Package 'GitStats'

March 17, 2026

Title Standardized Git Repository Data

Version 2.4.0

Maintainer Maciej Banas <banasmaciek@gmail.com>

Description Obtain standardized data from multiple 'Git' services, including 'GitHub' and 'GitLab'. Designed to be 'Git' service-agnostic, this package assists teams with activities spread across various 'Git' platforms by providing a unified way to access repository data.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

URL https://r-world-devs.github.io/GitStats/, https://github.com/r-world-devs/GitStats

BugReports https://github.com/r-world-devs/GitStats/issues

Imports cli, dplyr, glue, httr2, lubridate (>= 1.8.0), rlang (>= 1.1.0), R6, purrr (>= 1.0.0), stringr

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

Config/testthat/edition 3

VignetteBuilder knitr

Language en-US

Depends R (>= 4.1.0)

NeedsCompilation no

Author Maciej Banas [aut, cre], Kamil Koziej [aut], Karolina Marcinkowska [aut], Matt Secrest [aut]

Repository CRAN

Date/Publication 2026-03-17 08:40:15 UTC

Contents

`create_gitstats` *Create a* `GitStats` *object*

Description

Create a `GitStats` object

Usage

```
create_gitstats()
```

Value

A `GitStats` object.

Examples

```
my_gitstats <- create_gitstats()
```

get_commits *Get data on commits*

Description

List all commits from all repositories for an organization or a vector of repositories.

Usage

```
get_commits(
  gitstats,
  since = NULL,
  until = Sys.Date(),
  cache = TRUE,
  verbose = FALSE,
  progress = TRUE
)
```

Arguments

gitstats	A GitStats object.
since	A starting date.
until	An end date.
cache	A logical, if set to TRUE GitStats will retrieve the last result from its storage.
verbose	A logical, TRUE by default. If FALSE messages and printing output is switched off.
progress	A logical, by default set to verbose value. If FALSE no cli progress bar will be displayed.

Value

A table of tibble and gitstats_commits classes.

Examples

```
## Not run:
my_gitstats <- create_gitstats() |>
  set_github_host(
    token = Sys.getenv("GITHUB_PAT"),
    repos = c("openpharma/DataFakeR", "openpharma/visR")
  ) |>
  set_gitlab_host(
    token = Sys.getenv("GITLAB_PAT_PUBLIC"),
    orgs = "mbtests"
  )
 get_commits(my_gitstats, since = "2018-01-01")

## End(Not run)
```

`get_commits_stats` *Get commits statistics*

Description

Prepare statistics from the pulled commits data.

Usage

```
get_commits_stats(
  commits,
  time_aggregation = c("year", "month", "week", "day"),
  group_var
)
```

Arguments

`commits`	A `gitstats_commits` S3 class table object (output of `get_commits()`).
`time_aggregation`	A character, specifying time aggregation of statistics.
`group_var`	Other grouping variable to be passed to `dplyr::group_by()` function apart from `stats_date` and `githost`. Could be: `author`, `author_login`, `author_name` or `organization`. Should be passed without quotation marks.

Details

To make function work, you need first to get commits data with `GitStats`. See examples section.

Value

A table of `commits_stats` class.

Examples

```
## Not run:
 my_gitstats <- create_gitstats() |>
   set_github_host(
     token = Sys.getenv("GITHUB_PAT"),
     repos = c("r-world-devs/GitStats", "openpharma/visR")
   ) |>
   get_commits(my_gitstats, since = "2022-01-01") |>
   get_commits_stats(
     time_aggregation = "year",
     group_var = author
   )

## End(Not run)
```

get_files *Get files*

Description

Pulls text files and their content.

Usage

```
get_files(
  gitstats,
  pattern = NULL,
  depth = Inf,
  file_path = NULL,
  cache = TRUE,
  verbose = FALSE,
  progress = TRUE
)
```

Arguments

gitstats	A GitStats object.
pattern	A regular expression. If defined, it pulls content of all files in a repository matching this pattern reaching to the level of directories defined by depth parameter. Works only if file_path stays NULL.
depth	Defines level of directories to retrieve files from. E.g. if set to 0, it will pull files only from root, if 1L, will take data from root directory and directories visible in root directory. If left with no argument, will pull files from all directories.
file_path	A specific path to file(s) in repositories. May be a character vector if multiple files are to be pulled. If defined, the function pulls content of this specific file_path. Can be defined only if pattern stays NULL.
cache	A logical, if set to TRUE GitStats will retrieve the last result from its storage.
verbose	A logical, TRUE by default. If FALSE messages and printing output is switched off.
progress	A logical, by default set to verbose value. If FALSE no cli progress bar will be displayed.

Details

get_files() may be used in two ways: either with pattern (with optional depth) or file_path argument defined.

In the first scenario, GitStats will pull first a files structure responding to the passed pattern and depth arguments and afterwards files content for all of these files. In the second scenario, GitStats will pull only the content of files for the specific file_path of the repository.

If user wants to pull a particular file or files, a `file_path` approach seems more reasonable, as it is a faster way since it omits pulling the whole file structure from the repo.

For example, if user wants to pull content of `README.md` and/or `NEWS.md` files placed in the `root` directories of the repositories, he should take the `file_path` approach as he already knows precisely paths of the files.

On the other hand, if user wants to pull specific type of files (e.g. all `.md` or `.Rmd` files in the repository), without knowing their path, it is recommended to use a `pattern` approach, which will trigger `GitStats` to find all the files in the repository on the given level of directories (`pattern` argument) and afterwards pull their content.

The latter approach is slower than the former but may be more useful depending on users' goals. Both approaches return data in the same format: `tibble` with data on `files`, namely their `path` and their `content`.

Value

A data.frame.

Examples

```
## Not run:
 git_stats <- create_gitstats() |>
  set_github_host(
    token = Sys.getenv("GITHUB_PAT"),
    orgs = c("r-world-devs")
  ) |>
  set_gitlab_host(
    token = Sys.getenv("GITLAB_PAT_PUBLIC"),
    orgs = "mbtests"
  )

 rmd_files <- get_files(
   gitstats = git_stats,
   pattern = "\\.Rmd",
   depth = 2L
 )

 app_files <- get_files(
   gitstats = git_stats,
   file_path = c("R/app.R", "R/ui.R", "R/server.R")
 )


## End(Not run)
```

`get_issues` *Get data on issues*

Description

List all issues from all repositories for an organization or a vector of repositories.

Usage

```
get_issues(
  gitstats,
  since = NULL,
  until = Sys.Date(),
  state = NULL,
  cache = TRUE,
  verbose = FALSE,
  progress = TRUE
)
```

Arguments

gitstats	A GitStats object.
since	A starting date.
until	An end date.
state	An optional character, by default NULL, may be set to "open" or "closed" if user wants one type of issues.
cache	A logical, if set to TRUE GitStats will retrieve the last result from its storage.
verbose	A logical, TRUE by default. If FALSE messages and printing output is switched off.
progress	A logical, by default set to verbose value. If FALSE no cli progress bar will be displayed.

Value

A table of tibble and gitstats_issues classes.

Examples

```
## Not run:
my_gitstats <- create_gitstats() |>
  set_github_host(
    token = Sys.getenv("GITHUB_PAT"),
    repos = c("openpharma/DataFakeR", "openpharma/visR")
  ) |>
  set_gitlab_host(
    token = Sys.getenv("GITLAB_PAT_PUBLIC"),
    orgs = "mbtests"
  )
 get_issues(my_gitstats, since = "2018-01-01", state = "open")

## End(Not run)
```

`get_issues_stats` *Get issues statistics*

Description

Prepare statistics from the pulled issues data.

Usage

```
get_issues_stats(
  issues,
  time_aggregation = c("year", "month", "week", "day"),
  group_var
)
```

Arguments

`issues`	A `gitstats_issue` S3 class table object (output of `get_issues()`).
`time_aggregation`	A character, specifying time aggregation of statistics.
`group_var`	Other grouping variable to be passed to `dplyr::group_by()` function apart from `stats_date` and `githost`. Could be: `author`, `state` or `organization`. Should be passed without quotation marks.

Details

To make function work, you need first to get issues data with `GitStats`. See examples section.

Value

A table of `issues_stats` class.

Examples

```
## Not run:
 my_gitstats <- create_gitstats() |>
   set_github_host(
     token = Sys.getenv("GITHUB_PAT"),
     repos = c("r-world-devs/GitStats", "openpharma/visR")
   ) |>
   get_issues(my_gitstats, since = "2022-01-01") |>
   get_issues_stats(
     time_aggregation = "month",
     group_var = state
   )

## End(Not run)
```

get_orgs *Get data on organizations*

Description

Pulls data on all organizations from a Git host and parses it into table format.

Usage

```
get_orgs(gitstats, cache = TRUE, verbose = FALSE)
```

Arguments

gitstats	A GitStats object.
cache	A logical, if set to TRUE GitStats will retrieve the last result from its storage.
verbose	A logical, TRUE by default. If FALSE messages and printing output is switched off.

Value

A data.frame.

Examples

```
## Not run:
my_gitstats <- create_gitstats() |>
  set_github_host(
    orgs = c("r-world-devs", "openpharma"),
    token = Sys.getenv("GITHUB_PAT")
  ) |>
  set_gitlab_host(
    orgs = "mbtests",
    token = Sys.getenv("GITLAB_PAT_PUBLIC")
  )
get_orgs(my_gitstats)

## End(Not run)
```

get_pull_requests *Get data on pull requests*

Description

List all pull requests from all repositories for an organization or a vector of repositories.

Usage

```
get_pull_requests(
  gitstats,
  since = NULL,
  until = Sys.Date(),
  state = NULL,
  cache = TRUE,
  verbose = FALSE,
  progress = TRUE
)
```

Arguments

gitstats	A GitStats object.
since	A starting date.
until	An end date.
state	An optional character, by default NULL, may be set to "open" or "closed" if user wants one type of issues.
cache	A logical, if set to TRUE GitStats will retrieve the last result from its storage.
verbose	A logical, TRUE by default. If FALSE messages and printing output is switched off.
progress	A logical, by default set to verbose value. If FALSE no cli progress bar will be displayed.

Value

A table of tibble and gitstats_pull_requests classes.

Examples

```
## Not run:
my_gitstats <- create_gitstats() |>
  set_github_host(
    token = Sys.getenv("GITHUB_PAT"),
    repos = c("openpharma/DataFakeR", "openpharma/visR")
  ) |>
  set_gitlab_host(
    token = Sys.getenv("GITLAB_PAT_PUBLIC"),
    orgs = "mbtests"
  )
 get_pull_requests(my_gitstats, since = "2018-01-01", state = "open")

## End(Not run)
```

`get_pull_requests_stats` *Get pull requests statistics*

Description

Prepare statistics from the pulled pull requests data.

Usage

```
get_pull_requests_stats(
  pull_requests,
  time_aggregation = c("year", "month", "week", "day"),
  group_var
)
```

Arguments

`pull_requests`	A `gitstats_pr_stats` S3 class table object (output of `get_pull_requests()`).
`time_aggregation`	A character, specifying time aggregation of statistics.
`group_var`	Other grouping variable to be passed to `dplyr::group_by()` function apart from `stats_date` and `githost`. Could be: `author`, `state` or `organization`. Should be passed without quotation marks.

Details

To make function work, you need first to get pull requests data with `GitStats`. See examples section.

Value

A table of `pull_requests_stats` class.

Examples

```
## Not run:
 my_gitstats <- create_gitstats() |>
   set_github_host(
     token = Sys.getenv("GITHUB_PAT"),
     repos = c("r-world-devs/GitStats", "openpharma/visR")
   ) |>
   get_pull_requests(my_gitstats, since = "2022-01-01") |>
   get_pull_requests_stats(
     time_aggregation = "month",
     group_var = author
   )

## End(Not run)
```

`get_release_logs` *Get release logs*

Description

Pull release logs from repositories.

Usage

```
get_release_logs(
  gitstats,
  since = NULL,
  until = Sys.Date(),
  cache = TRUE,
  verbose = FALSE,
  progress = TRUE
)
```

Arguments

`gitstats`	A `GitStats` object.
`since`	A starting date.
`until`	An end date.
`cache`	A logical, if set to `TRUE` GitStats will retrieve the last result from its storage.
`verbose`	A logical, `TRUE` by default. If `FALSE` messages and printing output is switched off.
`progress`	A logical, by default set to `verbose` value. If `FALSE` no `cli` progress bar will be displayed.

Value

A data.frame.

Examples

```
## Not run:
 my_gitstats <- create_gitstats() |>
  set_github_host(
    token = Sys.getenv("GITHUB_PAT"),
    orgs = c("r-world-devs", "openpharma")
  )
  get_release_logs(my_gistats, since = "2024-01-01")

## End(Not run)
```

`get_repos` *Get data on repositories*

Description

Pulls data on all repositories for an organization, individual user or those with a given text in code blobs (`with_code` parameter) or a file (`with_files` parameter) and parse it into table format.

Usage

```
get_repos(
  gitstats,
  add_contributors = TRUE,
  with_code = NULL,
  in_files = NULL,
  with_files = NULL,
  language = NULL,
  cache = TRUE,
  verbose = FALSE,
  progress = TRUE
)
```

Arguments

`gitstats`	A GitStats object.
`add_contributors`	A logical parameter to decide whether to add information about repositories' contributors to the repositories output (table). If set to `FALSE` it makes function run faster as, in the case of `org` search mode, it reaches only `GraphQL` endpoint with a query on repositories, and in the case of `code` search mode it reaches only `repositories REST API` endpoint. However, the pitfall is that the result does not convey information on contributors. When set to `TRUE` (by default), `GitStats` iterates additionally over pulled repositories and reaches to the `contributors APIs`, which makes it slower, but gives additional information.
`with_code`	A character vector, if defined, GitStats will pull repositories with specified code phrases in code blobs.
`in_files`	A character vector of file names. Works when `with_code` is set - then it searches code blobs only in files passed to `in_files` parameter.
`with_files`	A character vector, if defined, GitStats will pull repositories with specified files.
`language`	A character. If defined, GitStats will return only repositories with given language.
`cache`	A logical, if set to `TRUE` GitStats will retrieve the last result from its storage.

verbose	A logical, TRUE by default. If FALSE messages and printing output is switched off.
progress	A logical, by default set to verbose value. If FALSE no cli progress bar will be displayed.

Value

A data.frame.

Examples

```
## Not run:
my_gitstats <- create_gitstats() |>
  set_github_host(
    token = Sys.getenv("GITHUB_PAT"),
    orgs = c("r-world-devs", "openpharma")
  ) |>
  set_gitlab_host(
    token = Sys.getenv("GITLAB_PAT_PUBLIC"),
    orgs = "mbtests"
  )
get_repos(my_gitstats)
get_repos(my_gitstats, add_contributors = FALSE)
get_repos(my_gitstats, with_code = "Shiny", in_files = "renv.lock")
get_repos(my_gitstats, with_files = "DESCRIPTION")

## End(Not run)
```

---

get_repos_trees *Get data on files trees across repositories*

Description

Pulls files tree (structure) per repository. Files trees are then stored as character vectors in files_tree column of output table.

Usage

```
get_repos_trees(
  gitstats,
  pattern = NULL,
  depth = Inf,
  cache = TRUE,
  verbose = FALSE,
  progress = TRUE
)
```

Arguments

gitstats	A GitStats object.
pattern	A regular expression. If defined, it pulls structure of files in a repository matching this pattern reaching to the level of directories defined by depth parameter.
depth	Defines level of directories to reach for files structure from. E.g. if set to 0, it will pull files tree only from root, if 1L, will take data from root directory and directories visible in root directory. If left with no argument, will pull files tree down to every directory in a repo.
cache	A logical, if set to TRUE GitStats will retrieve the last result from its storage.
verbose	A logical, TRUE by default. If FALSE messages and printing output is switched off.
progress	A logical, by default set to verbose value. If FALSE no cli progress bar will be displayed.

Value

A tibble.

Examples

```
## Not run:
 my_gitstats <- create_gitstats() |>
  set_github_host(
    token = Sys.getenv("GITHUB_PAT"),
    orgs = c("r-world-devs", "openpharma")
  )

 get_repos_trees(
   gitstats = my_gitstats,
   pattern = "\\.md"
 )

## End(Not run)
```

---

get_repos_urls *Get repository URLS*

Description

Pulls a vector of repositories URLs (web or API): either all for an organization or those with a given text in code blobs (with_code parameter) or a file (with_files parameter).

Usage

```
get_repos_urls(
  gitstats,
  type = "api",
  with_code = NULL,
  in_files = NULL,
  with_files = NULL,
  cache = TRUE,
  verbose = FALSE,
  progress = TRUE
)
```

Arguments

gitstats	A GitStats object.
type	A character, choose if `api` or `web` (`html`) URLs should be returned. `api` type is set by default as setting `web` results in parsing which may be time consuming in case of large number of repositories.
with_code	A character vector, if defined, `GitStats` will pull repositories with specified code phrases in code blobs.
in_files	A character vector of file names. Works when `with_code` is set - then it searches code blobs only in files passed to `in_files` parameter.
with_files	A character vector, if defined, GitStats will pull repositories with specified files.
cache	A logical, if set to `TRUE` GitStats will retrieve the last result from its storage.
verbose	A logical, `TRUE` by default. If `FALSE` messages and printing output is switched off.
progress	A logical, by default set to `verbose` value. If `FALSE` no `cli` progress bar will be displayed.

Value

A character vector.

Examples

```
## Not run:
my_gitstats <- create_gitstats() |>
  set_github_host(
    token = Sys.getenv("GITHUB_PAT"),
    orgs = c("r-world-devs", "openpharma")
  ) |>
  set_gitlab_host(
    token = Sys.getenv("GITLAB_PAT_PUBLIC"),
    orgs = "mbtests"
  )
get_repos_urls(my_gitstats, with_files = c("DESCRIPTION", "LICENSE"))

## End(Not run)
```

get_storage *Get data from* GitStats *storage*

Description

Retrieves whole or particular data (see storage parameter) pulled earlier with GitStats.

Usage

```
get_storage(gitstats, storage = NULL)
```

Arguments

gitstats	A GitStats object.
storage	A character, type of the data you want to get from storage: commits, repositories, release_logs, users, files, files_structure, R_package_usage or release_logs.

Value

A list of tibbles (if storage set to NULL) or a tibble (if storage defined).

Examples

```
## Not run:
 my_gitstats <- create_gitstats() |>
  set_github_host(
    token = Sys.getenv("GITHUB_PAT"),
    orgs = c("r-world-devs", "openpharma")
  )
  get_release_logs(my_gistats, since = "2024-01-01")
  get_repos(my_gitstats)

  release_logs <- get_storage(
    gitstats = my_gitstats,
    storage = "release_logs"
  )

## End(Not run)
```

`get_users` *Get users data*

Description

Get users data

Usage

```
get_users(gitstats, logins, cache = TRUE, verbose = FALSE)
```

Arguments

`gitstats`	A GitStats object.
`logins`	A character vector of logins.
`cache`	A logical, if set to `TRUE` GitStats will retrieve the last result from its storage.
`verbose`	A logical, `TRUE` by default. If `FALSE` messages and printing output is switched off.

Value

A data.frame.

Examples

```
## Not run:
 my_gitstats <- create_gitstats() |>
  set_github_host(
    token = Sys.getenv("GITHUB_PAT"),
    orgs = c("r-world-devs")
  ) |>
  set_gitlab_host(
    token = Sys.getenv("GITLAB_PAT_PUBLIC"),
    orgs = "mbtests"
  )
 get_users(my_gitstats, c("maciekabanas", "marcinkowskak"))

## End(Not run)
```

is_verbose *Is verbose mode switched on*

Description

Is verbose mode switched on

Usage

```
is_verbose(gitstats)
```

Arguments

gitstats	A GitStats object.

set_github_host *Set GitHub host*

Description

Set GitHub host

Usage

```
set_github_host(
  gitstats,
  host = NULL,
  token = NULL,
  orgs = NULL,
  repos = NULL,
  verbose = is_verbose(gitstats),
  .error = TRUE
)
```

Arguments

gitstats	A GitStats object.
host	A character, optional, URL name of the host. If not passed, a public host will be used.
token	A token.
orgs	An optional character vector of organisations. If you pass it, repos parameter should stay NULL.
repos	An optional character vector of repositories full names (organization and repository name, e.g. "r-world-devs/GitStats"). If you pass it, orgs parameter should stay NULL.

verbose	A logical, TRUE by default. If FALSE messages and printing output is switched off.
.error	A logical to control if passing wrong input (repositories and organizations) should end with an error or not.

Details

If you do not define orgs and repos, GitStats will be set to scan whole Git platform (such as enterprise version of GitHub or GitLab), unless it is a public platform. In case of a public one (like GitHub) you need to define orgs or repos as scanning through all organizations may take large amount of time.

Value

A GitStats object with added information on host.

Examples

```
## Not run:
my_gitstats <- create_gitstats() |>
  set_github_host(
    orgs = c("r-world-devs", "openpharma", "pharmaverse")
  )

## End(Not run)
```

---

set_gitlab_host *Set GitLab host*

Description

Set GitLab host

Usage

```
set_gitlab_host(
  gitstats,
  host = NULL,
  token = NULL,
  orgs = NULL,
  repos = NULL,
  verbose = is_verbose(gitstats),
  .error = TRUE
)
```

Arguments

gitstats	A GitStats object.
host	A character, optional, URL name of the host. If not passed, a public host will be used.
token	A token.
orgs	An optional character vector of organisations. If you pass it, repos parameter should stay NULL.
repos	An optional character vector of repositories full names (organization and repository name, e.g. "r-world-devs/GitStats"). If you pass it, orgs parameter should stay NULL.
verbose	A logical, TRUE by default. If FALSE messages and printing output is switched off.
.error	A logical to control if passing wrong input (repositories and organizations) should end with an error or not.

Details

If you do not define orgs and repos, GitStats will be set to scan whole Git platform (such as enterprise version of GitHub or GitLab), unless it is a public platform. In case of a public one (like GitHub) you need to define orgs or repos as scanning through all organizations may take large amount of time.

Value

A GitStats object with added information on host.

Examples

```
## Not run:
my_gitstats <- create_gitstats() |>
  set_gitlab_host(
    token = Sys.getenv("GITLAB_PAT_PUBLIC"),
    orgs = "mbtests"
  )

## End(Not run)
```

---

show_hosts *Show hosts set in* GitStats

Description

Retrieves hosts set by GitStats with set_*_host() functions.

Usage

```
show_hosts(gitstats)
```

Arguments

gitstats	A GitStats object.

Value

A list of hosts.

---

show_orgs *Show organizations set in* GitStats

Description

Retrieves organizations set or pulled by GitStats. Especially helpful when user is scanning whole git platform and wants to have a glimpse at organizations.

Usage

```
show_orgs(gitstats)
```

Arguments

gitstats	A GitStats object.

Value

A vector of organizations.

---

verbose_off *Switch off verbose mode*

Description

Stop printing messages and output.

Usage

```
verbose_off(gitstats)
```

Arguments

gitstats	A GitStats object.

Value

A GitStats object.

verbose_on *Switch on verbose mode*

Description

Print all messages and output.

Usage

```
verbose_on(gitstats)
```

Arguments

gitstats	A GitStats object.

Value

A GitStats object.

Index