Image Manipulation: Bands, Arithmetic, Thresholds, and Masks (F2.0)

Authors

Karen Dyson, Andréa Puzzi Nicolau, David Saah, and Nicholas Clinton

Overview

Once images have been identified in Earth Engine, they can be viewed in a wide array of band combinations for targeted purposes. For users who are already versed in remote sensing concepts, this chapter shows how to do familiar tasks on this platform; for those who are entirely new to such concepts, it introduces the idea of band combinations.

Learning Outcomes

- Understanding what spectral indices are and why they are useful.
- Being introduced to a range of example spectral indices used for a variety of purposes.

Assumes you know how to:

- Import images and image collections, filter, and visualize (Part F1).

Introduction to Theory

Spectral indices are based on the fact that different objects and land covers on the Earth's surface reflect different amounts of light from the Sun at different wavelengths. In the visible part of the spectrum, for example, a healthy green plant reflects a large amount of green light while absorbing blue and red light—which is why it appears green to our eyes. Light also arrives from the Sun at wavelengths outside what the human eye can see, and there are large differences in reflectances between living and nonliving land covers, and between different types of vegetation, both in the visible and outside the visible wavelengths. We visualized this earlier, in Chaps. F1.1 and F1.3 when we mapped color-infrared images (Fig. F2.0.1).

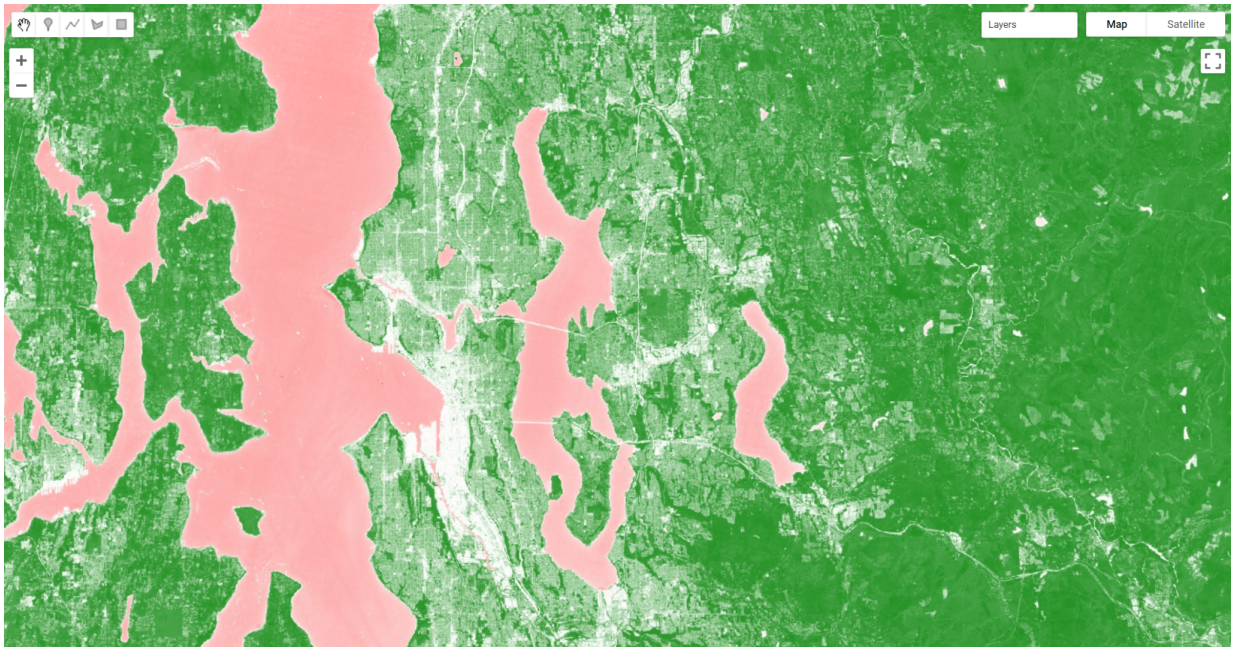

Fig. F2.0.6 NDVI image of Sentinel-2 imagery over Seattle, Washington, USA

Inspect the image. We can see that vegetated areas are darker green while non-vegetated locations are white and water is pink. If we use the **Inspector** to query our image, we can see that parks and other forested areas have an NDVI over about 0.5. Thus, it would make sense to define areas with NDVI values greater than 0.5 as forested, and those below that threshold as not forested.

Now let's define that value as a threshold and use it to threshold our vegetated areas.

```
// Implement a threshold.
var seaVeg = seaNDVI.gt(0.5);

// Map the threshold.
Map.addLayer(seaVeg,
    {
        min: 0,
        max: 1,
        palette: ['white', 'green']
    },
    'Non-forest vs. Forest');
```

The `gt` method is from the family of Boolean operators—that is, `gt` is a function that performs a test in each pixel and returns the value 1 if the test evaluates to true, and 0

otherwise. Here, for every pixel in the image, it tests whether the NDVI value is greater than 0.5. When this condition is met, the layer `seaVeg` gets the value 1. When the condition is false, it receives the value 0.

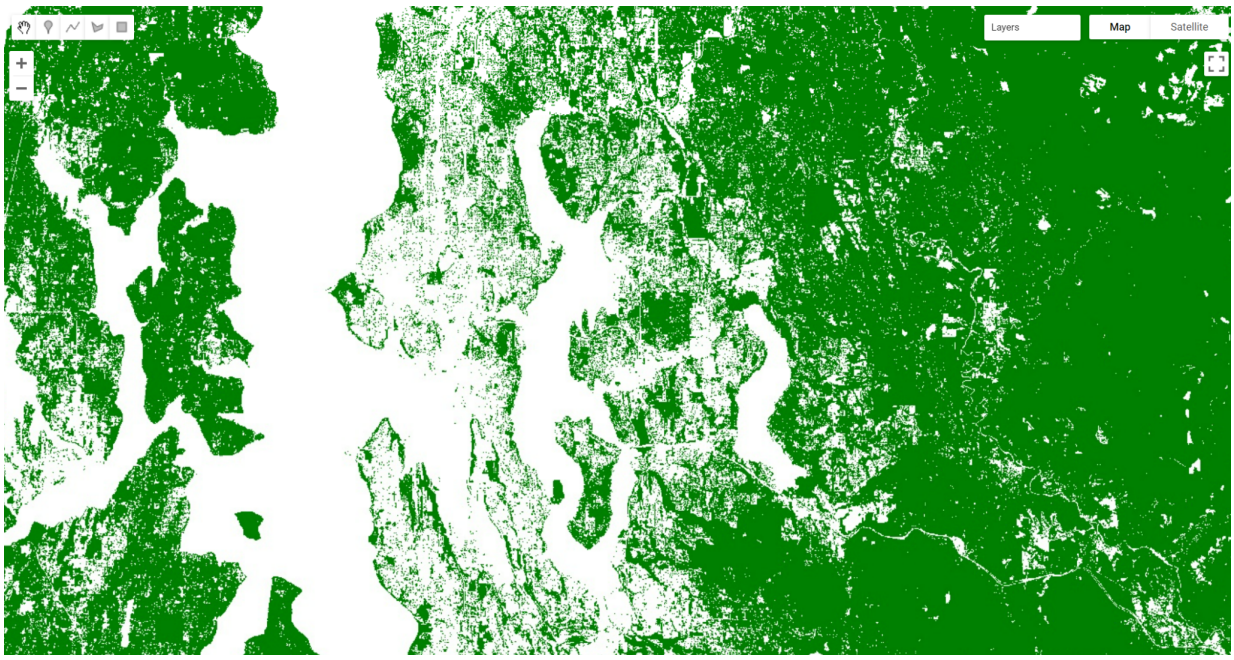

Fig. F2.0.7 Thresholded forest and non-forest image based on NDVI for Seattle, Washington, USA

Use the **Inspector** tool to explore this new layer. If you click on a green location, that NDVI should be greater than 0.5. If you click on a white pixel, the NDVI value should be equal to or less than 0.5.

Other operators in this Boolean family include less than (`lt`), less than or equal to (`lte`), equal to (`eq`), not equal to (`neq`), and greater than or equal to (`gte`) and more.

Building Complex Categorizations with .where

A binary map classifying NDVI is very useful. However, there are situations where you may want to split your image into more than two bins. Earth Engine provides a tool, the `where` method, that conditionally evaluates to true or false within each pixel depending on the outcome of a test. This is analogous to an *if* statement seen commonly in other languages. However, to perform this logic when programming for Earth Engine, we avoid using the JavaScript *if* statement. Importantly, JavaScript *if* commands are not calculated on Google's servers, and can create serious problems when running your code—in effect, the servers try to ship all of the information to be executed to your own

computer's browser, which is very underequipped for such enormous tasks. Instead, we use the `where` clause for conditional logic.

Suppose instead of just splitting the forested areas from the non-forested areas in our NDVI, we want to split the image into likely water, non-forested, and forested areas. We can use `where` and thresholds of -0.1 and 0.5. We will start by creating an image using `ee.Image`. We then clip the new image so that it covers the same area as our `seaNDVI` layer.

```
// Implement .where.
// Create a starting image with all values = 1.
var seaWhere = ee.Image(1)
    // Use clip to constrain the size of the new image.
    .clip(seaNDVI.geometry());

// Make all NDVI values less than -0.1 equal 0.
seaWhere = seaWhere.where(seaNDVI.lte(-0.1), 0);

// Make all NDVI values greater than 0.5 equal 2.
seaWhere = seaWhere.where(seaNDVI.gte(0.5), 2);

// Map our layer that has been divided into three classes.
Map.addLayer(seaWhere,
    {
        min: 0,
        max: 2,
        palette: ['blue', 'white', 'green']
    },
    'Water, Non-forest, Forest');
```

There are a few interesting things to note about this code that you may not have seen before. First, we're not defining a new variable for each `where` call. As a result, we can perform many `where` calls without creating a new variable each time and needing to keep track of them. Second, when we created the starting image, we set the value to 1. This means that we could easily set the bottom and top values with one `where` clause each. Finally, while we did not do it here, we can combine multiple `where` clauses using `and` and `or`. For example, we could identify pixels with an intermediate level of NDVI using `seaNDVI.gte(-0.1).and(seaNDVI.lt(0.5))`.

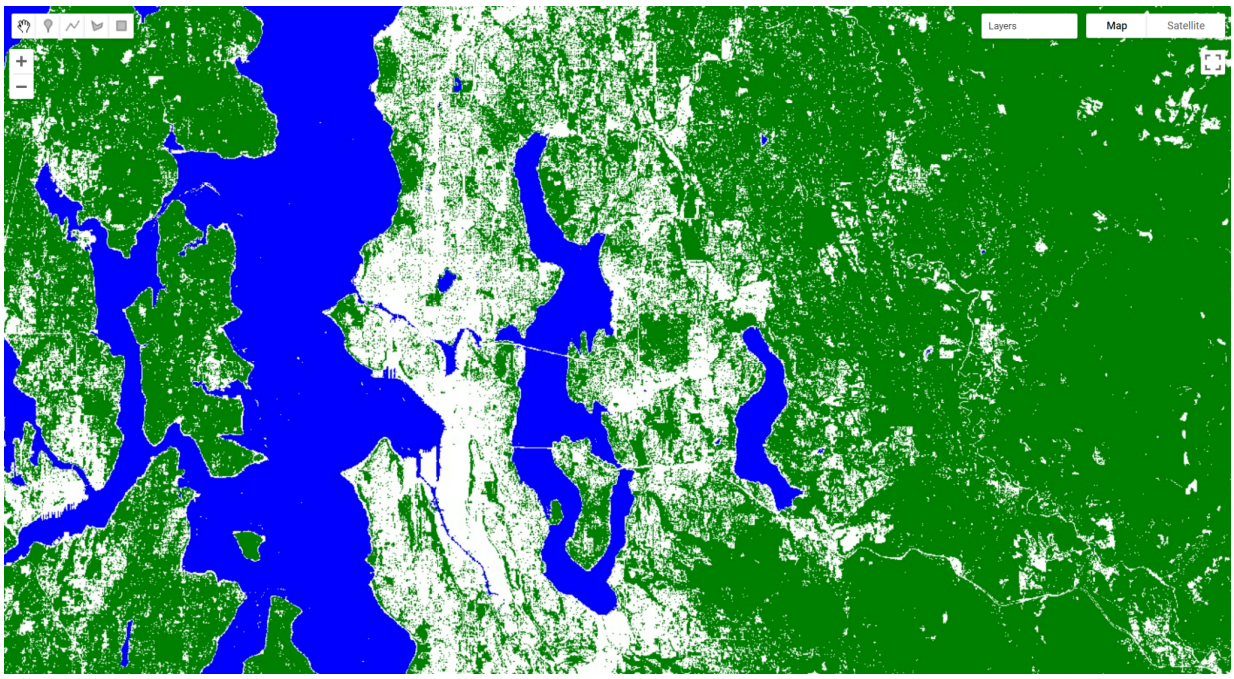

Fig. F2.0.8 Thresholded water, forest, and non-forest image based on NDVI for Seattle, Washington, USA.

Masking Specific Values in an Image

Masking an image is a technique that removes specific areas of an image—those covered by the mask—from being displayed or analyzed. Earth Engine allows you to both view the current mask and update the mask.

```
// Implement masking.
// View the seaVeg layer's current mask.
Map.centerObject(seaPoint, 9);
Map.addLayer(seaVeg.mask(), {}, 'seaVeg Mask');
```

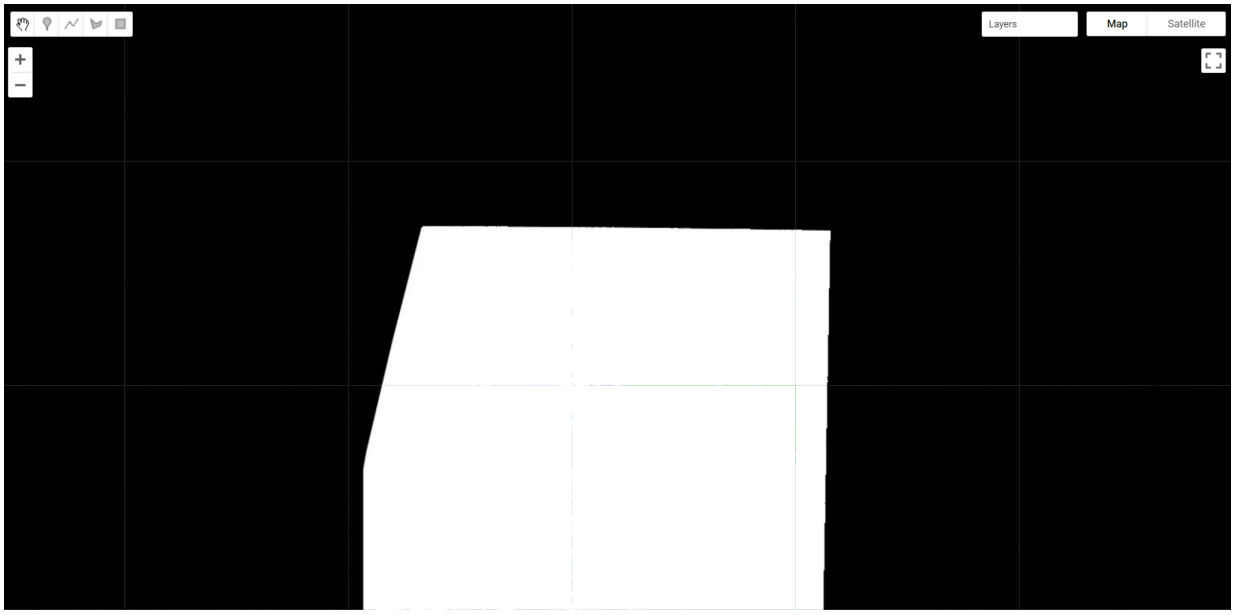

Fig. F2.0.9 The existing mask for the `seaVeg` layer we created previously

You can use the **Inspector** to see that the black area is masked and the white area has a constant value of 1. This means that data values are mapped and available for analysis within the white area only.

Now suppose we only want to display and conduct analyses in the forested areas. Let's mask out the non-forested areas from our image. First, we create a binary mask using the equals (`eq`) method.

```
// Create a binary mask of non-forest.
var vegMask = seaVeg.eq(1);
```

In making a mask, you set the values you want to see and analyze to be a number greater than 0. The idea is to set unwanted values to get the value of 0. Pixels that had 0 values become masked out (in practice, they do not appear on the screen at all) once we use the `updateMask` method to add these values to the existing mask.

```
// Update the seaVeg mask with the non-forest mask.
var maskedVeg = seaVeg.updateMask(vegMask);

// Map the updated Veg layer
Map.addLayer(maskedVeg,
    {
```

```
        min: 0,
        max: 1,
        palette: ['green']
    },
    'Masked Forest Layer');
```

Turn off all of the other layers. You can see how the `maskedVeg` layer now has masked out all non-forested areas.

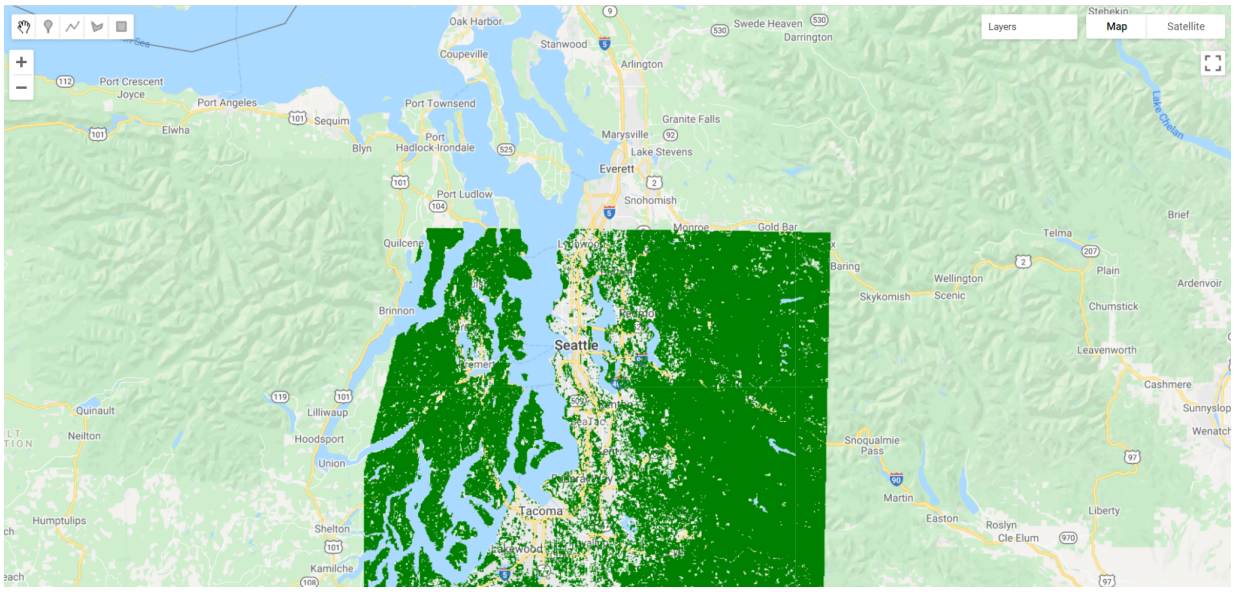

Fig. F2.0.10 An updated mask now displays only the forested areas. Non-forested areas are masked out and transparent.

Map the updated mask for the layer and you can see why this is.

```
// Map the updated mask
Map.addLayer(maskedVeg.mask(), {}, 'maskedVeg Mask');
```

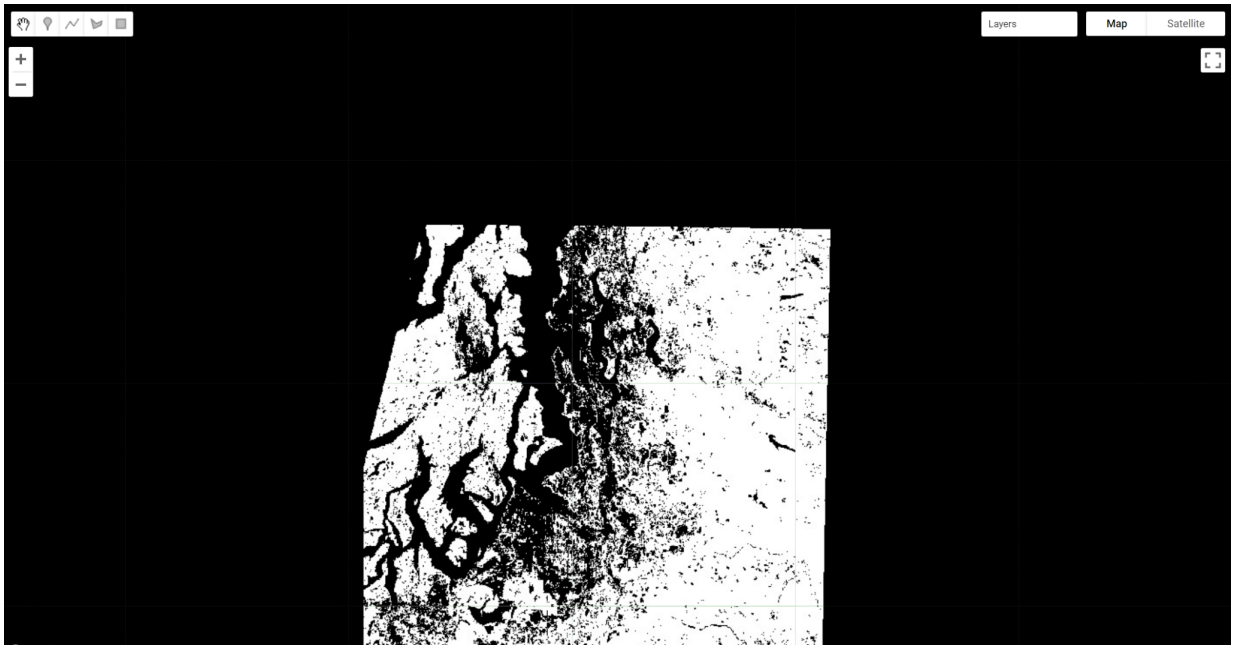

Fig. F2.0.11 The updated mask. Areas of non-forest are now masked out as well (black areas of the image).

Remapping Values in an Image

Remapping takes specific values in an image and assigns them a different value. This is particularly useful for categorical datasets, including those you read about in Chap. F1.2 and those we have created earlier in this chapter.

Let's use the `remap` method to change the values for our `seaWhere` layer. Note that since we're changing the middle value to be the largest, we'll need to adjust our palette as well.

```
// Implement remapping.
// Remap the values from the seaWhere layer.
var seaRemap = seaWhere.remap([0, 1, 2], // Existing values.
    [9, 11, 10]); // Remapped values.

Map.addLayer(seaRemap,
    {
        min: 9,
        max: 11,
        palette: ['blue', 'green', 'white']
    },
    'Remapped Values');
```

Use the inspector to compare values between our original `seaWhere` (displayed as Water, Non-Forest, Forest) and the `seaRemap`, marked as “Remapped Values.” Click on a forested area and you should see that the Remapped Values should be 10, instead of 2 (Fig. F2.0.12).

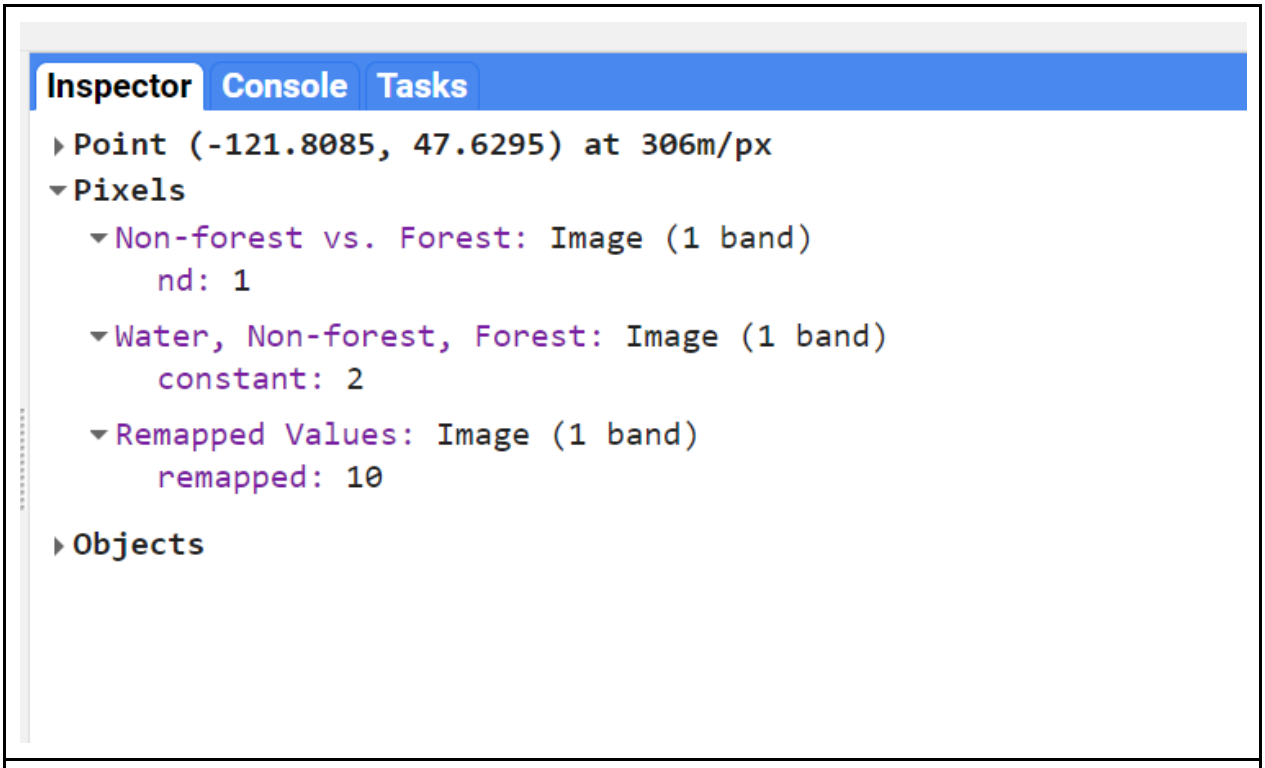

Fig. F2.0.12 For forested areas, the remapped layer has a value of 10, compared with the original layer, which has a value of 2. You may have more layers in your **Inspector**.

Code Checkpoint F20b. The book’s repository contains a script that shows what your code should look like at this point.

Synthesis

Assignment 1. In addition to vegetation indices and other land cover indices, you can use properties of different soil types to create geological indices. The Clay Minerals Ratio (CMR) is one of these. This index highlights soils containing clay and alunite, which absorb radiation in the SWIR portion (2.0–2.3 μm) of the spectrum.

$$CMR = \frac{SWIR\ 1}{SWIR\ 2}$$

SWIR 1 should be in the 1.55–1.75 µm range, and SWIR 2 should be in the 2.08–2.35 µm range. Calculate and display CMR at the following point: `ee.Geometry.Point(-100.543, 33.456)`. Don't forget to use `Map.centerObject`.

We've selected an area of Texas known for its clay soils. Compare this with an area without clay soils (for example, try an area around Seattle or Tacoma, Washington, USA). Note that this index will also pick up roads and other paved areas.

Assignment 2. Calculate the Iron Oxide Ratio, which can be used to detect hydrothermally altered rocks (e.g., from volcanoes) that contain iron-bearing sulfides which have been oxidized (Segal, 1982).

Here's the formula:

$$IOR = \frac{Red}{Blue}$$

Red should be the 0.63–0.69 µm spectral range and Blue the 0.45–0.52 µm. Using Landsat 8, you can also find an interesting area to map by considering where these types of rocks might occur.

Assignment 3. Calculate the Normalized Difference Built-Up Index (NDBI) for the `sfoImage` used in this chapter.

The NDBI was developed by Zha et al. (2003) to aid in differentiating urban areas (e.g., densely clustered buildings and roads) from other land cover types. The index exploits the fact that urban areas, which generally have a great deal of impervious surface cover, reflect SWIR very strongly. If you like, refer back to Fig. F2.0.2.

The formula is:

$$NDBI = \frac{SWIR - NIR}{SWIR + NIR}$$

Using what we know about Sentinel-2 bands, compute NDBI and display it.

Bonus: Note that NDBI is the negative of NDWI computed earlier. We can prove this by using the JavaScript *reverse* method to reverse the palette used for NDWI in Earth Engine. This method reverses the order of items in the JavaScript list. Create a new palette for NDBI using the reverse method and display the map. As a hint, here is code to use the reverse method.

```
var barePalette = waterPalette.reverse();
```

Conclusion

In this chapter, you learned how to select multiple bands from an image and calculate indices. You also learned about thresholding values in an image, slicing them into multiple categories using thresholds. It is also possible to work with one set of class numbers and remap them quickly to another set. Using these techniques, you have some of the basic tools of image manipulation. In subsequent chapters you will encounter more complex and specialized image manipulation techniques, including pixel-based image transformations (Chap. F3.1), neighborhood-based image transformations (Chap. F3.2), and object-based image analysis (Chap. F3.3).

Feedback

To review this chapter and make suggestions or note any problems, please go now to bit.ly/EEFA-review. You can find summary statistics from past reviews at bit.ly/EEFA-reviews-stats.

References

Baig MHA, Zhang L, Shuai T, Tong Q (2014) Derivation of a tasselled cap transformation based on Landsat 8 at-satellite reflectance. Remote Sens Lett 5:423–431. https://doi.org/10.1080/2150704X.2014.915434

Crist EP (1985) A TM tasseled cap equivalent transformation for reflectance factor data. Remote Sens Environ 17:301–306. https://doi.org/10.1016/0034-4257(85)90102-6

Drury SA (1987) Image interpretation in geology. Geocarto Int 2:48. https://doi.org/10.1080/10106048709354098

Gao BC (1996) NDWI - A normalized difference water index for remote sensing of vegetation liquid water from space. Remote Sens Environ 58:257–266. https://doi.org/10.1016/S0034-4257(96)00067-3

Huang C, Wylie B, Yang L, et al (2002) Derivation of a tasselled cap transformation based on Landsat 7 at-satellite reflectance. Int J Remote Sens 23:1741–1748. https://doi.org/10.1080/01431160110106113

Jackson RD, Huete AR (1991) Interpreting vegetation indices. Prev Vet Med 11:185–200. https://doi.org/10.1016/S0167-5877(05)80004-2

Martín MP (1998) Cartografía e inventario de incendios forestales en la Península Ibérica a partir de imágenes NOAA-AVHRR. Universidad de Alcalá

McFeeters SK (1996) The use of the Normalized Difference Water Index (NDWI) in the delineation of open water features. Int J Remote Sens 17:1425–1432. https://doi.org/10.1080/01431169608948714

Nath B, Niu Z, Mitra AK (2019) Observation of short-term variations in the clay minerals ratio after the 2015 Chile great earthquake (8.3 Mw) using Landsat 8 OLI data. J Earth Syst Sci 128:1–21. https://doi.org/10.1007/s12040-019-1129-2

Schultz M, Clevers JGPW, Carter S, et al (2016) Performance of vegetation indices from Landsat time series in deforestation monitoring. Int J Appl Earth Obs Geoinf 52:318–327. https://doi.org/10.1016/j.jag.2016.06.020

Segal D (1982) Theoretical basis for differentiation of ferric-iron bearing minerals, using Landsat MSS data. In: Proceedings of Symposium for Remote Sensing of Environment, 2nd Thematic Conference on Remote Sensing for Exploratory Geology, Fort Worth, TX. pp 949–951

Souza Jr CM, Roberts DA, Cochrane MA (2005) Combining spectral and spatial information to map canopy damage from selective logging and forest fires. Remote Sens Environ 98:329–343. https://doi.org/10.1016/j.rse.2005.07.013

Souza Jr CM, Siqueira JV, Sales MH, et al (2013) Ten-year Landsat classification of deforestation and forest degradation in the Brazilian Amazon. Remote Sens 5:5493–5513. https://doi.org/10.3390/rs5115493