Package 'piggyback'

July 23, 2025

Version 0.1.5

Title Managing Larger Data on a GitHub Repository

Description Because larger (> 50 MB) data files cannot easily be committed to git, a different approach is required to manage data associated with an analysis in a GitHub repository. This package provides a simple work-around by allowing larger (up to 2 GB) data files to piggyback on a repository as assets attached to individual GitHub releases. These files are not handled by git in any way, but instead are uploaded, downloaded, or edited directly by calls through the GitHub API. These data files can be versioned manually by creating different releases. This approach works equally well with public or private repositories. Data can be uploaded and downloaded programmatically from scripts. No authentication is required to download data from public repositories.

URL https://github.com/ropensci/piggyback

BugReports https://github.com/ropensci/piggyback/issues

License GPL-3

Encoding UTF-8

ByteCompile true

Imports cli, glue, gh, httr, jsonlite, fs, lubridate, memoise

Suggests spelling, readr, covr, testthat, knitr, rmarkdown, gert, withr, magrittr

VignetteBuilder knitr

RoxygenNote 7.2.1

Language en-US

NeedsCompilation no

Author Carl Boettiger [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-1642-628X>), Tan Ho [aut] (ORCID: <https://orcid.org/0000-0001-8388-5155>), Mark Padgham [ctb] (ORCID: <https://orcid.org/0000-0003-2172-5265>), Jeffrey O Hanson [ctb] (ORCID: <https://orcid.org/0000-0002-4716-6134>), Kevin Kuo [ctb] (ORCID: <https://orcid.org/0000-0001-7803-7901>)

Maintainer Carl Boettiger <cboettig@gmail.com>

Repository CRAN

Date/Publication 2023-07-10 23:40:12 UTC

Contents

`piggyback-package` *piggyback: Managing Larger Data on a GitHub Repository*

Description

Because larger (> 50 MB) data files cannot easily be committed to git, a different approach is required to manage data associated with an analysis in a GitHub repository. This package provides a simple work-around by allowing larger (up to 2 GB) data files to piggyback on a repository as assets attached to individual GitHub releases. These files are not handled by git in any way, but instead are uploaded, downloaded, or edited directly by calls through the GitHub API. These data files can be versioned manually by creating different releases. This approach works equally well with public or private repositories. Data can be uploaded and downloaded programmatically from scripts. No authentication is required to download data from public repositories.

Author(s)

Maintainer: Carl Boettiger <cboettig@gmail.com> (ORCID) [copyright holder]

Authors:

- Tan Ho (ORCID)

Other contributors:

- Mark Padgham (ORCID) [contributor]
- Jeffrey O Hanson (ORCID) [contributor]
- Kevin Kuo (ORCID) [contributor]

See Also

Useful links:

- https://github.com/ropensci/piggyback
- Report bugs at https://github.com/ropensci/piggyback/issues

---

pb_delete *Delete an asset attached to a release*

Description

Delete an asset attached to a release

Usage

```
pb_delete(
  file = NULL,
  repo = guess_repo(),
  tag = "latest",
  .token = gh::gh_token()
)
```

Arguments

file	file(s) to be deleted from the release. If NULL (default when argument is omitted), function will delete all attachments to the release. delete
repo	Repository name in format "owner/repo". Defaults to guess_repo().
tag	tag for the GitHub release to which this data should be attached.
.token	GitHub authentication token, see [gh::gh_token()]

Value

TRUE (invisibly) if a file is found and deleted. Otherwise, returns NULL (invisibly) if no file matching the name was found.

Examples

```
## Not run:
readr::write_tsv(mtcars, "mtcars.tsv.gz")
## Upload
pb_upload("mtcars.tsv.gz",
          repo = "cboettig/piggyback-tests",
          overwrite = TRUE)
pb_delete("mtcars.tsv.gz",
          repo = "cboettig/piggyback-tests",
          tag = "v0.0.1")

## End(Not run)
```

pb_download *Download data from an existing release*

Description

Download data from an existing release

Usage

```
pb_download(
  file = NULL,
  dest = ".",
  repo = guess_repo(),
  tag = "latest",
  overwrite = TRUE,
  ignore = "manifest.json",
  use_timestamps = TRUE,
  show_progress = getOption("piggyback.verbose", default = interactive()),
  .token = gh::gh_token()
)
```

Arguments

file	name or vector of names of files to be downloaded. If NULL, all assets attached to the release will be downloaded.
dest	name of vector of names of where file should be downloaded. Can be a directory or a list of filenames the same length as file vector. Any directories in the path provided must already exist.
repo	Repository name in format "owner/repo". Defaults to guess_repo().
tag	tag for the GitHub release to which this data should be attached.
overwrite	Should any local files of the same name be overwritten? default TRUE.
ignore	a list of files to ignore (if downloading "all" because file=NULL).
use_timestamps	DEPRECATED.
show_progress	logical, show a progress bar be shown for uploading? Defaults to [interactive()] - can also set globally with options("piggyback.verbose")
.token	GitHub authentication token, see [gh::gh_token()]

Examples

```
## Not run:
 ## Download a specific file.
 ## (dest can be omitted when run inside and R project)
 piggyback::pb_download("iris.tsv.gz",
                        repo = "cboettig/piggyback-tests",
                        dest = tempdir())
```

```
## End(Not run)
## Not run:
 ## Download all files
 piggyback::pb_download(repo = "cboettig/piggyback-tests",
                        dest = tempdir())


## End(Not run)
```

---

pb_download_url *Get the download url of a given file*

Description

Returns the URL download for a public file. This can be useful when writing scripts that may want to download the file directly without introducing any dependency on piggyback or authentication steps.

Usage

```
pb_download_url(
  file = NULL,
  repo = guess_repo(),
  tag = "latest",
  .token = gh::gh_token()
)
```

Arguments

file	name or vector of names of files to be downloaded. If NULL, all assets attached to the release will be downloaded.
repo	Repository name in format "owner/repo". Defaults to guess_repo().
tag	tag for the GitHub release to which this data should be attached.
.token	GitHub authentication token, see [gh::gh_token()]

Value

the URL to download a file

Examples

```
## Not run:

pb_download_url("iris.tsv.xz",
                repo = "cboettig/piggyback-tests",
                tag = "v0.0.1")
```

```
## End(Not run)
```

---

pb_list *List all assets attached to a release*

Description

List all assets attached to a release

Usage

```
pb_list(repo = guess_repo(), tag = NULL, .token = gh::gh_token())
```

Arguments

repo	Repository name in format "owner/repo". Defaults to guess_repo().
tag	which release tag(s) do we want information for? If NULL (default), will return a table for all available release tags.
.token	GitHub authentication token, see [gh::gh_token()]

Value

a data.frame of release asset names, release tag, timestamp, owner, and repo.

See Also

pb_releases for a list of all releases in repository

Examples

```
## Not run:
pb_list("cboettig/piggyback-tests")

## End(Not run)
```

pb_releases *List releases in repository*

Description

This function retrieves information about all releases attached to a given repository.

Usage

```
pb_releases(
  repo = guess_repo(),
  .token = gh::gh_token(),
  verbose = getOption("piggyback.verbose", default = TRUE)
)
```

Arguments

repo	GitHub repository specification in the form of "owner/repo", if not specified will try to guess repo based on current working directory.
.token	a GitHub API token, defaults to gh::gh_token()
verbose	defaults to TRUE, use FALSE to silence messages

Value

a dataframe of all releases available within a repository.

Examples

```
try({ # wrapped in try block to prevent CRAN errors
 pb_releases("nflverse/nflverse-data")
})
```

pb_release_create *Create a new release on GitHub repo*

Description

Create a new release on GitHub repo

Usage

```
pb_release_create(
  repo = guess_repo(),
  tag,
  commit = NULL,
  name = tag,
  body = "Data release",
  draft = FALSE,
  prerelease = FALSE,
  .token = gh::gh_token()
)
```

Arguments

repo	Repository name in format "owner/repo". Will guess the current repo if not specified.
tag	tag to create for this release
commit	Specifies the commit-ish value that determines where the Git tag is created from. Can be any branch or full commit SHA (not the short hash). Unused if the git tag already exists. Default: the repository's default branch (usually `master`).
name	The name of the release. Defaults to tag.
body	Text describing the contents of the tag. default text is "Data release".
draft	default `FALSE`. Set to `TRUE` to create a draft (unpublished) release.
prerelease	default `FALSE`. Set to `TRUE` to identify the release as a pre-release.
.token	GitHub authentication token, see `[gh::gh_token()]`

See Also

Other release_management: `pb_release_delete()`

Examples

```
## Not run:
pb_release_create("cboettig/piggyback-tests", "v0.0.5")

## End(Not run)
```

---

pb_release_delete *Delete release from GitHub repo*

Description

Delete release from GitHub repo

Usage

```
pb_release_delete(repo = guess_repo(), tag, .token = gh::gh_token())
```

Arguments

repo	Repository name in format "owner/repo". Defaults to guess_repo().
tag	tag name to delete. Must be one of those found in pb_releases()$tag_name.
.token	GitHub authentication token, see [gh::gh_token()]

See Also

Other release_management: pb_release_create()

Examples

```
## Not run:
pb_release_delete("cboettig/piggyback-tests", "v0.0.5")

## End(Not run)
```

---

pb_upload *Upload data to an existing release*

Description

NOTE: you must first create a release if one does not already exists.

Usage

```
pb_upload(
  file,
  repo = guess_repo(),
  tag = "latest",
  name = NULL,
  overwrite = "use_timestamps",
  use_timestamps = NULL,
  show_progress = getOption("piggyback.verbose", default = interactive()),
  .token = gh::gh_token(),
  dir = NULL
)
```

Arguments

file	path to file to be uploaded
repo	Repository name in format "owner/repo". Defaults to guess_repo().
tag	tag for the GitHub release to which this data should be attached.
name	name for uploaded file. If not provided will use the basename of file (i.e. filename without directory)
overwrite	overwrite any existing file with the same name already attached to the on release? Default behavior is based on timestamps, only overwriting those files which are older.
use_timestamps	DEPRECATED.
show_progress	logical, show a progress bar be shown for uploading? Defaults to [interactive()] - can also set globally with options("piggyback.verbose")
.token	GitHub authentication token, see [gh::gh_token()]
dir	directory relative to which file names should be based, defaults to NULL for current working directory.

Examples

```
## Not run:
# Needs your real token to run

readr::write_tsv(mtcars,"mtcars.tsv.xz")
pb_upload("mtcars.tsv.xz", "cboettig/piggyback-tests")

## End(Not run)
```

Index