# Package: trackdown (via r-universe)

October 21, 2024

**Title** Collaborative Editing of Rmd (or Quarto / Rnw) Documents in Google Drive

**Version** 1.5.1

**Description** Collaborative writing and editing of R Markdown (or Quarto / Sweave) documents. The local .Rmd (or Quarto / .Rnw) is uploaded as a plain-text file to Google Drive. By taking advantage of the easily readable Markdown (or LaTeX) syntax and the well-known online interface offered by Google Docs, collaborators can easily contribute to the writing and editing process. After integrating all authors' contributions, the final document can be downloaded and rendered locally.

**Depends** R (>= 3.5)

**License** GPL-3

**Encoding** UTF-8

**Imports** cli (>= 3.0.0), gargle (>= 1.4.0), googledrive (>= 2.1.0), lifecycle, rlang (>= 1.0.2), rmarkdown

**Suggests** covr, devtools, httr, knitr, pagedown, sodium, testthat (>= 3.0.0), vcr

**Config/testthat/edition** 3

**RoxygenNote** 7.2.3

**VignetteBuilder** knitr

**URL** https://github.com/claudiozandonella/trackdown/, https://claudiozandonella.github.io/trackdown/

**BugReports** https://github.com/claudiozandonella/trackdown/issues

**Roxygen** list(markdown = TRUE)

**Repository** https://claudiozandonella.r-universe.dev

**RemoteUrl** https://github.com/claudiozandonella/trackdown

**RemoteRef** HEAD

**RemoteSha** 242d7ff615721eca613937c3765d7fbcc6b9c0b7

# Contents

## `download_file` *Downloads from Google Docs*

### Description

Download edited version of a file from Google Drive updating the local version with the new changes.

*Use with caution as local version of the file will be overwritten!*

To know more about `trackdown` workflow and features see `trackdown-package()` help page.

### Usage

```
download_file(
  file,
  gfile = NULL,
  gpath = "trackdown",
  shared_drive = NULL,
  rm_gcomments = FALSE,
  force = FALSE
)
```

### Arguments

| | |
|---|---|
| `file` | character. The path of a local `.Rmd`, Quarto, or `.Rnw` file. |
| `gfile` | character. The name of a Google Drive file (defaults to local file name). |
| `gpath` | character. (Sub)directory in My Drive or a shared drive (optional). By default files are uploaded in the folder "trackdown". To specify another folder the full path is required (e.g., "trackdown/my_folder"). Use `NULL` to upload directly at the root level, although it is not recommended. |
| `shared_drive` | character. The name of a Google Drive shared drive (optional). |

| | |
|---|---|
| `rm_gcomments` | **[Experimental]** logical value indicating whether or not to remove Google comments. |
| `force` | logical value indicating whether to skip confirm check by user (default is `FALSE`). |

### Value

`TRUE` if file from Google Drive was saved, `FALSE` otherwise.

## `render_file` *Render file from Google Drive*

### Description

Render file from Google Drive if there have been edits

To know more about `trackdown` workflow and features see `trackdown-package()` help page.

### Usage

```
render_file(
  file,
  gfile = NULL,
  gpath = "trackdown",
  shared_drive = NULL,
  force = FALSE,
  rm_gcomments = FALSE
)
```

### Arguments

| | |
|---|---|
| `file` | character. The path of a local `.Rmd`, Quarto, or `.Rnw` file. |
| `gfile` | character. The name of a Google Drive file (defaults to local file name). |
| `gpath` | character. (Sub)directory in My Drive or a shared drive (optional). By default files are uploaded in the folder "trackdown". To specify another folder the full path is required (e.g., "trackdown/my_folder"). Use `NULL` to upload directly at the root level, although it is not recommended. |
| `shared_drive` | character. The name of a Google Drive shared drive (optional). |
| `force` | logical value indicating whether to skip confirm check by user (default is `FALSE`). |
| `rm_gcomments` | **[Experimental]** logical value indicating whether or not to remove Google comments. |

### Value

`TRUE` if file from Google Drive was saved and rendered, `FALSE` otherwise.

# trackdown_auth *Authorize trackdown*

## Description

Authorize trackdown to view and manage your trackdown files. This function is a wrapper around `gargle::token_fetch()`.

By default, you are directed to a web browser, asked to sign in to your Google account, and to grant trackdown permission to operate on your behalf with trackdown. By default, with your permission, these user credentials are cached in a folder below your home directory, from where they can be automatically refreshed, as necessary. Storage at the user level means the same token can be used across multiple projects and tokens are less likely to be synced to the cloud by accident.

If you are interacting with R within a browser (applies to RStudio Server, Posit Workbench, and Posit Cloud), you need a variant of this flow, known as out-of-band auth ("oob"). If this does not happen automatically, you can request it yourself with `use_oob = TRUE` or, more persistently, by setting an option via `options(gargle_oob_default = TRUE)`.

## Usage

```
trackdown_auth(
  email = gargle::gargle_oauth_email(),
  path = NULL,
  scopes = "https://www.googleapis.com/auth/drive",
  cache = gargle::gargle_oauth_cache(),
  use_oob = gargle::gargle_oob_default(),
  token = NULL
)
```

## Arguments

`email` Optional. Allows user to target a specific Google identity. If specified, this is used for token lookup, i.e. to determine if a suitable token is already available in the cache. If no such token is found, `email` is used to pre-select the targetted Google identity in the OAuth chooser. Note, however, that the email associated with a token when it's cached is always determined from the token itself, never from this argument. Use `NA` or `FALSE` to match nothing and force the OAuth dance in the browser. Use `TRUE` to allow email auto-discovery, if exactly one matching token is found in the cache. Specify just the domain with a glob pattern, e.g. `"*@example.com"`, to create code that "just works" for both `alice@example.com` and `bob@example.com`. Defaults to the option named "gargle_oauth_email", retrieved by `gargle_oauth_email()`.

`path` JSON identifying the service account, in one of the forms supported for the `txt` argument of `jsonlite::fromJSON()` (typically, a file path or JSON string).

`scopes` A character vector of scopes to request. Pick from those listed at `https://developers.google.com/identity/protocols/oauth2/scopes`.

For certain token flows, the "https://www.googleapis.com/auth/userinfo.email" scope is unconditionally included. This grants permission to retrieve the email address associated with a token; gargle uses this to index cached OAuth tokens. This grants no permission to view or send email and is generally considered a low-value scope.

cache Specifies the OAuth token cache. Defaults to the option named "gargle_oauth_cache", retrieved via gargle_oauth_cache().

use_oob Whether to prefer out-of-band authentication. Defaults to the value returned by gargle_oob_default().

token A token with class Token2.0 or an object of httr's class request, i.e. a token that has been prepared with httr::config() and has a Token2.0 in the auth_token component.

### Details

Most users, most of the time, do not need to call trackdown_auth() explicitly – it is triggered by the first action that requires authorization. Even when called, the default arguments often suffice. However, when necessary, this function allows the user to explicitly:

- Declare which Google identity to use, via an email address. If there are multiple cached tokens, this can clarify which one to use. It can also force trackdown to switch from one identity to another. If there's no cached token for the email, this triggers a return to the browser to choose the identity and give consent. You can specify just the domain by using a glob pattern. This means that a script containing email = "*@example.com" can be run without further tweaks on the machine of either alice@example.com or bob@example.com.
- Use a service account token or workload identity federation.
- Bring their own Token2.0.
- Specify non-default behavior re: token caching and out-of-bound authentication.
- Customize scopes.

For details on the many ways to find a token, see gargle::token_fetch(). For deeper control over auth, use trackdown_auth_configure() to bring your own OAuth client or API key. Read more about gargle options, see gargle::gargle_options.

### See Also

Other auth functions: trackdown_auth_configure(), trackdown_deauth()

### Examples

```
## Not run:
# load/refresh existing credentials, if available
# otherwise, go to browser for authentication and authorization
trackdown_auth()

# see user associated with current token
trackdown_user()
```

```
# force use of a token associated with a specific email
trackdown_auth(email = "myemail@example.com")
trackdown_user()

# force a menu where you can choose from existing tokens or
# choose to get a new one
trackdown_auth(email = NA)

# use a 'read only' scope, so it's impossible to edit or delete files
# NOTE thaat this will allow only to download files from the drive
trackdown_auth(
  scopes = "https://www.googleapis.com/auth/drive.readonly"
)

# use a service account token
trackdown_auth(path = "foofy-83ee9e7c9c48.json")

## End(Not run)
```

## trackdown_auth_configure

*Edit and view auth configuration*

### Description

These functions give more control over and visibility into the auth configuration than trackdown_auth() does. trackdown_auth_configure() lets the user specify their own:

- OAuth client, which is used when obtaining a user token.

See the vignette("get-api-credentials", package = "gargle") for more. If the user does not configure these settings, internal defaults are used.

trackdown_oauth_client() retrieves the currently configured OAuth client.

### Usage

```
trackdown_auth_configure(client, path, app = deprecated())

trackdown_oauth_client()
```

### Arguments

| | |
|---|---|
| client | A Google OAuth client, presumably constructed via gargle::gargle_oauth_client_from_json(). Note, however, that it is preferred to specify the client with JSON, using the path argument. |
| path | JSON downloaded from Google Cloud Console, containing a client id and secret, in one of the forms supported for the txt argument of jsonlite::fromJSON() (typically, a file path or JSON string). |
| app | **[Deprecated]** Replaced by the client argument. |

### Value

- `trackdown_auth_configure()`: An object of R6 class gargle::AuthState, invisibly.
- `trackdown_oauth_client()`: the current user-configured OAuth client.

### See Also

Other auth functions: `trackdown_auth()`, `trackdown_deauth()`

### Examples

```
# see the current user-configured OAuth client (probaby `NULL`)
trackdown_oauth_client()

if (require(httr)) {

  # store current state, so we can restore
  original_client <- trackdown_oauth_client()

  # bring your own client via client id (aka key) and secret
  google_client <- gargle::gargle_oauth_client(
    id = "123456789.apps.googleusercontent.com",
    secret = "abcdefghijklmnopqrstuvwxyz",
    name = "my-awesome-google-api-wrapping-package",
  )
  trackdown_auth_configure(client = google_client)

  # confirm current client
  trackdown_oauth_client()

  # restore original state
  trackdown_auth_configure(client = original_client)
  trackdown_oauth_client()
}

## Not run:
# bring your own client via JSON downloaded from Google Developers Console
trackdown_auth_configure(
  path = "/path/to/the/JSON/you/downloaded/from/google/dev/console.json"
)

## End(Not run)
```

## `trackdown_deauth` *Clear current token*

### Description

Clears any currently stored token. The next time trackdown needs a token, the token acquisition process starts over, with a fresh call to `trackdown_auth()` and, therefore, internally, a call to `gargle::token_fetch()`. Unlike some other packages that use gargle, trackdown is not usable in a de-authorized state. Therefore, calling `trackdown_deauth()` only clears the token, i.e. it does NOT imply that subsequent requests are made with an API key in lieu of a token.

### Usage

```
trackdown_deauth()
```

### See Also

Other auth functions: `trackdown_auth_configure()`, `trackdown_auth()`

### Examples

```
## Not run:
trackdown_deauth()
trackdown_user()

## End(Not run)
```

---

## `trackdown_has_token` *Is there a token on hand?*

---

### Description

Reports whether trackdown has stored a token, ready for use in downstream requests.

### Usage

```
trackdown_has_token()
```

### Value

Logical.

### See Also

Other low-level API functions: `trackdown_token()`

### Examples

```
trackdown_has_token()
```

## `trackdown_token` *Produce configured token*

### Description

For internal use or for those programming around the trackdown API. Returns a token pre-processed with `httr::config()`. Most users do not need to handle tokens "by hand" or, even if they need some control, `trackdown_auth()` is what they need. If there is no current token, `trackdown_auth()` is called to either load from cache or initiate OAuth2.0 flow. If auth has been deactivated via `trackdown_deauth()`, `trackdown_token()` returns `NULL`.

### Usage

```
trackdown_token()
```

### Value

A `request` object (an S3 class provided by httr).

### See Also

Other low-level API functions: `trackdown_has_token()`

### Examples

```
## Not run:
trackdown_token()

## End(Not run)
```

## `trackdown_user` *Get info on current user*

### Description

Reveals the email address of the user associated with the current token. If no token has been loaded yet, this function does not initiate auth.

### Usage

```
trackdown_user()
```

### Value

An email address or, if no token has been loaded, `NULL`.

## See Also

gargle::token_userinfo(), gargle::token_email(), gargle::token_tokeninfo()

## Examples

```
## Not run:
trackdown_user()

## End(Not run)
```

# update_file *Updates file in Google Drive*

## Description

Replaces the content of an existing file in Google Drive with the contents of a local file. It is also possible to update (or upload if not already present) the output (pdf or html) of the file specifying the `path_output` argument. In case of html files, if pagedown package and Chrome are available, users can decide to upload a pdf version of the html file.

*Use with caution as tracked changes in the Google Drive file will be lost!*

To know more about `trackdown` workflow and features see `trackdown-package()` help page.

## Usage

```
update_file(
  file,
  gfile = NULL,
  gpath = "trackdown",
  shared_drive = NULL,
  hide_code = FALSE,
  path_output = NULL,
  rich_text = TRUE,
  rich_text_par = NULL,
  force = FALSE,
  open = rlang::is_interactive()
)
```

## Arguments

`file` character. The path of a local `.Rmd`, Quarto, or `.Rnw` file.

`gfile` character. The name of a Google Drive file (defaults to local file name).

`gpath` character. (Sub)directory in My Drive or a shared drive (optional). By default files are uploaded in the folder "trackdown". To specify another folder the full path is required (e.g., "trackdown/my_folder"). Use `NULL` to upload directly at the root level, although it is not recommended.

`shared_drive` character. The name of a Google Drive shared drive (optional).

`hide_code` logical value indicating whether to remove code from the text document (chunks and header). Placeholders of type `"[[chunk-<name>]]"` are displayed instead.

`path_output` default `NULL`, specify the path to the output to upload together with the other file. PDF are directly uploaded, HTML can be first converted into PDF if package `pagedown` and Chrome are available.

`rich_text` **[Experimental]** logical value (default is `TRUE`) indicating whether to upload to Google Docs a rich document (i.e., important text that should not be changed is highlighted). See “Rich Text” in details section.

`rich_text_par` **[Experimental]** argument used to pass a list with custom settings for rich_text. See “Rich Text” in details section.

`force` logical value indicating whether to skip confirm check by user (default is `FALSE`).

`open` logical value indicating whether to open the created document in a browser (default is `TRUE` in interactive sessions).

## Details

### Rich Text [Experimental]

The `rich_text` option (default is `TRUE`) allows to upload a rich document to Google Docs. Important text that should not be changed is highlighted. This includes, added instructions at the top of the document, placeholders hiding the code, header of the document (YAML header or LaTeX preamble), code chunks, and in-line code.

Default colour is opaque yellow. You can customize the colour specifying the `rgb_color` option in the `rich_text_par` argument. The `rgb_color` has to be a list with elements `red`, `green`, and `blue`. Each element has to be a numeric value between 0 and 1. See example below.

## Value

a dribble of the uploaded file (and output if specified).

## Examples

```
## Not run:

# Change default color to opaque light-blue
update_file(file = "path-to/my-file", rich_text = TRUE,
            rich_text_par = list(rgb_color = list(red = 102/255,
                                                  green = 204/255,
                                                  blue = 255/255)))

## End(Not run)
```

## `upload_file` *Upload file to Google Drive for collaborative writing and editing*

### Description

Uploads a local file to Google Drive as a plain text document. Will only upload the file if it doesn't already exist in the chosen location. By default files are uploaded in the folder "trackdown", if is not available on Google Drive, permission to create it is required to the user. To update an already existing file see `update_file()`. It is also possible to upload the output (pdf or html) of the file specifying the `path_output` argument. In case of html files, if `pagedown` package and Chrome are available, users can decide to upload a pdf version of the html file.

To know more about `trackdown` workflow and features see `trackdown-package()` help page.

### Usage

```
upload_file(
  file,
  gfile = NULL,
  gpath = "trackdown",
  shared_drive = NULL,
  hide_code = FALSE,
  path_output = NULL,
  rich_text = TRUE,
  rich_text_par = NULL,
  force = FALSE,
  open = rlang::is_interactive()
)
```

### Arguments

`file` character. The path of a local `.Rmd`, Quarto, or `.Rnw` file.

`gfile` character. The name of a Google Drive file (defaults to local file name).

`gpath` character. (Sub)directory in My Drive or a shared drive (optional). By default files are uploaded in the folder "trackdown". To specify another folder the full path is required (e.g., "trackdown/my_folder"). Use `NULL` to upload directly at the root level, although it is not recommended.

`shared_drive` character. The name of a Google Drive shared drive (optional).

`hide_code` logical value indicating whether to remove code from the text document (chunks and header). Placeholders of type `"[[chunk-<name>]]"` are displayed instead.

`path_output` default `NULL`, specify the path to the output to upload together with the other file. PDF are directly uploaded, HTML can be first converted into PDF if package `pagedown` and Chrome are available.

`rich_text` **[Experimental]** logical value (default is `TRUE`) indicating whether to upload to Google Docs a rich document (i.e., important text that should not be changed is highlighted). See “Rich Text” in details section.

`rich_text_par` **[Experimental]** argument used to pass a list with custom settings for rich_text. See “Rich Text” in details section.

`force` logical value indicating whether to skip confirm check by user (default is `FALSE`).

`open` logical value indicating whether to open the created document in a browser (default is `TRUE` in interactive sessions).

## Details

### Rich Text [Experimental]

The `rich_text` option (default is `TRUE`) allows to upload a rich document to Google Docs. Important text that should not be changed is highlighted. This includes, added instructions at the top of the document, placeholders hiding the code, header of the document (YAML header or LaTeX preamble), code chunks, and in-line code.

Default colour is opaque yellow. You can customize the colour specifying the `rgb_color` option in the `rich_text_par` argument. The `rgb_color` has to be a list with elements `red`, `green`, and `blue`. Each element has to be a numeric value between 0 and 1. See example below.

## Value

a dribble of the uploaded file (and output if specified).

## Examples

```
## Not run:

# Change default color to opaque light-blue
upload_file(file = "path-to/my-file", rich_text = TRUE,
            rich_text_par = list(rgb_color = list(red = 102/255,
                                                  green = 204/255,
                                                  blue = 255/255)))

## End(Not run)
```

# Index