Package ‘GCSConnection’

October 17, 2020

Type Package

Title Creating R Connection with Google Cloud Storage

Version 1.0.1

Date 2019-10-24

Author Jiefei Wang

Maintainer Jiefei Wang <szwjf08@gmail.com>

Description Create R 'connection' objects to google cloud storage buckets using the Google REST interface. Both read and write connections are supported. The package also provide functions to view and manage files on Google Cloud.

License GPL (>= 2)

Depends R (>= 4.0.0)

Imports Rcpp (>= 1.0.2), httr, googleAuthR, googleCloudStorageR, methods, jsonlite, utils

Suggests testthat, knitr, rmarkdown, BiocStyle

biocViews Infrastructure

LinkingTo Rcpp

RoxygenNote 7.1.0

Encoding UTF-8

VignetteBuilder knitr

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

git_branch RELEASE_3_11

git_last_commit 8a085ed

git_last_commit_date 2020-06-02

Date/Publication 2020-10-16

R topics documented:

`FileClass-class` *File class*

Description

The properties of file class object can be accessed via ‘$‘ and ‘[[‘ operators. The symbol ‘..‘ can be used to go to the parent folder of a file class object.

`FolderClass-class` *Folder class view and access files. You can change the current directory by ‘$‘ and ‘[[‘ operators. The symbol ‘..‘ can be used to go to the parent folder of a folder object.*

Description

Folder class

view and access files. You can change the current directory by ‘$‘ and ‘[[‘ operators. The symbol ‘..‘ can be used to go to the parent folder of a folder object.

`gcs_cloud_auth` *Get/Set google credentials*

Description

Authenticate with Google Cloud Storage. You can download the JSON credential file from Google Gloud Platform. The package will search for the credentials from evironment variables ‘GOOGLE_APPLICATION_CRE
or ‘GCS_AUTH_FILE‘ when it is onloaded. To redo the credentials initialization process after the package is loaded. Simply call the ‘gcs_cloud_auth‘ function with no argument.

Usage

```
gcs_cloud_auth(json_file, gcloud = FALSE, email = NULL)

gcs_get_cloud_auth()

## S3 method for class 'auth'
print(x, ...)
```

Arguments

`json_file` character(1). A JSON file that can be used to authenticate with Google Cloud Storage. If the value is ‘NULL‘, the current credential will be erased.

`gcloud` logical. Whether use gcloud to authenticate with Google Cloud Storage. If the value is ‘TRUE‘, the parameter ‘json_file‘ will be ignored.

`email` Character(1) or NULL. For gcloud only. Account to get the access token for. If not specified, the current active account in gcloud will be used.

`x` Used for the S3 ‘print‘ function only

`...` Used for the S3 ‘print‘ function only

Details

When the package is loaded, it first searches the credential file from the enviroment variable ‘GOOGLE_APPLICATION_ If the credentials is not found, the environment variable ‘GCS_AUTH_FILE‘ will be used intead. If both variables are not specified. Users need to specify the credentials by calling ‘gcs_cloud_auth‘ function.

Value

gcs_cloud_auth : No return value

gcs_get_cloud_auth : An S3 ‘auth‘ class containing credentials information

Examples

```
## Default authentication process
gcs_cloud_auth()
gcs_get_cloud_auth()
```

gcs_connection *Connection to google cloud storage*

Description

This function creates an R connection to a file on google cloud storage. A service account credentials is required for accessing private data, the credentials can be set via ‘gcs_cloud_auth‘

Usage

```
gcs_connection(
  description,
  open = "rb",
  encoding = getOption("encoding"),
  bucket = NULL
)
```

Arguments

`description`	character(1). The name of the file that you want to connect to. It can be either the file name or a full path to the file.
`open`	character(1). A description of how to open the connection. See details for possible values. If not specified, the default value will be "rb" if a credential is set or "rbp" if not.
`encoding`	character(1). The encoding of the input/output stream of a connection. Currently the parameter ‘encoding‘ should be either ‘native.enc‘ or ‘UTF8‘. see ‘?connections‘ for more detail.
`bucket`	character(1). The name of the bucket that the file is located in. If not supplied, value in ‘gcs_get_global_bucket()‘ will be used. If a full path to the file is provided in ‘description‘, this parameter will be ignored.

Details

Possible values for the argument ‘open‘ are the combination of the following characters:

"r" or "w" : read or write mode. The GCS connection cannot be in both read and write modes.

"t" or "b" : text or binary mode. If not specified, the default is text mode.

Value

A connection

Examples

```
## Open for reading the public Landsat data
## on google cloud storage in text mode

file <- "gs://genomics-public-data/NA12878.chr20.sample.DeepVariant-0.7.2.vcf"
con <- gcs_connection(description = file, open = "rt")
readLines(con, n = 4L)
close(con)
```

`gcs_cp` *copy files to and from buckets*

Description

The function supports moving files or folders from bucket to bucket, disk to bucket and bucket to disk. Note that the existing destination file will be overwritten.

Usage

```
gcs_cp(from, to, recursive = TRUE)
```

Arguments

`from, to`	Character(1). The path to the folder/file. At least one path must be a google URI. It is recommended to explicitly add a "/" at the end of the path if the path is a folder path.
`recursive`	logical(1). Whether recursively copy the files in the subfolders.

Value

No return value

Examples

```
tmp_path <- tempdir()
## Download a file to a disk
gcs_cp("gs://genomics-public-data/NA12878.chr20.sample.bam", tmp_path)
## Check the file existance
file.exists(file.path(tmp_path, "NA12878.chr20.sample.bam"))

## Download all files in a path.
## The files in the subfolders will not be copied due to `recursive = FALSE`
folder_path <- file.path(tmp_path, "example")
gcs_cp("gs://genomics-public-data/", folder_path, recursive = FALSE)
## Check the file existance
list.files(folder_path)
```

`gcs_dir` *List bucket/folder/object*

Description

list objects in a bucket/folder or get the description of a file. You can change the current direction via '[[' or '$' operator. '..' can be used to go to the parent folder. For reducing the number of request sent to the network, it is recommended to add a trailing slash if the path is a folder.

Usage

```
gcs_dir(path, delimiter = TRUE, recursive = FALSE, depth = 2L)
```

Arguments

`path`	Character(1), the path to the bucket/folder/file.
`delimiter`	Logical(1), whether to use '/' as a path delimiter. If not, the path will be treated as the path to a file even when it ends with '/'
`recursive`	Logical(1), whether recursively query all subdirectory. If 'TRUE', all information of the subdirectories will be downloaded. The time cost can be significantly reduced if the value is 'FALSE'. The parameter only works with bucket/folder.
`depth`	Integer(1), the depth of the recursive download.

Value

A 'FolderClass' object or a 'FileClass' object

Examples

```
## List files in a bucket
## Equivalent: gcs_dir(path = "gs://genomics-public-data/")
gcs_dir(path = "genomics-public-data/")

## List files in a folder
gcs_dir(path = "genomics-public-data/clinvar/")

## List the information of a file
gcs_dir(path = "genomics-public-data/clinvar/README.txt")
```

---

gcs_set_read_buff *Get/Set read/write connection buffer size*

Description

Get/Set read/write connection buffer size, the buffer size can be set at any time but only takes effect on the connections created after the change. The default value is 1024 *1024 bytes (1 Mega bytes) for both read and write connections.

Usage

```
gcs_set_read_buff(buff_size = 1024L * 1024L)

gcs_set_write_buff(buff_size = 1024L * 1024L)

gcs_get_read_buff()

gcs_get_write_buff()
```

Arguments

buff_size	Integer. The buffer size

Value

Get functions: the current buffer size. Set functions: the previous buffer size.

Examples

```
gcs_get_read_buff()
gcs_get_write_buff()
```

names,FileClass-method *The Names of an Object*

Description

Functions to get or set the names of an object.

Usage

```
## S4 method for signature 'FileClass'
names(x)
```

Arguments

x an R object.

Details

names is a generic accessor function, and names<- is a generic replacement function. The default methods get and set the "names" attribute of a vector (including a list) or pairlist.

For an environment env, names(env) gives the names of the corresponding list, i.e., names(as.list(env,all.names = TRUE)) which are also given by ls(env,all.names = TRUE,sorted = FALSE). If the environment is used as a hash table, names(env) are its "keys".

If value is shorter than x, it is extended by character NAs to the length of x.

It is possible to update just part of the names attribute via the general rules: see the examples. This works because the expression there is evaluated as z <-"names<-"(z,"[<-"(names(z),3,"c2")).

The name "" is special: it is used to indicate that there is no name associated with an element of a (atomic or generic) vector. Subscripting by "" will match nothing (not even elements which have no name).

A name can be character NA, but such a name will never be matched and is likely to lead to confusion.

Both are primitive functions.

Value

For names, NULL or a character vector of the same length as x. (NULL is given if the object has no names, including for objects of types which cannot have names.) For an environment, the length is the number of objects in the environment but the order of the names is arbitrary.

For names<-, the updated object. (Note that the value of names(x) <-value is that of the assignment, value, not the return value from the left-hand side.)

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

slotNames, dimnames.

Examples

```
# print the names attribute of the islands data set
names(islands)

# remove the names attribute
names(islands) <- NULL
islands
rm(islands) # remove the copy made

z <- list(a = 1, b = "c", c = 1:3)
names(z)
# change just the name of the third element.
names(z)[3] <- "c2"
z

z <- 1:3
names(z)
## assign just one name
names(z)[2] <- "b"
z
```

---

names,FolderClass-method

The Names of an Object

Description

Functions to get or set the names of an object.

Usage

```
## S4 method for signature 'FolderClass'
names(x)
```

Arguments

x	an R object.

Details

names is a generic accessor function, and names<- is a generic replacement function. The default methods get and set the "names" attribute of a vector (including a list) or pairlist.

For an environment env, names(env) gives the names of the corresponding list, i.e., names(as.list(env,all.names = TRUE)) which are also given by ls(env,all.names = TRUE,sorted = FALSE). If the environment is used as a hash table, names(env) are its "keys".

If value is shorter than x, it is extended by character NAs to the length of x.

It is possible to update just part of the names attribute via the general rules: see the examples. This works because the expression there is evaluated as z <-"names<-"(z,"[<-"(names(z),3,"c2")).

The name "" is special: it is used to indicate that there is no name associated with an element of a (atomic or generic) vector. Subscripting by "" will match nothing (not even elements which have no name).

A name can be character NA, but such a name will never be matched and is likely to lead to confusion.

Both are primitive functions.

Value

For names, NULL or a character vector of the same length as x. (NULL is given if the object has no names, including for objects of types which cannot have names.) For an environment, the length is the number of objects in the environment but the order of the names is arbitrary.

For names<-, the updated object. (Note that the value of names(x) <-value is that of the assignment, value, not the return value from the left-hand side.)

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

slotNames, dimnames.

Examples

```
# print the names attribute of the islands data set
names(islands)

# remove the names attribute
names(islands) <- NULL
islands
rm(islands) # remove the copy made

z <- list(a = 1, b = "c", c = 1:3)
names(z)
# change just the name of the third element.
names(z)[3] <- "c2"
z

z <- 1:3
names(z)
## assign just one name
names(z)[2] <- "b"
z
```

show,FileClass-method *Print object of class ‘FileClass‘*

Description

Print object of class ‘FileClass‘

Usage

```
## S4 method for signature 'FileClass'
show(object)
```

Arguments

object	an object of class ‘FileClass‘

Value

Invisible ‘Object‘

show,FolderClass-method *Print object of class ‘FolderClass‘*

Description

Print object of class ‘FolderClass‘

Usage

```
## S4 method for signature 'FolderClass'
show(object)
```

Arguments

object	an object of class ‘FolderClass‘

Value

invisible NULL

$,FileClass-method *Get an element from ‘FileClass‘ object*

Description

Get an element from ‘FileClass‘ object

Usage

```
## S4 method for signature 'FileClass'
x$name

## S4 method for signature 'FileClass'
x[[i, exact = TRUE]]
```

Arguments

x	an object of class ‘FileClass‘
name, i	Character(1), the name of the element
exact	Logical(1), Controls possible partial matching of ‘[[‘ when extracting by a character(1)

Value

A ‘FolderClass‘ object or a ‘FileClass‘ object

$,FolderClass-method *Get an element from ‘FolderClass‘ object*

Description

Get an element from ‘FolderClass‘ object

Usage

```
## S4 method for signature 'FolderClass'
x$name

## S4 method for signature 'FolderClass'
x[[i, exact = TRUE]]
```

Arguments

x	an object of class ‘FolderClass‘
name, i	Character(1), the name of the element
exact	Logical(1), Controls possible partial matching of ‘[[‘ when extracting by a character(1)

Value

A ‘FolderClass‘ object or a ‘FileClass‘ object

Index