eemont

Release 0.1.7

David Montero Loaiza

Feb 09, 2021

DOCUMENTATION

CHAPTER

ONE

EE.IMAGE

Extended methods for the ee.Image class:

eemont.image.**index**(*self*, *index='NDVI'*, *G=2.5*, *C1=6.0*, *C2=7.5*, *L=1.0*)

Computes one or more spectral indices (indices are added as bands) for an image.

Parameters

- **self** (*ee.Image [this]*) – Image to compute indices on. Must be scaled to [0,1]. Check the supported platforms in User Guide > Spectral Indices > Supported Platforms.
- **index** (*string | list[string], default = 'NDVI'*) – Index or list of indices to compute.

  Available options:

  - 'vegetation' : Compute all vegetation indices.
  - 'burn' : Compute all burn indices.
  - 'water' : Compute all water indices.
  - 'snow' : Compute all snow indices.
  - 'all' : Compute all indices listed below.

  Vegetation indices:

  - 'BNDVI' : Blue Normalized Difference Vegetation Index.
  - 'CIG' : Chlorophyll Index - Green.
  - 'CVI' : Chlorophyll Vegetation Index.
  - 'EVI' : Enhanced Vegetation Index.
  - 'GBNDVI' : Green-Blue Normalized Difference Vegetation Index.
  - 'GNDVI' : Green Normalized Difference Vegetation Index.
  - 'GRNDVI' : Green-Red Normalized Difference Vegetation Index.
  - 'MNDVI' : Modified Normalized Difference Vegetation Index.
  - 'NDVI' : Normalized Difference Vegetation Index.
  - 'NGRDI' : Normalized Green Red Difference Index.
  - 'RVI' : Ratio Vegetation Index.
  - 'SAVI' : Soil-Adjusted Vegetation Index.

  Burn and fire indices:

- 'BAI' : Burned Area Index.
- 'BAIS2' : Burned Area Index for Sentinel 2.
- 'NBR' : Normalized Burn Ratio.

Water indices:

- 'MNDWI' : Modified Normalized Difference Water Index.
- 'NDWI' : Normalized Difference Water Index.

Snow indices:

- 'NDSI' : Normalized Difference Snow Index.

- **G** (*float, default = 2.5*) – Gain factor. Used just for index = 'EVI'.
- **C1** (*float, default = 6.0*) – Coefficient 1 for the aerosol resistance term. Used just for index = 'EVI'.
- **C2** (*float, default = 7.5*) – Coefficient 2 for the aerosol resistance term. Used just for index = 'EVI'.
- **L** (*float, default = 1.0*) – Canopy background adjustment. Used just for index = ['EVI','SAVI'].

Returns Image with the computed spectral index, or indices, as new bands.

Return type ee.Image

eemont.image.**maskClouds**(*self*, *method='cloud_prob'*, *prob=60*, *maskCirrus=True*, *maskShadows=True*, *scaledImage=False*, *dark=0.15*, *cloudDist=1000*, *buffer=250*, *cdi=None*)

Masks clouds and shadows in an image (valid just for Surface Reflectance products).

Parameters

- **self** (*ee.Image [this]*) – Image to mask. Check the supported platforms in User Guide > Masking Clouds and Shadows > Supported Platforms.
- **method** (*string, default = 'cloud_prob'*) – Method used to mask clouds.

  Available options:

  - 'cloud_prob' : Use cloud probability.
  - 'qa' : Use Quality Assessment band.

  This parameter is ignored for Landsat products.
- **prob** (*numeric [0, 100], default = 60*) – Cloud probability threshold. Valid just for method = 'prob'. This parameter is ignored for Landsat products.
- **maskCirrus** (*boolean, default = True*) – Whether to mask cirrus clouds. Valid just for method = 'qa'. This parameter is ignored for Landsat products.
- **maskShadows** (*boolean, default = True*) – Whether to mask cloud shadows. For more info see 'Braaten, J. 2020. Sentinel-2 Cloud Masking with s2cloudless. Google Earth Engine, Community Tutorials'.
- **scaledImage** (*boolean, default = False*) – Whether the pixel values are scaled to the range [0,1] (reflectance values). This parameter is ignored for Landsat products.
- **dark** (*float [0,1], default = 0.15*) – NIR threshold. NIR values below this threshold are potential cloud shadows. This parameter is ignored for Landsat products.

- **cloudDist** (*int, default = 1000*) – Maximum distance in meters (m) to look for cloud shadows from cloud edges. This parameter is ignored for Landsat products.
- **buffer** (*int, default = 250*) – Distance in meters (m) to dilate cloud and cloud shadows objects. This parameter is ignored for Landsat products.
- **cdi** (*float [-1,1], default = None*) – Cloud Displacement Index threshold. Values below this threshold are considered potential clouds. A cdi = None means that the index is not used. For more info see 'Frantz, D., HaS, E., Uhl, A., Stoffels, J., Hill, J. 2018. Improvement of the Fmask algorithm for Sentinel-2 images: Separating clouds from bright surfaces based on parallax effects. Remote Sensing of Environment 2015: 471-481'. This parameter is ignored for Landsat products.

Returns Cloud-shadow masked image.

Return type ee.Image

eemont.image.**scale**(*self*)

Scales bands on an image.

Parameters **self** (*ee.Image [this]*) – Image to scale. Check the supported platforms in User Guide > Image Scaling > Supported Platforms.

Returns Scaled image.

Return type ee.Image

CHAPTER

TWO

EE.IMAGECOLLECTION

Extended methods for the ee.ImageCollection class:

eemont.imagecollection.**closest**(*self*, *date*, *tolerance=1*, *unit='month'*)

Gets the closest image (or set of images if the collection intersects a region that requires multiple scenes) to the specified date.

Parameters

- **self** (*ee.ImageCollection [this]*) – Image Collection from which to get the closest image to the specified date.
- **date** (*ee.Date | string*) – Date of interest. The method will look for images closest to this date.
- **tolerance** (*float, default = 1*) – Filter the collection to [date - tolerance, date + tolerance) before searching the closest image. This speeds up the searching process for collections with a high temporal resolution.
- **unit** (*string, default = 'month'*) – Units for tolerance. Available units: 'year', 'month', 'week', 'day', 'hour', 'minute' or 'second'.

Returns Closest images to the specified date.

Return type ee.ImageCollection

eemont.imagecollection.**index**(*self*, *index='NDVI'*, *G=2.5*, *C1=6.0*, *C2=7.5*, *L=1.0*)

Computes one or more spectral indices (indices are added as bands) for an image collection.

Parameters

- **self** (*ee.ImageCollection*) – Image collection to compute indices on. Must be scaled to [0,1]. Check the supported platforms in User Guide > Spectral Indices > Supported Platforms.
- **index** (*string | list[string], default = 'NDVI'*) – Index or list of indices to compute.

  Available options:

  – 'vegetation' : Compute all vegetation indices.

  – 'burn' : Compute all burn indices.

  – 'water' : Compute all water indices.

  – 'snow' : Compute all snow indices.

  – 'all' : Compute all indices listed below.

  Vegetation indices:

- 'BNDVI' : Blue Normalized Difference Vegetation Index.
- 'CIG' : Chlorophyll Index - Green.
- 'CVI' : Chlorophyll Vegetation Index.
- 'EVI' : Enhanced Vegetation Index.
- 'GBNDVI' : Green-Blue Normalized Difference Vegetation Index.
- 'GNDVI' : Green Normalized Difference Vegetation Index.
- 'GRNDVI' : Green-Red Normalized Difference Vegetation Index.
- 'MNDVI' : Modified Normalized Difference Vegetation Index.
- 'NDVI' : Normalized Difference Vegetation Index.
- 'NGRDI' : Normalized Green Red Difference Index.
- 'RVI' : Ratio Vegetation Index.
- 'SAVI' : Soil-Adjusted Vegetation Index.

Burn and fire indices:

- 'BAI' : Burned Area Index.
- 'BAIS2' : Burned Area Index for Sentinel 2.
- 'NBR' : Normalized Burn Ratio.

Water indices:

- 'MNDWI' : Modified Normalized Difference Water Index.
- 'NDWI' : Normalized Difference Water Index.

Snow indices:

- 'NDSI' : Normalized Difference Snow Index.

- **G** (*float, default = 2.5*) – Gain factor. Used just for index = 'EVI'.
- **C1** (*float, default = 6.0*) – Coefficient 1 for the aerosol resistance term. Used just for index = 'EVI'.
- **C2** (*float, default = 7.5*) – Coefficient 2 for the aerosol resistance term. Used just for index = 'EVI'.
- **L** (*float, default = 1.0*) – Canopy background adjustment. Used just for index = ['EVI','SAVI'].

Returns Image collection with the computed spectral index, or indices, as new bands.

Return type ee.ImageCollection

eemont.imagecollection.**maskClouds**(*self*, *method='cloud_prob'*, *prob=60*, *maskCirrus=True*, *maskShadows=True*, *scaledImage=False*, *dark=0.15*, *cloudDist=1000*, *buffer=250*, *cdi=None*)

Masks clouds and shadows in an image collection (valid just for Surface Reflectance products).

Parameters

- **self** (*ee.ImageCollection [this]*) – Image collection to mask. Check the supported platforms in User Guide > Masking Clouds and Shadows > Supported Platforms.
- **method** (*string, default = 'cloud_prob'*) – Method used to mask clouds.

  Available options:

- 'cloud_prob' : Use cloud probability.
- 'qa' : Use Quality Assessment band.

This parameter is ignored for Landsat products.

- **prob** (*numeric [0, 100], default = 60*) – Cloud probability threshold. Valid just for method = 'prob'. This parameter is ignored for Landsat products.
- **maskCirrus** (*boolean, default = True*) – Whether to mask cirrus clouds. Valid just for method = 'qa'. This parameter is ignored for Landsat products.
- **maskShadows** (*boolean, default = True*) – Whether to mask cloud shadows. For more info see 'Braaten, J. 2020. Sentinel-2 Cloud Masking with s2cloudless. Google Earth Engine, Community Tutorials'.
- **scaledImage** (*boolean, default = False*) – Whether the pixel values are scaled to the range [0,1] (reflectance values). This parameter is ignored for Landsat products.
- **dark** (*float [0,1], default = 0.15*) – NIR threshold. NIR values below this threshold are potential cloud shadows. This parameter is ignored for Landsat products.
- **cloudDist** (*int, default = 1000*) – Maximum distance in meters (m) to look for cloud shadows from cloud edges. This parameter is ignored for Landsat products.
- **buffer** (*int, default = 250*) – Distance in meters (m) to dilate cloud and cloud shadows objects. This parameter is ignored for Landsat products.
- **cdi** (*float [-1,1], default = None*) – Cloud Displacement Index threshold. Values below this threshold are considered potential clouds. A cdi = None means that the index is not used. For more info see 'Frantz, D., HaS, E., Uhl, A., Stoffels, J., Hill, J. 2018. Improvement of the Fmask algorithm for Sentinel-2 images: Separating clouds from bright surfaces based on parallax effects. Remote Sensing of Environment 2015: 471-481'. This parameter is ignored for Landsat products.

Returns Cloud-shadow masked image collection.

Return type ee.ImageCollection

eemont.imagecollection.**scale**(*self*)

Scales bands on an image collection.

Parameters **self** (*ee.ImageCollection (this)*) – Image collection to scale. Check the supported platforms in User Guide > Image Scaling > Supported Platforms.

Returns Scaled image collection.

Return type ee.ImageCollection

CHAPTER

THREE

PD.DATAFRAME

Extended methods for the pd.DataFrame class:

eemont.dataframe.**toEEFeatureCollection**(*self*, *latitude=None*, *longitude=None*)

Converts a pd.DataFrame object into an ee.FeatureCollection object. If lat/lon coordinates are available, the Data Frame can be converted into a Feature Collection with an associated geometry.

Parameters

- **self** (*pd.DataFrame [this]*) – Data Frame to convert into a Feature Collection.
- **latitude** (*string*) – Name of a latitude column, if available. Coupled with a longitude column, an ee.Geometry.Point is created and associated to each Feature.
- **longitude** (*string*) – Name of a longitude column, if available. Coupled with a latitude column, an ee.Geometry.Point is created and associated to each Feature.

Returns Data Frame converted into a Feature Collection.

Return type ee.FeatureCollection

CHAPTER

FOUR

OVERLOADED OPERATORS

Let's see how to use overloaded operators in eemont!

Before anything, let's import our modules and authenticate in Google Earth Engine:

```
import ee, eemont

ee.Authenticate()
ee.Initialize()
```

Now, we are ready to go!

4.1 Overview

The eemont package extends the ee.Image with the binary and unary operators (including rich comparisons).

4.2 List of Operators

4.2.1 Binary Operators

The following table shows the list of binary operators that are overloaded:

Table 1: Binary operators.

Operation	GEE Python method	Overloaded Operator
Addition	Image1.add(Image2)	Image1 + Image2
Subtraction	Image1.subtract(Image2)	Image1 - Image2
Multiplication	Image1.multiply(Image2)	Image1 * Image2
Division	Image1.divide(Image2)	Image1 / Image2
Floor Division	Image1.divide(Image2).floor()	Image1 // Image2
Modulo	Image1.mod(Image2)	Image1 % Image2
Power	Image1.pow(Image2)	Image1 ** Image2
Left Shift	Image1.leftShift(Image2)	Image1 << Image2
Right Shift	Image1.rightShift(Image2)	Image1 >> Image2
And	Image1.And(Image2)	Image1 & Image2
Or	Image1.Or(Image2)	Image1 | Image2

4.2.2 Rich Comparisons

The following table shows the list of rich comparisons that are overloaded:

Table 2: Rich comparisons.

Operation	GEE Python method	Overloaded Operator
Lower Than	Image1.lt(Image2)	Image1 < Image2
Lower Than or Equal	Image1.lte(Image2)	Image1 <= Image2
Equal	Image1.eq(Image2)	Image1 == Image2
Not Equal	Image1.neq(Image2)	Image1 != Image2
Greater Than	Image1.gt(Image2)	Image1 > Image2
Greater Than or Equal	Image1.gte(Image2)	Image1 >= Image2
Equal	Image1.eq(Image2)	Image1 == Image2

4.2.3 Unary Operators

The following table shows the list of unary operators that are overloaded:

Table 3: Unary operators.

Operation	GEE Python method	Overloaded Operator
Negation	Image.multiply(-1)	-Image
Invert	Image.Not()	~ Image

4.3 Usage

Overloaded operators can be used on any ee.Image object. Let's see how to compute the EVI using overloaded operators!

Let's take the Sentinel-2 SR image collection as example (remember to scale your image or image collection!):

```
point = ee.Geometry.Point([-76.0269,2.92846])
S2 = (ee.ImageCollection('COPERNICUS/S2_SR')
   .filterBounds(point)
   .sort('CLOUDY_PIXEL_PERCENTAGE')
   .first()
   .maskClouds()
   .scale())
```

Now, let's take apart the bands that we need (it is not necessary, but it's easier to use `N` instead of `S2.select('B8')`):

```
N = S2.select('B8')
R = S2.select('B4')
B = S2.select('B2')
```

And finally, let's compute the EVI using overloaded operators:

```
EVI = 2.5 * (N - R) / (N + 6.0 * R - 7.5 * B + 1.0)
```

Let's see another example, but using rich comparisons. We are going to compute a snow cover mask!

First, compute the NDSI:

```
S2 = S2.index('NDSI')
```

And now, let's take apart the bands that we need:

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

Finally, compute the snow cover mask (Hall et al., 2001):

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

And update the mask (if required):

```
S2 = S2.updateMask(snowPixels)
```

CHAPTER

FIVE

CLOSEST IMAGE TO A SPECIFIC DATE

Let's see how to get the closest image (or set of images) to a specific date.

Before anything, let's import our modules and authenticate in Google Earth Engine:

```
import ee, eemont

ee.Authenticate()
ee.Initialize()
```

Now, we are ready to go!

5.1 Overview

The eemont package extends the ee.ImageCollection class with the method `closest()`:

closest(self, date[, tolerance, unit])	Gets the closest image (or set of images if the collection intersects a region that requires multiple scenes) to the specified date.

This method automatically filters any image collection to get the closest image to a specific date.

Warning: This method uses the `system:time_start` property, therefore, make sure your image collection has it!

5.2 Usage

The `closest()` method works on any image colection that has a `system:time_start` property.

First, let's take the Sentinel-2 image collection as example:

```
S2 = ee.ImageCollection('COPERNICUS/S2_SR')
```

Now, we have to filter the collection to our ROI. The result of the `closest()` method will vary depending on this. Let's assume a single point is our ROI.

```
ROI = ee.Geometry.Point([-76.45, 4.32])
S2 = S2.filterBounds(ROI)
```

Now, the `closest()` method has just one parameter, `date`, and this parameter can be a string...

```
S2.closest('2020-10-15')
```

Or an ee.Date class:

```
dateOfInterest = ee.Date('2020-10-15')
S2.closest(dateOfInterest)
```

Both chunks will give you the same result here: an ee.ImageCollection of size 1. The result has just one image since our ROI intersects just one scene. To get that image as a single image, we can use the `first()` method.

```
S2.closest('2020-10-15').first()
```

By default, the image collection is filtered according to +/- 1 month from the `date` parameter (`tolerance = 1` and `unit = 'month'`). This is done to speed up the searching process, but if required (if there are not images in that range), the `tolerance` and `unit` parameters can be modified:

```
S2.closest('2020-10-15', tolerance = 2, unit = 'year')
```

Now, let's assume that our ROI is larger, in this case, a whole department (state) of Colombia:

```
ROI = (ee.FeatureCollection('FAO/GAUL_SIMPLIFIED_500m/2015/level1')
    .filter(ee.Filter.eq('ADM1_NAME','Valle Del Cauca')))
S2 = ee.ImageCollection('COPERNICUS/S2_SR').filterBounds(ROI).closest('2020-10-15')
```

You'll note that the size of the resulting ee.ImageCollection here is greater than 1. This result has more than one image since our ROI now intersects more than one scene. To get those images together as a single image, you can mosaic them or use an ee.Reducer, for example `median()`.

```
S2.median()
```

CHAPTER

SIX

MASKING CLOUDS AND SHADOWS

Masking clouds and shadows may seem hard, but it isn't! Let's take a look on it!

Before anything, let's import our modules and authenticate in Google Earth Engine:

```
import ee, eemont

ee.Authenticate()
ee.Initialize()
```

Now, we are ready to go!

6.1 Overview

The eemont package extends the ee.Image and ee.ImageCollection classes with the method `maskClouds()`:

6.1.1 ee.Image

maskClouds(self[, method, prob, maskCirrus, ...])	Masks clouds and shadows in an image (valid just for Surface Reflectance products).

6.1.2 ee.ImageCollection

maskClouds(self[, method, prob, maskCirrus, ...])	Masks clouds and shadows in an image collection (valid just for Surface Reflectance products).

6.2 Supported Platforms

This method automatically masks clouds and shadows on the following supported satellite platforms:

6.2.1 Sentinel Missions

- Sentinel-3 OLCI EFR: Ocean and Land Color Instrument Earth Observation Full Resolution
- Sentinel-2 MSI: MultiSpectral Instrument, Level-2A

6.2.2 Landsat Missions

- USGS Landsat 8 Surface Reflectance Tier 1 and 2
- USGS Landsat 7 Surface Reflectance Tier 1 and 2
- USGS Landsat 5 Surface Reflectance Tier 1 and 2
- USGS Landsat 4 Surface Reflectance Tier 1 and 2

6.2.3 MODIS Products

- MOD09GA.006 Terra Surface Reflectance Daily Global 1km and 500m
- MOD09Q1.006 Terra Surface Reflectance 8-Day Global 250m
- MOD09A1.006 Terra Surface Reflectance 8-Day Global 500m
- MCD15A3H.006 MODIS Leaf Area Index/FPAR 4-Day Global 500m
- MOD17A2H.006: Terra Gross Primary Productivity 8-Day Global 500M 500m
- MOD16A2.006: Terra Net Evapotranspiration 8-Day Global 500m
- MOD13Q1.006 Terra Vegetation Indices 16-Day Global 250m
- MOD13A1.006 Terra Vegetation Indices 16-Day Global 500m
- MOD13A2.006 Terra Vegetation Indices 16-Day Global 1km

Warning: Not supported satellite platforms will raise an *Exception*.

6.3 QA Method

By default, the `maskClouds()` uses the QA band of each paltform to compute the clouds and shadows masks (except for Sentinel-2, where the default method is Cloud Probability). The following table shows the band and the bits used for each platform (The value in parentheses is the valid value of the bitmask):

Table 3: QA bits used for clouds/shadows masking

Platform	QA Band	Cloud Bitmask	Cirrus Bitmask	Shadow Bitmask
Sentinel-3	quality_flags	27 (0)		
Sentinel-2	QA60	10 (0)	11 (0)	
Landsat Series	pixel_qa	5 (0)		3 (0)
MOD09GA	state_1km	0 (0)	8 (0)	2 (0)
MOD09Q1	State	0 (0)	8 (0)	2 (0)
MOD09A1	StateQA	0 (0)	8 (0)	2 (0)
MCD15A3H	FparExtra_QC	5 (0)	4 (0)	6 (0)
MOD17A2H	Psn_QC	3 (0)		
MOD16A2	ET_QC	3 (0)		
MOD13Q1	SummaryQA	0 (0)		
MOD13A1	SummaryQA	0 (0)		

6.4 Usage

Let's check how to use the `maskClouds()` method for different platforms:

6.4.1 Sentinel-3

On Sentinel 3, clouds are masked according to the bright pixels in the quality_flags band of the Sentinel-3 OLCI EFR: Ocean and Land Color Instrument Earth Observation Full Resolution.

> **Warning:** This method may mask water as well on Sentinel-3 images.

Let's take the Sentinel-3 image collection as example:

```
S3 = ee.ImageCollection('COPERNICUS/S3/OLCI')
```

There is no need to specify any arguments, since they're ignored.

```
S3.maskClouds()
```

This method can also be applied to a single image:

```
S3.first().maskClouds()
```

And can be used for scaled images without specifying it:

```
S3.scale().maskClouds()
```

6.4.2 Sentinel-2

On Sentinel 2, clouds can be masked using two methods: *QA* and *Cloud Probability*. The *QA* method uses the QA60 band in the Surface Reflectance Product to mask clouds, while the *Cloud Probability* method uses the COPERNICUS/S2_CLOUD_PROBABILITY collection to do it.

Shadows are masked based on the clouds mask, where shadows are searched within a range from clouds edges in the shadows direction.

See also:

For more info on masking shadows, please visit 'Braaten, J. 2020. Sentinel-2 Cloud Masking with s2cloudless. Google Earth Engine, Community Tutorials'.

First, let's take the Sentinel-2 image collection:

```
S2 = ee.ImageCollection('COPERNICUS/S2_SR')
```

In order to use the *QA* method, it must be specified using the `method` parameter:

```
S2.maskClouds(method = 'qa')
```

This line maps the *QA* masking method over the whole collection, but the method can also be applied to a single image:

```
S2.first().maskClouds(method = 'qa')
```

The *QA* method gives us the option to avoid masking cirrus clouds, but it must be specified using the `maskCirrus` parameter:

```
S2.maskClouds(method = 'qa', maskCirrus = False)
```

And we can also avoid masking shadows by specifying the `maskShadows` parameter:

```
S2.maskClouds(method = 'qa', maskShadows = False)
```

Now, in order to use the *Cloud Probability* method, we can specify it in the `method` parameter:

```
S2.maskClouds(method = 'cloud_prob')
```

But, it is the default method, so you can just let the extended method with no additional parameters:

```
S2.maskClouds()
```

The *Cloud Probability* method uses a probability threshold to mask clouds, by default, the threshold is set to 60, but it can be modified using the `prob` parameter:

```
S2.maskClouds(prob = 70)
```

If your image or collection is scaled, the `scaledImage` parameter must be set to `True`:

```
S2.scale().maskClouds(scaledImage = True)
```

In order to search for shadows, portental shadow pixels must be specified. Pixels with a NIR reflectance below 0.15 are considered potential shadow pixels, but this can be modified using the `dark` parameter:

```
S2.maskClouds(dark = 0.2)
```

Shadows are searched whitin a maximum range of 1000 m in the shadow direction from cloud edges, but this range can be modified using the `cloudDist` parameter:

```
S2.maskClouds(cloudDist = 1500)
```

After finding all clouds and shadows, the mask can be dilated to avoid border effects. By default, clouds and shadows are dilated by 250 m, but this can be modified using the `buffer` parameter:

```
S2.maskClouds(buffer = 100)
```

Finally, in order to avoid confusion between clouds and bright surface objects, the Cloud Displacement Index (CDI) can be used. By default, the CDI is not used, but it can be modified using the `cdi` parameter:

```
S2.maskClouds(cdi = -0.5)
```

See also:

For more info on CDI, please visit 'Frantz, D., HaS, E., Uhl, A., Stoffels, J., Hill, J. 2018. Improvement of the Fmask algorithm for Sentinel-2 images: Separating clouds from bright surfaces based on parallax effects. Remote Sensing of Environment 2015: 471-481'.

6.4.3 Landsat Series

On Landsat Series, both clouds and shadows are masked based on the pixel_qa band in the Surface Reflectance Products.

Let's take the Landsat 8 image collection as example:

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

There is no need to specify most of the arguments showed for Sentinel-2, since they're ignored.

```
L8.maskClouds()
```

Shadows are masked by default, but if required, the `maskShadows` parameter can be modified.

```
L8.maskClouds(maskShadows = False)
```

This method can also be applied to a single image:

```
L8.first().maskClouds()
```

And can be used for scaled images without specifying it:

```
L8.scale().maskClouds()
```

6.4.4 MODIS Products

On MODIS Products, clouds and shadows are masked according to the specific QA band.

Let's take the MOD13Q1 image collection as example:

```
MOD13Q1 = ee.ImageCollection('MODIS/006/MOD13Q1')
```

There is no need to specify most of the arguments showed for Sentinel-2, since they're ignored.

```
MOD13Q1.maskClouds()
```

MOD13Q1, MOD13A1, MOD17A2H and MOD16A2 products don't have cirrus and shadow bitmasks, therefore, the arguments `maskShadows` and `maskCirrus` are ignored. MOD09GA, MOD09Q1, MOD09A1 and MCD15A3H products have cirrus and shadows bitmasks, and by default, they are set to *True*. If required, they can be set to *False*:

```
MOD09GA = ee.ImageCollection('MODIS/006/MOD09GA').maskClouds(maskShadows = False,␣
↪maskCirrus = False)
```

This method can also be applied to a single image:

```
MOD09GA.first().maskClouds()
```

And can be used for scaled images without specifying it:

```
MOD09GA.scale().maskClouds()
```

MOD13A2 doesn't have a bitmask QA band, instead, it has a Class QA band, where a value of zero means that the pixel has good data.

```
MOD13A2 = ee.ImageCollection('MODIS/006/MOD13A2').maskClouds()
```

CHAPTER

SEVEN

IMAGE SCALING

Image scaling now is A LOT EASIER with eemont! Let's see how!

Before anything, let's import our modules and authenticate in Google Earth Engine:

```
import ee, eemont

ee.Authenticate()
ee.Initialize()
```

Now, we are ready to go!

7.1 Overview

The eemont package extends the ee.Image and ee.ImageCollection classes with the method `scale()`:

7.1.1 ee.Image

scale(self)	Scales bands on an image.

7.1.2 ee.ImageCollection

scale(self)	Scales bands on an image collection.

7.2 Supported Platforms

This method automatically scales images from the following supported satellite platforms:

7.2.1 Sentinel Missions

- Sentinel-3 OLCI EFR: Ocean and Land Color Instrument Earth Observation Full Resolution
- Sentinel-2 MSI: MultiSpectral Instrument, Level-2A
- Sentinel-2 MSI: MultiSpectral Instrument, Level-1C

7.2.2 Landsat Missions

- USGS Landsat 8 Surface Reflectance Tier 1 and 2
- USGS Landsat 7 Surface Reflectance Tier 1 and 2
- USGS Landsat 5 Surface Reflectance Tier 1 and 2
- USGS Landsat 4 Surface Reflectance Tier 1 and 2

7.2.3 MODIS Products

- MCD43A4.006 MODIS Nadir BRDF-Adjusted Reflectance Daily 500m
- MCD43A3.006 MODIS Albedo Daily 500m
- MOD09GQ.006 Terra Surface Reflectance Daily Global 250m
- MOD09GA.006 Terra Surface Reflectance Daily Global 1km and 500m
- MOD09Q1.006 Terra Surface Reflectance 8-Day Global 250m
- MOD09A1.006 Terra Surface Reflectance 8-Day Global 500m
- MOD10A1.006 Terra Snow Cover Daily Global 500m
- MOD11A1.006 Terra Land Surface Temperature and Emissivity Daily Global 1km
- MOD11A2.006 Terra Land Surface Temperature and Emissivity 8-Day Global 1km
- MODOCGA.006 Terra Ocean Reflectance Daily Global 1km
- MOD14A1.006: Terra Thermal Anomalies & Fire Daily Global 1km
- MCD43A1.006 MODIS BRDF-Albedo Model Parameters Daily 500m
- MCD15A3H.006 MODIS Leaf Area Index/FPAR 4-Day Global 500m
- MOD17A2H.006: Terra Gross Primary Productivity 8-Day Global 500M 500m
- MOD17A3HGF.006: Terra Net Primary Production Gap-Filled Yearly Global 500m
- MOD16A2.006: Terra Net Evapotranspiration 8-Day Global 500m
- MOD13Q1.006 Terra Vegetation Indices 16-Day Global 250m
- MOD13A1.006 Terra Vegetation Indices 16-Day Global 500m
- MOD13A2.006 Terra Vegetation Indices 16-Day Global 1km
- MOD08_M3.061 Terra Atmosphere Monthly Global Product

Warning: Not supported satellite platforms will raise an *Exception*.

7.3 Usage

The `scale()` method scales each image according to the *Scale* and *Offset* parameters.

Let's take the Sentinel-2 SR image collection as example:

```
S2 = ee.ImageCollection('COPERNICUS/S2_SR')
```

The spectral bands from Sentinel-2 have values close to the (0, 10000) range, but they're unscaled. In order to get the real values, each spectral band must be multiplied by 0.0001, while AOT and WVP bands must be multiplied by 0.001. This scaling is automatically done by the `scale()` method, without any additional parameters:

```
S2.scale()
```

The *Scale* and *Offset* parameters vary according to the satellite platform and the `scale()` method detects the platform and do the scaling according to its parameters.

Let's take now the MOD11A2 product from MODIS. The LST_Day_1km and LST_Night_1km bands must be multiplied by 0.02, the Day_view_time and Night_view_time bands must be multiplied by 0.1, the Emis_31 and Emis_32 bands must be multiplied by 0.002 and added by 0.49, while the Day_view_angl and Night_view_angl bands must be added by -65. All of this scaling is simply done by the `scale()` method:

```
MOD11A2scaled = ee.ImageCollection('MODIS/006/MOD11A2').scale()
```

The `scale()` method can be applied to single images as well:

```
MOD11A2scaled = ee.ImageCollection('MODIS/006/MOD11A2').first().scale()
```

CHAPTER

EIGHT

SPECTRAL INDICES

Let's see how to compute built-in spectral indices with eemont!

Before anything, let's import our modules and authenticate in Google Earth Engine:

```
import ee, eemont

ee.Authenticate()
ee.Initialize()
```

Now, we are ready to go!

8.1 Overview

The eemont package extends the ee.Image and ee.ImageCollection classes with the method `index()`:

8.1.1 ee.Image

index(self[, index, G, C1, C2, L])	Computes one or more spectral indices (indices are added as bands) for an image.

8.1.2 ee.ImageCollection

index(self[, index, G, C1, C2, L])	Computes one or more spectral indices (indices are added as bands) for an image collection.

8.2 Supported Platforms

This method automatically computes spectral indices for the following supported satellite platforms:

8.2.1 Sentinel Missions

- Sentinel-2 MSI: MultiSpectral Instrument, Level-2A
- Sentinel-2 MSI: MultiSpectral Instrument, Level-1C

8.2.2 Landsat Missions

- USGS Landsat 8 Surface Reflectance Tier 1 and 2
- USGS Landsat 7 Surface Reflectance Tier 1 and 2
- USGS Landsat 5 Surface Reflectance Tier 1 and 2
- USGS Landsat 4 Surface Reflectance Tier 1 and 2

Important: It is highly recommended to scale the image (or image collection) before computing spectral indices. See the `scale()` method for more info.

8.3 List of Indices

8.3.1 Vegetation Indices

The following table shows the list of built-in vegetation indices:

Table 3: Built-in vegetation indices.

Index	Description	Reference
BNDVI	Blue Normalized Difference Vegetation Index	Index DataBase BNDVI
CIG	Chlorophyll Index - Green	Index DataBase CIG
CVI	Chlorophyll Vegetation Index	Index DataBase CVI
EVI	Enhanced Vegetation Index	Index DataBase EVI
GBNDVI	Green-Blue Normalized Difference Vegetation Index	Index DataBase GBNDVI
GNDVI	Green Normalized Difference Vegetation Index	Index DataBase GNDVI
GRNDVI	Green-Red Normalized Difference Vegetation Index	Index DataBase GRNDVI
MNDVI	Modified Normalized Difference Vegetation Index	Index DataBase MNDVI
NDVI	Normalized Difference Vegetation Index	Index DataBase NDVI
NGRDI	Normalized Green Red Difference Index	Index DataBase NGRDI
RVI	Ratio Vegetation Index	Index DataBase RVI
SAVI	Soil-Adjusted Vegetation Index	Index DataBase SAVI

8.3.2 Burn Indices

The following table shows the list of built-in burn indices:

Table 4: Built-in burn indices.

Index	Description	Reference
BAI	Burned Area Index	(Martín, 1998) [spanish]
BAIS2	Burned Area Index for Sentinel 2	(Filipponi, 2018)
NBR	Normalized Burn Ratio	Index DataBase NBR

8.3.3 Water Indices

The following table shows the list of built-in water indices:

Table 5: Built-in water indices.

Index	Description	Reference
MNDWI	Modified Normalized Difference Water Index	(Xu, 2006)
NDWI	Normalized Difference Water Index	(McFeeters, 1996)

8.3.4 Snow Indices

The following table shows the list of built-in snow indices:

Table 6: Built-in snow indices.

Index	Description	Reference
NDSI	Normalized Difference Snow Index	(Riggs et al., 1994)

8.4 List of Bands

The following table shows the list of bands used for spectral indices computation:

Table 7: Bands used for spectral indices computation.

Description	Name	Sentinel-2	Landsat 8	Landsat 4, 5, 7
Aerosols	A	B1	B1	
Blue	B	B2	B2	B1
Green	G	B3	B3	B2
Red	R	B4	B4	B3
Red Edge 1	RE1	B5		
Red Edge 2	RE2	B6		
Red Edge 3	RE3	B7		
Red Edge 4	RE4	B8A		
NIR	N	B8	B5	B4
SWIR 1	S1	B11	B6	B5
SWIR 2	S2	B12	B7	B7
Thermal 1	T1		B10	B6
Thermal 2	T2		B11	

Warning: If the satellite platform doesn't have the required bands for computing an index, it won't be computed.

8.5 Usage

The `index()` method computes the specified spectral index and adds it as a new band.

Let's take the Sentinel-2 SR image collection as example (remember to scale your image or image collection!):

```
S2 = ee.ImageCollection('COPERNICUS/S2_SR').scale()
```

By default, the `index()` method computes the NDVI:

```
S2withIndices = S2.index()
S2withIndices.select('NDVI')
```

If required, any of the above-mentioned indices can be computed by modifying the `index` parameter:

```
S2withIndices = S2.index(index = 'EVI')
S2withIndices.select('EVI')
```

Specific index-parameters can be changed, for example, the canopy background adjustment L is set to 1.0 for EVI, but for SAVI it can be changed to 0.5:

```
S2withIndices = S2.index('SAVI',L = 0.5)
S2withIndices.select('SAVI')
```

If more than one index is required, a list of indices can be used:

```
S2withIndices = S2.index(['CIG','NBR','NDWI'])
S2withIndices.select('CIG')
S2withIndices.select('NBR')
S2withIndices.select('NDWI')
```

Indices can also be computed for single images:

```
S2withIndices = S2.first().index(['GBNDVI','MNDVI','EVI'])
S2withIndices.select('GBNDVI')
S2withIndices.select('MNDVI')
S2withIndices.select('EVI')
```

All vegetation indices can be computed by setting `index = vegetation`:

```
S2withIndices = S2.index('vegetation')
S2withIndices.select('NDVI')
S2withIndices.select('GNDVI')
S2withIndices.select('RVI')
# ...
```

All burn indices can be computed by setting `index = burn`:

```
S2withIndices = S2.index('burn')
S2withIndices.select('BAI')
S2withIndices.select('BAIS2')
S2withIndices.select('NBR')
```

All water indices can be computed by setting `index = water`:

```
S2withIndices = S2.index('water')
S2withIndices.select('NDWI')
S2withIndices.select('MNDWI')
```

All snow indices can be computed by setting `index = snow`:

```
S2withIndices = S2.index('snow')
S2withIndices.select('NDSI')
```

If you want to compute all available indices, you can set `index = all`:

```
S2withIndices = S2.index('all')
S2withIndices.select('NDVI')
S2withIndices.select('BAI')
S2withIndices.select('NDWI')
S2withIndices.select('NDSI')
# ...
```

CHAPTER

NINE

DATA CONVERSION

Let's see how to convert non-Earth Engine classes to Earth Engine classes.

Before anything, let's import our modules and authenticate in Google Earth Engine:

```
import ee, eemont
import pandas as pd

ee.Authenticate()
ee.Initialize()
```

Now, we are ready to go!

9.1 Overview

The eemont package extends the pd.DataFrame classes with the method `toEEFeatureCollection()`:

9.1.1 pd.DataFrame

toEEFeatureCollection(self[, latitude, ...])	Converts a pd.DataFrame object into an ee.FeatureCollection object.

9.2 Methods

A table of availabe conversion options is shown below:

Table 2: Available options

From	To	Method
pd.DataFrame	ee.FeatureCollection	`toEEFeatureCollection()`

9.3 Usage

Let's create a pandas data frame:

```
df = pd.DataFrame()
df['lat'] = [2.92846, 4.8927]
df['lon'] = [-76.0269, -75.3188]
df['name'] = ['Nevado del Huila', 'Nevado del Ruiz']
```

This data frame can be easily converted into a ee.FeatureCollection (with no geometries) using the `toEEFeatureCollection()` method for pd.DataFrame classes:

```
fcWithNoGeometries = df.toEEFeatureCollection()
```

If the data frame has latitude and longitude columns, these can be specified in the `latitude` and `longitude` parameters:

```
fcWithGeometries = df.toEEFeatureCollection(latitude = 'lat',longitude = 'lon')
```

CHAPTER

TEN

CHANGELOG

10.1 v0.1.7

10.1.1 New Modules

- The *pd.DataFrame* module was created.
- The *common* module was created (it feeds the `index()`, `scale()` and `maskClouds()` methods for both ee.Image and ee.ImageCollection).

10.1.2 New Features

- The `toEEFeatureCollection()` extended method for pd.DataFrame classes was created.
- The binary operators (+, -, *, /, //, %, **, <<, >>, &, |) were overloaded for ee.Image objects.
- The rich comparisons (<, <=, ==, !=, >, >=) were overloaded for ee.Image objects.
- The unary operators (-, ~) were overloaded for ee.Image objects.

10.1.3 Improvements

- *Exceptions* and *Warnings* were added to most methods.
- Conflicts between the Gain factor and the Green band in the `index()` method were solved.
- `tolerance` and `unit` parameters were added to the `closest()` extended method for ee.ImageCollection classes.
- The `maskClouds()` extended method for ee.Image and ee.ImageCollection classes now supports the following platforms:
  - Sentinel-3 OLCI EFR: Ocean and Land Color Instrument Earth Observation Full Resolution
  - MOD09GA.006 Terra Surface Reflectance Daily Global 1km and 500m
  - MOD09Q1.006 Terra Surface Reflectance 8-Day Global 250m
  - MOD09A1.006 Terra Surface Reflectance 8-Day Global 500m
  - MCD15A3H.006 MODIS Leaf Area Index/FPAR 4-Day Global 500m
  - MOD17A2H.006: Terra Gross Primary Productivity 8-Day Global 500M 500m
  - MOD16A2.006: Terra Net Evapotranspiration 8-Day Global 500m
  - MOD13Q1.006 Terra Vegetation Indices 16-Day Global 250m

  - MOD13A1.006 Terra Vegetation Indices 16-Day Global 500m
  - MOD13A2.006 Terra Vegetation Indices 16-Day Global 1km
- The `scale()` extended method for ee.Image and ee.ImageCollection classes now supports the following platforms:
  - Sentinel-3 OLCI EFR: Ocean and Land Color Instrument Earth Observation Full Resolution
  - MCD43A4.006 MODIS Nadir BRDF-Adjusted Reflectance Daily 500m
  - MCD43A3.006 MODIS Albedo Daily 500m
  - MOD09GQ.006 Terra Surface Reflectance Daily Global 250m
  - MOD09GA.006 Terra Surface Reflectance Daily Global 1km and 500m
  - MOD09Q1.006 Terra Surface Reflectance 8-Day Global 250m
  - MOD09A1.006 Terra Surface Reflectance 8-Day Global 500m
  - MOD10A1.006 Terra Snow Cover Daily Global 500m
  - MOD11A1.006 Terra Land Surface Temperature and Emissivity Daily Global 1km
  - MOD11A2.006 Terra Land Surface Temperature and Emissivity 8-Day Global 1km
  - MODOCGA.006 Terra Ocean Reflectance Daily Global 1km
  - MOD14A1.006: Terra Thermal Anomalies & Fire Daily Global 1km
  - MCD43A1.006 MODIS BRDF-Albedo Model Parameters Daily 500m
  - MCD15A3H.006 MODIS Leaf Area Index/FPAR 4-Day Global 500m
  - MOD17A2H.006: Terra Gross Primary Productivity 8-Day Global 500M 500m
  - MOD17A3HGF.006: Terra Net Primary Production Gap-Filled Yearly Global 500m
  - MOD16A2.006: Terra Net Evapotranspiration 8-Day Global 500m
  - MOD13Q1.006 Terra Vegetation Indices 16-Day Global 250m
  - MOD13A1.006 Terra Vegetation Indices 16-Day Global 500m
  - MOD13A2.006 Terra Vegetation Indices 16-Day Global 1km
  - MOD08_M3.061 Terra Atmosphere Monthly Global Product
- The following vegetation indices were added to the `index()` extended method for ee.Image and ee.ImageCollection:
  - 'GBNDVI' : Green-Blue Normalized Difference Vegetation Index.
  - 'GRNDVI' : Green-Red Normalized Difference Vegetation Index.
  - 'MNDVI' : Modified Normalized Difference Vegetation Index.
- The following snow indices were added to the `index()` extended method for ee.Image and ee.ImageCollection:
  - 'NDSI' : Normalized Difference Snow Index.
- The 'SR' vegetation index was replaced by 'RVI' in the `index()` extended method for ee.Image and ee.ImageCollection.

The eemont package extends Google Earth Engine with pre-processing and processing tools for the most used satellite platforms.

CHAPTER

ELEVEN

HOW DOES IT WORK?

Earth Engine classes, such as ee.Image and ee.ImageCollection, are extended with eemont. New methods are added to these classes to make the code more fluid.

Look at this simple example where a Sentinel-2 collection is pre-processed and processed in just one step:

```
import ee, eemont

ee.Authenticate()
ee.Initialize()

point = ee.Geometry.Point([-76.21, 3.45])

S2 = (ee.ImageCollection('COPERNICUS/S2_SR')
    .filterBounds(point)
    .closest('2020-10-15') # Extended (pre-processing)
    .maskClouds(prob = 70) # Extended (pre-processing)
    .scale() # Extended (pre-processing)
    .index(['NDVI','NDWI','BAIS2'])) # Extended (processing)
```

And just like that, the collection was pre-processed and processed!

CHAPTER

TWELVE

INSTALLATION

Install the latest eemont version from PyPI by running:

```
pip install eemont
```

CHAPTER

THIRTEEN

FEATURES

The following features are extended through eemont:

```
point = ee.Geometry.Point([-76.21, 3.45]) # Example ROI
```

- Overloaded operators (+, -, *, /, //, %, **, <<, >>, &, |, <, <=, ==, !=, >, >=, -, ~):

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

N = S2.select('B8')
R = S2.select('B4')
B = S2.select('B2')

EVI = 2.5 * (N - R) / (N + 6.0 * R - 7.5 * B + 1.0) # Overloaded operators
```

- Clouds and shadows masking:

```
S2 = (ee.ImageCollection('COPERNICUS/S2_SR')
    .maskClouds(prob = 65, cdi = -0.5, buffer = 300) # Clouds and shadows masking
    .first())
```

- Image scaling:

```
MOD13Q1 = ee.ImageCollection('MODIS/006/MOD13Q1').scale() # Image scaling
```

- Spectral indices computation (vegetation, burn, water and snow indices):

```
L8 = (ee.ImageCollection('LANDSAT/LC08/C01/T1_SR')
    .filterBounds(point)
    .maskClouds()
    .scale()
    .index(['GNDVI','NDWI','BAI','NDSI'])) # Indices computation
```

- Closest image to a specific date:

```
S5NO2 = (ee.ImageCollection('COPERNICUS/S5P/OFFL/L3_NO2')
    .filterBounds(point)
    .closest('2020-10-15')) # Closest image to a date
```

CHAPTER

FOURTEEN

METHODS

The above-mentioned features extends both ee.Image and ee.ImageCollection classes:

14.1 ee.Image

index(self[, index, G, C1, C2, L])	Computes one or more spectral indices (indices are added as bands) for an image.
maskClouds(self[, method, prob, maskCirrus, ...])	Masks clouds and shadows in an image (valid just for Surface Reflectance products).
scale(self)	Scales bands on an image.

14.2 ee.ImageCollection

closest(self, date[, tolerance, unit])	Gets the closest image (or set of images if the collection intersects a region that requires multiple scenes) to the specified date.
index(self[, index, G, C1, C2, L])	Computes one or more spectral indices (indices are added as bands) for an image collection.
maskClouds(self[, method, prob, maskCirrus, ...])	Masks clouds and shadows in an image collection (valid just for Surface Reflectance products).
scale(self)	Scales bands on an image collection.

Non-Earth Engine classes such as pd.DataFrame are also extended:

14.3 pd.DataFrame

toEEFeatureCollection(self[, latitude, ...])	Converts a pd.DataFrame object into an ee.FeatureCollection object.

CHAPTER

FIFTEEN

SUPPORTED PLATFORMS

The Supported Platforms for each method can be found in the eemont documentation.

- Masking clouds and shadows supports Sentinel Missions (Sentinel-2 SR and Sentinel-3), Landsat Missions (SR products) and some MODIS Products. Check all details in User Guide > Masking Clouds and Shadows > Supported Platforms.
- Image scaling supports Sentinel Missions (Sentinel-2 and Sentinel-3), Landsat Missions and most MODIS Products. Check all details in User Guide > Image Scaling > Supported Platforms.
- Spectral indices computation supports Sentinel-2 and Landsat Missions. Check all details in User Guide > Spectral Indices > Supported Platforms.
- Getting the closest image to a specific date supports all image collections with the `system:time_start` property.

CHAPTER

SIXTEEN

LICENSE

The project is licensed under the MIT license.

PYTHON MODULE INDEX

e

INDEX