GeoPython 2021

Worldwide Online

April 22-23

eemont

eemont

A Python package that extends Google Earth Engine

David Montero Loaiza, M.Sc.

@dmlmont

@davemlz

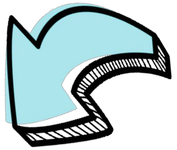
GeoPython 2021
Worldwide Online April 22-23

eemont

eemont

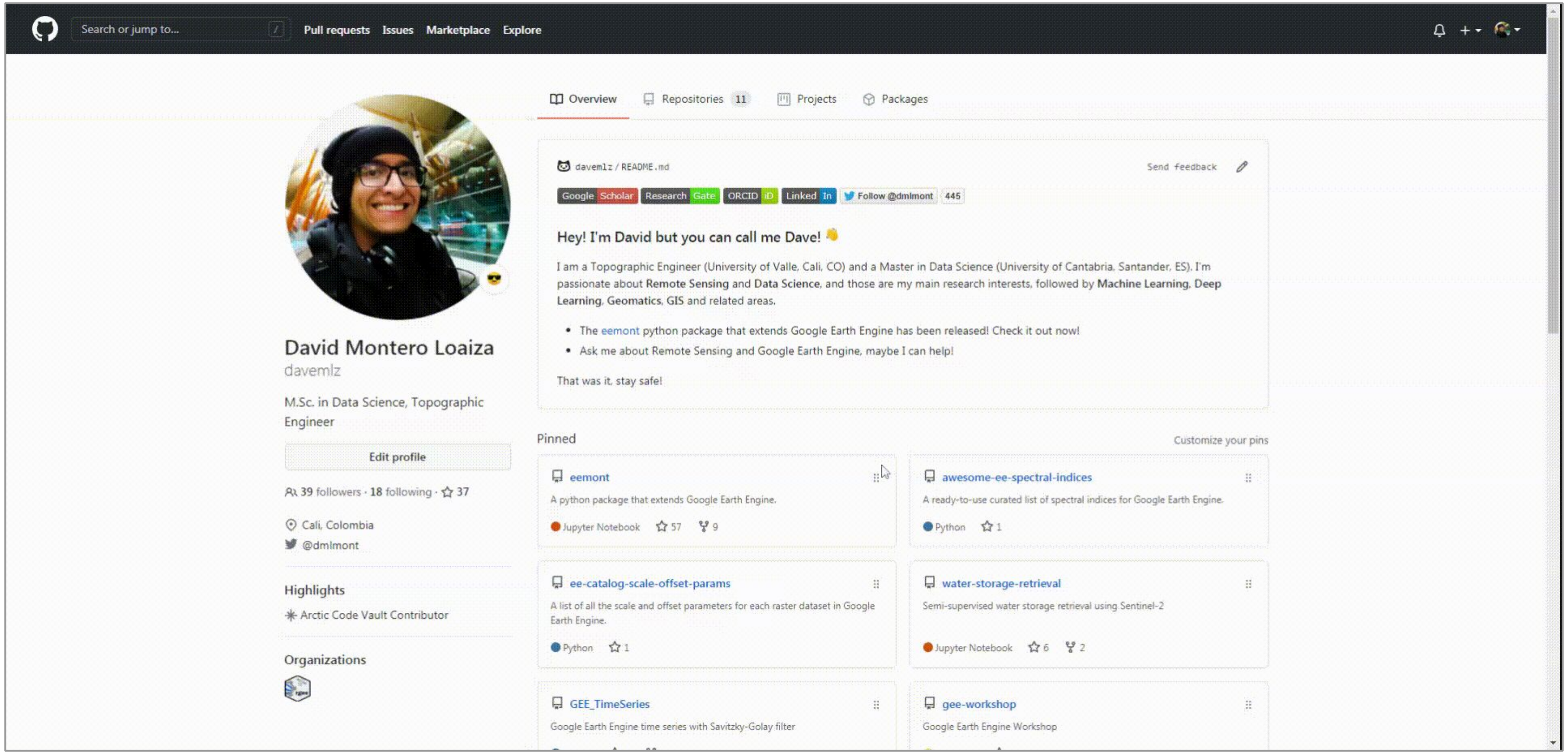
A Python package that extends Google Earth Engine

David Montero Loaiza, M.Sc.



@dmlmont

@davemlz



davemlz / README.md

Google Scholar | Research Gate | ORCID iD | Linked In | Follow @dmlmont 445

Hey! I'm David but you can call me Dave! 👋

I am a Topographic Engineer (University of Valle, Cali, CO) and a Master in Data Science (University of Cantabria, Santander, ES). I'm passionate about **Remote Sensing** and **Data Science**, and those are my main research interests, followed by **Machine Learning**, **Deep Learning**, **Geomatics**, **GIS** and related areas.

- The eemont python package that extends Google Earth Engine has been released! Check it out now!
- Ask me about Remote Sensing and Google Earth Engine, maybe I can help!

That was it, stay safe!

David Montero Loaiza
davemlz
M.Sc. in Data Science, Topographic Engineer
39 followers · 18 following · ☆ 37
Cali, Colombia
@dmlmont

Highlights
Arctic Code Vault Contributor

Organizations

Pinned

eemont
A python package that extends Google Earth Engine.
● Jupyter Notebook ☆ 57 9

awesome-ee-spectral-indices
A ready-to-use curated list of spectral indices for Google Earth Engine.
● Python ☆ 1

ee-catalog-scale-offset-params
A list of all the scale and offset parameters for each raster dataset in Google Earth Engine.
● Python ☆ 1

water-storage-retrieval
Semi-supervised water storage retrieval using Sentinel-2
● Jupyter Notebook ☆ 6 2

GEE_TimeSeries
Google Earth Engine time series with Savitzky-Golay filter

gee-workshop
Google Earth Engine Workshop

Overview
Installation
Documentation
Key Features
Tutorials
Coming soon...

Overview

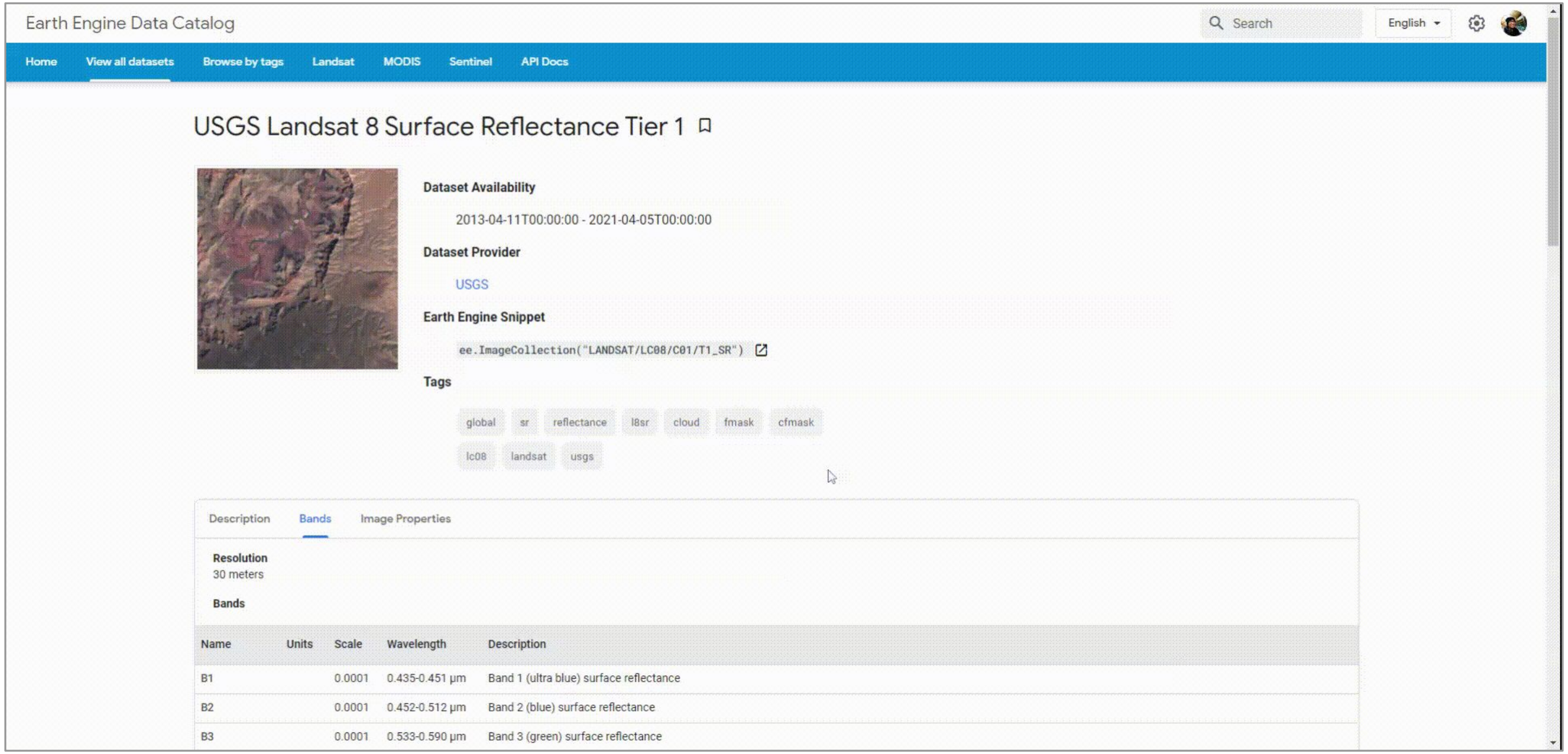
Let’s pre-process (and process) a Landsat 8 SR Collection…

```python
import ee

ee.Authenticate()
ee.Initialize()

L8 = ee.ImageCollection('LANDSAT/LC08/C01/T1_SR')
```

First, we need to mask clouds!

GeoPython 2021 Worldwide Online April 22-23

eemont


Earth Engine Data Catalog

USGS Landsat 8 Surface Reflectance Tier 1

Dataset Availability

2013-04-11T00:00:00 - 2021-04-05T00:00:00

Dataset Provider

USGS

Earth Engine Snippet

```
ee.ImageCollection("LANDSAT/LC08/C01/T1_SR")
```

Tags

global sr reflectance l8sr cloud fmask cfmask lc08 landsat usgs

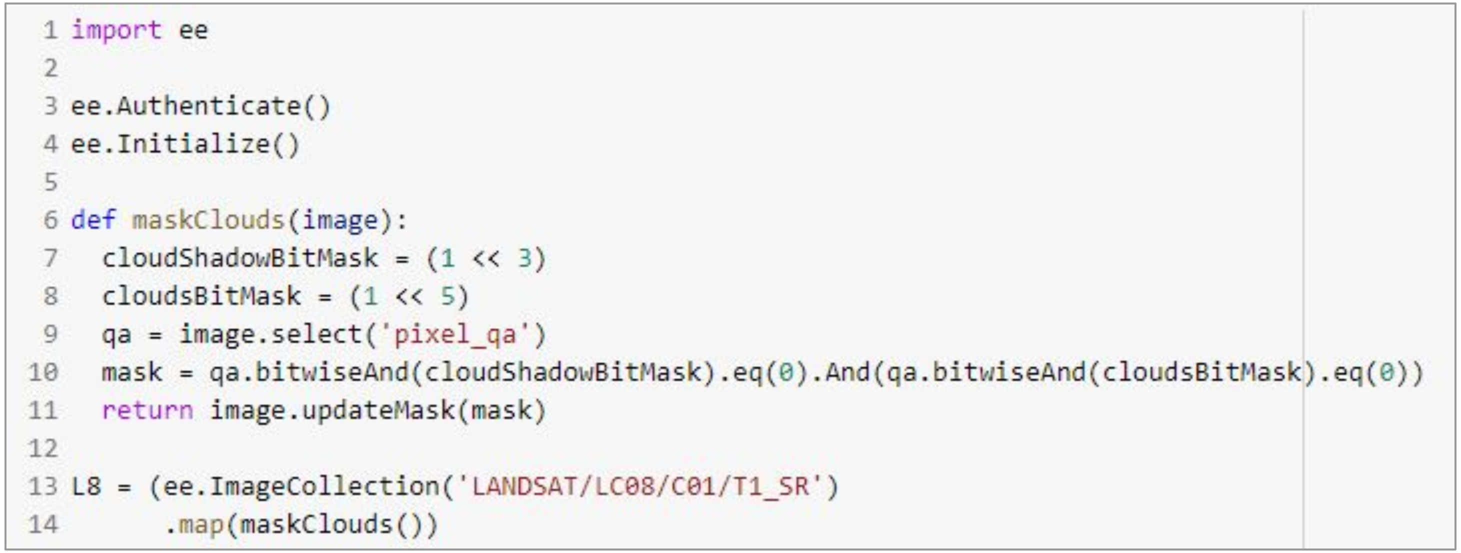
Resolution
30 meters

Bands

Name	Units	Scale	Wavelength	Description
B1		0.0001	0.435-0.451 µm	Band 1 (ultra blue) surface reflectance
B2		0.0001	0.452-0.512 µm	Band 2 (blue) surface reflectance
B3		0.0001	0.533-0.590 µm	Band 3 (green) surface reflectance

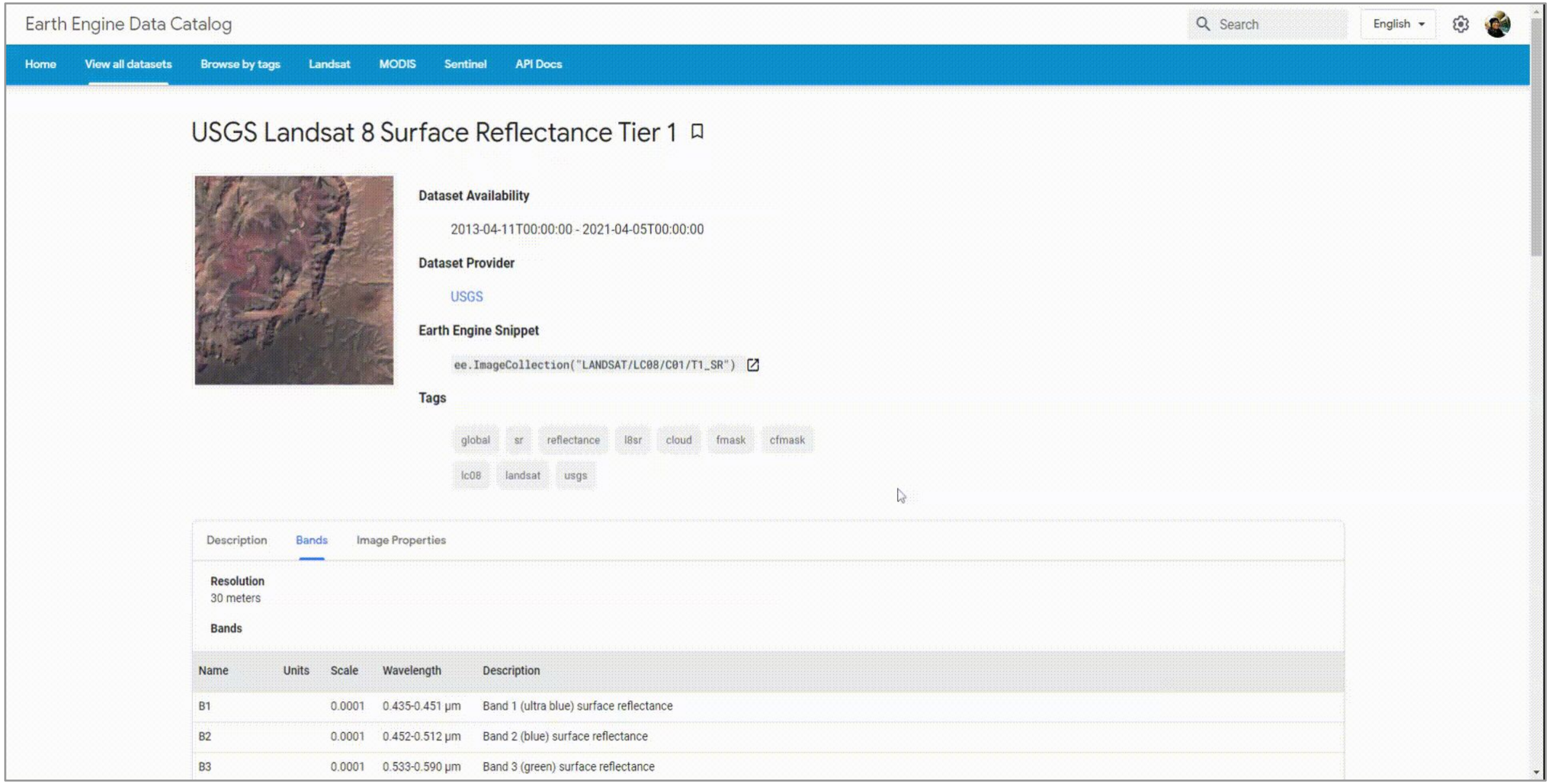
@dmlmont

@davemlz



```python
import ee

ee.Authenticate()
ee.Initialize()

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

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

Second, we need to scale the image!



USGS Landsat 8 Surface Reflectance Tier 1

Dataset Availability

2013-04-11T00:00:00 - 2021-04-05T00:00:00

Dataset Provider

USGS

Earth Engine Snippet

```
ee.ImageCollection("LANDSAT/LC08/C01/T1_SR")
```

Tags

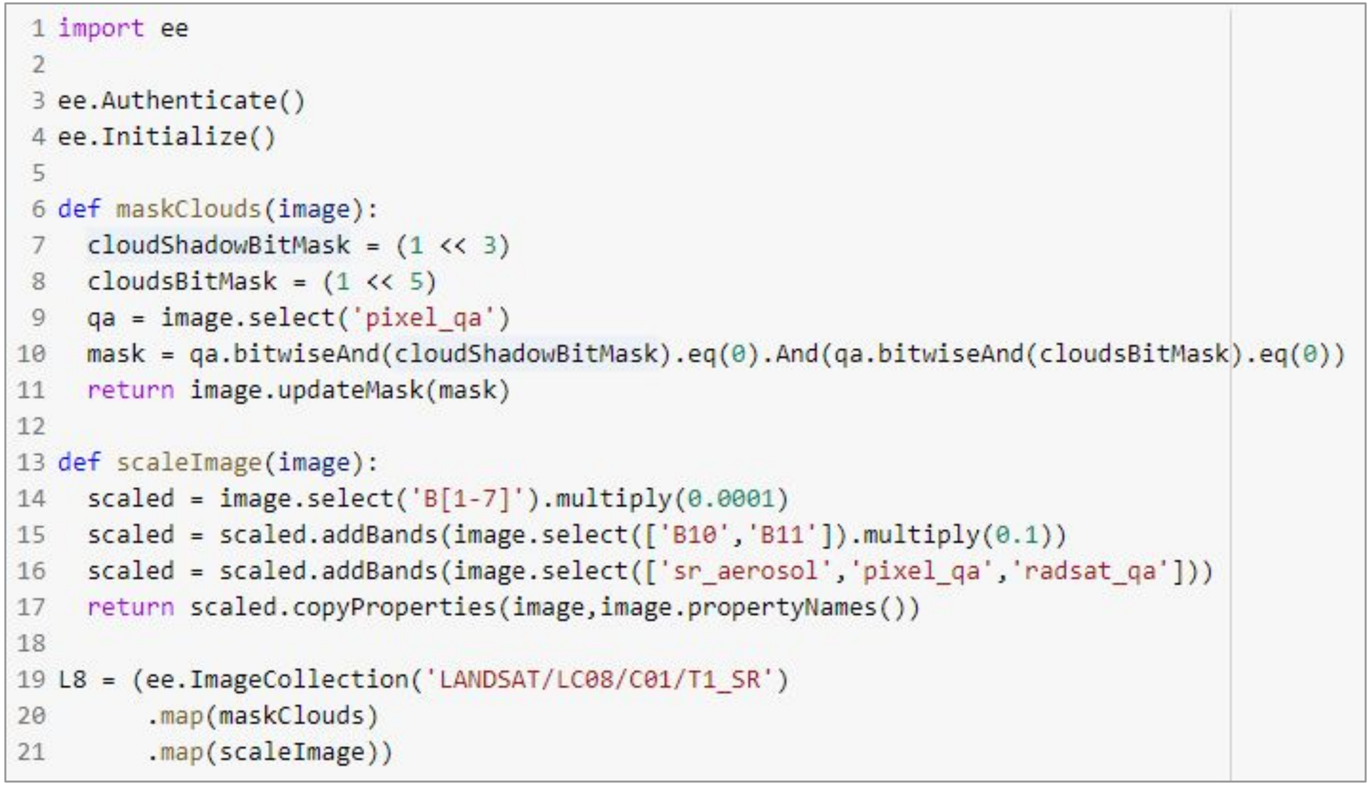
global sr reflectance l8sr cloud fmask cfmask lc08 landsat usgs

Description | Bands | Image Properties

Resolution
30 meters

Bands

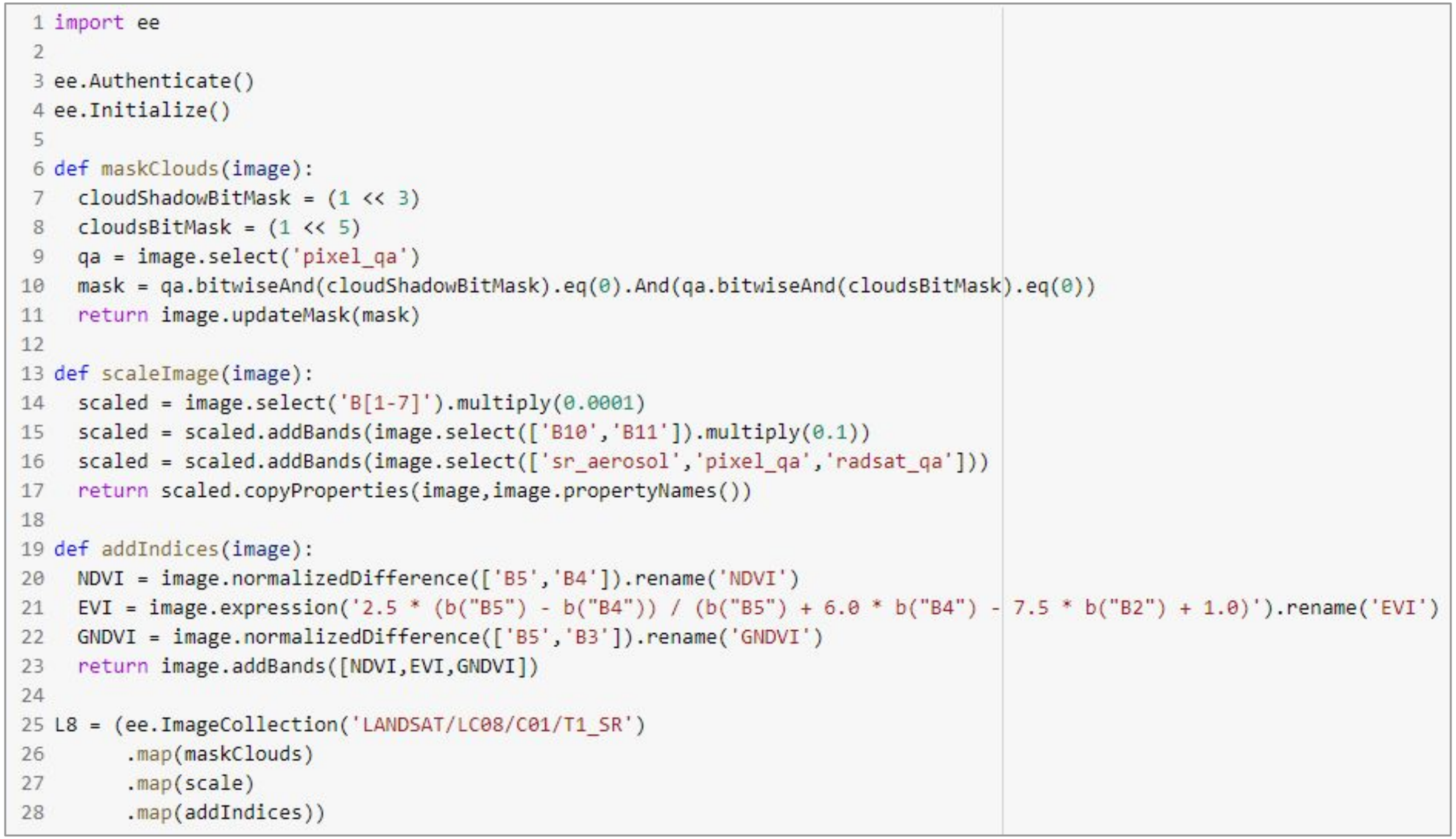
Name	Units	Scale	Wavelength	Description
B1		0.0001	0.435-0.451 µm	Band 1 (ultra blue) surface reflectance
B2		0.0001	0.452-0.512 µm	Band 2 (blue) surface reflectance
B3		0.0001	0.533-0.590 µm	Band 3 (green) surface reflectance



```python
import ee

ee.Authenticate()
ee.Initialize()

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

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

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

Finally, let’s compute some
spectral indices!

e.g. NDVI, EVI and GNDVI…

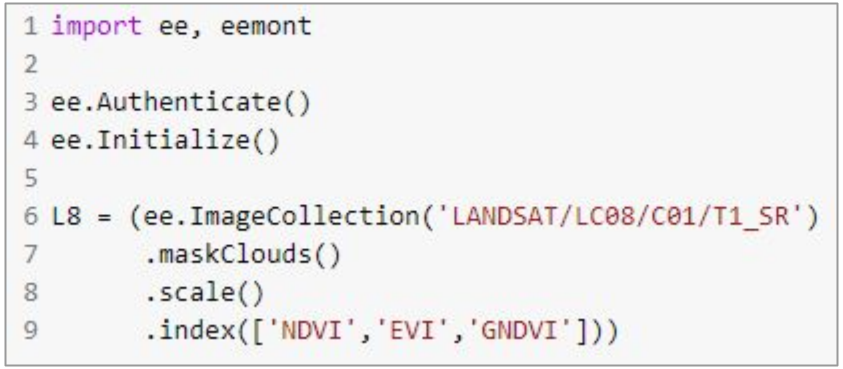


```python
import ee

ee.Authenticate()
ee.Initialize()

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

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

def addIndices(image):
  NDVI = image.normalizedDifference(['B5','B4']).rename('NDVI')
  EVI = image.expression('2.5 * (b("B5") - b("B4")) / (b("B5") + 6.0 * b("B4") - 7.5 * b("B2") + 1.0)').rename('EVI')
  GNDVI = image.normalizedDifference(['B5','B3']).rename('GNDVI')
  return image.addBands([NDVI,EVI,GNDVI])

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

28 lines of code? Well, it's not **THAT** bad!

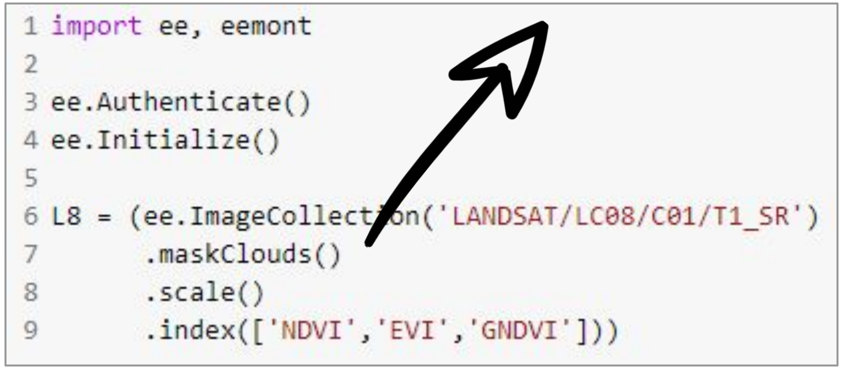
But.. we have to do it over, and over, and over again...

What if we make it... easier?



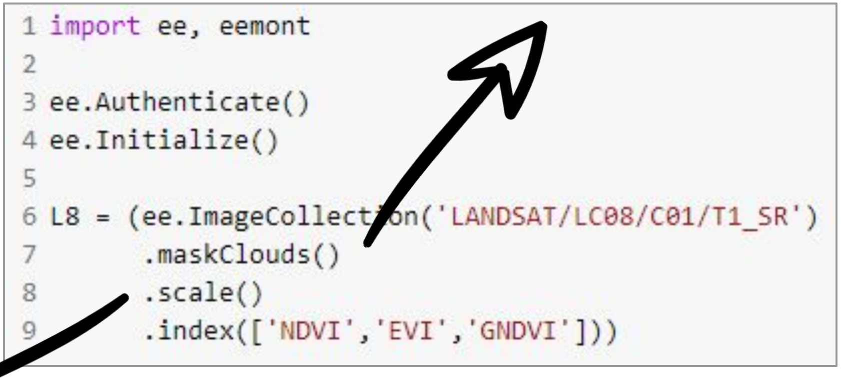
```python
import ee, eemont

ee.Authenticate()
ee.Initialize()

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

Clouds and shadows masking



```
import ee, eemont

ee.Authenticate()
ee.Initialize()

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

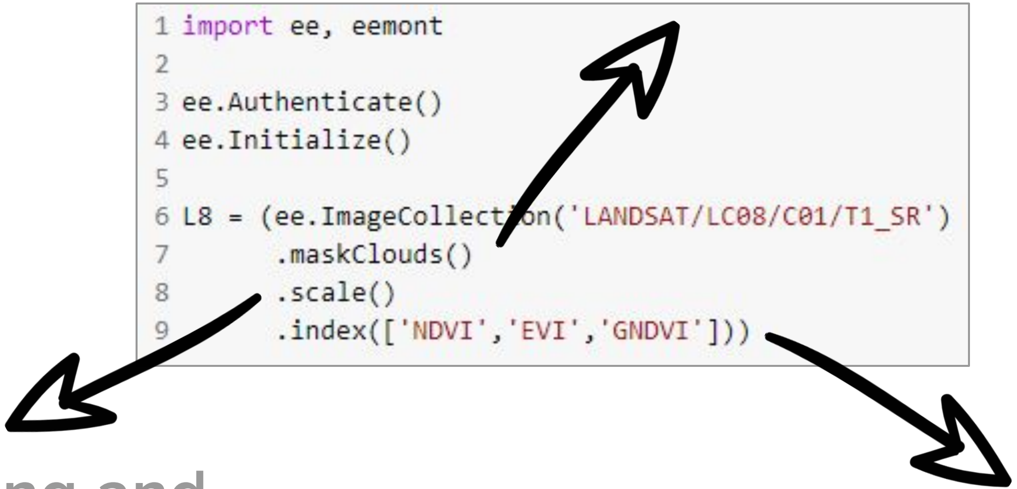
Clouds and shadows masking



```python
import ee, eemont

ee.Authenticate()
ee.Initialize()

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

Image scaling and offsetting

Clouds and shadows masking



```python
import ee, eemont

ee.Authenticate()
ee.Initialize()

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

Image scaling and offsetting

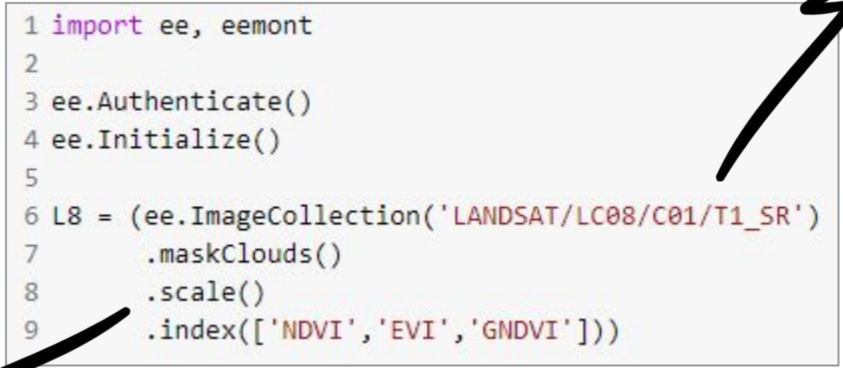
Spectral indices computation

How does it work?

ee.ImageCollection (or ee.Image) object class

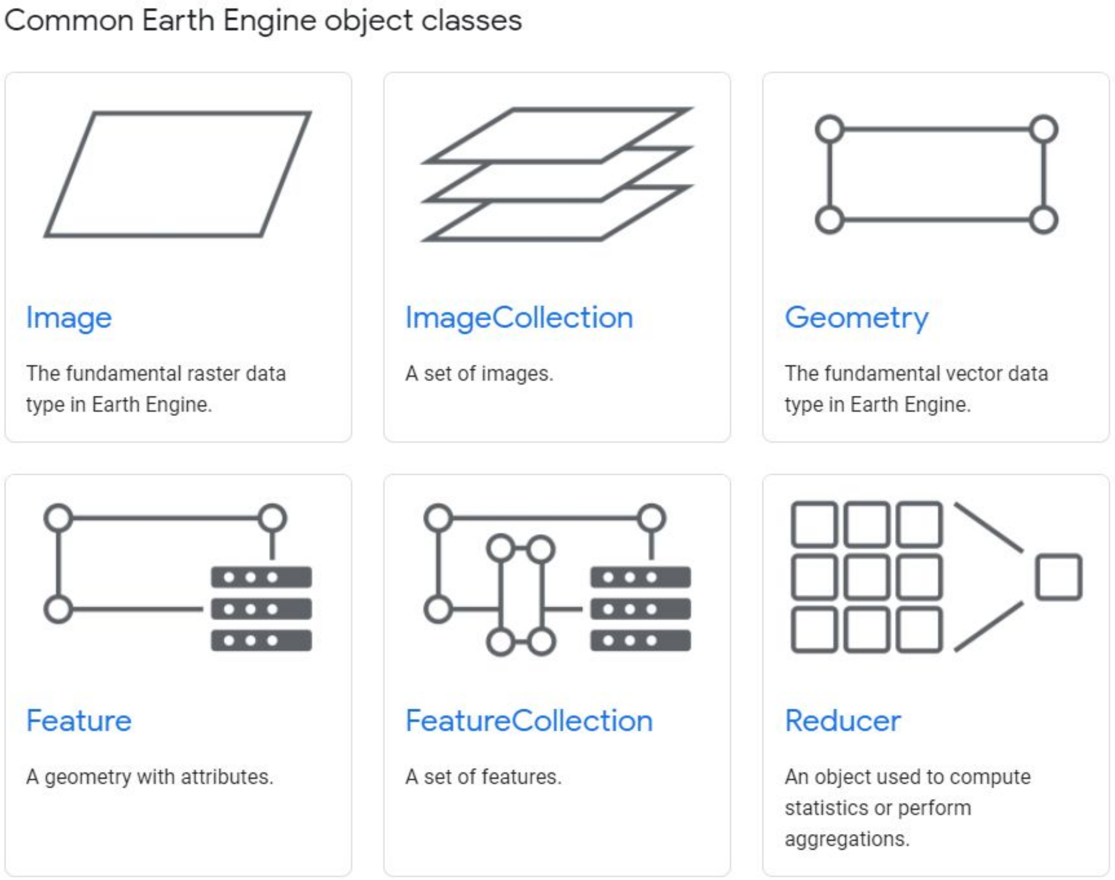
```python
import ee, eemont

ee.Authenticate()
ee.Initialize()

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

ee.ImageCollection (or ee.Image) object class



```python
import ee, eemont

ee.Authenticate()
ee.Initialize()

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

New methods



Common Earth Engine object classes

Image

The fundamental raster data type in Earth Engine.

ImageCollection

A set of images.

Geometry

The fundamental vector data type in Earth Engine.

Feature

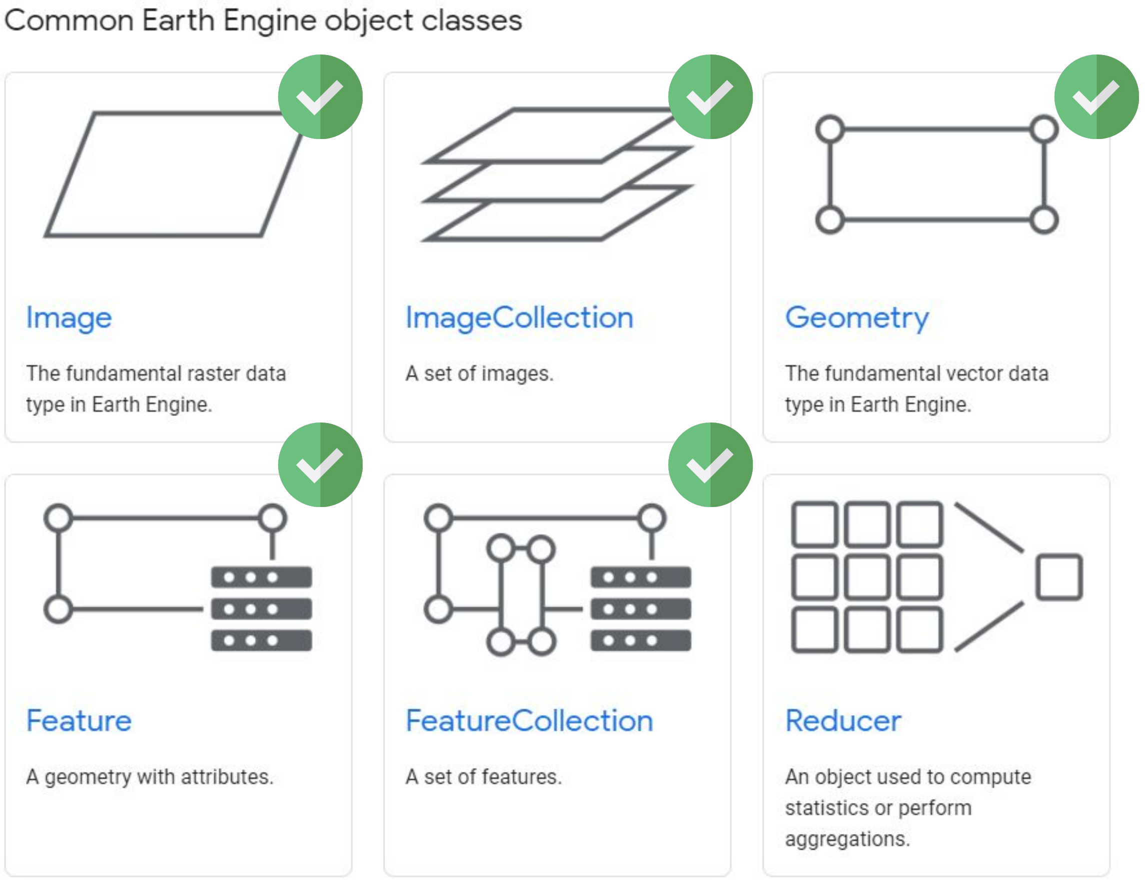
A geometry with attributes.

FeatureCollection

A set of features.

Reducer

An object used to compute statistics or perform aggregations.



Common Earth Engine object classes

Image ✓

The fundamental raster data type in Earth Engine.

ImageCollection ✓

A set of images.

Geometry ✓

The fundamental vector data type in Earth Engine.

Feature ✓

A geometry with attributes.

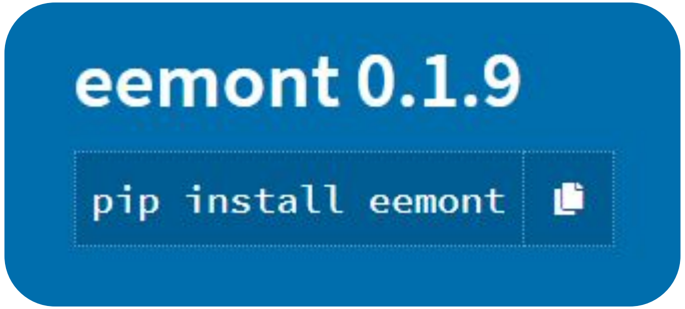
FeatureCollection ✓

A set of features.

Reducer

An object used to compute statistics or perform aggregations.

Installation



eemont 0.1.9

```
pip install eemont
```

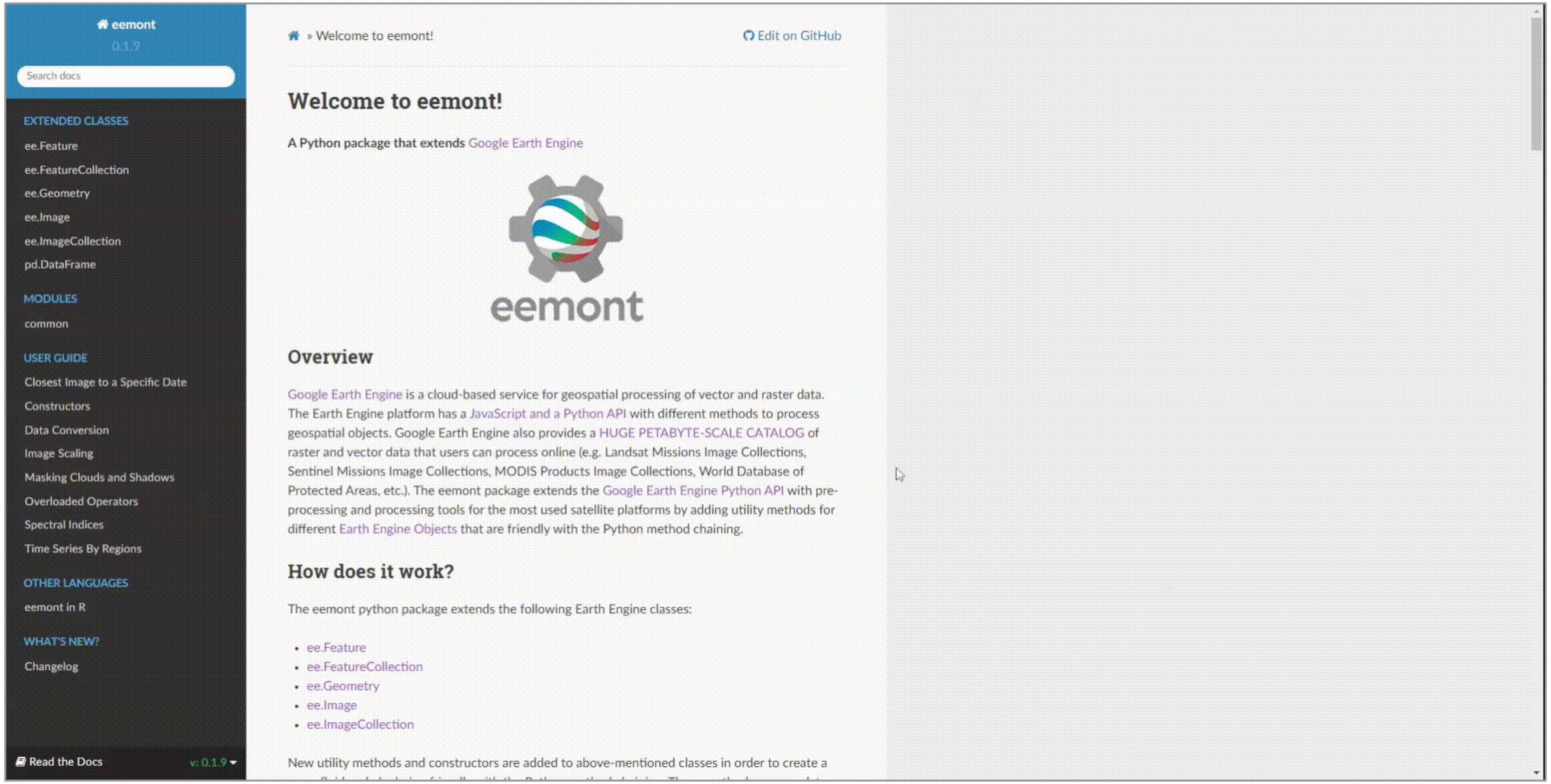
Documentation

Read *the* Docs

https://eemont.readthedocs.io/en/0.1.9/

GeoPython 2021
Worldwide Online
April 22-23

eemont



eemont
0.1.9
Search docs

EXTENDED CLASSES
ee.Feature
ee.FeatureCollection
ee.Geometry
ee.Image
ee.ImageCollection
pd.DataFrame

MODULES
common

USER GUIDE
Closest Image to a Specific Date
Constructors
Data Conversion
Image Scaling
Masking Clouds and Shadows
Overloaded Operators
Spectral Indices
Time Series By Regions

OTHER LANGUAGES
eemont in R

WHAT'S NEW?
Changelog

Read the Docs v: 0.1.9

» Welcome to eemont!
Edit on GitHub

Welcome to eemont!

A Python package that extends Google Earth Engine

eemont

Overview

Google Earth Engine is a cloud-based service for geospatial processing of vector and raster data. The Earth Engine platform has a JavaScript and a Python API with different methods to process geospatial objects. Google Earth Engine also provides a HUGE PETABYTE-SCALE CATALOG of raster and vector data that users can process online (e.g. Landsat Missions Image Collections, Sentinel Missions Image Collections, MODIS Products Image Collections, World Database of Protected Areas, etc.). The eemont package extends the Google Earth Engine Python API with pre-processing and processing tools for the most used satellite platforms by adding utility methods for different Earth Engine Objects that are friendly with the Python method chaining.

How does it work?

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

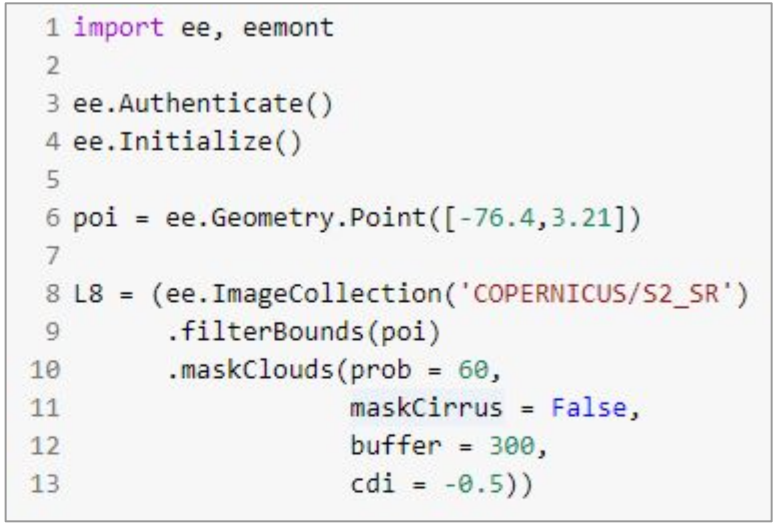
@dmlmont
@davemlz

Key Features

Closest image to a specific date

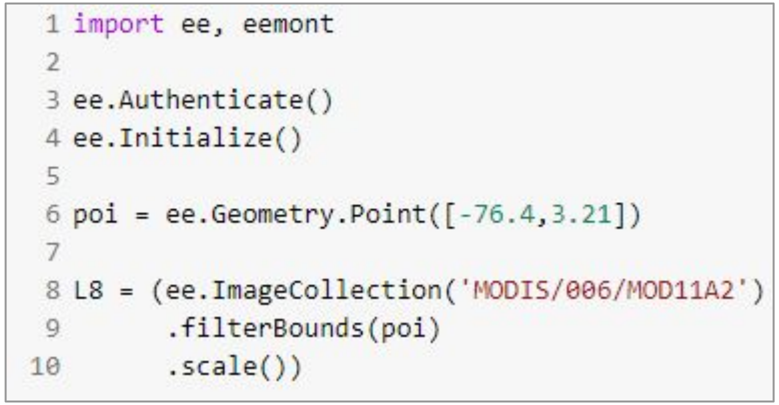
```python
import ee, eemont

ee.Authenticate()
ee.Initialize()

poi = ee.Geometry.Point([-76.4,3.21])

L8 = (ee.ImageCollection('LANDSAT/LC08/C01/T1_SR')
      .filterBounds(poi)
      .closest('2021-10-15'))
```

Masking clouds and shadows



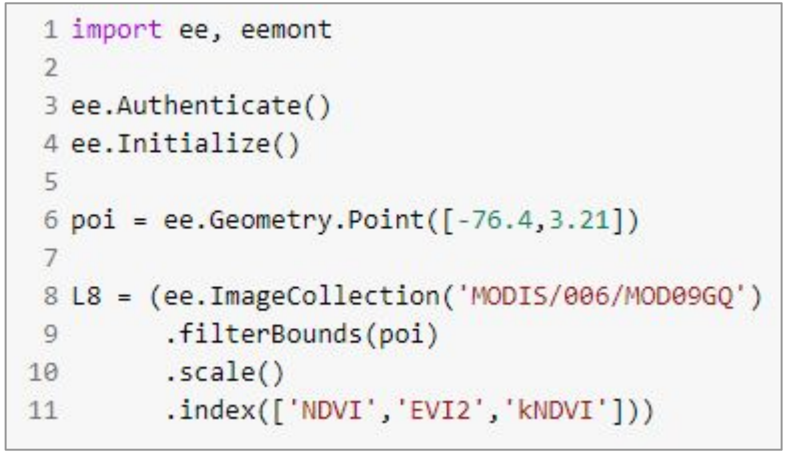
```python
import ee, eemont

ee.Authenticate()
ee.Initialize()

poi = ee.Geometry.Point([-76.4,3.21])

L8 = (ee.ImageCollection('COPERNICUS/S2_SR')
      .filterBounds(poi)
      .maskClouds(prob = 60,
                  maskCirrus = False,
                  buffer = 300,
                  cdi = -0.5))
```

Image scaling (and offsetting)



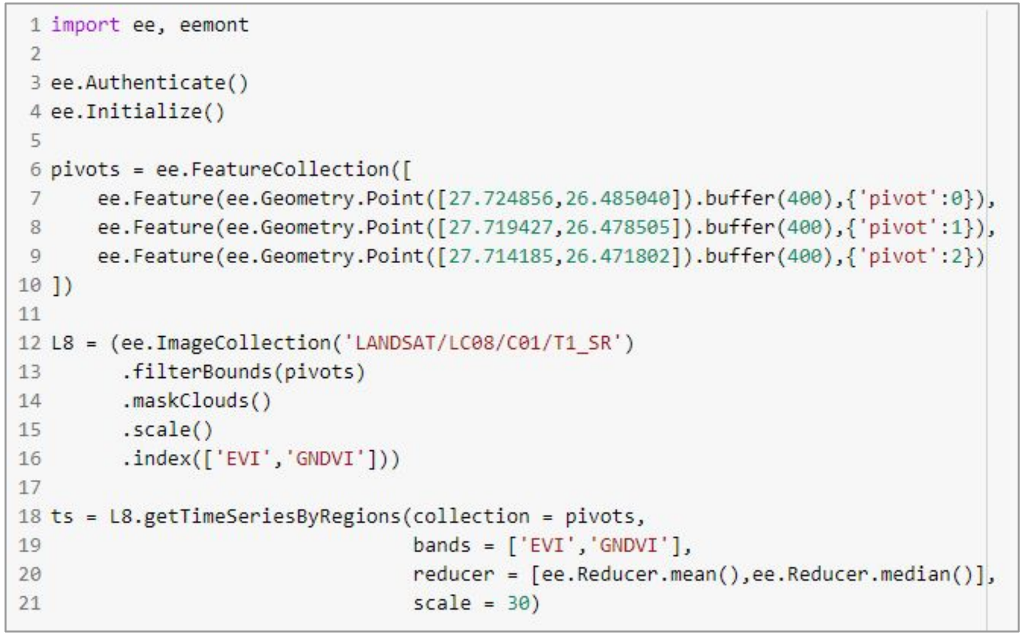
```python
import ee, eemont

ee.Authenticate()
ee.Initialize()

poi = ee.Geometry.Point([-76.4,3.21])

L8 = (ee.ImageCollection('MODIS/006/MOD11A2')
      .filterBounds(poi)
      .scale())
```

Spectral indices computation



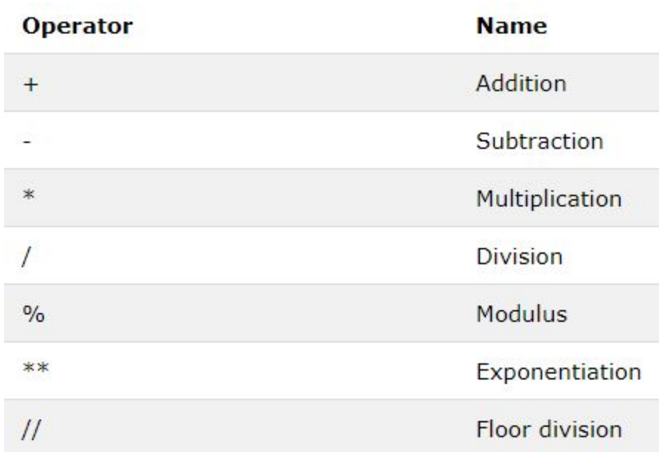
```python
import ee, eemont

ee.Authenticate()
ee.Initialize()

poi = ee.Geometry.Point([-76.4,3.21])

L8 = (ee.ImageCollection('MODIS/006/MOD09GQ')
        .filterBounds(poi)
        .scale()
        .index(['NDVI','EVI2','kNDVI']))
```

Time series by region (or regions)

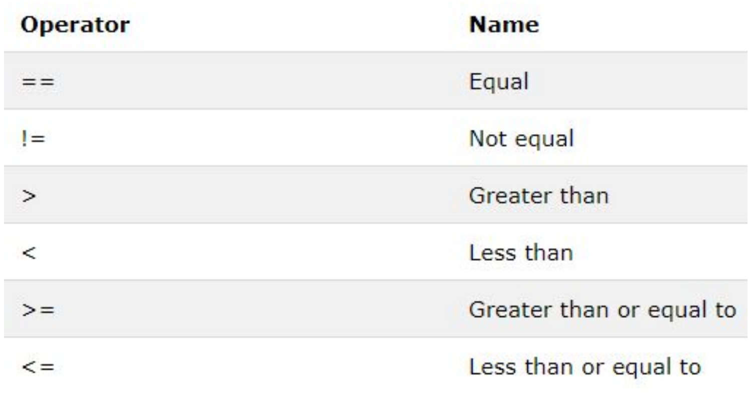


```python
import ee, eemont

ee.Authenticate()
ee.Initialize()

pivots = ee.FeatureCollection([
    ee.Feature(ee.Geometry.Point([27.724856,26.485040]).buffer(400),{'pivot':0}),
    ee.Feature(ee.Geometry.Point([27.719427,26.478505]).buffer(400),{'pivot':1}),
    ee.Feature(ee.Geometry.Point([27.714185,26.471802]).buffer(400),{'pivot':2})
])

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

ts = L8.getTimeSeriesByRegions(collection = pivots,
                               bands = ['EVI','GNDVI'],
                               reducer = [ee.Reducer.mean(),ee.Reducer.median()],
                               scale = 30)
```

Overloaded operators

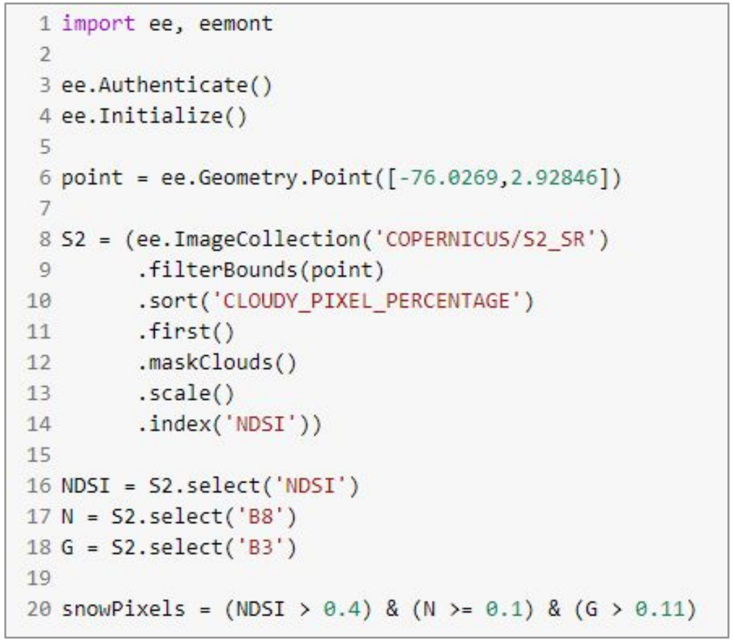


Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
**	Exponentiation
//	Floor division

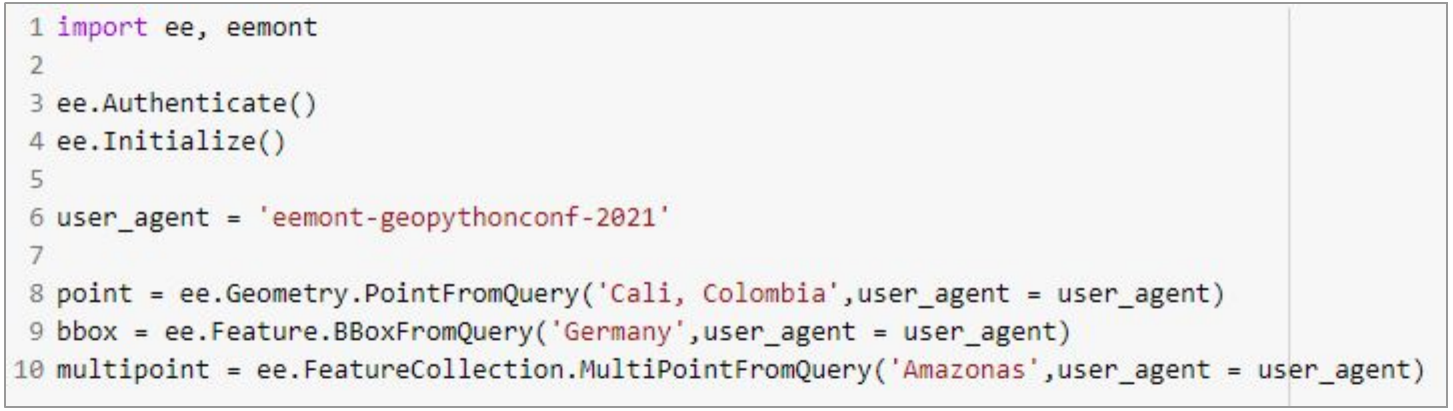


Operator	Name
==	Equal
!=	Not equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Overloaded operators


```python
import ee, eemont

ee.Authenticate()
ee.Initialize()

point = ee.Geometry.Point([-76.0269,2.92846])

S2 = (ee.ImageCollection('COPERNICUS/S2_SR')
      .filterBounds(point)
      .sort('CLOUDY_PIXEL_PERCENTAGE')
      .first()
      .maskClouds()
      .scale()
      .index('NDSI'))

NDSI = S2.select('NDSI')
N = S2.select('B8')
G = S2.select('B3')

snowPixels = (NDSI > 0.4) & (N >= 0.1) & (G > 0.11)
```

Constructors from queries



```python
import ee, eemont

ee.Authenticate()
ee.Initialize()

user_agent = 'eemont-geopythonconf-2021'

point = ee.Geometry.PointFromQuery('Cali, Colombia',user_agent = user_agent)
bbox = ee.Feature.BBoxFromQuery('Germany',user_agent = user_agent)
multipoint = ee.FeatureCollection.MultiPointFromQuery('Amazonas',user_agent = user_agent)
```

eemont GeoPy

BONUS

Compatibility with R

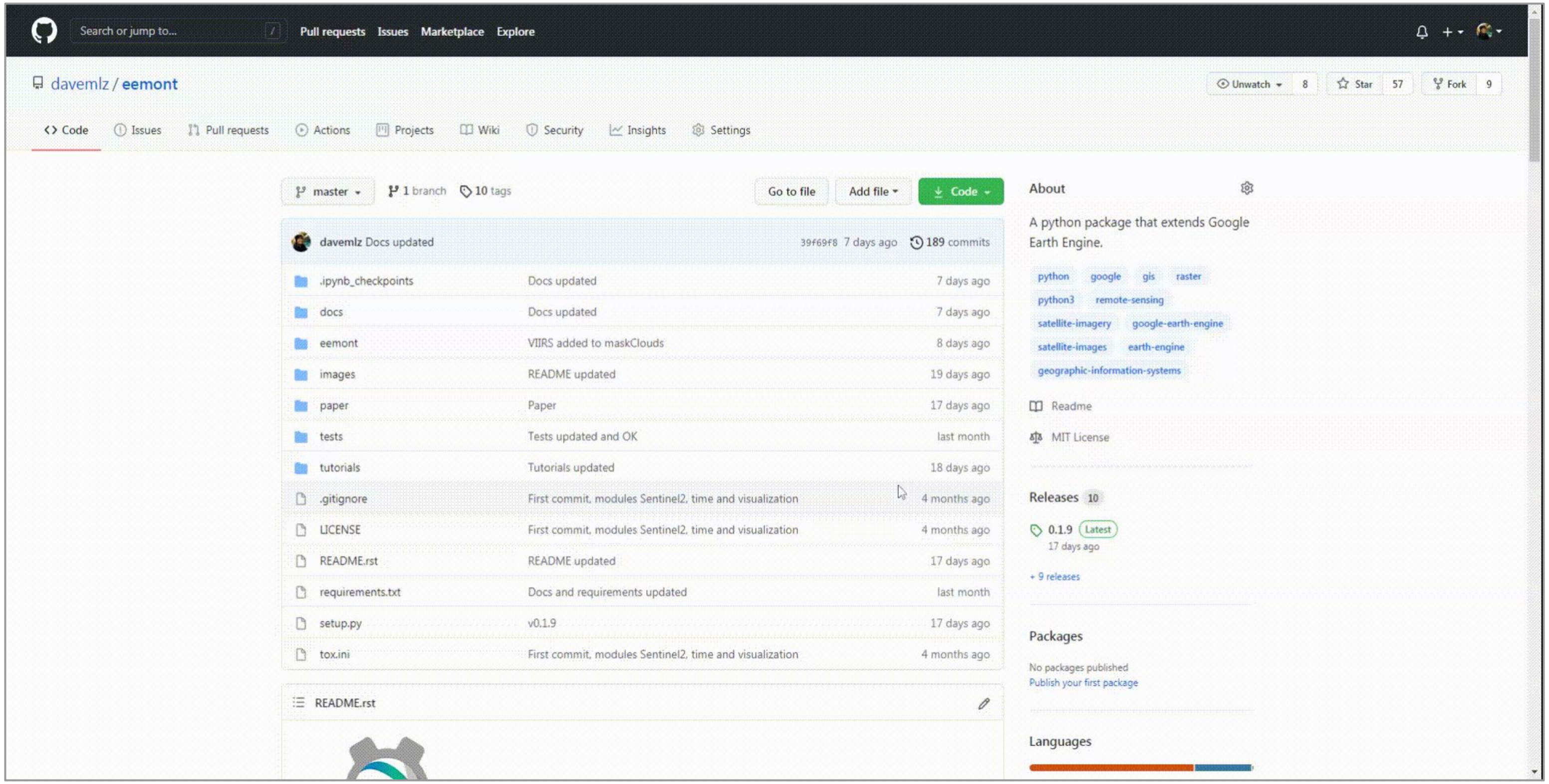
```r
library(rgee)
library(reticulate)

ee_Initialize()

py_install("eemont")

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")
```

eemont

rgee

Tutorials

[001] Masking Clouds and Shadows in Sentinel-2
[002] Scaling a Sentinel-2 Image Collection
[003] Getting the Closest Image to a Specific Date
[004] Computing Spectral Indices on Landsat 8
[005] Computing the EVI with Overloaded Operators
[006] Computing NDSI and Snow Cover using Overloaded Operators and Rich Comparisons
[007] Masking Clouds in Sentinel-3
[008] Cloudless MOD09Q1 Median Composite
[009] Using eemont and geemap in R with rgee
[010] Creating Points From Queries
[011] Creating a Bounding Box From a Query
[012] Computing Spectral Indices on MODIS
[013] Time Series By Region and Conversion to Pandas
[014] Time Series By Regions and Conversion to Pandas



davemlz / eemont

davemlz Docs updated 39f69f8 7 days ago 189 commits

.ipynb_checkpoints	Docs updated	7 days ago
docs	Docs updated	7 days ago
eemont	VIIRS added to maskClouds	8 days ago
images	README updated	19 days ago
paper	Paper	17 days ago
tests	Tests updated and OK	last month
tutorials	Tutorials updated	18 days ago
.gitignore	First commit, modules Sentinel2, time and visualization	4 months ago
LICENSE	First commit, modules Sentinel2, time and visualization	4 months ago
README.rst	README updated	17 days ago
requirements.txt	Docs and requirements updated	last month
setup.py	v0.1.9	17 days ago
tox.ini	First commit, modules Sentinel2, time and visualization	4 months ago

README.rst

About

A python package that extends Google Earth Engine.

python google gis raster python3 remote-sensing satellite-imagery google-earth-engine satellite-images earth-engine geographic-information-systems

Readme

MIT License

Releases 10

0.1.9 Latest
17 days ago

+ 9 releases

Packages

No packages published
Publish your first package

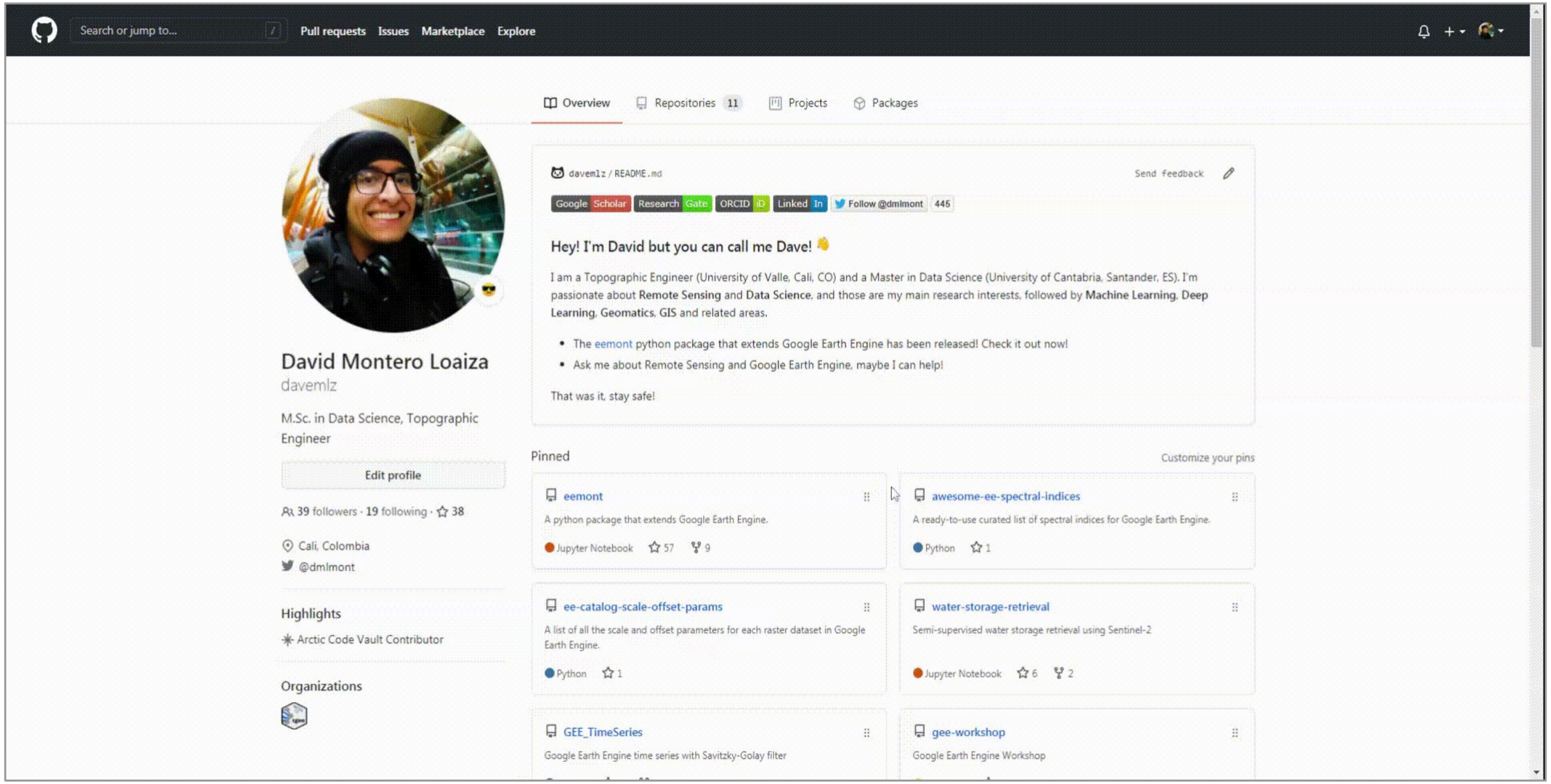
Languages

Coming soon

Spectral indices from the Awesome List of Spectral Indices for Google Earth Engine

awesome

https://github.com/davemlz/awesome-ee-spectral-indices



davemlz / README.md   Send feedback

Google Scholar | Research Gate | ORCID iD | Linked In | Follow @dmlmont 445

Hey! I'm David but you can call me Dave! 👋

I am a Topographic Engineer (University of Valle, Cali, CO) and a Master in Data Science (University of Cantabria, Santander, ES). I'm passionate about **Remote Sensing** and **Data Science**, and those are my main research interests, followed by **Machine Learning**, **Deep Learning**, **Geomatics**, **GIS** and related areas.

- The eemont python package that extends Google Earth Engine has been released! Check it out now!
- Ask me about Remote Sensing and Google Earth Engine, maybe I can help!

That was it, stay safe!

David Montero Loaiza
davemlz

M.Sc. in Data Science, Topographic Engineer

Edit profile

39 followers · 19 following · ☆ 38

Cali, Colombia
@dmlmont

Highlights
Arctic Code Vault Contributor

Organizations

Pinned   Customize your pins

eemont
A python package that extends Google Earth Engine.
● Jupyter Notebook ☆ 57 ⑂ 9

awesome-ee-spectral-indices
A ready-to-use curated list of spectral indices for Google Earth Engine.
● Python ☆ 1

ee-catalog-scale-offset-params
A list of all the scale and offset parameters for each raster dataset in Google Earth Engine.
● Python ☆ 1

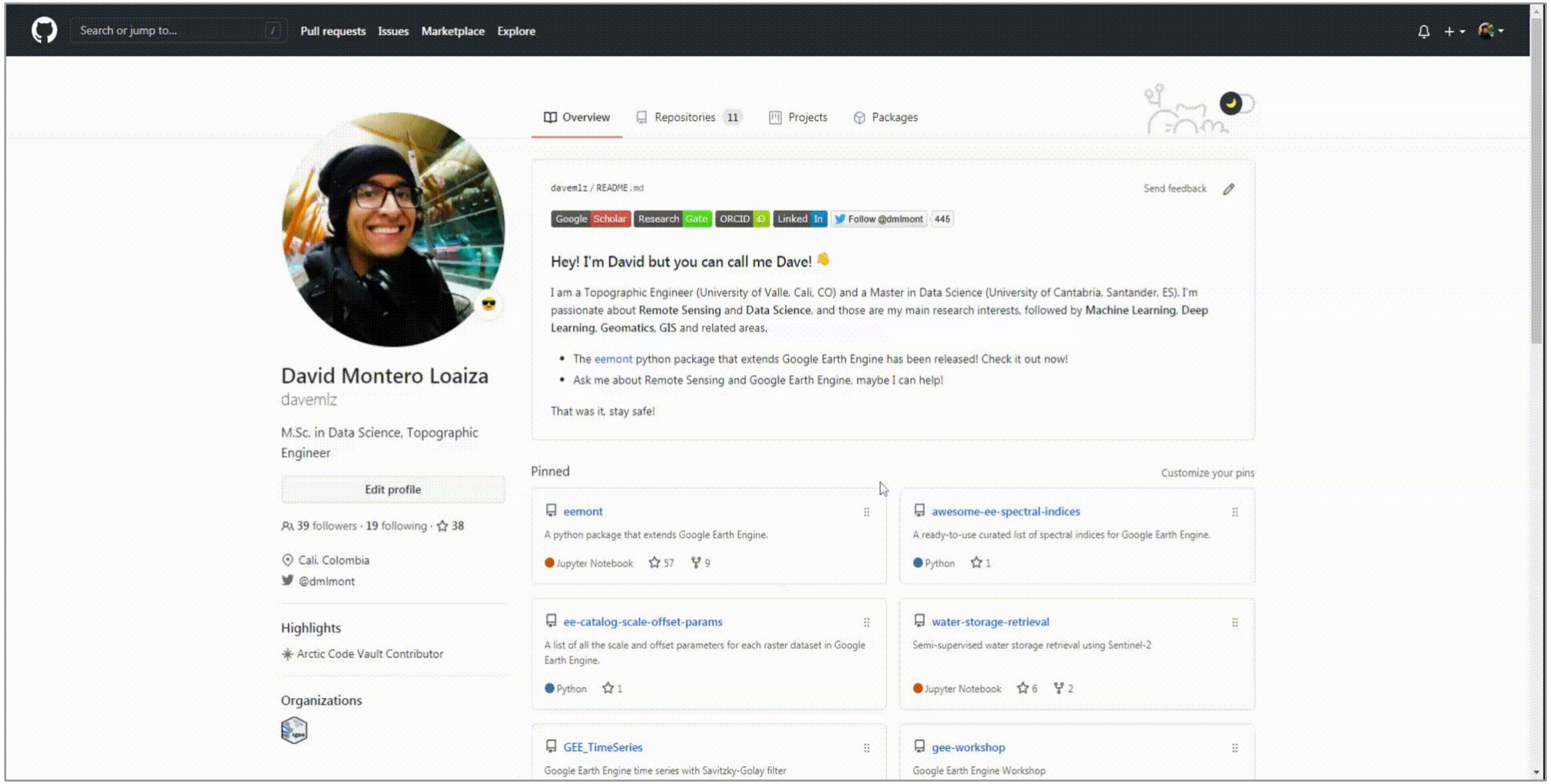
water-storage-retrieval
Semi-supervised water storage retrieval using Sentinel-2
● Jupyter Notebook ☆ 6 ⑂ 2

GEE_TimeSeries
Google Earth Engine time series with Savitzky-Golay filter

gee-workshop
Google Earth Engine Workshop

Image scaling and offsetting for **ALL RASTER DATASETS** in Google Earth Engine using the latest **GEE STAC**

https://github.com/davemlz/ee-catalog-scale-offset-params



davemlz / README.md

Hey! I'm David but you can call me Dave! 👋

I am a Topographic Engineer (University of Valle, Cali, CO) and a Master in Data Science (University of Cantabria, Santander, ES). I'm passionate about **Remote Sensing** and **Data Science**, and those are my main research interests, followed by **Machine Learning**, **Deep Learning**, **Geomatics**, **GIS** and related areas.

- The eemont python package that extends Google Earth Engine has been released! Check it out now!
- Ask me about Remote Sensing and Google Earth Engine, maybe I can help!

That was it, stay safe!

David Montero Loaiza
davemlz

M.Sc. in Data Science, Topographic Engineer

39 followers · 19 following · ☆ 38

Cali, Colombia
@dmlmont

Highlights
Arctic Code Vault Contributor

Organizations

Pinned

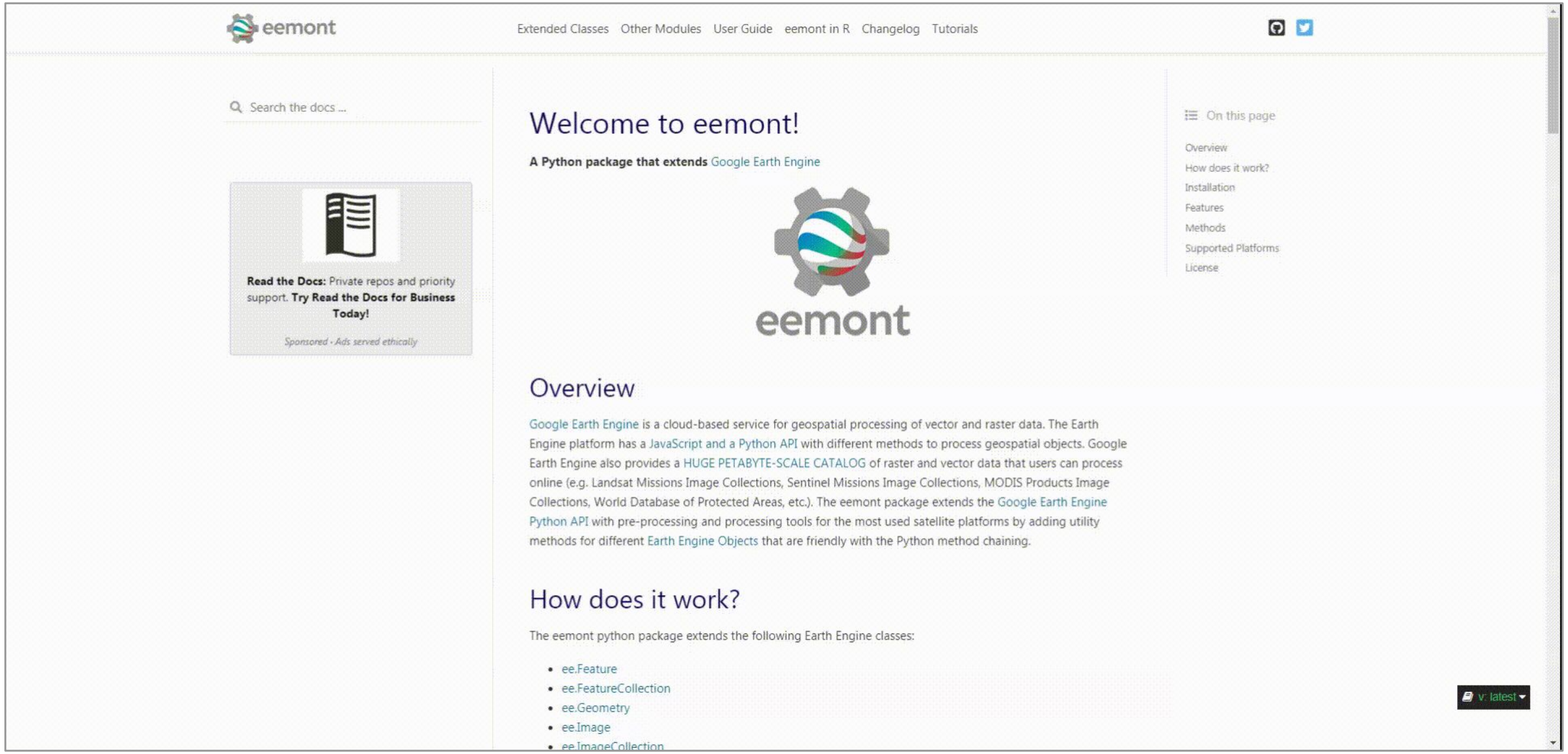
eemont A python package that extends Google Earth Engine. ● Jupyter Notebook ☆ 57 9	**awesome-ee-spectral-indices** A ready-to-use curated list of spectral indices for Google Earth Engine. ● Python ☆ 1
ee-catalog-scale-offset-params A list of all the scale and offset parameters for each raster dataset in Google Earth Engine. ● Python ☆ 1	**water-storage-retrieval** Semi-supervised water storage retrieval using Sentinel-2 ● Jupyter Notebook ☆ 6 2
GEE_TimeSeries Google Earth Engine time series with Savitzky-Golay filter	**gee-workshop** Google Earth Engine Workshop

New methods for the following Earth Engine object classes:

- ee.Number
- ee.Array
- ee.List
- ee.ConfusionMatrix

PyData Theme Documentation

https://eemont.readthedocs.io/en/latest/index.html



eemont Extended Classes Other Modules User Guide eemont in R Changelog Tutorials

Search the docs ...

Read the Docs: Private repos and priority support. **Try Read the Docs for Business Today!**

Sponsored · Ads served ethically

Welcome to eemont!

A Python package that extends Google Earth Engine

eemont

Overview

Google Earth Engine is a cloud-based service for geospatial processing of vector and raster data. The Earth Engine platform has a JavaScript and a Python API with different methods to process geospatial objects. Google Earth Engine also provides a HUGE PETABYTE-SCALE CATALOG of raster and vector data that users can process online (e.g. Landsat Missions Image Collections, Sentinel Missions Image Collections, MODIS Products Image Collections, World Database of Protected Areas, etc.). The eemont package extends the Google Earth Engine Python API with pre-processing and processing tools for the most used satellite platforms by adding utility methods for different Earth Engine Objects that are friendly with the Python method chaining.

How does it work?

The eemont python package extends the following Earth Engine classes:

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

v: latest

JOSS: Under review

JOSS
The Journal of Open Source Software

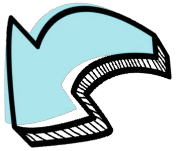
eemont on conda-forge

GeoPython 2021 Worldwide Online April 22-23

eemont

Thank you!

David Montero Loaiza, M.Sc.



@dmlmont

@davemlz