rgee: An R package for interacting with Google Earth Engine

April 30, 2020

Summary

Google Earth Engine (GEE) (Gorelick et al. 2017) is a cloud-based platform specifically designed for reproducible planetary-scale environmental data analysis. Currently, GEE is made up of 3 components. The data catalog which is continuously updated and permits users to access a dataset of over 40 years of satellite imagery for the whole world. The Google's geocomputational infrastructure, highly optimized, reducing mostly the time execution of spatial non-recursively procedures. Finally, the Web REST API and the two client libraries (in JavaScript and Python) which permits users to interact with the server-side without the necessity of understanding the complex system architecture and data distributions models behind GEE. Although the GEE functionality is powerful with more than 800 functions, and the possibility of chaining operations, there are limitations to creating straightforward input/output pipelines, quality static visualization, metadata display, and efficient management of Earth Engine asset resources. This becomes a more challenging task outside the Python Earth Engine API (Markert 2019).

This paper introduces **rgee**, an Earth Engine client library for R. All the classes and the existing functionality of the two Google's supported client libraries can be called through the dollar sign (`$`). `rgee` adds several new features such as (i) new I/O design, (ii) multiple user support, (iii) easily extraction of time series and zonal statistics, (iv) asset manage interface, and (v) metadata display, also with `rgee` is possible the execution of Earth Engine Python code from within R which make the translation of large Python projects unnecessary. The goal of **rgee** is to allows users to leverage the strengths of the R spatial ecosystem and Google Earth Engine in the same workflow.

Features

I/O Enhanced

rgee implements several functions to support the download/upload of image and vector datasets (Table 1 and Figure 1). For instance, to download images located in the server-side you might use either `ee_image_as_raster` or `ee_image_as_stars`. All the download functions implemented in `rgee` have the option to download via using an intermediate container (Google Drive or Google Cloud Storage) or a REST call ("$getInfo"). Although the last option permits users a direct download, there is a limitation of 262144 pixels (for images) or 5000 elements (for featurecollections) by request which makes it not recommendable for large objects. The upload process follows the same path. In **rgee** we implement `raster_as_ee`, `stars_as_ee` for upload image and `sf_as_ee` for vector data. Large uploads are just possible through a Google Cloud Storage account active.

		from	to	return
Download	Image	ee_image_to_drive	EE server-side	Google Drive	Unstarted task
		ee_image_to_gcs	EE server-side	Google Cloud Storage	Unstarted task
		ee_image_to_asset	EE server-side	EE asset	Unstarted task
		ee_as_raster	EE server-side	Local	RasterStack object

			from	to	return
		ee_as_stars	EE server-side	Local	Proxy-stars object
	Table	ee_table_to_drive	EE server-side	Google Drive	Unstarted task
		ee_table_to_gcs	EE server-side	Google Cloud Storage	Unstarted task
		ee_table_to_asset	EE server-side	EE asset	Unstarted task
		ee_as_sf	EE server-side	Local	sf object
	Generic	ee_drive_to_local	Google Drive	Local	object filename
		ee_gcs_to_local	Google Cloud Storage	Local	object filename
Upload	Images	gcs_to_ee_image	Google Cloud Storage	EE asset	ee.Image object
		raster_as_ee	Local	EE asset	ee.Image object
		stars_as_ee	Local	EE asset	ee.Image object
	Table	gcs_to_ee_table	Google Cloud Storage	EE asset	ee.FeatureCollection object
		sf_as_ee	Local	EE asset	ee.FeatureCollection object
	Generic	local_to_gcs	Local	Google Cloud Storage	GCS filename

Multiple users

Extraction of time series

```
library(rgee)
library(sf)

ee_Initialize()

# Define a Image or ImageCollection e.g. Terraclimate
# Mean composite
terraclimate <- ee$ImageCollection("IDAHO_EPSCOR/TERRACLIMATE")$
  filterDate("2001-01-01", "2002-01-01")$
  map(function(x) x$select("pr"))$
  mean()$rename("pp_mean")

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

# Extract the average areal rainfall
ee_nc_rain <- ee_extract(terraclimate, nc, sf = TRUE)
```

rgee asset Manage Interface

Metadata display

Availability

`rgee` is open source software made available under the Apache v2 license. It can be installed through CRAN (——) using: install.packages("——"). `rgee` can also be installed from its GitHub repository using the remotes package: remo tes::install_github("——-").

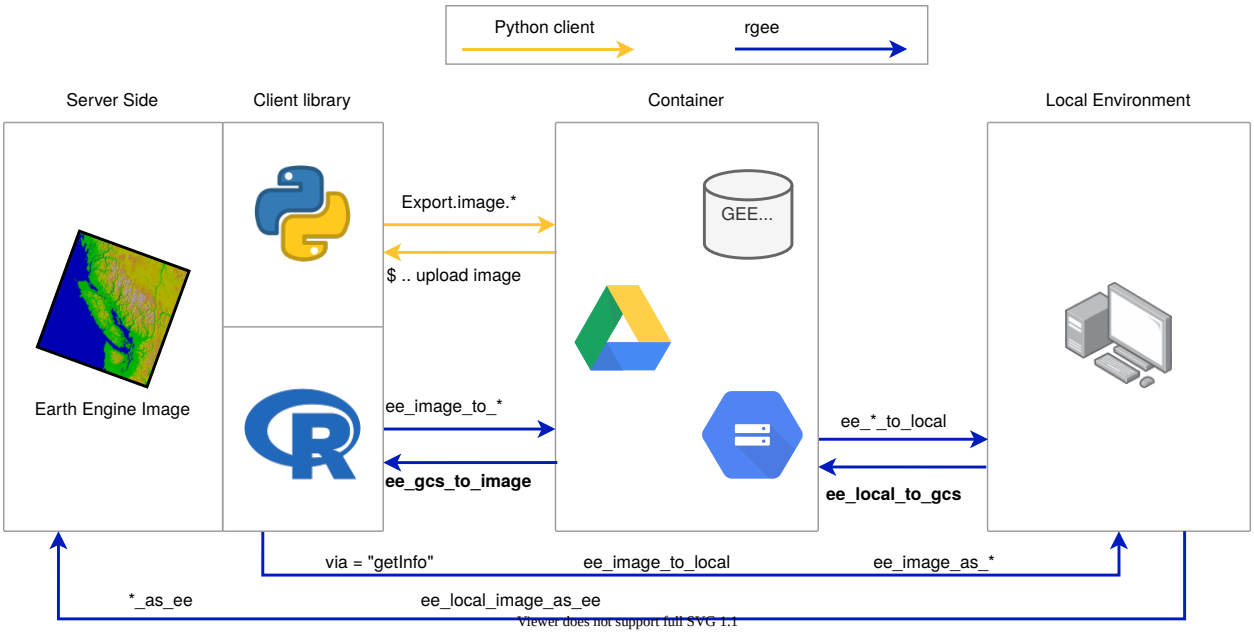

Figure 1: Comparison between the Python and R Earth Engine client libraries to transfer data from server to client-side and vice-versa

References

Gorelick, Noel, Matt Hancher, Mike Dixon, Simon Ilyushchenko, David Thau, and Rebecca Moore. 2017. "Google Earth Engine: Planetary-Scale Geospatial Analysis for Everyone." *Remote Sensing of Environment* 202. Elsevier: 18–27.

Markert, Kel. 2019. "Cartoee: Publication Quality Maps Using Earth Engine." *Journal of Open Source Software* 4 (33): 1207.