Package ‘rgeedim’

January 18, 2024

Type Package

Title Search, Composite, and Download 'Google Earth Engine' Imagery with the 'Python' Module 'geedim'

Version 0.2.7

Maintainer Andrew Brown <brown.andrewg@gmail.com>

URL https://humus.rocks/rgeedim/, https://github.com/brownag/rgeedim, https://geedim.readthedocs.io/

BugReports https://github.com/brownag/rgeedim/issues

Repository CRAN

Description Search, composite, and download 'Google Earth Engine' imagery with 'reticulate' bindings for the 'Python' module 'geedim' by Dugal Harris. Read the 'geedim' documentation here: <https://geedim.readthedocs.io/>.
Wrapper functions are provided to make it more convenient to use 'geedim' to download images larger than the 'Google Earth Engine' size limit <https://developers.google.com/earth-engine/apidocs/ee-image-getdownloadurl>.
By default the ``High Volume'' API endpoint <https://developers.google.com/earth-engine/cloud/highvolume> is used to download data and this URL can be customized during initialization of the package.

SystemRequirements Python (>= 3.6.0)

Config/reticulate list(packages = list(list(package = ``earthengine-api''), list(package = ``geedim'')))

License Apache License (>= 2)

Language en-US

RoxygenNote 7.2.3

Imports utils, methods, reticulate, jsonlite

Suggests terra, raster, tinytest, knitr, rmarkdown

Depends R (>= 3.5)

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation no

Author Andrew Brown [aut, cre],
Dugal Harris [cph] ('geedim' 'Python' module)

Date/Publication 2024-01-18 11:50:05 UTC

R topics documented:

`earthengine` *Get Earth Engine* `Module(earthengine-api)` *Instance*

Description

Gets the `earthengine-api` module instance in use by `geedim` package in current session.

`gd_ee_version()` Gets the `earthengine-api` version using `importlib.metadata.version()`

Usage

```
earthengine()

gd_ee_version()
```

Value

character. Version Number.

`gd_authenticate` *Authenticate with Google Earth Engine using* `gcloud`*, "Notebook Authenticator" or other method*

Description

Calls `ee.Authenticate(...)` to create a local instance of persistent credentials for Google Earth Engine. These credentials are used on subsequent calls to `ee.Initialize(...)` via `gd_initialize()`.

Usage

```
gd_authenticate(
  authorization_code = NULL,
  quiet = FALSE,
  code_verifier = NULL,
  auth_mode = NULL,
  scopes = NULL,
  force = TRUE
)
```

Arguments

`authorization_code`	Default: `NULL`
`quiet`	Suppress warnings, errors, messages? Default: `FALSE`
`code_verifier`	Code verifier (required if `authorization_code` is specified). Default: `NULL`
`auth_mode`	One of `"notebook"`, `"colab"`, `"gcloud"`, `"gcloud-legacy"` or (default) `NULL` to guess based on the current environment.
`scopes`	List of scopes to use for authentication. Defaults `NULL` corresponds to `c('https://www.googleapis.com` `'https://www.googleapis.com/auth/devstorage.full_control')`
`force`	Force authentication even if valid credentials exist? Default: `TRUE`

Details

This method should be called once to set up a machine/project with a particular authentication method.

- `auth_mode="gcloud"` (default) fetches credentials using `gcloud`. Requires installation of command-line Google Cloud tools; see `https://cloud.google.com/cli` for details. This mode will open a web page where you can sign into your Google Account, then a local JSON file will be stored in `gcloud` configuration folder with your credentials. These credentials will be used by any library that requests Application Default Credentials (ADC) which are preferred for long-term storage.
- `auth_mode="notebook"` argument is intended primarily for interactive or other short-term use. This mode will open a web page where you can sign into your Google Account to generate a short-term, revocable token to paste into the console prompt.

- `auth_mode="appdefault"` mode uses locally stored credentials `gcloud` configuration stored in 'application_default_credentials.json' or JSON file specified by `GOOGLE_APPLICATION_CREDENTIALS` environment variable.

Value

This function is primarily used for the side-effect of authentication with the 'Google Earth Engine' servers. Invisibly returns `try-error` on error.

Examples

```
## Not run:
# opens web page to complete authentication/provide authorization code
gd_authenticate(auth_mode = "notebook")

## End(Not run)
```

`gd_band_names` *Get Names of Layers in an Earth Engine Image*

Description

Calls `bandNames()` method from `ee.Image` class.

Usage

```
gd_band_names(x)
```

Arguments

`x`	a Google Earth Engine Image object, such as from `gd_image_from_id()`

Value

character. Vector of names of each layer in an image.

Examples

```
if (gd_is_initialized())
  gd_band_names(gd_image_from_id("USGS/NED"))
```

`gd_band_properties` *Get Properties of Layers in an Earth Engine Image*

Description

Gets combined Earth Engine and STAC properties.

Usage

```
gd_band_properties(x)
```

Arguments

`x`	a Google Earth Engine Image object, such as from `gd_image_from_id()`

Value

list. Each element is a list that corresponds to a layer in `x`, each with one or more elements for properties of that layer.

Examples

```
if (gd_is_initialized())
  gd_band_properties(gd_image_from_id("USGS/NED"))
```

`gd_bbox` *Prepare Bounding Box Region from X/Y Limits*

Description

Create a bounding box polygon Python object for use with `gd_download()`. The coordinates of the bounding box are expressed in WGS84 decimal degrees (`"OGC:CRS84"`).

Usage

```
gd_bbox(...)
```

Arguments

`...`	One or more `SpatRaster`, `SpatRasterCollection`, `SpatVector`, `SpatVectorProxy` or `SpatExtent` objects (whose combined bounding box extent will be returned); or the following *named* numeric arguments: `xmin`/`ymax`/`xmax`/`ymin`. If these four limit arguments are not named they should be in the stated order.

Details

Expecting total of 4 bounding box arguments, If arguments are unnamed they should be in the following order: "xmin", "ymax", "xmax", "ymin".

Value

a *list* object describing a GeoJSON bounding rectangular polygon suitable for use as `regions` argument to `gd_download()` or `gd_search()`

Examples

```
gd_bbox(
  xmin = 5.744140,
  ymax = 50.18162,
  xmax = 6.528252,
  ymin = 49.44781
)
```

`gd_composite` *Composite an Image Collection*

Description

Create a composite image from elements of an image collection.

Usage

```
gd_composite(x, ...)
```

Arguments

`x`	an object inheriting from `geedim.collection.MaskedCollection`, such as from `gd_search()` or `gd_collection_from_list()`
`...`	additional arguments to `geedim.collection.MaskedCollection$composite()`

Value

a composite `geedim.mask.MaskedImage` object

Examples

```
library(terra)

b <- terra::vect('POLYGON((-121.355 37.560,
                           -121.355 37.555,
                           -121.350 37.555,
                              -121.350 37.560,
                              -121.355 37.560))',
                    crs = "OGC:CRS84")

    if (gd_is_initialized())
      gd_composite(gd_search(gd_collection_from_name("USGS/3DEP/1m"),
                             region = b),
                   resampling = "bilinear")
```

gd_download *Download a Google Earth Engine Image*

Description

Download a Google Earth Engine Image

Usage

```
gd_download(
  x,
  filename = tempfile(fileext = ".tif"),
  region = NULL,
  composite = TRUE,
  overwrite = TRUE,
  silent = TRUE,
  ...
)
```

Arguments

`x,`	ID or Name, or a reference to an object inheriting from `geedim.download.BaseImage` or `geedim.collection.MaskedCollection`
`filename`	path to output file, defaults to temporary GeoTIFF file path; if `composite=FALSE` then this path should be to a parent directory. File names will be calculated from the internal name of the image and the requested scale.
`region`	a GeoJSON-like list, or other R spatial object describing region of interest, see `gd_region()` and `gd_bbox()` for details. `NULL` region (default) will download the whole image.
`composite`	logical. Composite Image Collection into single image for download? Default: `TRUE`
`overwrite`	Overwrite existing file? Default: `TRUE`
`silent`	Silence errors? Default: `TRUE`
`...`	Additional arguments (e.g. `scale`) passed to `geedim.mask.MaskedImage$download(...)` and, when `composite=TRUE`, `geedim.collection.MaskedCollection$composite()`

Details

The region argument is *optional* for downloading images. When downloading a composite Image Collection, you must specify region, scale and crs arguments. When downloading an image collection as a set of GeoTIFF files (composite=FALSE), then filename is the destination directory, and scale must be specified. The default resampling method in geedim is resampling="near" (Nearest Neighbor). Other options for resampling include: "average", "bicubic", "bilinear". See gd_resampling_methods().

Value

Invisible path to downloaded image, or try-error on error

See Also

gd_region() gd_bbox()

Examples

```
 r <- gd_bbox(
   xmin = -121,
   xmax = -120.5,
   ymin = 38.5,
   ymax = 39
 )

if (gd_is_initialized()) {
 x <- gd_image_from_id('CSP/ERGo/1_0/Global/SRTM_topoDiversity')
 tf <- tempfile(fileext = ".tif")

 # fast sample download at 10x aggregation (900m v.s. 90m)
 img <- gd_download(x, filename = tf,
                    region = r, scale = 900,
                    overwrite = TRUE, silent = FALSE)

 if (requireNamespace("terra")) {
   library(terra)
   f <- rast(img)
   plot(f[[1]])
   # inspect object
   f
 }
 unlink(tf)
}
```

gd_enum_names — geedim *Enums*

Description

geedim Enums

Usage

```
gd_enum_names()

gd_enum_elements(enum = gd_enum_names())

gd_resampling_methods()

gd_cloud_mask_methods()

gd_composite_methods()

gd_export_types()

gd_spectral_distance_metrics()
```

Arguments

`enum`	Enum name, one or more of: `"CloudMaskMethod"`, `"CompositeMethod"`, `"ResamplingMethod"`

Value

`gd_enum_names()`: character vector containing names of Enums

`gd_enum_elements()`: element values of an Enum

`gd_resampling_methods()`: character vector of resampling methods (Enum `"ResamplingMethod"`)

`gd_cloud_mask_methods()`: character vector of cloud mask methods (Enum `"CloudMaskMethod"`)

`gd_composite_methods()`: character vector of composite methods (Enum `"CompositeMethod"`)

`gd_export_types()`: character vector of export types (Enum `"ExportType"`)

`gd_spectral_distance_metrics()`: character vector of spectral distance metrics (Enum `"SpectralDistanceMetric"`)

Examples

```
if (gd_is_initialized())
 gd_enum_names()
```

```
if (gd_is_initialized())
 gd_enum_elements()
```

```
if (gd_is_initialized())
 gd_resampling_methods()
```

```
if (gd_is_initialized())
 gd_cloud_mask_methods()
```

```
if (gd_is_initialized())
 gd_composite_methods()
```

```
if (gd_is_initialized())
 gd_export_types()
```

```
if (gd_is_initialized())
 gd_spectral_distance_metrics()
```

`gd_export` *Export image to Earth Engine Asset, Google Cloud Storage Bucket, or Google Drive*

Description

Exports an encapsulated image to the destination specified by `type`, `folder` and `filename`

Usage

```
gd_export(
  x,
  filename,
  type = "drive",
  folder = dirname(filename),
  region,
  wait = TRUE,
  ...
)
```

Arguments

`x`	An object that inherits from `geedim.download.BaseImage`
`filename`	Output filename. If `type` is `"asset"` and `folder` is not specified, `filename` should be a valid Earth Engine asset ID.
`type`	Export type. Defaults to `"drive"`; other options include `"asset"`, and `"cloud"`. See `gd_export_types()`
`folder`	Destination folder. Defaults to `dirname(filename)`.
`region`	Region e.g. from `gd_bbox()` or `gd_region()`
`wait`	Wait for completion? Default: `TRUE`
`...`	Additional arguments to `geedim.download.BaseImage.export()`

Details

See the geedim.mask.MaskedImage.export() documentation for details on additional arguments. Requires 'geedim' >1.6.0.

Value

an `ee.batch.Task` object

Examples

```
## Not run:
if (gd_is_initialized()) {
 r <- gd_bbox(
   xmin = -120.6032,
   xmax = -120.5377,
   ymin = 38.0807,
   ymax = 38.1043
 )

 i <- gd_image_from_id('CSP/ERGo/1_0/US/CHILI')

 ## export to Google Drive (default `type="drive"`)
 # res <- gd_export(i, filename = "RGEEDIM_TEST.tif", scale = 100, region = r)

 ## export to `type="asset"`, then download by ID (stored in project assets)
 # res <- gd_export(
 #   i,
 #   "RGEEDIM_TEST",
 #   type = "asset",
 #   folder = "your-project-name",
 #   scale = 100,
 #   region = r
 # )
 # gd_download("projects/your-project-name/assets/RGEEDIM_TEST", filename = "test.tif")

 ## export to Google Cloud Bucket with `type="cloud"`,
 ##   where `folder` is the bucket path without `"gs://"`
 # res <- gd_export(i, filename = "RGEEDIM_TEST.tif", type = "cloud",
 #                  folder = "your-bucket-name", scale = 100, region = r)
}

## End(Not run)
```

gd_footprint *Get Footprint of Masked Image*

Description

Gets GeoJSON-style list containing footprint of a `geedim.mask.MaskedImage` object

Usage

```
gd_footprint(x)
```

Arguments

x	a `geedim.mask.MaskedImage` object

Value

list.

Examples

```
if (gd_is_initialized())
  gd_footprint(gd_image_from_id("USGS/NED"))
```

gd_get_asset *Get, Update, or Delete an Earth Engine Asset by ID*

Description

Get, Update, or Delete an Earth Engine Asset by ID

Usage

```
gd_get_asset(x, silent = FALSE)

gd_update_asset(
  x,
  asset,
  update = c("start_time", "end_time", "properties"),
  silent = FALSE
)

gd_delete_asset(x, silent = FALSE)
```

Arguments

`x`	Asset ID name
`silent`	Silence errors? Default: `FALSE`
`asset`	Used only for `gd_update_asset()`: a named list, with names representing elements of `x` to replace. Only `"start_time"`, `"end_time"`, and `"properties"` fields can be updated.
`update`	Used only for `gd_update_asset()`: A character vector of field names to update. Default: `"start_time"`, and `"end_time"` to update timestamps; and `"properties"` to update all properties.

Value

`try-error` on error. `gd_get_asset()`: a named list containing information and properties of an Earth Engine asset

`gd_update_asset()`: This function is called for side-effects (updates the specified asset fields)

`gd_delete_asset()`: This function is called for side-effects (deletes the specified asset)

Examples

```
## Not run:
# get asset from project by ID
a <- gd_get_asset("projects/your-project-name/assets/YOUR_ASSET_ID")

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

```
# change description in `"properties"`
a$properties$description <- "foo"

# update asset
gd_update_asset("projects/your-project-name/assets/YOUR_ASSET_ID", a, "properties")

## End(Not run)
## Not run:
# remove an asset from project
gd_delete_asset("projects/your-project-name/assets/YOUR_ASSET_ID")

## End(Not run)
```

gd_image_from_id — *Reference Google Earth Engine Image or Image Collection by ID or Name*

Description

Create references to a Google Earth Engine Image or Image Collection based on IDs or names, or combine Images into Image Collections.

Usage

```
gd_image_from_id(x)

gd_collection_from_name(x)

gd_collection_from_list(x)

gd_asset_id(filename, folder = NULL)

gd_list_assets(parent)
```

Arguments

`x`	character. `id` of Image, `name` of Image Collection, or a vector of Image `id` to create a new Image Collection
`filename`	File or Asset Name
`folder`	Optional: Project Name
`parent`	Full path to project folder (with or without `"/assets"` suffix)

Value

`geedim.MaskedImage` or `geedim.MaskedCollection` object, or `try-error` on error

Examples

```
if (gd_is_initialized())
  gd_image_from_id('CSP/ERGo/1_0/Global/SRTM_topoDiversity')




if (gd_is_initialized())

  # Find 1m DEMs in arbitrary extent
  r <- gd_bbox(xmin = -121.4, xmax = -121.35, ymin = 37.55, ymax = 37.6)

  # collection of individual tiles of DEM
  x <- gd_collection_from_name("USGS/3DEP/1m")

  # search within region
  y <- gd_search(x, r)

  gd_properties(y)




if (gd_is_initialized())
  # Find 1m DEM in arbitrary extent
  r <- gd_bbox(xmin = -121.4, xmax = -121.35, ymin = 37.55, ymax = 37.6)

  # collection of individual tiles of DEM
  x <- gd_collection_from_name("USGS/3DEP/1m")

  # search within region
  y <- gd_search(x, r)

  # select images with some condition of interest
  z <- subset(gd_properties(y),
              grepl("UpperSouthAmerican_Eldorado_2019", id) > 0)

  # create encapsulated images from IDs returned by search
  l <- lapply(z$id, gd_image_from_id)

  # create a new collection from the list of images
  l2 <- gd_collection_from_list(l)
  l2

### download composite of custom collection
#  gd_download(gd_composite(l2),
#              filename = "test.tif",
#              region = r,
#              crs = "EPSG:5070",
#              scale = 30)
```

```
if (gd_is_initialized())
  gd_asset_id("RGEEDIM_TEST", "your-project-name")
```

```
if (gd_is_initialized())
  gd_list_assets("projects/your-project-name")
```

---

gd_initialize *Initialize* geedim

Description

Calls geedim `Initialize()` method. This method should be called at the beginning of each session.

Usage

```
gd_initialize(
  private_key_file = NULL,
  credentials = "persistent",
  cloud_api_key = NULL,
  url = "https://earthengine-highvolume.googleapis.com",
  opt_url = NULL,
  http_transport = NULL,
  project = NULL,
  quiet = TRUE
)

gd_is_initialized(...)
```

Arguments

`private_key_file`
: character. Optional: Path to JSON file containing client information and private key. Alternately, the contents of a JSON file. Instead of setting this argument you may specify `EE_SERVICE_ACC_PRIVATE_KEY` environment variable with path to JSON file.

`credentials`
: Default: `'persistent'` uses credentials already stored in the filesystem, or raise an explanatory exception guiding the user to create those credentials.

`cloud_api_key`	An optional API key to use the Cloud API. Default: `NULL`.
`url`	The base url for the EarthEngine REST API to connect to. Defaults to "High Volume" endpoint: `"https://earthengine-highvolume.googleapis.com"`
`opt_url`	(deprecated) Use `url`.
`http_transport`	The HTTP transport method to use when making requests. Default: `NULL`
`project`	The client project ID or number to use when making API calls. Default: `NULL`
`quiet`	Suppress error messages on load? Default: `FALSE`
`...`	Additional arguments passed to `gd_initialize()`

Value

`gd_initialize()`: try-error (invisibly) on error.

`gd_is_initialized()`: logical. `TRUE` if initialized successfully.

See Also

`gd_authenticate()`

Examples

```
## Not run:
gd_initialize()

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

`gd_install` *Install Required Python Modules*

Description

This function installs the latest `numpy`, `earthengine-api`, and `geedim` modules. The default uses `pip` for package installation. You can configure custom environments with `pip=FALSE` and additional arguments that are passed to `reticulate::py_install()`.

Usage

```
gd_install(pip = TRUE, system = FALSE, force = FALSE, ...)
```

Arguments

`pip`	Use `pip` package manager? Default: `TRUE`. To use a virtual or conda environment specify `method="virtualenv"` or `method="conda"`, respectively. See details.
`system`	Use a `system()` call to `python -m pip install --user ...` instead of `reticulate::py_install()`. Default: `FALSE`.
`force`	Force update (uninstall/reinstall) and ignore existing installed packages? Default: `FALSE`. Applies to `pip=TRUE`.
`...`	Additional arguments passed to `reticulate::py_install()`

Details

This function provides a basic wrapper around `reticulate::py_install()`, except it defaults to using the Python package manager `pip`. If you specify `method="virtualenv"` or `method="conda` then the default `envname` is `"r-reticulate"` unless you set it to something else. If an environment of that name does not exist it is created.

Value

`NULL`, or `try-error` (invisibly) on R code execution error.

Examples

```
## Not run:

# install with pip (with reticulate)
gd_install()

# use virtual environment with default name "r-reticulate"
gd_install(method = "virtualenv")

# use "conda" environment named "foo"
gd_install(method = "conda", envname = "foo")

# install with pip (system() call)
gd_install(system = TRUE)


## End(Not run)
```

`gd_mask_clouds` *Mask Clouds or Apply Fill Mask*

Description

Apply the cloud/shadow mask if supported, otherwise apply the fill mask.

Usage

```
gd_mask_clouds(x)
```

Arguments

x	a `geedim.mask.MaskedImage`

Value

a `geedim.mask.MaskedImage`

`gd_projection` *Get Projection Information from Google Earth Engine Asset*

Description

Get Projection Information from Google Earth Engine Asset

Usage

```
gd_projection(x)
```

Arguments

x	character ID referencing asset, or an image object (subclass of `ee.image.Image` or `geedim.download.BaseImage`)

Value

`ee.Projection` object

Examples

```
if (gd_is_initialized())
  gd_projection(gd_image_from_id('CSP/ERGo/1_0/Global/SRTM_topoDiversity'))
```

gd_properties *Get Properties of an Image Collection*

Description

Get Properties of an Image Collection

Usage

```
gd_properties(x)
```

Arguments

x	`geedim.collection.MaskedCollection` object

Value

`data.frame` containing properties table from `x`; `NULL` if no properties table.

Examples

```
library(terra)

b <- terra::vect('POLYGON((-121.355 37.560,
                           -121.355 37.555,
                           -121.350 37.555,
                           -121.350 37.560,
                           -121.355 37.560))',
                 crs = "OGC:CRS84")

if (gd_is_initialized()) {
  x <- gd_search(gd_collection_from_name("USGS/3DEP/1m"),
                 region = gd_region(b))
  gd_properties(x)
}
```

gd_region *Create GeoJSON Region from R Spatial Objects*

Description

Creates a suitable input for the `region` argument to `gd_download(<Image>)` or `gd_search()` for Image Collections.

`gd_region_to_vect()` is the inverse function of gd_region/gd_bbox; convert GeoJSON-like list to Well-Known Text(WKT)/*SpatVector*. This may be useful, for example. when `gd_region()`-output was derived from an Earth Engine asset rather than local R object.

Usage

```
gd_region(x)

gd_region_to_vect(x, crs = "OGC:CRS84", as_wkt = FALSE, ...)
```

Arguments

`x`	either a WKT string (character), a SpatRaster(Collection)/SpatVector(Collection)/SpatExtent, an sf object, an Spatial* object or a RasterLayer/RasterStack.
`crs`	character. Default for GeoJSON sources is `"OGC:CRS84"`.
`as_wkt`	logical. Return Well-Known Text (WKT) string as character? Default: `FALSE` returns a 'terra' *SpatRaster*.
`...`	Additional arguments to `gd_region_to_vect()` are passed to `terra::vect()` when `as_wkt=FALSE` (default).

Details

If `x` is an R spatial object, each vertex (possibly after converting object extent to vector) is used to create the GeoJSON object. Otherwise, the extent is determined and passed to `gd_bbox()`.

Value

list representing a GeoJSON extent

`gd_region_to_vect()`: a 'terra' *SpatVector* object, or *character* containing Well-Known Text.

See Also

`gd_bbox()`

Examples

```
library(terra)

b <- terra::vect('POLYGON((-121.355 37.560,
                           -121.355 37.555,
                           -121.350 37.555,
                           -121.350 37.560,
                           -121.355 37.560))',
                 crs = "OGC:CRS84")

gd_region(b)
```

`gd_search` *Search an Image Collection*

Description

Search an Image Collection

Usage

```
gd_search(
  x,
  region,
  start_date = "2000-01-01",
  end_date = as.character(Sys.Date()),
  ...
)
```

Arguments

x	`geedim.collection.MaskedCollection` object
region	list / Python GeoJSON object describing region, e.g. as created by `gd_bbox()`
start_date	Default: `'2020-01-01'`
end_date	Default: `Sys.Date()`
...	additional arguments to `geedim.MaskedCollection.search()` e.g. `cloudless_portion`, `fill_portion`

Value

`geedim.MaskedCollection` object suitable for querying properties

Examples

```
b <- terra::vect('POLYGON((-121.355 37.56,-121.355 37.555,
                    -121.35 37.555,-121.35 37.56,
                    -121.355 37.56))',
          crs = "OGC:CRS84")
if (gd_is_initialized())
  gd_search(gd_collection_from_name("USGS/3DEP/1m"),
            region = gd_region(b))
```

`gd_task_status` *Get Earth Engine Task Status*

Description

`gd_task_status()` and `gd_task_uri()` are helper functions for working with tasks scheduled with `gd_export()`

Usage

```
gd_task_status(x)

gd_task_uri(x, asset_only = TRUE)
```

Arguments

`x`	An object of class `"ee.batch.Task"`
`asset_only`	Default: `TRUE`. For export tasks with `type="asset"`, return only the asset ID, rather than whole URL. Other export task types return a full path to either Google Drive or Google Cloud location. When `FALSE` the path is a HTTPS link to an Earth Engine asset.

Value

`gd_task_status()`: returns the status from an `"ee.batch.Task"` object

`gd_task_uri()`: returns the destination URI(s) associated with a task.

See Also

`gd_export()` `gd_download()`

Examples

```
## Not run:
if (gd_is_initialized()) {
  r <- gd_bbox(
    xmin = -120.6032,
    xmax = -120.5377,
    ymin = 38.0807,
    ymax = 38.1043
  )

  i <- gd_image_from_id('CSP/ERGo/1_0/US/CHILI')
  ex <- gd_export(
    i,
    region = r,
    type = "asset",
    filename = "RGEEDIM_TEST",
    folder = "your-project-name",
    scale = 30
  )

  gd_task_status(ex)

  r <- gd_download(
    gd_task_uri(ex),
    filename = "image.tif",
    region = r,
    overwrite = TRUE
  )

  library(terra)
  plot(rast(r))
}

## End(Not run)
```

`geedim` `Module(geedim)` - *Get* `geedim` *Module Instance*

Description

Gets the `geedim` module instance in use by the package in current **R**/`reticulate` session.

Usage

```
geedim()

gd_version()
```

Value

character. Version Number.

Index