


Lab 02 - Exploring Image Collections, Temporal Mosaics, Spectral Signatures

Introduction

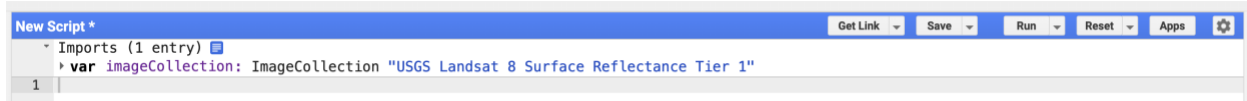
Satellite imagery provided by NASA, USGS, and the European Space Agency (among others) have been preprocessed in order to provide the user with standardized imagery that can be compared between platforms and across time. These datasets, or “collections” as GEE calls them, typically come with not only the individual spectral bands, but also have information layers and metadata that can help the user better understand the quality of the data they are using.

The goal of this exercise will be to explore one of these image collections and then use the data that has been provided with each image collection to help us analyze imagery.


For the purposes of this lab, we will focus on the **Landsat 8 Surface Reflectance** product.

In the GEE Code Editor, use the search bar at the top to locate the *USGS Landsat 8 Surface Reflectance Tier 1* and  it into a new script window.

You should see this at the top of your script editor:



If you click on the collection name, the metadata box should appear again.

Once you have the metadata pop-up box, click on the  icon at the top right to expand the metadata window into a new browser window. Let's explore this page.

In the GEE Data Catalog window for the Landsat 8 Surface Reflectance Collection, you will see a bunch of info that relates to the collection. At the top, there's the “Dataset Availability” which identifies the start and end time for the Collection.

Q1: What are the start and end dates for the Landsat 8 Surface Reflectance collection?

Towards the middle of the page, you will see three tabs: **Description**, **Bands**, and **Image Properties**.

“**Description**” is self-explanatory

“**Bands**” contains a list of the spectral bands (or layers) contained in the collection. Each image in the collection has these bands associated with it.

“**Image Properties**” defines the attribute data contained in the image Metadata record (sorta like an attribute tables in ArcGIS)

Bands -

It is **very important** to understand what data is in the collection we are working with. This is what you should see when you click on “Bands”

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
B4		0.0001	0.636-0.673 μm	Band 4 (red) surface reflectance
B5		0.0001	0.851-0.879 μm	Band 5 (near infrared) surface reflectance
B6		0.0001	1.566-1.651 μm	Band 6 (shortwave infrared 1) surface reflectance
B7		0.0001	2.107-2.294 μm	Band 7 (shortwave infrared 2) surface reflectance
B10	Kelvin	0.1	10.60-11.19 μm	Band 10 brightness temperature. This band, while originally collected with a resolution of 100m / pixel, has been resampled using cubic convolution to 30m.
B11	Kelvin	0.1	11.50-12.51 μm	Band 11 brightness temperature. This band, while originally collected with a resolution of 100m / pixel, has been resampled using cubic convolution to 30m.
sr_aerosol Aerosol attributes				
+ Bitmask for sr_aerosol				
pixel_qa Pixel quality attributes generated from the CFMASK algorithm.				
+ Bitmask for pixel_qa				
radsat_qa Radiometric saturation QA				
+ Bitmask for radsat_qa				

This particular collection has 12 layers, there are 9 spectral bands: B1-B7, and B10-B11 (huh!?, what happened to B8 & B9?? More on that later). The three other layers are information layers which focus on various aspects of per pixel image quality.

Looking at the info for each band, there are columns for “Units”, “Scale”, “Wavelength”, and “Description”. I think you can figure out the “Description” part, but the others need some explaining.

Units: units of measure. Not sure why all of the bands except the thermal bands (10 & 11) are blank. The units of measure for this collection for bands 1-7 are % reflectance (what % of total incoming light was reflected and recorded by the sensor for that wavelength)

Scale: These are values used to calculate the original pixel reflectance value from the pixel value stored in the collection. For most of these datasets, the unit of measure is **reflectance** calculated as a proportion of 1. Therefore, reflectance ranges between 0 and 1 with 0 = no reflectance, and 1 = 100% reflectance. For various reasons, these numbers are scaled to integer values. For instance, we can take a reflectance value of 0.50, multiply it by 100 and get 50. This is what we typically do to calculate a percentage. We just moved the decimal place over two spaces. The **Scale** value for this collection is ‘**0.0001**’ (one ten thousandths), meaning that the reflectance value was multiplied by 10,000 for the collection. If you want to revert to actual reflectance values, you would divide each pixel by 10,000. See, doesn’t that make sense???....., actually it does make sense if you know the back story.

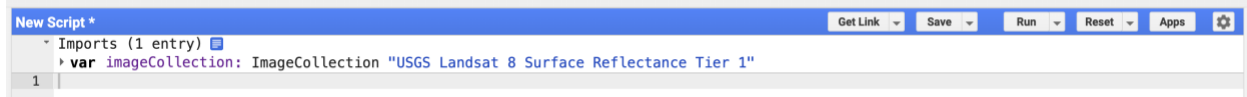
Wavelength: This is the portion of the EM spectrum that a particular band is sensitive to.

Let's look at the “**pixel_qa**” layer. If you click on the ‘+’ sign, it will expand to explain what is contained in that layer. In this case (qa layers in other collections may be different), the layer is made up of different “Bits” spanning 0-10. I'll have a lecture on “Bits” later. This layer type has been created using a technique called “bit packing”. [Read more about the Landsat bit-packed QA bands](#).

In the Landsat 8 surface reflectance collection, the “pixel_qa” layer identifies on a per-pixel basis whether a particular pixel contains water, clouds, shadow, snow, terrain occlusion or any combination of those. We can, for instance, use this layer to identify and “extract” pixels in our collection that are identified as clouds or cloud shadows.

Let's do some coding:

Going back to the code editor, you should have a blank screen with only the collection listed at the top:



The variable name that GEE gives this dataset by default is “**imageCollection**”. You can rename this by clicking on it and typing another, shorter and more descriptive name. I've renamed mine to “**L8_SR**”. You can name it anything you want.



Let's figure out what we're working with.

Type in the command: **print (<whateverYouCalledYourCollection>.size())** and hit **Run**.

*** if I ever use “<something>”, that means you should replace it (including the <>'s) with whatever you used.

Mine looks like this: `print(L8_SR.size());`

Q2: How many different Landsat 8 surface reflectance images are stored in this collection?

Q2a: What are the start and end dates for this collection?

Q3a: What Bits in the pixel_qa layer are associated with cloud and cloud shadow?

This image collection has a lot of data in it spanning 7 years. Too much for us to deal with.

Let's reduce the collection by filtering for a specific time period and a specific location.

If you look in the “Docs” tab (left window), you will find the command documentation. Since we're working with an image collection, let's look at the commands available to manipulate image collections, namely “**ee.ImageCollection**”. Expand that category and you will see a number of different tools that we can apply to our collection. We want to **filter** for a certain time window and location. If you scan down the list, you will come to the filter section and you will

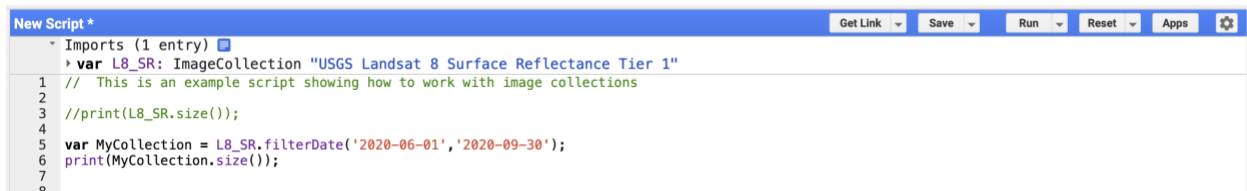
see a number of ways to filter our collection. The two we want are **filterDate** and **filterBounds**. The **filterDate** command has two parameters: **start** and **end** (start and end dates). In the script window apply the **filterDate** command to select only those images that were collected between two dates. Here's my command:

```
var MyCollection = L8_SR.filterDate('2020-06-01','2020-09-30')
```

This will create another collection called “MyCollection” and fill it with images from the L8_SR collection that were collected between June 1 and Sept 30 of 2020.

You can change “MyCollection” to anything you want and you can also change the dates. I'd encourage you to try different dates just for practice.

This is what my code looks like so far:




Notice that I commented out the first print statement since it takes a long time to count up more than 1 million images.

Q3: How many images are in your collection after filtering for a date range?? (No right answer since it will depend on your date range)

Whatever number you got, the collection is probably still too big and also still covers the entire globe. If that's the area you want to map, no problem. But what if you want to look at a smaller region?

There are a number of ways to filter for a specific geographic area. You've done it already in Lab 1 using a county boundary. Here's another way.

Pan and zoom the map display so that you're looking at the area you are interested in.

At the top left of the map display, you will see some drawing tools: 

Select the rectangle (you can also use the other ones) and draw around the area you want to filter your collection to.

I went to Mauritania in NW Africa. Why did I go there?? Because I wanted to, that's why!!

When I drew my rectangle, I noticed a new variable appear in my script just below where the image collection is identified. This one is called “**geometry**”. I renamed mine to “**Richat**”.

Here's my script now:



My map display area looks like this:



Add the “**filterBounds**” filter to the script. You can tag it to the end of the **filterDate** command by separating them with a ‘.’

```
New Script *
Imports (2 entries)
var L8_SR: ImageCollection "USGS Landsat 8 Surface Reflectance Tier 1"
var Richat: Polygon, 4 vertices
1 // This is an example script showing how to work with image collections
2
3 //print(L8_SR.size());
4
5 var MyCollection = L8_SR.filterDate('2020-06-01','2020-09-30').filterBounds(Richat);
6 print(MyCollection.size());
7
```

Hit Run.

Q4: How many images are in your collection after filtering for date range and geography?

Let's display what we have.

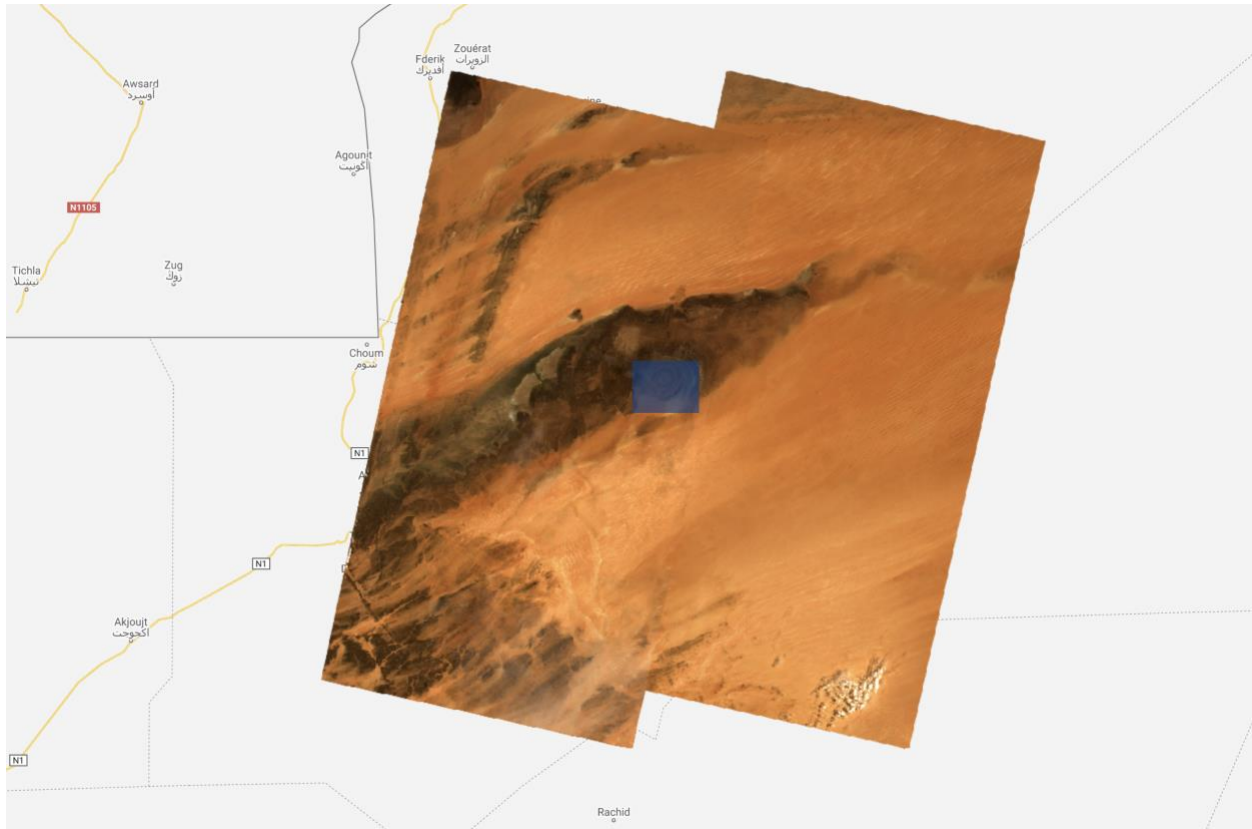
Image collections typically have more than one image. You need to tell GEE which image you want to look at or how you would like to visualize all of the images in the collection.

The simplest way is to flatten the collection into one image using a simple statistical function

I typically use a “**median()**” function to select the one pixel in the temporal stack that represents the median value of all of the other pixels at the same location. The resulting image will have parts of all of the other images that cover the same area on the earth's surface. I could also use “**mean()**”, “**max()**”, “**min()**”, “**mode()**”, or a number of other methods.

Here's what I have so far:

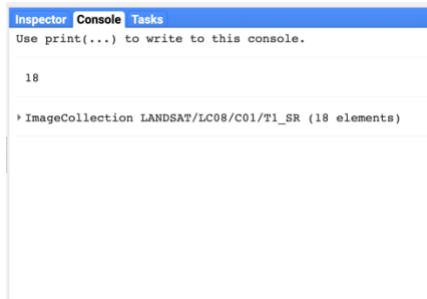
```
New Script *
Imports (2 entries)
var L8_SR: ImageCollection "USGS Landsat 8 Surface Reflectance Tier 1"
var Richat: Polygon, 4 vertices
1 // This is an example script showing how to work with image collections
2
3 //print(L8_SR.size());
4
5 var MyCollection = L8_SR.filterDate('2020-06-01','2020-09-30').filterBounds(Richat);
6 print(MyCollection.size());
7
8 var MyImage = MyCollection.median();
9 Map.addLayer(MyImage);
10
```



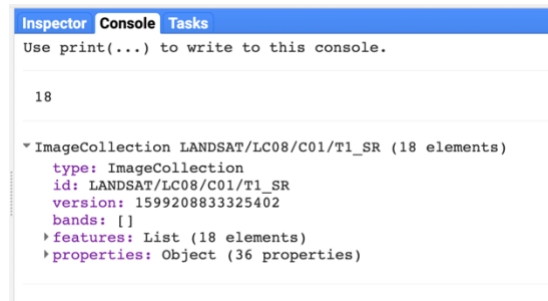
As you might be able to tell, the area I chose just happened to be at the intersection of four different image frames.

Add another **print** command to print out the information in your collection and hit run again. **print(MyCollection).**

Here are my results:



I had two print commands. The first listed the size of my collection (18) and the second the image collection itself. If I click on the image collection entry in the Console, it will open up to show all of the components of the collection.



Clicking on the “**features**” category will expand the view to show the different images in the collection.

Now I can see entries for every image in my collection. Clicking on any one of the images will allow me to explore the metadata for each image.

Looking through my images, I noticed that almost all of them had cloud cover of 0% (click on one image to open it up and then on “**properties**” to see the metadata for that specific image. The first item will be “**CLOUD_COVER**” with a number beside it. This is the estimated % cloud cover for the entire image.

```

Inspector Console Tasks
Use print(...) to write to this console.

18

* ImageCollection LANDSAT/LC08/C01/T1_SR (18 elements)
  type: ImageCollection
  id: LANDSAT/LC08/C01/T1_SR
  version: 1599208833325402
  bands: []
  *features: List (18 elements)
    *0: Image LANDSAT/LC08/C01/T1_SR/LC08_202045_20200625 (12 bands)
    *1: Image LANDSAT/LC08/C01/T1_SR/LC08_202045_20200711 (12 bands)
    *2: Image LANDSAT/LC08/C01/T1_SR/LC08_202045_20200727 (12 bands)
    *3: Image LANDSAT/LC08/C01/T1_SR/LC08_202045_20200812 (12 bands)
    *4: Image LANDSAT/LC08/C01/T1_SR/LC08_202046_20200625 (12 bands)
    *5: Image LANDSAT/LC08/C01/T1_SR/LC08_202046_20200711 (12 bands)
    *6: Image LANDSAT/LC08/C01/T1_SR/LC08_202046_20200727 (12 bands)
    *7: Image LANDSAT/LC08/C01/T1_SR/LC08_202046_20200812 (12 bands)
    *8: Image LANDSAT/LC08/C01/T1_SR/LC08_203045_20200616 (12 bands)
    *9: Image LANDSAT/LC08/C01/T1_SR/LC08_203045_20200702 (12 bands)
    *10: Image LANDSAT/LC08/C01/T1_SR/LC08_203045_20200718 (12 bands)
    *11: Image LANDSAT/LC08/C01/T1_SR/LC08_203045_20200803 (12 bands)
    *12: Image LANDSAT/LC08/C01/T1_SR/LC08_203045_20200819 (12 bands)
    *13: Image LANDSAT/LC08/C01/T1_SR/LC08_203046_20200616 (12 bands)
    *14: Image LANDSAT/LC08/C01/T1_SR/LC08_203046_20200702 (12 bands)
    *15: Image LANDSAT/LC08/C01/T1_SR/LC08_203046_20200718 (12 bands)
    *16: Image LANDSAT/LC08/C01/T1_SR/LC08_203046_20200803 (12 bands)
    *17: Image LANDSAT/LC08/C01/T1_SR/LC08_203046_20200819 (12 bands)
  *properties: Object (36 properties)
  
```

```

Inspector Console Tasks
11: Image LANDSAT/LC08/C01/T1_SR/LC08_203045_20200803 (12 bands)
12: Image LANDSAT/LC08/C01/T1_SR/LC08_203045_20200819 (12 bands)
13: Image LANDSAT/LC08/C01/T1_SR/LC08_203046_20200616 (12 bands)
14: Image LANDSAT/LC08/C01/T1_SR/LC08_203046_20200702 (12 bands)
15: Image LANDSAT/LC08/C01/T1_SR/LC08_203046_20200718 (12 bands)
  type: Image
  id: LANDSAT/LC08/C01/T1_SR/LC08_203046_20200718
  version: 1597223205159814
  *bands: List (12 elements)
  *properties: Object (24 properties)
    CLOUD_COVER: 14.39
    CLOUD_COVER_LAND: 14.39
    EARTH_SUN_DISTANCE: 1.016306
    ESPA_VERSION: 2_23_0_1b
    GEOMETRIC_RMSE_MODEL: 7.453
    GEOMETRIC_RMSE_MODEL_X: 4.948
    GEOMETRIC_RMSE_MODEL_Y: 5.573
    IMAGE_QUALITY_OLI: 9
    IMAGE_QUALITY_TIRS: 9
    LANDSAT_ID: LC08_L1TP_203046_20200718_20200722_01_T1
    LEVEL1_PRODUCTION_DATE: 1595434731000
    PIXEL_QA_VERSION: generate_pixel_qa_1.6.0
    SATELLITE: LANDSAT_8
    SENSING_TIME: 2020-07-18T11:13:24.2873520Z
    SOLAR_AZIMUTH_ANGLE: 83.896469
    SOLAR_ZENITH_ANGLE: 23.830101
    SR_APP_VERSION: LaSRC_1.3.0
    WRS_PATH: 203
    WRS_ROW: 46
    system:asset_size: 631594530
    *system:footprint: LinearRing, 21 vertices
    system:index: LC08_203046_20200718
    system:time_end: 1595070804287
    system:time_start: 1595070804287
  *16: Image LANDSAT/LC08/C01/T1_SR/LC08_203046_20200803 (12 bands)
  *17: Image LANDSAT/LC08/C01/T1_SR/LC08_203046_20200819 (12 bands)
  
```

There was one image (#15) that had quite a bit of cloud cover, 14.39%.

Depending on where your area is, you may have more images with cloud cover (like the Amazon Basin).

There are two ways to deal with cloudy images.

The first is to use the filter tools to only select images with 0% cloud cover. Here’s how that command works:

filterMetadata(name, operator, value)

in my case:

```

var MyCollection = L8_SR.filterDate('2020-06-01','2020-09-30')
    .filterBounds(Richat)
    .filterMetadata('CLOUD_COVER','equals',0);
  
```

Notice that I just added it to the end of my other filter commands and I also broke the command up into three different lines since it was starting to get a little long. I used “**equals**”, but you can use a number of other logical functions (you can find the possibilities in the **Docs** tab).

That added filter reduced my collection to just 9 images.

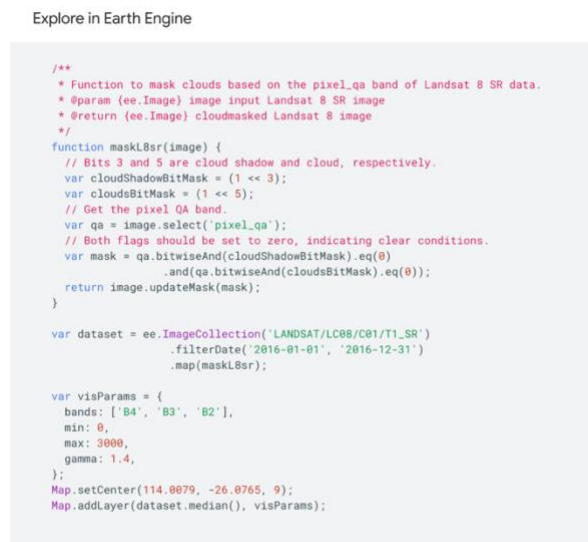
Now, I’m chose to work in the Sahara Desert where clouds tend to be a little sparse. What if you’re working in another location where there are NO (or few) images with 0% cloud cover?

This is where the **pixel_qa** layer comes in very handy. Using the “**CLOUD_COVER**” attribute of 0% eliminated entire images regardless of how many clouds or where they were located. Clouds tend to move so a cloud covering our study area today will be gone tomorrow. The

pixel_qa layer is a finer tool that can eliminate just the pixels that were cloud covered, not the entire image.

Go back to the Earth Engine Data Catalog web page that described the Landsat 8 surface reflectance collection.

At the bottom of that page you will see a script that will display the collection.



At the top of that script is a **function** called ‘**maskL8sr**’. The function is bounded by `{}`’s. Copy that section of the script.



And paste it into the top of your script.

Delete the `.filterMetadata` command and replace it with:

`.map(maskL8sr);`

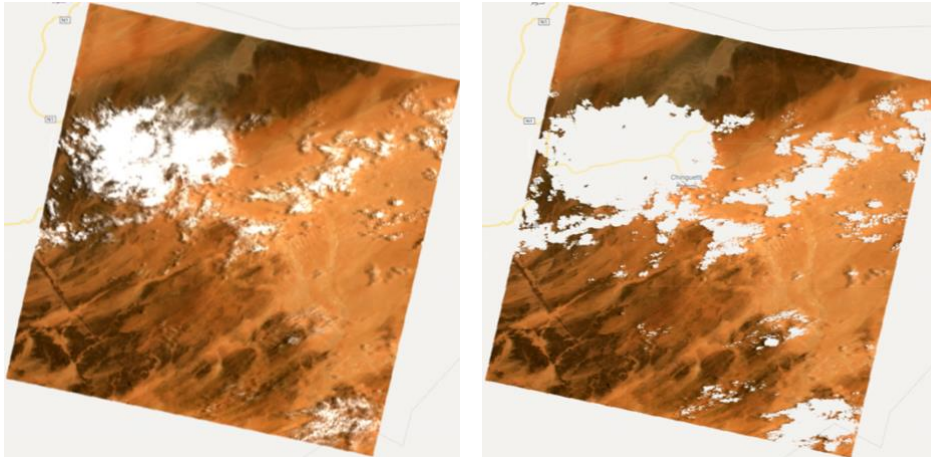
This is what mine looks like:



Run the script. The number of images in your collection should be back to where they were (mine is back to 18) and you should have a cloud-free image on your screen.

The function called `maskL8sr` iterates through all of the images in your collection and masks out any cloud or cloud shadow identified by the **pixel_qa** layer. Look at the function. It identified the specific Bits associated with clouds and cloud shadow (3 & 5), converted them to a mask and then used that mask to “delete” all of the pixels that coincided with the mask.

Here’s the image with the clouds before and after:



Notice that all of the clouds as well as the cloud shadows have been removed from the image.

////////////////

BTW, (An aside)

This particular image in my collection is the 15th of 18 images. I've shown you the **ImageCollection.first()** command which gets the first image in the collection. There's no automatic way, though, to select images from the middle of the collection.

For instance, if the first image was cloud free, but was collected during the winter when vegetation was not growing, you could either restrict the time to the summer or you could select another image from the list that you have. There are actually multiple ways of solving this problem. Here's one example:

Take the individual image ID's and put them in a list:

var ImageList = MyCollection.toList(MyCollection.size()); - in the Docs tab look up 'toList' and 'size' under 'ee.ImageCollection'

print(ImageList);

From that list, select one of the entries (remember that lists start with 0). The collection that I have consists of 18 different images numbered 0-17. Therefore, to get the 2nd image in the collection, I would do this:

var myImage = ee.Image(ImageList.get(1));

This method is equivalent to the first command we used to display a specific image:

var myimage = ee.Image('LANDSAT/LC08/C01/T1_SR/LC08_202045_20200711');

Change get(1) to get(2) , or 3, 4,5,....28, etc.

////////////////

Once you have your image collection “mapped” through the **maskL8sr** function to mask clouds and shadow, we can display an image that is completely cloud free (assuming that the **pixel_qa** image was 100% accurate in identifying clouds).

Since we can’t display image collections on the screen, only images, we need to convert the image collection to a single composite image made up of pixels selected from each image in that collection.

In the case that I have (and that you will also have), there are multiple images in my collection that are cloud free. Therefore, how do we select the pixel from a specific image that we want to use in our output image??

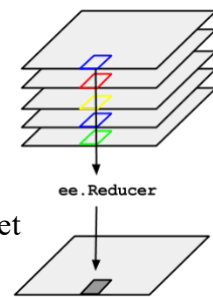
Going back to the “Docs” tab and then under **ee.ImageCollection**, there are a number of things that we can do to “reduce” the collection to a single image.

Here’s a link to the GEE user guide that describes [image collection reductions](#)

This graphic from that user guide page illustrates what we need to do:

Given a stack of overlapping images in a collection, for every pixel, select the reflectance value from each image that meets a particular criterion. Standard image reducers are “**median()**”, “**mean()**”, “**min()**”, “**max()**”. There are others.

I tend to use “**median()**”. The median value is the middle number of a sorted set of numbers. So, the median of ‘0,1,2,3,4,5,6,7,8’ is 4. Why do I use the median and not mean? Well, the mean can be biased by the distribution of numbers, while the median represents the true middle number. You should try each of these reducers to see what they do.

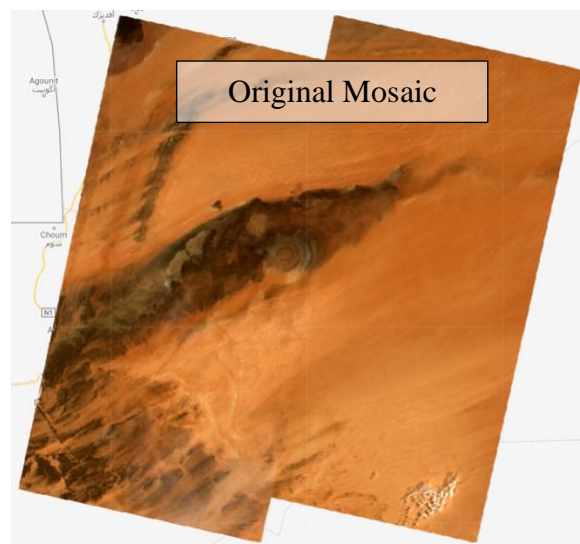
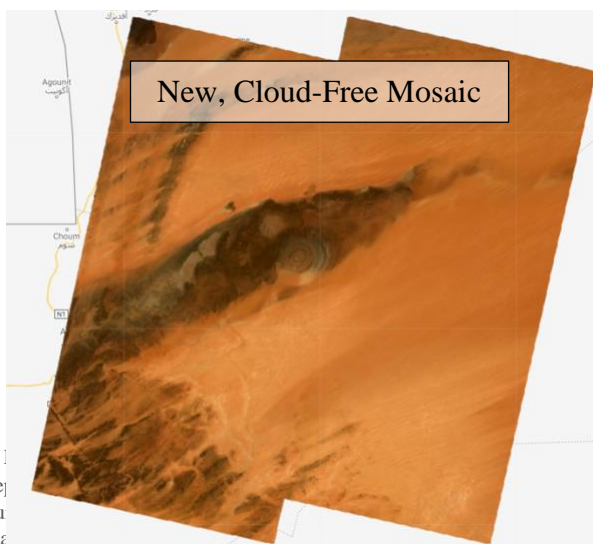


To convert my collection to a single image, I use this command:

```
var MyImage = MyCollection.median();
```

Then:

```
Map.addLayer(MyImage, Vis, “My Image”);
```

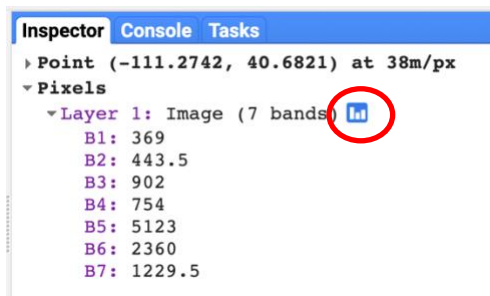


Spectral Signatures



When you finished the first in-class assignment, you no-doubt ran across this common figure that depicts the general shape of a spectral signature for vegetation. All surfaces have a spectral signature and it's this signature that allows us to map land cover types.

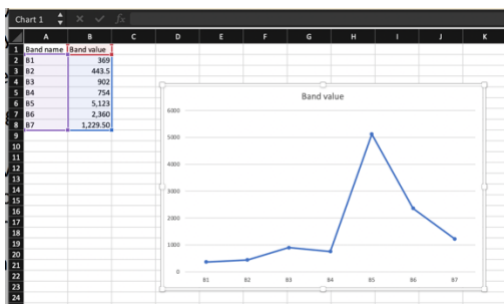
There's a couple of ways to extract the spectral signature for a given land cover type in GEE.

The brute-force method is to display the image you want to sample and then use the “**Inspector**” tab (right window) and click the mouse on the pixel you want to sample. The Inspector window should display the pixel values for the point that you clicked on:

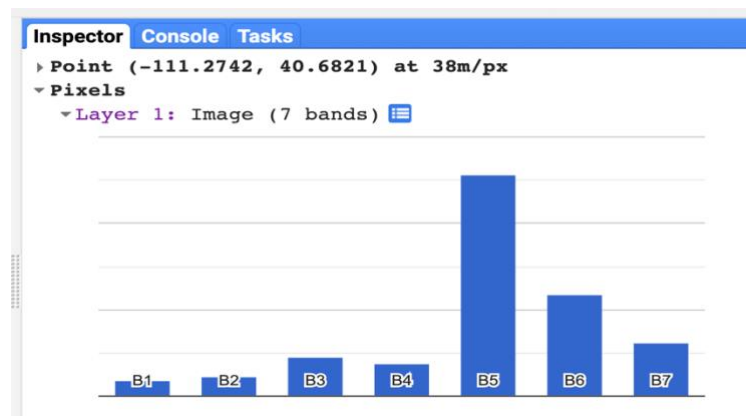


Ok, it's a bar graph. Not a line graph like we want. But that's ok, it basically shows what the reflectance values are per band.

You should see a  to the upper right of the bar graph. Click on it and the graph will expand to a new browser window. If you want a different kind of graph, click on the  button and you can download the data for that pixel, then import it into Microsoft Excel and make any kind of graph that you want. Like this:




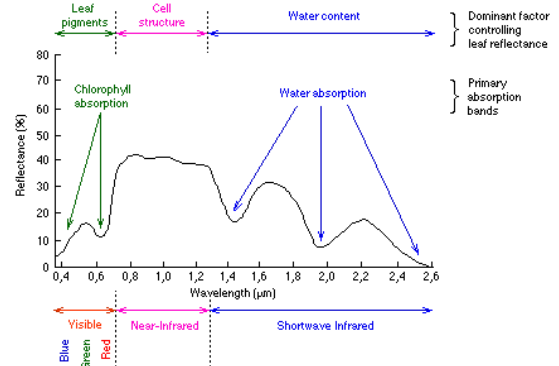
If you click on that small blue icon, the display will change from the numeric values to a graph.



That's the brute force way. It works, but not very sophisticated. We can get GEE do generate the plots for us.

Here's a link to a [spectral signature script](#) that I found in the example section of the “**Scripts**” window and modified to fit what I wanted to do.

I've imported three points by using the tool 



https://www.researchgate.net/figure/Reflectance-spectrum-of-different-materials_fig5_221915805

Q5: In this script:

- a. what image collection am I using and how is it filtered?
- b. What does this section of script do?
 - a. What does 'ff0000' or the other entries like it mean?
- c. On line 43, why do I only select spectral bands 1-7?
- d. What spectral bands am I displaying for the image on the screen?
- e. Looking at the plot that prints to the Console window, identify the same three bands. (hint, the graph shows them in order 1-7).
 - a. Why do fire scars appear red in the image (not the graph)??
 - b. Why does vegetation appear green?
 - c. Why does water look dark blue and black?
 - i. Why is Strawberry Reservoir closer to a black color than Utah lake which appears dark blue?
- d. In the **visParams** section of the script, change bands: ['B7', 'B5', 'B3'] to bands: 'B5', 'B3', 'B7'] and hit '**Run**'....., **what happened??**

```
//Colors for spectral plot lines
var COLOR = {
  FIRE: 'ff0000',
  WATER: '0000ff',
  AG: '00ff00'
};
```