Package 'GCPtools'

November 12, 2026

Title Tools for working with gcloud and gsutil

Version 1.2.1

Date 2025-09-25

Description Lower-level functionality to interface with Google Cloud Platform tools. 'gcloud' and 'gsutil' are both supported. The functionality provided centers around utilities for the AnVIL platform.

biocViews Software, Infrastructure, ThirdPartyClient, DataImport

SystemRequirements gsutil, gcloud

Depends R (>= 4.6.0)

Imports AnVILBase, BiocBaseUtils, dplyr, httr, rlang, tibble, tidyr, utils

Suggests BiocStyle, knitr, rmarkdown, testthat (>= 3.0.0)

License Artistic-2.0

URL https://github.com/Bioconductor/GCPtools

BugReports https://github.com/Bioconductor/GCPtools/issues

Encoding UTF-8

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

git_url https://git.bioconductor.org/packages/GCPtools

git_branch RELEASE_3_23

git_last_commit 0910a9e

git_last_commit_date 2026-05-26

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Marcel Ramos [aut, cre] (ORCID: <https://orcid.org/0000-0002-3242-0582>), Nitesh Turaga [aut], Martin Morgan [aut] (ORCID: <https://orcid.org/0000-0002-5874-8148>)

Maintainer Marcel Ramos <marcel.ramos@sph.cuny.edu>

Contents

gcloud 2
gcloud_access_token 3
gsutil 4

Index **8**

`gcloud` *gcloud command line utility interface*

Description

These functions invoke the `gcloud` command line utility. See gsutil for details on how `gcloud` is located.

`gcloud_exists()` tests whether the `gcloud()` command can be found on this system. After finding the binary location, it runs `gcloud version` to identify potentially misconfigured installations. See 'Details' section of `gsutil` for where the application is searched.

`gcloud_account()`: report the current gcloud account via `gcloud config get-value account`.

`gcloud_project()`: report the current gcloud project via `gcloud config get-value project`.

`gcloud_help()`: queries `gcloud` for help for a command or sub-comand via `gcloud help ...`.

`gcloud_cmd()` allows arbitrary `gcloud` command execution via `gcloud ...`. Use pre-defined functions in preference to this.

`gcloud_storage()` allows arbitrary `gcloud storage` command execution via `gcloud storage ...`. Typically used for bucket management commands such as `rm` and `cp`.

`gcloud_storage_buckets()` provides an interface to the `gcloud storage buckets` command. This command can be used to create a new bucket via `gcloud storage buckets create ...`.

Usage

```
gcloud_exists()

gcloud_account(account = NULL)

gcloud_project(project = NULL)

gcloud_help(...)

gcloud_cmd(cmd, ...)

gcloud_storage(cmd, ...)

gcloud_storage_buckets(bucket_cmd = "create", bucket, ...)
```

Arguments

account	character(1) Google account (e.g., user@gmail.com) to use for authentication.
project	character(1) billing project name.
...	Additional arguments appended to gcloud commands.
cmd	character(1) representing a command used to evaluate gcloud cmd ....
bucket_cmd	character(1) representing a buckets command typically used to create a new bucket. It can also be used to add-iam-policy-binding or remove-iam-policy-binding to a bucket.
bucket	character(1) representing a unique bucket name to be created or modified.

Value

gcloud_exists() returns TRUE when the gcloud application can be found, FALSE otherwise.

gcloud_account() returns a character(1) vector containing the active gcloud account, typically a gmail email address.

gcloud_project() returns a character(1) vector containing the active gcloud project.

gcloud_help() returns an unquoted character() vector representing the text of the help manual page returned by gcloud help ....

gcloud_cmd() returns a character() vector representing the text of the output of gcloud cmd ...

Examples

```
gcloud_exists()

gcloud_account()
```

---

gcloud_access_token *Obtain an access token for a service account*

Description

gcloud_access_token() generates a token for the given service account. The token is cached for the duration of its validity. The token is refreshed when it expires. The token is obtained using the gcloud command line utility for the given gcloud_account(). The function is mainly used internally by API service functions, e.g., AnVIL::Terra()

Usage

```
gcloud_access_token(service)
```

Arguments

service	character(1) The name of the service, e.g. "terra" for which to obtain an access token for.

Value

`gcloud_access_token()` returns a simple token string to be used with the given service.

Examples

```
gcloud_access_token("rawls") |> invisible()
```

`gsutil` *gsutil command line utility interface*

Description

These functions invoke the `gsutil` command line utility. See the "Details:" section if you have gsutil installed but the package cannot find it.

`gsutil_is_uri()`: check if the source is a valid Google Storage URI.

`gsutil_sh_quote()`: quote a character vector for use in a shell command. This is useful to ensure that file names with spaces or other special characters are handled correctly.

`gsutil_requesterpays()`: does the google bucket require that the requester pay for access?

`gsutil_requesterpays_flag()`: return the `-u <project>` flag for the `gsutil` command if the source requires that the requester pays for access.

`gsutil_exists()`: check if the bucket or object exists.

`gsutil_stat()`: print, as a side effect, the status of a bucket, directory, or file.

`gsutil_rsync()`: synchronize a source and a destination. If the destination is on the local file system, it must be a directory or not yet exist (in which case a directory will be created).

`gsutil_cat()`: concatenate bucket objects to standard output

`gsutil_help()`: print 'man' page for the `gsutil` command or subcommand. Note that only commandes documented on this R help page are supported.

`gsutil_pipe()`: create a pipe to read from or write to a Google bucket object.

`gsutil_ls()`: List contents of a google cloud bucket or, if `source` is missing, all Cloud Storage buckets under your default project ID

`gsutil_cp()`: copy contents of `source` to `destination`. At least one of `source` or `destination` must be Google cloud bucket; `source` can be a character vector with length greater than 1. Use `gsutil_help("cp")` for `gsutil` help.

`gsutil_rm()`: remove contents of a google cloud bucket.

Usage

```
gsutil_is_uri(source)

gsutil_sh_quote(source)

gsutil_requesterpays(source)

gsutil_requesterpays_flag(source)

gsutil_exists(source)

gsutil_stat(source)

gsutil_rsync(
  source,
  destination,
  ...,
  exclude = NULL,
  dry = TRUE,
  delete = FALSE,
  recursive = FALSE,
  parallel = TRUE
)

gsutil_cat(source, ..., header = FALSE, range = integer())

gsutil_help(cmd = character(0))

gsutil_pipe(source, open = "r", ...)

gsutil_ls(source = character(), ..., recursive = FALSE)

gsutil_cp(source, destination, ..., recursive = FALSE, parallel = TRUE)

gsutil_rm(source, ..., force = FALSE, recursive = FALSE, parallel = TRUE)
```

Arguments

source	character(1), (character() for gsutil_requesterpays(), gsutil_ls(), gsutil_exists(), gsutil_cp()): paths to a Google Storage Bucket, possibly with wild-cards for file-level pattern matching.
destination	character(1) google cloud bucket or local file system destination path.
...	additional arguments passed as-is to the gsutil subcommand.
exclude	character(1) a python regular expression of bucket paths to exclude from synchronization. E.g., '.*(\\.png|\\.txt)$" excludes '.png' and .txt' files.
dry	logical(1), when TRUE (default), return the consequences of the operation without actually performing the operation.

delete	logical(1), when TRUE, remove files in destination that are not in source. Exercise caution when you use this option: it's possible to delete large amounts of data accidentally if, for example, you erroneously reverse source and destination.
recursive	logical(1); perform operation recursively from source?. Default: FALSE.
parallel	logical(1), perform parallel multi-threaded / multi-processing (default is TRUE).
header	logical(1) when TRUE annotate each
range	(optional) integer(2) vector used to form a range from-to of bytes to concatenate. NA values signify concatenation from the start (first position) or to the end (second position) of the file.
cmd	character() (optional) command name, e.g., "ls" for help.
open	character(1) either "r" (read) or "w" (write) from the bucket.
force	logical(1): continue silently despite errors when removing multiple objects. Default: FALSE.

Details

The gsutil system command is required. The search for gsutil starts with environment variable GCLOUD_SDK_PATH providing a path to a directory containing a bin directory containingin gsutil, gcloud, etc. The path variable is searched for first as an option() and then system variable. If no option or global variable is found, Sys.which() is tried. If that fails, gsutil is searched for on defined paths. On Windows, the search tries to find Google\\Cloud SDK\\google-cloud-sdk\\bin\\gsutil.cmd in the LOCAL APP DATA, Program Files, and Program Files (x86) directories. On linux / macOS, the search continues with ~/google-cloud-sdk.
gsutil_rsync()': To make "gs://mybucket/data"match the contents of the local directory"data"‘ you could do:

```
gsutil_rsync("data", "gs://mybucket/data", delete = TRUE)
```

To make the local directory "data" the same as the contents of gs://mybucket/data:

```
gsutil_rsync("gs://mybucket/data", "data", delete = TRUE)
```

If destination is a local path and does not exist, it will be created.

Value

gsutil_requesterpays(): named logical() vector TRUE when requester-pays is enabled.

gsutil_requesterpays_flag(): character() vector of "-u <gcloud_project>" if the source requires that the requester pays for access, or NULL if otherwise.

gsutil_exists(): logical(1) TRUE if bucket or object exists.

gsutil_stat(): tibble() summarizing status of each bucket member.

gsutil_rsync(): exit status of gsutil_rsync(), invisibly.

gsutil_cat() returns the content as a character vector.

gsutil_help(): character() help text for subcommand cmd.

gsutil_pipe() an unopened R pipe(); the mode is *not* specified, and the pipe must be used in the appropriate context (e.g., a pipe created with open = "r" for input as read.csv())

`gsutil_ls()`: `character()` listing of `source` content.

`gsutil_cp()`: exit status of `gsutil_cp()`, invisibly.

`gsutil_rm()`: exit status of `gsutil_rm()`, invisibly.

Examples

```
src <-
  "gs://genomics-public-data/1000-genomes/other/sample_info/sample_info.csv"

gsutil_requesterpays(src) # FALSE -- no cost download


gsutil_exists(src)
gsutil_stat(src)


gsutil_help("ls")


df <- read.csv(gsutil_pipe(src), 5L)
class(df)
dim(df)
head(df)


  gsutil_cp(src, tempdir())
  ## gsutil_*() commands work with spaces in the source or destination
  destination <- file.path(tempdir(), "foo bar")
  gsutil_cp(src, destination)
  file.exists(destination)
```

Index

gcloud, 2
gcloud_access_token, 3
gcloud_account *(gcloud)*, 2
gcloud_cmd *(gcloud)*, 2
gcloud_exists *(gcloud)*, 2
gcloud_help *(gcloud)*, 2
gcloud_project *(gcloud)*, 2
gcloud_storage *(gcloud)*, 2
gcloud_storage_buckets *(gcloud)*, 2
gsutil, 2, 4
gsutil_cat *(gsutil)*, 4
gsutil_cp *(gsutil)*, 4
gsutil_exists *(gsutil)*, 4
gsutil_help *(gsutil)*, 4
gsutil_is_uri *(gsutil)*, 4
gsutil_ls *(gsutil)*, 4
gsutil_pipe *(gsutil)*, 4
gsutil_requesterpays *(gsutil)*, 4
gsutil_requesterpays_flag *(gsutil)*, 4
gsutil_rm *(gsutil)*, 4
gsutil_rsync *(gsutil)*, 4
gsutil_sh_quote *(gsutil)*, 4
gsutil_stat *(gsutil)*, 4