Google Earth Engine with rgee : : CHEAT SHEET

Mission

The goal of rgee is to offer a user-friendly interface for analyzing spatial data on the Google Earth Engine (GEE) platform using the R programming language.

Installation

Windows

■ **It is necessary to have Rtools installed.**

```
> install.package ("rgee")
```

Mac OS

■ **In terminal execute, as follow:**

Linux

```
user:~$ sudo apt install  libjq-dev
user:~$ sudo apt install  libprotobuf-dev
user:~$ sudo apt install  protobuf-compiler
> install.package ("rgee")
```

Docker

■ **In terminal execute, as follow:**

```
user:~$ docker run -d -p 8787:8787 -e
USER=rgee -e PASSWORD=rgee --name
rgee-dev csaybar/rgee
```

For **Python requirements installation**, use **ee_install**:

```
> rgee::ee_install()
```

only run once rgee is installed

See the **Python** section in **rgeebook** for more details.

Hello world Earth Engine

```
> library("rgee")
> ee_Initialize(user, drive , gcs)
```

GEE username (Optional)

Connect GEE with GD.

Connect GEE with GCS.

```
# Earth Engine API style (chaining methods)
> ee$String("Hello World from Earth Engine!")$
  getInfo()
```

Fetch and return information. From GEE server to local.

```
> [1] "Hello World from Earth Engine!"
```

Pipe integration %>%

Pipe operator has been included into rgee to provide functional programming style.

```
# Earth Engine API with pipes style
> ee$String("Hello World from Earth Engine!")%>%
ee$String$getInfo()
```

```
> [1] "Hello World from Earth Engine!"
```

Basic classes

Basic data structures available in GEE..

Type	Class	Example
Number	ee$Number	`> ee$Number(2021)`
String	ee$String	`> ee$String("Hello")`
List	ee$List	`> ee$List(c("Hi","amy"))`
Dictionary	ee$Dictionary	`> ee$Dictionary(list(year = 2021))`
Array	ee$Array	`> ee$Array(26, 9, 2021)`
Date	ee$Date	`> ee$Date("1990-01-01")`

ee$Geometry

A collection of geometric forms that describe an object spatially.

Type	Geom	Function
Point		**ee$Geometry$Point** *sf:: st_point*
LineString		**ee$Geometry$LineString** *sf:: st_linestring*
LineRing		**ee$Geometry$LineRing** *sf:: st_linestring*
Polygon		**ee$Geometry$Polygon** *sf:: st_polygon*
Multipoint		**ee$Geometry$Multipoint** *sf::st_multipoint*
MultiLineString		**ee$Geometry$MultiLineString** *sf::st_multilinestring*
MultiGeometry		**ee$Geometry$MultiGeometry** *sf::st_geometrycollection*

Geometric operations

Type	Function
Buffer	*$buffer
Intersection	*$intersection
Union	*$union
Difference	*$difference
Symmetric difference	*$symmetricdifference

(* : The symbol mean is a type of GEE geometry, for example : a ee$Geometry$Polygon)

Data catalog

The Earth Engine catalogue can be accessed interactively from R with rgee.

Function	Example
ee_utils_dataset_display	`> ee_utils_dataset_display("Landsat")`

Visualization

rgee supports the visualization of spatial Earth Engine objects such as Image, ImageCollection, Feature, FeatureCollection, and allows users to customize the legend using the **Map$addLegend** method.

Object	Geom	Method	Arguments
Image		**Map$addLayer**	■ **eeObject*** ■ **VisParams** ■ **name** ■ **show** ■ **opacity**
Feature			
FeatureCollection			
ImageCollection		**Map$addLayers**	■ **nmax**

* ***eeObject*** can also be a Cloud Optimized GeoTIFF (COG) file.

Map$Legend needs that users pass the same *visParams* used in **Map$addLayer**.

Data	Function	Type
Categorical	Map$addlegend(...) ■ **visParams** ■ **name** ■ **position** ■ **color_mapping** ■ **opacity**	color_mapping = "categorical"
Continue		color_mapping = "continue"
Discrete		color_mapping = "discrete"
Customize		color_mapping = "character"

Example

```
> image <- ee$Image$Dataset$CGIAR_SRTM90_V4
> visparams <- list(min = 0, max = 3000)
> m1 <- Map$addLayer(image, visparams, "DEM")
> m1 + Map$addLegend(visparams, "DEM", "bottomright", 8)
```

rgee also supports the metadata display of GEE spatial objects (ee_print).

```
> ee$Image("CGIAR/SRTM90_V4") %>% ee_print(srtm)
```

Google Earth Engine with rgee : : CHEAT SHEET

Considerations

Some issues can occurs when reticulate translate the R code into Python. We detected four cases:

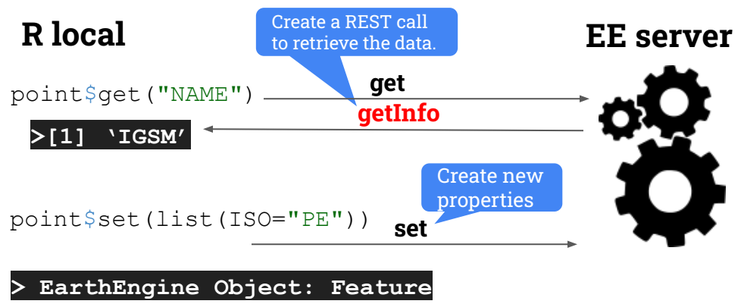
1. map method in **ee$List** objects.	**Solution:** *Use ee_utils_pyfunc.*
2. Strict integer number data type.	**Solution:** *Add "L" at the end. For instace:* > ee$Number(20L)
3. Be careful with **ee$Date** objects.	**Solution:** *Use eedate_to_rdate and rdate_to_eedate.*
4. Reserved words.	**Solution:** *Use quotation marks. For instace:* > x$'repeat'(20, 2)

ee$Feature

It is an GEE geometry + properties.

```
> xy <- c(-77.08643, -12.05536)
> geom <-  ee$Geometry$Point(xy)
> props <- list(ID = 1, NAME = IGSM)
> point <- ee$Feature(geom, props)
```

Longitude and latitude in a vector




```
> EarthEngine Object: Feature
```

ee$FeatureCollection

It is an set of GEE features + properties.

min-max coords for create new rectangle

```
> minmax <- c(-77.08, -12.05, -77.08, -12.05)
> box <- ee$Geometry$Rectangle(minmax)
> lf1 <- ee$Feature(box, list(ISO="PE"))
> lf2 <- ee$Feature(box, list(ISO="RU"))
> prps <- list(ID=1, NAME="polygons")
> fc <- ee$FeatureCollection(c(lf1,lf2), prps)
> print(fc)
```

```
> EarthEngine Object: FeatureCollection
```

ee$Image

It is an set of bands. An band is array of values + properties.

```
> image1 <- ee$Image(1)
> image1
```

Create a constant image

```
> EarthEngine Object: Image
```

```
> image2 <- ee$Image(2)
> list_img <- list(image1, image2)
> image3 <- ee$Image(list_img)
> image3
```

Concatenate two single-band images into one multi-band image

```
> EarthEngine Object: Image
```

Image I/O

Functions	FROM	TO	RETURN
ee_as_raster	EE server	Local	R object
ee_image_to_asset	EE server	EE asset	Unstarted task
ee_image_to_gcs	EE server	GCS	Unstarted task
ee_image_to_drive	EE server	GD	Unstarted task
ee_as_stars	EE server	Local	R object
raster_as_ee	Local	EE server	GEE object
stars_as_ee	Local	EE server	GEE object

ee$ImageCollection

It is an set of GEE images + properties.

```
> ic <- ee$ImageCollection(list_img)
> ic
```

```
> EarthEngine Object: ImageCollection
```

ImageCollection I/O

Functions	FROM	TO	RETURN
ee_get_date_ic	EE server	Local	R data.frame
ee_imagecollection_to_local	EE server	Local	R object

FeatureCollection Export (Table)

Set of functions to fetch and return GEE FeatureCollections.

Functions	FROM	TO	RETURN
gcs_to_ee_table	GCS	EE server	Unstarted task
ee_as_sf	EE server	Local	R object
ee_table_to_drive	EE server	GD	Unstarted task
ee_table_to_gcs	EE server	GCS	Unstarted task
ee_table_to_asset	EE server	EE asset	Unstarted task
sf_as_ee	Local	EE server	GEE object

GEE Asset Manager

Set of functions to interact with the GEE asset manager. Batch operations are supported.

FUNCTIONS	DESCRIPTION
ee_manage_create	Create an empty folder or ic.
ee_manage_delete	Delete an GEE asset.
ee_manage_assetlist	List files in a folder or ic.
ee_manage_quota	Show user GEE quota.
ee_manage_copy	Copy a paste GEE asset.
ee_manage_move	Cut and paste a GEE asset.
ee_manage_set_properties	Set GEE asset properties.
ee_manage_delete_properties	Delete GEE asset properties.
ee_manage_asset_access	Change IAM policy.
ee_manage_task	Show the task's user history.
ee_manage_cancel_all_running_task	Cancel all the running task.

Custom Animations

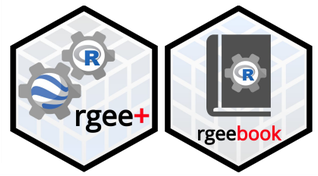
Auxiliary functions to create GIF files with Earth Engine. They depend of the **magick** package. **rgeeExtra** now include these functions.

FUNCTIONS	DESCRIPTION
ee_utils_gif_annotate	Add text to a GIF.
ee_utils_gif_creator	From ee$ImageCollection to GIF.
ee_utils_gif_save	Write a magick object as a GIF file.

Miscellaneous

FUNCTIONS	DESCRIPTION
ee_utils_create_json	Convert a R list into a JSON.
ee_utils_create_manifest_image	GEE Image manifest creator.
ee_utils_create_manifest_table	GEE Table manifest creator.
ee_utils_dataset_display	Search into the GEE Data Catalog.
ee_utils_future_value	Return the future values object.
ee_utils_get_crs	Convert SR-ORG into a OGC WKT.
ee_utils_py_to_r	Translate Python objects to R.
ee_utils_pyfunc	Wrap a R function in Python.
ee_utils_shp_to_zip	Create a zip from an sf object.
ee_utils_cog_metadata	Metadata of a COG tile server.

Others



This cheatsheet was created using the rgee reference manual and the rgee vignettes. Visit the rgeebook for additional information about this package.