Package 'rgee'

March 16, 2022

Title R Bindings for Calling the 'Earth Engine' API

Version 1.1.3

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 false

RoxygenNote 7.1.1

Depends R (>= 3.3.0)

Imports methods, reticulate (>= 1.24), rstudioapi (>= 0.7), leaflet (>= 2.0.2), magrittr, jsonlite, processx, leafem, crayon, R6, cli

Suggests leaflet.extras2, magick, geojsonio, sf, stars, googledrive (>= 2.0.0), googleCloudStorageR (>= 0.6.0), gargle, raster, rgdal, httr, digest, spelling, testthat, future, covr, knitr, rmarkdown, png

URL https://github.com/r-spatial/rgee/, https://r-spatial.github.io/rgee/, https://github.com/google/earthengine-api/

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

VignetteBuilder knitr

Language en-US

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/0000-0002-4035-0289>),
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 2022-03-16 15:50:02 UTC

R topics documented:

`rgee-package` *rgee: An R package for interacting with Google Earth Engine*

Description

Google Earth Engine (Gorelick et al., 2017) is a cloud computing platform designed for planetary-scale environmental data analysis that only can be accessed via the Earth Engine code editor, third-party web apps, and the JavaScript and Python client libraries. `rgee` is a non-official client library for R that uses `reticulate` to wrap the Earth Engine Python API and provide R users with a

familiar interface, rapid development features, and flexibility to analyze data using open-source, R third-party packages.

Details

The package implements and supports:

- Earth Engine Module
- Install or set all rgee dependencies
- Check non-R dependencies
- Clean non-R dependencies
- Session management
- Transform an R Date to an EE Date or vice versa
- Create Interactive visualization Maps
- Image download
- Vector download
- Generic download
- Assets management
- Upload raster
- Upload vector
- Upload generic
- Extract values
- Helper functions
- Utils functions

I. Earth Engine Module

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.).

ee	Main Earth Engine module.

II. Install or set non-R rgee dependencies

ee_install	Create an isolated Python virtual environment with all rgee dependencies.
ee_install_set_pyenv	Configure which version of Python to use with rgee.
ee_install_upgrade	Upgrade the Earth Engine Python API.

III. Check non-R dependencies

ee_check	Check all non-R dependencies.
ee_check_python	Check Python environment.
ee_check_credentials	Check Google credentials.
ee_check_python_packages	Check Python packages: earthengine-api and numpy.

IV. Clean container, credentials, or rgee system variables

ee_clean_container	Delete files from either a Folder or a Bucket.
ee_clean_credentials	Delete Credentials.
ee_clean_pyenv	Remove rgee system variables from .Renviron.

V. Session management

ee_Initialize	Authenticate and Initialize Earth Engine.
ee_version	Earth Engine API version.
ee_user_info	Display the credentials and general info of the initialized user.
ee_users	Display the credentials of all users as a table.
ee_get_assethome	Get the Asset home name.
ee_get_earthengine_path	Get the path where the credentials are stored.

VII. Transform an R Date to an EE Date or vice versa

eedate_to_rdate	Pass an Earth Engine date object to R.
rdate_to_eedate	Pass an R date object to Earth Engine.
ee_get_date_img	Get the date of a EE Image.
ee_get_date_ic	Get the date of a EE ImageCollection.

VIII. Visualization Map

Map	R6 object (Map) to display Earth Engine (EE) spatial objects.
R6Map	R6 class to display Earth Engine (EE) spatial objects.

IX. Image download

ee_as_raster	Convert an Earth Engine (EE) image in a raster object.
ee_as_stars	Convert an Earth Engine (EE) image in a stars object.
ee_as_thumbnail	Create an R spatial gridded object from an EE thumbnail image.
ee_image_to_asset	Creates a task to export an EE Image to their EE Assets.
ee_image_to_drive	Creates a task to export an EE Image to Drive.
ee_image_to_gcs	Creates a task to export an EE Image to Google Cloud Storage.
ee_image_info	Approximate size of an EE Image object.
ee_imagecollection_to_local	Save an EE ImageCollection in their local system.

X. Vector download

ee_as_sf	Convert an Earth Engine table in an sf object.
ee_table_to_asset	Creates a task to export a FeatureCollection to an EE table asset.
ee_table_to_drive	Creates a task to export a FeatureCollection to Google Drive.
ee_table_to_gcs	Creates a task to export a FeatureCollection to Google Cloud Storage.

XI. Generic download

ee_drive_to_local	Move results from Google Drive to a local directory.
ee_gcs_to_local	Move results from Google Cloud Storage to a local directory.

XII. Assets management

ee_manage-tools	Interface to manage the Earth Engine Asset.

XIII. Upload raster

stars_as_ee	Convert a stars or stars-proxy object into an EE Image object.
raster_as_ee	Convert a Raster* object into an EE Image object.
gcs_to_ee_image	Move a GeoTIFF image from GCS to their EE assets.

XIV. Upload vector

gcs_to_ee_table	Move a zipped shapefile from GCS to their EE Assets.
sf_as_ee	Convert an sf object to an EE object.

XV. Upload generic

local_to_gcs	Upload local files to Google Cloud Storage.

XVI. Extract values

ee_extract	Extract values from EE Images or ImageCollections objects.

XVII. Helper functions

ee_help	Documentation for Earth Engine Objects.
ee_print	Print and return metadata about Spatial Earth Engine Objects.
ee_monitoring	Monitoring Earth Engine task progress.
print	Print Earth Engine objects.

XVIII. Utils functions

ee_utils_py_to_r	Convert between Python and R objects.
ee_utils_pyfunc	Wrap an R function in a Python function with the same signature.
ee_utils_shp_to_zip	Create a zip file from an sf object.
ee_utils_create_json	Convert a R list into a JSON file.
ee_utils_create_manifest_image	Create a manifest to upload an image.
ee_utils_create_manifest_table	Create a manifest to upload a table.
ee_utils_get_crs	Convert EPSG, ESRI or SR-ORG code into a OGC WKT.
ee_utils_future_value	The value of a future or the values of all elements in a container.
ee_utils_dataset_display	Search into the Earth Engine Data Catalog.

Acknowledgments

We want to offer a special thanks to Justin Braaten for his wise and helpful comments in the whole development of rgee. As well, we would like to mention the following third-party R/Python packages for contributing indirectly to the improvement of rgee:

- gee_asset_manager - Lukasz Tracewski
- geeup - Samapriya Roy

- geeadd - Samapriya Roy
- eemont - David Montero Loaiza
- cartoee - Kel Markert
- geetools - Rodrigo E. Principe
- landsat-extract-gee - Loïc Dutrieux
- earthEngineGrabR - JesJehle
- sf - Edzer Pebesma
- stars - Edzer Pebesma
- gdalcubes - Marius Appel

Author(s)

Maintainer: Cesar Aybar <csaybar@gmail.com> (ORCID)

Other contributors:

- Wu Qiusheng <giswqs@gmail.com> (ORCID) [contributor]
- Lesly Bautista <leslyarcelly.213@gmail.com> (ORCID) [contributor]
- Roy Yali <ryali93@gmail.com> (ORCID) [contributor]
- Antony Barja <antony.barja8@gmail.com> (ORCID) [contributor]
- Kevin Ushey <kevin@rstudio.com> [contributor]
- Jeroen Ooms <jeroen@berkeley.edu> (ORCID) [contributor]
- Tim Appelhans <tim.appelhans@gmail.com> [contributor]
- JJ Allaire <jj@rstudio.com> [contributor]
- Yuan Tang <terrytangyuan@gmail.com> [contributor]
- Samapriya Roy <samapriya.roy@gmail.com> [contributor]
- MariaElena Adauto <2a.mariaelena@gmail.com> (ORCID) [contributor]
- Gabriel Carrasco <gabriel.carrasco@upch.pe> (ORCID) [contributor]
- Henrik Bengtsson <henrikb@braju.com> [contributor]
- Jeffrey Hollister <hollister.jeff@epa.gov> (Hollister reviewed the package for JOSS, see https://github.com/openjournals/joss-reviews/issues/2272/) [reviewer]
- Gennadii Donchyts (Gena reviewed the package for JOSS, see https://github.com/openjournals/joss-reviews/issues/2272/) [reviewer]
- Marius Appel <marius.appel@uni-muenster.de> (Appel reviewed the package for JOSS, see https://github.com/openjournals/joss-reviews/issues/2272/) [reviewer]

See Also

Useful links:

- https://github.com/r-spatial/rgee/
- https://r-spatial.github.io/rgee/
- https://github.com/google/earthengine-api/
- Report bugs at https://github.com/r-spatial/rgee/issues/

ee *Main Earth Engine module*

Description

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

Usage

```
ee
```

Format

Earth Engine module

Examples

```
## 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

```
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 supports 32 bit signed (such integers can only count up to about 2 billion). This range is notably insufficient for dealing with GEE date objects represented by timestamps in milliseconds since the UNIX epoch. eedate_to_rdate uses Python in the 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

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

eeDate <- ee$Date$fromYMD(2010,1,1)
eedate_to_rdate(eeDate,timestamp = TRUE) # good
eeDate$getInfo()$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

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

```
  timePrefix = TRUE,
  quiet = FALSE,
  ...
)
```

Arguments

image	ee$Image to be converted into a raster object.
region	EE Geometry (ee$Geometry$Polygon) which specifies the region to export. CRS needs to be the same that the argument image. Otherwise, it will be forced. If not specified, image bounds are taken.
dsn	Character. Output filename. If missing, a temporary file is created.
via	Character. Method to export the image. Two methods are implemented: "drive", "gcs". See details.
container	Character. Name of the folder ('drive') or bucket ('gcs') to be exported.
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 is 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_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 is started (i.e., ee$batch$Task$start()) to move the ee$Image from Earth Engine to the intermediate container specified in the argument via.
- 2. If the argument lazy is TRUE, the task is not be monitored. This is useful to lunch several tasks simultaneously and calls them later using ee_utils_future_value or future::value. At the end of this step, the ee$Image is stored on the path specified in the argument dsn.
- 3. Finally, if the argument add_metadata is TRUE, a list with the following elements are 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 `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

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

ee_Initialize(drive = TRUE, gcs = TRUE)
ee_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_raster(
```

```
  image = img,
  region = geometry,
  via = "drive"
)

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

img_02_result <- img_02 %>% ee_utils_future_value()
img_02_result@history$metadata # metadata

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

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

img_03_result <- img_03 %>% ee_utils_future_value()
img_03_result@history$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_sf *Convert an Earth Engine table into a sf object*

Description

Convert an Earth Engine table into a sf object

Usage

```
ee_as_sf(
  x,
  dsn,
  overwrite = TRUE,
  via = "getInfo",
  container = "rgee_backup",
  crs = NULL,
  maxFeatures = 5000,
  selectors = NULL,
  lazy = FALSE,
  public = TRUE,
  add_metadata = TRUE,
  timePrefix = TRUE,
  quiet = FALSE
)
```

Arguments

x	Earth Engine table (ee$FeatureCollection) to be converted into a sf object.
dsn	Character. Output filename. In case dsn is missing, a shapefile is created in the tmp() directory.
overwrite	Logical. Delete data source dsn before attempting to write?.
via	Character. Method to export the image. Three method are implemented: "getInfo", "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").
crs	Integer or Character. Coordinate Reference System (CRS) for the EE table. If it is NULL, ee_as_sf will take the CRS of the first element.
maxFeatures	Numeric. The maximum allowed number of features to export (ignored if via is not set as "getInfo"). The task will fail if the exported region covers more features than the specified in maxFeatures. Defaults to 5000.
selectors	The list of properties to include in the output, as a list/vector of strings or a comma-separated string. By default, all properties are included.
lazy	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.
public	Logical. If TRUE, a public link to the file is created. See details.
add_metadata	Add metadata to the sf object. See details.
timePrefix	Logical. Add current date and time (Sys.time()) as a prefix to export files. This parameter helps to avoid exported files with the same name. By default TRUE.
quiet	logical. Suppress info message.

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, making it not recommendable for large FeatureCollection. Instead of "getInfo", the options: "drive" and "gcs" are suitable for large FeatureCollections due to the use of an intermediate container. When `via` is set as "drive" or "gcs" `ee_as_sf` perform the following steps:

- 1. A task is started (i.e., `ee$batch$Task$start()`) to move the EE Table from Earth Engine to the file storage system (Google Drive or Google Cloud Storage) specified in the argument `via`.
- 2. If the argument `lazy` is TRUE, the task will not be monitored. This is useful to launch several tasks simultaneously and calls them later using `ee_utils_future_value` or `future::value`. At the end of this step, the EE Table is stored under the path specified by the argument `dsn`.
- 3. Finally, if the argument `add_metadata` is TRUE, a list with the following elements is 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.

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

Value

An sf object.

Examples

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

ee_Initialize(drive = TRUE, gcs = TRUE)

# Region of interest
```

```
roi <- ee$Geometry$Polygon(list(
  c(-122.275, 37.891),
  c(-122.275, 37.868),
  c(-122.240, 37.868),
  c(-122.240, 37.891)
))

# TIGER: US Census Blocks Dataset
blocks <- ee$FeatureCollection("TIGER/2010/Blocks")
subset <- blocks$filterBounds(roi)
sf_subset <- ee_as_sf(x = subset)
plot(sf_subset)

# Create Random points in Earth Engine
region <- ee$Geometry$Rectangle(-119.224, 34.669, -99.536, 50.064)
ee_help(ee$FeatureCollection$randomPoints)
ee_randomPoints <- ee$FeatureCollection$randomPoints(region, 100)

# Download via GetInfo
sf_randomPoints <- ee_as_sf(ee_randomPoints)
plot(sf_randomPoints)

# Download via drive
sf_randomPoints_drive <- ee_as_sf(
  x = ee_randomPoints,
  via = 'drive'
)

# Download via GCS
sf_randomPoints_gcs <- ee_as_sf(
 x = subset,
 via = 'gcs',
 container = 'rgee_dev' #GCS bucket name
)

## End(Not run)
```

`ee_as_stars` *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 = "rgee_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 specifies the region to export. CRS needs to be the same that the argument image. Otherwise, it will be forced. If not specified, image bounds are taken.
dsn	Character. Output filename. If missing, a temporary file is created.
via	Character. Method to export the image. Two methods are implemented: "drive", "gcs". See details.
container	Character. Name of the folder ('drive') or bucket ('gcs') to be exported.
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 is 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 export files. 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 is started (i.e. ee$batch$Task$start()) to move the ee$Image from Earth Engine to the intermediate container specified in the argument via.

- 2. If the argument `lazy` is TRUE, the task will not be monitored. This is useful to lunch several tasks simultaneously and calls them later using `ee_utils_future_value` or `future::value`. At the end of this step, the `ee$Image` is stored on the path specified in the argument `dsn`.
- 3. Finally, if the argument `add_metadata` is TRUE, a list with the following elements is 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.

Value

A stars-proxy object

See Also

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

Examples

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

ee_Initialize(drive = TRUE, gcs = TRUE)
ee_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

```
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 to 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`.

Parameter	Description	Type
bands	Comma-delimited list of three band (RGB)	list
min	Value(s) to map to 0	number or list of three numbers, one for each band
max	Value(s) to map to 1	number or list of three numbers, one for each band
gain	Value(s) by which to multiply each pixel value	number or list of three numbers, one for each band
bias	Value(s) to add to each Digital Number value	number or list of three numbers, one for each band

gamma	Gamma correction factor(s)	number or list of three numbers, one for each band
palette	List of CSS-style color strings (single-band only)	comma-separated list of hex strings
opacity	The opacity of the layer (from 0 to 1)	number

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

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

ee_Initialize()

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 <- ee$Image("CGIAR/SRTM90_V4")
world_region <- ee$Geometry$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 Third-Party Python Packages used by rgee.

Usage

```
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

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

ee_check_python()
ee_check_python_packages()
ee_check_credentials()
ee_check() # put them all together

## End(Not run)
```

---

ee_clean_container *Delete files from either a Folder (Google Drive) or a Bucket (GCS)*

Description

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

Usage

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

Arguments

name	Character. Name of the folder (Google Drive) or bucket (GCS) to delete all files.
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_credentials *Delete Credentials*

Description

Delete all the credentials according to a specific user. The credentials (Google Earth Engine, Google Drive and Google Cloud Storage) are created after running ee_Initialize(...) successfully. They are saved in the path rgee::ee_get_earthengine_path().

Usage

```
ee_clean_credentials(user = "not_defined", quiet = FALSE)
```

Arguments

user	Character. Earth Engine user (e.g. data.colec.fbf).
quiet	Logical. Suppress info messages.

Value

No return value, called for cleaning Google Drive, Google Cloud Storage, and/or Earth Engine credentials.

See Also

Other ee_clean functions: ee_clean_container(), ee_clean_pyenv()

ee_clean_pyenv *Remove rgee system variables from .Renviron*

Description

Remove rgee system variables from .Renviron

Usage

```
ee_clean_pyenv(Renviron = "global")
```

Arguments

Renviron	Character. If it is "global" the environment variables in the .Renviron located in the Sys.getenv("HOME") folder will be deleted. On the other hand, if it is "local" the environment variables in the .Renviron on the working directory (getwd()) will be deleted. Finally, users can also enter a specific absolute path (see examples).

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 *Move results from Google Drive to a local directory*

Description

Move results of an EE task saved in Google Drive to a local directory.

Usage

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

Arguments

task	List generated after finished an EE task correctly. See details.
dsn	Character. Output filename. If missing, a temporary file will be assigned.
overwrite	A boolean argument that indicates whether "filename" should be overwritten. By default TRUE.
consider	Interactive. See details.
public	Logical. If TRUE, a public link to the Google Drive resource is created.
metadata	Logical. If TRUE, export the metadata related to the Google Drive resource. See details.
quiet	logical. Suppress info message.

Details

The task argument needs a status as task "COMPLETED" to work, due that the parameters necessary to identify EE objects into Google Drive are obtained from `ee$batch$Export$*$toDrive(...)$start()$status()`.

`consider` argument is necessary due that 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.

Finally, if the argument `metadata` is TRUE, a list with the following elements are exported join with the output filename (dsn):

- **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.

Value

If `metadata` is FALSE, will return the filename of the Google Drive resource on their system. Otherwise, a list with two elements (`dns` and `metadata`) is returned.

See Also

Other generic download functions: `ee_gcs_to_local()`

Examples

```
## 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$xmin, rlist$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_drive(
  image = mean_l5_Amarakaeri,
  folder = "Amarakaeri",
  fileFormat = "GEO_TIFF",
  region = ee_ROI,
  fileNamePrefix = "my_image_demo"
)

task_img$start()
ee_monitoring(task_img)

# 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 an `ee$Image` at the locations of a geometry object. Users can use `ee$Geometry$*`, `ee$Feature`, `ee$FeatureCollection`, sf or sfc object to filter spatially. This function mimicking how extract currently works.

Usage

```
ee_extract(
  x,
  y,
  fun = ee$Reducer$mean(),
  scale = NULL,
  sf = FALSE,
  via = "getInfo",
  container = "rgee_backup",
  lazy = 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 an sf object?
`via`	Character. Method to export the image. Three method are implemented: "getInfo", "drive", "gcs".
`container`	Character. Name of the folder ('drive') or bucket ('gcs') to be exported into (ignore if `via` is not defined as "drive" or "gcs").
`lazy`	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.
`quiet`	Logical. Suppress info message.
`...`	ee$Image$reduceRegions additional parameters. See `ee_help(ee$Image$reduceRegions)` for more details.

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

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

ee_Initialize(gcs = TRUE, drive = TRUE)

# Define a Image or ImageCollection: Terraclimate
terraclimate <- 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 - getInfo
ee_nc_rain <- ee_extract(
 x = terraclimate,
 y = nc["NAME"],
 scale = 250,
 fun = ee$Reducer$mean(),
 sf = TRUE
)

# Extract values - drive (lazy = TRUE)
ee_nc_rain <- ee_extract(
 x = terraclimate,
 y = nc["NAME"],
 scale = 250,
 fun = ee$Reducer$mean(),
 via = "drive",
 lazy = TRUE,
 sf = TRUE
)
ee_nc_rain <- ee_nc_rain %>% ee_utils_future_value()

# Extract values - gcs (lazy = FALSE)
ee_nc_rain <- ee_extract(
 x = terraclimate,
 y = nc["NAME"],
 scale = 250,
```

```
 fun = ee$Reducer$mean(),
 via = "gcs",
 container = "rgee_dev",
 sf = TRUE
)

# Spatial plot
plot(
 ee_nc_rain["X200101_Terraclimate_pp_2001_01_01"],
 main = "2001 Jan Precipitation - Terraclimate",
 reset = FALSE
)

## End(Not run)
```

`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

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

Arguments

`task`	List generated after finished an EE task correctly. See details.
`dsn`	Character. Output filename. If missing, a temporary file (i.e. `tempfile()`) is assigned.
`public`	Logical. If TRUE, a public link to Google Cloud Storage resource is created.
`metadata`	Logical. If TRUE, export the metadata related to the Google Cloud Storage resource. See details.
`overwrite`	A boolean argument that 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 due to that the parameters necessaries to locate the file into Google Cloud Storage are obtained from `ee$batch$Export$*$toCloudStorage(...)$start()$status()`.

If the argument `metadata` is TRUE, a list with the following elements is exported join with the output filename (dsn):

- **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.

Value

If `metadata` is FALSE, will return the filename of the Google Cloud Storage resource on their system. Otherwise, a list with two elements (`dns` and `metadata`) is returned.

See Also

Other generic download functions: `ee_drive_to_local()`

Examples

```
## 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(
   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
img <- ee_gcs_to_local(task = task_img)

## End(Not run)
```

`ee_get_assethome` *Get the Asset home name*

Description

Get the Asset home name

Usage

```
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

```
## Not run:
library(rgee)
ee_Initialize()
ee_get_assethome()

## End(Not run)
```

`ee_get_date_ic` *Get the date of a EE ImageCollection*

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` Sets 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` (only if the argument `time_end` is set as TRUE). The number of rows depends on 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

```
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 sets 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 (only if the time_end argument is TRUE).

See Also

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

Examples

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

eeobject	Earth Engine Object to print documentation.
browser	Logical. Display documentation in the browser.

Value

No return value, called for displaying Earth Engine documentation.

See Also

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

Examples

```
## 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

```
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 methods 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 is 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 export files. 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

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 is started (i.e., ee$batch$Task$start()) to move the ee$Image from Earth Engine to the intermediate container specified in the argument via.
- 2. If the argument lazy is TRUE, the task will not be monitored. This is useful to lunch several tasks simultaneously and calls them later using ee_utils_future_value or future::value. At the end of this step, the ee$Images are 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.

Value

If add_metadata is FALSE, `ee_imagecollection_to_local` will return a character vector containing the filename of the images downloaded. Otherwise, if add_metadata is TRUE, will return a list with extra information related to the exportation (see details).

See Also

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

Examples

```
## Not run:
library(rgee)
library(raster)
ee_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)
```

`ee_image_info` *Approximate size of an EE Image object*

Description

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

Usage

```
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 is estimated.
`compression_ratio`	Numeric. Measurement of the relative data size reduction 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

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

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

```
# Landast8
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

```
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. It 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. It can be specified as 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. **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)

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()


# 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
)

task_img$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

```
ee_image_to_drive(
  image,
  description = "myExportImageTask",
  folder = "rgee_backup",
```

```
  fileNamePrefix = NULL,
  timePrefix = TRUE,
  dimensions = NULL,
  region = NULL,
  scale = NULL,
  crs = NULL,
  crsTransform = NULL,
  maxPixels = NULL,
  shardSize = NULL,
  fileDimensions = NULL,
  skipEmptyTiles = NULL,
  fileFormat = NULL,
  formatOptions = NULL
)
```

Arguments

image	The image to be exported.
description	Human-readable name of the task.
folder	The name of a folder in their Drive account to be exported into. By default "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. It 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. It can be specified as 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', '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

```
## 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$xmin, rlist$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_drive(
  image = mean_l5_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

```
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,
  fileFormat = NULL,
  formatOptions = 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. It can be specified as 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 assset 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

```
## 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 GCS
task_img <- ee_image_to_gcs(
 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 GCS to local
ee_gcs_to_local(task = task_img)


## End(Not run)
```

`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 users to sign into your Google account and allows you to grant permission to manage resources. This function is a wrapper around `rgee::ee$Initialize()`.

Usage

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

Arguments

`user`	Character (optional, e.g. `data.colec.fbf`). The user argument is used to create a folder inside the path `~/.config/earthengine/` that save all the credentials for a specific Google identity.
`drive`	Logical (optional). If TRUE, the drive credential is cached in the path `~/.config/earthengine/`.
`gcs`	Logical (optional). If TRUE, the Google Cloud Storage credential is 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 "user" argument, all user credentials are saved in the the subdirectory ~/.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

```
## 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. If it exists, delete the previous Python environment specified in the py_env argument.
- 3. Create a new Python environment (See py_env) argument.
- 4. Set the environment variable EARTHENGINE_PYTHON and EARTHENGINE_ENV. 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

```
ee_install(
  py_env = "rgee",
  earthengine_version = ee_version(),
  python_version = "3.8",
  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().
python_version	Only windows users. The Python version to be used in this conda environment. When NULL, the default python package will be used instead. For example, use python_version = "3.6" to request that the conda environment be created with a copy of Python 3.6.
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()

Examples

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

## End(Not run)
```

---

ee_install_set_pyenv *Specify a Python environment for rgee*

Description

Specify a Python environment to use with rgee. This function creates a .Renviron file that contains two environmental variables: 'EARTHENGINE PYTHON' and 'EARTHENGINE ENV'. If an .Renviron file is already in use, ee_install_set_pyenv will append the two previous environmental variables to the end of the file. If the prior two environmental variables were previously set, ee_install_set_pyenv will simply overwrite them. See details to get more information.

Usage

```
ee_install_set_pyenv(
  py_path,
  py_env = NULL,
  Renviron = "global",
  confirm = interactive(),
  quiet = FALSE
)
```

Arguments

py_path	The path to a Python interpreter
py_env	The name of the conda or venv environment. If NULL, ee_install_upgrade and py_install functions will not work.
Renviron	Character. If it is "global" the environment variables are set in the .Renviron located in the Sys.getenv("HOME") folder. On the other hand, if it is "local" the environment variables are set in the .Renviron on the working directory (getwd()). Finally, users can also enter a specific path (see examples).
confirm	Logical. Confirm before restarting R?.
quiet	Logical. Suppress info message

Details

The 'EARTHENGINE_PYTHON' set the Python interpreter path to use with rgee. In the other hand, the 'EARTHENGINE ENV' set the Python environment name. Both variables are storage in an .Renviron file. See Startup documentation to get more information about startup files in R.

Value

no return value, called for setting EARTHENGINE_PYTHON in .Renviron

See Also

Other ee_install functions: ee_install_upgrade(), ee_install()

Examples

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

## IMPORTANT: Change 'py_path' argument according to your own Python PATH
## For Anaconda users - Windows OS
## OBS: Anaconda Python PATH can vary, run "where anaconda" in console.
# win_py_path = paste0(
#    "C:/Users/UNICORN/AppData/Local/Programs/Python/",
#    "Python37/python.exe"
# )
# ee_install_set_pyenv(
#   py_path = win_py_path,
```

```
#   py_env = "rgee" # Change it for your own Python ENV
# )

## For Anaconda users - MacOS users
# ee_install_set_pyenv(
#   py_path = "/Users/UNICORN/opt/anaconda3/bin/python",
#   py_env = "rgee" # Change it for your own Python ENV
# )
#
## For Miniconda users - Windows OS
# win_py_path = paste0(
#   "C:/Users/UNICORN/AppData/Local/r-miniconda/envs/rgee/",
#   "python.exe"
# )
# ee_install_set_pyenv(
#   py_path = win_py_path,
#   py_env = "rgee" # Change it for your own Python ENV
# )

## For Miniconda users - Linux/MacOS users
# unix_py_path = paste0(
#   "/home/UNICORN/.local/share/r-miniconda/envs/",
#   "rgee/bin/python3"
# )
# ee_install_set_pyenv(
#   py_path = unix_py_path,
#   py_env = "rgee" # Change it for your own Python ENV
# )

## For virtualenv users - Linux/MacOS users
# ee_install_set_pyenv(
#   py_path = "/home/UNICORN/.virtualenvs/rgee/bin/python",
#   py_env = "rgee" # Change it for your own Python ENV
# )

## For Python root user - Linux/MacOS users
# ee_install_set_pyenv(
#   py_path = "/usr/bin/python3",
#   py_env = NULL,
#   Renviron = "global" # Save ENV variables in the global .Renv file
# )

# ee_install_set_pyenv(
#   py_path = "/usr/bin/python3",
#   py_env = NULL,
#   Renviron = "local" # Save ENV variables in a local .Renv file
# )

## End(Not run)
```

`ee_install_upgrade` *Upgrade the Earth Engine Python API*

Description

Upgrade the Earth Engine Python API

Usage

```
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()

Examples

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

## End(Not run)
```

---

ee_manage-tools *Interface to manage the Earth Engine Asset*

Description

R functions to manage 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

```
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

path_asset	Character. Name of the EE asset (Table, Image, Folder or ImageCollection).
asset_type	Character. The asset type to create ('Folder' or 'ImageCollection').
quiet	Logical. Suppress info message.
strict	Character vector. If TRUE, the existence of the asset will be evaluated before performing the task.
final_path	Character. Output filename (e.g users/datacolecfbf/ic_moved)
add_properties	List. Set of parameters to established as a property of an EE object. See details.
del_properties	Character. Names of properties to be deleted. See details.
owner	Character vector. Define owner user in the IAM Policy.
editor	Character vector. Define editor users in the IAM Policy.
viewer	Character vector. Define viewer users in the IAM Policy.

`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 and improved by csaybar.

Examples

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

ee_Initialize()
ee_user_info()

# Change datacolecfbf 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_manage_assetlist(path_asset = addm("rgee/rgee_folder"))

# 5. Set properties to an EE object.
ee_manage_set_properties(
  path_asset = addm("rgee/rgee_folder/rgee_ic_moved"),
  add_properties = list(message = "hello-world", language = "R")
)

ic_id <- addm("rgee/rgee_folder/rgee_ic_moved")
test_ic <- ee$ImageCollection(ic_id)
test_ic$getInfo()

# 6. Delete properties
ee_manage_delete_properties(
  path_asset = addm("rgee/rgee_folder/rgee_ic_moved"),
  del_properties = c("message", "language")
)
test_ic$getInfo()

# 7. Create a report based on all the tasks
# that are running or have already been completed.
ee_manage_task()

# 8. Cancel all the running task
ee_manage_cancel_all_running_task()

# 9. Delete EE objects or folders
ee_manage_delete(addm("rgee/"))

## End(Not run)
```

`ee_monitoring` *Monitoring Earth Engine task progress*

Description

Monitoring Earth Engine task progress

Usage

```
ee_monitoring(
  task,
  task_time = 5,
  eeTaskList = FALSE,
  quiet = FALSE,
  max_attempts = 5
)

ee_check_task_status(task, quiet = FALSE)
```

Arguments

task	List generated after a task is started (i.e., after run ee$batch$Task$start()) or a character that represents the ID of a EE task started.
task_time	Numeric. How often (in seconds) should a task be polled?
eeTaskList	Logical. If TRUE, all Earth Engine tasks will be listed.
quiet	Logical. Suppress info message
max_attempts	Number of times to monitor the tasks before ending.

Value

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

See Also

Other helper functions: ee_help(), ee_print()

Examples

```
## Not run:
library(rgee)
ee_Initialize()
ee_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

```
ee_print(eeobject, ...)

## S3 method for class 'ee.geometry.Geometry'
ee_print(eeobject, ..., clean = FALSE, quiet = FALSE)

## S3 method for class 'ee.feature.Feature'
ee_print(eeobject, ..., clean = FALSE, quiet = FALSE)

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

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

## S3 method for class 'ee.image.Image'
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 data size reduction produced by a data compression algorithm (ignored if eeobject is not an ee$Image or ee$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

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

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

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

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

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

srtm_clip <- ee$Image("CGIAR/SRTM90_V4")$clip(geom)
srtm_metadata <- ee_print(srtm_clip)
srtm_metadata$img_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(ee_print)

## End(Not run)
```

---

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
)
```

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

```
## 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
)
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

```
## 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$xmin, rlist$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")))
)

task_vector <- ee_table_to_drive(
  collection = amk_fc,
  fileFormat = "GEO_JSON",
  fileNamePrefix = "geom_Amarakaeri"
)
task_vector$start()
ee_monitoring(task_vector) # 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(...)`.

Usage

```
ee_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

```
## 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()

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

```
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()

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_cog_metadata` *Return metadata of a COG tile server*

Description

Return metadata of a COG tile server

Usage

```
ee_utils_cog_metadata(
  resource,
  visParams,
  titiler_server = "https://api.cogeo.xyz/"
)
```

Arguments

`resource`	Character that represents a COG tile server file.
`visParams`	Visualization parameters see "https://api.cogeo.xyz/docs".
`titiler_server`	TiTiler endpoint. Defaults to "https://api.cogeo.xyz/".

Value

A metadata list for a COG file.

Examples

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

server <- "https://s3-us-west-2.amazonaws.com/planet-disaster-data/hurricane-harvey/"
file <- "SkySat_Freeport_s03_20170831T162740Z3.tif"
resource <- paste0(server, file)
visParams <- list(nodata = 0, expression = "B3, B2, B1", rescale = "3000, 13500")
ee_utils_cog_metadata(resource, visParams)

## End(Not run)
```

ee_utils_create_json *Convert an R list into a JSON file in the temp() file*

Description

Convert an R list into a JSON file in the temp() file

Usage

```
ee_utils_create_json(x)
```

Arguments

x	List to convert into a JSON file.

Value

A JSON file saved in a /tmp dir.

Examples

```
## 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 asset folder. The "manifest" is simply a JSON file that describe all the upload parameters. See https://developers.google.com/earth-engine/guides/image_manifest to get more details.

Usage

```
ee_utils_create_manifest_image(
  gs_uri,
  assetId,
  properties = NULL,
  start_time = "1970-01-01",
  end_time = "1970-01-01",
  pyramiding_policy = "MEAN",
```

```
  returnList = FALSE,
  quiet = FALSE
)
```

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/datacolecfbf/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

```
## Not run:
library(rgee)
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/datacolecfbf/l8"
)

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

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

`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 that describe all the upload parameters. See https://developers.google.com/earth-engine/guides/image_manifest to get more details.

Usage

```
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

```
## Not run:
library(rgee)
library(sf)
ee_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_dataset_display

Search into the Earth Engine Data Catalog

Description

Search into the Earth Engine Data Catalog

Usage

```
ee_utils_dataset_display(ee_search_dataset)
```

Arguments

ee_search_dataset	Character that represents the EE dataset ID.

Value

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

Examples

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

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

## 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

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

Arguments

`future,`	x 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/

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

```
## 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)
```

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

See Also

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

Examples

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

# Earth Engine List
ee_SimpleList <- ee$List$sequence(0, 12)
ee_NewList <- ee_SimpleList$map(
  ee_utils_pyfunc(
    function(x) {
      ee$Number(x)$add(x)
    }
  )
)

ee_NewList$getInfo()

# Earth Engine ImageCollection
constant1 <- ee$Image(1)
constant2 <- ee$Image(2)
ee_ic <- ee$ImageCollection(c(constant2, constant1))
ee_newic <- ee_ic$map(
  ee_utils_pyfunc(
    function(x) ee$Image(x)$add(x)
  )
)
ee_newic$mean()$getInfo()$type

## End(Not run)
```

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

Value

An R object

See Also

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

ee_utils_sak_copy *Stores a Service account key (SaK) inside the EE folder*

Description

Copy SaK in the ~/.config/earthengine/$USER.

Usage

```
ee_utils_sak_copy(sakfile, users = NULL, delete = FALSE, quiet = FALSE)
```

Arguments

sakfile	Character. SaK filename. If missing, the SaK of the first user is used.
users	Character. The user related to the SaK file. A SaK file can be related to multiple users.
delete	Logical. If TRUE, the SaK filename is deleted after copy.
quiet	Logical. Suppress info message

Examples

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

ee_Initialize()

# sakfile <- "/home/rgee_dev/sak_file.json"
## Copy sakfile to the users 'csaybar' and 'ndef'
# ee_utils_sak_copy(sakfile = sakfile, users = c("csaybar", "ndef"))

# # Copy the sakfile of the user1 to the user2 and user3.
# ee_utils_sak_copy(users = c("csaybar", "ndef", "ryali93"))

## End(Not run)
```

`ee_utils_sak_validate` *Validate a Service account key (SaK)*

Description

Validate a Service account key (SaK). local_to_gcs, raster_as_ee, stars_as_ee, and sf_as_ee(via = "gcs_to_asset", ...) need that the SaK have privileges to write/read objects in a GCS bucket.

Usage

```
ee_utils_sak_validate(sakfile, bucket = NULL, quiet = FALSE)
```

Arguments

`sakfile`	Character. SaK filename.
`bucket`	Character. Name of the GCS bucket. If bucket is not set, rgee will tries to create a bucket using `googleCloudStorageR::gcs_create_bucket`.
`quiet`	Logical. Suppress info message

Examples

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

ee_Initialize(gcs = TRUE)

# Check a specific SaK
sakfile <- "/home/rgee_dev/sak_file.json"
ee_utils_sak_validate(sakfile, bucket = "rgee_dev")

# Check the SaK for the current user
ee_utils_sak_validate()

## End(Not run)
```

`ee_utils_shp_to_zip` *Create a zip file from an sf object*

Description

Create a zip file from an 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 created zip file.

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 *Earth Engine API version*

Description

Earth Engine API version

Usage

```
ee_version()
```

Value

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

See Also

Other session management functions: ee_Initialize(), ee_user_info(), ee_users()

gcs_to_ee_image *Move a GeoTIFF image from GCS to their EE assets*

Description

Move a GeoTIFF image from GCS to their EE assets

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

```
## 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 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
```

```
ee_nc <- gcs_to_ee_table(manifest, overwrite = TRUE)
ee_monitoring()
Map$addLayer(ee$FeatureCollection(ee_nc))

## End(Not run)
```

`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, predefinedAcl = "bucketLevel", quiet = FALSE)
```

Arguments

`x`	Character. filename.
`bucket`	bucket name you are uploading to
`predefinedAcl`	Specify user access to object. Passed to `googleCloudStorageR::gcs_upload`.
`quiet`	Logical. Suppress info message.

Value

Character that 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

```
## 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)
```

`Map` *R6 object (Map) to display Earth Engine (EE) spatial objects*

Description

Create interactive visualizations of spatial EE objects (ee$FeatureCollection, ee$ImageCollection, ee$Geometry, ee$Feature, and ee$Image.) using `leaflet` in the backend.

Usage

```
Map
```

Format

An object of class environment with the following functions:

- **addLayer(eeObject, visParams, name = NULL, shown = TRUE, opacity = 1, titiler_viz_convert = TRUE, titiler_server = "https://api.cogeo.xyz/")**: Adds a given EE object to the map as a layer.
  - **eeObject:** The object to add to the interactive map.
  - **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 is represented as a number between 0 and 1. Defaults to 1.
  - **titiler_viz_convert:** Logical. If it is TRUE, Map$addLayer will transform the visParams to titiler style. Ignored if eeObject is not a COG file.
  - **titiler_server:** TiTiler endpoint. Defaults to "https://api.cogeo.xyz/".
- **addLayers(eeObject, visParams, name = NULL, shown = TRUE, opacity = 1)**: Adds a given ee$ImageCollection to the map as multiple layers.
  - **eeObject:** The ee$ImageCollection to add to the interactive map.
  - **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 is represented as a number between 0 and 1. Defaults to 1.

  - **nmax:** Numeric. The maximum number of images to display. By default 5.

- **addLegend(visParams, name = "Legend", position = c("bottomright", "topright", "bottomleft", "topleft"), color_mapping= "numeric", opacity = 1, ...)**: Adds a given ee$ImageCollection to the map as multiple layers.

  - **visParams:** List of parameters for visualization.

  - **name:** The title of the legend.

  - **position:** Character. The position of the legend. By default bottomright.

  - **color_mapping:** Map data values (numeric or factor/character) to colors according to a given palette. Use "numeric" ("discrete") for continuous (categorical) data. For display characters use "character" and add to visParams the element "values" containing the desired character names.

  - **opacity:** The legend's opacity is represented as a number between 0 and 1. Defaults to 1.

  - **...:** Extra legend creator arguments. See addLegend.

- **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 `leaflet` 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:

Parameter	Description	Type
bands	Comma-delimited list of three band (RGB)	list
min	Value(s) to map to 0	number or list of three numbers, one for each band
max	Value(s) to map to 1	number or list of three numbers, one for each band
gain	Value(s) by which to multiply each pixel value	number or list of three numbers, one for each band
bias	Value(s) to add to each Digital Number value	number or list of three numbers, one for each band
gamma	Gamma correction factor(s)	number or list of three numbers, one for each band
palette	List of CSS-style color strings (single-band only)	comma-separated list of hex strings
opacity	The opacity of the layer (from 0 to 1)	number

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

```
## Not run:
library(rgeeExtra)
library(rgee)
library(sf)

ee_Initialize()

# 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)
m5

# Case 6: Map comparison
image <- ee$Image("LANDSAT/LC08/C01/T1/LC08_044034_20140318")
Map$centerObject(image)
m_ndvi <- Map$addLayer(
  eeObject = image$normalizedDifference(list("B5", "B4")),
  visParams = list(min = 0, max = 0.7),
  name = "SF_NDVI"
) + Map$addLegend(list(min = 0, max = 0.7), name = "NDVI", position = "bottomright", bins = 4)
m6 <- m4 | m_ndvi
m6

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

# Case 8: COG support
# See parameters here: https://api.cogeo.xyz/docs

server <- "https://storage.googleapis.com/pdd-stac/disasters/"
file <- "hurricane-harvey/0831/20170831_172754_101c_3B_AnalyticMS.tif"
resource <- paste0(server, file)
visParams <- list(bands = c("B3", "B2", "B1"), min = 3000, max = 13500, nodata = 0)
Map$centerObject(resource)
```

```
Map$addLayer(resource, visParams = visParams, shown = TRUE)

## End(Not run)
```

`map-operator` *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.

`print.ee.computedobject.ComputedObject` *print Earth Engine object*

Description

print Earth Engine object

Usage

```
## 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.

R6Map *R6 class 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 `leaflet`.

Details

`R6Map` uses the Earth Engine method getMapId to fetch and return an ID dictionary used to create layers in a `leaflet` 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:

Parameter	Description	Type
bands	Comma-delimited list of three band (RGB)	list
min	Value(s) to map to 0	number or list of three numbers, one for each band
max	Value(s) to map to 1	number or list of three numbers, one for each band
gain	Value(s) by which to multiply each pixel value	number or list of three numbers, one for each band
bias	Value(s) to add to each Digital Number value	number or list of three numbers, one for each band
gamma	Gamma correction factor(s)	number or list of three numbers, one for each band
palette	List of CSS-style color strings (single-band only)	comma-separated list of hex strings
opacity	The opacity of the layer (from 0 to 1)	number

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

Public fields

`lon` The longitude of the center, in degrees.

`lat` The latitude of the center, in degrees.

`zoom` The zoom level, from 1 to 24.

`save_maps` Should `R6Map` save the previous maps?. If TRUE, Map will work in an OOP style. Otherwise it will be a functional programming style.

`previous_map_left` Container on maps in the left side.

`previous_map_right` Container on maps in the right side.

Methods

Public methods:

- `R6Map$new()`
- `R6Map$reset()`
- `R6Map$print()`
- `R6Map$setCenter()`
- `R6Map$setZoom()`
- `R6Map$centerObject()`
- `R6Map$addLayer()`
- `R6Map$addLayers()`
- `R6Map$addLegend()`
- `R6Map$clone()`

Method `new()`**:** Constructor of R6Map.

Usage:

```
R6Map$new(lon = 0, lat = 0, zoom = 1, save_maps = TRUE)
```

Arguments:

`lon` The longitude of the center, in degrees. By default -76.942478.

`lat` The latitude of the center, in degrees. By default -12.172116.

`zoom` The zoom level, from 1 to 24. By default 18.

`save_maps` Should `R6Map` save previous maps?.

Returns: A new `EarthEngineMap` object.

Method `reset()`**:** Reset to initial arguments.

Usage:

```
R6Map$reset(lon = 0, lat = 0, zoom = 1, save_maps = TRUE)
```

Arguments:

lon The longitude of the center, in degrees. By default -76.942478.

lat The latitude of the center, in degrees. By default -12.172116.

zoom The zoom level, from 1 to 24. By default 18.

save_maps Should R6Map save previous maps?.

Returns: A new EarthEngineMap object.

Examples:

```
\dontrun{
library(rgee)
ee_Initialize()

# Load an Image
image <- ee$Image("LANDSAT/LC08/C01/T1/LC08_044034_20140318")

# Create
Map <- R6Map$new()
Map$centerObject(image)

# Simple display: Map just will
Map$addLayer(
  eeObject = image,
  visParams = list(min=0, max = 10000, bands = c("B4", "B3", "B2")),
  name = "l8_01"
)
Map # display map

Map$reset() # Reset arguments
Map
}
```

Method `print()`**:** Display a EarthEngineMap object.

Usage:

```
R6Map$print()
```

Returns: An EarthEngineMap object.

Method `setCenter()`**:** Centers the map view at the given coordinates with the given zoom level. If no zoom level is provided, it uses 10 by default.

Usage:

```
R6Map$setCenter(lon = 0, lat = 0, zoom = 10)
```

Arguments:

lon The longitude of the center, in degrees. By default -76.942478.

lat The latitude of the center, in degrees. By default -12.172116.

zoom The zoom level, from 1 to 24. By default 18.

Returns: No return value, called to set initial coordinates and zoom.

Examples:

```
\dontrun{
library(rgee)

ee_Initialize()

Map <- R6Map$new()
Map$setCenter(lon = -76, lat = 0, zoom = 5)
Map

# Map$lat
# Map$lon
# Map$zoom
}
```

Method `setZoom()`**:** Sets the zoom level of the map.

Usage:

```
R6Map$setZoom(zoom = 10)
```

Arguments:

`zoom` The zoom level, from 1 to 24. By default 10.

Returns: No return value, called to set zoom.

Examples:

```
\dontrun{
library(rgee)

ee_Initialize()

Map <- R6Map$new()
Map$setZoom(zoom = 4)
Map

# Map$lat
# Map$lon
# Map$zoom
}
```

Method `centerObject()`**:** 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.

Usage:

```
R6Map$centerObject(
  eeObject,
  zoom = NULL,
  maxError = ee$ErrorMargin(1),
  titiler_server = "https://api.cogeo.xyz/"
)
```

Arguments:

`eeObject` Earth Engine spatial object.

`zoom` The zoom level, from 1 to 24. By default NULL.

`maxError` Max error when input image must be reprojected to an explicitly requested result projection or geodesic state.

`titiler_server` TiTiler endpoint. Defaults to "https://api.cogeo.xyz/".

Returns: No return value, called to set zoom.

Examples:

```
\dontrun{
library(rgee)

ee_Initialize()

Map <- R6Map$new()
image <- ee$Image("LANDSAT/LC08/C01/T1/LC08_044034_20140318")
Map$centerObject(image)
Map
}
```

Method `addLayer()`**:** Adds a given Earth Engine spatial object to the map as a layer

Usage:

```
R6Map$addLayer(
  eeObject,
  visParams = NULL,
  name = NULL,
  shown = TRUE,
  opacity = 1,
  position = NULL,
  titiler_viz_convert = TRUE,
  titiler_server = "https://api.cogeo.xyz/"
)
```

Arguments:

`eeObject` The Earth Engine spatial object to display in the interactive map.

`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 is represented as a number between 0 and 1. Defaults to 1.

`position` Character. Activate panel creation. If "left" the map will be displayed in the left panel. Otherwise, if it is "right" the map will be displayed in the right panel. By default NULL (No panel will be created).

`titiler_viz_convert` Logical. If it is TRUE, Map$addLayer will transform the visParams to titiler style. Ignored if eeObject is not a COG file.

`titiler_server` TiTiler endpoint. Defaults to "https://api.cogeo.xyz/".

Returns: An `EarthEngineMap` object.

Examples:

```
\dontrun{
library(rgee)
ee_Initialize()

# Load an Image
image <- ee$Image("LANDSAT/LC08/C01/T1/LC08_044034_20140318")

# Create
Map <- R6Map$new()
Map$centerObject(image)

# Simple display: Map just will
Map$addLayer(
  eeObject = image,
  visParams = list(min=0, max = 10000, bands = c("B4", "B3", "B2")),
  name = "l8_01"
)

Map$addLayer(
  eeObject = image,
  visParams = list(min=0, max = 20000, bands = c("B4", "B3", "B2")),
  name = "l8_02"
)

# Simple display: Map just will (if the position is not specified it will
# be saved on the right side)
Map$reset() # Reset Map to the initial arguments.
Map$centerObject(image)
Map$addLayer(
  eeObject = image,
  visParams = list(min=0, max=10000, bands = c("B4", "B3", "B2")),
  name = "l8_left",
  position = "left"
)

Map$addLayer(
  eeObject = image,
  visParams = list(min=0, max=20000, bands = c("B4", "B3", "B2")),
  name = "l8_right"
)

Map$reset()
}
```

Method `addLayers()`**:** Adds a given ee$ImageCollection to the map as multiple layers.

Usage:

```
R6Map$addLayers(
  eeObject,
```

```
  visParams = NULL,
  nmax = 5,
  name = NULL,
  shown = TRUE,
  position = NULL,
  opacity = 1
)
```

Arguments:

`eeObject` ee$ImageCollection to display in the interactive map.

`visParams` List of parameters for visualization. See details.

`nmax` Numeric. The maximum number of images to display. By default 5.

`name` The name of layers.

`shown` A flag indicating whether layers should be on by default.

`position` Character. Activate panel creation. If "left" the map will be displayed in the left panel. Otherwise, if it is "right" the map will be displayed in the right panel. By default NULL (No panel will be created).

`opacity` The layer's opacity is represented as a number between 0 and 1. Defaults to 1.

Returns: A `EarthEngineMap` object.

Examples:

```
\dontrun{
library(sf)
library(rgee)
library(rgeeExtra)

ee_Initialize()

Map <- R6Map$new()

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(0:2)

Map$centerObject(nc$geometry())
Map$addLayers(eeObject = ee_s2,position = "right")

# digging up the metadata
Map$previous_map_right$rgee$tokens

Map$reset()
}
```

Method `addLegend()`**:** Adds a color legend to an EarthEngineMap.

Usage:

```
R6Map$addLegend(
  visParams,
  name = "Legend",
  position = c("bottomright", "topright", "bottomleft", "topleft"),
  color_mapping = "numeric",
  opacity = 1,
  ...
)
```

Arguments:

`visParams` List of parameters for visualization.

`name` The title of the legend.

`position` Character. The position of the legend. By default bottomright.

`color_mapping` Map data values (numeric or factor/character) to colors according to a given palette. Use "numeric" ("discrete") for continuous (categorical) data. For display characters use "character" and add to visParams the element "values" containing the desired character names.

`opacity` The legend's opacity is represented as a number between 0 and 1. Defaults to 1.

`...` Extra legend creator arguments. See addLegend.

Returns: A `EarthEngineMap` object.

Examples:

```
\dontrun{
library(leaflet)
library(rgee)
ee_Initialize()

Map$reset()

# Load MODIS ImageCollection
imgcol <- ee$ImageCollection$Dataset$MODIS_006_MOD13Q1

# Parameters for visualization
labels <- c("good", "marginal", "snow", "cloud")
cols   <- c("#999999", "#00BFC4", "#F8766D", "#C77CFF")
vis_qc <- list(min = 0, max = 3, palette = cols, bands = "SummaryQA", values = labels)

# Create interactive map
m_qc <- Map$addLayer(imgcol$median(), vis_qc, "QC")

# continous palette
Map$addLegend(vis_qc)

# categorical palette
Map$addLegend(vis_qc, name = "Legend1", color_mapping = "discrete")

# character palette
```

```
Map$addLegend(vis_qc, name = "Legend2", color_mapping = "character")
}
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
R6Map$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
## ------------------------------------------------
## Method `R6Map$reset`
## ------------------------------------------------

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

# Load an Image
image <- ee$Image("LANDSAT/LC08/C01/T1/LC08_044034_20140318")

# Create
Map <- R6Map$new()
Map$centerObject(image)

# Simple display: Map just will
Map$addLayer(
  eeObject = image,
  visParams = list(min=0, max = 10000, bands = c("B4", "B3", "B2")),
  name = "l8_01"
)
Map # display map

Map$reset() # Reset arguments
Map

## End(Not run)

## ------------------------------------------------
## Method `R6Map$setCenter`
## ------------------------------------------------

## Not run:
library(rgee)

ee_Initialize()

Map <- R6Map$new()
Map$setCenter(lon = -76, lat = 0, zoom = 5)
```

```
Map

# Map$lat
# Map$lon
# Map$zoom

## End(Not run)

## ------------------------------------------------
## Method `R6Map$setZoom`
## ------------------------------------------------

## Not run:
library(rgee)

ee_Initialize()

Map <- R6Map$new()
Map$setZoom(zoom = 4)
Map

# Map$lat
# Map$lon
# Map$zoom

## End(Not run)

## ------------------------------------------------
## Method `R6Map$centerObject`
## ------------------------------------------------

## Not run:
library(rgee)

ee_Initialize()

Map <- R6Map$new()
image <- ee$Image("LANDSAT/LC08/C01/T1/LC08_044034_20140318")
Map$centerObject(image)
Map

## End(Not run)

## ------------------------------------------------
## Method `R6Map$addLayer`
## ------------------------------------------------

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

# Load an Image
image <- ee$Image("LANDSAT/LC08/C01/T1/LC08_044034_20140318")
```

```
# Create
Map <- R6Map$new()
Map$centerObject(image)

# Simple display: Map just will
Map$addLayer(
  eeObject = image,
  visParams = list(min=0, max = 10000, bands = c("B4", "B3", "B2")),
  name = "l8_01"
)

Map$addLayer(
  eeObject = image,
  visParams = list(min=0, max = 20000, bands = c("B4", "B3", "B2")),
  name = "l8_02"
)

# Simple display: Map just will (if the position is not specified it will
# be saved on the right side)
Map$reset() # Reset Map to the initial arguments.
Map$centerObject(image)
Map$addLayer(
  eeObject = image,
  visParams = list(min=0, max=10000, bands = c("B4", "B3", "B2")),
  name = "l8_left",
  position = "left"
)

Map$addLayer(
  eeObject = image,
  visParams = list(min=0, max=20000, bands = c("B4", "B3", "B2")),
  name = "l8_right"
)

Map$reset()

## End(Not run)

## ------------------------------------------------
## Method `R6Map$addLayers`
## ------------------------------------------------

## Not run:
library(sf)
library(rgee)
library(rgeeExtra)

ee_Initialize()

Map <- R6Map$new()

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(0:2)

Map$centerObject(nc$geometry())
Map$addLayers(eeObject = ee_s2,position = "right")

# digging up the metadata
Map$previous_map_right$rgee$tokens

Map$reset()

## End(Not run)

## ------------------------------------------------
## Method `R6Map$addLegend`
## ------------------------------------------------

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

Map$reset()

# Load MODIS ImageCollection
imgcol <- ee$ImageCollection$Dataset$MODIS_006_MOD13Q1

# Parameters for visualization
labels <- c("good", "marginal", "snow", "cloud")
cols   <- c("#999999", "#00BFC4", "#F8766D", "#C77CFF")
vis_qc <- list(min = 0, max = 3, palette = cols, bands = "SummaryQA", values = labels)

# Create interactive map
m_qc <- Map$addLayer(imgcol$median(), vis_qc, "QC")

# continous palette
Map$addLegend(vis_qc)

# categorical palette
Map$addLegend(vis_qc, name = "Legend1", color_mapping = "discrete")

# character palette
Map$addLegend(vis_qc, name = "Legend2", color_mapping = "character")

## End(Not run)
```

raster_as_ee *Convert a Raster* object into an EE Image object*

Description

Convert a Raster* object into an EE Image object

Usage

```
raster_as_ee(
  x,
  assetId,
  bucket = NULL,
  predefinedAcl = "bucketLevel",
  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.
predefinedAcl	Specify user access to object. Passed to googleCloudStorageR::gcs_upload.
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

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

ee_Initialize(gcs = TRUE)

# Get the filename of a image
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(max_attempts = Inf)

# 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

```
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()

Examples

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

## End(Not run)
```

---

sf_as_ee *Convert an sf object to an EE object*

Description

Load an sf object to Earth Engine.

Usage

```
sf_as_ee(
  x,
  via = "getInfo",
  assetId = NULL,
  bucket = NULL,
  predefinedAcl = "bucketLevel",
```

```
  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").
predefinedAcl	Specify user access to object. Passed to googleCloudStorageR::gcs_upload.
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	A boolean argument that indicates indicating 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

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

# 1. Handling geometry parameters
# Simple
ee_x <- st_read(system.file("shape/nc.shp", package = "sf")) %>%
  sf_as_ee()

Map$centerObject(eeObject = ee_x)
Map$addLayer(ee_x)

# 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)

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

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

```
# 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
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
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(max_attempts = Inf)

## 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,
  predefinedAcl = "bucketLevel",
  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.
predefinedAcl	Specify user access to object. Passed to googleCloudStorageR::gcs_upload.
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_Initialize(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(max_attempts = Inf)

# 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))

## End(Not run)
```

Index