Package 'projects'

December 30, 2018

Title A Project Infrastructure for Researchers

Version 0.1.0

Description Provides a project infrastructure with a focus on manuscript creation. Creates a project folder with a single command, containing subdirectories for specific components, templates for manuscripts, and so on.

License MIT + file LICENSE

URL https://www.github.com/NikKrieger/projects

Depends R (>= 3.4.0)

Imports checkmate (>= 1.8.5), dplyr (>= 0.7.5), fs (>= 1.2.6), magrittr (>= 1.5), methods, purrr (>= 0.2.4), readr (>= 1.1.1), rlang (>= 0.3.0.1), rstudioapi (>= 0.7), stringr (>= 1.3.1), tibble (>= 1.4.2)

Suggests here, tidyverse

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Nik Krieger [aut, cre], Adam Perzynski [aut], Jarrod Dalton [aut]

Maintainer Nik Krieger <nk@case.edu>

Repository CRAN

Date/Publication 2018-12-30 17:40:03 UTC

R topics documented:

`projects-package` *projects: A project infrastructure for researchers.*

Description

The `projects` package provides a project infrastructure with a focus on manuscript creation. It creates a project folder with a single command, containing subdirectories for specific components, templates for manuscripts, and so on.

Knitting

There are several functions that require interactive user confirmation via the console. Since interactive console input is incompatible with knitting via R Markdown files, the `projects` package was coded such that user confirmation is bypassed when `isTRUE(getOption('knitr.in.progress')) == TRUE`. Therefore, all `projects` package functions are usable when knitting. **Knit with caution!**

Acknowledgements

The authors of this package acknowledge the support provided by members of the Northeast Ohio Cohort for Atherosclerotic Risk Estimation (NEOCARE) investigative team: Claudia Coulton, Douglas Gunzler, Darcy Freedman, Neal Dawson, Michael Rothberg, David Zidar, David Kaelber, Douglas Einstadter, Alex Milinovich, Monica Webb Hooper, Kristen Hassmiller-Lich, Ye Tian (Devin), Kristen Berg, and Sandy Andrukat.

Funding

This work was supported by The National Institute on Aging of the National Institutes of Health under award number R01AG055480. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

See Also

`setup_projects()` for getting started.

`affiliations` *View the* `projects()`*,* `authors()`*, and* `affiliations()` *tables*

Description

Returns a tibble of the projects/authors/affiliations, filtered and joined according to the entirely optional arguments.

Usage

```
affiliations(affiliation, authors = FALSE)

authors(author, affiliations = FALSE, projects = FALSE)

projects(project, authors = FALSE, archived = FALSE)
```

Arguments

`projects, authors, affiliations`	Logical values indicating whether or not to perform a left join with another metadata tibble. All `FALSE` by default.
`project, author, affiliation`	An optional (unique) vector of ids and/or names. Only rows matching one or more entries will be returned.
`archived`	Logical, indicating whether or not to include projects that have been archived using `archive_project()`. False by default.

Details

If one or more of the `projects`, `authors`, or `affiliations` arguments to set to `TRUE`, a `left_join` is performed, with the "left" table being the one sharing the name of the function being used. As such, rows that don't have matches in any other tables will still show up in the output, and rows that have multiple matches in other tables will yield multiple rows in the output. The "right" table's `id` column will be renamed.

Since all these functions return `tibble`s, the user can further manipulate them using `dplyr` functions like `select` and `filter`. See the last example.

Examples

```
# Included in \donttest{} to save time on example checking.

# SETUP
old_path <- Sys.getenv("PROJECTS_FOLDER_PATH")
setup_projects(path = tempdir(), .Renviron_path = fs::path_temp(".Renviron"))
new_affiliation(department_name = "Math Dept.",
                institution_name = "Springfield College",
```

```
                address = "123 College St, Springfield, AB")
new_affiliation(department_name = "Art Department",
                institution_name = "Springfield College",
                address = "321 University Boulevard, Springfield, AB",
                id = 42)
new_affiliation(department_name = "Central Intelligence Agency",
                institution_name = "United States Government",
                address = "888 Classified Dr, Washington DC")
new_affiliation(department_name = "Pyrotechnics",
                institution_name = "ACME")
new_author(given_names = "Spiro", last_name = "Agnew", degree = "LLB",
           affiliations = "Art D", id = 13)
new_author(given_names = "Plato", id = 303)
new_author(given_names = "Condoleezza", last_name = "Rice",
           affiliations = c(1, 42, "Agency", "ACME"))
new_project(title = "Test project 1", current_owner = "Plato")
new_project(title = "Test project 2", current_owner = "eezza")
new_project(title = "Test project 3", current_owner = "Plato")
new_project(title = "Fun project 4",  current_owner = "Rice")
new_project(title = "Fun project 5",  current_owner = "Rice")
################################################################################

# View entire affiliations table
affiliations()

# View affiliations table joined to authors table
# Notice that multiple rows are created for each affiliation-author combination
affiliations(authors = TRUE)

# Using dplyr functions to query the tables:

# View authors table joined to affiliations table, and filter out duplicate
# author ids, leaving one row for each author, each including the author's
# primary (i.e., first) affiliation
authors(affiliations = TRUE) %>%
  dplyr::distinct(id, .keep_all = TRUE) %>%
  dplyr::select(id, given_names, last_name, email, phone, address)

# View all projects with "Test" in their title
projects(project = "Test")

# View projects table, including only projects with Plato as current owner
projects() %>% dplyr::filter(current_owner == 303)

# Wrapped in if(interactive()) because it requires interactive console input
# and fails automated testing.
if(interactive()) {
  # Archive Fun project 5
  archive_project("Fun project 5")

  # Default behavior is to not include archived projects in projects() table
  projects("Fun")
  projects("Fun", archived = TRUE)
```

```
}

################################################################################
# CLEANUP
Sys.setenv(PROJECTS_FOLDER_PATH = old_path)
fs::file_delete(c(fs::path_temp("projects"), fs::path_temp(".Renviron")))
```

`edit_project` *Create, edit or delete projects, authors and affiliations*

Description

These functions create, edit, or delete rows in the `projects()`, `authors()`, and `affiliations()` tibbles, which are stored in the *.metadata* subdirectory of the main `projects_folder`.

Usage

```
edit_project(project, title = NA, short_title = NA, authors,
  current_owner = NA, status = NA, deadline_type = NA,
  deadline = NA, stage = c("1: design", "2: data collection",
  "3: analysis", "4: manuscript", "5: under review", "6: accepted"),
  corresp_auth = NA, creator = NA, reprint_header = TRUE,
  archived = FALSE)

edit_author(author, given_names = NA, last_name = NA, affiliations,
  title = NA, degree = NA, email = NA, phone = NA)

edit_affiliation(affiliation, department_name = NA,
  institution_name = NA, address = NA)

delete_project(project, archived = FALSE)

delete_author(author)

delete_affiliation(affiliation)

new_project(title = NA, short_title = NA, authors = NULL,
  current_owner = NA, status = "just created", deadline_type = NA,
  deadline = NA, stage = c("1: design", "2: data collection",
  "3: analysis", "4: manuscript", "5: under review", "6: accepted"),
  path = projects_folder(), make_directories = FALSE,
  corresp_auth = NA, creator = Sys.info()["user"], id = NA,
  protocol = c("01_protocol.Rmd", "STROBE_protocol.Rmd",
  "CONSORT_protocol.Rmd"), datawork = "02_datawork.Rmd",
  analysis = "03_analysis.Rmd", report = "04_report.Rmd",
  css = "style.css", Rproj = "pXXXX.Rproj", use_bib = FALSE)
```

```
new_author(given_names = NA, last_name = NA, title = NA,
  affiliations, degree = NA, email = NA, phone = NA, id = NA)

new_affiliation(department_name = NA, institution_name = NA,
  address = NA, id = NA)
```

Arguments

`project, author, affiliation`
: The `id` or unambiguous names of a project/author/affiliation to edit or delete.

`title`
: For the `\*_project()` functions, the title of the project; for `new_project()` only, the user input is coerced to title case using `tools::toTitleCase()`.

  For the `\*_author()` functions, the job title of the author.

`short_title`
: A nickname for the project. Can be used in other `projects` package functions whenever needing to specify a project.

`authors, affiliations`
: For `new_project()`/`new_author()`, a vector of `id`s or unambiguous `given_names`/`last_name` or `department_name`/`institution_name` of `authors`/`affiliations`. Order will be preserved.

  For `edit_project()`/`edit_author()`, a formula specifying `authors`/`affiliations` to add or remove from the project/author. Formulas must have no left-hand side (i.e., begin with `~`) and use `+` to add authors and `-` to remove authors. Authors may be specified by `id` or name.

  Each element must match a row in the `authors` tibble.

`status`
: A free text field, intended to communicate the most current condition the project is in.

  For `new_project()`, default is `"just created"`.

`deadline_type`
: A free text field, intended to communicate the meaning of the next field, `deadline`.

`deadline`
: A `Date` or a character string that can be coerced to a `Date`.

`stage`
: A factor with the levels `c("1: design", "2: data collection", "3: analysis", "4: manuscript" accepted")`, communicating the stage the project is in.

`corresp_auth, current_owner`
: An `id` or unambiguous `last_name`/`given_names` of one of the authors in the `authors` table.

  If `corresp_auth` is specified, all of this author's contact information will be especially included in the project's `header`.

`creator`
: The author who created the project. If it is equal to `Sys.info()["user"]` (the default value), it is kept as is. Otherwise it will be validated against the `authors()` tibble and populated with the matching author `id`.

`reprint_header`
: Logical, indicating whether or not to reprint the project `header` after editing project information.

`archived`
: Logical indicating whether or not the function should consider archived projects when determining which project the user is referring to in the `project` argument. `FALSE` by default.

See the **Details** section of archive_project() for more information on the "archived" status of a project.

given_names, last_name, department_name, institution_name
: Each a single character string. Can be used whenever needing to specify a specific author/affiliation.

degree
: A character string (preferably an abbreviation) denoting the author's academic degree(s). Will be written next to author names in the header.

email, phone
: A character string denoting the email/phone of the author. Email will be coerced to lowercase. When a project is given a corresp_auth, these items will be included in the header.

address
: A character string indicating the address of the affiliation.

path
: A character string that can be read as a file path. Can be either:

  1) the *absolute* path of the projects_folder (default)

  2) an *absolute* path pointing to a subfolder within the projects_folder

  3) a *relative* path (leading "." optional) that will be appended onto the end of the projects_folder.

  In any case, the result is that the new project folder will be a subdirectory of the main projects_folder. See also setup_projects().

make_directories
: Logical, indicating whether or not new_project() should create subdirectories specified in the path argument that do not already exist. Ignored if path is left as the default or if all directories in path already exist.

id
: An integer that will become the item's permanent identification number. Must be in the range 1-9999 or left blank. If left blank, the lowest available integer in the aforementioned range will be selected.

  For new_project, this number will also determine the project folder's and *.Rproj* file's names, which are of the form "p*XXXX*". If the id number is not four digits long, it will be padded on the left side with 0s.

protocol, datawork, analysis, report, css, Rproj
: A character string matching the name of a corresponding template file in the *.templates* subdirectory of the main projects_folder. Default templates are placed there when setup_projects() is run, and the user can edit these if desired.

  Multiple default protocol templates are available. *01_protocol.Rmd*, which by default is the same as *STROBE_protocol.Rmd*, will be selected if protocol is unspecified. Users can edit these default templates.

  If using a custom template, make sure to match the case and file extension exactly.

use_bib
: Logical. If TRUE, a blank *.bib* file will be written into the **progs** subdirectory of the newly created project folder. Its name will be of the form *pXXXX.bib*, and the YAML header of *progs/01_protocol.Rmd* and *progs/04_report.Rmd* will include the line bibliography: pXXXX.bib.

Details

`new_project()` creates a new R project folder that is automatically filled with a *.Rproj* file, helpful subdirectories, and *.Rmd* files to get your project workflow started; `delete_project()` deletes them. The `edit_\*()` functions and the other `new_\*()` and `delete_\*()` functions only create or edit rows in the *.metadata* tibbles.

Newly created project folders (and the *.Rproj* files they contain) both have names of the form "p*XXXX*", where "*XXXX*" denotes the project `id` number. The folder will be an immediate subdirectory of the main `projects_folder` (see also `setup_projects()`) unless the argument `path` specifies a deeper subdirectory. The user may enter various metadata about the project that is stored and may be called forth using the `projects()` function. Some of this metadata will automatically be added to the `header` atop the automatically created *.Rmd* files called *progs/01_protocol.Rmd* and *progs/04_report.Rmd*.

Examples

```
# SETUP
old_path <- Sys.getenv("PROJECTS_FOLDER_PATH")
setup_projects(path = tempdir(), .Renviron_path = fs::path_temp(".Renviron"))
################################################################################

# Creating affiliations
new_affiliation(department_name = "Math Dept.",
                institution_name = "Springfield College",
                address = "123 College St, Springfield, AB")
new_affiliation(department_name = "Art Department",
                institution_name = "Springfield College",
                address = "321 University Boulevard, Springfield, AB",
                id = 42)

# Editing an affiliation
edit_affiliation("Math Dept", department_name = "Mathematics Department")

# Creating authors
new_author(given_names = "Rosetta", last_name = "Stone",
           affiliations = c(42, "Math"), degree = "PhD",
           email = "slab@rock.net", phone = "867-555-5309", id = 8888)
new_author(given_names = "Spiro", last_name = "Agnew", degree = "LLB",
           affiliations = "Art D", id = 13)
new_author(last_name = "Plato", id = 303)

# Editing an author, showcasing the removal of a text element (last_name)
edit_author(author = 303, given_names = "Plato", last_name = NULL)

# Editing an author, showcasing the addition and removal of affiliations
edit_author("Spiro", affiliations = ~ -"Art D" + Math)

# Creating a project
new_project(title = "Understanding the Construction of the United States",
            short_title = "USA", authors = c(13, "Stone"),
            stage = 4, deadline = "2055-02-28", deadline_type = "submission",
            path = "famous_studied/philosophers/rocks",
```

```
            corresp_auth = "Stone", current_owner = "agnew",
            make_directories = TRUE, use_bib = TRUE,
            status = "waiting on IRB")

# Editing a project, showcasing the addition and removal of authors
edit_project("Understanding", short_title = "usa1",
             authors = ~ + "303" - 13 - Stone)

# Wrapped in if(interactive()) because it requires interactive console input
# and fails automated package checking and testing.
if(interactive()) {
  delete_project("usa1")
  delete_author(303)
  delete_affiliation("Math")
}

################################################################################
# CLEANUP
Sys.setenv(PROJECTS_FOLDER_PATH = old_path)
fs::file_delete(c(fs::path_temp("projects"), fs::path_temp(".Renviron")))
```

`file_management` *file management*

Description

Tools for Organizing and Managing Project Files

Usage

```
new_project_group(path)

move_project(project, path, make_directories = FALSE, archived = FALSE)

copy_project(project_to_copy, path, new_id = NA,
  make_directories = FALSE, archived = FALSE)

archive_project(project)

open_project(project, new_session = FALSE, archived = FALSE)
```

Arguments

`path`	A valid path string. See the `path` argument in `new_project()` for details, the one difference being that there is no default (i.e., the user cannot leave `path` blank in these functions).
`project`	Project `id` or unambiguous substring of the project name from the `projects()` tibble.

make_directories
: Logical. If the path represented by the path parameter does not exist, should the needed directories be created?

archived
: Logical indicating whether or not the function should consider archived projects when determining which project the user is referring to in the project/project_to_copy argument. FALSE by default.

  See the **Details** section of archive_project() for more information on the "archived" status of a project.

project_to_copy
: Project id or unambiguous substring of the project name corresponding to the project that is to be copied.

new_id
: Optional integer, ranging from 1 to 9999, used as the newly-created project ID. Must not already exist in projects()$id. If left blank, the lowest available id will be automatically used.

new_session
: Same as the newSession argument in rstudio::openProject().

Details

Projects can be moved (move_project()), copied (copy_project()), deleted (delete_project()) or archived (archive_project).

The difference between delete_project() and archive_project() is that the latter will just move the project to a directory called *archive*, located in the same parent directory as the project. This directory gets created if it doesn't yet exist. Some functions that perform actions on projects will exclude archived projects by default in order to make it easier for the user to enter a nonambiguous string that will match an active (i.e., non-archived) project.

Projects can also be organized into groups. By default, all projects are created within the main projects_folder. To create a project group, which essentially is a subfolder of projects that sits within the main projects_folder (or recursively within another project group's folder), use new_project_group().

open_project() is a wrapper around openProject, but the user only needs to know the project's id, title, or short_title instead of the file path of the project's *.Rproj* file.

See Also

new_project() and delete_project() for other functions that write and delete files

Examples

```
# SETUP
old_path <- Sys.getenv("PROJECTS_FOLDER_PATH")
setup_projects(path = tempdir(), .Renviron_path = fs::path_temp(".Renviron"))
################################################################################

# setting up a simple project directory tree
new_project_group("kidney/clinical")
new_project_group("kidney/genomics")
new_project_group("prostate/clinical")
new_project_group("prostate/genomics")
```

```
# Wrapped in if(interactive()) because it requires interactive console input
# and fails automated package checking and testing.
if(interactive()){
  new_project(title = "Sample Authorless Project", path = "kidney")

  # Moving the project folder, then moving it again.
  move_project(project = 1, "kidney/genomics")
  move_project(project = "Sample Authorless Project", "prostate/clinical")

  # Copying the project
  copy_project(project_to_copy = 1, "kidney/clinical")

  # Archiving the copy of the project
  archive_project(2)

  # Opens the project in same session
  open_project("Sample")

  # Opens the project in a new session
  open_project(1, new_session = TRUE)
}
################################################################################
# CLEANUP
Sys.setenv(PROJECTS_FOLDER_PATH = old_path)
fs::file_delete(c(fs::path_temp("projects"), fs::path_temp(".Renviron")))
```

`header` *Print project header to console*

Description

This function displays the report header for a project. The project header consists of: 1) the project title; 2) the author list; 3) the list of author affiliations; and 4) corresponding author information. The function is helpful when, after editing details of the project (e.g., any of the above information), you want to update your markdown documents. The displayed markdown can be pasted directly in place of the header within the markdown documents (specifically *01_protocol.Rmd* and *04_report.Rmd*).

Usage

```
header(project, archived = FALSE)
```

Arguments

`project`	Project `id` or unambiguous substring of the project name from the `projects()` tibble.
`archived`	Logical indicating whether or not the function should consider archived projects when determining which project the user is referring to in the `project` argument. `FALSE` by default.

See the **Details** section of archive_project() for more information on the "archived" status of a project.

Examples

```
# Included in \donttest{} to save time on example checking.

# SETUP
old_path <- Sys.getenv("PROJECTS_FOLDER_PATH")
setup_projects(path = tempdir(), .Renviron_path = fs::path_temp(".Renviron"))
new_affiliation(department_name = "Math Dept.",
                institution_name = "Springfield College",
                address = "123 College St, Springfield, AB")
new_affiliation(department_name = "Art Department",
                institution_name = "Springfield College",
                address = "321 University Boulevard, Springfield, AB",
                id = 42)
new_affiliation(department_name = "Central Intelligence Agency",
                institution_name = "United States Government",
                address = "888 Classified Dr, Washington DC")
new_affiliation(department_name = "Pyrotechnics",
                institution_name = "ACME")
new_author(given_names = "Rosetta", last_name = "Stone",
           affiliations = c(42, "Math"), degree = "PhD",
           email = "slab@rock.net", phone = "867-555-5309", id = 8888)
new_author(given_names = "Spiro", last_name = "Agnew", degree = "LLB",
           affiliations = "Art D", id = 13)
new_author(given_names = "Plato", id = 303)
new_project(title = "Test Project 1", authors = c(13, "303", "Stone"),
            corresp_auth = "Stone")
################################################################################

header(1)

################################################################################
# CLEANUP
Sys.setenv(PROJECTS_FOLDER_PATH = old_path)
fs::file_delete(c(fs::path_temp("projects"), fs::path_temp(".Renviron")))
```

---

projects_folder *Get file path of main projects folder*

Description

Returns the file path of the main projects folder if it has been established.

Usage

```
projects_folder()
```

Details

The file path is returned as a simple character string. It simply returns the value of `Sys.getenv("PRJOECTS_FOLDER_PATH")`, provided that its value is a file path of a directory that actually exists (i.e., `setup_projects()` has been successfully run).

If it can't find a directory with that path, it returns this string:

`"projects" folder not found. Please run setup_projects()`

Examples

```
projects_folder()
```

---

reordering *Reordering authors and affiliations*

Description

These functions allow the user to reorder authors on a project or to reorder an author's affiliations.

Usage

```
reorder_authors(project, ..., after = 0L, reprint_header = TRUE,
  archived = FALSE)

reorder_affiliations(author, ..., after = 0L)
```

Arguments

`project, author`	The `id` or unambiguous names of a project/author whose authors/affiliations you want to reorder.
`...`	The `id`s or names of authors/affiliations you want to reorder, optionally with their new ranks explicitly stated. See details.
`after`	If not specifying explicit ranks in `...`, the position you want the elements to come after. Works like the `after` argument in `append` or `forcats::fct_relevel`. Ignored if ranks are explicitly provided in `...`.
`reprint_header`	Should the `project`'s header be printed to the console?
`archived`	Logical indicating whether or not the function should consider archived projects when determining which project the user is referring to in the `project` argument. `FALSE` by default. See the **Details** section of `archive_project()` for more information on the "archived" status of a project.

Details

The order of these affects the order of authors and affiliations in headers.

When specifying explicit ranks, enter ... as name-value pairs, in which the names are names of authors/affiliations (or even their [back]quoted ids) and the values are integer ranks you want them to occupy. If entering an integer greater than the total number of authors/affiliations, the element will be put at the end. The after argument will be ignored in this case.

When not specifying explicit ranks, simply enter author/affiliations ids or names in the order you want them, and the ones you entered will be inserted after the position specified by the after argument. By default (after = 0), the authors/affiliations in ... will be moved to the front.

Examples

```
# SETUP
old_path <- Sys.getenv("PROJECTS_FOLDER_PATH")
setup_projects(path = tempdir(), .Renviron_path = fs::path_temp(".Renviron"))
new_affiliation(department_name = "Math Dept.",
                institution_name = "Springfield College",
                address = "123 College St, Springfield, AB")
new_affiliation(department_name = "Art Department",
                institution_name = "Springfield College",
                address = "321 University Boulevard, Springfield, AB",
                id = 42)
new_affiliation(department_name = "Central Intelligence Agency",
                institution_name = "United States Government",
                address = "888 Classified Dr, Washington DC")
new_affiliation(department_name = "Pyrotechnics",
                institution_name = "ACME")
new_author(given_names = "Rosetta", last_name = "Stone",
           affiliations = c(42, "Math"), degree = "PhD",
           email = "slab@rock.net", phone = "867-555-5309", id = 8888)
new_author(given_names = "Spiro", last_name = "Agnew", degree = "LLB",
           affiliations = "Art D", id = 13)
new_author(given_names = "Plato", id = 303)
new_author(given_names = "Condoleezza", last_name = "Rice", degree = "PhD",
           affiliations = c(1, 42, "Agency", "ACME"), phone = "555-555-5555",
           email = "condoleeza@ri.ce")
new_author(given_names = "Jane", last_name = "Goodall", degree = "PhD",
           affiliations = 3, id = 5)
new_project(title = "Understanding the Construction of the United States",
            short_title = "USA",
            authors = c(13, "Stone", "zz", "303", "Jane Goodall"),
            stage = 4, deadline = "2055-02-28", deadline_type = "submission",
            path = "famous_studied/philosophers/rocks",
            corresp_auth = "Stone", current_owner = "agnew",
            make_directories = TRUE, use_bib = TRUE,
            status = "waiting on IRB")
################################################################################

# Reordering with unnamed arguments
reorder_affiliations(author = "RICE", "ACME", 42, after = 1)
```

```
# Reordering with named arguments
reorder_authors(project = 1, "Rosetta" = 99, `303` = 2, "5" = 1)

################################################################################
# CLEANUP
Sys.setenv(PROJECTS_FOLDER_PATH = old_path)
fs::file_delete(c(fs::path_temp("projects"), fs::path_temp(".Renviron")))
```

`setup_projects` *Set up the projects folder*

Description

Creates or restores the projects folder at the user-specified path.

Usage

```
setup_projects(path, overwrite = FALSE, make_directories = FALSE,
  .Renviron_path = fs::path_home(".Renviron"))
```

Arguments

`path`	The full file path where the user would like a directory called "projects" to be created, wherein all projects and their data will dwell.
`overwrite`	Logical indicating whether or not to abandon any previously stored projects folders stored in the system.
`make_directories`	Logical indicating whether or not the function should write any directories specified in the `path` argument that don't already exist.'
`.Renviron_path`	The full file path of the .Renviron file where the user would like to store the `projects_folder()` path. Default is the home .Renviron file. If the file doesn't exist it will be created.

Details

The `projects` package remembers where the `projects_folder()` is located by storing its file path in an .Renviron file (the home .Renviron file by default). The entry is named `PROJECTS_FOLDER_PATH`.

Note that changing the .Renviron_path argument may create an .Renviron file that R will not notice or use. See Startup for more details.

Value

The project folder's path, invisibly. It will be "" if it doesn't exist.

Default contents

The projects_folder automatically contains the subdirectories *.metadata* and *.template*, which are hidden by default on some operating systems.

The *.metadata* folder and its contents should **never** be manually moved or modified.

The *.templates* will contain several templates that new_project() reads when creating a new project. Advanced users may edit these templates or add their own. See new_project() for details.

Behavior when projects folder already exists

If overwrite = TRUE, the function will run no matter what. Use with caution.

If the user has a pre-existing projects_folder and runs this command with the pre-existing projects_folder's path, nothing will be deleted.

Therefore, if the user "broke" the projects folder (e.g., by deleting metadata; by changing the "PROJECTS_FOLDER_PATH" line in the *.Renviron* file), the user can "fix" the projects folder to some degree by running this function with the folder's actual file path (e.g., restore all default templates; restore missing metadata files).

See Also

new_project() for information on templates

Startup for more information on how *.Renviron* files work

Examples

```
# This sequence is used in all other examples in this package.

# Back up old projects_folder()
old_path <- Sys.getenv("PROJECTS_FOLDER_PATH")

# This sets up an example projects_folder() in a temporary directory.
# It will not edit any of the user's .Renviron files.
setup_projects(path = tempdir(), .Renviron_path = fs::path_temp(".Renviron"))

# Cleanup
Sys.setenv(PROJECTS_FOLDER_PATH = old_path)
fs::file_delete(c(fs::path_temp("projects"), fs::path_temp(".Renviron")))
```

Index