

# Google Earth Engine

Hossein Noorazar

December 13, 2021

## Contents

|           |  |           |
|-----------|--|-----------|
| <b>1</b>  | <b>Preface</b>                                       | <b>2</b>  |
| <b>2</b>  | <b>JavaScript or Python Interface</b>                | <b>3</b>  |
| <b>3</b>  | <b>Landsat Products and Differences</b>              | <b>3</b>  |
| <b>4</b>  | <b>Access a Feature/Entry of a FeatureCollection</b> | <b>5</b>  |
| <b>5</b>  | <b>Add a Property to a Feature</b>                   | <b>5</b>  |
| <b>6</b>  | <b>Find Centroid of Polygons</b>                     | <b>6</b>  |
| <b>7</b>  | <b>Cloud Filtering</b>                               | <b>8</b>  |
| <b>8</b>  | <b>Timelines</b>                                     | <b>11</b> |
| <b>9</b>  | <b>Band Names and Indices</b>                        | <b>11</b> |
| <b>10</b> | <b>Tiny Tips, Big Problems</b>                       | <b>13</b> |

# 1 Preface

Google Earth Engine sucks! Below (Fig. 1) we have a simple example to show GEE is very specific. Accessing to elements/entries of its object is not intuitive. Figuring out every single step is a challenge.



Figure 1: GEE sucks.

Here is the code used for generation of Fig. 1.

## Algorithm 1: GEE Sucks.

```
1. print('Print #1: 3+10', 3+10);
2. var x=3;
3. var y=10;
4. print('Print #2: x+y', x+y);

5. var big_delta_x = 3;
6. print('Print #3: big_delta_x', big_delta_x);

7. var x_big = ee.List.sequence(-125.0, -111.3, big_delta_x);
;

8. print('Print #4: x_big', x_big);
9. print('Print #5: x_big.get(1)', x_big.get(1));
10. print('Print #6',
ee.Number(big_delta_x).add(ee.Number(x_big.get(1))));

11. var aaa = x_big.get(1) + big_delta_x;
12. // print('Print #7: aaa', aaa);
13. print('Print #8: ee.Number(aaa)', ee.Number(aaa));
```

## 2 JavaScript or Python Interface

I think Python should be avoided in this particular case for the following reasons:

1. The interface is too slow,
2. The interface needs authentication every single time,
3. Google does not maintain the Python. Therefore, the functions are first written/updated for the JavaScript (JS) by Google, and the Python equivalents/updates will not be provided in a timely manner (who knows when?).
4. The tutorials for JS is already hard to find, it is much worse for Python. Again, since Google is responsible for JavaScript, it releases the tutorials for it, but not Python.

P.S. tutorials for JS might be abundant, but finding your exact needs might be hard. Even when you find something you may not be sure if that is the best possible solution.

## 3 Landsat Products and Differences

There are different products<sup>1</sup> that fall under different labels; tier 1 vs tier 2, collection 1 vs collection 2, level 1 and level 2. Some of these have the same description on Google developer pages. For example, [USGS Landsat 8 Surface Reflectance Tier 1](#) and [USGS Landsat 8 Surface Reflectance Tier 2](#) have the same description and identical bands. In this particular example we want to use Tier 1. But we need a deeper understanding of differences(?)

Based on the information below and references therein, Collection 2 is an improvement over Collection 1<sup>2</sup>. It seems Collection-2 Level-2 Tier-1 should be the best, but in our plots it was not different from T1\_SR (Fig. 2). Also keep in mind **Collection-2 Level-2 bands must be scaled.**

---

<sup>1</sup>start here to collect some information. some of the products are deprecated and superseded and Google does not show them easily: [here](#)

<sup>2</sup>Is there any time period for which Collection 2 does not exist but 1 does?

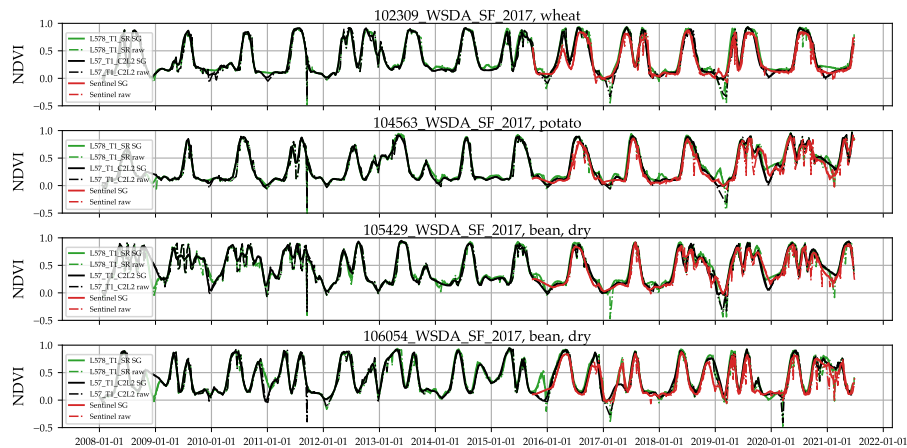


Figure 2: In this plot the data points from Landsat-5, -7, and -8 (Tier 1, Surface Reflectance, from GEE collection LANDSAT/LE07/C01/T1\_SR) are merged together to form one vector. The same is done to Landsat-5 and -7 Collection-2 Level-2 (from GEE collection LANDSAT/LE07/C02/T1.L2). We can see they all are performing well.

Moreover, GEE [1] says “This dataset is the atmospherically corrected surface reflectance from the Landsat 7 ETM+ sensor.” about “USGS Landsat 7 Surface Reflectance Tier 1” (LANDSAT/LE07/C01/T1\_SR). On the other hand, it also says “Caution: This dataset has been superseded by LANDSAT/LC08/C02/T1.L2.”

Collection-1 has only Level-1 data, however, Collection-2 has level-1 as well as Level-2.

**Collection 1** Landsat Collection 1 was established in 2016 to improve archive management. [Learn more about Collection 1 from the USGS.](#)

Landsat Collection 1 consists of Level-1 data products generated from Landsat 8 Operational Land Imager (OLI)/Thermal Infrared Sensor (TIRS), Landsat 7 Enhanced Thematic Mapper Plus (ETM+), Landsat 4-5 Thematic Mapper (TM)\*, and Landsat 1-5 Multispectral Scanner (MSS) instruments. **Collection 1 Tiers:**

**Tier 1** “Landsat scenes with the highest available data quality are placed into Tier 1 and are considered suitable for time-series analysis.” [2]

**Tier 2** “Landsat scenes not meeting Tier 1 criteria during processing are assigned to Tier 2. Tier 2 scenes adhere to the same radiometric standard as Tier 1 scenes, but do not meet the Tier 1 geometry specification due to less accurate orbital information (specific to older Landsat sensors), significant cloud cover, insufficient ground control, or other factors.” [2]

**Collection 2** Landsat Collection 2 marks the second major reprocessing effort on the Landsat archive by the USGS that results in several data product improvements that harness recent advancements in data processing, algorithm development, and data access and distribution capabilities. [Learn more about Collection 2 from the USGS.](#)

Collection-2 Level-1 has different processings for different satellites [3]. It seems Collection-2 level-1 is TOA and Collection-2 level-2 is Surface Reflectance. "Collection-2 Level-2 science products are generated from Collection 2 Level-1 inputs that meet the 176 degrees Solar Zenith Angle constraint and include the required auxiliary data inputs to generate a scientifically viable product." [4]. "**Surface reflectance** (unitless) measures the fraction of incoming solar radiation that is reflected from the Earth's surface to the Landsat sensor. The LEDAPS and LaSRC surface reflectance algorithms correct for the temporally, spatially and spectrally varying scattering and absorbing effects of atmospheric gases, aerosols, and water vapor, which is necessary to reliably characterize the Earth's land surface." [4]. For the enhancement details please see [4].

## 4 Access a Feature/Entry of a FeatureCollection

Suppose your `featurecollection` is called `SF`. In order to access its entries you have to convert it to a `list` and then use `get(.)`:

```
print ('SF.get(0)', SF.toList(4).get(0));
```

where `4` is the size of `SF` known in advance, and `0` is index of first entry of `SF`. In general you can use:

```
print ('SF.get(0)', SF.toList(SF.size()).get(index));
```

Please note if you use `SF.get(0)` you will get an error.

## 5 Add a Property to a Feature

Suppose you have uploaded a shapefile `SF` into your assets. The shapefiles usually have a component/slice called `data` (which is of type `datatable`) that can be accessed via `SF@data` in R. This component stores metadata corresponding to each polygon.

Say each polygon is an agricultural field that has some attributes associated with it such as irrigation type, area of the field, etc. After some computations on GEE you may want to attach these metadata to the output to use later. These metadata is referred to by `properties` on GEE. If you want to manually add a property to a feature you should use:

```
a_feature = a_feature.set('my_property', 1);
```

If you want to copy `properties` (metadata) of `feature_b` into `feature_a` you can do:

```
feature_a = feature_a.copyProperties(feature_b, ['ID', 'Irrigation_type']);
```

where `['ID', 'Irrigation_type']` is a subset of `properties` of `feature_b` to be copied into `feature_a`. I guess if that argument is dropped, then all `properties` will be copied.

## 6 Find Centroid of Polygons

Suppose you have a shapefile that you have uploaded to GEE as an *asset*. Here we will see how to find the centroids of the polygons in the shapefile. Let the name of shapefile be `Our_ShapeFile`. The function to compute centroids of the polygons in `Our_ShapeFile` is given by Alg. 2<sup>3</sup>. Line 4 of the Alg. 2 is keeping the columns of data slice in `Our_ShapeFile`; `Our_ShapeFile@data`.

---

**Algorithm 2:** Find Centroids of Polygons in a Shapefile.

---

```
1. function getCentroid(feature) {  
  2. // Keep this list of properties;  
  3. var keepProperties = ['ID', 'county'];  
  
  4. // Get the centroid of the feature's geometry;  
  5. var centroid = feature.geometry().centroid();  
  
  6. // Return a new Feature, copying properties from the  
    old Feature;  
  7. return ee.Feature(centroid).copyProperties(feature,  
                                                keepProperties);  
8. }  
  
9. var SF = ee.FeatureCollection(Our_ShapeFile);  
10. var centroids_from_GEE = SF.map(getCentroid);
```

---

**Warning:** Imagine your polygon looks like a doughnut (non-convex shape). Then the centroid would be in the center of the disk in the center of the doughnut which is not part of the doughnut/polygon/region of interest. So, if you want to look at an area around the centroid, then that area (or parts of it, depending on how large the area is) would not belong to the polygon (See

---

<sup>3</sup>This algorithm is accessible on GEE [here](#).

Fig. 3b; it is not a doughnut, but it delivers the message!) By adding one line

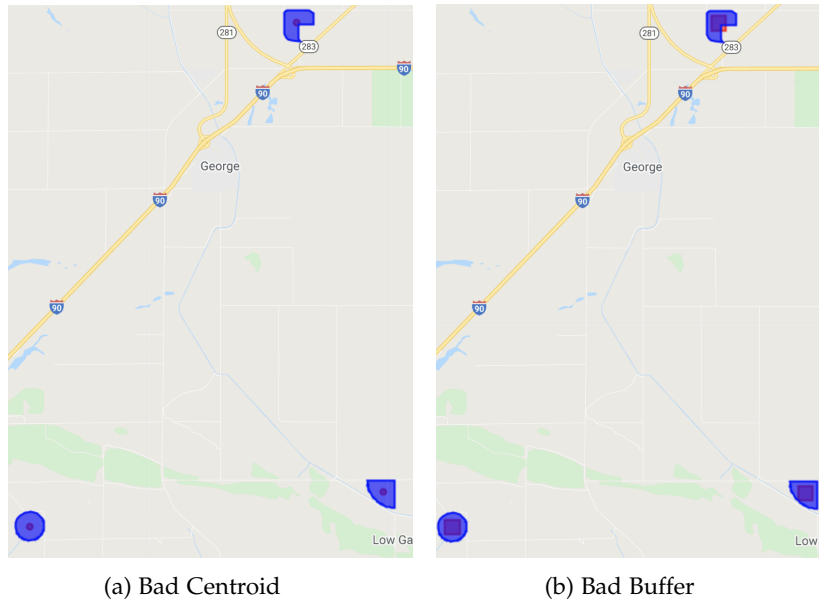


Figure 3: Centroids and buffers around the centroids of polygons in a shapefile.

(line 5.5 in Alg. 3) to the function `getCentroid(.)` we can get a buffer (a rectangular or a circle area) around the centroids.

---

**Algorithm 3:** Make a Buffer Around Centroids of Polygons.

---

```
1. function get_rectangle_around_centroid(feature){
  2. // Keep this list of properties;
  3. var keepProperties = ['ID', 'county'];

  4. // Get the centroid of the feature's geometry;
  5. var centroid = feature.geometry().centroid();

  5.5 centroid = ee.Feature(centroid.buffer(200).bounds());

  6. // Return a new Feature, copying properties from the
  old Feature;
  7. return ee.Feature(centroid).copyProperties(feature,
                                             keepProperties);
  8. }

  9. var SF = ee.FeatureCollection(Our_ShapeFile);
  10. var centroids_from_GEE =
      SF.map(get_rectangle_around_centroid);
```

---

## 7 Cloud Filtering

Handling clouds for Sentinel and Landsat are different. Let us start by **Sentinel**.

First, the followings are equivalent:

- `var filtered = my_IC.filterMetadata('CLOUDY_PIXEL_PERCENTAGE',  
 'less_than', 70);`
- `var filtered = my_IC.filter('CLOUDY_PIXEL_PERCENTAGE < 70')`
- `var filtered = my_IC.filter(ee.Filter.lte('CLOUDY_PIXEL_PERCENTAGE',  
 70))`

They all filter out *images* with cloud cover less than or equal to 70%. Those images will NOT be in our filtered collection. Said differently, our filtered collection may include images that are covered by cloud up to 70%.

This is a pre-filtering step. Later, we can toss out the cloudy *pixels* from every single image.



---

**Algorithm 4:** Filter Cloudy Pixels for Sentinel.

---

```
1. function maskS2clouds(image) {  
  
    2. // Each Sentinel-2 image has a bitmask band with cloud  
       mask information QA60.;  
    3. var qa = image.select('QA60');  
  
    4. // Bits 10 and 11 are clouds and cirrus, respectively;  
    5. var cloudBitMask = 1 << 10 ;  
    6. var cirrusBitMask = 1 << 11 ;  
  
    7. // Both flags should be set to zero, indicating clear  
       conditions.;  
    8. var mask = qa.bitwiseAnd(cloudBitMask).eq(0).and(  
                   qa.bitwiseAnd(cirrusBitMask).eq(0));  
  
    9. // Return the masked and scaled data, without  
       the QA bands.  
    10. return image.updateMask(mask)  
        .divide(10000)  
        .select('B.*')  
        .copyProperties( image, [ 'system:time_start' ] );  
11. }
```

---

**Note 1:** Please note the last line in Alg. 4 is copying the system start time into the image which has nothing to do with clouds. It may be handy later.

**Note 2:** Please note the three (equivalent) pre-filtering of images mentioned above do not exist for Landsat!

Landsat(s) is a different satellite, and therefore, the cloud filtering must be handled differently; the band names that includes cloud information are different between Sentinel and Landsat or even among different Landsats.

Landsat-8 *Surface Reflectance* cloud mask [5]:

---

**Algorithm 5:** Filter Cloudy Pixels for Landsat-8 Tier 1 and 2 Surface Reflectance.

---

```
1. function maskL8sr(image) {  
  2. // Bits 3 and 5 are cloud shadow and cloud,  
    respectively;  
  3. var cloudShadowBitMask = (1 << 3);  
  4. var cloudsBitMask = (1 << 5);  
  
  5. // Get the pixel QA band;  
  6. var qa = image.select('pixel_qa');  
  
  7. // Both flags should be set to zero, indicating clear  
    conditions;  
  8. var mask = qa.bitwiseAnd(cloudShadowBitMask).eq(0)  
    .and(qa.bitwiseAnd(cloudsBitMask).eq(0));  
  
  9. return image.updateMask(mask);  
10. }
```

---

**Note:** This is written for Landsat-8 (Surface Reflectance Tier 1 and 2).

The code for masking the cloudy pixels in Landsat-4, 5, and 7 Surface Reflectance is given by [6] that is given below by Alg. 6:

---

**Algorithm 6:** Filter Cloudy Pixels for Landsat-4, 5, and 7 Tier 1 and 2 Surface Reflectance.

---

```
1. function cloudMaskL457(image) {  
  2. var qa = image.select('pixel_qa');  
  
  3. // If the cloud bit (5) is set and the cloud confidence (7)  
    is high or the cloud shadow bit is set (3),  
    then it's a bad pixel.  
  4. var cloud = qa.bitwiseAnd(1 << 5)  
    .and(qa.bitwiseAnd(1 << 7))  
    .or(qa.bitwiseAnd(1 << 3));  
  
  5. // Remove edge pixels that don't occur in all bands  
  6. var mask2 = image.mask().reduce(ee.Reducer.min());  
  7. return image.updateMask(cloud.not()).updateMask(mask2);  
10. }
```

---

Table 1: Landsat timeline table.

| Satellite | Launched                    | Terminated     |
|-----------|-----------------------------|----------------|
| Landsat 5 | 1 March 1984                | 5 June 2013    |
| Landsat 6 | 5 October 1993              | 5 October 1993 |
| Landsat 7 | 15 April 1999               | Still active   |
| Landsat 8 | 11 February 2013            | Still active   |
| Landsat 9 | 16 September 2021 (planned) | -              |

I have copied the cloud masking functions from GEE development/data-product pages into a script that can be found [here](#) [7]. More on masking clouds of Sentinel-2 and shadows are provided [here](#) by GEE developers [8].

## 8 Timelines

Figure 4 shows the timeline of Landsat satellites [9] and Table 1 shows the exact dates.

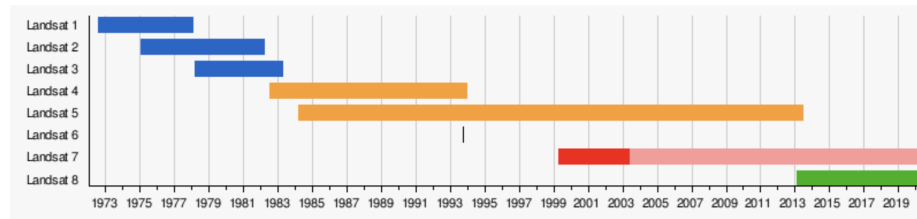


Figure 4: Landsat Timeline.

## 9 Band Names and Indices

Band names are different in each instrument (see Table 2). Hence the indices must be defined differently using proper band names. Below we see some of indices. Table 3 also provides more insight about the bandwidths of the satellites. The bandwidths are very similar. If their minimal differences makes any difference I am not aware of it and do not care. Go nuts if you wish; figure out why, what, how. Bandwidths of Sentinel-2 is found on Wikipedia [10] and Bandwidths of Landsats can be found on GEE pages (e.g. [1]).

Table 2: Some Band Names in Satellites.

| Satellite | NIR | Red | Blue |
|-----------|-----|-----|------|
| Sentinel  | B8  | B4  | B2   |
| Landsat-8 | B5  | B4  | B2   |
| Landsat-7 | B4  | B3  | B1   |
| Landsat-5 | B4  | B3  | B1   |

Table 3: Some Band Wavelengths. The bandwidths are very similar. If their minimal differences makes any difference I am not aware of it and do not care. Go nuts if you wish; figure out why, what, how.

| Satellite       | NIR                        | Red                        | Blue                       |
|-----------------|----------------------------|----------------------------|----------------------------|
| Sentinel-2A     | B8: 0.77 – 0.88 $\mu m$    | B4: 0.65 – 0.68 $\mu m$    | B2: 0.46 – 0.52 $\mu m$    |
| Sentinel-2B     | B8: 0.78 – 0.88 $\mu m$    | B4: 0.65 – 0.68 $\mu m$    | B2: 0.46 – 0.52 $\mu m$    |
| Landsat-8       | B5: 0.85 – 0.88 $\mu m$    | B4: 0.64 – 0.67 $\mu m$    | B2: 0.45 – 0.51 $\mu m$    |
| Landsat-7       | B4: 0.77 – 0.90 $\mu m$    | B3: 0.63 – 0.69 $\mu m$    | B1: 0.45 – 0.52 $\mu m$    |
| Landsat-5       | B4: 0.77 – 0.90 $\mu m$    | B3: 0.63 – 0.69 $\mu m$    | B1: 0.45 – 0.52 $\mu m$    |
| Landsat-7 C2 L2 | SR_B4: 0.77 – 0.90 $\mu m$ | SR_B3: 0.63 – 0.69 $\mu m$ | SR_B1: 0.45 – 0.52 $\mu m$ |
| Landsat-5 C2 L2 | SR_B4: 0.77 – 0.90 $\mu m$ | SR_B3: 0.63 – 0.69 $\mu m$ | SR_B1: 0.45 – 0.52 $\mu m$ |

$$\begin{aligned}
 EVI &= G \times \frac{NIR - R}{NIR + C1 \times R - C2 \times B + L} \\
 EVI_5 &= 2.5 \times \frac{B8 - B4}{B8 + 6 \times B4 - 7.5 \times B2 + 1} \\
 EVI_8 &= 2.5 \times \frac{B5 - B4}{B5 + 6 \times B4 - 7.5 \times B2 + 1} \\
 EVI_7 &= 2.5 \times \frac{B4 - B3}{B4 + 6 \times B3 - 7.5 \times B1 + 1}
 \end{aligned} \tag{1}$$

where  $NIR$  is near infrared,  $R$  is Red,  $B$  is blue,  $EVI_8$  is the Enhanced Vegetation Index (EVI) in Landsat-8 [11], and  $EVI_5$  is the EVI in Sentinel; The NIR band in Landsat-8 is  $B5$  [12] and for Sentinel is  $B8$ .

“EVI is similar to Normalized Difference Vegetation Index (NDVI) and can be used to quantify vegetation greenness. However, EVI corrects for some atmospheric conditions and canopy background noise and is more sensitive in areas with dense vegetation. It incorporates an “ $L$ ” value to adjust for canopy background, “ $C$ ” values as coefficients for atmospheric resistance, and values from the blue band ( $B$ ). These enhancements allow for index calculation as a ratio between the  $R$  and  $NIR$  values, while reducing the background noise, atmospheric noise, and saturation in most cases” [11].

Below are the NDVIs for Landsat-4 to Landsat-7 [13], Landsat-8 [13], and

Sentinel:

$$\begin{aligned}NDVI &= \frac{NIR - R}{NIR + R} \\NDVI_5 &= \frac{B5 - B4}{B5 + B4} \\NDVI_8 &= \frac{B8 - B4}{B8 + B4} \\NDVI_{4-7} &= \frac{B4 - B3}{B4 + B3}\end{aligned}\tag{2}$$

Landsat-7 has 8-day NDVI composite already provided by GEE [14]. This product is based on TOA data which is not perfect! However, it seems running some smoothing methods on it can make it useful.

## 10 Tiny Tips, Big Problems

Some times you may find yourself in a situation for which you are using the biggest sledgehammer to deal with the tiniest nail. In these scenarios the empire of Google does not have a function (for good reasons most likely) to do the job. If brute force is the chosen approach these tips may help.

**Object Types** There are two types of objects or functions. Some are called server-side. Some are called client-side. Here is an [example](#) that shows a client-side object does not work with server-side object.

It is strongly advised to avoid using/writing client-side objects/functions. The client-side objects also make the server/code/interface be very slow, freeze at times.

**Batch Export** This is an example that Google does not think is useful. But if you need to export a collection of images you can do it either using a for-loop for which you may need to look at the previous example. Or, you can use `batch.Download.ImageCollection.toDrive()`. Both of these approaches are demonstrated [here](#).

Two remarks in this regard. First, the function for downloading the image collection as a batch<sup>4</sup> did not work for me, but the for-loop did<sup>5</sup>. The images I exported turned out to be black and white. Secondly, any time a data is exported on GEE interface, you need to click on the **Run** button on **Task** tab. Perhaps Python can be used to avoid this problem, as well as server-side/client-side problem altogether.

---

<sup>4</sup>`batch.Download.ImageCollection.toDrive()`

<sup>5</sup>I was visualizing the images as RGB images and exporting them; `var imageRGB = an_image.visualize(vizParams)`. I am not too sure if the batch download's problem is specific to RGB images.

## References

- [1] Landsat-7 Tier-1 Surface Reflectance Bandwidths. [https://developers.google.com/earth-engine/datasets/catalog/LANDSAT\\_LE07\\_C01\\_T1\\_SR?hl=da](https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LE07_C01_T1_SR?hl=da). [Online; accessed December 13, 2021].
- [2] Landsat collection-1 description. [https://www.usgs.gov/core-science-systems/nli/landsat/landsat-collection-1?qt-science\\_support\\_page\\_related\\_con=1#qt-science\\_support\\_page\\_related\\_con](https://www.usgs.gov/core-science-systems/nli/landsat/landsat-collection-1?qt-science_support_page_related_con=1#qt-science_support_page_related_con). [Online; accessed December 13, 2021].
- [3] Collection-2 Level-1 Description. [https://www.usgs.gov/core-science-systems/nli/landsat/landsat-collection-2-level-1-data?qt-science\\_support\\_page\\_related\\_con=1#qt-science\\_support\\_page\\_related\\_con](https://www.usgs.gov/core-science-systems/nli/landsat/landsat-collection-2-level-1-data?qt-science_support_page_related_con=1#qt-science_support_page_related_con). [Online; accessed December 13, 2021].
- [4] Collection-2 Level-2 Description. <https://www.usgs.gov/core-science-systems/nli/landsat/landsat-collection-2-level-2-science-products>. [Online; accessed December 13, 2021].
- [5] Landsat-8 cloud mask. [https://developers.google.com/earth-engine/datasets/catalog/LANDSAT\\_LC08\\_C01\\_T2\\_SR#bands](https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C01_T2_SR#bands). [Online; accessed December 13, 2021].
- [6] Cloud cleaning in Landsat4-7. [https://developers.google.com/earth-engine/datasets/catalog/LANDSAT\\_LE07\\_C01\\_T2\\_SR](https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LE07_C01_T2_SR). [Online; accessed December 13, 2021].
- [7] Cloud masking functions on GEE - a part of this mini-tutorial. [https://code.earthengine.google.com/?scriptPath=users%2Fhnoorazar%2FGEE\\_Mini\\_Tutorial%3ACloudMaskings](https://code.earthengine.google.com/?scriptPath=users%2Fhnoorazar%2FGEE_Mini_Tutorial%3ACloudMaskings). [Online; accessed December 13, 2021].
- [8] Cloud and shadow masking in Sentinel. <https://developers.google.com/earth-engine/tutorials/community/sentinel-2-s2cloudless>. [Online; accessed December 13, 2021].
- [9] Landsat timelines from Wikipedia. [https://en.wikipedia.org/wiki/Landsat\\_program](https://en.wikipedia.org/wiki/Landsat_program). [Online; accessed December 13, 2021].
- [10] Sentinel Bandwidths. <https://en.wikipedia.org/wiki/Sentinel-2>. [Online; accessed December 13, 2021].
- [11] Landsat-8 EVI. [https://www.usgs.gov/core-science-systems/nli/landsat/landsat-enhanced-vegetation-index?qt-science\\_support\\_page\\_related\\_con=0#qt-science\\_support\\_page\\_related\\_con](https://www.usgs.gov/core-science-systems/nli/landsat/landsat-enhanced-vegetation-index?qt-science_support_page_related_con=0#qt-science_support_page_related_con). [Online; accessed December 13, 2021].

- [12] Landsat-8 band names. [https://www.usgs.gov/faqs/what-are-band-designations-landsat-satellites?qt-news\\_science\\_products=0#qt-news\\_science\\_products](https://www.usgs.gov/faqs/what-are-band-designations-landsat-satellites?qt-news_science_products=0#qt-news_science_products). [Online; accessed December 13, 2021].
- [13] Landsat-4 NDVI. [https://www.usgs.gov/core-science-systems/nli/landsat/landsat-normalized-difference-vegetation-index?qt-science\\_support\\_page\\_related\\_con=0#qt-science\\_support\\_page\\_related\\_con](https://www.usgs.gov/core-science-systems/nli/landsat/landsat-normalized-difference-vegetation-index?qt-science_support_page_related_con=0#qt-science_support_page_related_con). [Online; accessed December 13, 2021].
- [14] Landsat-7 8-day NDVI composite. [https://developers.google.com/earth-engine/datasets/catalog/LANDSAT\\_LE07\\_C01\\_T1\\_8DAY\\_NDVI](https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LE07_C01_T1_8DAY_NDVI). [Online; accessed December 13, 2021].