Package 'workflowr'

December 21, 2021

Type Package

Title A Framework for Reproducible and Collaborative Data Science

Version 1.7.0

Description Provides a workflow for your analysis projects by combining literate programming ('knitr' and 'rmarkdown') and version control ('Git', via 'git2r') to generate a website containing time-stamped, versioned, and documented results.

URL https://github.com/workflowr/workflowr

BugReports https://github.com/workflowr/workflowr/issues

Depends R (>= 3.3.0)

License MIT + file LICENSE

Encoding UTF-8

Imports callr (>= 3.7.0), fs (>= 1.2.7), getPass, git2r (>= 0.26.0), glue, httpuv (>= 1.2.2), httr, knitr (>= 1.29), rmarkdown (>= 1.18), rprojroot (>= 1.2), rstudioapi (>= 0.6), stringr (>= 1.3.0), tools, utils, whisker (>= 0.3-2), xfun, yaml

RoxygenNote 7.1.2

Suggests clipr (>= 0.7.0), covr, devtools, miniUI (>= 0.1.1), reticulate (>= 1.15), shiny (>= 0.14), spelling (>= 2.0), testthat (>= 2.0.0), withr (>= 2.0.0)

VignetteBuilder knitr

SystemRequirements pandoc (>= 1.14) - https://pandoc.org

Language en-US

NeedsCompilation no

Author John Blischak [aut, cre] (<https://orcid.org/0000-0003-2634-9879>), Peter Carbonetto [aut] (<https://orcid.org/0000-0003-1144-6780>), Matthew Stephens [aut] (<https://orcid.org/0000-0001-5397-9257>), Luke Zappia [ctb] (Instructions for hosting with GitLab), Pierre Formont [ctb] (Support for hosting with Shiny Server), Tim Trice [ctb] (Instructions for sharing common code),

Jiaxiang Li [ctb] (Function wflow_toc() to create table of contents),
Michael J. Kane [ctb] (<https://orcid.org/0000-0003-1899-6662>, Option suppress_report),
Anh Tran [ctb] (Updated RStudio Project Template),
Sydney Purdue [ctb] (Improved wflow_start() error handling),
Giorgio Comai [ctb] (Added argument only_published to wflow_toc())

Maintainer John Blischak <jdblischak@gmail.com>

Repository CRAN

Date/Publication 2021-12-21 07:00:02 UTC

R topics documented:

`extract_commit` *Extract a commit from a Git repository*

Description

`extract_commit` extracts the 7-digit SHA1 identifier and message for a specified commit.

Usage

```
extract_commit(path, num)
```

Arguments

`path`	character. Specify the path to a directory that is a Git repository (or any subdirectory of the Git repository).
`num`	numeric. The number of the commit to extract in reverse chronological order. In other words, 1 is the most recent commit, 2 is the second most recent commit, etc.

Value

A list with the named elements `sha1` and `message` (both characters). If a Git repository is not found at `path`, both are `NA`.

Examples

```
## Not run:
# Most recent commit
extract_commit(".", 1)
# Penultimate commit
extract_commit(".", 2)

## End(Not run)
```

`wflow_build` *Build the site*

Description

`wflow_build` builds the website from the files in the analysis directory. This is intended to be used when developing your code to preview the changes. When you are ready to commit the files, use `wflow_publish`.

Usage

```
wflow_build(
  files = NULL,
  make = is.null(files),
  update = FALSE,
  republish = FALSE,
  combine = "or",
  view = getOption("workflowr.view"),
  clean_fig_files = FALSE,
  delete_cache = FALSE,
  seed = 12345,
  log_dir = NULL,
  verbose = FALSE,
  local = FALSE,
  dry_run = FALSE,
  project = "."
)
```

Arguments

files	character (default: NULL). Files to build. Only allows files in the analysis directory with the extension Rmd or rmd. If `files` is `NULL`, the default behavior is to build all outdated files (see argument `make` below). Supports file globbing. The files are always built in the order they are listed.
make	logical (default: `is.null(files)`). When `make = TRUE`, build any files that have been modified more recently than their corresponding HTML files (inspired by Make). This is the default action if no files are specified.
update	logical (default: FALSE). Build any files that have been committed more recently than their corresponding HTML files (and do not have any unstaged or staged changes). This ensures that the commit version ID inserted into the HTML corresponds to the exact version of the source file that was used to produce it.
republish	logical (default: FALSE). Build all published R Markdown files (that do not have any unstaged or staged changes). Useful for site-wide changes like updating the theme, navigation bar, or any other setting in `_site.yml`.
combine	character (default: `"or"`). Determine how to combine the files from the arguments `files`, `make` (`wflow_build()` only), `update`, and `republish`. When `combine` is `"or"`, any file specified by at least one of these arguments will be built. When `combine` is `"and"`, only files specified by all of these arguments will be built.
view	logical (default: `getOption("workflowr.view")`). View the website with `wflow_view` after building files. If only one file is built, it is opened. If more than one file is built, the main index page is opened. Not applicable if no files are built or if `dry_run = TRUE`.
clean_fig_files	logical (default: FALSE). Delete existing figure files for each R Markdown file prior to building it. This ensures that only relevant figure files are saved. As you

develop an analysis, it is easy to generate lots of unused plots due to changes in the number of code chunks and their names. However, if you are caching chunks during code development, this could cause figures to disappear. Note that wflow_publish uses clean_fig_files = TRUE to ensure the results can be reproduced.

delete_cache — logical (default: FALSE). Delete the cache directory (if it exists) for each R Markdown file prior to building it.

seed — numeric (default: 12345). The seed to set before building each file. Passed to set.seed. **DEPRECATED:** The seed set here has no effect if you are using wflow_html as the output format defined in _site.yml. This argument is for backwards compatibility with previous versions of workflowr.

log_dir — character (default: NULL). The directory to save log files from building files. It will be created if necessary and ignored if local = TRUE. The default is to use a directory named workflowr in tempdir.

verbose — logical (default: FALSE). Display the build log directly in the R console as each file is built. This is useful for monitoring long-running code chunks.

local — logical (default: FALSE). Build files locally in the R console. This should only be used for debugging purposes. The default is to build each file in its own separate fresh R process to ensure each file is reproducible in isolation. This is done using callr::r_safe.

dry_run — logical (default: FALSE). List the files to be built, without building them.

project — character (default: ".") By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory.

Details

wflow_build has multiple, non-mutually exclusive options for deciding which files to build. If multiple options are used, then the argument combine determines which files will be built. If combine == "or" (the default), then any file specified by at least one of the arguments will be built. In contrast, if combine == "and", then only files specified by all of the arguments will be built. The argument make is the most useful for interactively performing your analysis. The other options are more useful when you are ready to publish specific files with wflow_publish (which passes these arguments to wflow_build). Here are the options for specifying files to be built:

- Files specified via the argument files
- make = TRUE - Files which have been modified more recently than their corresponding HTML files
- update = TRUE - Previously published files which have been committed more recently than their corresponding HTML files. However, files which currently have staged or unstaged changes are not included.
- republish = TRUE - All published files. However, files which currently have staged or unstaged changes are not included.

Under the hood, wflow_build is a wrapper for render_site from the package rmarkdown. By default (local = FALSE), the code is executed in an isolated R session. This is done using callr::r_safe.

Value

An object of class `wflow_build`, which is a list with the following elements:

- **files**: The input argument `files`
- **make**: The input argument `make`
- **update**: The input argument `update`
- **republish**: The input argument `republish`
- **view**: The input argument `view`
- **clean_fig_files**: The input argument `clean_fig_files`
- **delete_cache**: The input argument `delete_cache`
- **seed**: The input argument `seed`
- **log_dir**: The directory where the log files were saved
- **verbose**: The input argument `verbose`
- **local**: The input argument `local`
- **dry_run**: The input argument `dry_run`
- **built**: The relative paths to the built R Markdown files
- **html**: The relative paths to the corresponding HTML files

Comparison to RStudio Knit button

`wflow_build` is intentionally designed to be similar to clicking on the Knit button in RStudio. Both isolate the code execution in a separate R process, thus ensuring the results are not dependent on the state of the current R session. However, they do differ in a few ways:

Number of files The RStudio Knit button only builds the current Rmd file open in the editor. In contrast, `wflow_build` can build any number of Rmd files (each in their own separate R process) with a single invocation, including accepting file globs.

System and user profiles The two methods diverge the most in their use of `.Rprofile` files. `wflow_build` ignores any system or user profiles (i.e. `~/.Rprofile` on Linux/macOS or `~/Documents/.Rprofile` on Windows). This is the default behavior of `callr::r_safe`, which it calls to run the separate R process. This is ideal for reproducibility. Otherwise the results could be affected by custom settings made only on the user's machine. In contrast, the RStudio Knit button loads any system or user level profiles, consistent with its role as a development tool.

Project-specific profiles A project-specific `.Rprofile` is treated differently than system or user profiles. `wflow_build` only loads a project-specific `.Rprofile` if it is located in the current working directory in which `wflow_build` is invoked. This may be confusing if this differs from the directory in which the code in the Rmd is actually executed (the option `knit_root_dir` defined in `_workflowr.yml`). The RStudio Knit button only loads a project-specific `.Rprofile` if it is located in the same directory as its setting "Knit Directory" is configured. For example, if "Knit Directory" is set to "Document Directory", it will ignore any `.Rprofile` in the root of the project. But it would load the `.Rprofile` if "Knit Directory" was changed to "Project Directory".

The main takeaway from the above is that you should try to limit settings and options defined in `.Rprofile` to affect the interactive R experience and superficial behavior, e.g. the option `max.print` to limit the number of lines that can be printed to the console. Any critical settings that affect the results of the analysis should be explicitly set in the Rmd file.

See Also

wflow_publish

Examples

```
## Not run:

# Build any files which have been modified
wflow_build() # equivalent to wflow_build(make = TRUE)
# Build a single file
wflow_build("file.Rmd")
# Build multiple files
wflow_build(c("file1.Rmd", "file2.Rmd"))
# Build multiple files using a file glob
wflow_build("file*.Rmd")
# Build every published file
wflow_build(republish = TRUE)
# Build file.Rmd and any files which have been modified
wflow_build("file.Rmd", make = TRUE)

## End(Not run)
```

---

wflow_git_commit *Commit files*

Description

wflow_git_commit adds and commits files with Git. This is a convenience function to run Git commands from the R console instead of the shell. For most use cases, you should use wflow_publish instead, which calls wflow_git_commit and then subsequently also builds and commits the website files.

Usage

```
wflow_git_commit(
  files = NULL,
  message = NULL,
  all = FALSE,
  force = FALSE,
  dry_run = FALSE,
  project = "."
)
```

Arguments

files	character (default: NULL). Files to be added and committed with Git. Supports file globbing.

message	character (default: NULL). A commit message.
all	logical (default: FALSE). Automatically stage files that have been modified and deleted. Equivalent to: `git commit -a`
force	logical (default: FALSE). Allow adding otherwise ignored files. Equivalent to: `git add -f`
dry_run	logical (default: FALSE). Preview the proposed action but do not actually add or commit any files.
project	character (default: ".") By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory.

Details

Some potential use cases for `wflow_git_commit`:

- Commit drafts which you do not yet want to be included in the website
- Commit files which do not directly affect the website (e.g. when you are writing scripts for a data processing pipeline)
- Manually commit files in `docs/` (proceed with caution!). This should only be done for content that is not automatically generated from the source files in the analysis directory, e.g. an image file you want to include in one of your pages.

Under the hood, `wflow_git_commit` is a wrapper for `add` and `commit` from the package git2r.

Value

An object of class `wflow_git_commit`, which is a list with the following elements:

- **files**: The input argument `files`.
- **message**: The message describing the commit.
- **all**: The input argument `all`.
- **force**: The input argument `force`.
- **dry_run**: The input argument `dry_run`.
- **commit**: The object returned by git2r::`commit` (only included if `dry_run == FALSE`).
- **commit_files**: The relative path(s) to the file(s) included in the commit (only included if `dry_run == FALSE`).

See Also

`wflow_publish`

Examples

```
## Not run:

# Commit a single file
wflow_git_commit("analysis/file.Rmd", "Add new analysis")
# Commit multiple files
wflow_git_commit(c("code/process-data.sh", "output/small-data.txt"),
             "Process data set")
# Add and commit all tracked files, similar to `git commit -a`
wflow_git_commit(message = "Lots of changes", all = TRUE)

## End(Not run)
```

`wflow_git_config` *Configure Git settings*

Description

`wflow_git_config` configures the global Git settings on the current machine. This is a convenience function to run Git commands from the R console instead of the Terminal. The same functionality can be achieved by running `git config` in the Terminal.

Usage

```
wflow_git_config(user.name = NULL, user.email = NULL, ..., overwrite = FALSE)
```

Arguments

`user.name`	character (default: NULL). Git user name. Git assigns an author when committing (i.e. saving) changes. If you have never used Git before on your computer, make sure to set this.
`user.email`	character (default: NULL). Git user email. Git assigns an email when committing (i.e. saving) changes. If you have never used Git before on your computer, make sure to set this.
`...`	Arbitrary Git settings, e.g. `core.editor = "nano"`.
`overwrite`	logical (default: FALSE). Overwrite existing Git global settings.

Details

The main purpose of `wflow_git_config` is to set the user.name and user.email to use with Git commits. Note that these do not need to match the name and email you used to register your online account with a Git hosting service (e.g. GitHub or GitLab). However, it can also handle arbitrary Git settings (see examples below).

There are two main limitations of `wflow_git_config` for the sake of simplicity. First, `wflow_git_config` only affects the global Git settings that apply to all Git repositories on the local machine and is unable to configure settings for one specific Git repository. Second, `wflow_git_config` can only add

or change the user.name and user.email settings, but not delete them. To perform either of these actions, please use `git config` in the Terminal.

Under the hood, `wflow_git_config` is a wrapper for `config` from the package git2r.

To learn more about how to configure Git, see the Software Carpentry lesson Setting Up Git.

Value

An object of class `wflow_git_config`, which is a list with the following elements:

- **user.name**: The current global Git user.name
- **user.email**: The current global Git user.email
- **all_settings**: A list of all current global Git settings

Examples

```
## Not run:

# View current Git settings
wflow_git_config()
# Set user.name and user.email
wflow_git_config(user.name = "A Name", user.email = "email@domain")
# Set core.editor (the text editor that Git opens to write commit messages)
wflow_git_config(core.editor = "nano")


## End(Not run)
```

---

wflow_git_pull *Pull files from remote repository*

Description

`wflow_git_pull` pulls the remote files from your remote repository online (e.g. GitHub or GitLab) into your repository on your local machine. This is a convenience function to run Git commands from the R console instead of the Terminal. The same functionality can be achieved by running `git pull` in the Terminal.

Usage

```
wflow_git_pull(
  remote = NULL,
  branch = NULL,
  username = NULL,
  password = NULL,
  fail = TRUE,
  dry_run = FALSE,
  project = "."
)
```

Arguments

remote	character (default: NULL). The name of the remote repository. See Details for the default behavior.
branch	character (default: NULL). The name of the branch in the remote repository to pull from. If NULL, the name of the current local branch is used.
username	character (default: NULL). Username for online Git hosting service (e.g. GitHub or GitLab). The user is prompted if necessary.
password	character (default: NULL). Password for online Git hosting service (e.g. GitHub or GitLab). The user is prompted if necessary.
fail	logical (default: TRUE) Abort the pull if any merge conflicts are detected. If you are sure you want to manually cleanup the merge conflicts, set fail = FALSE. The argument fail is passed to the git2r function merge.git_repository.
dry_run	logical (default: FALSE). Preview the proposed action but do not actually pull from the remote repository.
project	character (default: ".") By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory.

Details

wflow_git_pull tries to choose sensible defaults if the user does not explicitly specify the remote repository and/or the remote branch:

- If both remote and branch are NULL, wflow_git_pull checks to see if the current local branch is tracking a remote branch. If yes, it pulls to this tracked remote branch.
- If the argument remote is left as NULL and there is only one remote, it is used. If there is more than one remote, the one named "origin" is used.
- If the argument branch is left as NULL, the name of the current local branch is used (referred to as HEAD by Git).

Under the hood, wflow_git_pull is a wrapper for pull from the package git2r.

Value

An object of class wflow_git_pull, which is a list with the following elements:

- **remote**: The remote repository.
- **branch**: The branch of the remote repository.
- **username**: Username for online Git hosting service (e.g. GitHub or GitLab).
- **merge_result**: The git_merge_result object returned by git2r (only included if dry_run == FALSE).
- **fail**: The input argument fail.
- **dry_run**: The input argument dry_run.
- **protocol**: The authentication protocol for the remote repository (either "https" or "ssh".
- **project**: The input argument project.

Examples

```
## Not run:

# Pull from remote repository
wflow_git_pull()
# Preview by running in dry run mode
wflow_git_pull(dry_run = TRUE)

## End(Not run)
```

`wflow_git_push` *Push files to remote repository*

Description

`wflow_git_push` pushes the local files on your machine to your remote repository on a remote Git hosting service (e.g. GitHub or GitLab). This is a convenience function to run Git commands from the R console instead of the Terminal. The same functionality can be achieved by running `git push` in the Terminal.

Usage

```
wflow_git_push(
  remote = NULL,
  branch = NULL,
  username = NULL,
  password = NULL,
  force = FALSE,
  set_upstream = TRUE,
  view = getOption("workflowr.view"),
  dry_run = FALSE,
  project = "."
)
```

Arguments

`remote`	character (default: NULL). The name of the remote repository. See Details for the default behavior.
`branch`	character (default: NULL). The name of the branch to push to in the remote repository. If `NULL`, the name of the current local branch is used.
`username`	character (default: NULL). Username for online Git hosting service (e.g. GitHub or GitLab). The user is prompted if necessary.
`password`	character (default: NULL). Password for online Git hosting service (e.g. GitHub or GitLab). The user is prompted if necessary.

force	logical (default: FALSE). Force the push to the remote repository. Do not use this if you are not 100% sure of what it is doing. Equivalent to: `git push -f`
set_upstream	logical (default: TRUE). Set the current local branch to track the remote branch if it isn't already tracking one. This is likely what you want. Equivalent to: `git push -u remote branch`
view	logical (default: `getOption("workflowr.view")`). Open the URL to the repository in the browser. Ignored if `dry_run = TRUE`. Also note that this only works if the option `browser` is set, which you can check with `getOption("browser")`.
dry_run	logical (default: FALSE). Preview the proposed action but do not actually push to the remote repository.
project	character (default: ".") By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory.

Details

`wflow_git_push` tries to choose sensible defaults if the user does not explicitly specify the remote repository and/or the remote branch:

- If both `remote` and `branch` are `NULL`, `wflow_git_push` checks to see if the current local branch is tracking a remote branch. If yes, it pushes to this tracked remote branch.
- If the argument `remote` is left as `NULL` and there is only one remote, it is used. If there is more than one remote, the one named "origin" is used.
- If the argument `branch` is left as `NULL`, the name of the current local branch is used (referred to as `HEAD` by Git).

Under the hood, `wflow_git_push` is a wrapper for push from the package git2r.

Value

An object of class `wflow_git_push`, which is a list with the following elements:

- **remote**: The remote repository.
- **branch**: The branch of the remote repository.
- **username**: Username for online Git hosting service (e.g. GitHub or GitLab).
- **force**: The input argument `force`.
- **set_upstream**: The input argument `set_upstream`.
- **view**: The input argument `view`.
- **dry_run**: The input argument `dry_run`.
- **protocol**: The authentication protocol for the remote repository (either `"https"` or `"ssh"`.

Examples

```
## Not run:

# Push to remote repository
wflow_git_push()
# Preview by running in dry run mode
wflow_git_push(dry_run = TRUE)

## End(Not run)
```

`wflow_git_remote` *Manage remote Git repositories*

Description

`wflow_git_remote` is a convenience function for managing remote repositories from R. By default it displays the current remote repositories (analogous to `git remote -v`). It can add a remote, remove a remote, or update the URL for an existing remote.

Usage

```
wflow_git_remote(
  remote = NULL,
  user = NULL,
  repo = NULL,
  protocol = "https",
  action = "add",
  domain = "github.com",
  verbose = TRUE,
  project = "."
)
```

Arguments

`remote`	character (default: NULL). The name of the remote.
`user`	character (default: NULL). The username for the remote repository.
`repo`	character (default: NULL). The name of the remote repository on the Git hosting service (e.g. GitHub or GitLab).
`protocol`	character (default: "https"). The protocol for communicating with the Git hosting service (e.g. GitHub or GitLab). Must be either "https" or "ssh".
`action`	character (default: "add"). The action to perform on the remotes. Must be one of "add", "remove", or "set_url". This argument is ignored if `remote = NULL`.
`domain`	character (default: "github.com"). The domain of the remote host. For example, if you want to host your Git repository at GitLab, you would specify "gitlab.com".

verbose — logical (default: TRUE). Display the current remotes. Analogous to `git remote -v`.

project — character (default: ".") By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory.

Details

`wflow_git_remote` constructs a URL to a remote repository based on the input username, repository name, protocol (https or ssh), and domain (e.g. "github.com" or "gitlab.com"). It can add a remote (`action = "add"`), remove a remote (`action = "remove"`), or update the URL for an existing remote (`action = "set_url"`).

This function cannot change the name of an existing remote. To accomplish this, you could run Git from the Terminal (`git remote rename <old> <new>`) or use `git2r::remote_rename` from R.

Value

Invisibly returns a named character vector of the remote URLs.

Examples

```
## Not run:

# Display the current remotes
wflow_git_remote()

# Add a remote called origin that points to the
# GitHub repository example_repo owned by
# the GitHub user example_user
wflow_git_remote("origin", "example_user", "example_repo")

# Remove the remote named upstream
wflow_git_remote("upstream", action = "remove")

# Change the protocol of the remote origin from https to ssh
wflow_git_remote("origin", "example_user", "example_repo", protocol = "ssh",
              action = "set_url")

# Add a remote called origin that points to the
# GitLab repository example_repo owned by
# the GitLab user example_user
wflow_git_remote("origin", "example_user", "example_repo", domain = "gitlab.com")

## End(Not run)
```

wflow_html *Convert to a workflowr HTML document*

Description

Workflowr custom format for converting from R Markdown to an HTML document. `wflow_html` has two distinct functionalities: 1) configure the formatting of the HTML by extending `html_document` (see the RStudio documentation for the available options), and 2) configure the workflowr reproducibility features (typically specified in a file named `_workflowr.yml`). `wflow_html` is intended to be used to generate webpages for a workflowr website, but it can also be used outside a workflowr project to implement reproducibility features for single R Markdown documents.

Usage

```
wflow_html(...)
```

Arguments

`...`	Arguments passed to `html_document`.

Value

An `output_format` object to pass to `render`.

HTML formatting

`wflow_html` extends `html_document`. To set default formatting options to be shared across all of your HTML files, set them in the file `analysis/_site.yml`. This special file can also be used to configure other aspects of the website like the navigation bar (for more details see the documentation on R Markdown websites). For example, to use the theme "cosmo" and add a table of contents to every webpage, you would add the following to `analysis/_site.yml`:

```
output:
  workflowr::wflow_html:
    toc: true
    theme: cosmo
```

Formatting options can also be set for a specific file, which will override the default options set in `analysis/_site.yml`. For example, to remove the table of contents from one specific file, you would add the following to the YAML header of that file:

```
output:
  workflowr::wflow_html:
    toc: false
```

However, this will preserve any of the other shared options (e.g. the theme in the above example). If you are not overriding any of the shared options, it is not necessary to specify `wflow_html` in the YAML header of your workflowr R Markdown files.

Reproducibility features

wflow_html also implements the workflowr reproducibility features. For example, it automatically sets a seed with set.seed; inserts the current code version (i.e. Git commit ID); runs sessionInfo at the end of the document; and inserts links to past versions of the file and figures.

These reproducibility options are not passed directly as arguments to wflow_html. Instead these options are specified in _workflowr.yml or in the YAML header of an R Markdown file (using the field workflowr:). These options (along with their default values) are as follows:

knit_root_dir The directory where code inside an R Markdown file is executed; this ultimately sets argument knit_root_dir in render. By default, wflow_start sets knit_root_dir in the file _workflowr.yml to be the path ".". This path is a relative path from the location of _workflowr.yml to the directory for the code to be executed. The path "." is shorthand for "current working directory", and thus code is executed in the root of the workflowr project. You can change this to be a relative path to any subdirectory of your project. Also, if you were to delete this line from _workflowr.yml, then this would cause the code to be executed from the same directory in which the R Markdown files are located (i.e. analysis/ in the default workflowr setup).

It is also possible (though in general not recommended) to configure the knit_root_dir to apply to only one of the R Markdown files by specifying it in the YAML header of that particular file. In this case, the supplied path is interpreted as relative to the R Markdown file itself. Thus knit_root_dir: "../data" would execute the code in the subdirectory data/.

seed The seed argument in the call to set.seed, which is added to the beginning of an R Markdown file. In wflow_start, this is set to the date using the format YYYYMMDD. If no seed is specified, the default is 12345.

sessioninfo The function that is run to record the session information. The default is "sessionInfo()".

github The URL of the remote repository for creating links to past results. If unspecified, the URL is guessed from the "git remote" settings (see wflow_git_remote). Specifying this setting inside _workflowr.yml is especially helpful if multiple users are collaborating on a project since it ensures that everyone generates the same URLs.

suppress_report By default a workflowr report is inserted at the top of every HTML file containing useful summaries of the reproducibility features and links to past versions of the analysis. To suppress this report, set suppress_report to TRUE.

In the default workflowr setup, the file _workflowr.yml is located in the root of the project. For most users it is best to leave it there, but if you are interested in experimenting with the directory layout, the _workflowr.yml file can be located in the same directory as the R Markdown files or in any directory upstream of that directory.

Here is an example of a customized _workflowr.yml file:

```
# Execute code in project directory
knit_root_dir: "."
# Set a custom seed
seed: 4815162342
# Use devtools to generate the session information.
sessioninfo: "devtools::session_info()"
# Use this URL when inserting links to past results.
github: https://github.com/repoowner/mainrepo
```

And here is an example of a YAML header inside an R Markdown file with the same exact custom settings as above:

```
---
title: "About"
output:
  workflowr::wflow_html:
    toc: false
workflowr:
  knit_root_dir: ".."
  seed: 4815162342
  sessioninfo: "devtools::session_info()"
  github: https://github.com/repoowner/mainrepo
---
```

Note that the path passed to `knit_root_dir` changed to `".."` because it is relative to the R Markdown file instead of `_workflowr.yml`. Both have the effect of having the code executed in the root of the workflowr project.

See Also

wflow_pre_knit, wflow_post_knit, wflow_pre_processor

`wflow_open` *Open R Markdown analysis file(s)*

Description

`wflow_open` opens R Markdown files in RStudio and sets the working directory to the knit directory (see Details). If a file does not exist, a minimal one is created.

Usage

```
wflow_open(files, change_wd = TRUE, edit_in_rstudio = TRUE, project = ".")
```

Arguments

`files`	character. R Markdown file(s) to open. Files must have the extension Rmd or rmd. Supports file globbing. Set `project = NULL` to create an R Markdown file outside of the R Markdown directory of a workflowr project.
`change_wd`	logical (default: TRUE). Change the working directory to the knit directory. If `project = NULL`, the working directory is **not** changed.
`edit_in_rstudio`	logical (default: TRUE). Open the file(s) in the RStudio editor.
`project`	character (or NULL). By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory. Set `project = NULL` if running this command to create a file for a non-workflowr project.

Details

wflow_open is a convenience function to make it easier to begin working, especially when starting a new analysis. First, it creates a new file if necessary and tries to make educated guesses about metadata like the title, author, and date. Second, it sets the working directory to the knit directory. The knit directory is where the code in the R Markdown files is executed, and may be defined via the field knit_root_dir in the file _workflowr.yml (see wflow_html for all the details). If this field is not defined, then the knit directory is the R Markdown directory. Third, it opens the file(s) in RStudio if applicable. The latter two side effects can be turned off if desired.

If you would like to create an R Markdown file with wflow_open for an analysis that is not part of a workflowr project, set project = NULL. Otherwise wflow_open will throw an error. Note that the working directory is **not** changed when project = NULL.

Value

An object of class wflow_open, which is a list with the following elements:

files	The input argument files as absolute paths.
change_wd	The input argument change_wd.
edit_in_rstudio	The input argument edit_in_rstudio.
knit_root_dir	The knit directory (see wflow_html for details). This is NULL if project was set to NULL.
previous_wd	The working directory in which wflow_open was executed.
new_wd	The working directory that wflow_open changed to. The value is NULL if the working directory was not changed.
files_new	The subset of the input argument files that were newly created. Paths are absolute.

Examples

```
## Not run:
wflow_open("analysis/model-data.Rmd")
# Multiple files
wflow_open(c("analysis/model-data.Rmd", "analysis/another-analysis.Rmd"))
# Open all R Markdown files
wflow_open("analysis/*Rmd")
# Create an R Markdown file in a non-worklowr project
wflow_open("model-data.Rmd", project = NULL)

## End(Not run)
```

wflow_post_knit *post_knit function for workflowr*

Description

This is the `post_knit` function that `wflow_html` passes to the function `output_format` from the package rmarkdown. For advanced usage only.

Usage

```
wflow_post_knit(metadata, input_file, runtime, encoding, ...)
```

Arguments

`metadata`	The metadata specified in the YAML header of the R Markdown file
`input_file`	Name of R Markdown file
`runtime`	The runtime target for rendering. The `static` option produces output intended for static files; `shiny` produces output suitable for use in a Shiny document (see `run`). The default, `auto`, allows the `runtime` target specified in the YAML metadata to take precedence, and renders for a `static` runtime target otherwise.
`encoding`	Ignored. The encoding is always assumed to be UTF-8.
`...`	arguments passed to the `post_knit` function of `rmarkdown::html_document`

Details

If you'd like to combine workflowr with another R Markdown output format, you may need to use `wflow_post_knit`. This function fixes the path to the R Markdown file (which is manipulated by `wflow_pre_knit`).

See Also

`wflow_html`, `wflow_pre_knit`, `wflow_pre_processor`

wflow_pre_knit *pre_knit function for workflowr*

Description

This is the `pre_knit` function that `wflow_html` passes to the function `output_format` from the package rmarkdown. For advanced usage only.

Usage

```
wflow_pre_knit(input, ...)
```

Arguments

input	Name of R Markdown file
...	currently ignored

Details

If you'd like to insert the workflowr reproducibility report into other R Markdown output formats such as blogdown::html_page, you can use wflow_pre_knit.

See Also

wflow_html, wflow_post_knit, wflow_pre_processor

---

wflow_pre_processor *pre_processor function for workflowr*

Description

This is the pre_processor function that wflow_html passes to the function output_format from the package rmarkdown. For advanced usage only.

Usage

```
wflow_pre_processor(
  metadata,
  input_file,
  runtime,
  knit_meta,
  files_dir,
  output_dir
)
```

Arguments

metadata	The metadata specified in the YAML header of the R Markdown file
input_file	Name of Markdown file created by knitr::knit to be passed to pandoc
runtime	The runtime target for rendering. The static option produces output intended for static files; shiny produces output suitable for use in a Shiny document (see run). The default, auto, allows the runtime target specified in the YAML metadata to take precedence, and renders for a static runtime target otherwise.
knit_meta	(This option is reserved for expert use.) Metadata generated by **knitr**.
files_dir	Directory for saving intermediate files

output_dir — The output directory for the rendered output_file. This allows for a choice of an alternate directory to which the output file should be written (the default output directory of that of the input file). If a path is provided with a filename in output_file the directory specified here will take precedence. Please note that any directory path provided will create any necessary directories if they do not exist.

Details

If you'd like to combine workflowr with another R Markdown output format, you may need to use wflow_pre_processor. This function only has minor effects on the style of the resulting HTML page, and is not essential for using workflowr.

See Also

wflow_html, wflow_pre_knit, wflow_post_knit

---

wflow_publish *Publish the site*

Description

wflow_publish is the main workflowr function. Use it when you are ready to publish an analysis to your site. wflow_publish performs three steps: 1) commit the file(s) (can include both Rmd and non-Rmd files, e.g. _site.yml), 2) rebuild the R Markdown file(s), 3) commit the generated website file(s). These steps ensure that the version of the HTML file is created by the latest version of the R Markdown file, which is critical for reproducibility.

Usage

```
wflow_publish(
  files = NULL,
  message = NULL,
  all = FALSE,
  force = FALSE,
  update = FALSE,
  republish = FALSE,
  combine = "or",
  view = getOption("workflowr.view"),
  delete_cache = FALSE,
  seed = 12345,
  verbose = FALSE,
  dry_run = FALSE,
  project = "."
)
```

Arguments

files	character (default: NULL). R Markdown files and other files to be added and committed with Git (step 1). Any R Markdown files will also be built (step 2) and their output HTML and figures will be subsequently committed (step 3). Supports file globbing. The files are always built in the order they are listed.
message	character (default: NULL). A commit message.
all	logical (default: FALSE). Automatically stage files that have been modified and deleted. Equivalent to: `git commit -a`
force	logical (default: FALSE). Allow adding otherwise ignored files. Equivalent to: `git add -f`
update	logical (default: FALSE). Build any files that have been committed more recently than their corresponding HTML files (and do not have any unstaged or staged changes). This ensures that the commit version ID inserted into the HTML corresponds to the exact version of the source file that was used to produce it.
republish	logical (default: FALSE). Build all published R Markdown files (that do not have any unstaged or staged changes). Useful for site-wide changes like updating the theme, navigation bar, or any other setting in `_site.yml`.
combine	character (default: `"or"`). Determine how to combine the files from the arguments `files`, `make` (`wflow_build()` only), `update`, and `republish`. When `combine` is `"or"`, any file specified by at least one of these arguments will be built. When `combine` is `"and"`, only files specified by all of these arguments will be built.
view	logical (default: `getOption("workflowr.view")`). View the website with `wflow_view` after building files. If only one file is built, it is opened. If more than one file is built, the main index page is opened. Not applicable if no files are built or if `dry_run = TRUE`.
delete_cache	logical (default: FALSE). Delete the cache directory (if it exists) for each R Markdown file prior to building it.
seed	numeric (default: 12345). The seed to set before building each file. Passed to `set.seed`. **DEPRECATED:** The seed set here has no effect if you are using `wflow_html` as the output format defined in `_site.yml`. This argument is for backwards compatibility with previous versions of workflowr.
verbose	logical (default: FALSE). Display the build log directly in the R console as each file is built. This is useful for monitoring long-running code chunks.
dry_run	logical (default: FALSE). Preview the proposed action but do not actually add or commit any files.
project	character (default: ".") By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory.

Value

Returns an object of class `wflow_publish`, which is a list with the following elements:

- **step1**: An object of class `wflow_git_commit` from the first step of committing the files.
- **step2**: An object of class `wflow_build` from the second step of building the HTML files.
- **step3**: An object of class `wflow_git_commit` from the third step of committing the HTML files.

See Also

`wflow_git_commit`, `wflow_build`

Examples

```
## Not run:
# single file
wflow_publish("analysis/file.Rmd", "Informative commit message")
# All tracked files that have been edited
wflow_publish(all = TRUE, message = "Informative commit message")
# A new file plus all tracked files that have been edited
wflow_publish("analysis/file.Rmd", "Informative commit message", all = TRUE)
# Multiple files
wflow_publish(c("analysis/file.Rmd", "analysis/another.Rmd"),
              "Informative commit message")
# All R Markdown files that start with the pattern "new_"
wflow_publish("analysis/new_*Rmd", "Informative commit message")
# Republish all published files even though they haven't been modified.
# Useful for changing some universal aspect of the site, e.g. the theme
# specified in _site.yml.
wflow_publish("analysis/_site.yml", "Informative commit message",
              republish = TRUE)
# Publish all previously published files that have been committed more
# recently than their corresponding HTML files. This is useful if you like to
# manually commit your R Markdown files.
wflow_publish(update = TRUE)

## End(Not run)
```

`wflow_quickstart` *Quickly start a workflowr project*

Description

`wflow_quickstart` provides a simple interface to effortlessly create a workflowr project from an existing data analysis.

Usage

```
wflow_quickstart(
  files,
  username = NULL,
```

```
  organization = NULL,
  supporting_files = NULL,
  directory = NULL,
  change_wd = TRUE,
  delete_on_error = TRUE,
  view = getOption("workflowr.view"),
  git.user.name = NULL,
  git.user.email = NULL,
  host = c("github", "gitlab"),
  create_on_github = NULL
)
```

Arguments

files	character. The R Markdown file(s) to be copied into the subdirectory `analysis/` of the newly created workflowr project. If the argument `directory` is left as `NULL`, the workflowr project will be named after the first Rmd file. This new directory will be located in the current working directory. Supports file globbing.
username	character (default: NULL). The GitHub or GitLab personal account you want to use to create the remote Git repository. It can also be the name of a GitLab Group that you belong to. However, if it is a GitHub organization, instead use the argument `organization`.
organization	The GitHub organization account you want to use to create the remote Git repository.
supporting_files	character (default: NULL) Supporting files or directories that are used by the Rmd files. These will be copied to the root of the project. Since by default Rmd files are executed in the root of the project, any relative file paths should still work. Long term it is recommended to move these supporting files to subdirectories of the workflowr project, e.g. `data/`.
directory	character (default: NULL). The path to the directory to create the workflowr project. This directory will also be used to name the remote Git repository. If left as `NULL`, the name is derived from the first Rmd file that is passed to the argument `files`.
change_wd	logical (default: TRUE). Change the working directory to the newly created workflowr project. Passed to `wflow_start`.
delete_on_error	logical (default: TRUE). Delete the newly created project if any error occurs.
view	logical (default: `getOption("workflowr.view")`). View the local website after it is built (will open the home page in the RStudio Viewer pane or your web browser).
git.user.name	character (default: `NULL`). The user name used by Git to sign commits, e.g., "Ada Lovelace". This setting only applies to the workflowr project being created. To specify the global setting for the Git user name, use `wflow_git_config` instead. When `user.name = NULL`, no user name is recorded for the project, and the global setting will be used. This setting can be modified later by running `git config --local` in the Terminal.

`git.user.email` character (default: `NULL`). The email address used by Git to sign commits, e.g., "ada.lovelace@ox.ac.uk". This setting only applies to the workflowr project being created. To specify the global setting for the Git email address, use `wflow_git_config` instead. When `user.name = NULL`, no email address is recorded for the project, and the global setting will be used. This setting can be modified later by running `git config --local` in the Terminal.

`host` character. Choose the service for hosting the Git repository. Must be either "github" for GitHub.com or "gitlab" for GitLab.com.

`create_on_github` logical (default: NULL). Should workflowr create the repository on GitHub? This requires logging into your GitHub account to authenticate workflowr to act on your behalf. The default behavior is to ask the user. Note that this only works for public repositories on github.com. If you want to create a private repository or are using GitHub Enterprise, you will need to manually create the repository.

Details

`wflow_quickstart` performs the following steps:

- Starts a new project with `wflow_start`
- Copies the Rmd file(s) to the subdirectory `analysis/`
- Copies the supporting file(s) and/or directory(s) to the root of the project (Note: by default Rmd files are executed in the root of the project, so relative file paths should still work)
- Adds link(s) to the results to the main index page
- Publishes the Rmd files with `wflow_publish`
- Configures the remote repository with `wflow_use_github` or `wflow_use_gitlab`

Once it has completed, you can push to the remote service with `wflow_git_push`. Alternatively you can run `git push` in the terminal.

If you are using GitHub and you chose to not allow workflowr to create the repository for you, then you will have to login to your account and create the new repository yourself. If you're using GitLab, you don't have to worry about this because the new repository will be automatically created when you push.

Value

Invisibly returns the absolute path to the newly created workflowr project.

See Also

workflowr, `wflow_start`, `wflow_publish`, `wflow_use_github`, `wflow_use_gitlab`, `wflow_git_push`

Examples

```
## Not run:

wflow_quickstart(files = "existing-analysis.Rmd", username = "your-github-username")
```

```
## End(Not run)
```

`wflow_remove` *Remove files*

Description

`wflow_remove` removes files. If the file to be removed is an R Markdown file, the corresponding HTML and other related files are also removed. If the workflowr project uses Git, `wflow_remove` commits the changes.

Usage

```
wflow_remove(files, message = NULL, git = TRUE, dry_run = FALSE, project = ".")
```

Arguments

`files`	character. Files to be removed. Supports file globbing.
`message`	character (default: NULL). A commit message.
`git`	logical (default: TRUE). Commit the changes (only applicable if Git repository is present).
`dry_run`	logical (default: FALSE). Preview the files to be removed but do not actually remove them.
`project`	character (default: ".") By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory.

Value

An object of class `wflow_remove`, which is a list with the following elements:

- **files**: The relative path(s) to the removed file(s).
- **message**: The message describing the commit (if applicable).
- **dry_run**: The input argument `dry_run`.
- **commit**:The object returned by git2r::`commit` (only included if `dry_run` == `FALSE`).
- **files_git**: The relative path(s) to the file(s) removed from the Git repository.

See Also

`wflow_git_commit`

Examples

```
## Not run:

# Remove a single file
wflow_remove("analysis/file.Rmd", "Remove old analysis.")
# Remove multiple files
wflow_remove(c("analysis/file.Rmd", "output/small-data.txt"),
             "Remove old analysis and its associated data.")

## End(Not run)
```

`wflow_rename` *Rename files and directories*

Description

`wflow_rename` renames files and directories. If the file to be renamed is an R Markdown file, the corresponding HTML and other related files are also renamed. If the workflowr project uses Git, `wflow_rename` commits the changes.

Usage

```
wflow_rename(
  files,
  to,
  message = NULL,
  git = TRUE,
  dry_run = FALSE,
  project = "."
)
```

Arguments

`files`	character. Files to be renamed. Supports file globbing.
`to`	character. New names for the files. Must be the same length as `files`.
`message`	character (default: NULL). A commit message.
`git`	logical (default: TRUE). Commit the changes (only applicable if Git repository is present).
`dry_run`	logical (default: FALSE). Preview the files to be renamed but do not actually rename them.
`project`	character (default: ".") By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory.

Value

An object of class `wflow_rename`, which is a list with the following elements:

- **files**: The relative path(s) to the renamed file(s).
- **to**: The new relative path(s) to rename the file(s).
- **message**: The message describing the commit (if applicable).
- **git**: The input argument `git`.
- **dry_run**: The input argument `dry_run`.
- **commit**:The object returned by git2r::`commit` (only included if `dry_run == FALSE`).
- **files_git**: The relative path(s) to the file(s) renamed from the Git repository.

See Also

`wflow_git_commit`

Examples

```
## Not run:

# rename a single file
wflow_rename("analysis/file.Rmd", "analysis/new.Rmd", "rename old analysis.")
# rename multiple files
wflow_rename(c("analysis/file.Rmd", "output/small-data.txt"),
             c("analysis/new.Rmd", "output/new-data.txt"),
             "rename old analysis and its associated data.")

## End(Not run)
```

`wflow_rename_proj` *Rename a workflowr project*

Description

If you want to rename an existing workflowr project, use `wflow_rename_proj` to update the name throughout all the project files.

Usage

```
wflow_rename_proj(
  name,
  rproj = TRUE,
  remote = TRUE,
  navbar = TRUE,
  readme = TRUE,
  commit = TRUE,
  directory = TRUE,
  project = "."
)
```

Arguments

name	character. The new name for the workflowr project.
rproj	logical (default: TRUE). Rename the RStudio Project file.
remote	logical (default: TRUE). Rename the remote URL.
navbar	logical (default: TRUE). Rename the navbar title.
readme	logical (default: TRUE). Rename the README title.
commit	logical (default: TRUE). Commit the changes to Git.
directory	logical (default: TRUE). Rename the project directory.
project	character (default: ".") By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory.

Details

`wflow_rename_proj` performs the following steps and then commits the changes:

- Rename RStudio Project file (`.Rproj`)
- Update URL of remote repository (see `wflow_git_remote`)
- Update project name in the navigation bar (defined in `_site.yml`)
- Update title of README file
- Rename the project directory itself

After renaming the project with `wflow_rename_proj`, you should republish the R Markdown files with `wflow_publish(republish = TRUE)`. Also, you should go to the settings of your Git repository on the online Git hosting platform to change its name.

Value

Invisibly returns the path to the project directory

See Also

`wflow_publish`

Examples

```
## Not run:

wflow_rename_proj("new-project-name")

## End(Not run)
```

wflow_run *Run the code*

Description

wflow_run executes the code chunks of an R Markdown file in the current R session without affecting any of the website files. This is meant to be used while interactively developing an analysis. It does **not** change the working directory, isolate the computation from the current R session, nor set the seed of the random number generator. This is analogous to the RStudio option "Run all" to run all the code chunks. Use wflow_publish when you are ready to add the results to the website.

Usage

```
wflow_run(file = NULL, verbose = TRUE, project = ".")
```

Arguments

file	character (default: NULL). The R Markdown file to execute. Must have file extension Rmd or rmd. If NULL, the most recently modified Rmd file will be executed.
verbose	logical (default: TRUE). Should the lines of code (and their output) be echoed in the R console as they are executed? This argument is passed directly to the argument echo of the function source.
project	character (default: ".") By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory.

Value

Invisibly returns the path to the Rmd file that was executed

See Also

wflow_build with argument local = TRUE, source with argument echo = TRUE

Examples

```
## Not run:

# Run the most recently modified Rmd file
wflow_run()
# Run a specific Rmd file
wflow_run("analysis/file.Rmd")

## End(Not run)
```

wflow_site *Custom site generator for workflowr websites*

Description

wflow_site is a custom site generator to be used in combination with the R Markdown output format wflow_html.

Usage

```
wflow_site(input, encoding = getOption("encoding"), ...)
```

Arguments

input	character. The name of the website directory or a specific R Markdown file in the website directory.
encoding	character. The character encoding to use to read the file.
...	Placeholder for potential future use.

Details

Do not call the function wflow_site directly. Instead insert the line below directly into the YAML header of the file index.Rmd:

```
---
title: "Home"
site: workflowr::wflow_site
output:
  workflowr::wflow_html:
    toc: false
---
```

Then you can build the website by running render_site in the R console or clicking the Knit button in RStudio.

If you receive an error when using the RStudio Knit button (the error is about an unused argument), make sure the Knit Directory is set to Document Directory (you can set this with the dropdown menu next to the Knit button).

See Also

wflow_html, render_site

wflow_start *Start a new workflowr project*

Description

wflow_start creates a directory with the essential files for a workflowr project. The default behavior is to add these files to a new directory, but it is also possible to populate an existing directory. By default, the working directory is changed to the workflowr project directory.

Usage

```
wflow_start(
  directory,
  name = NULL,
  git = TRUE,
  existing = FALSE,
  overwrite = FALSE,
  change_wd = TRUE,
  disable_remote = FALSE,
  dry_run = FALSE,
  user.name = NULL,
  user.email = NULL
)
```

Arguments

directory	character. The directory where the workflowr project files will be added, e.g., "~/my-wflow-project". When existing = FALSE, the directory will be created.
name	character (default: NULL). The name of the project, e.g. "My Workflowr Project". When name = NULL, the project name is automatically determined based on directory. For example, if directory = "~/projects/my-wflow-project", then name is set to "my-wflow-project". The project name is displayed on the website's navigation bar and in the README.md file.
git	logical (default: TRUE). Should the workflowr files be committed with Git? If git = TRUE and no existing Git repository is detected, wflow_start will initialize the repository and make an initial commit. If a Git repository already exists in the chosen directory, wflow_start will commit any newly created or modified files to the existing repository (also need to set existing = TRUE). If git = FALSE, wflow_start will not perform any Git commands.
existing	logical (default: FALSE). Indicate whether directory already exists. This argument is added to prevent accidental creation of files in an existing directory; setting existing = FALSE prevents files from being created if the specified directory already exists.
overwrite	logical (default: FALSE). Similar to existing, this argument prevents files from accidentally being overwritten when overwrite = FALSE. When overwrite = TRUE, any existing file in directory that has the same name as a workflowr

file will be replaced by the workflowr file. When `git = TRUE`, all the standard workflowr files will be added and committed (regardless of whether they were overwritten or still contain the original content).

`change_wd` logical (default: `TRUE`). Change the working directory to the `directory`.

`disable_remote` logical (default: `FALSE`). Create a Git pre-push hook that prevents pushing to a remote Git repository (i.e. using `wflow_git_push`). This is useful for extremely confidential projects that cannot be shared via an online Git hosting service (e.g. GitHub or GitLab). The hook is saved in the file `.git/hooks/pre-push`. If you change your mind and want to push the repository, you can delete that file. Note that this option is only available if `git = TRUE`. Note that this is currently only supported for Linux and macOS.

`dry_run` logical (default: `FALSE`). When `dry_run = TRUE`, the actions are previewed without executing them.

`user.name` character (default: `NULL`). The user name used by Git to sign commits, e.g., "Ada Lovelace". This setting only applies to the workflowr project being created. To specify the global setting for the Git user name, use `wflow_git_config` instead. When `user.name = NULL`, no user name is recorded for the project, and the global setting will be used. This setting can be modified later by running `git config --local` in the Terminal.

`user.email` character (default: `NULL`). The email address used by Git to sign commits, e.g., "ada.lovelace@ox.ac.uk". This setting only applies to the workflowr project being created. To specify the global setting for the Git email address, use `wflow_git_config` instead. When `user.name = NULL`, no email address is recorded for the project, and the global setting will be used. This setting can be modified later by running `git config --local` in the Terminal.

Details

This is recommended function to set up the file infrastructure for a workflowr project. If you are using RStudio, you can also create a new workflowr project as an "RStudio Project Template". Go to "File" -> "New Project..." then select "workflowr project" from the list of project types. In the future, you can return to your project by choosing menu option "Open Project..." and selecting the `.Rproj` file located at the root of the workflowr project directory. In RStudio, opening this file will change the working directory to the appropriate location, set the file navigator to the workflowr project directory, and configure the Git pane.

`wflow_start` populates the chosen directory with the following files:

```
|--- .gitignore
|--- .Rprofile
|--- _workflowr.yml
|--- analysis/
|   |--- about.Rmd
|   |--- index.Rmd
|   |--- license.Rmd
|   |--- _site.yml
|--- code/
|   |--- README.md
```

```
|--- data/
|   |--- README.md
|--- docs/
|--- <directory>.Rproj
|--- output/
|   |--- README.md
|--- README.md
```

The two **required** subdirectories are `analysis/` and `docs/`. These directories should never be removed from the workflowr project.

`analysis/` contains all the source R Markdown files that implement the analyses for your project. It contains a special R Markdown file, `index.Rmd`, that typically does not include R code, and is will be used to generate `index.html`, the homepage for the project website. Additionally, this directory contains the important configuration file `_site.yml`. The website theme, navigation bar, and other properties can be controlled through this file (for more details see the documentation on R Markdown websites). Do not delete `index.Rmd` or `_site.yml`.

`docs/` will contain all the webpages generated from the R Markdown files in `analysis/`. Any figures generated by rendering the R Markdown files are also stored here. Each figure is saved according to the following convention: `docs/figure/<Rmd-filename>/<chunk-name>-#.png`, where # corresponds to which of the plots the chunk generated (one chunk can produce several plots).

`_workflowr.yml` is an additional configuration file used only by workflowr. It is used to apply the workflowr reproducibility checks consistently across all R Markdown files. The most important setting is `knit_root_dir` which determines the directory where the scripts in `analysis/` are executed. The default is to run code from the project root (*i.e.,* `"."`). To execute the code from `analysis/`, for example, change the setting to `knit_root_dir: "analysis"`. See `wflow_html` for more details.

Another required file is the RStudio project file (ending in `.Rproj`). *Do not delete this file even if you do not use RStudio; among other essential tasks, it is used to determine the project root directory.*

The **optional** directories are `data/`, `code/`, and `output/`. These directories are suggestions for organizing your workflowr project and can be removed if you do not find them relevant to your project.

`data/` should be used to store "raw" (unprocessed) data files.

`code/` should be used to store additional code that might not be appropriate to include in R Markdown files (e.g., code to preprocess the data, long-running scripts, or functions that are used in multiple R Markdown files).

`output/` should be used to store processed data files and other outputs generated from the code and analyses. For example, scripts in `code/` that pre-process raw data files from `data/` should save the processed data files in `output/`.

All these subdirectories except for `docs/` include a README file summarizing the contents of the subdirectory, and can be modified as desired, for example, to document the files stored in each directory.

`.Rprofile` is an optional file in the root directory of the workflowr project containing R code that is executed whenever the `.Rproj` file is loaded in RStudio, or whenever R is started up inside the project root directory. This file includes the line of code `library("workflowr")` to ensure that the workflowr package is loaded.

Finally, `.gitignore` is an optional file that indicates to Git which files should be ignored—that is, files that are never committed to the repository. Some suggested files to ignore such as `.Rhistory` and `.Rdata` are listed here.

Value

An object of class `wflow_start`, which is a list with the following elements:

`directory`	The input argument `directory`.
`name`	The input argument `name`.
`git`	The input argument `git`.
`existing`	The input argument `existing`.
`overwrite`	The input argument `overwrite`.
`change_wd`	The input argument `change_wd`.
`disable_remote`	The input argument `disable_remote`.
`dry_run`	The input argument `dry_run`.
`user.name`	The input argument `user.name`.
`user.email`	The input argument `user.email`.
`commit`	The object returned by git2r::`commit`, or `NULL` if `git` = `FALSE`.

Note

Do not delete the file `.Rproj` even if you do not use RStudio; workflowr will not work correctly unless this file is there.

See Also

vignette("wflow-01-getting-started")

Examples

```
## Not run:

wflow_start("path/to/new-project")

# Provide a custom name for the project.
wflow_start("path/to/new-project", name = "My Project")

# Preview what wflow_start would do
wflow_start("path/to/new-project", dry_run = TRUE)

# Add workflowr files to an existing project.
wflow_start("path/to/current-project", existing = TRUE)

# Add workflowr files to an existing project, but do not automatically
# commit them.
wflow_start("path/to/current-project", git = FALSE, existing = TRUE)
```

```
## End(Not run)
```

`wflow_status` *Report status of workflowr project*

Description

`wflow_status` reports the analysis files that require user action.

Usage

```
wflow_status(files = NULL, include_git_status = TRUE, project = ".")
```

Arguments

`files`	character (default: NULL) The analysis file(s) to report the status. By default checks the status of all analysis files. Supports file globbing.
`include_git_status`	logical (default: TRUE) Include the Git status of the project files in the output. Note that this excludes any files in the website directory, since these generated files should only be committed by workflowr, and not the user.
`project`	character (default: ".") By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory.

Details

`wflow_status` reports analysis files with one of the following statuses:

- **Mod**: Modified file. Any published file that has been modified since the last time the HTML was published.
- **Unp**: Unpublished file. Any tracked file whose corresponding HTML is not tracked. May or may not have staged or unstaged changes.
- **Scr**: Scratch file. Any untracked file that is not specifically ignored.

`wflow_status` only works for workflowr projects that use Git.

Value

Returns an object of class `wflow_status`, which is a list with the following elements:

- **root**: The relative path to the root directory of the workflowr project (i.e. contains the RStudio .Rproj file).
- **analysis**: The relative path to the directory that contains `_site.yml` and the R Markdown files.
- **docs**: The relative path to the directory that contains the HTML files and figures.

- **git**: The relative path to the `.git` directory that contains the history of the Git repository.
- **site_yml**: `TRUE` if the configuration file `_site.yml` has uncommitted changes, otherwise `FALSE`.
- **wflow_yml**: `TRUE` if the configuration file `_workflowr.yml` has uncommitted changes, otherwise `FALSE`. If the file does not exist, the result is `NULL`. If the file was recently deleted and not yet committed to Git, then it will be `TRUE`.
- **git_status** The Git status as a `git_status` object from the package git2r (see `git2r::status`).
- **include_git_status** The argument `include_git_status` indicating whether the Git status should be printed along with the status of the Rmd files.
- **status**: A data frame with detailed information on the status of each R Markdown file (see below).

The data frame `status` contains the following non-mutually exclusive columns (all logical vectors):

- **ignored**: The R Markdown file has been ignored by Git according to the patterns in the file `.gitignore`.
- **mod_unstaged**: The R Markdown file has unstaged modifications.
- **conflicted**: The R Markdown file has merge conflicts.
- **mod_staged**: The R Markdown file has staged modifications.
- **tracked**: The R Markdown file is tracked by Git.
- **committed**: The R Markdown file has been previously committed to the Git repository.
- **published**: The corresponding HTML file has been previously committed.
- **mod_committed**: The R Markdown file has modifications that have been committed since the last time the HTML was built and committed.
- **modified**: The R Markdown file has been modified since it was last published (i.e. `mod_unstaged` or `mod_staged` or `mod_committed`).
- **unpublished**: The R Markdown file is tracked by Git but not published (i.e. the HTML has not been committed).
- **scratch**: The R Markdown file is untracked by Git, i.e. it is considered a scratch file until it is committed.

Examples

```
## Not run:

wflow_status()
# Get status of specific file(s)
wflow_status("analysis/file.Rmd")
# Save the results
s <- wflow_status()

## End(Not run)
```

`wflow_toc` *Create table of contents*

Description

`wflow_toc` creates a table of contents of the published R Markdown files. The output is in markdown format, so you can paste it into a document such as `index.Rmd`. If the R package clipr is installed, the table of contents is copied to the clipboard. Otherwise the output is sent to the R console.

Usage

```
wflow_toc(
  ignore_nav_bar = TRUE,
  clipboard = TRUE,
  only_published = TRUE,
  project = "."
)
```

Arguments

`ignore_nav_bar`	logical (default: TRUE). Ignore any HTML files included as links in the navigation bar.
`clipboard`	logical (default: TRUE) Attempt to copy table of contents to clipboard. Only relevant if clipr package is installed and the system keyboard is available.
`only_published`	logical (default: TRUE) Include only published contents.
`project`	character (default: ".") By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory.

Details

The default behavior is to attempt to copy the table of contents to the clipboard for easy pasting into an R Markdown document. If this isn't working for you, you can try the following:

- Check that the clipr package is installed: `install.packages("clipr")`
- Check that the system keyboard is writable. Run `clipr_available` and `dr_clipr`.
- If it's still not working, set `clipboard = FALSE` to send the table of contents to the R console to manually copy-paste.

Value

Invisibly returns the table of contents as a character vector.

`wflow_use_github` *Deploy site with GitHub*

Description

`wflow_use_github` automates all the local configuration necessary to deploy your workflowr project with GitHub Pages. Optionally, it can also create the new repository on GitHub (only applies to public repositories hosted on github.com). Afterwards, you will need to run `wflow_git_push` in the R console (or `git push` in the terminal) to push the code to GitHub.

Usage

```
wflow_use_github(
  username = NULL,
  repository = NULL,
  organization = NULL,
  navbar_link = TRUE,
  create_on_github = NULL,
  protocol = "https",
  domain = "github.com",
  project = "."
)
```

Arguments

`username`	character (default: NULL). The GitHub account associated with the GitHub repository. This should be your personal GitHub username. If the repository will be created for a GitHub organization, instead use the argument `organization`. It will be combined with the arguments `repository` and `domain` to determine the URL of the new repository, e.g. the default is https://github.com/username/repository. It will be combined with the arguments `repository`, `domain`, and `protocol` to determine the URL for Git to use to push and pull from GitHub, e.g. the default is https://github.com/username/repository.git. If `username` is not specified, `wflow_use_github` will first attempt to guess it from the current setting for the remote URL named "origin". If you haven't previously configured a remote for this workflowr project (or you are unsure what that means), then you should specify your GitHub username when calling this function.
`repository`	character (default: NULL). The name of the remote repository on GitHub. If not specified, workflowr will guess the name of the repository. First, it will check the current setting for the remote URL named "origin". Second, it will use the name of the root directory of the workflowr project.
`organization`	character (default: NULL). The GitHub organization associated with the GitHub repository. Only set one of `organization` or `username`. See the argument `username` above for more details.
`navbar_link`	logical (default: TRUE). Insert a link to the GitHub repository into the navigation bar.

`create_on_github`
: logical (default: NULL). Should workflowr create the repository on GitHub? This requires logging into your GitHub account to authenticate workflowr to act on your behalf. The default behavior is to ask the user. Note that this only works for public repositories on github.com. If you want to create a private repository or are using GitHub Enterprise, you will need to manually create the repository.

`protocol`
: character (default: "https"). The protocol for communicating with GitHub. Must be either "https" or "ssh".

`domain`
: character (default: "github.com"). The domain of the remote host. You only need to change this if your organization is using GitHub Enterprise.

`project`
: character (default: ".") By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory.

Details

`wflow_use_github` performs the following steps and then commits the changes:

- Adds a link to the GitHub repository in the navigation bar
- Configures the Git remote settings to use GitHub (via `wflow_git_remote`)
- (Only if necessary) Renames the website directory to `docs/`
- (Only if necessary) Edits the setting `output_dir` in the file `_site.yml` to save the website files in `docs/`

Furthermore, you have two options for creating the remote repository on GitHub. In an interactive R session, you will be prompted to choose one of the options below. To bypass the prompt, you can set the argument `create_on_github`.

- 1. Have workflowr create the new repository on GitHub. If you accept, your browser will open for you to provide authorization. If you are not logged into GitHub, you will be prompted to login. Then you will be asked to give permission to the workflowr-oauth-app to create the new repository for you on your behalf. This will allow `wflow_use_github`, running on your own machine, to create your new repository. Once `wflow_use_github` finishes, workflowr can no longer access your GitHub account.
- 2. Create the remote repository yourself by going to `https://github.com/new` and entering the Repository name that matches the name of the directory of your workflowr project (if you used the argument `repository` to make it a different name, make sure to instead use that one).

Once the GitHub repository has been created either by `wflow_use_github` or yourself, run `wflow_git_push` in the R console (or `git push origin master` in the terminal) to push your code to GitHub.

Value

Invisibly returns a list of class `wflow_use_github`. This is currently for internal use only. Please open an Issue if you'd like to use this information.

Troubleshooting

The feature to automatically create the GitHub repository for you may fail since it involves using your web browser to authenticate with your GitHub account. If it fails for any reason, it'd probably be easier to manually login to GitHub and create the repository yourself (instructions from GitHub). However, if you have time, please file an Issue on GitHub to report what happened, and importantly include which web browser you were using.

We have observed the following problems before:

- The green button to approve the authentication of the workflowr GitHub app to create the repository on your behalf is grayed out, and unable to be clicked. This is likely a JavaScript problem. Make sure you don't have JavaScript disabled in your web browser. Also, you can try using a different browser.

See Also

wflow_git_push, wflow_git_remote, wflow_use_gitlab

Examples

```
## Not run:

wflow_use_github("your-username", "name-of-repository")
# Login with GitHub account and create new repository
wflow_git_push()

# Create a repository for an organization  you belong to
wflow_use_github(organization = "my-org")

## End(Not run)
```

---

wflow_use_gitlab *Deploy site with GitLab*

Description

wflow_use_gitlab automates all the local configuration necessary to deploy your workflowr project with GitLab Pages. Afterwards, you will need to run wflow_git_push in the R console (or git push in the terminal) to push the code to GitLab. Note that this will also create the repository if it doesn't exist yet (this requires GitLab 10.5 or greater). Alternatively, you could manually login to your account and create the new repository on GitLab prior to pushing.

Usage

```
wflow_use_gitlab(
  username = NULL,
  repository = NULL,
  navbar_link = TRUE,
```

```
  protocol = "https",
  domain = "gitlab.com",
  project = "."
)
```

Arguments

username	character (default: NULL). The GitLab account associated with the GitLab repository. This is likely your personal GitLab username, but it could also be the name of a GitLab Group you belong to. It will be combined with the arguments repository and domain to determine the URL of the new repository, e.g. the default is https://gitlab.com/username/repository. It will be combined with the arguments repository, domain, and protocol to determine the URL for Git to use to push and pull from GitLab, e.g. the default is https://gitlab.com/username/repository.git. If username is not specified, wflow_use_gitlab will first attempt to guess it from the current setting for the remote URL named "origin". If you haven't previously configured a remote for this workflowr project (or you are unsure what that means), then you should specify your GitLab username when calling this function.
repository	character (default: NULL). The name of the remote repository on GitLab. If not specified, workflowr will guess the name of the repository. First, it will check the current setting for the remote URL named "origin". Second, it will use the name of the root directory of the workflowr project.
navbar_link	logical (default: TRUE). Insert a link to the GitLab repository into the navigation bar.
protocol	character (default: "https"). The protocol for communicating with GitLab. Must be either "https" or "ssh".
domain	character (default: "gitlab.com"). The domain of the remote host. You only need to change this if you are using a custom GitLab instance hosted by your organization. For example, "git.rcc.uchicago.edu" is the domain for the GitLab instance hosted by the University of Chicago Research Computing Center.
project	character (default: ".") By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory.

Details

wflow_use_gitlab performs the following steps and then commits the changes:

- Renames the website directory from docs/ to public/
- Edits the setting output_dir in the file _site.yml to save the website files in public/
- Adds a link to the GitLab repository in the navigation bar
- Creates the required file .gitlab-ci.yml
- Configures the Git remote settings to use GitLab

By default the GitLab repository is set to private, so you are the only one that can access it. If you need to keep it private, you can grant access to collaborators in Settings->Members. Otherwise, you can make it public in Settings->General->Visibility.

For more details, read the documentation provided by GitLab Pages.

Value

Invisibly returns a list of class `wflow_use_gitlab`. This is currently for internal use only. Please open an Issue if you'd like to use this information.

See Also

`wflow_git_push`, `wflow_git_remote`, `wflow_use_github`, vignette("wflow-06-gitlab")

Examples

```
## Not run:

wflow_use_gitlab("your-username", "name-of-repository")
# Login with GitLab account and create new repository
wflow_git_push()

## End(Not run)
```

---

wflow_view *View research website locally*

Description

`wflow_view` displays the website locally in your browser or the RStudio Viewer pane.

Usage

```
wflow_view(files = NULL, latest = FALSE, dry_run = FALSE, project = ".")
```

Arguments

`files`	character (default: NULL). Name(s) of the specific file(s) to view. These can be either the name(s) of the R Markdown file(s) in the analysis directory or the HTML file(s) in the docs directory. Supports file globbing.
`latest`	logical (default: FALSE). Display the HTML file with the most recent modification time (in addition to those specified in `files`). If `files = NULL` and `latest = FALSE`, then `index.html` is viewed.
`dry_run`	logical (default: FALSE). Do not actually view file(s). Mainly useful for testing.
`project`	character (default: "."). By default the function assumes the current working directory is within the project. If this is not true, you'll need to provide the path to the project directory.

Details

wflow_view by default displays the file index.html. To view the most recently modified HTML file, set latest = TRUE. To specify which file(s) to view, specify either the name(s) of the R Markdown or HTML file(s).

wflow_view uses browseURL to view the HTML files in the browser. If you wish to do something non-traditional like view an HTML file that is not in the docs directory or not part of a workflowr project, you can use that function directly.

If wflow_view is run in the RStudio IDE and only one file has been requested to be viewed, the file is displayed in the RStudio Viewer.

If R has no default browser set (determined by getOption("browser")), then wflow_view cannot open any HTML files. See browseURL for setup instructions.

Value

An object of class wflow_view, which is a list with the following elements:

files	The input argument files (converted to relative paths).
latest	The input argument latest.
dry_run	The input argument dry_run.
browser	Logical indicating if a default browser has been set. If FALSE, no HTML files can be opened. This is determined by the value returned by getOption("browser").
opened	The HTML files opened by wflow_view.

See Also

browseURL

Examples

```
## Not run:

# View index.html
wflow_view()

# View the most recently modified HTML file
wflow_view(latest = TRUE)

# View a file by specifying the R Markdown file
wflow_view("analysis/fname.Rmd")

# View a file by specifying the HTML file
wflow_view("docs/fname.html")

# View multiple files
wflow_view(c("fname1.Rmd", "fname2.Rmd"))
wflow_view("docs/*html")

## End(Not run)
```

workflowr *workflowr: A workflow template for creating a research website*

Description

The workflowr package helps you create a research website using R Markdown and Git.

Vignettes

Run `browseVignettes("workflowr")` to read the package vignettes locally. Alternatively you can read the documentation online at https://workflowr.github.io/workflowr/.

Main workflowr functions

`wflow_start` Start a workflowr project.

`wflow_build` Build the site to view locally.

`wflow_publish` Publish analyses to include in the website.

`wflow_status` Report status of analysis files.

Supporting workflowr functions

For further information on workflowr, see the help pages for these functions:

`wflow_html` More technical details about how individual R Markdown files are converted to webpages, and how the rendering settings can be customized.

`wflow_site` This help page explains how project-wide rendering settings can be customized in the `_site.yml` file.

Package options

The following package options affect the default behavior of the workflowr functions. To permanently set any of these options, add a call to the function `options` in the file `.Rprofile` at the root of your workflowr project. For example:

```
# Do not use Git executable
options(workflowr.sysgit = "")
```

workflowr.autosave A logical indicating whether workflowr functions should automatically save files open in the RStudio editor before running. The default is `TRUE`. This requires RStudio 1.1.287 or later. Only files that have been previously saved are affected. In other words, unnamed files will be ignored.

workflowr.sysgit The path to the system Git executable. This is occasionally used to increase the speed of Git operations performed by workflowr. By default it is set to the first Git executable on the search path. You can specify a path to a different Git executable. Alternatively you can disable this behavior entirely by setting it to the empty string `""`.

workflowr.view A logical indicating whether workflowr functions should open webpages for viewing in the browser. The default is set to `interactive` (i.e. it is `TRUE` only if it is an interactive R session). This option is currently used by `wflow_build`, `wflow_git_push`, and `wflow_publish`.

Index