Package 'projects'

May 29, 2020

Title A Project Infrastructure for Researchers

Version 2.1.1

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://cran.r-project.org/package=projects

Depends R (>= 3.4.0)

Imports dplyr (>= 0.8.5), fs (>= 1.4.1), lubridate (>= 1.7.8), magrittr (>= 1.5), methods, purrr (>= 0.3.4), readr (>= 1.3.1), rlang (>= 0.4.6), rstudioapi (>= 0.11), sessioninfo (>= 1.1.1), stringr (>= 1.4.0), tibble (>= 3.0.1), vctrs (>= 0.2.4), zip (>= 2.0.4)

Suggests forcats, here (>= 0.1), testthat (>= 2.3.2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Collate 'set_generics.R' 'class-projects_author.R' 'class-projects_stage.R' 'new.R' 'edit.R' 'email_authors.R' 'file_management.R' 'getters.R' 'header.R' 'metadata_manipulation.R' 'projects.R' 'reproducibility.R' 'setup.R' 'update.R' 'utilities.R' 'utils-pipe.R' 'validation.R' 'zzz.R'

NeedsCompilation no

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

Maintainer Nik Krieger <nk@case.edu>

Repository CRAN

Date/Publication 2020-05-29 12:40:02 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 table 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,
  all_stages = FALSE,
  exclude = c(0L, 6L),
  path = NULL,
  archived = FALSE,
  verbose = FALSE,
  authors = FALSE
)

ideas(project, archived = FALSE, verbose = FALSE, authors = FALSE)

manuscripts(project, archived = FALSE, verbose = FALSE, authors = 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 `id`s and/or names. Only rows matching one or more entries will be returned. This is the one setting in which the package does not return throw an error if user input matches multiple projects.

`all_stages`
: Logical, indicating whether or not to include projects of all stages, **overriding the** `exclude` **argument**.

`exclude`
: A vector of numbers or character strings that can be validated against the list of project stages:

`0`: idea
`1`: design

2: data collection
3: analysis
4: manuscript
5: under review
6: accepted

Ignored if all_stages = TRUE

path — A single file path of a directory within the main projects folder; only projects whose folder is in this directory will be returned.

archived — Logical, indicating whether or not to include projects that have been archived using archive_project(). FALSE by default.

verbose — Logical, indicating whether or not to return all columns of the projects(); if FALSE, only the id, current_owner, status, and stage columns are returned. Defaults to FALSE.

Details

ideas() is a shortcut for projects(exclude = 1:6) (including only projects of stage 0: idea).

manuscripts() is a shortcut for projects(exclude = c(0:3,6)) (yielding only projects of stage 4: manuscript and 5: under review).

If one or more of the projects, authors, or affiliations arguments is set to TRUE, a dplyr::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.

Value

A tibble.

Examples

```
################################################################################
# SETUP
old_home <- Sys.getenv("HOME")
old_ppath <- Sys.getenv("PROJECTS_FOLDER_PATH")
temp_dir <- tempfile("dir")
dir.create(temp_dir)
Sys.unsetenv("PROJECTS_FOLDER_PATH")
Sys.setenv(HOME = temp_dir)
setup_projects(path = temp_dir)
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", stage = 1)
new_project(title = "Test project 2", current_owner = "eezza", stage = 2)
new_project(title = "Test project 3", current_owner = "Plato", stage = 3)
new_project(title = "Fun project 4",  current_owner = "Rice", stage = 4)
new_project(title = "Fun project 5",  current_owner = "Rice", stage = 5)
new_project(title = "Fun project 6",  current_owner = "Rice", stage = 6)
new_project(title = "Good idea",  current_owner = "Rice", stage = 0)
################################################################################

# View entire affiliations table
affiliations()

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

# View only active projects with "Fun" in their title.
projects("Fun")

# View all projects with "Rice" as the current_owner
projects(all_stages = TRUE) %>% dplyr::filter(current_owner == "Rice")

# View manuscripts
manuscripts()

# View ideas
ideas()

# 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
# (or, the user can just restart R)
Sys.setenv(HOME = old_home, PROJECTS_FOLDER_PATH = old_ppath)
```

email_authors *Write an email to project authors*

Description

Invokes `utils::browseURL("mailto://[author emails]")` for a specified project, or for the currently open project if `project` is left as `NULL`.

Usage

```
email_authors(
  project = NULL,
  browser = getOption("browser"),
  encodeIfNeeded = FALSE
)
```

Arguments

`project`	Project `id` or unambiguous substring of the project name from the `projects()` table. Defaults to `NULL` (see **Details**).
`browser, encodeIfNeeded`	See `utils::browseURL()`.

Details

The success of this function depends on the platform and the specified `browser`. See the **Details** and **URL schemes** sections of `utils::browseURL()`.

If `project = NULL`, the function selects the project in the `projects()` table whose `path` is equal to `rstudioapi::getActiveProject()`.

See Also

`utils::browseURL()`; `rstudioapi::getActiveProject()` for information on `browser` and `encodeIfNeeded` arguments.

Examples

```
# Wrapped in if (interactive()) because this function is interactive by nature.
if (interactive()) {

  # If you have a projects() project open, just run it:
  email_authors()

  # Otherwise, specify a project:

  ###########################################################################
  # Setup
  old_home <- Sys.getenv("HOME")
```

```
  old_ppath <- Sys.getenv("PROJECTS_FOLDER_PATH")
  temp_dir <- tempfile("dir")
  dir.create(temp_dir)
  Sys.unsetenv("PROJECTS_FOLDER_PATH")
  Sys.setenv(HOME = temp_dir)
  setup_projects(path = temp_dir)
  new_author("Rhonda", "Rondale", email = "ronda.rondale@co.uk")
  new_author("Betty", "Betts", email = "betty@co.uk")
  new_project("Inventing the Ring of Power", authors = c("Betty", "Ron"))
  ##############################################################################

  email_authors("Ring of Power")

  ##############################################################################
  # Cleanup (or just restart R)
  Sys.setenv(HOME = old_home, PROJECTS_FOLDER_PATH = old_ppath)
}
```

`export_project` *Compress a project folder*

Description

Creates a compressed file out of a user-specified project folder for sharing.

Usage

```
export_project(project, zipfile, include_hidden = FALSE, exclude = NULL)
```

Arguments

`project`	Project `id` or unambiguous substring of the project name from the `projects()` table.
`zipfile`	Desired file path of the resulting compressed folder file, including the file's desired name and file extension. See the `zipfile` argument for the `zip::zipr()` function.
`include_hidden`	Logical indicating whether or not to include hidden folders and files (e.g., those with names that begin with a period). Defaults to `FALSE`.
`exclude`	Character vector of exact names of first-level subdirectories of the project folder to exclude from the resulting compressed folder file.

Details

Currently, this function uses `zip::zipr()`.

Value

The name of the created zip file, invisibly.

file_management *file management*

Description

Tools for Organizing and Managing Project Files

Usage

```
new_project_group(path)

rename_folder(project, new_folder_name, archived = FALSE)

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

copy_project(
  project_to_copy,
  path,
  new_id = NA,
  new_folder_name = paste0("p", stringr::str_pad(new_id, 4, pad = "0")),
  new_short_title = NA,
  make_directories = FALSE,
  archived = FALSE
)

archive_project(project)

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

move_projects_folder(
  new_path,
  make_directories = FALSE,
  .Renviron_path = file.path(Sys.getenv("HOME"), ".Renviron")
)

rename_projects_folder(
  new_name,
  .Renviron_path = file.path(Sys.getenv("HOME"), ".Renviron")
)
```

Arguments

path	A valid path string. For copy_project() only, if left blank, the preexisting project's directory is used. All other functions here require a valid path. See the path argument in new_project() for details on valid paths.

`project` Project `id` or unambiguous substring of the project name from the `projects()` table

`new_folder_name` Character string of new name for project folder. Always processed with `fs::path_sanitize()`.

`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 **Details**.

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

`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_short_title` Optional character string that becomes the `short_title` of the project copy. It also becomes the project copy's folder name under normal circumstances (see **Details**).

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

`new_path` A valid string indicating a path where the projects folder should be moved. The projects folder will have the same name, but it will be moved into this directory.

`.Renviron_path` The full file path of the .Renviron file where the user would like to store the updated `projects_folder()` path. Default is the home .Renviron file. If the file doesn't exist it will be created. See also `setup_projects()`.

`new_name` A valid directory name for the projects folder.

Details

Projects can be moved (`move_project()`), copied (`copy_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. Most 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 is essentially a subfolder of the main projects folder, use `new_project_group()`.

`open_project()` is a wrapper around `rstudioapi::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. If there is no *.Rproj* file in the project's folder, the user has the option to restore a default *.Rproj* file. If there are multiple *.Rproj* files, an error is thrown.

`move_projects_folder()` allows the user to move the entire projects folder created by `setup_projects()` into a different directory, and `rename_projects_folder()` changes its name.

See Also

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

Examples

```
################################################################################
# SETUP
old_home <- Sys.getenv("HOME")
old_ppath <- Sys.getenv("PROJECTS_FOLDER_PATH")
temp_dir <- tempfile("dir")
dir.create(temp_dir)
Sys.unsetenv("PROJECTS_FOLDER_PATH")
Sys.setenv(HOME = temp_dir)
setup_projects(path = temp_dir)
################################################################################

# 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", parent_directory = "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")

  # Renaming the folder of the copy of the project
  rename_folder(project = 2, "copy")

  # Archiving the copy of the project
  archive_project(2)

  # Moving and renaming the entire projects folder
  temp_dir2 <- tempfile("dir")
  dir.create(temp_dir2)
  move_projects_folder(temp_dir2)
  projects_folder()
  rename_projects_folder("foobar")
  projects_folder()

  # 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(HOME = old_home, PROJECTS_FOLDER_PATH = old_ppath)
```

`header` *Print project header to console*

Description

Prints a header to the console to be copied and pasted into the YAML of a project protocol or manuscript R Markdown file. These lines essentially produce a title page when the R Markdown file is knitted.

Usage

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

Arguments

`project`	Project `id` or unambiguous substring of the project name from the `projects()` table.
`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 project header consists of:

1. the project title
2. the author list
3. the list of author affiliations
4. corresponding author information

Examples

```
################################################################################
# SETUP
old_home <- Sys.getenv("HOME")
old_ppath <- Sys.getenv("PROJECTS_FOLDER_PATH")
temp_dir <- tempfile("dir")
dir.create(temp_dir)
Sys.unsetenv("PROJECTS_FOLDER_PATH")
```

```
Sys.setenv(HOME = temp_dir)
setup_projects(path = temp_dir)
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(HOME = old_home, PROJECTS_FOLDER_PATH = old_ppath)
```

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

Description

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

Usage

```
new_project(
  title = NA,
  current_owner = NA,
  stage = c("1: design", "2: data collection", "3: analysis", "4: manuscript",
    "5: under review", "6: accepted", "0: idea"),
  status = "just created",
  short_title = NA,
  authors = NULL,
  corresp_auth = NA,
```

```
  creator = NA,
  deadline_type = NA,
  deadline = NA,
  id = NA,
  folder_name = paste0("p", stringr::str_pad(id, 4, pad = "0")),
  parent_directory = projects_folder(),
  make_directories = FALSE,
  template_folder = "default_folder"
)

new_idea(title, status = "just an idea", ...)

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

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

edit_project(
  project,
  title = NULL,
  short_title = NULL,
  authors = NULL,
  current_owner = NULL,
  status = NULL,
  deadline_type = NULL,
  deadline = NULL,
  stage = NULL,
  corresp_auth = NULL,
  creator = NULL,
  archived = FALSE
)

edit_author(
  author,
  given_names = NULL,
```

```
  last_name = NULL,
  affiliations = NULL,
  title = NULL,
  degree = NULL,
  email = NULL,
  phone = NULL
)

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

delete_project(project, archived = FALSE)

delete_author(author)

delete_affiliation(affiliation)
```

Arguments

`title`	For `new_project()`, `new_idea()`, and `edit_project()`, the title of the project. For the `new_author()` and `edit_author()`, the job title of the author.
`current_owner, corresp_auth, creator`	An `id` or unambiguous `last_name/given_names` of one of the authors in the `authors()` table, which will be coerced into a projects_author-class object. If `corresp_auth` is specified, this author's contact information will be especially included in the output of `header()`. If `creator` is left blank, the numeric portion of the resulting projects_author-class object will be `0:`, followed by the value of `Sys.info()["user"]` (e.g., `0: user_j_smith`).
`stage`	A number or string that will partially match exactly one of `c("1: design","2: data collection","3: analysis","4: manuscript","5: under review","6: accepted","0: ideas")`, communicating the stage the project is in. This will be coerced to be a character vector of class `projects_stage`. For `new_project()`, defaults to `"1: design"`. See projects_stage-class.
`status`	A free text field, intended to communicate the most current condition the project is in. For `new_project()`, default is `"just created"`. For `new_idea()`, default is `"just an idea"`.
`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 and `-` to remove (see formula).

  Authors and affiliations may be specified by `id` or name. Each element must match an existing row in the `authors()`/`affiliations()` table.

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

`deadline`
: A `POSIXct` object or something coercible to one (via `lubridate::as_datetime()`).

`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.

`folder_name`
: A character string that can serve as a valid directory name. By default, it is "p" followed by the project `id` number left-filled with "0" until the number is four digits long.

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

  1. the *absolute* path of the projects folder (i.e., the value of `projects_folder()`, which is the 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 value of `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.

`template_folder`
: A character string naming a folder in the *.templates* folder that will be copied into the projects folder as the new project folder, renamed according to the value of the `folder_name` argument. See also **Details** below.

`...`
: Additional arguments to be passed to `new_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 when `header()` is run.

`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`, `email` will be included in "Corresponding author:" section written by `header()`.

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

`project, author, affiliation`
: The `id` or unambiguous name(s) of a project/author/affiliation to `edit_*()` or to `delete_*()`.

`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

`new_project()` copies the folder in the *.templates* folder named by the `template_name` argument into the projects folder, giving it the name specified by the `folder_name` argument. It then creates a line in the `projects()` table for the newly created project, filling many of its fields with the contents of corresponding arguments of this function. See `setup_projects()` for more information on the *.templates* folder.

`delete_project()` deletes project folders and removes their line from the `projects()` table.

The `edit_*()` functions and the other `new_*()` and `delete_*()` functions only create or edit rows in the *.metadata* tables.

`new_idea()` is a convenience function for quickly creating projects in the `"0: idea"` stage.

Value

`new_affiliation()` and `edit_affiliation()` simply return the new or edited row of the `affiliations()` `tibble`.

`new_project()`, `new_author()`, `edit_project()`, `edit_author()`, and the `delete_*()` functions invisibly return the row of the corresponding metadata `tibble` that was added/edited/deleted, although the contents of this row are printed as a side-effect along with the other relevant information where applicable (e.g., project authors, author affiliations, project file paths).

`new_idea()` returns the `id`, `title`, and `status` columns of the newly created row of the `projects()` `tibble`.

Examples

```
################################################################################
# SETUP
old_home <- Sys.getenv("HOME")
old_ppath <- Sys.getenv("PROJECTS_FOLDER_PATH")
temp_dir <- tempfile("dir")
dir.create(temp_dir)
Sys.unsetenv("PROJECTS_FOLDER_PATH")
Sys.setenv(HOME = temp_dir)
setup_projects(path = temp_dir)
################################################################################

# 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 = NA)

# 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",
  parent_directory = "famous_studied/philosophers/rocks",
  corresp_auth = "Stone",
  current_owner = "agnew",
  make_directories = 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
```

```
)

new_idea(title = "Boiling the Ocean")

# 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(HOME = old_home, PROJECTS_FOLDER_PATH = old_ppath)
```

---

projects_author projects_author *vector*

Description

Objects of this class contain both the `id` and the `last_name` of an author so that the package and the user, respectively, can easily identify the author.

Usage

```
projects_author(x = character())

match.projects_author(x, table, nomatch = NA_integer_, incomparables = NULL)

## S4 method for signature 'projects_author,ANY'
match(x, table, nomatch = NA_integer_, incomparables = NULL)

## S4 method for signature 'ANY,projects_author'
match(x, table, nomatch = NA_integer_, incomparables = NULL)

## S4 method for signature 'projects_author,projects_author'
match(x, table, nomatch = NA_integer_, incomparables = NULL)

`%in%.projects_author`(x, table)

## S4 method for signature 'projects_author'
x %in% table
```

Arguments

x	For `projects_author()`, an integer or character vector. For For `match()` and `%in%`, an integer, a character string, or a `projects_author` object. See `match()` and **Equality and value matching methods** below.

table	An integer number, a character string, or a projects_author object. See match() and **Equality and value matching methods** below.
nomatch	See match().
incomparables	An integer number, a character string, or a projects_author object. See match().

Details

Essentially, this is a character string of the form:

id: last_name

projects_author() coerces an integer or character vector to a projects_author object, validating each element against the existing authors() table.

Numeric coercion methods

as.integer(), as.double(), and as.numeric() return the id portion of the projects_author object as an integer/double. The methods for the equality and value matching functions described below make use of these numeric coercion methods. Users desiring to apply value matching functions other than the ones described below may similarly take advantage of these.

Equality and value matching methods

Methods for ==, !=, match(), and %in% enable users to test equality and to value match among projects_author objects and as well as between projects_author objects and unclassed numbers/characters. When testing or matching against a numeric vector, the projects_author object is first coerced to an integer with the as.integer() method described above. When testing or matching against a character vector, the character vector is validated against the authors() table.

See Also

Ops; Methods_for_Nongenerics.

Examples

```
################################################################################
# SETUP
old_home <- Sys.getenv("HOME")
old_ppath <- Sys.getenv("PROJECTS_FOLDER_PATH")
temp_dir <- tempfile("dir")
dir.create(temp_dir)
Sys.unsetenv("PROJECTS_FOLDER_PATH")
Sys.setenv(HOME = temp_dir)
setup_projects(path = temp_dir)
new_author("chuck", "jonesman", id = 33)
new_author("Hattie", "Hatsman", id = 45)
################################################################################

jones <- projects_author("33: Jones")

jones
```

```
as.integer(jones) # 33

jones == 33       # TRUE
jones == 10       # FALSE
jones != 33       # FALSE

jones %in% c(20:40)     # TRUE
match(jones, c(31:40))  # 3

# Comparing a projects_author object to a character vector results in the
# character strings being validated against the authors() table. Then, the id
# numbers are compared.
jones == c("jOnES", "hat")   # TRUE FALSE

################################################################################
# Cleanup (or just restart R)
Sys.setenv(HOME = old_home, PROJECTS_FOLDER_PATH = old_ppath)
```

`projects_folder` *projects folder path*

Description

Returns the file path of the main projects folder if it has been established via `setup_projects()`.

Usage

```
projects_folder()
```

Details

The file path is returned as a simple character string. It simply returns the value of `Sys.getenv("PROJECTS_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()`

See Also

`setup_projects()` for setting up the projects folder.

Examples

```
projects_folder()
```

projects_stage projects_stage *vector*

Description

Objects of this class are merely a character string containing a number and a name of one of seven project development stages.

Usage

```
projects_stage(x = character())

match.projects_stage(x, table, nomatch = NA_integer_, incomparables = NULL)

## S4 method for signature 'projects_stage,ANY'
match(x, table, nomatch = NA_integer_, incomparables = NULL)

## S4 method for signature 'ANY,projects_stage'
match(x, table, nomatch = NA_integer_, incomparables = NULL)

## S4 method for signature 'projects_stage,projects_stage'
match(x, table, nomatch = NA_integer_, incomparables = NULL)

`%in%.projects_stage`(x, table)

## S4 method for signature 'projects_stage'
x %in% table
```

Arguments

x	For projects_stage(), an integer or character vector. For For match() and %in%, an integer, a character string, or a projects_stage object. See match() and **Comparison and value matching methods** below.
table	An integer number, a character string, or a projects_stage object. See match() and **Comparison and value matching methods** below.
nomatch	See match().
incomparables	An integer number, a character string, or a projects_stage object. See match().

Details

A projects_stage object is either a missing value (NA) or one of:

0: idea
1: design
2: data collection
3: analysis
4: manuscript

5: under review
6: accepted

projects_stage() validates and coerces a vector of the above integers or strings to a projects_stage S3 vector.

Value

For projects_stage(), an S3 vector of class projects_stage.

Numeric coercion methods

as.integer(), as.double(), and as.numeric() return the stage number of the projects_stage object as an integer/double. The methods for the comparison and value matching functions described below make use of these numeric coercion methods. Users desiring to apply value matching functions other than the ones described below may similarly take advantage of these.

Comparison and value matching methods

Methods for the Comparison operators as well as match() and %in% enable users to test equality and to value match among projects_stage objects and as well as between projects_stage objects and unclassed numbers/characters. When comparing or value matching against a numeric vector, the projects_stage object is first coerced to an integer with the as.integer() method described above. When testing or value matching against a character vector, the character vector is validated against the list of project stages enumerated above.

See Also

Ops; Methods_for_Nongenerics.

Examples

```
stage <- projects_stage("4: manuscript")

as.integer(stage) # 4

stage == 4       # TRUE
stage != 4       # FALSE
stage <  6       # TRUE

stage %in% c(3:6)  # TRUE
match(stage, 0:4)  # 5

stage %in% c("design", "manusc", "idea")  # TRUE

more_stages <- projects_stage(c("0: idea", "4: manuscript", "1: design"))

match("MAnuscRIPT", more_stages)      # 2
```

reordering *Reordering authors and affiliations*

Description

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

Usage

```
reorder_authors(project, ..., after = 0L, 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 `ids` 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 `...`.
`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 authors and affiliations affects the order in which these items appear in the output of `header()`.

When specifying explicit ranks, enter `...` as name-value pairs (e.g., Johnson = 2, "Baron Cohen" = 4). You can even enumerate authors/affiliations by their corresponding (quoted) `id` numbers (e.g., '7' = 2, ACME = 4, '22' = 6). 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. This behavior corresponds to that of `append()` or `forcats::fct_relevel()`.

Examples

```
################################################################################
# SETUP
old_home <- Sys.getenv("HOME")
old_ppath <- Sys.getenv("PROJECTS_FOLDER_PATH")
temp_dir <- tempfile("dir")
dir.create(temp_dir)
Sys.unsetenv("PROJECTS_FOLDER_PATH")
Sys.setenv(HOME = temp_dir)
setup_projects(path = temp_dir)
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",
            parent_directory = "famous_studied/philosophers/rocks",
            corresp_auth = "Stone", current_owner = "agnew",
            make_directories = TRUE,
            status = "waiting on IRB")
################################################################################

# Rice's affiliations before reordering:
authors("rice", affiliations = TRUE)

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

# Rice's affiliations after reordering:
authors("rice", affiliations = TRUE)
```

```
# Project 1 header before reordering authors:
header(1)

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

# Project 1 header after reordering authors:
header(1)

################################################################################
# CLEANUP
Sys.setenv(HOME = old_home, PROJECTS_FOLDER_PATH = old_ppath)
```

`save_session_info` *Save R session information*

Description

Creates a dated text file (.txt) containing the contents of `sessioninfo::session_info()`.

Usage

```
save_session_info(path_dir = here::here("progs", "session_info"))
```

Arguments

`path_dir`	The full path of the directory where the session information text file shall be written. If it doesn't exist, it is written with `fs::dir_create()`.

Details

The date and time when this function was run is included in the resulting .txt file's name and first line. This date and time is obtained from `Sys.time()`.

For the file name, hyphens (-) are removed from the date, spaces are replaced with underscores (_), and colons (:) are replaced with a modifier letter colon (U+A789).

Value

A list of two:

`$ time` : the value of `Sys.time()` that the function used

`$ session_info()` : the value of `sessioninfo::session_info()` that the function used

`setup_projects` *Set up the projects folder*

Description

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

Usage

```
setup_projects(
  path,
  folder_name = "projects",
  overwrite = FALSE,
  make_directories = FALSE,
  .Renviron_path = file.path(Sys.getenv("HOME"), ".Renviron")
)
```

Arguments

`path`	The file path of the **directory inside of which** the user would like the projects folder to be created. Do not include the name of the projects folder itself (i.e., the value of the argument `folder_name` below).
`folder_name`	The name of the projects folder that will be created in the directory specified by the argument `path` above. Defaults to `"projects"`.
`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 a .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.

Default contents

The projects folder automatically contains the subdirectories *.metadata* and *.templates*, which are hidden by default on some operating systems.

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

The *.templates* folder is where template project files and folders should be stored. When this function is successfully run, the default projects folder template is created (as "default_folder") alongside a few other template files. When a new project is created, `new_project()` looks here for the folder named by its `template_folder` argument (`"default_folder"` by default), and this folder is copied into the projects folder (with name specified by the `folder_name` argument) as the new project folder. Users are able and encouraged to customize the `default_folder` to suit their research needs, and may even create multiple project folder templates for different situations.

The default templates are in the folder located at the path produced by running: `system.file("templates",package = "projects")`

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

```
################################################################################
# Setup
# Any existing "projects" folder is left totally untouched,
# and the user's home directory and .Renviron file are also left untouched.
old_home  <- Sys.getenv("HOME")
old_ppath <- Sys.getenv("PROJECTS_FOLDER_PATH")
temp_dir <- tempfile("dir")
dir.create(temp_dir)
Sys.setenv(HOME = temp_dir)
Sys.unsetenv("PROJECTS_FOLDER_PATH")
################################################################################

# Creating the projects folder
setup_projects(path = temp_dir)

# Viewing the projects folder path:
path1 <- projects_folder()
```

```
# Viewing the contents of the projects folder:
list.files(path1, full.names = TRUE, recursive = TRUE,  all.files = TRUE)

# Create an arbitrary subfolder in temp_dir:
subfolder_path <- file.path(temp_dir, "test")
dir.create(subfolder_path)


# Wrapped in if (interactive()) because it requires user input
if (interactive()) {
  # The function won't let the user abandon the old projects folder...
  setup_projects(path = subfolder_path)

  # ...unless overwrite = TRUE
  setup_projects(path = file.path(temp_dir, "test"), overwrite = TRUE)

  # Even then, only the stored location of the projects folder is overwritten.
  # The old projects folder still exists:
  list.files(path1, full.names = TRUE, recursive = TRUE,  all.files = TRUE)

  # Giving the "projects" folder a different name:
  setup_projects(path = temp_dir, folder_name = "studies", overwrite = TRUE)
}

################################################################################
# Cleanup
# (or, the user can just restart R)
Sys.setenv(HOME = old_home, PROJECTS_FOLDER_PATH = old_ppath)
################################################################################
```

`update_metadata` *Update the project metadata*

Description

Safely updates existing project metadata to be compatible with `projects` 1.X.X.

Usage

```
update_metadata(ask = TRUE)
```

Arguments

`ask`	Logical, indicating whether or not the user would be asked at the command line whether or not to proceed. Defaults to `TRUE`.

Details

Prior to projects 1.X.X, the stage, current_owner, corresp_auth, and creator columns of the projects() table were different.

The stage column was a factor, and users had to type stage names exactly, down to the integer, colon, and space. Now, this column is of class projects_stage-class.

The latter three columns were integers corresponding to ids in the authors() table, so users would have to query that table if they did not remember which author was denoted by the integer id.

See Also

projects_stage-class; projects_author-class.

Index