Package ‘archivist.github’

October 12, 2022

Version 0.2.6
Type Package
Title Tools for Archiving, Managing and Sharing R Objects via GitHub
Description The extension of the ‘archivist’ package integrating the archivist with GitHub via GitHub API, ‘git2r’ packages and ‘httr’ package.
Repository CRAN
License GPL-2
LazyLoad yes
LazyData yes
Depends R (>= 3.2.0), archivist
Imports httr, git2r, jsonlite, digest
Suggests knitr

BugReports https://github.com/MarcinKosinski/archivist.github/issues
URL http://marcinkosinski.github.io/archivist.github/
RoxygenNote 6.0.1
NeedsCompilation no
VignetteBuilder knitr
Author Marcin Kosinski [aut, cre],
Przemyslaw Biecek [aut]
Maintainer Marcin Kosinski <m.p.kosinski@gmail.com>
Date/Publication 2018-08-07 14:50:03 UTC

R topics documented:

archivist.github-package ........................................ 2
addHooksToPrintGitHub .......................................... 3
archive .............................................................. 4
archivist-github-integration .................................... 6
authoriseGitHub ..................................................... 8
archivist.github-package

Tools for Archiving, Managing and Sharing R Objects via GitHub

Description

GitHub integration extension of the archivist package - tool for storing, restoring and searching for R objects. More about archivist.github can be found on marcinkosinski.github.io/archivist.github/

Details

More about archivist.github can be found on marcinkosinski.github.io/archivist.github/ and about archivist in posts’ history on https://pbiecek.github.io/archivist/articles/posts.html

Posts

This package is well explained on this http://r-bloggers.com/r-hero-saves-backup-city-with-archivist-and-github blog post.

Note

Bug reports and feature requests can be sent to https://github.com/MarcinKosinski/archivist.github/issues

Author(s)

Marcin Kosinski [aut, cre] <m.p.kosinski@gmail.com>
Przemyslaw Biecek [aut] <przemyslaw.biecek@gmail.com>

References

More about archivist.github can be found on marcinkosinski.github.io/archivist.github/ and about archivist in posts’ history on https://pbiecek.github.io/archivist/articles/posts.html

See Also

Other archivist.github: archive, authoriseGitHub, cloneGitHubRepo, createGitHubRepo, deleteGitHubRepo, pushGitHubRepo
addHooksToPrintGitHub  Add *archivist* Hooks to *rmarkdown* markdown/\LaTeX\ Reports and Archive Artifact on GitHub

**Description**

`addHooksToPrintGitHub` adds an overloaded version of the print function for objects of selected class. The overloaded function will add all objects of selected class to the Repository and then will add hooks to the HTML report (generated in `rmarkdown`) for these objects (artifacts - *archivist-package*). This is GitHub version of `addHooksToPrint` and it automatically stores artifacts on GitHub - see examples.


**Usage**

```r
addHooksToPrintGitHub(class = "ggplot", repo = aoptions("repo"),
user = aoptions("user"), password = aoptions("password"),
format = "markdown")
```

**Arguments**

- `class` A character containing a name of class (one or more) that should be archived.
- `repo` A character containing a name of a GitHub repository on which the Repository is archived. If `repo = NULL` then hooks will be added to files in local directories.
- `user` A character containing a name of a GitHub user on whose account the repo is created.
- `password` A character denoting GitHub user password. Can be set globally with `aoptions("password", password)`. See *agithub*.
- `format` The same as in `alink`.

**Note**

One can specify userTags as in `archive` for artifacts by adding "tags" attribute. See note section about that in `archive`.

Bug reports and feature requests can be sent to [https://github.com/pbiecek/archivist/issues](https://github.com/pbiecek/archivist/issues)

**Author(s)**

Przemyslaw Biecek ([addHooksToPrint](https)), <przemyslaw.biecek@gmail.com>

Marcin Kosinski ([addHooksToPrintGitHub](https)), <m.p.kosinski@gmail.com>

**References**

More about `archivist.github` can be found on [marcinkosinski.github.io/archivist.github/](https://marcinkosinski.github.io/archivist.github/) and about `archivist` in posts' history on [https://pbiecek.github.io/archivist/articles/posts.html](https://pbiecek.github.io/archivist/articles/posts.html)
See Also

Other archivist: archivist-github-integration

Examples

```r
## Not run:
# only in Rmd report, links to GitHub repository and archive artifact
# ' empty GitHub Repository creation
authoriseGitHub(ClientID, ClientSecret) -> github_token
# authoriseGitHub also does: aoptions("github_token", github_token)
aoptions("user", user.name)
aoptions("password", user.password)
createGitHubRepo("Museum", default = TRUE) # it does aoptions("repo", "Museum")

addHooksToPrintGitHub(class="ggplot") # takes default parameters from ?aoptions
qplot(mpg, wt, data = mtcars, geom = "path")
summaryRemoteRepo()
showRemoteRepo()

## End(Not run)
```

---

**archive**

*Archive Artifact to Local and GitHub Repository*

**Description**

archive stores artifacts in the local Repository and automatically pushes archived artifacts to the GitHub Repository with which the local Repository is synchronized (via `createGitHubRepo` or `cloneGitHubRepo`). Function stores artifacts on the same way as `saveToLocalRepo` function.


**Usage**

```r
archive(artifact, commitMessage = aoptions("commitMessage"),
   repo = aoptions("repo"), user = aoptions("user"),
   password = aoptions("password"), alink = aoptions("alink"),
   artifactName = deparse(substitute(artifact)), verbose = FALSE, ...)
```

**Arguments**

- **artifact**: An artifact to be archived on Local and Github Repository.
- **commitMessage**: A character denoting a message added to the commit while archiving artifact on GitHub Repository. By default, an artifact's md5hash is added to the commit message when it is specified to NULL.
- **repo**: A character denoting GitHub repository name and synchronized local existing directory in which an artifact will be saved.
user  A character denoting GitHub user name. Can be set globally with `aoptions("user", user)`. See `agithub`.

password  A character denoting GitHub user password. Can be set globally with `aoptions("password", password)`. See `agithub`.

alink  Logical. Whether the result should be put into `alink` function. If you would like to pass further arguments to `alink` then you should specify them with `aoptions` in this case.

artifactName  The name of the artifact with which it should be archived. If `NULL` then object’s MD5 hash will be used instead.

verbose  A logical value. If `TRUE` then additional messages will be printed out.

Further arguments passed to `saveToLocalRepo` function.

Details

To learn more about Archivist Integration With GitHub visit `agithub`.

Note

Bug reports and feature requests can be sent to https://github.com/MarcinKosinski/archivist.github/issues

Author(s)

Marcin Kosinski, <m.p.kosinski@gmail.com>

References

More about `archivist.github` can be found on marcinkosinski.github.io/archivist.github/ and about `archivist` in posts’ history on https://pbiecek.github.io/archivist/articles/posts.html

See Also

Other archivist.github: `archivist.github-package`, `authoriseGitHub`, `cloneGitHubRepo`, `createGitHubRepo`, `deleteGitHubRepo`, `pushGitHubRepo`

Examples

```r
## Not run:

# empty GitHub Repository creation
authoriseGitHub(ClientID, ClientSecret) -> github_token
# authoriseGitHub also does: aoptions("github_token", github_token)
aoptions("user", user.name)
aoptions("password", user.password)

createGitHubRepo("archive-test4", default = TRUE)
## artifact's archiving
exampleVec <- 1:100

# archiving
```
archive(exampleVec) -> md5hash_path

## proof that artifact is really archived
showGithubRepo() # uses options from setGithubRepo
# let's remove exampleVec
rm(exampleVec)
# and load it back from md5hash_path
aread(md5hash_path)

# clone example
unlink("archive-test", recursive = TRUE)
cloneGithubRepo(https://github.com/MarcinKosinski/archive-test)
setRemoteRepo(aoptions("user"), "archive-test")
data(iris)
archive(iris)
showRemoteRepo()

## alink() option
vectorLong <- 1:100
vectorShort <- 1:20
# archiving
alink(archive(vectorLong))
archive(vectorShort, alink = TRUE)
showRemoteRepo()

## End(Not run)

---

Archivist Integration With GitHub

**Description**

Set of functions to integrate archivist-package with GitHub API [https://developer.github.com/v3/](https://developer.github.com/v3/).

To start working with archivist.github one should run authoriseGitHub to create an OAuth token which is required by every function in this package.

It is possible to create new GitHub repository with an empty archivist-like Repository with create-GitHubRepo function.

archive stores artifacts in the Local Repository and automatically pushes archived artifacts to the GitHub Repository with which the Local Repository is synchronized.

cloneGitHubRepo clones GitHub Repository into the local directory.

deleteGitHubRepo can delete whole GitHub-Repository or only archivist-like Repository stored on a GitHub-Repository.

pushGitHubRepo and pullGitHubRepo synchronize Local and GitHub (remote) Repository.
Details

To use this set of functionalities, one firstly has to authorize himself to the GitHub API. It can be done by creating https://github.com/settings/developers (register new application). If you do not know what should be included as callback url, you may use http://localhost:1410 for testing purposes. When application is created, one will have to copy its Client ID and Client Secret and authorize his github user with this application by running those commands:

- myapp <- oauth_app("github", key = Client_ID, secret = Client_Secret).
- github_token <- oauth2.0_token(oauth_endpoints("github"), myapp, scope = "public_repo").

The scope limits can be found here https://developer.github.com/v3/oauth/#scopes. Basically, this is how you grant an access to your application and give permissions. With such a token one is authorized and can work with GitHub API and archivist functions devoted to GitHub integration.

To perform GitHub integration operations such as push, pull, commit, add etc. a user has to pass his GitHub user name (user.name parameter), user email (user.email parameter) and user password (user.password parameter). Those parameters can be set globbaly with aoptions("user.email", user.email), aoptions("user.name", user.name) and aoptions("user.password", user.password).

Note

Note that global configuration of the git config is used for initial commit. One can later specify local configuration for the repository with config, e.g config(repoName, user.name = "Alice", user.email = "mail_at_gmail.com").

Bug reports and feature requests can be sent to https://github.com/MarcinKosinski/archivist.github/issues

Author(s)

Marcin Kosinski, <m.p.kosinski@gmail.com>

References

More about archivist.github can be found on marcinkosinski.github.io/archivist.github/ and about archivist in posts’ history on https://pbiecek.github.io/archivist/articles/posts.html

See Also

Other archivist: addHooksToPrintGitHub

Examples

```r
## Not run:
authoriseGitHub(ClientID, ClientSecret) -> github_token
# authoriseGitHub also does: aoptions("github_token", github_token)
aoptions("user", user.name)
aoptions("password", user.password)

## here github_token is used
createGitHubRepo("Museum")
createGitHubRepo("Museum-Extras", response = TRUE)
createGitHubRepo("Gallery", readme = NULL)
```
authorizeGitHub

```r
createGitHubRepo("Landfill",
  repoDescription = "My models and stuff")

## End(Not run)
```

---

**authorizeGitHub**  
*Authorise with GitHub API*

**Description**

`authorizeGitHub` is a function that performs OAuth authorisation with GitHub API and stores resulting token in the `github_token` variable. In order to authorise your app you need ClientID and ClientSecret. They can be found here: https://github.com/settings/applications/new

**Usage**

```r
authorizeGitHub(ClientID, ClientSecret, scope = c("public_repo"))
```

**Arguments**

- **ClientID**  
  A 20 characters long string with Client ID. See https://github.com/settings/applications/ for more details.

- **ClientSecret**  
  A 40 characters long string with Client Secret. See https://github.com/settings/applications/ for more details.

- **scope**  
  A character vector with the list of availables scopes for the GitHub API token.  
  See https://developer.github.com/v3/oauth/#scopes. For repository deletion you will need to add "delete_repo" scope.

**Note**

Bug reports and feature requests can be sent to https://github.com/MarcinKosinski/archivist.github/issues

**Author(s)**

Przemyslaw Biecek, <przemyslaw.biecek@gmail.com>

**References**

More about `archivist.github` can be found on marcinkosinski.github.io/archivist.github/ and about `archivist` in posts’ history on https://pbiecek.github.io/archivist/articles/posts.html

**See Also**

Other archivist.github: `archive, archivist.github-package, cloneGitHubRepo, createGitHubRepo, deleteGitHubRepo, pushGitHubRepo`
**Examples**

```r
## Not run:
## GitHub version
authoriseGitHub(ClientID, ClientSecret) -> github_token

## End(Not run)
```

---

**cloneGitHubRepo**  
*Clone Github Repository*

**Description**

`cloneGitHubRepo` is a wrapper around `git clone` and clones GitHub Repository into the `repoDir` directory.

**Usage**

```r
closeGitHubRepo(repoURL, repoDir = NULL, default = FALSE, ...)
```

**Arguments**

- `repoURL`  
  The remote repository to clone.
- `repoDir`  
  Local directory to clone to. If NULL, by default, creates a local directory, which corresponds to the name after last `/` in `repoURL`.
- `default`  
  Sets cloned Repository as default Local and GitHub Repository. If `default = TRUE` then `repoDir` (last piece of `repoURL`) is set as default Local Repository and for GitHub repository also the user from `repoURL` is set as default GitHub user.
- `...`  
  Further parameters passed to `clone`.

**Details**

To learn more about Archivist Integration With GitHub visit [agithub](https://agithub).

**Note**

Bug reports and feature requests can be sent to [https://github.com/MarcinKosinski/archivist.github/issues](https://github.com/MarcinKosinski/archivist.github/issues).

**Author(s)**

Marcin Kosinski, <m.p.kosinski@gmail.com>

**References**

See Also

Other archivist.github: `archive`, `archivist.github-package`, `authoriseGitHub`, `createGitHubRepo`, `deleteGitHubRepo`, `pushGitHubRepo`

Examples

```r
## Not run:
cloneGitHubRepo("https://github.com/MarcinKosinski/Museum")
clone GitHubRepo("https://github.com/MarcinKosinski/Museum-Extra")

# empty Github Repository creation
authoriseGitHub(ClientID, ClientSecret) -> github_token
# authoriseGitHub also does: aoptions("github_token", github_token)
aoptions("user", user.name)
aoptions("password", user.password)

createEmptyGitHubRepo("archive-test4")
setRemoteRepo(aoptions("name"), "archive-test4")

## artifact's archiving
example <- 1:100

# archiving
archive(example) -> md5hash_path

## proof that artifact is really archived
showRemoteRepo() # uses options from setGithubRepo
# let's remove przyklad
rm(example)
# and load it back from md5hash_path
aread(md5hash_path)

## clone example
unlink("archive-test", recursive = TRUE)
cloneGitHubRepo('https://github.com/MarcinKosinski/archive-test')
setRemoteRepo(aoptions("name"), "archive-test")
# equivalent is cloneGitHubRepo('https://github.com/MarcinKosinski/archive-test', default = TRUE)
# check if default is set with
# aoptions('repoDir'); aoptions('repo'); aoptions('user')
data(iris)
archive(iris)
showRemoteRepo()

## End(Not run)
```
createGitHubRepo  

Create an Empty Repository on GitHub

Description

createGitHubRepo is a GitHub version of createLocalRepo and creates a new GitHub repository with an empty archivist-like Repository. It also creates a Local Repository which is gitsynchronized with new GitHub repository.

This function is well explained on this http://r-bloggers.com/r-hero-saves-backup-city-with-archivist-and-github blog post.

Usage

createGitHubRepo(repo, github_token = aoptions("github_token"),
user = aoptions("user"), repoDir = NULL,
password = aoptions("password"),
repoDescription = aoptions("repoDescription"),
readmeDescription = aoptions("readmeDescription"),
response = aoptions("response"), default = FALSE, verbose = FALSE, ...)

Arguments

repo  
While working with a Github repository. A character denoting new GitHub repository name. White spaces will be substiuted with a dash.

github_token  
While working with a Github repository. An OAuth GitHub Token created with the oauth2.0_token function. See archivist-github-integration. Can be set globally with aoptions("github_token", github_token).

user  
While working with a Github repository. A character denoting GitHub user name. Can be set globally with aoptions("user", user). See archivist-github-integration.

repoDir  
A character that specifies the directory for the Repository which is to be made. While working with GitHub Repository, this will be the directory of the synchronized Local Repository, in which the new Local Repository will be created (is NULL then is the same as repo).

password  
While working with a Github repository. A character denoting GitHub user password. Can be set globally with aoptions("password", password). See archivist-github-integration.

repoDescription  
While working with a Github repository. A character specifying the new GitHub repository description.

readmeDescription  
While working with a Github repository. A character of the content of README.md file. By default a description of Repository. Can be set globally with aoptions("readmeDescription", readmeDescription). In order to omit README.md file set aoptions("readmeDescription", NULL).
createGitHubRepo

response A logical value. Should the GitHub API response be returned.
default If default = TRUE then repoDir (repo) is set as default local repository. Also the user is set as default GitHub user.
verbose A logical value. If TRUE then additional messages will be printed out.
... further arguments passed to createLocalRepo such as force.

Details

To learn more about Archivist Integration With GitHub visit agithub.

At least one Repository must be initialized before using other functions from the archivist.github package. While working in groups, it is highly recommended to create a Repository on a shared Dropbox/GitHub folder.

All artifacts which are desired to be archived are going to be saved in the local Repository, which is an SQLite database stored in a file named backpack. After calling saveToLocalRepo function, each artifact will be archived in a md5hash.rda file. This file will be saved in a folder (under repoDir directory) named gallery. For every artifact, md5hash is a unique string of length 32 that is produced by digest function, which uses a cryptographical MD5 hash algorithm.

To learn more about artifacts visit archivist-package.

Created backpack database is a useful and fundamental tool for remembering artifact’s name, class, archiving date etc. (the so called Tags) or for keeping artifact’s md5hash.

Besides the backpack database, gallery folder is created in which all artifacts will be archived.

After every saveToLocalRepo call the database is refreshed. As a result, the artifact is available immediately in backpack.db database for other collaborators.

Note

Bug reports and feature requests can be sent to https://github.com/MarcinKosinski/archivist.github/issues

Author(s)

Marcin Kosinski, <m.p.kosinski@gmail.com>

References

More about archivist.github can be found on marcinkosinski.github.io/archivist.github/ and about archivist in posts’ history on https://pbiecek.github.io/archivist/articles/posts.html

See Also

Other archivist.github: archive, archivist.github-package, authoriseGitHub, cloneGitHubRepo, deleteGitHubRepo, pushGitHubRepo
deleteGitHubRepo

Delete the Existing GitHub Repository

Description

deleteGitHubRepo can delete whole GitHub-Repository or only archivist-like Repository stored on a GitHub-Repository (then it requires more parameters to be specified).

This function is well explained on this http://r-bloggers.com/r-hero-saves-backup-city-with-archivist-and-github blog post.

Examples

```r
## Not run:
# empty GitHub Repository creation
authoriseGitHub(ClientID, ClientSecret) -> github_token
# authoriseGitHub also does: aoptions("github_token", github_token)
aoptions("user", user.name)
aoptions("password", user.password)

createGitHubRepo("Museum")
createGitHubRepo("Museum-Extras", response = TRUE)
createGitHubRepo("Gallery", readmeDescription = NULL)
createGitHubRepo("Landfill", repoDescription = "My models and stuff")
createGitHubRepo("MuseumYYYY", repoDir = "Museum_YY")
createGitHubRepo("archive-test4", default = TRUE)

## artifact's archiving
przyklad <- 1:100

# archiving
archive(przyklad) -> md5hash_path

## proof that artifact is really archived
showRemoteRepo() # uses options from setGithubRepo
# let's remove przyklad
rm(przyklad)
# and load it back from md5hash_path
read(md5hash_path)

# clone example
unlink("archive-test", recursive = TRUE)
cloneGithubRepo(https://github.com/MarcinKosinski/archive-test)
setRemoteRepo(aoptions("user"), "archive-test")
data(iris)
archive(iris)
showRemoteRepo()
```

## End(Not run)
deleteGitHubRepo

Usage

```r
deleteGitHubRepo(repo, github_token = aoptions("github_token"),
user = aoptions("user"), password = aoptions("password"), unset = FALSE,
deleteRoot = FALSE, subdir = NULL, response = aoptions("response"))
```

Arguments

- **repo**: While working with a Github repository. A character denoting GitHub repository name to be deleted.
- **github_token**: While working with a Github repository. An OAuth GitHub Token created with the `oauth2.0_token` function. To delete GitHub Repository you need to have `delete_repo` scope set - see examples. See `archivist-github-integration`. Can be set globally with `aoptions("github_token", github_token)`.
- **user**: While working with a Github repository. A character denoting GitHub user name. Can be set globally with `aoptions("user", user)`. See `archivist-github-integration`.
- **password**: Only when `deleteRoot = FALSE`. While working with a Github repository. A character denoting GitHub user password. Can be set globally with `aoptions("password", password)`. See `archivist-github-integration`.
- **unset**: A logical. If deleted `repoDir/repo` was set to be default Local/GitHub Repository and unset is TRUE, then `repoDir/repo` is unset as a default Local/GitHub Repository (`aoptions('repoDir/repo', NULL, T)`).
- **deleteRoot**: A logical value that specifies if the repository root directory should be deleted for Local Repository or for GitHub whether to delete whole GitHub-Repository.
- **subdir**: Only when `deleteRoot = FALSE`. Subdirectory in which the archivist-Repository is stored on a GitHub Repository. If it’s in main directory, then specify to NULL (default).
- **response**: A logical value. Should the GitHub API response be returned (only when `deleteRoot = TRUE`).

Details

To learn more about Archivist Integration With GitHub visit `agithub`. To delete GitHub Repository you need to have `delete_repo` scope set - see examples.

Note

Bug reports and feature requests can be sent to [https://github.com/MarcinKosinski/archivist.github/issues](https://github.com/MarcinKosinski/archivist.github/issues)

Author(s)

Marcin Kosinski, &lt;m.p.kosinski@gmail.com&gt;

References

More about `archivist.github` can be found on [marcinkosinski.github.io/archivist.github/](https://marcinkosinski.github.io/archivist.github/) and about `archivist` in posts’ history on [https://pbiecek.github.io/archivist/articles/posts.html](https://pbiecek.github.io/archivist/articles/posts.html)
**pushGitHubRepo**

**Push and Pull for Repository**

**Description**

pushGitHubRepo adds files, commits them and pushes from Local Repository to synchronized GitHub one. pullGitHubRepo pulls (git pull) changes from remote GitHub Repository to the corresponding Local one.


**Usage**

```r
pushGitHubRepo(repoDir = aoptions("repoDir"),
               commitMessage = aoptions("commitMessage"),
               repo = aoptions("repo"),
               user = aoptions("user"),
               password = aoptions("password"),
               files = c("gallery", "backpack.db"), ...)

pullGitHubRepo(repoDir = aoptions("repoDir"),
               user = aoptions("user"),
               password = aoptions("password"), ...)
```

**Arguments**

- **repoDir** A character specifying the directory to Local Repository to/from which artifacts will be pulled/pushed to GitHub.

**See Also**

Other archivist.github: *archive, archivist.github-package, authoriseGitHub, cloneGitHubRepo, createGitHubRepo, pushGitHubRepo*

**Examples**

```r
## Not run:

authoriseGitHub(ClientID, ClientSecret,
                 scope = c("public_repo", "delete_repo"): -> github_token
# authoriseGitHub also does: aoptions("github_token", github_token)
aoptions("user", user.name)
aoptions("password", user.password)

createGitHubRepo("Museum")
deleteGitHubRepo(repo = "Museum", deleteRoot = TRUE, response = TRUE)

## End(Not run)
```
commitMessage A character denoting a message added to the commit while performing push. By default specified to NULL which corresponds to commit message archivist::pushGitHubRepo.

repo A character denoting GitHub repository name and synchronized local existing directory in which an artifact will be saved.

user A character denoting GitHub user name. Can be set globally with aoptions("user", user). See archivist-github-integration.

password A character denoting GitHub user password. Can be set globally with aoptions("password", password). See archivist-github-integration.

files A character vector containing directories to files that should be committed and pushed. The working directory is repoDir. By default all uncommitted artifacts and backpack.db will be pushed.

... Further arguments passed to push or pull.

Details
To learn more about Archivist Integration With GitHub visit agithub. To check the status (git status) of the Repository use git2r::status(repository(repoDir)). See examples.

Note
Bug reports and feature requests can be sent to https://github.com/MarcinKosinski/archivist.github/issues

Author(s)
Marcin Kosinski, <m.p.kosinski@gmail.com>

References
More about archivist.github can be found on marcinkosinski.github.io/archivist.github/ and about archivist in posts’ history on https://pbiecek.github.io/archivist/articles/posts.html

See Also
Other archivist.github: archive, archivist.github-package, authoriseGitHub, cloneGitHubRepo, createGitHubRepo, deleteGitHubRepo

Examples
## Not run:

authoriseGitHub(ClientID, ClientSecret) -> github_token
# authoriseGitHub also does: aoptions("github_token", github_token)
aoptions("user", user.name)
aoptions("password", user.password)

createGitHubRepo("Museum", default = TRUE) # here github_token is used
data(iris)
saveToLocalRepo(iris)
pushGitHubRepo

```r
  git2r::status(repository('Museum'))
pushGitHubRepo(commitMessage = "add iris")
git2r::status(repository('Museum'))
```

## End(Not run)
Index

addHooksToPrint, 3
addHooksToPrintGitHub, 3, 3, 7
agithub, 3, 5, 9, 12, 14, 16
agithub (archivist-github-integration), 6
alink, 3, 5
aoptions, 5
archive, 2, 3, 4, 6, 8, 10, 12, 15, 16
archivist-github-integration, 6, 11, 14, 16
archivist-package, 3, 6, 12
archivist.github-package, 2
authoriseGitHub, 2, 5, 6, 8, 10, 12, 15, 16
cClone, 9
cCloneGitHubRepo, 2, 4–6, 8, 9, 12, 15, 16
cConfig, 7
createGitHubRepo, 2, 4–6, 8, 10, 11, 15, 16
cCreateLocalRepo, 11, 12
deleteGitHubRepo, 2, 5, 6, 8, 10, 12, 13, 16
digest, 12
md5hash, 4
oauth2.0_token, 11, 14

pull, 16
pullGitHubRepo, 6
pullGitHubRepo (pushGitHubRepo), 15
pullGitHubRepo (pushGitHubRepo), 15
push, 16
pushGitHubRepo, 2, 5, 6, 8, 10, 12, 15, 15
Repository, 3, 4, 6, 11, 13, 15
saveToLocalRepo, 4, 5
Tags, 12