Package ‘rgee’

January 27, 2021

Title  R Bindings for Calling the 'Earth Engine' API

Version  1.0.8

Description  Earth Engine <https://earthengine.google.com/> client library for R. All of the 'Earth Engine' API classes, modules, and functions are made available. Additional functions implemented include importing (exporting) of Earth Engine spatial objects, extraction of time series, interactive map display, assets management interface, and metadata display. See <https://r-spatial.github.io/rgee/> for further details.

License  Apache License (>= 2.0)

Encoding  UTF-8

LazyData  true

RoxygenNote  7.1.1

Depends  R (>= 3.3.0)

Imports  methods, reticulate (>= 1.15), magrittr, processx, crayon, cli

Suggests  leaflet (>= 2.0.2), leaflet.extras2, magick, geojsonio, sf, stars, googledrive (>= 1.0.0), googleCloudStorageR (>= 0.5.1), rstudioapi (>= 0.7), jsonlite, leafem, raster, rgdal, httr, digest, spelling, testthat, future, covr, knitr, rmarkdown, png


BugReports  https://github.com/r-spatial/rgee/issues/

VignetteBuilder  knitr

Language  en-US

Config/reticulate  list( packages = list( list(package = "earthengine-api", pip = TRUE) ) )

NeedsCompilation  no
Author Cesar Aybar [aut, cre] (<https://orcid.org/0000-0003-2745-9535>),
Wu Qiusheng [ctb] (<https://orcid.org/0000-0001-5437-4073>),
Lesly Bautista [ctb] (<https://orcid.org/0000-0003-3523-8687>),
Roy Yali [ctb] (<https://orcid.org/0000-0003-4542-3755>),
Antony Barja [ctb] (<https://orcid.org/0000-0001-5921-2858>),
Kevin Ushey [ctb],
Jeroen Ooms [ctb] (<https://orcid.org/0000-0000-0003-0289>),
Tim Appelhans [ctb],
JJ Allaire [ctb],
Yuan Tang [ctb],
Samapriya Roy [ctb],
MariaElena Adauto [ctb] (<https://orcid.org/0000-0002-2154-2429>),
Gabriel Carrasco [ctb] (<https://orcid.org/0000-0002-6945-0419>),
Henrik Bengtsson [ctb],
Jeffrey Hollister [rev] (Hollister reviewed the package for JOSS, see
https://github.com/openjournals/joss-reviews/issues/2272/),
Gennadii Donchyts [rev] (Gena reviewed the package for JOSS, see
https://github.com/openjournals/joss-reviews/issues/2272/),
Marius Appel [rev] (Appel reviewed the package for JOSS, see
https://github.com/openjournals/joss-reviews/issues/2272/)
Maintainer Cesar Aybar <csaybar@gmail.com>
Repository CRAN
Date/Publication 2021-01-27 08:50:03 UTC

R topics documented:

```
R topics documented:

  ee ................................................................. 4
  eedate_to_rdate ................................................ 4
  ee_as_raster .................................................... 5
  ee_as_sf ......................................................... 8
  ee_as_stars ...................................................... 11
  ee_as_thumbnail ................................................ 14
  ee_check-tools ................................................... 17
  ee_clean_container ............................................. 18
  ee_clean_credentials .......................................... 19
  ee_clean_pyenv .................................................. 20
  ee_drive_to_local .............................................. 20
  ee_extract ....................................................... 22
  ee_gcs_to_local ................................................ 25
  ee_get .......................................................... 27
  ee_get_assethome ............................................... 28
  ee_get_date_ic .................................................. 29
  ee_get_date_img ................................................ 30
  ee_get_earthengine_path ....................................... 31
  ee_help .......................................................... 31
  ee_imagecollection_to_local ................................... 32
  ee_image_info .................................................... 35
```
R topics documented:

ee_image_to_asset ................................. 36
ee_image_to_drive ............................... 38
ee_image_to_gcs ................................ 41
ee_Initialize .................................... 44
ee_install ........................................ 45
ee_install_set_pyenv ............................ 46
ee_install Upgrade ............................... 47
ee_manage-tools .................................. 47
ee_monitoring ..................................... 50
ee_print ........................................... 51
ee_table_to_asset ............................... 53
ee_table_to_drive ................................. 55
ee_table_to_gcs .................................. 56
ee_users ........................................... 58
ee_user_info ...................................... 59
ee_utils_create_json ............................. 60
ee_utils_create_manifest_image ............... 60
ee_utils_create_manifest_table ................ 62
ee_utils_future_value ......................... 63
ee_utils_get_crs ................................ 64
nee_utilis gif_annotate ......................... 65
ee_utilis gif_creator ............................ 67
ee_utilis gif_save ............................... 69
ee_utilis_pyfunc ................................ 71
nee_utilis_py_to_r ............................... 72
nee_utilis_search_display ..................... 73
ee_utilis_shp_to_zip ............................ 74
nee_version ...................................... 75
gcs_to_ee_image ................................ 75
gcs_to_ee_table ................................ 76
local_to_gcs .................................... 78
Map ............................................... 79
null-default ..................................... 82
print.ee.computedobject.ComputedObject .... 83
raster_as_ee ..................................... 83
rdate_to_eedate ................................ 85
sf_as_ee ......................................... 86
stars_as_ee ..................................... 88

Index ................................. 91
ee

Main Earth Engine module

Description

Interface to main Earth Engine module. Provides access to top level classes and functions as well as sub-modules (e.g. `ee$Image`, `ee$FeatureCollection$first`, etc.).

Usage

```r
ee
```

Format

Earth Engine module

Examples

```r
## Not run:
library(rgee)

ee_Initialize()

ee_img <- ee$Image(0)
ee_ic <- ee$ImageCollection(ee_img)

print(ee_img$getInfo())
print(ee_ic$getInfo())

## End(Not run)
```

eedate_to_rdate

Pass an Earth Engine date object to R

Description

Pass an Earth Engine date object to R

Usage

```r
eedate_to_rdate(ee_date, timestamp = FALSE)
```

Arguments

- **ee_date**: ee$date object (ee$Date)
- **timestamp**: Logical. If TRUE, return the date in milliseconds from the Unix Epoch (1970-01-01 00:00:00 UTC) otherwise return the date as a POSIXct object. By default FALSE.
Details

`eedate_to_rdate` is essential to avoid potential errors that might appear when users call to retrieve dates. Currently, R integer only support 32 bit signed, such integers can only count up to about 2 billion. This range is extremely insufficient for dealing with Google Earth Engine date objects (`ee$Date`), which are represented by timestamps in milliseconds since the UNIX epoch. `eedate_to_rdate` use Python as a backend to obtain the date and convert it in float before exporting to R.

Value

`eedate_to_rdate` will return either a numeric timestamp or a POSIXct object depending on the `timestamp` argument.

See Also

Other date functions: `ee_get_date_ic()`, `ee_get_date_img()`, `rdate_to_eedate()`

Examples

```r
## Not run:
library(rgee)
eeverify()
eDate <- ee$Date$fromYMD(2010,1,1)
eedate_to_rdate(eeDate,timestamp = TRUE) # good
eDate$getInfo()$value # bad
## End(Not run)
```

### ee_as_raster

Convert an Earth Engine (EE) image in a raster object

Description

Convert an ee$Image in a raster object

Usage

```r
ee_as_raster(
  image,
  region = NULL,
  dsn = NULL,
  via = "drive",
  container = "rgee_backup",
  scale = NULL,
  maxPixels = 1e+09,
  lazy = FALSE,
  public = TRUE,
)```
Arguments

- image: ee$Image to be converted into a raster object
- region: EE Geometry (ee$Geometry$Polygon) which specify the region to export. CRS needs to be the same that the argument image, otherwise, it will be forced. If not specified, image bounds will be taken.
- dsn: Character. Output filename. If missing, a temporary file will be created.
- via: Character. Method to export the image. Two method are implemented: "drive", "gcs". See details.
- container: Character. Name of the folder ('drive') or bucket ('gcs') to be exported into.
- scale: Numeric. The resolution in meters per pixel. Defaults to the native resolution of the image.
- maxPixels: Numeric. The maximum allowed number of pixels in the exported image. The task will fail if the exported region covers more pixels in the specified projection. Defaults to 100,000,000.
- lazy: Logical. If TRUE, a future::sequential object is created to evaluate the task in the future. See details.
- public: Logical. If TRUE, a public link to the image will be created.
- add_metadata: Logical. Add metadata to the stars_proxy object. See details.
- timePrefix: Logical. Add current date and time (Sys.time()) as a prefix to files to export. This parameter helps to avoid exported files with the same name. By default TRUE.
- quiet: Logical. Suppress info message
- ...

Extra exporting argument. See ee_image_to_drive and ee_image_to_gcs.

Details

ee_as_raster supports the download of ee$Images by two different options: "drive" (Google Drive) and "gcs" (Google Cloud Storage). In both cases ee_as_stars works as follow:

1. A task will be started (i.e. ee$batch$Task$start()) to move the ee$Image from Earth Engine to the intermediate container specified in argument via.
2. If the argument lazy is TRUE, the task will not be monitored. This is useful to lunch several tasks at the same time and call them later using ee_utils_future_value or future::value. At the end of this step, the ee$Image will be stored on the path specified in the argument dsn.
3. Finally if the argument add_metadata is TRUE, a list with the following elements will be added to the stars-proxy object.
   - if via is "drive":
   - if via is "gcs":

• If via is "gcs", the list will include:
  - google_cloud_storage_container: Character. Name of the bucket on GCS.
  - google_cloud_storage_object: Character. Name of the object to store on GCS.
  - google_cloud_storage_token: Character. Token to access GCS.

• If via is "drive", the list will include:
  - drive_folder: Character. Name of the folder on Drive.
  - drive_file: Character. Name of the file to store on Drive.
  - drive_token: Character. Token to access Drive.

• If add_metadata is TRUE, the list will include:
  - metadata: List. A list of metadata to add to the stars-proxy object.

• If timePrefix is TRUE, the list will include:
  - time_prefix: Character. Current date and time (Sys.time()) as a prefix to files to export.

• If public is TRUE, the list will include:
  - public_link: Character. Public link to the image.

• If lazy is TRUE, the list will include:
  - lazy_task: future::sequential. The task is not monitored.

• If quiet is TRUE, the list will include:
  - info_message: Logical. Suppress info message.

• If maxPixels is specified, the list will include:
  - max_pixels: Numeric. The maximum allowed number of pixels in the exported image.

• If scale is specified, the list will include:
  - resolution: Numeric. The resolution in meters per pixel.

• If container is specified, the list will include:
  - container: Character. Name of the folder or bucket to export into.

• If dsn is specified, the list will include:
  - dsn: Character. Output filename.

• If region is specified, the list will include:
  - region: ee$Geometry$Polygon. The region to export.

• If image is specified, the list will include:
  - image: ee$Image. The image to convert into a raster object.

• If via is not specified, the list will include:
  - region: ee$Geometry$Polygon. The region to export.
  - dsn: Character. Output filename.
  - maxPixels: Numeric. The maximum allowed number of pixels in the exported image.
  - lazy: Logical. If TRUE, a future::sequential object is created to evaluate the task in the future.
  - public: Logical. If TRUE, a public link to the image will be created.
  - add_metadata: Logical. Add metadata to the stars_proxy object.
  - timePrefix: Logical. Add current date and time (Sys.time()) as a prefix to files to export.
  - quiet: Logical. Suppress info message.
  - maxPixels: Numeric. The maximum allowed number of pixels in the exported image.
  - lazy: Logical. If TRUE, a future::sequential object is created to evaluate the task in the future.
  - public: Logical. If TRUE, a public link to the image will be created.
  - add_metadata: Logical. Add metadata to the stars_proxy object.
  - timePrefix: Logical. Add current date and time (Sys.time()) as a prefix to files to export.
  - quiet: Logical. Suppress info message.
  - maxPixels: Numeric. The maximum allowed number of pixels in the exported image.
  - lazy: Logical. If TRUE, a future::sequential object is created to evaluate the task in the future.
  - public: Logical. If TRUE, a public link to the image will be created.
  - add_metadata: Logical. Add metadata to the stars_proxy object.
  - timePrefix: Logical. Add current date and time (Sys.time()) as a prefix to files to export.
  - quiet: Logical. Suppress info message.
  - maxPixels: Numeric. The maximum allowed number of pixels in the exported image.
  - lazy: Logical. If TRUE, a future::sequential object is created to evaluate the task in the future.
  - public: Logical. If TRUE, a public link to the image will be created.
  - add_metadata: Logical. Add metadata to the stars_proxy object.
  - timePrefix: Logical. Add current date and time (Sys.time()) as a prefix to files to export.
  - quiet: Logical. Suppress info message.
  - maxPixels: Numeric. The maximum allowed number of pixels in the exported image.
  - lazy: Logical. If TRUE, a future::sequential object is created to evaluate the task in the future.
  - public: Logical. If TRUE, a public link to the image will be created.
  - add_metadata: Logical. Add metadata to the stars_proxy object.
  - timePrefix: Logical. Add current date and time (Sys.time()) as a prefix to files to export.
  - quiet: Logical. Suppress info message.
  - maxPixels: Numeric. The maximum allowed number of pixels in the exported image.
  - lazy: Logical. If TRUE, a future::sequential object is created to evaluate the task in the future.
  - public: Logical. If TRUE, a public link to the image will be created.
  - add_metadata: Logical. Add metadata to the stars_proxy object.
  - timePrefix: Logical. Add current date and time (Sys.time()) as a prefix to files to export.
  - quiet: Logical. Suppress info message.
  - maxPixels: Numeric. The maximum allowed number of pixels in the exported image.
  - lazy: Logical. If TRUE, a future::sequential object is created to evaluate the task in the future.
  - public: Logical. If TRUE, a public link to the image will be created.
  - add_metadata: Logical. Add metadata to the stars_proxy object.
  - timePrefix: Logical. Add current date and time (Sys.time()) as a prefix to files to export.
  - quiet: Logical. Suppress info message.
  - maxPixels: Numeric. The maximum allowed number of pixels in the exported image.
  - lazy: Logical. If TRUE, a future::sequential object is created to evaluate the task in the future.
  - public: Logical. If TRUE, a public link to the image will be created.
  - add_metadata: Logical. Add metadata to the stars_proxy object.
  - timePrefix: Logical. Add current date and time (Sys.time()) as a prefix to files to export.
  - quiet: Logical. Suppress info message.
  - maxPixels: Numeric. The maximum allowed number of pixels in the exported image.
  - lazy: Logical. If TRUE, a future::sequential object is created to evaluate the task in the future.
  - public: Logical. If TRUE, a public link to the image will be created.
  - add_metadata: Logical. Add metadata to the stars_proxy object.
  - timePrefix: Logical. Add current date and time (Sys.time()) as a prefix to files to export.
  - quiet: Logical. Suppress info message.
  - maxPixels: Numeric. The maximum allowed number of pixels in the exported image.
  - lazy: Logical. If TRUE, a future::sequential object is created to evaluate the task in the future.
  - public: Logical. If TRUE, a public link to the image will be created.
  - add_metadata: Logical. Add metadata to the stars_proxy object.
  - timePrefix: Logical. Add current date and time (Sys.time()) as a prefix to files to export.
  - quiet: Logical. Suppress info message.
  - maxPixels: Numeric. The maximum allowed number of pixels in the exported image.
  - lazy: Logical. If TRUE, a future::sequential object is created to evaluate the task in the future.
  - public: Logical. If TRUE, a public link to the image will be created.
  - add_metadata: Logical. Add metadata to the stars_proxy object.
  - timePrefix: Logical. Add current date and time (Sys.time()) as a prefix to files to export.
  - quiet: Logical. Suppress info message.
  - maxPixels: Numeric. The maximum allowed number of pixels in the exported image.
  - lazy: Logical. If TRUE, a future::sequential object is created to evaluate the task in the future.
  - public: Logical. If TRUE, a public link to the image will be created.
  - add_metadata: Logical. Add metadata to the stars_proxy object.
  - timePrefix: Logical. Add current date and time (Sys.time()) as a prefix to files to export.
  - quiet: Logical. Suppress info message.
  - maxPixels: Numeric. The maximum allowed number of pixels in the exported image.
  - lazy: Logical. If TRUE, a future::sequential object is created to evaluate the task in the future.
  - public: Logical. If TRUE, a public link to the image will be created.
  - add_metadata: Logical. Add metadata to the stars_proxy object.
  - timePrefix: Logical. Add current date and time (Sys.time()) as a prefix to files to export.
  - quiet: Logical. Suppress info message.
  - maxPixels: Numeric. The maximum allowed number of pixels in the exported image.
  - lazy: Logical. If TRUE, a future::sequential object is created to evaluate the task in the future.
  - public: Logical. If TRUE, a public link to the image will be created.
  - add_metadata: Logical. Add metadata to the stars_proxy object.
  - timePrefix: Logical. Add current date and time (Sys.time()) as a prefix to files to export.
  - quiet: Logical. Suppress info message.
  - maxPixels: Numeric. The maximum allowed number of pixels in the exported image.
* `ee_id`: Name of the Earth Engine task.
* `drive_name`: Name of the Image in Google Drive.
* `drive_id`: Id of the Image in Google Drive.
* `drive_download_link`: Download link to the image.

– if `via` is "gcs":

* `ee_id`: Name of the Earth Engine task.
* `gcs_name`: Name of the Image in Google Cloud Storage.
* `gcs_bucket`: Name of the bucket.
* `gcs_fileFormat`: Format of the image.
* `gcs_public_link`: Download link to the image.
* `gcs_URI`: gs:// link to the image.

Run `raster@history@metadata` to get the list.

For getting more information about exporting data from Earth Engine, take a look at the Google Earth Engine Guide - Export data.

**Value**

A RasterStack object

**See Also**

Other image download functions: `ee_as_stars()`, `ee_as_thumbnail()`, `ee_imagecollection_to_local()`

**Examples**

```r
## Not run:
library(rgdee)

ee_initialize(drive = TRUE, gcs = TRUE)
e_user_info()

# Define an image.
img <- ee$Image("LANDSAT/LC08/C01/T1_SR/LC08_038029_20180810")$
   select(c("B4", "B3", "B2"))$
   divide(10000)

# OPTIONAL display it using Map
Map$centerObject(eeObject = img)
Map$addLayer(eeObject = img, visParams = list(max = 0.04, gamma = 0.1))

# Define an area of interest.
geometry <- ee$Geometry$Rectangle(
   coords = c(-110.8, 44.6, -110.6, 44.7),
   proj = "EPSG:4326",
   geodesic = FALSE
)

## drive - Method 01
# Simple

```r
ing_02 <- ee_as_raster(
  image = img,
  region = geometry,
  via = "drive"
)
```

# Lazy

```r
img_02 <- ee_as_raster(
  image = img,
  region = geometry,
  via = "drive",
  lazy = TRUE
)

img_02_result <- img_02 %>% ee_utils_future_value()
```

```r
img_02_result@history$metadata # metadata
```

## gcs - Method 02

# Simple

```r
img_03 <- ee_as_raster(
  image = img,
  region = geometry,
  container = "rgee_dev",
  via = "gcs"
)
```

# Lazy

```r
img_03 <- ee_as_raster(
  image = img,
  region = geometry,
  container = "rgee_dev",
  lazy = TRUE,
  via = "gcs"
)
```

```r
img_03_result <- img_03 %>% ee_utils_future_value()
```

```r
img_03_result@history$metadata # metadata
```

# OPTIONAL: clean containers

```r
ee_clean_container(name = "rgee_backup", type = "drive")
ee_clean_container(name = "rgee_dev", type = "gcs")
```

## End(Not run)

---

### `ee_as_sf`

**Convert an Earth Engine table in a sf object**

**Description**

Convert an Earth Engine table in a sf object
Usage

\[
e \text{ee_as_sf}(\text{x, dsn, overwrite = TRUE,}}
\]

Arguments

\begin{itemize}
\item \textbf{x} \hspace{1cm} \text{Earth Engine table (ee$FeatureCollection) to be converted in a sf object.}
\item \textbf{dsn} \hspace{1cm} \text{Character. Output filename; in case dsn is missing a shapefile will be created in the tmp() directory.}
\item \textbf{overwrite} \hspace{1cm} \text{Logical. Delete data source dsn before attempting to write?}
\item \textbf{via} \hspace{1cm} \text{Character. Method to export the image. Three method are implemented: "getInfo", "drive", "gcs". See details.}
\item \textbf{container} \hspace{1cm} \text{Character. Name of the folder ('drive') or bucket ('gcs') to be exported into (ignore if via is not defined as 'drive' or 'gcs').}
\item \textbf{crs} \hspace{1cm} \text{Integer or character. coordinate reference system for the EE table. If is NULL, ee_as_sf will take the CRS of the first element.}
\item \textbf{maxFeatures} \hspace{1cm} \text{Numeric. The maximum allowed number of features to export (ignore if via is not set as "getInfo"). The task will fail if the exported region covers more features. Defaults to 5000.}
\item \textbf{selectors} \hspace{1cm} \text{The list of properties to include in the output, as a list of strings or a comma-separated string. By default, all properties are included.}
\item \textbf{lazy} \hspace{1cm} \text{Logical. If TRUE, a future::sequential object is created to evaluate the task in the future. Ignore if via is set as "getInfo". See details.}
\item \textbf{public} \hspace{1cm} \text{Logical. If TRUE, a public link to the image will be created.}
\item \textbf{add_metadata} \hspace{1cm} \text{Add metadata to the sf object. See details.}
\item \textbf{timePrefix} \hspace{1cm} \text{Logical. Add current date and time (Sys.time()) as a prefix to files to export. This parameter helps to avoid exported files with the same name. By default TRUE.}
\item \textbf{quiet} \hspace{1cm} \text{logical. Suppress info message.}
\end{itemize}
Details

`ee_as_sf` supports the download of `ee$Geometry`, `ee$Feature`, and `ee$FeatureCollection` by three different options: "getInfo" (which make an REST call to retrieve the data), "drive" (which use Google Drive) and "gcs" (which use Google Cloud Storage). The advantage of use "getInfo" is a direct and faster download. However, there is a limitation of 5000 features by request which makes it not recommendable for large FeatureCollections. Instead of "getInfo", the options: "drive" and "gcs" are suitable for large FeatureCollections since the use of an intermediate container. They work as follow:

- 1. A task will be started (i.e. `ee$batch$Task$start()`) to move the EE Table from Earth Engine to the intermediate container specified in argument `via`.
- 2. If the argument `lazy` is TRUE, the task will not be monitored. This is useful to lunch several tasks at the same time and call them later using `ee_utils_future_value` or `future::value`. At the end of this step, the EE Table will be stored on the path specified in the argument `dsn`.
- 3. Finally if the argument `add_metadata` is TRUE, a list with the following elements will be added to the sf object.

  - if `via` is "drive":
    - `ee_id`: Name of the Earth Engine task.
    - `drive_name`: Name of the Table in Google Drive.
    - `drive_id`: Id of the Table in Google Drive.
    - `drive_download_link`: Download link to the table.

  - if `via` is "gcs":
    - `ee_id`: Name of the Earth Engine task.
    - `gcs_name`: Name of the Table in Google Cloud Storage.
    - `gcs_bucket`: Name of the bucket.
    - `gcs_fileFormat`: Format of the table.
    - `gcs_public_link`: Download link to the table.
    - `gcs_URI`: gs:// link to the table.
  
  Run `attr(sf,"metadata")` to get the list.

For getting more information about exporting data from Earth Engine, take a look at the Google Earth Engine Guide - Export data.

Value

An sf object.

Examples

```r
# Not run:
library(rgge)

ee_Initialize(drive = TRUE, gcs = TRUE)

# Region of interest
roi <- ee$Geometry$Polygon(list(
```
Convert an Earth Engine (EE) image in a stars object

**Description**

Convert an ee$Image in a stars object.

**Usage**

```
ee_as_stars(
  image,
  region = NULL,
  dsn = NULL,
  via = "drive",
```
container = "rgge_backup",
scale = NULL,
maxPixels = 1e+09,
lazy = FALSE,
public = TRUE,
add_metadata = TRUE,
timePrefix = TRUE,
quiet = FALSE,
... )

Arguments

image ee$Image to be converted into a stars object.
region EE Geometry (ee$Geometry$Polygon) which specify the region to export. CRS needs to be the same that the argument image, otherwise, it will be forced. If not specified, image bounds will be taken.
dsn Character. Output filename. If missing, a temporary file will be created.
via Character. Method to export the image. Two method are implemented: "drive", "gcs". See details.
container Character. Name of the folder (‘drive’) or bucket (‘gcs’) to be exported into.
scale Numeric. The resolution in meters per pixel. Defaults to the native resolution of the image.
maxPixels Numeric. The maximum allowed number of pixels in the exported image. The task will fail if the exported region covers more pixels in the specified projection. Defaults to 100,000,000.
lazy Logical. If TRUE, a future::sequential object is created to evaluate the task in the future. See details.
public Logical. If TRUE, a public link to the image will be created.
add_metadata Add metadata to the stars_proxy object. See details.
timePrefix Logical. Add current date and time (Sys.time()) as a prefix to files to export. This parameter helps to avoid exported files with the same name. By default TRUE.
quiet Logical. Suppress info message
... Extra exporting argument. See ee_image_to_drive and ee_image_to_gcs.

Details

ee_as_stars supports the download of ee$Images by two different options: "drive" (Google Drive) and "gcs" (Google Cloud Storage). In both cases ee_as_stars works as follow:

• 1. A task will be started (i.e. ee$batch$Task$start()) to move the ee$Image from Earth Engine to the intermediate container specified in argument via.
• 2. If the argument lazy is TRUE, the task will not be monitored. This is useful to lunch several tasks at the same time and call them later using ee_utils_future_value or future::value. At the end of this step, the ee$Image will be stored on the path specified in the argument dsn.
3. Finally if the argument `add_metadata` is TRUE, a list with the following elements will be added to the stars-proxy object.

- **if via is "drive":**
  * ee_id: Name of the Earth Engine task.
  * drive_name: Name of the Image in Google Drive.
  * drive_id: Id of the Image in Google Drive.
  * drive_download_link: Download link to the image.

- **if via is "gcs":**
  * ee_id: Name of the Earth Engine task.
  * gcs_name: Name of the Image in Google Cloud Storage.
  * gcs_bucket: Name of the bucket.
  * gcs_fileFormat: Format of the image.
  * gcs_public_link: Download link to the image.
  * gcs_URI: gs:// link to the image.

Run `attr(stars,"metadata")` to get the list.

For getting more information about exporting data from Earth Engine, take a look at the [Google Earth Engine Guide - Export data](https://developers.google.com/earth-engine/guides/export).

**Value**

A stars-proxy object

**See Also**

Other image download functions: `ee_as_raster()`, `ee_as_thumbnail()`, `ee_imagecollection_to_local()`

**Examples**

```r
## Not run:
library(rgee)

ee_Initialize(drive = TRUE, gcs = TRUE)
e_user_info()

# Define an image.
img <- ee$Image("LANDSAT/LC08/C01/T1_SR/LC08_038029_20180810")$  
  select(c("B4", "B3", "B2"))$  
  divide(10000)

# OPTIONAL display it using Map
Map$centerObject(eeObject = img)
Map$addLayer(eeObject = img, visParams = list(max = 0.4, gamma=0.1))

# Define an area of interest.
geometry <- ee$Geometry$Rectangle(  
  coords = c(-110.8, 44.6, -110.6, 44.7),  
  proj = "EPSG:4326",
```
geodesic = FALSE
)

## drive - Method 01
## Simple
img_02 <- ee_as_stars(
  image = img,
  region = geometry,
  via = "drive"
)

## Lazy
img_02 <- ee_as_stars(
  image = img,
  region = geometry,
  via = "drive",
  lazy = TRUE
)

img_02_result <- img_02 %>% ee_utils_future_value()
attr(img_02_result, "metadata") # metadata

## gcs - Method 02
## Simple
img_03 <- ee_as_stars(
  image = img,
  region = geometry,
  container = "rgee_dev",
  via = "gcs"
)

## Lazy
img_03 <- ee_as_stars(
  image = img,
  region = geometry,
  container = "rgee_dev",
  lazy = TRUE,
  via = "gcs"
)

img_03_result <- img_03 %>% ee_utils_future_value()
attr(img_03_result, "metadata") # metadata

## OPTIONAL: clean containers
ee_clean_container(name = "rgee_backup", type = "drive")
ee_clean_container(name = "rgee_dev", type = "gcs")

## End(Not run)

ee_as_thumbnail

Create an R spatial gridded object from an EE thumbnail image
Description

Wrapper function around ee$Image$getThumbURL to create a stars or RasterLayer R object from a EE thumbnail image.

Usage

```r
ee_as_thumbnail(  
  image,  
  region,  
  dimensions,  
  vizparams = NULL,  
  raster = FALSE,  
  quiet = FALSE  
)
```

Arguments

- `image`: EE Image object to be converted into a stars object.
- `region`: EE Geometry Rectangle (ee$Geometry$Rectangle) specifying the region to export. The CRS needs to be the same as the x argument, otherwise, it will be forced.
- `dimensions`: Numeric vector of length 2. Thumbnail dimensions in pixel units. If a single integer is provided, it defines the size of the image’s larger aspect dimension and scales the smaller dimension proportionally. Defaults to 512 pixels for the larger image aspect dimension.
- `vizparams`: A list that contains the visualization parameters. See details.
- `raster`: Logical. Should the thumbnail image be saved as a RasterStack object?
- `quiet`: Logical; suppress info messages.

Details

`vizparams` set up the details of the thumbnail image. With `ee_as_thumbnail` only is possible export one-band (G) or three-band (RGB) images. Several parameters can be passed on to control color, intensity, the maximum and minimum values, etc. The table below provides all the parameters that admit `ee_as_thumbnail`.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bands</code></td>
<td>Comma-delimited list of three band names to be mapped to RGB</td>
<td>list</td>
</tr>
<tr>
<td><code>min</code></td>
<td>Value(s) to map to 0</td>
<td>number or list of three numbers, one for each band</td>
</tr>
<tr>
<td><code>max</code></td>
<td>Value(s) to map to 1</td>
<td>number or list of three numbers, one for each band</td>
</tr>
<tr>
<td><code>gain</code></td>
<td>Value(s) by which to multiply each pixel value</td>
<td>number or list of three numbers, one for each band</td>
</tr>
<tr>
<td><code>bias</code></td>
<td>Value(s) to add to each Digital Number (DN) value</td>
<td>number or list of three numbers, one for each band</td>
</tr>
<tr>
<td><code>gamma</code></td>
<td>Gamma correction factor(s)</td>
<td>number</td>
</tr>
<tr>
<td><code>palette</code></td>
<td>List of CSS-style color strings (single-band images only)</td>
<td>comma-separated list of hex strings</td>
</tr>
<tr>
<td><code>opacity</code></td>
<td>The opacity of the layer (0.0 is fully transparent and 1.0 is fully opaque)</td>
<td>number</td>
</tr>
</tbody>
</table>
Value
An `stars` or `Raster` object depending on the `raster` argument.

See Also
Other image download functions: `ee_as_raster()`, `ee_as_stars()`, `ee_imagecollection_to_local()`

Examples
```r
## Not run:
library(raster)
library(stars)
library(rgee)
eeeInitialize()

nc <- st_read(system.file("shp/arequipa.shp", package = "rgee"))
dem_palette <- c("#008435", "#1CAC17", "#48D00C", "#B3E34B", "#F4E467",
"#F4C84E", "#D59F3C", "#A36D2D", "#C6A889", "#FFFFFF")

## DEM data - SRTM v4.0
image <- eeImage("CGIAR/SRTM90_V4")
world_region <- eeGeometry$Rectangle(  
  coords = c(-180,-60,180,60),
  proj = "EPSG:4326",
  geodesic = FALSE
)

## world - elevation
world_dem <- ee_as_thumbnail(  
  image = image,
  region = world_region,
  dimensions = 1024,
  vizparams = list(min = 0, max = 5000)
)

world_dem[world_dem <= 0] <- NA
world_dem <- world_dem * 5000

plot(  
  x = world_dem, col = dem_palette, breaks = "equal",
  reset = FALSE, main = "SRTM - World"
)

## Arequipa-Peru
arequipa_region <- nc%>%  
  st_bbox() %>%
  st_as_sfc() %>%
  sf_as_ee()
```

arequipa_dem <- ee_as_thumbnail(
  image = image,
  region = arequipa_region$buffer(1000)$bounds(),
  dimensions = 512,
  vizparams = list(min = 0, max = 5000)
)

arequipa_dem <- arequipa_dem * 5000
st_crs(arequipa_dem) <- 4326
plot(
  x = arequipa_dem[nc], col = dem_palette, breaks = "equal",
  reset = FALSE, main = "SRTM - Arequipa"
)

suppressWarnings(plot(
  x = nc, col = NA, border = "black", add = TRUE,
  lwd = 1.5
))
dev.off()

## LANDSAT 8
img <- ee$Image("LANDSAT/LC08/C01/T1_SR/LC08_038029_20180810")$
  select(c("B4", "B3", "B2"))
Map$centerObject(img)
Map$addLayer(img, list(min = 0, max = 5000, gamma = 1.5))

## Teton Wilderness
l8_img <- ee_as_thumbnail(
  image = img,
  region = img$geometry()$bounds(),
  dimensions = 1024,
  vizparams = list(min = 0, max = 5000, gamma = 1.5),
  raster = TRUE
)


crs(l8_img) <- "+proj=longlat +datum=WGS84 +no_defs"
plotRGB(l8_img, stretch = "lin")

## End(Not run)

---

**ee_check-tools**  
*Interface to check Python and non-R dependencies*

**Description**

R functions for checking Google credentials (Google Earth Engine, Google Drive and Google Cloud Storage), Python environment and the Third-Party Python Packages used by rgee.

**Usage**

```r
ee_check(user = NULL, quiet = FALSE)
```
ee_check_python(quiet = FALSE)
ee_check_python_packages(quiet = FALSE)
ee_check_credentials(quiet = FALSE)

Arguments

user Character. User to check credentials. If it is not defined, ee_check will skip the check of credentials.
quiet Logical. Suppress info message

Value

No return value, called for checking non-R rgee dependencies.

Examples

```r
## Not run:
library(rgge)
ee_check_python()
ee_check_python_packages()
ee_check_credentials()
e_check() # put them all together
## End(Not run)
```

---

ee_clean_container Delete files from a either Folder or Bucket

Description

Delete all files from a folder (Google Drive) or a bucket (Google Cloud Storage). Caution: This will permanently delete your backup files generated by using ee_as_stars and ee_as_sf.

Usage

```r
ee_clean_container(name = "rgee_backup", type = "drive", quiet = FALSE)
```

Arguments

name Character. Name of the folder (Drive) or bucket (GCS) to delete all files into.
type Character. Name of the file storage web service. ‘drive’ and ‘gcs’ are supported.
quiet logical. Suppress info message
**Value**

No return value, called for cleaning Google Drive or Google Cloud Storage container.

**See Also**

Other ee_clean functions: `ee_clean_credentials()`, `ee_clean_pyenv()`
ee_clean_pyenv

Description
Remove rgee system variables from .Renviron

Usage
ee_clean_pyenv()

Value
No return value, called for cleaning environmental variables in their system.

See Also
Other ee_clean functions: ee_clean_container(), ee_clean_credentials()

ee_drive_to_local

Description
Move results from Google Drive to a local directory.

Usage
ee_drive_to_local(
  task,
  dsn,
  overwrite = TRUE,
  consider = TRUE,
  public = FALSE,
  metadata = FALSE,
  quiet = FALSE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>task</td>
<td>List generated after finished correctly a EE task. See details.</td>
</tr>
<tr>
<td>dsn</td>
<td>Character. Output filename. If missing, a temporary file will be assigned.</td>
</tr>
<tr>
<td>overwrite</td>
<td>A boolean argument which indicates indicating whether “filename” should be overwritten. By default TRUE.</td>
</tr>
<tr>
<td>consider</td>
<td>Interactive. See details.</td>
</tr>
</tbody>
</table>
The task argument needs a status as task "COMPLETED" to work, since the parameters necessary to identify EE objects into Google Drive are obtained from `ee$batch$Export*$toDrive(...)$start()$status()`. Consider argument is necessary since Google Drive permits users to create files with the same name. Consider uses an interactive R session by default to help users identify just the files that they want to download. Additionally, the options "last" and "all" are implemented. "last" will download just the last file saved in Google Drive while with "all" all files will be downloaded.

If `metadata` is FALSE will return the filename of the image. Otherwise, a list with two elements (dns and metadata) will be returned.

Other generic download functions: `ee_gcs_to_local()`

```r
## Not run:
library(rgee)
library(stars)
library(sf)
ee_users()
eelInitialize(drive = TRUE)

# Define study area (local -> earth engine)
# Communal Reserve Amarakaeri - Peru
rlist <- list(xmin = -71.13, xmax = -70.95, ymin = -12.89, ymax = -12.73)
ROI <- c(rlist$xmin, rlist$ymin, rlist xmax, rlist$ymin, rlist xmax, rlist$ymin, rlist xmin, rlist$ymin, rlist xmin, rlist$ymin)
eerROI <- matrix(ROI, ncol = 2, byrow = TRUE) =>

# Get the mean annual NDVI for 2011
cloudMaskL457 <- function(image) {

```
```r
qa <- image$select("pixel_qa")
cloud <- qa$bitwiseAnd(32L)$
And(qa$bitwiseAnd(128L))$Or(qa$bitwiseAnd(8L))
mask2 <- image$mask()$reduce(ee$Reducer$min())
image <- image$updateMask(cloud$Not())$updateMask(mask2)
image$normalizedDifference(list("B4", "B3"))

ic_l5 <- ee$ImageCollection("LANDSAT/LT05/C01/T1_SR")$
filterBounds(ee$FeatureCollection(ee_ROI))$
filterDate("2011-01-01", "2011-12-31")$
map(cloudMaskL457)

# Create simple composite
mean_l5 <- ic_l5$mean()$rename("NDVI")
mean_l5 <- mean_l5$reproject(crs = "EPSG:4326", scale = 500)
mean_l5_Amarakaeri <- mean_l5$clip(ee_ROI)

# Move results from Earth Engine to Drive
task_img <- ee_image_to_drive(
  image = mean_l5_Amarakaeri,
  folder = "Amarakaeri",
  fileFormat = "GEO_TIFF",
  region = ee_ROI,
  fileNamePrefix = "my_image_demo"
)
task_img$start()

# Move results from Drive to local
img <- ee_drive_to_local(task = task_img)

## End(Not run)
```

**ee_extract**

Extract values from EE Images or ImageCollections objects

**Description**

Extract values from a ee$Image at the locations of a geometry object. You can use ee$Geometry*, ee$Feature, ee$FeatureCollection, sf or sfc objects. This function mimicking how `extract` currently works.

**Usage**

```r
ee_extract(
  x,
  y,
```
fun = ee$Reducer$mean(),
scale = NULL,
sf = FALSE,
quiet = FALSE,
...
)

Arguments

x ee$Image.
y ee$Geometry*, ee$Feature, ee$FeatureCollection, sfc or sf objects.
fun ee$Reducer object. Function to summarize the values. The function must take a
single numeric value as an argument and return a single value. See details.
scale A nominal scale in meters of the Image projection to work in. By default 1000.
sf Logical. Should return a sf object?
quiet Logical. Suppress info message.
...

Details

The reducer functions that return one value are:

- **allNonZero**: Returns a Reducer that returns 1 if all of its inputs are non-zero, 0 otherwise.

- **anyNonZero**: Returns a Reducer that returns 1 if any of its inputs are non-zero, 0 otherwise.

- **bitwiseAnd**: Returns a Reducer that computes the bitwise-and summation of its inputs.

- **bitwiseOr**: Returns a Reducer that computes the bitwise-or summation of its inputs.

- **count**: Returns a Reducer that computes the number of non-null inputs.

- **first**: Returns a Reducer that returns the first of its inputs.

- **firstNonNull**: Returns a Reducer that returns the first of its non-null inputs.

- **kurtosis**: Returns a Reducer that Computes the kurtosis of its inputs.

- **last**: Returns a Reducer that returns the last of its inputs.

- **lastNonNull**: Returns a Reducer that returns the last of its non-null inputs.

- **max**: Creates a reducer that outputs the maximum value of its (first) input. If numInputs is
greater than one, also outputs the corresponding values of the additional inputs.

- **mean**: Returns a Reducer that computes the (weighted) arithmetic mean of its inputs.

- **median**: Create a reducer that will compute the median of the inputs. For small numbers of
inputs (up to maxRaw) the median will be computed directly; for larger numbers of inputs the
median will be derived from a histogram.

- **min**: Creates a reducer that outputs the minimum value of its (first) input. If numInputs is
greater than one, also outputs additional inputs.
- **mode**: Create a reducer that will compute the mode of the inputs. For small numbers of inputs (up to maxRaw) the mode will be computed directly; for larger numbers of inputs the mode will be derived from a histogram.
- **product**: Returns a Reducer that computes the product of its inputs.
- **sampleStdDev**: Returns a Reducer that computes the sample standard deviation of its inputs.
- **sampleVariance**: Returns a Reducer that computes the sample variance of its inputs.
- **stdDev**: Returns a Reducer that computes the standard deviation of its inputs.
- **sum**: Returns a Reducer that computes the (weighted) sum of its inputs.
- **variance**: Returns a Reducer that computes the variance of its inputs.

**Value**

A data.frame or an sf object depending on the sf argument. Column names are extracted from band names, use `ee$Image$rename` to rename the bands of an `ee$Image`. See `ee_help(ee$Image$rename).

**Examples**

```r
## Not run:
library(rgee)
library(sf)
ee_Initialize()

# Define a Image or ImageCollection: Terraclimate
terracclimate <- ee$ImageCollection("IDAHO_EPSCOR/TERRACLIMATE") %>%
ee$ImageCollection$filterDate("2001-01-01", "2002-01-01") %>%
ee$ImageCollection$map(
function(x) {
  date <- ee$Date(x$get("system:time_start"))$format("YYYY_MM_dd")
  name <- ee$String$cat("Terraclimate_pp_", date)
  x$select("pr")$rename(name)
}
)

# Define a geometry
nc <- st_read(
  dsn = system.file("shape/nc.shp", package = "sf"),
  stringsAsFactors = FALSE,
  quiet = TRUE
)

# Extract values
ee_nc_rain <- ee_extract(
  x = terracclimate,
  y = nc,
  scale = 250,
  fun = ee$Reducer$mean(),
  sf = TRUE
)
```
## Spatial plot
```r
plot(ee_nc_rain["Terraclimate_pp_2001_11_01"],
     main = "2001 Jan Precipitation - Terraclimate",
     reset = FALSE)
```

## ee_gcs_to_local

**Move results from Google Cloud Storage to a local directory**

### Description
Move results of an EE task saved in Google Cloud Storage to a local directory.

### Usage
```r
ee_gcs_to_local(
  task,
  dsn,
  public = FALSE,
  metadata = FALSE,
  overwrite = TRUE,
  quiet = FALSE
)
```

### Arguments
- **task**: List generated after finished correctly a EE task. See details.
- **dsn**: Character. Output filename. If missing, a temporary file will be assigned.
- **public**: Logical. If TRUE, a public link to the image will be created.
- **metadata**: Logical. If TRUE, export the metadata related to the image.
- **overwrite**: A boolean argument which indicates indicating whether "filename" should be overwritten. By default TRUE.
- **quiet**: Logical. Suppress info message

### Details
The task argument needs "COMPLETED" task state to work, since the parameters necessaries to locate the file into google cloud storage are obtained from `ee$batch$Export*$toCloudStorage(...)$start$status()`.

### Value
filename character vector.
See Also

Other generic download functions: `ee_drive_to_local()`

Examples

```r
## Not run:
library(rgee)
library(stars)
library(sf)

ee_users()
ee_Initialize(gcs = TRUE)

# Define study area (local -> earth engine)
# Communal Reserve Amarakaeri - Peru
rlist <- list(xmin = -71.13, xmax = -70.95, ymin = -12.89, ymax = -12.73)
ROI <- c(rlist$xmin, rlist$ymin,
          rlist$xmax, rlist$ymin,
          rlist$xmax, rlist$ymax,
          rlist$xmin, rlist$ymax,
          rlist$xmin, rlist$ymin)
ee_ROI <- matrix(ROI, ncol = 2, byrow = TRUE) %>%
           list() %>%
           st_polygon() %>%
           st_sfc() %>%
           st_set_crs(4326) %>%
           sf_as_ee()

# Get the mean annual NDVI for 2011
cloudMaskL457 <- function(image) {
  qa <- image$select("pixel_qa")
  cloud <- qa$bitwiseAnd(32L)$
    And(qa$bitwiseAnd(128L))$
    Or(qa$bitwiseAnd(8L))
  mask2 <- image$mask()$reduce(ee$Reducer$min())
  image <- image$updateMask(cloud$Not())$updateMask(mask2)
  image$normalizedDifference(list("B4", "B3"))
}
ic_l5 <- ee$ImageCollection("LANDSAT/LT05/C01/T1_SR")$filterBounds(ee$FeatureCollection(ee_ROI))$filterDate("2011-01-01", "2011-12-31")$map(cloudMaskL457)

# Create simple composite
mean_l5 <- ic_l5$mean()$rename("NDVI")
mean_l5 <- mean_l5$reproject(crs = "EPSG:4326", scale = 500)
mean_l5_Amarakaeri <- mean_l5$clip(ee_ROI)

# Move results from Earth Engine to Drive
task_img <- ee_image_to_gcs(
```
```r
image = mean_l5_Amarakaeri,
bucket = "rgee_dev",
fileFormat = "GEO_TIFF",
region = ee_ROI,
fileNamePrefix = "my_image_demo"
)

task_img$start()
ee_monitoring(task_img)

# Move results from Drive to local
ing <- ee_gcs_to_local(task = task_img)

## End(Not run)
```
```r
filterDate("2016-01-01", "2016-01-31")
filterBounds(nc)

ee_s2$size()$getInfo() # 126

# Get the first 5 elements
ee_get(ee_s2, index = 0:4)$size()$getInfo() # 5

## End(Not run)
```

---

**ee_get_assethome**

*Get the Asset home name*

**Description**

Get the Asset home name

**Usage**

```r
ee_get_assethome()
```

**Value**

Character. The name of the Earth Engine Asset home (e.g. users/datacolecfbf)

**See Also**

Other path utils: `ee_get_earthengine_path()`

**Examples**

```r
## Not run:
library(rgee)
ee_Initialize()
eee_get_assethome()

## End(Not run)
```
Description
Get the date of a EE ImageCollection

Usage
```
ee_get_date_ic(x, time_end = FALSE)
```

Arguments

- **x**: ee$ImageCollection object
- **time_end**: Logical. If TRUE, the system:time_end property is also returned. See details.

Details

system:time_start set the start period of data acquisition while system:time_end does the same for the end period. See the Earth Engine glossary for getting more information.

Value
A data.frame with the columns: id (ID of the image), time_start, and time_end (If the argument time_end is set as TRUE). The number of rows depends of the number of images (ee$ImageCollection$size).

See Also
Other date functions: `ee_get_date_img()`, `eedate_to_rdate()`, `rdate_to_eedate()`

Examples
```
## Not run:
library(rgee)
library(sf)
ee_Initialize()

nc <- st_read(system.file("shape/nc.shp", package = "sf")) %>%
  st_transform(4326) %>%
  sf_as_ee()

ee_s2 <- ee$ImageCollection("COPERNICUS/S2")$filterDate("2016-01-01", "2016-01-31")$filterBounds(nc)

ee_get_date_ic(ee_s2)

## End(Not run)
```
**ee_get_date_img**

Get the date of a EE Image

**Description**

Get the date of a EE Image

**Usage**

```r
ee_get_date_img(x, time_end = FALSE)
```

**Arguments**

- `x` : ee$Image or ee$ImageCollection object
- `time_end` : Logical. If TRUE, the `system:time_end` property is also returned. See details.

**Details**

`system:time_start` set the start period of data acquisition while `system:time_end` does the same for the end period. See the Earth Engine glossary for getting more information.

**Value**

An List object with the elements: id, time_start and time_end (if the `time_end` argument is TRUE).

**See Also**

Other date functions: `ee_get_date_ic()`, `eedate_to_rdate()`, `rdate_to_eedate()`

**Examples**

```r
## Not run:
library(rgge)
eE_Initialize()

l8 <- ee$Image('LANDSAT/LC08/C01/T1_TOA/LC08_044034_20140318')
ee_get_date_img(l8)
srtm <- ee$Image('CGIAR/SRTM90_V4')
ee_get_date_img(srtm, time_end = TRUE)

## End(Not run)
```
ee_get_earthengine_path

Get the path where the credentials are stored

Description

Get the path where the credentials are stored

Usage

ee_get_earthengine_path()

Value

A character which represents the path credential of a specific user

See Also

Other path utils: ee_get_assethome()

---

ee_help

Documentation for Earth Engine Objects

Description

Documentation for Earth Engine Objects

Usage

ee_help(eeobject, browser = FALSE)

Arguments

<table>
<thead>
<tr>
<th>eeobject</th>
<th>Earth Engine Object to print documentation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>browser</td>
<td>Logical. Display documentation in the browser.</td>
</tr>
</tbody>
</table>

Value

No return value, called for displaying Earth Engine documentation.

See Also

Other helper functions: ee_monitoring(), ee_print()
Examples

```r
## Not run:
library(rgee)
ee_initialize()

ee$Image()$geometry()$centroid %>% ee_help()
ee$Image()$geometry() %>% ee_help()
ee$Geometry$Rectangle(c(-110.8, 44.6, -110.6, 44.7)) %>% ee_help()
ee$Image %>% ee_help()
ee$Image %>% ee_help(browser = TRUE)

## End(Not run)
```

---

**ee_imagecollection_to_local**

*Save an EE ImageCollection in their local system*

### Description

Save an EE ImageCollection in their local system.

### Usage

```r
ee_imagecollection_to_local(
  ic,
  region,
  dsn = NULL,
  via = "drive",
  container = "rgee_backup",
  scale = NULL,
  maxPixels = 1e+09,
  lazy = FALSE,
  public = TRUE,
  add_metadata = TRUE,
  timePrefix = TRUE,
  quiet = FALSE,
  ...
)
```

### Arguments

- **ic**: ee$ImageCollection to be saved in the system.
- **region**: EE Geometry (ee$Geometry$Polygon). The CRS needs to be the same that the `ic` argument otherwise it will be forced.
- **dsn**: Character. Output filename. If missing, a temporary file will be created for each image.
via
Character. Method to export the image. Two method are implemented: "drive", "gcs". See details.

container
Character. Name of the folder ('drive') or bucket ('gcs') to be exported into (ignored if via is not defined as "drive" or "gcs").

scale
Numeric. The resolution in meters per pixel. Defaults to the native resolution of the image.

maxPixels
Numeric. The maximum allowed number of pixels in the exported image. The task will fail if the exported region covers more pixels in the specified projection. Defaults to 100,000,000.

lazy
Logical. If TRUE, a future::sequential object is created to evaluate the task in the future. See details.

public
Logical. If TRUE, a public link to the image will be created.

add_metadata
Add metadata to the stars_proxy object. See details.

timePrefix
Logical. Add current date and time (Sys.time()) as a prefix to files to export. This parameter helps to avoid exported files with the same name. By default TRUE.

quiet
Logical. Suppress info message

Extra exporting argument. See ee_image_to_drive and ee_imagecollection_to_local

Details

ee_imagecollection_to_local supports the download of ee$Images by two different options: "drive" (Google Drive) and "gcs" (Google Cloud Storage). In both cases ee_imagecollection_to_local works as follow:

• 1. A task will be started (i.e. ee$batch$Task$start()) to move the ee$Image from Earth Engine to the intermediate container specified in argument via.

• 2. If the argument lazy is TRUE, the task will not be monitored. This is useful to lunch several tasks at the same time and call them later using ee_utils_future_value or future::value. At the end of this step, the ee$Images will be stored on the path specified in the argument dsn.

• 3. Finally if the argument add_metadata is TRUE, a list with the following elements will be added to the argument dsn.

  – if via is "drive":
    * ee_id: Name of the Earth Engine task.
    * drive_name: Name of the Image in Google Drive.
    * drive_id: Id of the Image in Google Drive.
    * drive_download_link: Download link to the image.

  – if via is "gcs":
    * ee_id: Name of the Earth Engine task.
    * gcs_name: Name of the Image in Google Cloud Storage.
    * gcs_bucket: Name of the bucket.
    * gcs_fileFormat: Format of the image.
    * gcs_public_link: Download link to the image.
    * gcs_URI: gs:// link to the image.

For getting more information about exporting data from Earth Engine, take a look at the Google Earth Engine Guide - Export data.
ee_imagecollection_to_local

Value

If `add_metadata` is FALSE, a character vector containing the filename of the images downloaded. Otherwise a list adding information related to the exportation (see details).

See Also

Other image download functions: `ee_as_raster()`, `ee_as_stars()`, `ee_as_thumbnail()`

Examples

```r
## Not run:
library(rgee)
library(raster)
ees.Initialize(drive = TRUE, gcs = TRUE)

# USDA example
loc <- ee$Geometry$Point(-99.2222, 46.7816)
collection <- ee$ImageCollection('USDA/NAIP/DOQQ')$filterBounds(loc)$
  filterDate('2008-01-01', '2020-01-01')$
  filter(ee$Filter$listContains('system:band_names', "N"))

# From ImageCollection to local directory
ee_crs <- collection$first()$projection()$getInfo()$crs
geometry <- collection$first()$geometry(proj = ee_crs)$bounds()
tmp <- tempdir()

## Using drive
# one by once
ic_drive_files_1 <- ee_imagecollection_to_local(
  ic = collection,
  region = geometry,
  scale = 250,
  dsn = file.path(tmp, "drive_")
)

# all at once
ic_drive_files_2 <- ee_imagecollection_to_local(
  ic = collection,
  region = geometry,
  scale = 250,
  lazy = TRUE,
  dsn = file.path(tmp, "drive_")
)

# From Google Drive to client-side
doqq_dsn <- ic_drive_files_2 %>% ee_utils_future_value()$sapply(doqq_dsn, '[[', 1)

## End(Not run)
```
**Description**

Get the approximate number of rows, cols, and size of an single-band Earth Engine Image.

**Usage**

```r
ee_image_info(image, getsize = TRUE, compression_ratio = 20, quiet = FALSE)
```

**Arguments**

- `image` Single-band EE Image object.
- `getsize` Logical. If TRUE, the size of the object will be estimated.
- `compression_ratio` Numeric. Measurement of the relative reduction in size of data representation produced by a data compression algorithm (ignored if `getsize` is FALSE). By default is 20
- `quiet` Logical. Suppress info message

**Value**

A list containing information about the number of rows (`nrow`), number of columns (`ncol`), total number of pixels (`total_pixel`), and image size (`image_size`).

**Examples**

```r
## Not run:
library(rgee)
ee_Initiaize()

# World SRTM
srtm <- ee$Image("CGIAR/SRTM90_V4")
ee_image_info(srtm)

# Landsat8
l8 <- ee$Image("LANDSAT/LC08/C01/T1_SR/LC08_038029_20180810")$select("B4")
ee_image_info(l8)

## End(Not run)
```
ee_image_to_asset  Creates a task to export an EE Image to their EE Assets.

Description

Creates a task to export an EE Image to their EE Assets. This function is a wrapper around ee$batch$Export$image$toAsset(...).

Usage

```r
ee_image_to_asset(
    image,
    description = "myExportImageTask",
    assetId = NULL,
    overwrite = FALSE,
    pyramidingPolicy = NULL,
    dimensions = NULL,
    region = NULL,
    scale = NULL,
    crs = NULL,
    crsTransform = NULL,
    maxPixels = NULL
)
```

Arguments

- **image**: The image to be exported.
- **description**: Human-readable name of the task.
- **assetId**: The destination asset ID.
- **overwrite**: Logical. If TRUE, the assetId will be overwritten if it exists.
- **pyramidingPolicy**: The pyramiding policy to apply to each band in the image, a dictionary keyed by band name. Values must be one of: "mean", "sample", "min", "max", or "mode". Defaults to "mean". A special key, ".default", may be used to change the default for all bands.
- **dimensions**: The dimensions of the exported image. Takes either a single positive integer as the maximum dimension or "WIDTHxHEIGHT" where WIDTH and HEIGHT are each positive integers.
- **region**: The lon,lat coordinates for a LinearRing or Polygon specifying the region to export. Can be specified as a nested lists of numbers or a serialized string. Defaults to the image’s region.
- **scale**: The resolution in meters per pixel. Defaults to the native resolution of the image asset unless a crsTransform is specified.
- **crs**: The coordinate reference system of the exported image’s projection. Defaults to the image’s default projection.
crsTransform  A comma-separated string of 6 numbers describing the affine transform of the
coordinate reference system of the exported image’s projection, in the order:
xScale, xShearing, xTranslation, yShearing, yScale and yTranslation. Defaults
to the image’s native CRS transform.

maxPixels  The maximum allowed number of pixels in the exported image. The task will
fail if the exported region covers more pixels in the specified projection. De-
faults to 100,000,000. **kwargs: Holds other keyword arguments that may
have been deprecated such as ‘crs_transform’.

Value
An unstarted task

See Also
Other image export task creator: ee_image_to_drive(), ee_image_to_gcs()

Examples
## Not run:
library(rgee)
library(stars)
library(sf)
eе_users()
eе_Initialize()

# Define study area (local -> earth engine)
# Communal Reserve Amarakaeri - Peru
rlist <- list(xmin = -71.13, xmax = -70.95, ymin = -12.89, ymax = -12.73)
ROI <- c(rlist$xmin, rlist$ymin,
        rlistxmax, rlist$ymin,
        rlistxmax, rlist$ymax,
        rlistxmin, rlist$ymax,
        rlistxmin, rlist$ymin)
eе_ROI <- matrix(ROI, ncol = 2, byrow = TRUE) %>%
          list() %>%
          st_polygon() %>%
          st_sfc() %>%
          st_set_crs(4326) %>%
sf_as_ee()

# Get the mean annual NDVI for 2011
cloudMaskL457 <- function(image) {
  qa <- image$select("pixel_qa")
  cloud <- qa$bitwiseAnd(32L)$
  And(qa$bitwiseAnd(128L))$
  Or(qa$bitwiseAnd(8L))
  mask2 <- image$mask()$reduce(ee$Reducer$min())
  image <- image$updateMask(cloud$Not())$updateMask(mask2)
  image$normalizedDifference(list("B4", "B3"))
ic_l5 <- ee$ImageCollection("LANDSAT/LT05/C01/T1_SR")$filterBounds(ee$FeatureCollection(ee_ROI))$filterDate("2011-01-01", "2011-12-31")$map(cloudMaskL457)

# Create simple composite
mean_l5 <- ic_l5$mean()$rename("NDVI")
mean_l5 <- mean_l5$reproject(crs = "EPSG:4326", scale = 500)
mean_l5_Amarakaeri <- mean_l5$clip(ee_ROI)

# Move results from Earth Engine to Drive
assetid <- paste0(ee_get_assethome(), '/l5_Amarakaeri')
task_img <- ee_image_to_asset(
  image = mean_l5_Amarakaeri,
  assetId = assetid,
  overwrite = TRUE,
  scale = 500,
  region = ee_ROI
)

  start()
  ee_monitoring(task_img)

ee_l5 <- ee$Image(assetid)
Map$centerObject(ee_l5)
Map$addLayer(ee_l5)

## End(Not run)

---

**ee_image_to_drive**  
*Creates a task to export an EE Image to Drive.*

**Description**

Creates a task to export an EE Image to Drive. This function is a wrapper around `ee$batch$Export$image$toDrive(...)`.  

**Usage**

```r
ee_image_to_drive(
  image,
  description = "myExportImageTask",
  folder = "rgee_backup",
  fileNamePrefix = NULL,
  timePrefix = TRUE,
  dimensions = NULL,
  region = NULL,
  scale = NULL,
)```
Arguments

image  The image to be exported.
description  Human-readable name of the task.
folder  The name of a unique folder in your Drive account to be exported into. Defaults to the folder rgee-backup.
fileNamePrefix  The Google Drive filename for the export. Defaults to the name of the task.
timePrefix  Add current date and time as a prefix to files to export.
dimensions  The dimensions of the exported image. Takes either a single positive integer as the maximum dimension or "WIDTHxHEIGHT" where WIDTH and HEIGHT are each positive integers.
region  The lon,lat coordinates for a LinearRing or Polygon specifying the region to export. Can be specified as a nested lists of numbers or a serialized string. Defaults to the image’s region.
scale  The resolution in meters per pixel. Defaults to the native resolution of the image asset unless a crsTransform is specified.
crs  The coordinate reference system of the exported image’s projection. Defaults to the image’s native CRS.
crsTransform  A comma-separated string of 6 numbers describing the affine transform of the coordinate reference system of the exported image’s projection, in the order: xScale, xShearing, xTranslation, yShearing, yScale and yTranslation. Defaults to the image’s native CRS transform.
maxPixels  The maximum allowed number of pixels in the exported image. The task will fail if the exported region covers more pixels in the specified projection. Defaults to 100,000,000.
shardSize  Size in pixels of the shards in which this image will be computed. Defaults to 256.
fileDimensions  The dimensions in pixels of each image file, if the image is too large to fit in a single file. May specify a single number to indicate a square shape, or a list of two dimensions to indicate (width,height). Note that the image will still be clipped to the overall image dimensions. Must be a multiple of shardSize.
skipEmptyTiles  If TRUE, skip writing empty (i.e. fully-masked) image tiles. Defaults to FALSE.
fileFormat  The string file format to which the image is exported. Currently only 'GeoTIFF' and 'TFRecord' are supported, defaults to 'GeoTIFF'.
formatOptions  A dictionary of string keys to format specific options. **kwargs: Holds other keyword arguments that may have been deprecated such as 'crs_transform', 'driveFolder', and 'driveFileNamePrefix'.

**Value**

An unstarted Task that exports the image to Drive.

**See Also**

Other image export task creator: ee_image_to_asset(), ee_image_to_gcs()

**Examples**

```r
## Not run:
library(rgee)
library(stars)
library(sf)

e_users()
e_Initialize(drive = TRUE)

# Define study area (local --> earth engine)
# Communal Reserve Amarakaeri - Peru
rlist <- list(xmin = -71.13, xmax = -70.95, ymin = -12.89, ymax = -12.73)
ROI <- c(rlist$xmin, rlist$ymin, rlistxmax, rlist$ymin, rlistxmax, rlist$ymin, rlistxmin, rlistymax, rlistxmin, rlist$ymin)

ee_ROI <- matrix(ROI, ncol = 2, byrow = TRUE) %>%
list() %>%
st_polygon() %>%
st_sfc() %>%
st_set_crs(4326) %>%
sf_as_ee()

# Get the mean annual NDVI for 2011
cloudMaskL457 <- function(image) {
  qa <- image$select("pixel_qa")
  cloud <- qa$bitwiseAnd(32L)$And(qa$bitwiseAnd(128L))$Or(qa$bitwiseAnd(8L))
  mask2 <- image$mask()$reduce(ee$Reducer$min())
  image <- image$updateMask(cloud$Not())$updateMask(mask2)
  image$normalizedDifference(list("B4", "B3"))
}
ic_l5 <- ee$ImageCollection("LANDSAT/LT05/C01/T1_SR")$filterBounds(ee$FeatureCollection(ee_ROI))$
filterDate("2011-01-01", "2011-12-31")
map(cloudMaskL457)

# Create simple composite
mean_15 <- ic_l5$mean()$rename("NDVI")
mean_15 <- mean_15$reproject(crs = "EPSG:4326", scale = 500)
mean_15_Amarakaeri <- mean_15$clip(ee_ROI)

# Move results from Earth Engine to Drive
task_img <- ee_image_to_drive(
  image = mean_15_Amarakaeri,
  fileFormat = "GEO_TIFF",
  region = ee_ROI,
  fileNamePrefix = "my_image_demo"
)
task_img$start()
ee_monitoring(task_img)

# Move results from Drive to local
ee_drive_to_local(task = task_img)

## End(Not run)

### ee_image_to_gcs

Creates a task to export an EE Image to Google Cloud Storage.

**Description**

Creates a task to export an EE Image to Google Cloud Storage. This function is a wrapper around `ee$batch$Export$image$toCloudStorage(...)`.  

**Usage**

```r
ee_image_to_gcs(
  image,
  description = "myExportImageTask",
  bucket = NULL,
  fileNamePrefix = NULL,
  timePrefix = TRUE,
  dimensions = NULL,
  region = NULL,
  scale = NULL,
  crs = NULL,
  crsTransform = NULL,
  maxPixels = NULL,
  shardSize = NULL,
  fileDimensions = NULL,
  skipEmptyTiles = NULL,
```
Arguments

image  The image to be exported.
description  Human-readable name of the task.
bucket  The name of a Cloud Storage bucket for the export.
fileNamePrefix  Cloud Storage object name prefix for the export. Defaults to the name of the task.
timePrefix  Add current date and time as a prefix to files to export.
dimensions  The dimensions of the exported image. Takes either a single positive integer as the maximum dimension or "WIDTHxHEIGHT" where WIDTH and HEIGHT are each positive integers.
region  The lon,lat coordinates for a LinearRing or Polygon specifying the region to export. Can be specified as a nested lists of numbers or a serialized string. Defaults to the image’s region.
scale  The resolution in meters per pixel. Defaults to the native resolution of the image asset unless a crsTransform is specified.
crs  The coordinate reference system of the exported image’s projection. Defaults to the image’s default projection.
crsTransform  A comma-separated string of 6 numbers describing the affine transform of the coordinate reference system of the exported image’s projection, in the order: xScale, xShearing, xTranslation, yShearing, yScale and yTranslation. Defaults to the image’s native CRS transform.
maxPixels  The maximum allowed number of pixels in the exported image. The task will fail if the exported region covers more pixels in the specified projection. Defaults to 100,000,000.
shardSize  Size in pixels of the shards in which this image will be computed. Defaults to 256.
fileDimensions  The dimensions in pixels of each image file, if the image is too large to fit in a single file. May specify a single number to indicate a square shape, or a list of two dimensions to indicate (width,height). Note that the image will still be clipped to the overall image dimensions. Must be a multiple of shardSize.
skipEmptyTiles  If TRUE, skip writing empty (i.e. fully-masked) image tiles. Defaults to FALSE.
fileFormat  The string file format to which the image is exported. Currently only ‘GeoTIFF’ and ‘TFRecord’ are supported, defaults to ‘GeoTIFF’.
formatOptions  A dictionary of string keys to format specific options. **kwargs: Holds other keyword arguments that may have been deprecated such as ‘crs_transform’.

Value

An unstarted Task that exports the image to Google Cloud Storage.
See Also

Other image export task creator: `ee_image_to_asset()`, `ee_image_to_drive()`

Examples

```r
## Not run:
library(rgge)
library(stars)
library(sf)

ee_users()
ee_Initialize(gcs = TRUE)

# Define study area (local -> earth engine)
# Communal Reserve Amarakaeri - Peru
rlist <- list(xmin = -71.13, xmax = -70.95, ymin = -12.89, ymax = -12.73)
ROI <- c(rlist$xmin, rlist$ymin, 
         rlist$xmax, rlist$ymin, 
         rlist$xmax, rlist$ymax, 
         rlist$xmin, rlist$ymax, 
         rlist$xmin, rlist$ymin)

ee_ROI <- matrix(ROI, ncol = 2, byrow = TRUE) %>%
  list() %>%
  st_polygon() %>%
  st_sfc() %>%
  st_set_crs(4326) %>%
  sf_as_ee()

# Get the mean annual NDVI for 2011
cloudMaskL457 <- function(image) {
  qa <- image$select("pixel_qa")
  cloud <- qa$bitwiseAnd(32L)$
  And(qa$bitwiseAnd(128L))$
  Or(qa$bitwiseAnd(8L))$
  mask2 <- image$mask()$reduce(ee$Reducer$min())
  image <- image$updateMask(cloud$Not())$updateMask(mask2)
  image$normalizedDifference(list("B4", "B3"))
}

ic_l5 <- ee$ImageCollection("LANDSAT/LT05/C01/T1_SR")$
  filterBounds(ee$FeatureCollection(ee_ROI))$
  filterDate("2011-01-01", "2011-12-31")$
  map(cloudMaskL457)

# Create simple composite
mean_l5 <- ic_l5$mean()$rename("NDVI")
mean_l5 <- mean_l5$reproject(crs = "EPSG:4326", scale = 500)
mean_l5_Amarakaeri <- mean_l5$clip(ee_ROI)

# Move results from Earth Engine to GCS
task_img <- ee_image_to_gcs()
```
ee_Initialize

Authenticate and Initialize Earth Engine

Description

Authorize rgee to manage Earth Engine resources, Google Drive, and Google Cloud Storage. The `ee_initialize()` via web-browser will ask to sign in to your Google account and allows you to grant permission to manage resources. This function is a wrapper around `rgee::ee$Initialize()`.

Usage

```r
ee_Initialize(
  email = NULL,
  drive = FALSE,
  gcs = FALSE,
  display = FALSE,
  quiet = FALSE
)
```

Arguments

- **email**: Character (optional, e.g. `data.colec.fbf@gmail.com`). The email argument is used to create a folder inside the path `~/.config/earthengine/` that save all credentials for a specific Google identity.
- **drive**: Logical (optional). If TRUE, the drive credential will be cached in the path `~/.config/earthengine/`.
- **gcs**: Logical (optional). If TRUE, the Google Cloud Storage credential will be cached in the path `~/.config/earthengine/`.
- **display**: Logical. If TRUE display the earthengine authentication URL.
- **quiet**: Logical. Suppress info messages.
**Details**

`ee.Initialize(...)` can manage Google drive and Google Cloud Storage resources using the R packages googledrive and googlecloudStorageR, respectively. By default, rgee does not require them, these are only necessary to enable rgee I/O functionality. All user credentials are saved in the directory `~/.config/earthengine/`, if a user does not specify the email argument all user credentials will be saved in a subdirectory called `~/.config/earthengine/ndef`.

**Value**

No return value, called for initializing the earthengine-api.

**See Also**

Other session management functions: `ee_user_info()`, `ee_users()`, `ee_version()`

**Examples**

```r
## Not run:
library(rgee)

# Simple init - Load just the Earth Engine credential
ee.Initialize()
ee_user_info()

## End(Not run)
```

---

**ee_install**

*Create an isolated Python virtual environment with all rgee dependencies.*

**Description**

Create an isolated Python virtual environment with all rgee dependencies. `ee_install` realize the following six (6) tasks:

1. If you do not count with a Python environment, it will display an interactive menu to install Miniconda (a free minimal installer for conda).
2. Remove the previous Python environment defined in `py.env` if it exist.
3. Create a new Python environment (See `py.env`).
4. Set the environment variable EARTHENGINE_PYTHON. It is used to define RETICULATE_PYTHON when the library is loaded. See this article for further details.
5. Install rgee Python dependencies. Using reticulate::py_install.
6. Interactive menu to confirm if restart the R session to see changes.
Usage

```r
ee_install(
    py_env = "rgee",
    earthengine_version = ee_version(),
    confirm = interactive()
)
```

Arguments

- **py_env**: Character. The name, or full path, of the Python environment to be used by `rgee`.
- **earthengine_version**: Character. The Earth Engine Python API version to install. By default `rgee::ee_version()`.
- **confirm**: Logical. Confirm before restarting R?.

Value

No return value, called for installing non-R dependencies.

See Also

Other `ee_install` functions: `ee_install_set_pyenv()`, `ee_install_upgrade()`

---

### ee_install_set_pyenv

*Set the Python environment to be used by rgee*

#### Description

This function create a new environment variable called 'EARTHENGINE_PYTHON'. It is used to set the Python environment to be used by `rgee`. EARTHENGINE_PYTHON is saved into the file `.Renviron`.

#### Usage

```r
ee_install_set_pyenv(py_path = NULL, py_env = NULL, quiet = FALSE)
```

#### Arguments

- **py_path**: The path to a Python interpreter
- **py_env**: The name of the environment
- **quiet**: Logical. Suppress info message

#### Value

no return value, called for setting EARTHENGINE_PYTHON in `.Renviron`

#### See Also

Other `ee_install` functions: `ee_install_upgrade()`, `ee_install()`
ee_install_upgrade

 Upgrade the Earth Engine Python API

Description
Upgrade the Earth Engine Python API

Usage
```r
ee_install_upgrade(
  version = NULL,
  earthengine_env = Sys.getenv("EARTHENGINE_ENV")
)
```

Arguments
- `version`: Character. The Earth Engine Python API version to upgrade. By default `rgee::ee_version()`.
- `earthengine_env`: Character. The name, or full path, of the environment in which the earthengine-api packages are to be installed.

Value
no return value, called to upgrade the earthengine-api Python package

See Also
Other ee_install functions: `ee_install_set_pyenv()`, `ee_install()`

ee_manage-tools

 Interface for manage the Earth Engine Asset

Description
R functions for managing the Earth Engine Asset. The interface allows users to create and eliminate folders, move and copy assets, set and delete properties, handle access control lists, and manage and/or cancel tasks.
Usage

```r
ee_manage_create(path_asset, asset_type = "Folder", quiet = FALSE)

ee_manage_delete(path_asset, quiet = FALSE, strict = TRUE)

ee_manage_assetlist(path_asset, quiet = FALSE, strict = TRUE)

ee_manage_quota(quiet = FALSE)

ee_manage_copy(path_asset, final_path, strict = TRUE, quiet = FALSE)

ee_manage_move(path_asset, final_path, strict = TRUE, quiet = FALSE)

ee_manage_set_properties(path_asset, add_properties, strict = TRUE)

ee_manage_delete_properties(path_asset, del_properties = "ALL", strict = TRUE)

ee_manage_asset_access(
  path_asset,
  owner = NULL,
  editor = NULL,
  viewer = NULL,
  all_users_can_read = TRUE,
  quiet = FALSE
)

ee_manage_task(cache = FALSE)

ee_manage_cancel_all_running_task()

ee_manage_asset_size(path_asset, quiet = FALSE)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path_asset</td>
<td>Character. Name of the EE asset (Table, Image, Folder or ImageCollection).</td>
</tr>
<tr>
<td>asset_type</td>
<td>Character. The asset type to create ('Folder' or 'ImageCollection').</td>
</tr>
<tr>
<td>quiet</td>
<td>Logical. Suppress info message.</td>
</tr>
<tr>
<td>strict</td>
<td>Character vector. If TRUE, the existence of the asset will be evaluate before to perform the task.</td>
</tr>
<tr>
<td>final_path</td>
<td>Character. Output filename (e.g users/datacolcbbfjc_moved)</td>
</tr>
<tr>
<td>add_properties</td>
<td>List. Set of parameters to established as a property of an EE object. See details.</td>
</tr>
<tr>
<td>del_properties</td>
<td>Character. Names of properties to be deleted. See details.</td>
</tr>
<tr>
<td>owner</td>
<td>Character vector. Define owner user in the IAM Policy.</td>
</tr>
<tr>
<td>editor</td>
<td>Character vector. Define editor users in the IAM Policy.</td>
</tr>
<tr>
<td>viewer</td>
<td>Character vector. Define viewer users in the IAM Policy.</td>
</tr>
</tbody>
</table>
all_users_can_read
Logical. All users can see the asset element.
cache
Logical. If TRUE, the task report will be saved in the /temp directory and used when the function.

Details
If the argument del_properties is 'ALL', ee_manage_delete_properties will delete all the properties.

Author(s)
Samapriya Roy, adapted to R by csaybar.

Examples
```r
## Not run:
library(rgee)
ee_Initialize()
ee_user_info()

# Change datacolecbf by your EE user to be able to reproduce
user <- ee_get_assethome()
addm <- function(x) sprintf("%s/%s",user, x)
# 1. Create a folder or Image Collection
# Change path asset according to your specific user
ee_manage_create(addm("rgee"))

# 1. List all the elements inside a folder or a ImageCollection
ee_manage_assetlist(path_asset = addm("rgee"))

# 2. Create a Folder or a ImageCollection
ee_manage_create(path_asset = addm("rgee/rgee_folder"),
                 asset_type = "Folder")

ee_manage_create(path_asset = addm("rgee/rgee_ic"),
                 asset_type = "ImageCollection")

ee_manage_assetlist(path_asset = addm("rgee"))

# 3. Shows Earth Engine quota
ee_manage_quota()

# 4. Move an EE object to another folder
ee_manage_move(path_asset = addm("rgee/rgee_ic"),
               final_path = addm("rgee/rgee_folder/rgee_ic_moved"))
```
ee_monitoring

Description
Monitoring Earth Engine task progress

Usage
ee_monitoring(task, task_time = 5, eeTaskList = FALSE, quiet = FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>task</td>
<td>List generated after a task is started (i.e. after run \texttt{ee$batch$Task$start()}) or a character that represents the ID of a EE task started.</td>
</tr>
<tr>
<td>task_time</td>
<td>Numeric. How often (in seconds) should a task be polled?</td>
</tr>
<tr>
<td>eeTaskList</td>
<td>Logical. If \texttt{TRUE}, all Earth Engine tasks will be listed.</td>
</tr>
<tr>
<td>quiet</td>
<td>Logical. Suppress info message</td>
</tr>
</tbody>
</table>
Value

An ee$batch$Task object with a state "COMPLETED" or "FAILED" according to the response of the Earth Engine server.

See Also

Other helper functions: ee_help(), ee_print()

Examples

## Not run:
library(rgee)
e.Initialize()
e_monitoring(eeTaskList = TRUE)

## End(Not run)

---

**ee_print**

*Print and return metadata about Spatial Earth Engine Objects*

Description

Print and return metadata about Spatial Earth Engine Objects. ee_print can retrieve information about the number of images or features, number of bands or geometries, number of pixels, geotransform, data type, properties and object size.

Usage

```r
ee_print(eeobject, ...)
```

## S3 method for class 'ee.geometry.Geometry'

```r
ee_print(eeobject, ..., clean = FALSE, quiet = FALSE)
```

## S3 method for class 'ee.feature.Feature'

```r
ee_print(eeobject, ..., clean = FALSE, quiet = FALSE)
```

## S3 method for class 'ee.featurecollection.FeatureCollection'

```r
ee_print(eeobject, ..., f_index = 0, clean = FALSE, quiet = FALSE)
```

## S3 method for class 'ee.image.Image'

```r
ee_print(
  eeobject,
  ...,
  img_band,
  time_end = TRUE,
  compression_ratio = 20,
  clean = FALSE,
)```
quiet = FALSE
)

## S3 method for class 'ee.imagecollection.ImageCollection'
ee_print(
  eeobject,
  ...,
  time_end = TRUE,
  img_index = 0,
  img_band,
  compression_ratio = 20,
  clean = FALSE,
  quiet = FALSE
)

Arguments

**eeobject** Earth Engine Object. Available for: Geometry, Feature, FeatureCollection, Image or ImageCollection.

**...** ignored

**clean** Logical. If TRUE, the cache will be cleaned.

**quiet** Logical. Suppress info message

**f_index** Numeric. Index of the ee$FeatureCollection to fetch. Relevant just for ee$FeatureCollection objects.

**img_band** Character. Band name of the ee$Image to fetch. Relevant just for ee$ImageCollection and ee$Image objects.

**time_end** Logical. If TRUE, the system:time_end property in ee$Image is also returned. See rgee::ee_get_date_img for details.

**compression_ratio** Numeric. Measurement of the relative reduction in size of data representation produced by a data compression algorithm (ignored if eeobject is not a Image or ImageCollection). By default is 20.

**img_index** Numeric. Index of the ee$ImageCollection to fetch. Relevant just for ee$ImageCollection objects.

Value

A list with the metadata of the Earth Engine object.

See Also

Other helper functions: `ee_help()`, `ee_monitoring()`

Examples

```r
## Not run:
library(rgee)
```
ee_table_to_asset

      Creates a task to export a FeatureCollection to an EE table asset.

Description

Creates a task to export a FeatureCollection to an EE table asset. This function is a wrapper around ee$batch$Export$table$toAsset(...).

Usage

ee_table_to_asset(
  collection,
  description = "myExportTableTask",
  assetId = NULL,
  overwrite = FALSE
)

# Geometry
geom <- ee$Geometry$Rectangle(-10,-10,10,10)
Map$addLayer(geom)
eec_print(geom)

# Feature
feature <- ee$Feature(geom, list(rgee = "ee_print", data = TRUE))
eec_print(feature)

# FeatureCollection
featurecollection <- ee$FeatureCollection(feature)
eec_print(featurecollection)

# Image
srtm <- ee$Image("CGIAR/SRTM90_V4")
eec_print(srtm)

srtm_clip <- ee$Image("CGIAR/SRTM90_V4")$clip(geom)
srtm_metadata <- ee$print(srtm_clip)
srtm_metadata$imag_bands_names

# ImageCollection
object <- ee$ImageCollection("LANDSAT/LC08/C01/T1_TOA")$filter(ee$Filter()$eq("WRS_PATH", 44))$filter(ee$Filter()$eq("WRS_ROW", 34))$filterDate("2014-03-01", "2014-08-01")$aside(eec_print)

## End(Not run)
Arguments

- `collection`: The feature collection to be exported.
- `description`: Human-readable name of the task.
- `assetId`: The destination asset ID. **kwargs:** Holds other keyword arguments that may have been deprecated.
- `overwrite`: Logical. If TRUE, the assetId will be overwritten if it exists.

Value

An unstarted Task that exports the table to Earth Engine Asset.

See Also

Other vector export task creator: `ee_table_to_drive()`, `ee_table_to_gcs()`

Examples

```r
## Not run:
library(rgee)
library(stars)
library(sf)
ee_users()
ee_Initialize()

# Define study area (local -> earth engine)
# Communal Reserve Amarakaeri - Peru
rlist <- list(xmin = -71.13, xmax = -70.95, ymin = -12.89, ymax = -12.73)
ROI <- c(rlist$xmin, rlist$ymin,
    rlist$xmax, rlist$ymin,
    rlist$xmax, rlist$ymax,
    rlist$xmin, rlist$ymax,
    rlist$xmin, rlist$ymin)

ee_ROI <- matrix(ROI, ncol = 2, byrow = TRUE) %>%
    list() %>%
    st_polygon() %>%
    st_sfc() %>%
    st_set_crs(4326) %>%
    sf_as_ee()

amk_fc <- ee$FeatureCollection(  
    list(ee$Feature(ee_ROI, list(name = "Amarakaeri")))
 )

assetid <- paste0(ee_get_assethome(), '/geom_Amarakaeri')
task_vector <- ee_table_to_asset(  
    collection = amk_fc,  
    overwrite = TRUE,  
    assetId = assetid
 )
```
ee_table_to_drive

```
task_vector$start()
ee_monitoring(task_vector) # optional

ee_fc <- ee$FeatureCollection(assetid)
Map$centerObject(ee_fc)
Map$addLayer(ee_fc)

## End(Not run)
```

### ee_table_to_drive

*Creates a task to export a FeatureCollection to Google Drive.*

---

**Description**

Creates a task to export a FeatureCollection to Google Drive. This function is a wrapper around `ee$batch$Export$table$toDrive(...)`. 

**Usage**

```
ee_table_to_drive(
  collection, 
  description = "myExportTableTask", 
  folder = "rgee_backup", 
  fileNamePrefix = NULL, 
  timePrefix = TRUE, 
  fileFormat = NULL, 
  selectors = NULL
)
```

**Arguments**

- `collection`: The feature collection to be exported.
- `description`: Human-readable name of the task.
- `folder`: The name of a unique folder in your Drive account to export into. Defaults to the root of the drive.
- `fileNamePrefix`: The Google Drive filename for the export. Defaults to the name of the task.
- `timePrefix`: Add current date and time as a prefix to files to export.
- `fileFormat`: The output format: "CSV" (default), "GeoJSON", "KML", "KMZ", "SHP", or "TFRecord".
- `selectors`: The list of properties to include in the output, as a list of strings or a comma-separated string. By default, all properties are included. **kwargs: Holds other keyword arguments that may have been deprecated such as 'driveFolder' and 'driveFileNamePrefix'.

**Value**

An unstarted Task that exports the table to Google Drive.
See Also

Other vector export task creator: `ee_table_to_asset()`, `ee_table_to_gcs()`

Examples

```r
## Not run:
library(rgee)
library(stars)
library(sf)

ee_users()
ee_Initialize(drive = TRUE)

# Define study area (local -> earth engine)
# Communal Reserve Amarakaeri - Peru
rlist <- list(xmin = -71.13, xmax = -70.95, ymin = -12.89, ymax = -12.73)
ROI <- c(rlist$xmin, rlist$ymin,
         rlist$xmax, rlist$ymin,
         rlist$xmax, rlist$ymax,
         rlist$xmin, rlist$ymin,
         rlist$xmin, rlist$ymax)

ee_ROI <- matrix(ROI, ncol = 2, byrow = TRUE) %>%
  list() %>%
  st_polygon() %>%
  st_sfc() %>%
  st_set_crs(4326) %>%
  sf_as_ee()

amk_fc <- ee$FeatureCollection(
  list(ee$Feature(ee_ROI, list(name = "Amarakaeri")))
)

task_vector <- ee_table_to_drive(
  collection = amk_fc,
  fileFormat = "GEO_JSON",
  fileNamePrefix = "geom_Amarakaeri"
)
task_vector$start() # optional
ee_drive_to_local(task = task_vector)

## End(Not run)
```

---

**ee_table_to_gcs**  

Creates a task to export a FeatureCollection to Google Cloud Storage.

**Description**

Creates a task to export a FeatureCollection to Google Cloud Storage. This function is a wrapper around `ee$batch$Export$table$toCloudStorage(...)`. 
ee_table_to_gcs

Usage

```r
eee_table_to_gcs(
  collection,
  description = "myExportTableTask",
  bucket = NULL,
  fileNamePrefix = NULL,
  timePrefix = TRUE,
  fileFormat = NULL,
  selectors = NULL
)
```

Arguments

- **collection**: The feature collection to be exported.
- **description**: Human-readable name of the task.
- **bucket**: The name of a Cloud Storage bucket for the export.
- **fileNamePrefix**: Cloud Storage object name prefix for the export. Defaults to the name of the task.
- **timePrefix**: Add current date and time as a prefix to files to export.
- **fileFormat**: The output format: "CSV" (default), "GeoJSON", "KML", "KMZ", "SHP", or "TFRecord".
- **selectors**: The list of properties to include in the output, as a list of strings or a comma-separated string. By default, all properties are included. **kwargs: Holds other keyword arguments that may have been deprecated such as 'outputBucket'.

Value

An unstarted Task that exports the table to Google Cloud Storage.

See Also

Other vector export task creator: `ee_table_to_asset()`, `ee_table_to_drive()`

Examples

```r
## Not run:
library(rgee)
library(stars)
library(sf)
eee_users()
eee_Initialize(gcs = TRUE)

# Define study area (local -> earth engine)
# Communal Reserve Amarakaeri - Peru
rlist <- list(xmin = -71.13, xmax = -70.95, ymin = -12.89, ymax = -12.73)
ROI <- c(rlist$xmin, rlist$ymin,
  rlistxmax, rlist$ymin,
```
rlistxmax, rlistymax,
rlistxmin, rlistymax,
rlistxmin, rlistymin)
ee_ROI <- matrix(ROI, ncol = 2, byrow = TRUE) %>%
  list() %>%
  st_polygon() %>%
  st_sfc() %>%
  st_set_crs(4326) %>%
  sf_as_ee()

amk_fc <- ee$FeatureCollection(
  list(ee$Feature(ee_ROI, list(name = "Amarakaeri")))
)

task_vector <- ee_table_to_gcs(
  collection = amk_fc,
  bucket = "rgee_dev",
  fileFormat = "SHP",
  fileNamePrefix = "geom_Amarakaeri"
)
task_vector$start()
ee_monitoring(task_vector) # optional

amk_geom <- ee_gcs_to_local(task = task_vector)
plot(sf::read_sf(amk_geom[3]), border = "red", lwd = 10)

## End(Not run)

---

### ee_users

**Display the credentials of all users as a table**

**Description**

Display Earth Engine, Google Drive, and Google Cloud Storage Credentials as a table.

**Usage**

```r
ee_users(quiet = FALSE)
```

**Arguments**

- `quiet` Logical. Suppress info messages.

**Value**

A data.frame with credential information of all users.

**See Also**

Other session management functions: `ee_Initialize()`, `ee_user_info()`, `ee_version()`
ee_user_info

Examples

## Not run:
library(rgee)
ee_users()

## End(Not run)

---

**ee_user_info**

*Display the credentials and general info of the initialized user*

**Description**

Display the credentials and general info of the initialized user

**Usage**

```
ee_user_info(quiet = FALSE)
```

**Arguments**

- `quiet` Logical. Suppress info messages.

**Value**

A list with information about the Earth Engine user.

**See Also**

Other session management functions: `ee_Initialize()`, `ee_users()`, `ee_version()`

**Examples**

## Not run:
library(rgee)
ee_Initialize()
ee_user_info()

## End(Not run)
**ee_utils_create_json**  
*Convert a R list into a JSON file*

**Description**
Convert a R list into a JSON file

**Usage**
```r
ee_utils_create_json(x)
```

**Arguments**
- `x`  
  List to convert into a JSON file.

**Value**
A JSON file saved in a /tmp dir.

**Examples**
```r
## Not run:
library(rgee)
ee_utils_create_json(list(a=10,b=10))
## End(Not run)
```

---

**ee_utils_create_manifest_image**  
*Create a manifest to upload an image*

**Description**
Create a manifest to upload a GeoTIFF to Earth Engine assets folder. The "manifest" is simply a JSON file which describe all the upload parameters. See [https://developers.google.com/earth-engine/guides/image_manifest](https://developers.google.com/earth-engine/guides/image_manifest) to get more details.

**Usage**
```r
ee_utils_create_manifest_image(
  gs_uri,
  assetId,
  properties = NULL,
  start_time = "1970-01-01",
  end_time = "1970-01-01",
  pyramiding_policy = "MEAN",
)```
Arguments

- `gs_uri` Character. GCS full path of the image to upload to Earth Engine assets e.g. `gs://rgee_dev/l8.tif`
- `assetId` Character. How to call the file once uploaded to the Earth Engine Asset. e.g. `users/datacolec/fbf/l8`.
- `properties` List. Set of parameters to be set up as properties of the EE object.
- `start_time` Character. Sets the start time property (system:time_start) It could be a number (timestamp) or a date.
- `end_time` Character. Sets the end time property (system:time_end) It could be a number (timestamp) or a date.
- `pyramiding_policy` Character. The pyramid reduction policy to use.
- `returnList` Logical. If TRUE will return the "manifest" as a list otherwise will return a JSON file.
- `quiet` Logical. Suppress info message.

Value

If `returnList` is TRUE a list otherwise a JSON file.

See Also

Other generic upload functions: `ee_utils_create_manifest_table()`, `local_to_gcs()`

Examples

```r
eevalute(quote(library(rgee)));
eevaluate(quote(ee.Initialize()));

tif <- system.file("tif/L7_ETMs.tif", package = "stars")

# Return a JSON file
ee_utils_create_manifest_image(
  gs_uri = "gs://rgee_dev/l8.tif",
  assetId = "users/datacolec/fbf/l8"
)

# Return a list
ee_utils_create_manifest_image(
  gs_uri = "gs://rgee_dev/l8.tif",
  assetId = "users/datacolec/fbf/l8",
  returnList = TRUE
)
```
### ee_utils_create_manifest_table

Create a manifest to upload a table

#### Description

Create a manifest to upload a zipped shapefile to Earth Engine assets folder. The "manifest" is simply a JSON file which describe all the upload parameters. See [https://developers.google.com/earth-engine/guides/image_manifest](https://developers.google.com/earth-engine/guides/image_manifest) to get more details.

#### Usage

```r
ee_utils_create_manifest_table(
  gs_uri,
  assetId,
  start_time = "1970-01-01",
  end_time = "1970-01-01",
  properties = NULL,
  returnList = FALSE,
  quiet = FALSE
)
```

#### Arguments

- **gs_uri**: Character. GCS full path of the table to upload to Earth Engine assets e.g. `gs://rgee_dev/nc.zip`
- **assetId**: Character. How to call the file once uploaded to the Earth Engine Asset. e.g. `users/datacolecfbf/nc`.
- **start_time**: Character. Sets the start time property (system:time_start). It could be a number (timestamp) or a date.
- **end_time**: Character. Sets the end time property (system:time_end). It could be a number (timestamp) or a date.
- **properties**: List. Set of parameters to be set up as properties of the EE object.
- **returnList**: Logical. If TRUE will return the "manifest" as a list otherwise will return a JSON file.
- **quiet**: Logical. Suppress info message.

#### Value

If `returnList` is TRUE a list otherwise a JSON file.

#### See Also

Other generic upload functions: `ee_utils_create_manifest_image()`, `local_to_gcs()`
## Examples

```r
## Not run:
library(rgee)
library(sf)
eed.Initialize(gcs = TRUE)

x <- st_read(system.file("shape/nc.shp", package = "sf"))
shp_dir <- sprintf("%s.shp", tempfile())
geozip_dir <- ee_utils_shp_to_zip(x, shp_dir)

# Return a JSON file
manifest <- ee_utils_create_manifest_table(
  gs_uri = "gs://rgee_dev/nc.zip",
  assetId = "users/datacolecfbf/nc"
)

# Return a list
ee_utils_create_manifest_table(
  gs_uri = "gs://rgee_dev/nc.zip",
  assetId = "users/datacolecfbf/nc",
  returnList = TRUE
)

## End(Not run)
```

---

### ee_utils_future_value

*The value of a future or the values of all elements in a container*

#### Description

Gets the value of a future or the values of all elements (including futures) in a container such as a list, an environment, or a list environment. If one or more futures is unresolved, then this function blocks until all queried futures are resolved.

#### Usage

```r
ee_utils_future_value(future, stdout = TRUE, signal = TRUE, ...)
```

#### Arguments

- **future**: A Future, an environment, a list, or a list environment.
- **stdout**: If TRUE, standard output captured while resolving futures is relayed, otherwise not.
- **signal**: If TRUE, conditions captured while resolving futures are relayed, otherwise not.
- **...**: All arguments used by the S3 methods.
**Value**

value() of a Future object returns the value of the future, which can be any type of R object.

value() of a list, an environment, or a list environment returns an object with the same number of elements and of the same class. Names and dimension attributes are preserved, if available. All future elements are replaced by their corresponding value() values. For all other elements, the existing object is kept as-is.

If `signal` is `TRUE` and one of the futures produces an error, then that error is produced.

**Author(s)**

Henrik Bengtsson [https://github.com/HenrikBengtsson/](https://github.com/HenrikBengtsson/)

---

**ee_utils_get_crs**

*Convert EPSG, ESRI or SR-ORG code into a OGC WKT*

**Description**

Convert EPSG, ESRI or SR-ORG code into a OGC WKT

**Usage**

`ee_utils_get_crs(code)`

**Arguments**

- `code` The projection code.

**Value**

A character which represents the same projection in WKT2 string.

**Examples**

```r
## Not run:
library(rgee)

ee_utils_get_crs("SR-ORG:6864")  
ee_utils_get_crs("EPSG:4326")  
ee_utils_get_crs("ESRI:37002")
```

## End(Not run)
Description

Add text to a GIF (magick-image object). This function is a wrapper around image_annotate.

Usage

```
ee_utils_gif_annotate(
    image,
    text,
    gravity = "northwest",
    location = "+0+0",
    degrees = 0,
    size = 20,
    font = "sans",
    style = "normal",
    weight = 400,
    kerning = 0,
    decoration = NULL,
    color = NULL,
    strokecolor = NULL,
    boxcolor = NULL
)
```

Arguments

- **image** magick image object returned by `magick::image_read()` or `magick::image_graph()`
- **text** character vector of length equal to `image` or length 1
- **gravity** string with gravity value from `gravity_types`
- **location** geometry string with location relative to gravity
- **degrees** rotates text around center point
- **size** font-size in pixels
- **font** string with font family such as "sans", "mono", "serif", "Times", "Helvetica", "Trebuchet", "Georgia", "Palatino" or "Comic Sans"
- **style** value of `style_types` for example "italic"
- **weight** thickness of the font, 400 is normal and 700 is bold
- **kerning** increases or decreases whitespace between letters
- **decoration** value of `decoration_types` for example "underline"
- **color** a valid color string such as "navyblue" or "#000080". Use "none" for transparency
- **strokecolor** a color string adds a stroke (border around the text)
- **boxcolor** a color string for background color that annotation text is rendered on
Value

A magick-image object

Author(s)

Jeroen Ooms

See Also

Other GIF functions: `ee_utils_gif_creator()`, `ee_utils_gif_save()`

Examples

```r
## Not run:
library(rgge)

ee_Initialize()

col <- ee$ImageCollection("JRC/GSW1_1/YearlyHistory")$map(function(img) {
  year <- img$getDate()$get("year")
  yearImg <- img$gte(2)$multiply(year)
  despeckle <- yearImg$connectedPixelCount(15, TRUE)$eq(15)
  yearImg$updateMask(despeckle)$selfMask()$set("year", year)
})

appendReverse <- function(col) col$merge(col$sort("year", FALSE))

# -----------------------------------
# 1 Basic Animation - Ucayali Peru
# -----------------------------------

bgColor = "FFFFFF" # Assign white to background pixels.
riverColor = "0D0887" # Assign blue to river pixels.

## 1.1 Create the dataset
annualCol = col$map(function(img) {
  img$unmask(0)@
  visualize(min = 0, max = 1, palette = c(bgColor, riverColor))@$
  set("year", img$get("year"))
})

basicAnimation <- appendReverse(annualCol)

## 1.2 Set video arguments
aoi <- ee$Geometry$Rectangle(-74.327, -10.087, -73.931, -9.327)
videoArgs = list(
  dimensions = 600, # Max dimension (pixels), min dimension is proportionally scaled.
  region = aoi,
  framesPerSecond = 10
)

## 1.2 Download, display and save the GIF!
```
animation <- ee_utils_gif_creator(basicAnimation, videoArgs, mode = "wb")
get_years <- basicAnimation$aggregate_array("year")$getInfo()
animation %>%
ee_utils_gif_annotate("Ucayali, Peru") %>%
ee_utils_gif_annotate(get_years, size = 15, location = "+90+40",
   boxcolor = "#FFFFFF") %>%
ee_utils_gif_annotate("created using {magick} + {rgee},
   size = 15, font = "sans",location = "+70+20")
animation_wtxt
gc(reset = TRUE)
ee_utils_gif_save(animation_wtxt, path = paste0(tempfile(), ".gif"))

## End(Not run)

### ee_utils_gif_creator

Create a GIF from an Earth Engine ImageCollection

**Description**

Create an GIF (as a magick-image object) from a EE ImageCollection. Note: Animations can only be created when ImageCollections is composed by RGB or RGBA image. This can be done by mapping a visualization function onto an ImageCollection (e.g. `ic$map(function(img)
   img$visualize(...))`) or specifying three bands in parameters argument (See examples). `ee_utils_gif_creator` is a wrapper around `ee$ImageCollection$getVideoThumbURL`.

**Usage**

```r
ee_utils_gif_creator(ic, parameters, quiet = FALSE, ...)
```

**Arguments**

- `ic`: An ee$ImageCollection.
- `parameters`: List of parameters for visualization and animation. See details.
- `quiet`: Logical. Suppress info message.
- `...`: parameter(s) passed on to `download.file`

**Details**

The parameters argument is identical to `visParams` (See `rgee::Map$addLayer`), plus, optionally:

- **dimensions**: A number or pair of numbers in format c(WIDTH,HEIGHT). Max dimensions of the thumbnail to render, in pixels. If only one number is passed, it is used as the maximum, and the other dimension is computed by proportional scaling.
- **crs**: A CRS string specifying the projection of the output.
- **crs_transform**: The affine transform to use for the output pixel grid.
- **scale**: A scale to determine the output pixel grid; ignored if both crs and crs_transform are specified.
• **region**: ee$Geometry$Polygon, GeoJSON or c(E,S,W,N). Geospatial region of the result. By default, the whole image.

• **format**: String. The output format (only 'gif' is currently supported).

• **framesPerSecond**: String. Animation speed.

**Value**

A magick-image object of the specified ImageCollection.

**Author(s)**

Jeroen Ooms

**See Also**

Other GIF functions: **ee_utils_gif_annotate()**, **ee_utils_gif_save()**

**Examples**

```r
eelub_initialize()

col <- ee$ImageCollection("JRC/GSW1_1/YearlyHistory")$map(function(img) {
  year <- img$date()$get("year")
  yearImg <- img$gte(2)$multiply(year)
  despeckle <- yearImg$connectedPixelCount(15, TRUE)$eq(15)
  yearImg$updateMask(despeckle)$selfMask()$set("year", year)
})

appendReverse <- function(col) col$merge(col$sort('year', FALSE))

# 1 Basic Animation - Ucayali Peru

bgColor = "FFFFFF" # Assign white to background pixels.
riverColor = "0000FF" # Assign blue to river pixels.

## 1.1 Create the dataset
annualCol = col$map(function(img) {
  img$unmask(0)@
  visualize(min = 0, max = 1, palette = c(bgColor, riverColor))@
  set("year", img$get("year"))
})

basicAnimation <- appendReverse(annualCol)

## 1.2 Set video arguments
aoi <- ee$Geometry$Rectangle(-74.327, -10.087, -73.931, -9.327)
```
videoArgs = list(
  dimensions = 600, # Max dimension (pixels), min dimension is proportionally scaled.
  region = aoi,
  framesPerSecond = 10
)

## 1.2 Download, display and save the GIF!
animation <- ee_utils_gif_creator(basicAnimation, videoArgs, mode = "wb")
get_years <- basicAnimation$aggregate_array("year")$getInfo()
animation %>%
  ee_utils_gif_annotate("Ucayali, Peru") %>%
  ee_utils_gif_annotate(get_years, size = 15, location = "+90+40",
                        boxcolor = "#FFFFFF") %>%
  ee_utils_gif_annotate("created using {magick} + {rgee}",
                        size = 15, font = "sans", location = "+70+20") -> animation_wtxt
gc(reset = TRUE)
ee_utils_gif_save(animation_wtxt, path = paste0(tempfile(), ".gif"))

## End(Not run)

eee_utils_gif_save  Write a GIF

Description

Write a magick-image object as a GIF file using magick package. This function is a wrapper around
image_write.

Usage

```r
ee_utils_gif_save(
  image,
  path = NULL,
  format = NULL,
  quality = NULL,
  depth = NULL,
  density = NULL,
  comment = NULL,
  flatten = FALSE
)
```

Arguments

- `image`: magick image object returned by `image_read`.
- `path`: path a file, url, or raster object or bitmap array.
- `format`: output format such as "png", "jpeg", "gif", "rgb" or "rgba".
- `quality`: number between 0 and 100 for jpeg quality. Defaults to 75.
depth  
color depth (either 8 or 16).
density  
resolution to render pdf or svg.
comment  
text string added to the image metadata for supported formats.
flatten  
should image be flattened before writing? This also replaces transparency with background color.

Value
No return value, called to write a GIF file.

Author(s)
Jeroen Ooms

See Also
Other GIF functions: `ee_utils_gif_annotate()` , `ee_utils_gif_creator()` 

Examples
```r
## Not run:
library(rgee)
eelInitialize()

col <- ee$ImageCollection("JRC/GSW1_1/YearlyHistory")$map(function(img) {
  year <- img$date()$get("year")
  yearImg <- img$gte(2)$multiply(year)
  despeckle <- yearImg$connectedPixelCount(15, TRUE)$eq(15)
  yearImg$updateMask(despeckle)$selfMask()$set("year", year)
})

appendReverse <- function(col) col$merge(col$sort("Var_year", FALSE))

# -----------------------------------
# 1 Basic Animation - Ucayali Peru
# -----------------------------------

bgColor = "FFFFFF" # Assign white to background pixels.
riverColor = "0D0887" # Assign blue to river pixels.

## 1.1 Create the dataset
annualCol = col$map(function(img) {
  img$unmask(0)$
  visualize(min = 0, max = 1, palette = c(bgColor, riverColor))$
  set("year", img$get("year"))
})
basicAnimation <- appendReverse(annualCol)

## 1.2 Set video arguments
aoi <- ee$Geometry$Rectangle(-74.327, -10.087, -73.931, -9.327)
videoArgs = list(
  dimensions = 600, # Max dimension (pixels), min dimension is proportionally scaled.
  region = aoi,
  framesPerSecond = 10
)

## 1.2 Download, display and save the GIF!
animation <- ee_utils_gif_creator(basicAnimation, videoArgs, mode = "wb")
get_years <- basicAnimation$aggregate_array("year")$getInfo()
animation %>%
  ee_utils_gif_annotate("Ucayali, Peru") %>%
  ee_utils_gif_annotate(get_years, size = 15, location = "+90+40",
                        boxcolor = "#FFFFFF") %>%
  ee_utils_gif_annotate("created using {magick} + {rgee}",
                        size = 15, font = "sans", location = "+70+20") ->
animation_wtxt
gc(reset = TRUE)
ee_utils_gif_save(animation_wtxt, path = paste0(tempfile(), ".gif"))

## End(Not run)

---

**ee_utils_pyfunc**  

Wrap an R function in a Python function with the same signature.

### Description

This function could wrap an R function in a Python function with the same signature. Note that the signature of the R function must not contain esoteric Python-incompatible constructs.

### Usage

```
ee_utils_pyfunc(f)
```

### Arguments

- **f**  
  An R function

### Value

A Python function that calls the R function `f` with the same signature.

### Note

`py_func` has been renamed to `ee_utils_pyfunc` just to maintain the rgee functions name’s style. All recognition for this function must always be given to **reticulate**.

### Author(s)

Yuan Tang and J.J. Allaire
ee_utils_py_to_r

Convert between Python and R objects

Description

Convert between Python and R objects

Usage

```
ee_utils_py_to_r(x)
```

Arguments

- `x`: A python object
ee_utils_search_display

Search into the Earth Engine Data Catalog

Description

Search into the Earth Engine Data Catalog

Usage

ee_utils_search_display(ee_search_dataset)

Arguments

ee_search_dataset
    character which represents the EE dataset ID.

Value

No return value, called for displaying the Earth Engine dataset in the browser.

Examples

## Not run:
library(rgge)

ee_datasets <- c("WWF/HydroSHEDS/15DIR", "WWF/HydroSHEDS/03DIR")
eu_utils_search_display(ee_datasets)

## End(Not run)
ee_utils_shp_to_zip

Create a zip file from a sf object

Description

Create a zip file from a sf object

Usage

```
ee_utils_shp_to_zip(
  x, 
  filename, 
  SHP_EXTENSIONS = c("dbf", "prj", "shp", "shx")
)
```

Arguments

- `x`: sf object
- `filename`: data source name
- `SHP_EXTENSIONS`: file extension of the files to save into the zip file. By default: "dbf", "prj", "shp", "shx".

Value

Character. The full path of the zip file created.

See Also

Other ee_utils functions: `ee_utils_py_to_r()`, `ee_utils_pyfunc()`

Examples

```
## Not run:
library(rgee)
library(sf)
ee_Initialize(gcs = TRUE)

# Create sf object
nc <- st_read(system.file("shape/nc.shp", package="sf"))
zipfile <- ee_utils_shp_to_zip(nc)

## End(Not run)
```
ee_version

<table>
<thead>
<tr>
<th>Description</th>
<th>Earth Engine API version</th>
</tr>
</thead>
</table>

**Usage**

```
ee_version()
```

**Value**

Character. Earth Engine Python API version used to build rgee.

**See Also**

Other session management functions: `ee_Initiaize()`, `ee_user_info()`, `ee_users()`

gcs_to_ee_image

<table>
<thead>
<tr>
<th>Description</th>
<th>Move a GeoTIFF image from GCS to their EE assets</th>
</tr>
</thead>
</table>

**Usage**

```
gcs_to_ee_image(
  manifest,
  overwrite = FALSE,
  command_line_tool_path = NULL,
  quiet = FALSE
)
```

**Arguments**

- `manifest` Character. manifest upload file. See [ee_utils_create_manifest_image](#).
- `overwrite` Logical. If TRUE, the assetId will be overwritten if it exists.
- `command_line_tool_path` Character. Path to the Earth Engine command line tool (CLT). If NULL, rgee assumes that CLT is set in the system PATH. (ignore if via is not defined as "gcs_to_asset").
- `quiet` Logical. Suppress info message.
Value

Character. The Earth Engine asset ID.

Examples

```r
## Not run:
library(rgee)
library(stars)
ee_Initialize("csaybar", gcs = TRUE)

# 1. Read GeoTIFF file and create a output filename
tif <- system.file("tif/L7_ETMs.tif", package = "stars")
x <- read_stars(tif)
assetId <- sprintf("%s/%s",ee_get_assethome(),'stars_l7')

# # 2. From local to gcs
# gs_uri <- local_to_gcs(
# # x = tif,
# # bucket = 'rgee_dev' # Insert your own bucket here!
# # )
# #
# # 3. Create an Image Manifest
# manifest <- ee_utils_create_manifest_image(gs_uri, assetId)
# #
# # 4. From GCS to Earth Engine
# gcs_to_ee_image(
# # manifest = manifest,
# # overwrite = TRUE
# # )
# #
# # OPTIONAL: Monitoring progress
# ee_monitoring()
# #
# # OPTIONAL: Display results
# ee_stars_01 <- ee$Image(assetId)
# # ee_stars_01$bandNames()$getInfo()
# Map$centerObject(ee_stars_01)
# Map$addLayer(ee_stars_01, list(min = 0, max = 255, bands = c("b3", "b2", "b1")))
## End(Not run)
```

---

**gcs_to_ee_table**

Move a zipped shapefile from GCS to their EE Assets

**Description**

Move a zipped shapefile from GCS to their EE Assets
Usage

gcs_to_ee_table(
  manifest,
  command_line_tool_path = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

Arguments

manifest Character. manifest upload file. See \texttt{ee_utils_create_manifest_table}.
command_line_tool_path Character. Path to the Earth Engine command line tool (CLT). If NULL, rgee assumes that CLT is set in the system PATH. (ignore if via is not defined as "gcs\_to\_asset").
overwrite Logical. If TRUE, the assetId will be overwritten if it exists.
quiet Logical. Suppress info message.

Value

Character. The Earth Engine asset ID.

Examples

## Not run:
library(rgee)
library(sf)
ee\_Initialize(gcs = TRUE)

# 1. Read dataset and create a output filename
x <- st\_read(system\_file(\"shape/nc.shp\", package = \"sf\"))
assetId <- sprintf("%s/%s", ee\_get\_assethome(), \'toy\_poly\_gcs\')

# 2. From sf to .shp
shp\_dir <- sprintf("%s.shp", tempfile())
geozip\_dir <- ee\_utils\_shp\_to\_zip(x, shp\_dir)

# 3. From local to gcs
gcs\_filename <- local\_to\_gcs(
  x = geozip\_dir,
  bucket = \"rgee\_dev\" # Insert your own bucket here!
)

# 4. Create Table Manifest
manifest <- ee\_utils\_create\_manifest\_table(
  gs\_uri = gcs\_filename,
  assetId = assetId
)

# 5. From GCS to Earth Engine
local_to_gcs

Upload local files to Google Cloud Storage

Description

Upload images or tables to Google Cloud Storage

Usage

local_to_gcs(x, bucket = NULL, quiet = FALSE)

Arguments

x Character. filename.
bucket bucket name you are uploading to
quiet Logical. Suppress info message.

Value

Character which represents the full path of the object in the GCS bucket specified.

See Also

Other generic upload functions: *ee_utils_create_manifest_image()*,*ee_utils_create_manifest_table()*

Examples

```r
## Not run:
library(rgee)
library(stars)

# Initialize a specific Earth Engine account and
# Google Cloud Storage credentials
ee_Initialize(gcs = TRUE)

# # Define an image.
tif <- system.file("tif/L7_ETMs.tif", package = "stars")
local_to_gcs(x = tif, bucket = 'rgee_dev')

## End(Not run)
```
Module to display Earth Engine (EE) spatial objects

Description

Create interactive visualizations of spatial EE objects (ee$Geometry, ee$Image, ee$Feature, and ee$FeatureCollection) using mapview.

Usage

Map

Format

An object of class environment with the following functions:

- **addLayer(eeObject, visParams, name = NULL, shown = TRUE, opacity = 1, legend = FALSE)**: Adds a given EE object to the map as a layer.
  
  - **eeObject**: The object to add to mapview.
  
  - **visParams**: List of parameters for visualization. See details.
  
  - **name**: The name of the layer.
  
  - **shown**: A flag indicating whether the layer should be on by default.
  
  - **opacity**: The layer’s opacity represented as a number between 0 and 1. Defaults to 1.
  
  - **legend**: Should a legend be plotted?. Ignore if eeObject is not a single-band ee$Image.

- **addLayers(eeObject, visParams, name = NULL, shown = TRUE, opacity = 1, legend = FALSE)**: Adds a given ee$ImageCollection to the map as multiple layers.
  
  - **eeObject**: The ee$ImageCollection to add to mapview.
  
  - **visParams**: List of parameters for visualization. See details.
  
  - **name**: The name of layers.
  
  - **shown**: A flag indicating whether layers should be on by default.
  
  - **opacity**: The layer’s opacity represented as a number between 0 and 1. Defaults to 1.
  
  - **legend**: Should a legend be plotted?. Only the legend of the first image is displayed.
• **setCenter**(lon = 0, lat = 0, zoom = NULL): Centers the map view at the given coordinates with the given zoom level. If no zoom level is provided, it uses 1 by default.
  
  – **lon**: The longitude of the center, in degrees.
  
  – **lat**: The latitude of the center, in degrees.
  
  – **zoom**: The zoom level, from 1 to 24.

• **setZoom**(zoom = NULL): Sets the zoom level of the map.
  
  – **zoom**: The zoom level, from 1 to 24.

• **centerObject**(eeObject, zoom = NULL, maxError = ee$ErrorMargin(1)): Centers the map view on a given object. If no zoom level is provided, it will be predicted according to the bounds of the Earth Engine object specified.
  
  – **eeObject**: EE object.
  
  – **zoom**: The zoom level, from 1 to 24.
  
  – **maxError**: Max error when input image must be reprojected to an explicitly requested result projection or geodesic state.

Details

Map use the Earth Engine method `getMapId` to fetch and return an ID dictionary being used to create layers in a mapview object. Users can specify visualization parameters to `Map$addLayer` by using the `visParams` argument. Each Earth Engine spatial object has a specific format. For `ee$Image`, the parameters available are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bands</td>
<td>Comma-delimited list of three band names to be mapped to RGB</td>
<td>list</td>
</tr>
<tr>
<td>min</td>
<td>Value(s) to map to 0</td>
<td>number or list of three numbers, one for each band</td>
</tr>
<tr>
<td>max</td>
<td>Value(s) to map to 1</td>
<td>number or list of three numbers, one for each band</td>
</tr>
<tr>
<td>gain</td>
<td>Value(s) by which to multiply each pixel value</td>
<td>number or list of three numbers, one for each band</td>
</tr>
<tr>
<td>bias</td>
<td>Value(s) to add to each Digital Number (DN) value</td>
<td>number or list of three numbers, one for each band</td>
</tr>
<tr>
<td>gamma</td>
<td>Gamma correction factor(s)</td>
<td>number or list of three numbers, one for each band</td>
</tr>
<tr>
<td>palette</td>
<td>List of CSS-style color strings (single-band images only)</td>
<td>comma-separated list of hex strings</td>
</tr>
<tr>
<td>opacity</td>
<td>The opacity of the layer (0.0 is fully transparent and 1.0 is fully opaque)</td>
<td>number</td>
</tr>
</tbody>
</table>

If you add an `ee$Image` to `Map$addLayer` without any additional parameters, by default it assigns the first three bands to red, green, and blue bands, respectively. The default stretch is based on the min-max range. On the other hand, the available parameters for `ee$Geometry`, `ee$Feature`, and `ee$FeatureCollection` are:

- **color**: A hex string in the format RRGGBB specifying the color to use for drawing the features. By default #000000.
- **pointRadius**: The radius of the point markers. By default 3.
- **strokeWidth**: The width of lines and polygon borders. By default 3.
Value

Object of class leaflet, with the following extra parameters: tokens, name, opacity, shown, min, max, palette, and legend. Use the $ method to retrieve the data (e.g. m$rgee$min).

Examples

```r
## Not run:
library(geo)
library(sf)
eeInitialize()

# Case 1: Geometry
geom1 <- ee$Geometry$Point(list(-73.53, -15.75))
Map$centerObject(geom1, zoom = 8)
m1 <- Map$addLayer(
    eeObject = geom1,
    visParams = list(
        pointRadius = 10,
        color = "FF0000"
    ),
    name = "Geometry-Arequipa"
)

# Case 2: Feature
feature_arq <- ee$Feature(ee$Geometry$Point(list(-72.53, -15.75)))
m2 <- Map$addLayer(
    eeObject = feature_arq,
    name = "Feature-Arequipa"
)
m2 + m1

# Case 4: Image
image <- ee$Image("LANDSAT/LC08/C01/T1/LC08_044034_20140318")
Map$centerObject(image)
m4 <- Map$addLayer(
    eeObject = image,
    visParams = list(
        bands = c("B4", "B3", "B2"),
        max = 10000
    ),
    name = "SF"
)

# Case 5: ImageCollection
nc <- st_read(system.file("shape/nc.shp", package = "sf")) %>%
    st_transform(4326) %>%
sf_as_ee()

ee_s2 <- ee$ImageCollection("COPERNICUS/S2")$filterDate("2016-01-01", "2016-01-31")$
filterBounds(nc) %>%
ee_get(0:4)
```
Map$centerObject(nc$geometry())
m5 <- Map$addLayers(ee_s2, legend = TRUE)
m5

# Case 6: Map comparison
image <- eeImage("LANDSAT/LC08/C01/T1/LC08_044034_20140318")
Map$centerObject(image)
m_ndvi <- Map$addLayer(
  eeObject = image$normalizedDifference(list("B5", "B4")),
  visParams = list(max = 0.7),
  name = "SF_NDVI",
  legend = TRUE
)
m6 <- m4 | m_ndvi
m6

# Case 7: digging up the metadata
m6$rgee$tokens
m5$rgee$tokens

## End(Not run)

null-default EarthEngineMap + EarthEngineMap; adds data from the second map to the first

Description

EarthEngineMap + EarthEngineMap; adds data from the second map to the first

EarthEngineMap | EarthEngineMap provides a slider in the middle to compare two maps.

Usage

## S3 method for class 'EarthEngineMap'
e1 + e2

## S3 method for class 'EarthEngineMap'
e1 | e2

Arguments

e1 an EarthEngineMap object.
e2 an EarthEngineMap object.

Author(s)

tim-salabim. Adapted from mapview code.
tim-salabim. Adapted from mapview code.
### print.ee.computedobject.ComputedObject

**Description**

print Earth Engine object

**Usage**

```r
## S3 method for class 'ee.computedobject.ComputedObject'
print(x, ..., type = getOption("rgee.print.option"))
```

**Arguments**

- `x`: Earth Engine spatial object.
- `...`: ignored
- `type`: Character. What to show about the x object?. Three options are supported: "json", "simply", "ee_print". By default "simply".

**Value**

No return value, called for displaying Earth Engine objects.

### raster_as_ee

**Description**

Convert a Raster* object into an EE Image object

**Usage**

```r
raster_as_ee(
  x,
  assetId,
  bucket = NULL,
  command_line_tool_path = NULL,
  overwrite = FALSE,
  monitoring = TRUE,
  quiet = FALSE,
  ...
)
```
Arguments

- **x**: RasterLayer, RasterStack or RasterBrick object to be converted into an ee$Image.
- **assetId**: Character. Destination asset ID for the uploaded file.
- **bucket**: Character. Name of the GCS bucket.
- **command_line_tool_path**: Character. Path to the Earth Engine command line tool (CLT). If NULL, rgee assumes that CLT is set in the system PATH. (ignore if via is not defined as "gcs_to_asset").
- **overwrite**: Logical. If TRUE, the assetId will be overwritten.
- **monitoring**: Logical. If TRUE the exportation task will be monitored.
- **quiet**: Logical. Suppress info message.
- **...**: parameter(s) passed on to `ee_utils_create_manifest_image`

Value

An ee$Image object

See Also

Other image upload functions: `stars_as_ee()`

Examples

```r
## Not run:
library(raster)
library(stars)
library(rgee)

eeInitialize(gcs = TRUE)

tif <- system.file("tif/L7_ETMs.tif", package = "stars")
x <- stack(tif)
assetid <- sprintf("%s/%s", ee_get_assethome(), 'raster_l7')

# Method 1
# 1. Move from local to gcs
gs_uri <- local_to_gcs(x = tif, bucket = 'rgee_dev')

# 2. Create a manifest
manifest <- ee_utils_create_manifest_image(gs_uri, assetId)

# 3. Pass from gcs to asset
gcs_to_ee_image(
  manifest = manifest,
  overwrite = TRUE
)

# OPTIONAL: Monitoring progress
```
ee_monitoring()

# OPTIONAL: Display results
ee_stars_01 <- ee$Image(assetId)
Map$centerObject(ee_stars_01)
Map$addLayer(ee_stars_01, list(min = 0, max = 255))

# Method 2
ee_stars_02 <- raster_as_ee(
  x = x,
  overwrite = TRUE,
  assetId = assetId,
  bucket = "rgee_dev"
)
Map$centerObject(ee_stars_02)
Map$addLayer(ee_stars_02, list(min = 0, max = 255))

## End(Not run)

---

**rdate_to_eedate**  
*Pass an R date object to Earth Engine*

**Description**

Pass an R date object ("Date", "Numeric", "character", "POSIXt", and "POSIXct") to Google Earth Engine (ee$Date).

**Usage**

```r
rdate_to_eedate(date, timestamp = FALSE)
```

**Arguments**

- **date**: R date object
- **timestamp**: Logical. If TRUE, return the date in milliseconds from the Unix Epoch (1970-01-01 00:00:00 UTC) otherwise return a EE date object. By default FALSE.

**Value**

`rdate_to_eedate` will return either a numeric timestamp or an ee$Date depending on the `timestamp` argument.

**See Also**

Other date functions: `ee_get_date_ic()`, `ee_get_date_img()`., `eedate_to_rdate()`
## sf_as_ee

### Description

Load an sf object to Earth Engine.

### Usage

```r
sf_as_ee(
  x,
  via = "getInfo",
  assetId = NULL,
  bucket = NULL,
  command_line_tool_path = NULL,
  overwrite = TRUE,
  monitoring = TRUE,
  proj = "EPSG:4326",
  evenOdd = TRUE,
  geodesic = NULL,
  quiet = FALSE,
  ...
)
```

### Arguments

- **x**
  - object of class sf, sfc or sfg.

- **via**
  - Character. Upload method for sf objects. Three methods are implemented: `getInfo`, `getInfo_to_asset` and `gcs_to_asset`. See details.

- **assetId**
  - Character. Destination asset ID for the uploaded file. Ignore if `via` argument is "getInfo".

- **bucket**
  - Character. Name of the bucket (GCS) to save intermediate files (ignore if `via` is not defined as "gcs_to_asset").

- **command_line_tool_path**
  - Character. Path to the Earth Engine command line tool (CLT). If NULL, rgee assumes that CLT is set in the system PATH. (ignore if `via` is not defined as "gcs_to_asset").

### Examples

```r
## Not run:
library(rgee)
ee_Initilize()
rdate_to_eedate('2000-01-01')
rdate_to_eedate(315532800000) # float number

## End(Not run)
```
overwrite A boolean argument which indicates whether "filename" should be overwritten. Ignore if via argument is "getInfo". By default TRUE.

monitoring Logical. Ignore if via is not set as getInfo_to_asset or gcs_to_asset. If TRUE the exportation task will be monitored.

proj Integer or character. Coordinate Reference System (CRS) for the EE object, defaults to "EPSG:4326" (x=longitude, y=latitude).

evenOdd Logical. Ignored if x is not a Polygon. If TRUE, polygon interiors will be determined by the even/odd rule, where a point is inside if it crosses an odd number of edges to reach a point at infinity. Otherwise polygons use the left-inside rule, where interiors are on the left side of the shell’s edges when walking the vertices in the given order. If unspecified, defaults to TRUE.

geodesic Logical. Ignored if x is not a Polygon or LineString. Whether line segments should be interpreted as spherical geodesics. If FALSE, indicates that line segments should be interpreted as planar lines in the specified CRS. If absent, defaults to TRUE if the CRS is geographic (including the default EPSG:4326), or to FALSE if the CRS is projected.

quiet Logical. Suppress info message.

... ee_utils_create_manifest_table arguments might be included.

Details

sf_as_ee supports the upload of sf objects by three different options: "getInfo" (default), "getInfo_to_asset", and "gcs_to_asset". getInfo transforms sf objects (sfg, sfc, or sf) to GeoJSON (using geojsonio::geojson_json) and then encrusted them in an HTTP request using the server-side objects that are implemented in the Earth Engine API (i.e. ee$Geometry$...). If the sf object is too large (~ >1Mb) is likely to cause bottlenecks since it is a temporary file that is not saved in your EE Assets (server-side). The second option implemented is 'getInfo_to_asset'. It is similar to the previous one, with the difference that after create the server-side object will save it in your Earth Engine Assets. For dealing with very large spatial objects is preferable to use the third option 'gcs_to_asset'. This option firstly saves the sf object as a *.shp file in the /temp directory. Secondly, using the function local_to_gcs will move the shapefile from local to Google Cloud Storage. Finally, using the function gcs_to_ee_table the ESRI shapefile will be loaded to their EE Assets. See Importing table data documentation for more details.

Value

When via is "getInfo" and x is either an sf or sfc object with multiple geometries will return an ee$FeatureCollection. For single sfc and sfg objects will return an ee$Geometry$....

If via is either "getInfo_to_asset" or "gcs_to_asset" always will return an ee$FeatureCollection.

Examples

```r
## Not run:
library(rgge)
library(sf)
ee_Initialize()
```
# 1. Handling geometry parameters

## Simple

```r
ee_x <- st_read(system.file("shape/nc.shp", package = "sf")) %>%
  sf_as_ee()
```

Map$centerObject(eeObject = ee_x)

Map$addLayer(ee_x)

```r
# Create a right-inside polygon.
toy_poly <- matrix(data = c(-35,-10,-35,10,35,10,35,-10,-35,-10),
  ncol = 2,
  byrow = TRUE) %>%
  list() %>%
  st_polygon()

holePoly <- sf_as_ee(x = toy_poly, evenOdd = FALSE)
```

```r
# Create an even-odd version of the polygon.
evenOddPoly <- sf_as_ee(toy_poly, evenOdd = TRUE)
```

```r
# Create a point to test the insideness of the polygon.
pt <- ee$Geometry$Point(c(1.5, 1.5))
```

```r
# Check insideness with a contains operator.
print(holePoly$contains(pt)$getInfo() %>% ee_utils_py_to_r())
print(evenOddPoly$contains(pt)$getInfo() %>% ee_utils_py_to_r())
```

## 2. Upload small geometries to EE asset

```r
assetId <- sprintf("%s/%s", ee_get_assethome(), 'toy_poly')
eex <- sf_as_ee(
  x = toy_poly,
  overwrite = TRUE,
  assetId = assetId,
  via = "getInfo_to_asset")
```

## 3. Upload large geometries to EE asset

```r
ee_Initialize(gcs = TRUE)
assetId <- sprintf("%s/%s", ee_get_assethome(), 'toy_poly_gcs')
eex <- sf_as_ee(
  x = toy_poly,
  overwrite = TRUE,
  assetId = assetId,
  bucket = 'rgee_dev',
  monitoring = FALSE,
  via = 'gcs_to_asset'
)
```

ee_monitoring()

## End(Not run)

---

**stars_as_ee**

Convert a stars or stars-proxy object into an EE Image object
Description

Convert a stars or stars-proxy object into an EE Image object

Usage

stars_as_ee(
  x,
  assetId,
  bucket = NULL,
  command_line_tool_path = NULL,
  overwrite = FALSE,
  monitoring = TRUE,
  quiet = FALSE,
  ...
)

Arguments

x         stars or stars-proxy object to be converted into an ee$Image.
assetId   Character. Destination asset ID for the uploaded file.
bucket    Character. Name of the GCS bucket.
command_line_tool_path Character. Path to the Earth Engine command line tool (CLT). If NULL, rgee assumes that CLT is set in the system PATH. (ignore if via is not defined as "gcs_to_asset").
overwrite Logical. If TRUE, the assetId will be overwritten.
monitoring Logical. If TRUE the exportation task will be monitored.
quiet     Logical. Suppress info message.
...       parameter(s) passed on to ee_utils_create_manifest_image

Value

An ee$Image object

See Also

Other image upload functions: raster_as_ee()

Examples

## Not run:
library(rgee)
library(stars)
ee_Initiaize(gcs = TRUE)

# Get the filename of a image
tif <- system.file("tif/L7_ETMs.tif", package = "stars")
x <- read_stars(tif)
assetId <- sprintf("%s/%s", ee_get_assethome(), 'stars_l7')

# # Method 1
# 1. Move from local to gcs
gs_uri <- local_to_gcs(x = tif, bucket = 'rgee_dev')

# 2. Create a manifest
manifest <- ee_utils_create_manifest_image(gs_uri, assetId)

# 3. Pass from gcs to asset
gcs_to_ee_image(
  manifest = manifest,
  overwrite = TRUE
)

# OPTIONAL: Monitoring progress
ee_monitoring()

# OPTIONAL: Display results
ee_stars_01 <- ee$Image(assetId)
Map$centerObject(ee_stars_01)
Map$addLayer(ee_stars_01, list(min = 0, max = 255))

# Method 2
ee_stars_02 <- stars_as_ee(
  x = x,
  overwrite = TRUE,
  assetId = assetId,
  bucket = "rgee_dev"
)
Map$centerObject(ee_stars_02)
Map$addLayer(ee_stars_02, list(min = 0, max = 255))
Index

* GIF functions
  ee_utils_gif_annotate, 65
  ee_utils_gif_creator, 67
  ee_utils_gif_save, 69

* datasets
  ee, 4
  Map, 79

* date functions
  ee_get_date_ic, 29
  ee_get_date_img, 30
  eedate_to_rdate, 4
  rdate_to_eedate, 85

* ee_check functions
  ee_check-tools, 17

* ee_clean functions
  ee_clean_container, 18
  ee_clean_credentials, 19
  ee_clean_pyenv, 20

* ee_install functions
  ee_install, 45
  ee_install_set_pyenv, 46
  ee_install_upgrade, 47

* ee_utils functions
  ee_utils_py_to_r, 72
  ee_utils_pyfunc, 71
  ee_utils_shp_to_zip, 74

* generic download functions
  ee_drive_to_local, 20
  ee_gcs_to_local, 25

* generic upload functions
  ee_utils_create_manifest_image, 60
  ee_utils_create_manifest_table, 62
  local_to_gcs, 78

* helper functions
  ee_help, 31
  ee_monitoring, 50
  ee_print, 51

* image download functions
  ee_as_raster, 5
  ee_as_stars, 11
  ee_as_thumbnail, 14
  ee_imagecollection_to_local, 32

* image export task creator
  ee_image_to_asset, 36
  ee_image_to_drive, 38
  ee_image_to_gcs, 41

* image upload functions
  raster_as_ee, 83
  stars_as_ee, 88

* path utils
  ee_get_assethome, 28
  ee_get_earthengine_path, 31

* session management functions
  ee_Initialize, 44
  ee_user_info, 59
  ee_users, 58
  ee_version, 75

* vector download functions
  ee_as_sf, 8

* vector export task creator
  ee_table_to_asset, 53
  ee_table_to_drive, 55
  ee_table_to_gcs, 56

+.EarthEngineMap (null-default), 82
future::sequential, 6, 9, 12, 33

conditions, 63
decoration_types, 65
download.file, 67
EarthEngineMap, (null-default), 82
EarthEngineMap-method (null-default), 82
ee, 4
ee_as_raster, 5, 13, 16, 34
ee_as_sf, 8
ee_as_stars, 7, 11, 16, 34
eearch asThumbnail, 7, 13, 14, 34
INDEX

ee_check (ee_check-tools), 17
ee_check-tools, 17
ee_check_credentials (ee_check-tools), 17
ee_check_python (ee_check-tools), 17
ee_check_python_packages (ee_check-tools), 17
ee_clean_container, 18, 19, 20
ee_clean_credentials, 19, 19, 20
ee_clean_pyenv, 19, 20
ee_drive_to_local, 20, 26
ee_extract, 22
ee_gcs_to_local, 21, 25
ee_get, 27
ee_get_assethome, 28, 31
ee_get_date_ic, 5, 29, 30, 85
ee_get_date_img, 5, 29, 30, 85
ee_get_earthengine_path, 28, 31
nee help, 31, 51, 52
ee_image_info, 35
nee_image_to_asset, 36, 40, 43
ee_image_to_drive, 6, 12, 33, 37, 38, 43
ee_image_to_gcs, 6, 12, 37, 40, 41
ee_imagecollection_to_local, 7, 13, 16, 32
nee_initialize, 44, 58, 59, 75
nee_install, 45, 46, 47
nee_install_set_pyenv, 46, 46, 47
nee_install_upgrade, 46, 47
nee_manage-tools, 47
nee_manage_asset_access (nee_manage-tools), 47
nee_manage_asset_size (nee_manage-tools), 47
nee_manage_assetlist (nee_manage-tools), 47
nee_manage_cancel_all_running_task (nee_manage-tools), 47
nee_manage_copy (nee_manage-tools), 47
nee_manage_create (nee_manage-tools), 47
nee_manage_delete (nee_manage-tools), 47
nee_manage_delete_properties, 49
nee_manage_delete_properties (nee_manage-tools), 47
nee_manage_move (nee_manage-tools), 47
nee_manage_quota (nee_manage-tools), 47
nee_manage_set_properties (nee_manage-tools), 47
nee_monitoring, 31, 50, 52
nee_print, 31, 51, 51
nee_table_to_asset, 53, 56, 57
nee_table_to_drive, 54, 55, 57
nee_table_to_gcs, 54, 56, 56
nee_user_info, 45, 58, 59, 75
nee_users, 45, 58, 59, 75
nee_utils_create_json, 60
nee_utils_create_manifest_image, 60, 62, 75, 78, 84, 89
nee_utils_create_manifest_table, 61, 62, 77, 78, 87
nee_utils_future_value, 6, 10, 12, 33, 63
nee_utils_get_crs, 64
nee_utils_gif_anotate, 65, 68, 70
nee_utils_gif_creator, 66, 67, 67, 70
nee_utils_gif_save, 66, 68, 69
nee_utils_py_to_r, 72, 72, 74
nee_utils_pyfunc, 71, 73, 74
nee_utils_search_display, 73
nee_utils_shp_to_zip, 72, 73, 74
nee_version, 45, 58, 59, 75
eedate_to_rdate, 4, 29, 30, 85
extract, 22

future::value, 6, 10, 12, 33

gcs_to_ee_image, 75
gcs_to_ee_table, 76

image_annotate, 65
image_read, 69
image_write, 69

local_to_gcs, 61, 62, 78

magick::image_graph(), 65
magick::image_read(), 65
Map, 79

null-default, 82

print.ee.computedobject.ComputedObject, 83
py_func, 71

raster_as_ee, 83, 89
rdate_to_eedate, 5, 29, 30, 85
INDEX

sf_as_ee, 86
stars_as_ee, 84, 88
style_types, 65