Package: rbackupr (via r-universe)

September 18, 2024

Title An R package to backup folders to Google Drive with limited permissions

Version 0.1.0.9001

Description Backup files and folders to Google Drive without giving access to all of your drive.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports googledrive, magrittr, dplyr, stringr, fs, readr, tibble, usethis, RSQLite, purrr, httr, gargle, glue

Depends R (>= 2.10)

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.2.3

URL https://giocomai.github.io/rbackupr/

Repository https://giocomai.r-universe.dev

RemoteUrl https://github.com/giocomai/rbackupr

RemoteRef HEAD

RemoteSha f10f3e219d8b3958f4d62b6357456d402d883527

Contents

`rb_add_file_to_cache` *Add file to cache*

Description

Add file to cache

Usage

```
rb_add_file_to_cache(dribble, parent_id, project = NULL)
```

Arguments

`dribble`	A dribble.
`parent_id`	Identifier of the parent folder
`project`	Defaults to NULL. Can be set once per session with 'rb_get_project_name()'. If given, must be a character vector of length one: name of the project.

`rb_add_folder_to_cache` *Add folder to cache*

Description

Add folder to cache

Usage

```
rb_add_folder_to_cache(
  dribble,
  parent_id,
  relative_path,
  project = NULL,
  cache = TRUE
)
```

Arguments

dribble	A dribble.
parent_id	Identifier of the parent folder.
relative_path	Path relative to base folder.
project	Defaults to NULL. Can be set once per session with ‘rb_get_project_name()‘. If given, must be a character vector of length one: name of the project.

---

rb_backup *Backup files*

Description

Backup files

Usage

```
rb_backup(
  path,
  project = NULL,
  first_level_folders = NULL,
  first_level_files = TRUE,
  max_level = 100,
  recurse = TRUE,
  glob = NULL,
  create = TRUE,
  update = FALSE,
  cache = TRUE,
  base_folder = "rbackupr"
)
```

Arguments

path	Local path where files to backup are stored.
project	Defaults to NULL. Can be set once per session with ‘rb_get_project_name()‘. If given, must be a character vector of length one: name of the project.

first_level_folders
: Defaults to NULL. If given, clarifies which folders within the path should be uploaded, keeping the folder structure.

first_level_files
: Logical, defaults to TRUE. If FALSE, first level files (files that are directly under the project folder, rather than a subfolder) are not included in the backup.

max_level
: Defaults to 100. Maximum level of sub-folders to backup. Default means it will go 100 times deep into sub-folders. Used also to prevent infinite loops.

recurse
: Defaults to TRUE. Recurse up to one level.

glob
: Defaults to NULL. Can be used to filter type of files to upload, e.g. "*.jpg"

create
: Logical, defaults to TRUE. Create folders if missing. Set to FALSE if you are sure there are no new folders to raise an error if something unexpected happens.

update
: Logical, defaults to FALSE. If TRUE, checks on Google Drive for newly updated files or folders, otherwise it assumes that only files and folders listed in cache exist online.

cache
: Logical, defaults to TRUE. Stores locally cached information about base and project folder.

base_folder
: Name of base folder, defaults to ‘rbackupr‘

Examples

```
## Not run:
if (interactive()) {
  rb_backup(path = "folder_to_backup", project = "test_project")
}

## End(Not run)
```

---

rb_check_cache *Check caching status in the current session, and override it upon request*

Description

Mostly used internally in functions, exported for reference.

Usage

```
rb_check_cache(cache = NULL)
```

Arguments

cache
: Defaults to NULL. If NULL, checks current cache settings. If given, returns given value, ignoring cache.

Value

Either TRUE or FALSE, depending on current cache settings.

Examples

```
if (interactive()) {
  rb_check_cache()
}
```

`rb_check_cache_folder` *Checks if cache folder exists, if not returns an informative message*

Description

Checks if cache folder exists, if not returns an informative message

Usage

```
rb_check_cache_folder()
```

Value

If the cache folder exists, returns TRUE. Otherwise throws an error.

Examples

```
# If cache folder does not exist, it throws an error
tryCatch(rb_check_cache_folder(),
  error = function(e) {
    return(e)
  }
)

# Create cache folder
rb_set_cache_folder(path = fs::path(
  tempdir(),
  "rb_cache_folder"
))
rb_create_cache_folder(ask = FALSE)

rb_check_cache_folder()
```

`rb_create_cache_folder` *Creates the base cache folder where ‘rbackupr‘ caches data.*

Description

Creates the base cache folder where ‘rbackupr‘ caches data.

Usage

```
rb_create_cache_folder(ask = TRUE)
```

Arguments

`ask`	Logical, defaults to TRUE. If FALSE, and cache folder does not exist, it just creates it without asking (useful for non-interactive sessions).

Value

Nothing, used for its side effects.

Examples

```
if (interactive()) {
  rb_create_cache_folder()
}
```

`rb_disable_cache` *Disable caching for the current session*

Description

Disable caching for the current session

Usage

```
rb_disable_cache()
```

Value

Nothing, used for its side effects.

Examples

```
rb_disable_cache()
```

`rb_drive_auth` *Set up app and scope for the current session*

Description

Set up app and scope for the current session

Usage

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

Arguments

`client`	A Google client. See example, and dedicated gargle documentation.
`scopes`	Defaults to 'https://www.googleapis.com/auth/drive.file'. See Google api documentation for details
`email`	Optional. If specified, `email` can take several different forms: • `"jane@gmail.com"`, i.e. an actual email address. This allows the 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 targeted 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). • `"*@example.com"`, i.e. a domain-only glob pattern. This can be helpful if you need code that "just works" for both `alice@example.com` and `bob@example.com`. • `TRUE` means that you are approving email auto-discovery. If exactly one matching token is found in the cache, it will be used. • `FALSE` or `NA` mean that you want to ignore the token cache and force a new OAuth dance in the browser. Defaults to the option named `"gargle_oauth_email"`, retrieved by `gargle_oauth_email()` (unless a wrapper package implements different default behavior).
`cache`	Specifies the OAuth token cache. Defaults to the option named `"gargle_oauth_cache"`, retrieved via `gargle_oauth_cache()`.

use_oob — Whether to use out-of-band authentication (or, perhaps, a variant implemented by gargle and known as "pseudo-OOB") when first acquiring the token. Defaults to the value returned by `gargle_oob_default()`. Note that (pseudo-)OOB auth only affects the initial OAuth dance. If we retrieve (and possibly refresh) a cached token, `use_oob` has no effect.

If the OAuth client is provided implicitly by a wrapper package, its type probably defaults to the value returned by `gargle_oauth_client_type()`. You can take control of the client type by setting `options(gargle_oauth_client_type = "web")` or `options(gargle_oauth_client_type = "installed")`.

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).

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.

Value

Nothing, used for its side effects, i.e. logging in your Google account.

Examples

```
if (interactive()) {
  google_client <- httr::oauth_app(
    "my-very-own-google-app",
    key = "123456789.apps.googleusercontent.com",
    secret = "abcdefghijklmnopqrstuvwxyz"
  )
  rb_drive_auth(google_client)
}
```

---

rb_drive_create_folders

Create missing folders and get data frame with identifiers of all given folders

---

Description

Create missing folders and get data frame with identifiers of all given folders

Usage

```
rb_drive_create_folders(
  folders,
  parent_id,
  relative_path = NULL,
  project = NULL,
  update = FALSE,
```

```
  base_folder = "rbackupr",
  cache = TRUE
)
```

Arguments

folders	A character vector of folder names.
parent_id	Identifier of parent folder, where the new folders are to be created.
project	Defaults to NULL. Can be set once per session with ‘rb_get_project_name()‘. If given, must be a character vector of length one: name of the project.
update	Logical, defaults to FALSE. If TRUE, checks on Google Drive for newly updated files or folders, otherwise it assumes that only files and folders listed in cache exist online.

Examples

```
if (interactive()) {
  rb_drive_create_folders(
    folders = c("folder_a", "folder_b"),
    parent_id = rb_get_project()
  )
}
```

---

rb_drive_find_base_folder

Create or get base folder

Description

Create or get base folder

Usage

```
rb_drive_find_base_folder(
  base_folder = "rbackupr",
  create = FALSE,
  cache = TRUE
)
```

Arguments

base_folder	Name of base folder, defaults to "rbackupr".
create	Defaults to FALSE. If set to TRUE, the folder is create if not found.
cache	Logical, defaults to TRUE. Can be se to NULL, and managed with ‘rb_enable_cache()‘ / ‘rb_disable_cache()‘

Examples

```
## Not run:
if (interactive()) {
  rb_drive_find_base_folder()
}

## End(Not run)
```

`rb_drive_find_project` *Create rbackupr project*

Description

Create rbackupr project

Usage

```
rb_drive_find_project(
  project = NULL,
  base_folder = "rbackupr",
  create = TRUE,
  cache = TRUE
)

rb_drive_create_project(
  project = NULL,
  base_folder = "rbackupr",
  create = TRUE,
  cache = TRUE
)

rb_get_project(
  project = NULL,
  base_folder = "rbackupr",
  create = TRUE,
  cache = TRUE
)
```

Arguments

`project`	Defaults to NULL. Can be set once per session with 'rb_get_project_name()'. If given, must be a character vector of length one: name of the project.
`create`	Logical, defaults to TRUE. Creates folder if not existing.

Value

A dribble corresponding to the project folder.

Examples

```
## Not run:
if (interactive()) {
  rb_drive_find_project(project = "example")
}

## End(Not run)

## Not run:
if (interactive()) {
rb_drive_create_project(project = "example")
}

## End(Not run)
rb_get_project(project = "example")
```

`rb_enable_cache` *Enable caching for the current session*

Description

Enable caching for the current session

Usage

```
rb_enable_cache()
```

Value

Nothing, used for its side effects.

Examples

```
rb_enable_cache()
```

`rb_get_cache_file` *Gets location of cache file*

Description

Gets location of cache file

Usage

```
rb_get_cache_file(
  filename = "rbackupr_cache.sqlite",
  cache_folder = rbackupr::rb_get_cache_folder()
)
```

Arguments

filename	Defaults to "rbackupr_cache.sqlite".
cache_folder	Defaults to folder set with ‘rb_set_cache_folder()‘

Value

A character vector of length one with location of item cache file.

Examples

```
rb_set_cache_folder(path = tempdir())
sqlite_cache_file_location <- rb_get_cache_file() # outputs location of cache file
sqlite_cache_file_location
```

---

rb_get_cache_table_name

Gets name of table inside the database

Description

Gets name of table inside the database

Usage

```
rb_get_cache_table_name(
  type = "project",
  project = rbackupr::rb_get_project_name()
)
```

Arguments

type	Defaults to "project". Type of cache file to output. Values typically used by ‘rbackupr‘ include "base_folder", "projects", and "project".
project	Defaults to project name set with ‘rbackupr::rb_get_project_name()‘. Ignored if the parameter type is not set to "project"

Value

A character vector of length one with the name of the relevant table in the cache file.

Examples

```
# outputs name of table used in the cache database
rb_get_cache_table_name(type = "project", language = "testing_project")
```

rb_get_files *Check if new files appeared inside an online folder*

Description

Check if new files appeared inside an online folder

Usage

```
rb_get_files(dribble_id, update = FALSE, project = NULL, cache = TRUE)
```

Arguments

dribble_id	The dribble identifier of a folder on Google Drive.
update	Logical, defaults to FALSE. If TRUE, checks on Google Drive for newly updated folders.
project	Defaults to NULL. Can be set once per session with 'rb_get_project_name()'. If given, must be a character vector of length one: name of the project.
cache	Logical, defaults to TRUE. Stores locally cached information about base and project folder.

Value

A data frame with three columns: name (of the folder), id (of the dribble of the folder), parent_id (dribble id of the parent folder)

Examples

```
if (interactive()) {
  rb_get_files(rb_drive_find_project())
}
```

rb_get_folders *Gets cached folder with given parent, or update them from Google Drive upon request*

Description

Gets cached folder with given parent, or update them from Google Drive upon request

Usage

```
rb_get_folders(
  dribble_id,
  update = FALSE,
  project = NULL,
  base_folder = "rbackupr",
  cache = TRUE
)
```

Arguments

dribble_id	The dribble identifier of a folder on Google Drive.
update	Logical, defaults to FALSE. If TRUE, checks on Google Drive for newly updated folders.
project	Defaults to NULL. Can be set once per session with 'rb_get_project_name()'. If given, must be a character vector of length one: name of the project.
cache	Logical, defaults to TRUE. Stores locally cached information about base and project folder.

Value

A data frame with three columns: name (of the folder), id (of the dribble of the folder), parent_id (dribble id of the parent folder)

Examples

```
if (interactive()) {
  rb_get_folders(rb_drive_find_project())
}
```

---

rb_get_folder_name *Gets folder name based on id*

Description

Gets folder name based on id

Usage

```
rb_get_folder_name(dribble_id, project = NULL, base_folder = "rbackupr")
```

Arguments

dribble_id	The dribble identifier of a folder on Google Drive.
project	Defaults to NULL. Can be set once per session with 'rb_get_project_name()'. If given, must be a character vector of length one: name of the project.
base_folder	Name of base folder, defaults to 'rbackupr'
cache	Logical, defaults to TRUE. Stores locally cached information about base and project folder.

Value

A data frame with three columns: name (of the folder), id (of the dribble of the folder), parent_id (dribble id of the parent folder)

Examples

```
if (interactive()) {
  rb_get_folder_name(rb_drive_find_project())
}
```

---

rb_get_parent_folder *Gets parent id of a given folder*

Description

Gets parent id of a given folder

Usage

```
rb_get_parent_folder(dribble_id, project = NULL, base_folder = "rbackupr")
```

Arguments

dribble_id	The dribble identifier of a folder on Google Drive.
project	Defaults to NULL. Can be set once per session with 'rb_get_project_name()'. If given, must be a character vector of length one: name of the project.
cache	Logical, defaults to TRUE. Stores locally cached information about base and project folder.

Value

A data frame with three columns: name (of the folder), id (of the dribble of the folder), parent_id (dribble id of the parent folder)

Examples

```
if (interactive()) {
  rb_get_parent_folder(rb_drive_find_project())
}
```

rb_get_relative_path *Get relative path in local folder from identifier*

Description

Based only on cached data-

Usage

```
rb_get_relative_path(
  dribble_id,
  project = NULL,
  max_level = 100,
  base_folder = "rbackupr"
)
```

Arguments

dribble_id	The dribble identifier of a folder on Google Drive.
project	Defaults to NULL. Can be set once per session with 'rb_get_project_name()'. If given, must be a character vector of length one: name of the project.
max_level	Defaults to 100. Maximum level of sub-folders to backup. Default means it will go 100 times deep into sub-folders. Used also to prevent infinite loops.
base_folder	Name of base folder, defaults to 'rbackupr'

rb_set_cache_folder *Set folder for caching data*

Description

Consider using a folder out of your current project directory, e.g. 'rb_set_cache_folder("~/R/rbackupr_data/")': you will be able to use the same cache in different projects, and prevent cached files from being sync-ed if you use services such as Nextcloud or Dropbox.

Usage

```
rb_set_cache_folder(path = NULL)

rb_get_cache_folder(path = NULL)
```

Arguments

path	A path to a location used for caching data. If the folder does not exist, it will be created.

Value

The path to the caching folder, if previously set; the same path as given to the function; or the default, ‘rbackupr_data‘ is none is given.

Examples

```
if (interactive()) {
  rb_set_cache_folder(fs::path(fs::path_home_r(), "R", "rbackupr_data"))
}

rb_get_cache_folder()
```

`rb_set_project` *Set (or get) name of project for the current session.*

Description

Set (or get) name of project for the current session.

Usage

```
rb_set_project(project = NULL)

rb_get_project_name(project = NULL)
```

Arguments

`project`	Defaults to NULL. If given, it must be a character vector of length one. Name of a project. It will be used as the root folder for your current project, and located under the ‘base_folder‘ on your Google Drive.

Value

The project name, if previously set; the same as input if not NULL; or the default, ‘rbackupr_data‘ is none is given.

Examples

```
rb_set_project(project = "weather_csv_files")
rb_get_project_name()
```

Index