Package ‘strata’

November 27, 2024

Title Simple Framework for Simple Automation

Version 1.0.1

Description A tool suite for building project frameworks for users with access to only the most basic of automation tools.

License MIT + file LICENSE

URL https://github.com/asenetcky/strata, https://asenetcky.github.io/strata/

BugReports https://github.com/asenetcky/strata/issues

Imports checkmate (>= 2.3.2), dplyr (>= 1.1.0), fs (>= 1.6.4), glue (>= 1.8.0), lifecycle (>= 1.0.0), purrr (>= 1.0.2), readr (>= 2.0.0), rlang (>= 1.1.4), stringr (>= 1.5.1), tibble (>= 3.2.1)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Alexander Senetcky [aut, cre, cph] (<https://orcid.org/0009-0009-3730-5397>)

Maintainer Alexander Senetcky <asenetcky@gmail.com>

Repository CRAN

Date/Publication 2024-11-27 13:00:02 UTC

Contents

adhoc_lamina *Execute a single lamina ad hoc*

Description

`adhoc_lamina()` will execute *only* the lamina and the code therein contained as specified by `lamina_path` with or without log messages.

Usage

```
adhoc_lamina(lamina_path, silent = FALSE)
```

Arguments

`lamina_path`	Path to lamina.
`silent`	Suppress log messages? If `FALSE` (the default), log messages will be printed to the console. If `TRUE`, log messages will be suppressed.

Value

invisible data frame of execution plan.

Examples

```
tmp <- fs::dir_create(fs::file_temp())
result <- strata::build_quick_strata_project(tmp, 1, 1)
adhoc_lamina(
  fs::path(tmp, "strata", "stratum_1", "s1_lamina_1"),
)
fs::dir_delete(tmp)
```

`adhoc_stratum` *Execute a single stratum ad hoc*

Description

`adhoc_stratum()` will execute *only* the stratum, its child laminae and the code therein contained as specified by `stratum_path` with or without log messages.

Usage

```
adhoc_stratum(stratum_path, silent = FALSE)
```

Arguments

`stratum_path`	Path to stratum folder
`silent`	Suppress log messages? If `FALSE` (the default), log messages will be printed to the console. If `TRUE`, log messages will be suppressed.

Value

invisible data frame of execution plan.

Examples

```
tmp <- fs::dir_create(fs::file_temp())
result <- strata::build_quick_strata_project(tmp, 1, 1)
adhoc_stratum(
  fs::path(tmp, "strata", "stratum_1"),
)
fs::dir_delete(tmp)
```

`build_lamina` *Add a lamina to the project space*

Description

Add a lamina to the project space

Usage

```
build_lamina(lamina_name, stratum_path, order = 1, skip_if_fail = FALSE)
```

Arguments

`lamina_name`	Name of lamina
`stratum_path`	Path to stratum folder
`order`	Execution order, default is `1`
`skip_if_fail`	Skip this lamina if it fails, default is `FALSE`

Value

invisibly returns fs::path to lamina

Examples

```
tmp <- fs::dir_create(fs::file_temp())
result_stratum_path <- build_stratum("my_stratum_name", tmp)
result_lamina_path <- build_lamina("my_lamina_name", result_stratum_path)
result_lamina_path
fs::dir_delete(tmp)
```

`build_outlined_strata_project` *Build a strata project from an outline dataframe*

Description

Users with a specific idea in mind already can map out the intended project structure in an outline dataframe and use `build_outlined_strata_project()` to build the project using the dataframe as a blueprint.

Usage

```
build_outlined_strata_project(outline)
```

Arguments

`outline`	A data frame with the following columns: project_path, stratum_name, stratum_order, lamina_name, lamina_order, skip_if_fail.

Value

invisible dataframe of the survey of the strata project.

Outline

The outline dataframe should have the following columns:

- `project_path`: The path to the project.
- `stratum_name`: The name of the stratum.
- `stratum_order`: The order of the stratum.
- `lamina_name`: The name of the lamina.
- `lamina_order`: The order of the lamina within the stratum.
- `skip_if_fail`: A logical indicating if the lamina should be skipped if it fails.

Each row of the outline dataframe represents a stratum and lamina combination to be created in the project. A Placeholder R script will be created in each lamina directory to help remind the user to replace it with their own code.

There can only be those 6 columns, and there can be no missing values in the dataframe. The `stratum_name` and `stratum_order` columns must contain unique values.

Examples

```
tmp <- fs::dir_create(fs::file_temp())
outline <- tibble::tibble(
  project_path = tmp,
  stratum_name = c("test1", "test2"),
  stratum_order = c(1, 2),
  lamina_name = c("lamina1", "lamina1"),
  lamina_order = c(1, 2),
  skip_if_fail = FALSE
)
result <- build_outlined_strata_project(outline)
dplyr::glimpse(result)
main(tmp)
fs::dir_delete(tmp)
```

`build_quick_strata_project` *Quickly build strata project with minimal input and standard names*

Description

`build_quick_strata_project` will create a project with the specified number of strata - `num_strata`, with the specified number of laminae

- `num_laminae_per` per stratum. The strata and laminae will be named `stratum_1`, `stratum_2`, etc. and `s1_lamina_1`, `s1_lamina_2`, etc.

Usage

```
build_quick_strata_project(project_path, num_strata = 1, num_laminae_per = 1)
```

Arguments

`project_path`	A path to strata project folder.
`num_strata`	Number of strata to create.
`num_laminae_per`	Number of laminae to create per stratum.

Value

invisible dataframe of the survey of the strata project.

Examples

```
tmp <- fs::dir_create(fs::file_temp())
result <- build_quick_strata_project(tmp, 2, 2)
dplyr::glimpse(result)
main(tmp)
fs::dir_delete(tmp)
```

`build_stratum` *Add a stratum to the project space*

Description

Add a stratum to the project space

Usage

```
build_stratum(stratum_name, project_path, order = 1)
```

Arguments

`stratum_name`	Name of stratum
`project_path`	A path to strata project folder.
`order`	Execution order, default is 1

Value

invisibly returns fs::path to stratum

Examples

```
tmp <- fs::dir_create(fs::file_temp())
result <- build_stratum("my_stratum_name", tmp)
result
fs::dir_delete(tmp)
```

`edit_toml` *Edit a toml file by providing a dataframe replacement*

Description

Users can use `edit_toml()` to edit a toml file (should they opt not to use a text editor) by providing a dataframe of the desired contents. The function will check the dataframe for validity and then rewrite the toml file using the dataframe as a blueprint.

Usage

```
edit_toml(original_toml_path, new_toml_dataframe)
```

Arguments

`original_toml_path`	Path to the original toml file.
`new_toml_dataframe`	Dataframe of the new toml file contents with the following columns: `type`, `name`, `order`, `skip_if_fail`, `created`.

Value

invisible original toml file path to toml file

`new_toml_dataframe`

`edit_toml()` will check the dataframe for the following columns:

- `type`: The type of the toml file, a character that is either "strata" or "laminae"
- `name`: The character string that is the name of the stratum or lamina
- `order`: The numeric order of the stratum or lamina
- `skip_if_fail`: (if type == laminae) A logical indicating if the lamina should be skipped if it fails
- `created`: A valid date that is the day the stratum or lamina was created

Unexpected columns will be dropped, and `edit_toml()` will warn the user. If there are any missing columns, `edit_toml()` will return an error, stop and inform the user what is missing.

If there are duplicates in the `order` than `strata` will rewrite the order using its best guess.

usage

Users using this function will likely want to combine some of the other helpers in `strata`. This may looks something like this:

- User runs `survey_tomls()` to find all the toml files in the project

- User runs `view_toml()` to view the contents of the toml file and saves to an object, like `original_toml` or similiar
- User edits the `original_toml` object to their liking and saves as a new object, like `new_toml`.
- User runs `edit_toml()` with the path to the original toml and `new_toml` objects and can then use `view_toml()` to confirm the changes.

Examples

```
tmp <- fs::dir_create(fs::file_temp())
strata::build_quick_strata_project(tmp, 2, 3)
original_toml_path <- survey_tomls(tmp)[[1]]
original_toml <- view_toml(original_toml_path)
original_toml
new_toml <- original_toml |>
  dplyr::mutate(
    created = as.Date("2021-01-01")
  )
new_toml_path <- edit_toml(original_toml_path, new_toml)
view_toml(new_toml_path)
fs::dir_delete(tmp)
```

`log_error` *Wrapper around log_message for ERROR messages in the log*

Description

`log_error()` does *not* stop the execution of the script, but it does print the message to stderr.

Usage

```
log_error(message)
```

Arguments

`message`	A string containing a message to log.

Value

A message printed to stderr

Examples

```
log_error("This is an error message")
```

`log_message` *Send a standardized log message to stdout or stderr*

Description

`log_message()` does *not* stop the execution of the script, regardless of the level of the message, and whether or not it prints to STDOUT or STDERR.

Usage

```
log_message(message, level = "INFO", out_or_err = "OUT")
```

Arguments

`message`	A string containing a message to log.
`level`	The level of the message (e.g. INFO, WARNING, ERROR), defaults to "INFO" but will accept any string.
`out_or_err`	Send log output to stdout or stderr, choices are `"OUT"` or `"ERR"` and the defaults is `"OUT"`.

Value

A message printed to stdout or stderr.

Examples

```
log_message("This is an info message", "INFO", "OUT")
log_message("This is an error message", "ERROR", "ERR")
log_message("This is a warning message", "WARNING", "OUT")
```

`log_total_time` *Print time difference in a standard message for logging purposes*

Description

Print time difference in a standard message for logging purposes

Usage

```
log_total_time(begin, end)
```

Arguments

`begin`	A data-time object, signifying the beginning or a process
`end`	A data-time object, signifying the end of a process

Value

A numeric value of the time difference in seconds

Examples

```
begin <- Sys.time()
# do something
end <- Sys.time() + 999
log_total_time(begin, end)
```

`main` *Execute entire strata project*

Description

`main()` will read the `.toml` files inside the `project_path` and begin sourcing the `strata` and `laminae` in the order specified by the user, with or without logging messages.

When a strata project is created `main.R` is added to the project root. This script houses `main()`, and this file is the entry point to the project and should be the target for automation. However, `main()` can be called from anywhere, and users can opt to not use `main.R` at all.

Usage

```
main(project_path, silent = FALSE)
```

Arguments

`project_path`	A path to strata project folder.
`silent`	Suppress log messages? If `FALSE` (the default), log messages will be printed to the console. If `TRUE`, log messages will be suppressed.

Value

invisible execution plan.

.toml files

There are two types of `.toml` files that `main()` will read:

- `.strata.toml` - a singular file inside the `<project_path>/strata` folder
- `.laminae.toml` - a file inside each `<project_path>/strata/<stratum_name>` folder

These files are created by the `strata` functions and are used to determine primarily the order of execution for the strata and laminae. Anything not referenced by a .toml will be ignored by `main()` and other functions such as `survey_strata()`, `adhoc_stratum()`, and `adhoc_lamina()`. Users can safely add other folders and files in the project root, and even within the subfolders and they will be ignored, unless users have code known by a `.toml` that references them.

Users can use the functions `survey_tomls()` and [view_toml())] to find and view the `.toml` files in their project.

[view_toml())]: R:view_toml())

Examples

```
tmp <- fs::dir_create(fs::file_temp())
result <- strata::build_quick_strata_project(tmp, 1, 1)
main(tmp)
fs::dir_delete(tmp)
```

`survey_strata` *Survey the layout and execution order of your project*

Description

`survey_strata()` will examine the .tomls in `project_path` provided and return a dataframe with the following information about the project:

- `stratum_name`: the name of the stratum
- `lamina_name`: the name of the lamina
- `execution_order`: the order in which the stratum-lamina-code combination will be executed
- `script_name`: the name of the script to be executed
- `script_path`: the path to the script

This is based on the contents of the .toml files, everything else is "invisible" inside the strata project.

Usage

```
survey_strata(project_path)
```

Arguments

`project_path`	A path to strata project folder.

Value

dataframe housing the layout of your project based on the .tomls.

Examples

```
tmp <- fs::dir_create(fs::file_temp())
build_quick_strata_project(tmp, 2, 2)
survey_strata(tmp)
fs::dir_delete(tmp)
```

survey_tomls — *Find all toml files in a project*

Description

Find all toml files in a project

Usage

```
survey_tomls(project_path)
```

Arguments

`project_path`	A path to strata project folder.

Value

an fs_path object of all toml files.

Examples

```
tmp <- fs::dir_create(fs::file_temp())
strata::build_quick_strata_project(tmp, 2, 3)
survey_tomls(tmp)
fs::dir_delete(tmp)
```

view_toml — *View the contents of a toml file as a dataframe*

Description

View the contents of a toml file as a dataframe

Usage

```
view_toml(toml_path)
```

Arguments

`toml_path`	Path to the toml file

Value

a dataframe of the toml file contents.

Examples

```
tmp <- fs::dir_create(fs::file_temp())
strata::build_quick_strata_project(tmp, 2, 3)
proj_tomls <- survey_tomls(tmp)
purrr::map(proj_tomls, view_toml)
fs::dir_delete(tmp)
```

Index