The Journal of Open Source Software

eemont: A Python package that extends Google Earth Engine

David Montero[1]

1 Independent Researcher

DOI: 10.21105/joss.03168

Software

- Review
- Repository
- Archive

Editor: Katy Barnhart

Reviewers:

- @giswqs
- @elbeejay
- @patrickcgray

Submitted: 27 March 2021
Published: 08 June 2021

Summary

Google Earth Engine (GEE) is a cloud-based service for geospatial processing of vector and raster data (Gorelick et al., 2017). The GEE platform has JavaScript and Python APIs with different methods to process geospatial objects. GEE also provides a multi-petabyte data catalog of geospatial datasets and satellite imagery (e.g., Landsat, Sentinel, MODIS). The eemont package extends the GEE Python API with pre-processing and processing tools for the commonly used satellite imagery (e.g., Landsat, Sentinel, MODIS) by adding new methods for different Earth Engine Objects that are friendly with the Python method chaining. The package can be used with geemap (Wu, 2020) for interactive visualization in Jupyter Notebooks, with the GEE Plugin for QGIS for processing and visualization inside QGIS (QGIS Development Team, 2021), and it can be used in R with rgee (Aybar et al., 2020).

Statement of need

The typical pre-processing and processing steps of satellite imagery are long and complex, making it challenging for analysts to move from data curation to analysis. These steps have been simplified through eemont with simple and clearer pythonic methods by extending the main Earth Engine objects.

The eemont python package extends the following Earth Engine classes:

- ee.Feature
- ee.FeatureCollection
- ee.Geometry
- ee.Image
- ee.ImageCollection

New utility methods and constructors are added to above-mentioned classes in order to create a more fluid code by being friendly with the Python method chaining. The added methods are useful for pre-processing and processing tasks (e.g., clouds masking, shadows masking, image scaling, spectral indices computation, time series, etc.), and they are presented as simple functions that give researchers, students and analysts the chance to process a large number of geospatial datasets with a few lines of code, improving code-writing and producing analysis-ready geospatial datasets.

The following script shows an example of the required code to pre-process and process the Landsat-8 Surface Reflectance Image Collection using the standard GEE Python API:

```
import ee

ee.Authenticate()
ee.Initialize()

point = ee.Geometry.Point([-74.0592,11.3172])

def maskClouds(img):
    cloudShadowBitMask = (1 << 3)
    cloudsBitMask = (1 << 5)
    qa = img.select('pixel_qa')
    cloudShadowMask = qa.bitwiseAnd(cloudShadowBitMask).eq(0)
    cloudMask = qa.bitwiseAnd(cloudsBitMask).eq(0)
    mask = cloudShadowMask.And(cloudMask)
    return image.updateMask(mask)

def scaleImage(img):
    sc = img.select('B[1-7]').multiply(0.0001)
    sc = sc.addBands(img.select(['B10','B11']).multiply(0.1))
    sc = sc.addBands(img.select(['sr_aerosol','pixel_qa','radsat_qa']))
    return sc.copyProperties(img,img.propertyNames())

def addIndices(img):
    NDVI = img.normalizedDifference(['B5','B4']).rename('NDVI')
    imgDict = {
        'N': img.select('B5'),
        'R': img.select('B4'),
        'B': img.select('B2')
    }
    formula = '2.5 * (N - R) / (N + 6.0 * R - 7.5 * B + 1.0)'
    EVI = img.expression(formula,imgDict).rename('EVI')
    GNDVI = img.normalizedDifference(['B5','B3']).rename('GNDVI')
    return img.addBands([NDVI,EVI,GNDVI])

L8 = (ee.ImageCollection('LANDSAT/LC08/C01/T1_SR')
    .filterBounds(point)
    .map(maskClouds)
    .map(scaleImage)
    .map(addIndices)
```

The above 39 lines of code can be simplified as 9 lines of code using eemont, which supports method chaining:

```
import ee, eemont

ee.Authenticate()
ee.Initialize()

L8 = (ee.ImageCollection('LANDSAT/LC08/C01/T1_SR')
    .maskClouds()
    .scale()
    .index(['NDVI','EVI','GNDVI']))
```

These methods support multiple datasets from the GEE STAC and can be coupled with

additional packages built on top of the GEE Python API such as geemap (Wu, 2020), geetools (Principe, 2021) and restee (Markert, 2021), extending and simplifying the use of GEE.

Google Earth Engine Community: Developer Resources

The eemont package is featured on GEE's official website (GEE Community: Developer Resources) together with a curated list of resources developed by the GEE developer community, and works as one of the modules that simplify workflows, extending the GEE Python API for the community.

Compatibility with QGIS

The eemont python package can be used in QGIS with the GEE Plugin for QGIS by installing the package using the OSGeo4W Shell:

```
py3_env
python -m pip install eemont
```

After installation, eemont can be used in the Python console inside QGIS:

```
import ee, eemont
from ee_plugin import Map

S2 = (ee.ImageCollection('COPERNICUS/S2_SR')
    .maskClouds()
    .scale()
    .index(['NDVI','EVI','GNDVI'])
    .first())

Map.addLayer(S2,{'min':0,'max':1,'bands':'NDVI'},'NDVI',True)
```

Compatibility with R

The eemont python package can be used in R with rgee by using the reticulate package (Ushey et al., 2021). This compatibility increases the number of researchers who can use the eemont functionalities by using a different programming language. The following chunk shows the eemont configuration and usage for R:

```
library(rgee)
library(reticulate)

ee_Initialize()

py_install("eemont",pip = TRUE)

eemont <- import("eemont")

point <- ee$Geometry$Point(c(-74.0592,11.3172))
L8 <- ee$ImageCollection('LANDSAT/LC08/C01/T1_SR')$filterBounds(point)
L8 <- L8$maskClouds()$scale()$index("NDWI")
```

Acknowledgements

The author would like to thank the Google Earth Engine team, Justin Braaten, Qiusheng Wu, César Aybar and Gennadii Donchyts for their big contribution to the GEE community.

References

Aybar, C., Wu, Q., Bautista, L., Yali, R., & Barja, A. (2020). rgee: An R package for interacting with Google Earth Engine. *The Journal of Open Source Software*, *5*(51), 2272. https://doi.org/10.21105/joss.02272

Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., & Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*, *202*, 18–27. https://doi.org/10.1016/j.rse.2017.06.031

Markert, K. (2021). *restee: Python package to call processed EE objects via the REST API to local data*. https://kmarkert.github.io/restee/

Principe, R. (2021). *geetools: Google Earth Engine Tools (scripts)*. https://github.com/gee-community/gee_tools

QGIS Development Team. (2021). *QGIS Geographic Information System*. QGIS Association. https://www.qgis.org

Ushey, K., Allaire, J., & Tang, Y. (2021). *reticulate: Interface to 'Python'*. https://rstudio.github.io/reticulate/

Wu, Q. (2020). geemap: A Python package for interactive mapping with Google Earth Engine. *The Journal of Open Source Software*, *5*(51), 2305. https://doi.org/10.21105/joss.02305