

Package: codeInspectr (via r-universe)

October 3, 2026

Title Summarize Package Repositories And Their Relationships

Version 0.0.0.9000

Description Helps understand github-based R packages and their relationships. Includes tools to summarize package status, issues, pull requests, as well as download complete backups of Github Issues and Pull Requests.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

URL <https://github.com/cbedwards-dfw/codeInspectr>,
<https://cbedwards-dfw.github.io/codeInspectr/>

BugReports <https://github.com/cbedwards-dfw/codeInspectr/issues>

Suggests callr, framrsquared, knitr, fs, rmarkdown, spelling, withr, testthat (>= 3.0.0), devtools, usethis

Language en-US

Imports cli, codetools, curl, dplyr, gh, glue, httr2, jsonlite, purrr, rlang, stringr, tibble, tidyr, visNetwork, yaml

VignetteBuilder knitr

Remotes FRAMverse/framrsquared

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake git make libicu-dev libuv1-dev libssl-dev

Repository <https://framverse.r-universe.dev>

Date/Publication 2026-10-02 23:45:06 UTC

RemoteUrl <https://github.com/cbedwards-dfw/codeInspectr>

RemoteRef HEAD

RemoteSha ec7d46dd6c9e6521c61ba68cb6c8d08e284326d1

Contents

backup_github_info	2
build_package_manual	3
build_package_network	4
build_recursive_edges	5
compile_package_manuals	5
compile_vignettes	6
find_function_dependencies	7
find_recursive_dependencies	8
find_reverse_dependencies	9
get_branch_activity	9
get_github_activity	10
get_issues	11
get_new_repos	12
get_pull_requests	12
get_raw_issues	13
get_raw_pulls	14
is_r_package	14
list_repos	15
package_functions	16
parse_name	17
plot_function_dependencies	17
process_repo	18
safe_dir_delete	19
summarize_github_activity	19
summarize_repository	21
Index	22

backup_github_info	<i>Copy all Issues and PRs for a user or organization</i>
--------------------	---

Description

To provide a backup on github failure, this function streamlines downloading all github Issues and Pull Requests of all repositories of a focal user or organization into local jsons.

Usage

```
backup_github_info(name, save_location, entity_type = c("user", "org"))
```

Arguments

name	Name of github user or organization (e.g., "cbedwards-dfw", "FRAMverse"). Character atomic.
save_location	Filepath for directory to save issue and pulls to.
entity_type	Either "user" or "org" to identify if name is describing a github user or github organization. Defaults to "user"

Value

Invisibly returns a tibble with:

- \$repo_name = name of repository
- \$full_name = full github name ({user or org}/{repo name})
- \$issues_path = filepath to the json of issues
- \$pulls_path = filepath to the json of pulls
- \$issues_status, \$pulls_status = Did the backup work? "success" or "error"
- \$issues_error_msg, \$pulls_error_msg = error messages on attempting to back up (NA if there was no error.)

Examples

```
## Not run:
## backing up all issues and PRs for repos hosted by `cbedwards-dfw`
out <- backup_github_info("cbedwards-dfw", "user", save_location = "C:/Repos/cbe_github_backups")

## End(Not run)
```

build_package_manual *Build manual (and optionally vignettes) for individual repository*

Description

For an R package stored on Github, compile a pdf reference manual and optionally build any vignettes. Requires LaTeX to run. Under the hood, downloads temporary copy of github repo, compiles document(s), then deletes repository.

Usage

```
build_package_manual(
  repo_address,
  output_path,
  build_vignettes = FALSE,
  verbose = TRUE
)
```

Arguments

repo_address	Github repository "full name" of form {user}/{repository}, as in "FRAMverse/framrsquared". Also accepts full url (e.g., "https://github.com/FRAMverse/framrsquared/"). Character atomic.
output_path	Filepath for a directory into which to save manuals. Character atomic.
build_vignettes	Attempt to build vignettes for any package that has them? Logical, defaults to FALSE.
verbose	Print progress statements? Logical, defaults to TRUE.

Value

Nothing

Examples

```
## Not run:
build_package_manual(repo = "FRAMverse/framrsquared",
  output_path = "C:/Repos/test_manuals",
  build_vignettes = TRUE)

## End(Not run)
```

build_package_network *Build edges of all function dependencies in package*

Description

Build edges of all function dependencies in package

Usage

```
build_package_network(package_name, include = "package only")
```

Arguments

package_name	package name as character string
include	Should only package functions be returned ("package only"), or all functions except for base R operators ("no operators"), or all functions ("all")?

Value

dataframe of dependency edges, with \$from and \$to

Examples

```
build_package_network("codeInspectr")
```

build_recursive_edges *Build dataframe of edges of recursive dependencies*

Description

Build dataframe of edges of recursive dependencies

Usage

```
build_recursive_edges(
  package_name,
  fun,
  pattern_ignore = NULL,
  pattern_dont_recurse = NULL
)
```

Arguments

package_name	Name of pattern. Character vector, length 1
fun	Name of function. Character vector, length 1
pattern_ignore	NULL or character vector of regex patterns. If present, recursive_deps will ignore any functions that match the pattern(s) character vector. Can take multiple patterns.
pattern_dont_recurse	NULL or character vector of regex patterns. If present, recursive_deps will list functions that match the pattern(s), but will not recurse into those functions.. Can take multiple patterns.

Value

dataframe, with \$from and \$to representing the "from" and "to" nodes of every edge

Examples

```
build_recursive_edges("purrr", "map")
```

compile_package_manuals

Save R package documentation as PDFs

Description

Creates PDF manuals for all R packages of a given user or organization. Manuals are created with devtools::build_manual. This requires a version of LaTeX installed. If you don't have a version installed, you can install it with tinytex::install_tinytex(). Optionally can also build vignettes for each package. Successfully building vignettes may require additional packages be installed (whatever packages the vignettes depend on).

Usage

```
compile_package_manuals(
  output_path,
  name,
  entity_type = c("user", "org"),
  build_vignettes = FALSE,
  repo_type = c("all", "public", "private", "forks"),
  verbose = TRUE
)
```

Arguments

output_path	Filepath for a directory into which to save manuals. Character atomic.
name	Name of user or organization (e.g., "FRAMverse" or "cbedwards-dfw"). Character atomic.
entity_type	Is this a Github User ("user") or organization ("org")? Character atomic, defaults to "user".
build_vignettes	Attempt to build vignettes for any package that has them? Logical, defaults to FALSE.
repo_type	List "all" repos, "public" repos, "private" repos, or only "fork" repos? Character atomic, defaults to "all". Private repos will only be provided if you have access to the private repos of that user or org.
verbose	Print progress statements? Logical, defaults to TRUE.

Value

Nothing

Examples

```
## Not run:
# build manuals and vignettes for all R packages in the FRAMverse organization.
compile_package_manuals(output_path = "C:/Repos/manuals",
  name = "FRAMverse",
  entity_type = "org",
  build_vignettes = TRUE)

## End(Not run)
```

compile_vignettes	<i>Create vignettes for R package</i>
-------------------	---------------------------------------

Description

If the git repository for an R package contains vignettes, builds them and saves them in output_path, with prefix matching the package name.

Usage

```
compile_vignettes(repo_address, repo_dir, output_path, verbose)
```

Arguments

repo_address	Github repository "full name" of form {user}/{repository}, as in "FRAMverse/framrsquared". Also accepts full url (e.g., "https://github.com/FRAMverse/framrsquared/"). Character atomic.
repo_dir	Temporary filepath for cloned github repository
output_path	Filepath for a directory into which to save manuals. Character atomic.
verbose	Print progress statements? Logical, defaults to TRUE.

Value

Character vector of vignette filepaths (if successful and vignettes exist) or empty character string

```
find_function_dependencies
```

List the functions in a package that depend on a function

Description

Returns a character vector of all functions in package package_name that depend on function fun.

Usage

```
find_function_dependencies(package_name, fun, include = "package only")
```

Arguments

package_name	package name as character string
fun	function name as character string
include	Should only package functions be returned ("package only"), or all functions except for base R operators ("no operators"), or all functions ("all")?

Value

character vector of function names

Examples

```
find_function_dependencies("codeInspectr", "find_recursive_dependencies")
body(codeInspectr::find_recursive_dependencies)
```

`find_recursive_dependencies`

Identify all functions that are called by the specified function, and all functions called by them, etc

Description

Recursively walk downwards through function calls. Two options to reduce traversal of functions that may not be of interest (e.g., "validate_.*" functions): `pattern_ignore` will completely ignore functions matching a

Usage

```
find_recursive_dependencies(  
  package_name,  
  fun,  
  pattern_ignore = NULL,  
  pattern_dont_recurse = NULL  
)
```

Arguments

<code>package_name</code>	Name of pattern. Character vector, length 1
<code>fun</code>	Name of function. Character vector, length 1
<code>pattern_ignore</code>	NULL or character vector of regex patterns. If present, <code>recursive_deps</code> will ignore any functions that match the pattern(s) character vector. Can take multiple patterns.
<code>pattern_dont_recurse</code>	NULL or character vector of regex patterns. If present, <code>recursive_deps</code> will list functions that match the pattern(s), but will not recurse into those functions.. Can take multiple patterns.

Value

character vector

Examples

```
find_recursive_dependencies("purrr", "map")
```

`find_reverse_dependencies`*List functions that call the specified functions*

Description

To search for functions that are dependent on a non-package function (e.g., tracking down potential issues if a function from another package changes), set `include` to "all".

Usage

```
find_reverse_dependencies(package_name, fun, include = "package only")
```

Arguments

<code>package_name</code>	package name as character string
<code>fun</code>	function name as character string
<code>include</code>	Should only package functions be returned ("package only"), or all functions except for base R operators ("no operators"), or all functions ("all")?

Value

character vector

Examples

```
find_reverse_dependencies("codeInspectr", "package_functions")
```

<code>get_branch_activity</code>	<i>Return active branches and the most recent updates of github repository</i>
----------------------------------	--

Description

Summarizes recent activity on a remote github repository, listing active branches and their most recent updates. # Will only process the first 100 branches of a repository. Repository does not need to be an R package, or even contain R code. Will not list activity for "gh-pages", which is the branch used for pkgdown.

Usage

```
get_branch_activity(repo_address)
```

Arguments

repo_address Github repository address, of the form "user/repository", as in "FRAMverse/framrsquared". Does not need to be an R package.

Value

Dataframe with \$branch (branch name) and \$most_recent_update (date of most recent commit)

Examples

```
get_branch_activity("FRAMverse/framrsquared")
get_branch_activity("tidyverse/dplyr")
```

get_github_activity	<i>List issues closed and pull requests opened or merged for a user or organization</i>
---------------------	---

Description

Searches all repositories owned by a Github user or organization for issues closed, pull requests opened, and pull requests merged within a date window. Github search returns at most 1,000 results per query; when a window holds more than that, it is split into smaller windows automatically.

Usage

```
get_github_activity(owner, since, until = Sys.Date(), author = NULL)
```

Arguments

owner	Github user or organization, e.g. "FRAMverse". Character atomic.
since	Start of the reporting window (inclusive). Date, or character in "YYYY-MM-DD" form.
until	End of the reporting window (inclusive). Date, or character in "YYYY-MM-DD" form. Optional, Defaults to today.
author	Optional Github username. If provided, pull request counts are restricted to pull requests authored by this person. Use "@me" for the account your Github token belongs to. Issue counts are not affected, since Github search cannot filter by who closed an issue. Character atomic, defaults to NULL (all authors).

Value

Tibble with one row per issue or pull request, with columns

- \$owner: the user or organization searched
- \$repo: repository address, e.g. "FRAMverse/framrsquared"
- \$activity: "issues_closed", "prs_opened", or "prs_merged". A pull request opened and merged in the window appears once under each.

- \$number, \$title, \$author: issue or pull request number, title, and author's Github username
- \$state_reason: for issues, "completed", "not_planned", or NA
- \$created, \$closed, \$merged: dates
- \$link: web address

Examples

```
## Not run:
get_github_activity("FRAMverse", since = "2026-01-01")

## End(Not run)
```

get_issues	<i>Return summary of open Github Issues</i>
------------	---

Description

Return summary of open Github Issues

Usage

```
get_issues(repo_address, max_char = 97)
```

Arguments

repo_address	Github repository address, of the form "user/repository", as in "FRAMverse/framrsquared". Does not need to be an R package.
max_char	Maximum number of characters to return in the "body" column.

Value

dataframe with \$title, \$body, and \$date of open issues.

Examples

```
get_issues("FRAMverse/framrsquared")
get_issues("tidyverse/dplyr")
```

get_new_repos	<i>List repositories created by a user or organization within a date window</i>
---------------	---

Description

Unlike `list_repos()`, this includes your own private repositories, since it uses the Github search API (which returns any repository your token can see).

Usage

```
get_new_repos(owner, since, until = Sys.Date(), include_forks = TRUE)
```

Arguments

owner	Github user or organization, e.g. "FRAMverse". Character atomic.
since	Start of the reporting window (inclusive). Date, or character in "YYYY-MM-DD" form.
until	End of the reporting window (inclusive). Date, or character in "YYYY-MM-DD" form. Optional, Defaults to today.
include_forks	Should forks count as new repos? Logical, defaults to TRUE.

Value

Tibble with \$owner, \$repo (e.g. "FRAMverse/framrsquared"), \$description, \$private, \$fork, \$created (date), and \$link.

Examples

```
## Not run:
get_new_repos("FRAMverse", since = "2026-01-01")

## End(Not run)
```

get_pull_requests	<i>Return summary of open Github Pull Requests</i>
-------------------	--

Description

Including title, initial text of body, and date

Usage

```
get_pull_requests(repo_address, max_char = 97)
```

Arguments

repo_address	Github repository address, of the form "user/repository", as in "FRAMverse/framrsquared". Does not need to be an R package.
max_char	Maximum number of characters to return in the "body" column.

Value

dataframe with \$title, \$body, and \$date of open pull requests.

Examples

```
get_pull_requests("FRAMverse/framrsquared")
get_pull_requests("tidyverse/dplyr")
```

get_raw_issues	<i>Obtain Github Issues from a repository</i>
----------------	---

Description

Returns github issues as a list. Github issues themselves are structured as jsons.

Usage

```
get_raw_issues(repo_address = NULL, entity_name = NULL, repo_name = NULL)
```

Arguments

repo_address	Github identifier in the form of {name}/{repository}, as in "FRAMverse/codeInspectr". Can also be in the form of full URL (e.g. "https://github.com/FRAMverse/framrsquared/"). Character string, defaults to NULL. If not provided, users must instead provide the same information in entity_name and repo_name.
entity_name	Only needed if repo_address is not provided. Name of a github user or organization repository, e.g., "FRAMverse". Character atomic, defaults to NULL.
repo_name	Only needed if repo_address is not provided. Name of individual repository, e.g., "codeInspectr". Character atomic, defaults to NULL.

Value

A list of class gh_response containing all issues from a repository.

Examples

```
temp <- get_raw_issues("cbedwards-dfw/codeInspectr")
```

get_raw_pulls	<i>Obtain Github pull requests from a repository</i>
---------------	--

Description

Returns github issues as a list. Github issues themselves are structured as jsons.

Usage

```
get_raw_pulls(repo_address = NULL, entity_name = NULL, repo_name = NULL)
```

Arguments

repo_address	Github identifier in the form of {name}/{repository}, as in "FRAMverse/codeInspectr". Can also be in the form of full URL (e.g. "https://github.com/FRAMverse/framrsquared/"). Character string, defaults to NULL. If not provided, users must instead provide the same information in entity_name and repo_name.
entity_name	Only needed if repo_address is not provided. Name of a github user or organization repository, e.g., "FRAMverse". Character atomic, defaults to NULL.
repo_name	Only needed if repo_address is not provided. Name of individual repository, e.g., "codeInspectr". Character atomic, defaults to NULL.

Value

A list of class gh_response containing all pull requests from a repository.

Examples

```
# temp <- get_raw_pulls("cbedwards-dfw/codeInspectr")
```

is_r_package	<i>Test if repository is an R package</i>
--------------	---

Description

Uses the existence of a DESCRIPTION file and the first line being "Package:..." to decide if the provided repository is an R package. Accesses first few lines of DESCRIPTION in raw.githubusercontent.com/ for speed.

Usage

```
is_r_package(repo_address, default_branch = "main")
```

Arguments

- repo_address Repository name including user or organization prefix, e.g., "FRAMverse/framrsquared". Character atomic
- default_branch Name of the default/primary repository. Character atomic, defaults to "main". Older repositories often use "master" as the default repo name.

Value

Logical: TRUE if the repo is an R package, FALSE if it is not.

Examples

```
# framrsquared is an R package
is_r_package("FRAMverse/framrsquared")
# the snippets repository is not.
is_r_package("FRAMverse/snippets")
```

list_repos	<i>List all repositories of a Github user or organization</i>
------------	---

Description

Creates a dataframe of repositories and associated info for the user or organization specified

Usage

```
list_repos(
  name,
  entity_type = c("user", "org"),
  repo_type = c("all", "public", "private", "forks"),
  r_package_status = FALSE
)
```

Arguments

- name Name of user or organization (e.g., "FRAMverse" or "cbedwards-dfw"). Character atomic.
- entity_type Is this a Github User ("user") or organization ("org")? Character atomic, defaults to "user".
- repo_type List "all" repos, "public" repos, "private" repos, or only "fork" repos? Character atomic, defaults to "all". Private repos will only be provided if you have access to the private repos of that user or org.
- r_package_status Check if each repository is an R package. Increases runtime considerably (e.g., ~1 sec per repo). Logical, defaults to FALSE

Value

Data frame of repositories with columns

- `$name`: repository name, e.g., "codeInspectr"
- `$full_name`: {user/org}/{repository name}, e.g., "FRAMverse/codeInspectr"
- `$html_url`: web address of repository
- `$description`: description listed for the repository.
- `$stars`: number of followers of the repository
- `$language`: programming language(s) of the repository.
- `$default_branch`: the name of the default branch (e.g., "main").
- `$r_package`: is the repository an R package? Logical, only present for optional argument `r_package_status = TRUE`.

Examples

```
list_repos("FRAMverse", entity_type = "org", repo_type = "public")
```

package_functions	<i>List the functions of a package</i>
-------------------	--

Description

List the functions of a package

Usage

```
package_functions(package_name)
```

Arguments

`package_name` Name of package as character string

Value

character vector of function names

Examples

```
package_functions("codeInspectr")
```

parse_name	<i>Helper function to combine repo names if needed</i>
------------	--

Description

Helper function to combine repo names if needed

Usage

```
parse_name(
  repo_address = NULL,
  entity_name = NULL,
  repo_name = NULL,
  call = rlang::caller_env()
)
```

Arguments

repo_address	Github identifier in the form of {name}/{repository}, as in "FRAMverse/codeInspectr". Can also be in the form of full URL (e.g. "https://github.com/FRAMverse/framrsquared/"). Character string, defaults to NULL. If not provided, users must instead provide the same information in entity_name and repo_name.
entity_name	Only needed if repo_address is not provided. Name of a github user or organization repository, e.g., "FRAMverse". Character atomic, defaults to NULL.
repo_name	Only needed if repo_address is not provided. Name of individual repository, e.g., "codeInspectr". Character atomic, defaults to NULL.
call	Ensure error messages mention the calling function.

Value

A full github name, as in "FRAMverse/codeInspectr"

plot_function_dependencies	<i>Plot dependency map of a function</i>
----------------------------	--

Description

Plot dependency map of a function

Usage

```
plot_function_dependencies(
  package_name,
  fun,
  pattern_ignore = NULL,
  pattern_dont_recurse = NULL
)
```

Arguments

`package_name` Name of pattern. Character vector, length 1

`fun` Name of function. Character vector, length 1

`pattern_ignore` NULL or character vector of regex patterns. If present, `recursive_deps` will ignore any functions that match the pattern(s) character vector. Can take multiple patterns.

`pattern_dont_recurse` NULL or character vector of regex patterns. If present, `recursive_deps` will list functions that match the pattern(s), but will not recurse into those functions.. Can take multiple patterns.

Value

visNetwork plot

Examples

```
plot_function_dependencies("purrr", "map")
plot_function_dependencies("purrr", "map_df")
```

process_repo

Helper function to download issues/pulls of single repo

Description

Intended for use in purr workflow in [backup_github_info\(\)](#).

Usage

```
process_repo(repo_name, repo_address, save_location)
```

Arguments

`repo_name` Name of repository (e.g. "codeInspectr")

`repo_address` Full github name (e.g., "FRAMverse/codeInspectr")

`save_location` Filepath for directory to save issue and pulls to.

Value

Tibble with filepaths and success/failure info.

safe_dir_delete	<i>Delete a directory, retrying on transient Windows file locks</i>
-----------------	---

Description

Useful for deleting temporary Github clones.

Usage

```
safe_dir_delete(path, attempts = 5, wait_seconds = 0.3)
```

Arguments

path	Directory to delete.
attempts	Number of attempts before giving up.
wait_seconds	Delay between attempts (grows each retry).

Value

TRUE if deletion succeeded, FALSE otherwise (invisibly attempted).

summarize_github_activity	<i>Summarize Github activity across all repositories of users or organizations</i>
---------------------------	--

Description

Counts issues closed, pull requests opened, pull requests merged, and new repositories created since a given date, across every repository owned by the users and/or organizations in owners. Uses the Github search API, so private repositories are included whenever your Github token can see them.

Usage

```
summarize_github_activity(
  owners,
  since,
  until = Sys.Date(),
  author = NULL,
  include_forks = TRUE,
  verbose = TRUE
)
```

Arguments

owners	Github users and/or organizations whose repositories should be summarized, e.g. <code>c("cbedwards-dfw", "FRAMverse")</code> . Character vector. Whether each owner is a user or an organization is detected automatically.
since	Start of the reporting window (inclusive). Date, or character in "YYYY-MM-DD" form.
until	End of the reporting window (inclusive). Date, or character in "YYYY-MM-DD" form. Optional, Defaults to today.
author	Optional Github username. If provided, pull request counts are restricted to pull requests authored by this person. Use "@me" for the account your Github token belongs to. Issue counts are not affected, since Github search cannot filter by who closed an issue. Character atomic, defaults to NULL (all authors).
include_forks	Count newly created forks as new repositories? Logical, defaults to TRUE.
verbose	Print a summary to the console? Logical, defaults to TRUE.

Value

A list with

- `$totals`: one row per owner (plus a "TOTAL" row) with columns `$issues_closed`, `$prs_opened`, `$prs_merged`, and `$repos_created`.
- `$by_repo`: the same issue and pull request counts for each repository with any activity in the window.
- `$items`: one row per issue or pull request counted, with links (see [get_github_activity\(\)](#)). Issues closed as "not planned" can be dropped with `dplyr::filter(items, state_reason != "not_planned")`.
- `$new_repos`: repositories created in the window (see [get_new_repos\(\)](#)).
- `$since`, `$until`: the reporting window.

See Also

[get_github_activity\(\)](#), [get_new_repos\(\)](#), [list_repos\(\)](#)

Examples

```
## Not run:
activity <- summarize_github_activity(c("cbedwards-dfw", "FRAMverse"),
                                     since = "2026-01-01")

activity$totals

## only count pull requests you wrote yourself
summarize_github_activity("FRAMverse", since = "2026-01-01", author = "@me")

## End(Not run)
```

summarize_repository	<i>Summarize information about a github R package</i>
----------------------	---

Description

Primarily designed to pull information from R packages, but also works for other repositories.

Usage

```
summarize_repository(repo_address, max_char = 70)
```

Arguments

repo_address	Github repository address for an R package, of the form "user/repository", as in "FRAMverse/framrsquared". Also accepts full URL, as in "https://github.com/FRAMverse/framrsquared".
max_char	Maximum number of characters to return in the "body" column.

Value

List summarizing github repository information.

See Also

[get_issues\(\)](#), [get_pull_requests\(\)](#), [get_branch_activity\(\)](#)

Examples

```
summarize_repository("FRAMverse/framrsquared")
summarize_repository("cbedwards-dfw/xldiff")
```

Index

backup_github_info, [2](#)
backup_github_info(), [18](#)
build_package_manual, [3](#)
build_package_network, [4](#)
build_recursive_edges, [5](#)

compile_package_manuals, [5](#)
compile_vignettes, [6](#)

find_function_dependencies, [7](#)
find_recursive_dependencies, [8](#)
find_reverse_dependencies, [9](#)

get_branch_activity, [9](#)
get_branch_activity(), [21](#)
get_github_activity, [10](#)
get_github_activity(), [20](#)
get_issues, [11](#)
get_issues(), [21](#)
get_new_repos, [12](#)
get_new_repos(), [20](#)
get_pull_requests, [12](#)
get_pull_requests(), [21](#)
get_raw_issues, [13](#)
get_raw_pulls, [14](#)

is_r_package, [14](#)

list_repos, [15](#)
list_repos(), [12](#), [20](#)

package_functions, [16](#)
parse_name, [17](#)
plot_function_dependencies, [17](#)
process_repo, [18](#)

safe_dir_delete, [19](#)
summarize_github_activity, [19](#)
summarize_repository, [21](#)