Package 'precommit'

October 10, 2020

Title Pre-Commit Hooks

Version 0.1.3

Author Lorenz Walthert

Maintainer Lorenz Walthert <lorenz.walthert@icloud.com>

Description Useful git hooks for R building on top of the multi-language framework 'pre-commit' for hook management. This package provides git hooks for common tasks like formatting files with 'styler' or spell checking as well as wrapper functions to access the 'pre-commit' executable.

License GPL-3

URL https://lorenzwalthert.github.io/precommit/, https://github.com/lorenzwalthert/precommit

Imports docopt, fs, here, magrittr, purrr, R.cache, rlang, rprojroot, rstudioapi, usethis (>= 1.6.0), withr, yaml

Suggests desc, git2r, glue, knitr, lintr, reticulate (>= 1.14), rmarkdown, roxygen2, spelling, styler (>= 1.2), testthat (>= 2.1.0), versions

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2020-10-10 19:40:02 UTC

R topics documented:

`autoupdate` *Auto-update your hooks*

Description

Runs pre-commit autoupdate.

Usage

```
autoupdate(root = here::here())
```

Arguments

`root`	The path to the root directory of your project.

Value

The exit status from pre-commit autoupdate (invisibly).

Examples

```
## Not run:
autoupdate()

## End(Not run)
```

`diff_requires_run_roxygenize`
Check if we should run roxygen.

Description

This is the case if a new or replaced/removed line contains a roxygen2 comment in a file that is staged. This is only exported because it's used in a hook script: It's not intended to be called by users of precommit directly.

Usage

```
diff_requires_run_roxygenize(root = here::here())
```

Arguments

root	The root of project.

Value

A logical vector of length 1.

Examples

```
## Not run:
diff_requires_run_roxygenize()

## End(Not run)
```

---

install_precommit *Install pre-commit on your system.*

Description

This installs pre-commit in the conda environment r-precommit. It will be available to use across different git repositories.

Usage

```
install_precommit(force = FALSE)
```

Arguments

force	Whether or not to force a re-installation.

Value

The path to the pre-commit executable (invisibly).

Examples

```
## Not run:
install_precommit()

## End(Not run)
```

not_conda *The testing environment does not use a conda environment if the env variable PRECOMMIT_INSTALLATION_METHOD is not 'conda'.*

Description

The testing environment does not use a conda environment if the env variable PRECOMMIT_INSTALLATION_METHOD is not 'conda'.

Usage

```
not_conda()
```

open_config *Open pre-commit related files*

Description

Open pre-commit related files

Usage

```
open_config(root = here::here())

open_wordlist(root = here::here())
```

Arguments

root	The path to the root directory of your project.

Details

- `open_config()`: opens the pre-commit config file.
- `open_wordlist()`: opens the the WORDLIST file for the check-spelling hook in inst/WORDLIST.

Value

`NULL` (invisibly). The function is called for its side effects.

See Also

Other helpers: `use_precommit()`

Examples

```
## Not run:
open_config()

## End(Not run)
## Not run:
open_wordlist()

## End(Not run)
```

`path_precommit_exec` *Locate the pre-commit executable*

Description

`path_precommit_exec()` simply reads the R option `precommit.executable`, `path_pre_commit_exec()` is the old spelling and deprecated.

Usage

```
path_precommit_exec(check_if_exists = TRUE)

path_pre_commit_exec(check_if_exists = TRUE)
```

Arguments

`check_if_exists`	Whether or not to make sure the returned path also exists.

Value

A character vector of length one with the path to the pre-commit executable.

See Also

`path_derive_precommit_exec()` for the heuristic to derive it from scratch.

Examples

```
## Not run:
path_precommit_exec()

## End(Not run)
## Not run:
path_pre_commit_exec()

## End(Not run)
```

uninstall_precommit *Uninstall pre-commit*

Description

Remove pre-commit from a repo or from your system.

Usage

```
uninstall_precommit(scope = "repo", ask = "user", root = here::here())
```

Arguments

scope	Either "repo" or "user". "repo" removes pre-commit from your project, but you will be able to use it in other projects. With "user", you remove the pre-commit executable in the virtual python environment r-precommit so it won't be available in any project. When you want to do the latter, you should first do the former.
ask	Either "user", "repo" or "none" to determine in which case a prompt should show up to let the user confirm his action.
root	The path to the root directory of your project.

Value

NULL (invisibly). The function is called for its side effects.

Examples

```
## Not run:
uninstall_precommit()

## End(Not run)
```

use_precommit *Set up pre-commit*

Description

Get started.

Usage

```
use_precommit(
  config_source = getOption("precommit.config_source"),
  force = FALSE,
  legacy_hooks = "forbid",
  open = rstudioapi::isAvailable(),
  install_hooks = TRUE,
  root = here::here()
)
```

Arguments

config_source	Path or URL to a .pre-commit-config.yaml. This config file will be hard-copied into root. If NULL, we check if root is a package or project directory using rprojroot::find_package_root_file(), and resort to an appropriate default config. See section 'Copying an existing config file'.
force	Whether to replace an existing config file.
legacy_hooks	How to treat hooks already in the repo which are not managed by pre-commit. "forbid", the default, will cause use_precommit() to fail if there are such hooks. "allow" will run these along with pre-commit. "remove" will delete them.
open	Whether or not to open the .pre-commit-config.yaml after it's been placed in your repo. The default is TRUE when working in RStudio. Otherwise, we recommend manually inspecting the file.
install_hooks	Whether to install environments for all available hooks. If FALSE, environments are installed with first commit.
root	The path to the root directory of your project.

Value

NULL (invisibly). The function is called for its side effects.

When to call this function?

- You want to add pre-commit support to a git repo which does not have a .pre-commit-config.yaml. This involves adding a pre-commit config file and making sure git will call the hooks before the next commit.
- You cloned a repo that has a .pre-commit-config.yaml already. You need to make sure git calls the hooks before the next commit.

What does the function do?

- Sets up a template .pre-commit-config.yaml.
- Autoupdates the template to make sure you get the latest versions of the hooks.
- Installs the pre-commit script along with the hook environments with $ pre-commit install --install-hooks.
- Opens the config file if RStudio is running.

Copying an existing config file

You can use an existing `.pre-commit-config.yaml` file when initializing pre-commit with `use_precommit()` using the argument `config_source` to copy an existing config file into your repo. This argument defaults to the R option `precommit.config_source`, so you may want to set this option in your `.Rprofile` for convenience. Note that this is **not** equivalent to the `--config` option in the CLI command pre-commit install and similar, which do *not* copy a config file into a project root (and allow to put it under version control), but rather link it in some more or less transparent way.

See Also

Other helpers: `open_config()`

Examples

```
## Not run:
use_precommit()

## End(Not run)
```

---

use_precommit_config *Initiate a pre-commit config file*

Description

Initiate a pre-commit config file

Usage

```
use_precommit_config(
  config_source = getOption("precommit.config_source"),
  force = FALSE,
  open = rstudioapi::isAvailable(),
  verbose = FALSE,
  root = here::here()
)
```

Arguments

`config_source`	Path or URL to a `.pre-commit-config.yaml`. This config file will be hard-copied into `root`. If `NULL`, we check if `root` is a package or project directory using `rprojroot::find_package_root_file()`, and resort to an appropriate default config. See section 'Copying an existing config file'.
`force`	Whether to replace an existing config file.
`open`	Whether or not to open the .pre-commit-config.yaml after it's been placed in your repo. The default is `TRUE` when working in RStudio. Otherwise, we recommend manually inspecting the file.
`verbose`	Whether or not to communicate what's happening.
`root`	The path to the root directory of your project.

Value

Character vector of length one with the path to the config file used.

Copying an existing config file

You can use an existing `.pre-commit-config.yaml` file when initializing pre-commit with `use_precommit()` using the argument `config_source` to copy an existing config file into your repo. This argument defaults to the R option `precommit.config_source`, so you may want to set this option in your `.Rprofile` for convenience. Note that this is **not** equivalent to the `--config` option in the CLI command pre-commit install and similar, which do *not* copy a config file into a project root (and allow to put it under version control), but rather link it in some more or less transparent way.

Examples

```
## Not run:
use_precommit_config()

## End(Not run)
```

Index