Package 'gitr'

February 15, 2023

Title A Lightweight API for 'Git'

Version 0.0.1

Description A light-weight, dependency-free, application programming interface (API) to access system-level 'Git' commands from within 'R'. Contains wrappers and defaults for common data science workflows as well as 'Zsh' plugin aliases. A generalized API syntax is also available. A system installation of 'Git' <https://git-scm.com/downloads> is required.

License MIT + file LICENSE

URL https://stufield.github.io/gitr/

BugReports https://github.com/stufield/gitr/issues

Depends R (>= 4.1.0)

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Encoding UTF-8

LazyLoad true

Copyright Stu Field 2023

Config/testthat/edition 3

RoxygenNote 7.2.3

Language en-US

Collate 'params.R' 'gitr-package.R' 'git.R' 'git-branch.R' 'git-commit.R' 'git-lint-commit.R' 'git-pull-request.R' 'git-sitrep.R' 'git-tag.R' 'trim-sha.R' 'utils.R' 'zsh.R'

NeedsCompilation no

Author Stu Field [aut, cre, cph] (<https://orcid.org/0000-0002-1024-5859>)

Maintainer Stu Field <stu.g.field@gmail.com>

Repository CRAN

Date/Publication 2023-02-15 12:50:10 UTC

R topics documented:

`branch` *Git Branch Utilities*

Description

Git Branch Utilities

Usage

```
git_default_br()

git_current_br()
```

Value

Character. The name of the respective branch if found, otherwise `NULL`.

Functions

- `git_default_br()`: gets the default "main" branch, typically either `master`, `main`, or `trunk`.
- `git_current_br()`: gets the *current* branch.

commit *Git Commit Utilities*

Description

Git Commit Utilities

Usage

```
get_commit_msgs(sha = NULL, n = 1)

scrape_commits(n)

git_unstage(file = NULL)

git_reset_soft(n = 1)

git_uncommit()

git_reset_hard()

git_diffcommits(top = 1, n = 2)
```

Arguments

sha	Character. The commit SHA-1 hash to pull messages from. If NULL, the most recent commit on the current branch.
n	Numeric. How far back to go from current HEAD. Same as the command line git log -n parameter. For git stash commands, zero-index into the stash list.
file	Character. The path name to a file.
top	Numeric. The commit to consider the "top" of the commit stack. Defaults to HEAD or n = 1.

Value

NULL ... invisibly.

A list containing commit message entries. The sha and author of each commit is added as attributes.

Functions

- get_commit_msgs(): gets the commit messages corresponding to the commit sha.
- scrape_commits(): scrapes n commit messages for useful change log commits to be used to create a NEWS.md.

- `git_unstage()`: un-stages a file from the index to the working directory. Default un-stages *all* files.
- `git_reset_soft()`: un-commits the most recently committed file(s) and add them to the staging area.
- `git_uncommit()`: un-commits the most recently committed file(s) and add them to the staging area. Wrapper around `git_reset_soft()`
- `git_reset_hard()`: `git reset --hard origin/<branch>`.
- `git_diffcommits()`: gets the diff of the corresponding 2 commits. Order matters.

`git` *Git Utilities*

Description

Provides functionality for system-level git commands from within R.

Usage

```
git(..., echo_cmd = TRUE)

is_git()

git_version()

git_checkout(branch = NULL)
```

Arguments

`...`	Additional arguments passed to the system command-line `git <command> [<args>]` call.
`echo_cmd`	Logical. Whether to print the command to run to the console.
`branch`	Character. The name of a branch, typically a feature branch.

Value

`git()`: The system call ... invisibly.

`is_git()`: Logical.

`git_version()`: Character. The system version of `git`.

`git_checkout()`: `NULL` ... invisibly.

Functions

- `git()`: executes a `git` command line call from within R.
- `is_git()`: is current working directory a `git` repository?
- `git_version()`: gets the version of git in use.
- `git_checkout()`: `git checkout` as a branch if doesn't exist. Branch oriented workflow for switching between branches.

Examples

```
## Not run:
git("status", "-s")

get_commit_msgs()

get_commit_msgs(n = 3)

get_pr_msgs()

# lint most recent 3 commit message
lapply(get_commit_msgs(n = 3), lint_commit_msg)

# for a PR `branch` -> `remotes/origin/{main,master}`
lapply(get_pr_msgs(), lint_commit_msg)           # current branch
lapply(get_pr_msgs("feature"), lint_commit_msg)  # `feature` branch

get_recent_tag()

## End(Not run)
```

---

git_sitrep *Git Situation Report*

Description

Get a situation report of the current git repository.

Usage

```
git_sitrep()
```

Value

NULL ... invisibly.

---

lint *Common Lints for Commit Messages*

Description

Lint a commit message for typical commit style and best practices for git.

Usage

```
lint_commit_msg(x)
```

Arguments

x	A single commit message from `get_commit_msgs()`.

Value

Integer. Invisibly returns the number of detected lints in the message.

---

params *Common Parameters for gitr*

Description

Common Parameters for gitr

Arguments

n	Numeric. How far back to go from current `HEAD`. Same as the command line `git log -n` parameter. For `git stash` commands, zero-index into the stash list.
file	Character. The path name to a file.
branch	Character. The name of a branch, typically a feature branch.
sha	Character. The commit SHA-1 hash to pull messages from. If `NULL`, the most recent commit on the current branch.

---

pr *Git PR Utilities*

Description

Git PR Utilities

Usage

```
get_pr_msgs(branch = NULL)

get_pr_sha(branch = NULL)
```

Arguments

branch	Character. The name of a branch, typically a feature branch.

Value

`get_pr_msgs()`: see `get_commit_msgs()`.

`get_pr_sha()`: character vector of shas corresponding to the PR (relative to the default branch).

Functions

- `get_pr_msgs()`: gets the commit messages for the *current* branch relative to the `origin/{main,master}` branch in the remote. Typically these "new" commits that would be merged as part of a PR to `origin/{main,master}`.
- `get_pr_sha()`: gets the commit SHA1 *current* branch relative to the `default` branch in the remote, usually either `origin/main` or `origin/master`. See `git_default_br()`.

---

sha *SHA1 Utilities*

Description

SHA1 Utilities

Usage

```
trim_sha(sha)
```

Arguments

sha	Character. The commit SHA-1 hash to pull messages from. If `NULL`, the most recent commit on the current branch.

Value

Character. The trimmed `sha`.

Functions

- `trim_sha()`: trims the `SHA-1` hash from the default full length to the human-readable short version.

---

tag *Git Tag Utilities*

Description

Git Tag Utilities

Usage

```
git_recent_tag()

git_tag_info()
```

Value

`git_recent_tag()`: Character. The most recent tag.

`git_tag_info()`: A data frame summarizing the repository tags.

Functions

- `git_recent_tag()`: gets the *most* recent `git` tag.
- `git_tag_info()`: gets a data frame summary of the current git repository tags.

`zsh` *Z-shell Aliases*

Description

Provides functions to common Z-shell git plugin aliases.

Usage

```
glog(n = 10)

gcc(...)

gcmsg(msg = "wip")

gco(branch = NULL)

gcb(branch = NULL)

gpr()

gp(...)

gpu()

gpd()

gst()

gss()

gba()

gbd(branch = NULL, force = FALSE)

gbmm(branch = git_default_br())
```

```
gbnm(branch = git_default_br())

gbm(branch = NULL)

ga(...)

gaa()

gau()

gsta()

gstl()

gstaa(n = 0)

gstd(n = 0)

gstc()

gsts(text = FALSE)

gpop()

gstp()

gtn()

gfa()

gac()

gwip()

gclean(dry.run = TRUE)

gdf(file = NULL, staged = FALSE)

gpf()

gnuke()

gcf(global = FALSE)

gcm()

grm(...)
```

`grbc()`

`grba()`

`grbs()`

`grbm()`

`grv()`

Arguments

`n`	Numeric. How far back to go from current `HEAD`. Same as the command line `git log -n` parameter. For `git stash` commands, zero-index into the stash list.
`...`	Additional arguments passed to the system command-line `git <command> [<args>]` call.
`msg`	Character. The message for the commit subject line.
`branch`	Character. The name of a branch, typically a feature branch.
`force`	Logical. Should the branch delete be forced with the `-D` flag?
`text`	Logical. Show the text diffs from the stash.
`dry.run`	Logical. Clean as dry-run?
`file`	A full file path within the repository to diff.
`staged`	Logical. Compare a staged file to HEAD? Otherwise the working directory is compared to the index (staged or HEAD).
`global`	Logical. Query global repository. Alternatively local configuration only.

Value

Most aliases invisibly return `NULL` ... with some exceptions.

Functions

- `glog()`: Get the `git` log in a pretty format for the `n` most recent commits.
- `gcc()`: `git commit ...`. To avoid masking the `base::gc()` function, this alias has been re-mapped to `gcc()`.
- `gcmsg()`: `git commit -m <msg>`.
- `gco()`: `git checkout`.
- `gcb()`: `git checkout -b <branch>`.
- `gpr()`: `git pull --rebase`.
- `gp()`: `git push`.
- `gpu()`: `git push -u origin`.
- `gpd()`: `git push --dry-run`.
- `gst()`: `git status`.

- gss(): git status -s.
- gba(): git branch -a.
- gbd(): git branch -dD.
- gbmm(): git branch --merged <branch>.
- gbnm(): git branch --no-merged <branch>.
- gbm(): git branch -m.
- ga(): git add ..
- gaa(): git add --all.
- gau(): git add -u.
- gsta(): git stash.
- gstl(): git stash list.
- gstaa(): git stash apply. **Note**: zero-indexing!
- gstd(): git stash drop. **Note**: zero-indexing!
- gstc(): git stash clear. Danger!
- gsts(): git stash show.
- gpop(): git stash pop --quiet --index.
- gstp(): See gpop().
- gtn(): git tag -n.
- gfa(): git fetch --all --prune.
- gac(): git commit --no-verify --amend --no-edit.
- gwip(): git commit --no-verify -m 'wip'.
- gclean(): git clean -f -d.
- gdf(): git diff <file>.
- gpf(): git push --force-with-lease.
- gnuke(): git reset --hard && git clean -df.
- gcf(): git config --local or git config --global.
- gcm(): Checkout the default branch.
- grm(): git rm ..
- grbc(): git rebase --continue.
- grba(): git rebase --abort.
- grbs(): git rebase --skip.
- grbm(): git rebase git_default_br().
- grv(): git remote -v.

Examples

```
## Not run:
  glog()

## End(Not run)
```

Index