```
## Not run:
get_short_titles()

## End(Not run)
```

get_version_number *Get version number for a protocol*

Description

Looks up pre-existing version numbers of protocols in the main branch and calculates an incremented (next) version number for the currently checkout branch containing the created/in development/updated/ready to be released protocol. Your local main branch needs to up to date (aligned with remote) for this. If this is not the case, or other issues are detected regarding a non-clean local git repository - informative error messages will be given.

Usage

```
get_version_number(path = ".")
```

Arguments

path	Defaults to current working directory. This should correspond with the root directory of the protocolsource repo.

Value

A string containing the next (incremented) version number

See Also

Other utility: add_label(), get_path_to_protocol(), get_protocol_type(), get_protocolnumbers(), get_short_titles(), increment_version_number()

---

increment_version_number

Increment version number

Description

Given a set of published protocol version numbers, calculate the next version number

Usage

```
increment_version_number(versions)
```

Arguments

versions	a character vector with previously published version numbers

Value

A string containing the next (incremented) version number

See Also

Other utility: add_label(), get_path_to_protocol(), get_protocol_type(), get_protocolnumbers(), get_short_titles(), get_version_number()

`insert_protocolsection` *Function to add a chapter or a section of a published protocol for re-use in another protocol*

Description

The idea is to execute this function in an R chunk with knitr option `results="asis"`.

Usage

```
insert_protocolsection(
  code_subprotocol,
  version_number,
  file_name,
  section = NULL,
  demote_header = c(0, 1, 2, -1),
  fetch_remote = TRUE
)
```

Arguments

`code_subprotocol`	Character string giving the protocol code from which a sub-protocol will be made (usually a `sfp`-type protocol)
`version_number`	Character string with format `YYYY.NN`
`file_name`	Character string with the name of the Rmarkdown file (a chapter starting with a level 1 heading).
`section`	Optional character string with the name of a section within an Rmarkdown file. Can also be a unique substring of a section title. If not specified (the default): the whole file is taken. It is assumed that the section has a level 2 heading.
`demote_header`	Number of '#' to prefix to all titles before inserting in current protocol. Default is 0. A negative value can be used to remove '#' from all section titles. Allowed values are visible in the usage section.
`fetch_remote`	Whether or not to fetch the remote. Default TRUE.

Value

The function will return Rmarkdown

See Also

Other creation: `add_dependencies()`, `add_one_subprotocol()`, `add_subprotocols()`, `create_protocol()`, `update_protocol()`, `update_version_number()`

Examples

```
## Not run:
insert_protocolsection(
  code_subprotocol = "sfp-401-nl",
  version_number = "2021.01",
  file_name = "07_stappenplan.Rmd"
)

## End(Not run)
```

`protocolcheck` *The* `protocolcheck` *R6 class*

Description

A class that collects and shows all check results.

Public fields

`protocol_code` Character string giving the protocol code.

`path` Character string giving the relative path to the protocol.

`error` Character vector containing all errors found in the protocol

Methods

Public methods:

- `protocolcheck$new()`
- `protocolcheck$add_error()`
- `protocolcheck$check()`
- `protocolcheck$clone()`

Method `new()`**:** Create a new `Protocolcheck` object.

Usage:

```
protocolcheck$new(protocol_code)
```

Arguments:

`protocol_code` Character string giving the protocol code.

Returns: A new `Protocolcheck` object

Method `add_error()`**:** Add a new error to the `Protocolcheck` object.

Usage:

```
protocolcheck$add_error(msg)
```

Arguments:

`msg` Error message to be added.

Method `check()`**:** Give error report from `Protocolcheck` object.

Usage:

```
protocolcheck$check(fail)
```

Arguments:

`fail` Should an error be dropped if the report contains errors?

Returns: An error report (and if desired an error is dropped).

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other check: `check_all()`, `check_all_author_info()`, `check_frontmatter()`, `check_structure()`, `validate_orcid()`

`render_protocol` *Function to render a protocol to html and pdf.*

Description

This function is a simple wrapper around `bookdown::render_book()` and can be used to render a protocol to html and pdf in order to preview updates that have been made.

Usage

```
render_protocol(protocol_code = NULL, output_dir = NULL, ...)
```

Arguments

`protocol_code`	Character string giving the protocol code
`output_dir`	The output directory. If `NULL`, a field named `output_dir` in the configuration file '`_bookdown.yml`' will be used (possibly not specified, either, in which case a directory name '`_book`' will be used).
`...`	additional parameters passed on to `bookdown::render_book()`

Details

The rendered html and pdf file and associated files needed by the html file will be put in the directory implied by the `output_dir` parameter.

Examples

```
## Not run:
render_protocol(protocol_code = "sfp_401-nl")

## End(Not run)
```

`update_protocol` *Preparatory steps to start the update of a pre-existing version of a protocol*

Description

The function creates a branch named after the protocol_code

Usage

```
update_protocol(protocol_code)
```

Arguments

`protocol_code`	Character string giving the protocol code

Value

NULL invisibly

See Also

Other creation: `add_dependencies()`, `add_one_subprotocol()`, `add_subprotocols()`, `create_protocol()`, `insert_protocolsection()`, `update_version_number()`

`update_version_number` *Updates the version number in the YAML section of a protocol* `index.Rmd` *file and optionally in protocol* `NEWS.md`

Description

Makes use of `get_version_number` to get a new version number and changes this accordingly in the YAML section of `index.Rmd` file and optionally in `NEWS.md`.

Usage

```
update_version_number(
  protocol_code,
  commit = TRUE,
  update_news = TRUE,
  path = "."
)
```

Arguments

protocol_code	The protocol_code corresponding with the name of the branch that contains the new or updated protocol.
commit	Logical. Default TRUE. Whether or not to add and commit the changes to the protocol branch
update_news	Logical. Default TRUE. Whether or not to find and replace old version number by new version number in the NEWS.md heading 2.
path	Default is current working directory. Should correspond with root directory of protocolsource repo.

Value

TRUE if version number in yaml is updated. FALSE otherwise.

See Also

Other creation: add_dependencies(), add_one_subprotocol(), add_subprotocols(), create_protocol(), insert_protocolsection(), update_protocol()

---

validate_orcid *validate an ORCID string*

Description

Generates check digit as per ISO 7064 11,2. The last character in the ORCID ID is a checksum. In accordance with ISO/IEC 7064:2003, MOD 11-2, this checksum must be "0-9" or "X", a capital letter X which represents the value 10.

Usage

```
validate_orcid(orcid)
```

Arguments

orcid	An orcid in the 0000-0000-0000-0000 format

Value

Logical. TRUE if check digit is correct.

See Also

Other check: check_all(), check_all_author_info(), check_frontmatter(), check_structure(), protocolcheck

Examples

```
validate_orcid("0000-0002-6378-6229")
```

Index