Package 'gitdown'

October 13, 2022

Title Turn Your Git Commit Messages into a HTML Book

Version 0.1.6

Description Read all commit messages of your local git repository and sort them according to tags or specific text pattern into chapters of a HTML book using 'bookdown'. The git history book presentation helps organisms required to testify for every changes in their source code, in relation to features requests.

License MIT + file LICENSE

URL https://thinkr-open.github.io/gitdown/, https://github.com/Thinkr-open/gitdown

BugReports https://github.com/Thinkr-open/gitdown/issues

Depends R (>= 3.4)

Imports attempt, bookdown, dplyr, git2r (>= 0.26.0), knitr, magrittr, purrr, rmarkdown, stats, stringi, tidyr, utils

Suggests testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.1.2

NeedsCompilation no

Author Sébastien Rochette [aut, cre] (<https://orcid.org/0000-0002-1565-9313>), Cervan Girard [aut] (<https://orcid.org/0000-0002-4816-4624>), ThinkR [cph], Institut de Recherches Internationales Servier [spn]

Maintainer Sébastien Rochette <sebastien@thinkr.fr>

Repository CRAN

Date/Publication 2022-03-05 20:40:02 UTC

R topics documented:

`create_vignette_last_modif` *Creates and updates a vignette in the 'vignette/' directory of a package with last modification time of tracked files*

Description

Creates and updates a vignette in the 'vignette/' directory of a package with last modification time of tracked files

Usage

```
create_vignette_last_modif(
  repo = ".",
  path = "R",
  recursive = TRUE,
  untracked = TRUE
)

update_vignette_last_modif(
  repo = ".",
  path = "R",
  recursive = TRUE,
  untracked = TRUE
)
```

Arguments

`repo`	git repository
`path`	Default to R folder. Use "" for the complete directory
`recursive`	Logical. Should the listing recurse into directories?
`untracked`	Logical. Should the not tracked files be included?

Value

No return value, called for side effect. Creates and updates the content of the "modification_files.Rmd" vignette

Examples

```
# Creates vignette
repo <- fake_repo(as.package = TRUE)
if (rmarkdown::pandoc_available("2.0.0")) {
  create_vignette_last_modif(repo, path = "R")
}
# update vignette
repo <- fake_repo(as.package = TRUE)
if (rmarkdown::pandoc_available("2.0.0")) {
  update_vignette_last_modif(repo, path = "R")
  rmarkdown::render(file.path(repo, "vignettes", "modification_files.Rmd"))
}
```

---

each_pattern *Create the text to add in a markdown file to present each pattern as a chapter of the book*

Description

Create the text to add in a markdown file to present each pattern as a chapter of the book

Usage

```
each_pattern(nest_commits, pattern.type)
```

Arguments

nest_commits	commits as nested with nest_commits_by_pattern
pattern.type	Character name of the pattern to filter

Value

A tibble with a row for each different pattern found and a 'text' column to be included in a markdown file:

- pattern.content: pattern found in the commit message
- link_pattern: internal url of the pattern in the future HTML gitbook
- text: list of vectors of markdown text to present commits of each pattern in the HTML gitbook output
- pattern.type: name of the pattern found in the commit message

- pattern.title: pattern.content or title used in the pattern.table a nested 'data' column with all related commits
- pattern_numeric: extraction of numeric value in pattern.content
- data: a nested list of tibbles with commits content as issued from `get_commits_pattern()`

Examples

```
repo <- fake_repo()
res_commits <- nest_commits_by_pattern(
  repo,
  pattern = c("Tickets" = "ticket[[:digit:]]+"),
  ref = "main", silent = TRUE
)
each_pattern(res_commits, "Tickets")
```

`fake_repo` *Create a fake git repository in a specific folder*

Description

A fake repository allows to create reproducible examples for this package functions

Usage

```
fake_repo(path = tempfile(pattern = "git2r-"), as.package = FALSE)
```

Arguments

`path`	Path to fake repository
`as.package`	Logical. Whether to add R/ and vignettes/ directories to fake a package

Value

Character. Path of a fake repository used for reproducible examples. Fake repository contains a few files with an initiated git repository.

Examples

```
# Fake repository with git
fake_repo()
# Fake repository that looks like package with git
fake_repo(as.package = TRUE)
```

`get_commits_pattern` *Get commits associated with a text pattern*

Description

Get commits associated with a text pattern

Usage

```
get_commits_pattern(
  repo = ".",
  pattern = c(Ticket = "#[[:digit:]]+"),
  pattern.table = NULL,
  ref = "main",
  path = NULL,
  silent = FALSE
)
```

Arguments

`repo`	a path to a repository or a `git_repository` object. Default is '.'
`pattern`	Named vector with regex pattern to expose commits, like `c("Issues" = "#\[\[:digit:\]\]")` for issues
`pattern.table`	data.frame with two columns: pattern and description of the pattern. This is used as correspondence table to add some names to existing patterns.
`ref`	The name of a reference to list commits from e.g. a tag or a branch. The default is NULL for the current branch.
`path`	The path to a file. If not NULL, only commits modifying this file will be returned. Note that modifying commits that occurred before the file was given its present name are not returned; that is, the output of `git log` with `--no-follow` is reproduced.
`silent`	Logical. Whether to hide messages.

Value

A tibble with one line for each commit, duplicated if associated with multiple patterns and the following columns:

- sha: sha of the commit
- summary: First line of the commit message
- message: Full content of the commit message
- author: author of the commit
- email: email of the author
- when: commit time

- order: order of commit messages. 1 is the oldest.
- tag.name: name of tag associated with all commits since the last tag
- tag.message: message of the tagged commit
- pattern.type: name of the pattern found in the commit message
- pattern.content: pattern found in the commit message

Examples

```
repo <- fake_repo()
get_commits_pattern(repo = repo, pattern = c("Ticket" = "#[[:digit:]]+"))
get_commits_pattern(repo = repo,
  pattern = c("Ticket" = "ticket[[:digit:]]+", "Issues" = "#[[:digit:]]+"))
```

---

get_commits_tags *Get commits associated chronologically with tags*

Description

Get commits associated chronologically with tags

Usage

```
get_commits_tags(repo = ".", ref = "main", path = NULL, silent = FALSE)
```

Arguments

repo	a path to a repository or a `git_repository` object. Default is '.'
ref	The name of a reference to list commits from e.g. a tag or a branch. The default is NULL for the current branch.
path	The path to a file. If not NULL, only commits modifying this file will be returned. Note that modifying commits that occurred before the file was given its present name are not returned; that is, the output of `git log` with `--no-follow` is reproduced.
silent	Logical. Whether to hide messages.

Value

A tibble with one line for each commit and the following columns:

- sha: sha of the commit
- summary: First line of the commit message
- message: Full content of the commit message
- author: author of the commit
- email: email of the author
- when: commit time

- order: order of commit messages. 1 is the oldest.
- tag.name: name of tag associated with all commits since the last tag
- tag.message: message of the tagged commit

Examples

```
repo <- fake_repo()
get_commits_tags(repo = repo)
```

`get_info` *Get the first and last modification time for a specific file, based on git2r::blame().*

Description

Get the first and last modification time for a specific file, based on git2r::blame().

Usage

```
get_info(path, repo = ".")
```

Arguments

`path`	path to the file
`repo`	repo of the git project

Value

A list with information of the selected file:

- file: file name
- in_repository: Logical. Whether the file has already been commit once in the git repository
- first_modif: time of first modification. Commit time if in the git repository, system date of creation otherwise.
- last_modif: time of last modification. Commit time if in the git repository, system date of last modification otherwise.

Examples

```
repo <- fake_repo()
get_info(list.files(repo)[1], repo = repo)
```

get_last_modif *Get the first and last modification time of files of a directory*

Description

Get the first and last modification time of files of a directory

Usage

```
get_last_modif(repo = ".", path = "R", recursive = TRUE, untracked = TRUE)
```

Arguments

repo	git repository
path	Default to R folder. Use "" for the complete directory
recursive	Logical. Should the listing recurse into directories?
untracked	Logical. Should the not tracked files be included?

Value

A list of files with information of each file:

- file: file name
- in_repository: Logical. Whether the file has already been commit once in the git repository
- first_modif: time of first modification. Commit time if in the git repository, system date of creation otherwise.
- last_modif: time of last modification. Commit time if in the git repository, system date of last modification otherwise.

Examples

```
repo <- fake_repo()
# Complete repository
get_last_modif(repo = repo, path = "")
repo <- fake_repo(as.package = TRUE)
# Complete repository
get_last_modif(repo = repo, path = "")
```

git_down *Turns the active branch history of git into a bookdown.*

Description

Read all commit messages of your local git repository and sort them according to tags or specific text pattern into chapters of a HTML book using 'bookdown'. Each chapter is a group of commits. The first page gives a summary of all the groups.

Usage

```
git_down(
  repo = ".",
  book_path = "gitdown",
  open = TRUE,
  author = "John Doe",
  pattern = c(Issues = "#[[:digit:]]+"),
  pattern.table = NULL,
  ref = "main",
  ...
)
```

Arguments

repo	The path to a repository. Default is .
book_path	The path to the bookdown output. Default is "gitdown".
open	Should the bookdown be opened once compiled? Default is TRUE.
author	Author of the bookdown
pattern	Named vector with regex pattern to expose commits, like c("Issues" = "#\[\[:digit:\]\]") for issues
pattern.table	data.frame with two columns: pattern and description of the pattern. This is used as correspondence table to add some names to existing patterns.
ref	the name of the branch, by default main
...	Other parameters to pass to rmarkdown::render()

Value

Path of the HTML gitbook saved in the repo/book_path directory.

Examples

```
repo <- fake_repo()
if (rmarkdown::pandoc_available("2.0.0")) {
 res <- git_down(repo, pattern = c("Tickets" = "ticket[[:digit:]]+", "Issues" = "#[[:digit:]]+"),
    open = FALSE)
}
```

```
## Not run:
# Open the book
  browseURL(res)

## End(Not run)
# With table of correspondence
pattern.table <- data.frame(number = c("#2", "#1"),
  title = c("#2 A second issue to illustrate a blog post",
                       "#1 An example of issue"))
if (rmarkdown::pandoc_available("2.0.0")) {
  res <- git_down(repo, pattern = c("Issues" = "#[[:digit:]]+"),
    pattern.table = pattern.table, open = FALSE)
}
## Not run:
# Open the book
  browseURL(res)

## End(Not run)
```

`nest_commits_by_pattern` *Nest all commits by each pattern found in the commit messages*

Description

Nest all commits by each pattern found in the commit messages

Usage

```
nest_commits_by_pattern(
  repo,
  pattern = c(Issues = "#[[:digit:]]+"),
  pattern.table = NULL,
  ref = "main",
  silent = TRUE
)
```

Arguments

`repo`	a path to a repository or a `git_repository` object. Default is '.'
`pattern`	Named vector with regex pattern to expose commits, like `c("Issues" = "#\[\[:digit:\]\]")` for issues
`pattern.table`	data.frame with two columns: pattern and description of the pattern. This is used as correspondence table to add some names to existing patterns.
`ref`	The name of a reference to list commits from e.g. a tag or a branch. The default is NULL for the current branch.
`silent`	Logical. Whether to hide messages.

Value

A tibble with a row for each different pattern found in commit messages and following columns:

- pattern.type: name of the pattern found in the commit message
- pattern.content: pattern found in the commit message
- pattern.title: pattern.content or title used in the pattern.table a nested 'data' column with all related commits
- pattern_numeric: extraction of numeric value in pattern.content
- link_pattern: internal url of the pattern in the future HTML gitbook
- data: a nested list of tibbles with commits content as issued from `get_commits_pattern()`

Examples

```
repo <- fake_repo()
nest_commits_by_pattern(repo)

# With table of correspondence
pattern.table <- data.frame(
  number = c("#2", "#1", "#1000"),
  title = c("#2 A second issue to illustrate a blog post",
            "#1 An example of issue",
            "#1000 issue with no commit"))
 nest_commits_by_pattern(
  repo,
  pattern.table = pattern.table,
  pattern = c("Tickets" = "ticket[[:digit:]]+", "Issues" = "#[[:digit:]]+"))
```

---

present_files *Presenting results of files and last modification time in a printed table using 'kable()'*

Description

Presenting results of files and last modification time in a printed table using 'kable()'

Usage

```
present_files(repo = ".", path = "R", recursive = TRUE, untracked = TRUE)
```

Arguments

repo	git repository
path	Default to R folder. Use "" for the complete directory
recursive	Logical. Should the listing recurse into directories?
untracked	Logical. Should the not tracked files be included?

Value

A 'kable()' output to be included in a markdown file

Examples

```
repo <- fake_repo(as.package = TRUE)
cat(present_files(repo))
```

Index