Raster/Vector Conversions (F5.1)

Authors

Keiko Nomura, Samuel Bowers

Overview

The purpose of this chapter is to review methods of converting between raster and vector data formats, and to understand the circumstances in which this is useful. By way of example, this chapter focuses on topographic elevation and forest cover change in Colombia, but note that these are generic methods that can be applied in a wide variety of situations.

Learning Outcomes

- Understanding raster and vector data in Earth Engine and their differing properties.
- Knowing how and why to convert from raster to vector.
- Knowing how and why to convert from vector to raster.
- Write a function and `map` it over a `FeatureCollection`.

Assumes you know how to:

- Import images and image collections, filter, and visualize (Part F1).
- Understand distinctions among `Image`, `ImageCollection`, `Feature` and `FeatureCollection` Earth Engine objects (Part F1, Part F2, Part F5).
- Perform basic image analysis: select bands, compute indices, create masks (Part F2).
- Perform image morphological operations (Chap. F3.2).
- Understand the `filter`, `map`, `reduce` paradigm (Chap. F4.0).
- Write a function and `map` it over an `ImageCollection` (Chap. F4.0).
- Use `reduceRegions` to summarize an image in irregular shapes (Chap. F5.0).

Introduction to Theory

Raster data consists of regularly spaced pixels arranged into rows and columns, familiar as the format of satellite images. Vector data contains geometry features (i.e., points,

lines, and polygons) describing locations and areas. Each data format has its advantages, and both will be encountered as part of GIS operations.

Raster and vector data are commonly combined (e.g., extracting image information for a given location or clipping an image to an area of interest); however, there are also situations in which conversion between the two formats is useful. In making such conversions, it is important to consider the key advantages of each format. Rasters can store data efficiently where each pixel has a numerical value, while vector data can more effectively represent geometric features where homogenous areas have shared properties. Each format lends itself to distinctive analytical operations, and combining them can be powerful.

In this exercise, we'll use topographic elevation and forest change images in Colombia as well as a protected area feature collection to practice the conversion between raster and vector formats, and to identify situations in which this is worthwhile.

Practicum

Section 1. Raster to Vector Conversion

If you have not already done so, you can add the book's code repository to the Code Editor by entering https://code.earthengine.google.com/?accept_repo=projects/gee-edu/book (or the short URL bit.ly/EEFA-repo) into your browser. The book's scripts will then be available in the script manager panel to view, run, or modify. If you have trouble finding the repo, you can visit bit.ly/EEFA-repo-help for help.

Section 1.1. Raster to Polygons

In this section we will convert an elevation image (raster) to a feature collection (vector). We will start by loading the Global Multi-Resolution Terrain Elevation Data 2010 and the Global Administrative Unit Layers 2015 dataset to focus on Colombia. The elevation image is a raster at 7.5 arc-second spatial resolution containing a continuous measure of elevation in meters in each pixel.

```
// Load raster (elevation) and vector (colombia) datasets.
var elevation = ee.Image('USGS/GMTED2010').rename('elevation');
var colombia = ee.FeatureCollection(
        'FAO/GAUL_SIMPLIFIED_500m/2015/level0')
    .filter(ee.Filter.equals('ADM0_NAME', 'Colombia'));

// Display elevation image.
Map.centerObject(colombia, 7);
Map.addLayer(elevation, {
    min: 0,
    max: 4000
}, 'Elevation');
```

When converting an image to a feature collection, we will aggregate the categorical elevation values into a set of categories to create polygon shapes of connected pixels with similar elevations. For this exercise, we will create four zones of elevation by grouping the altitudes to 0-100 m = 0, 100–200 m = 1, 200–500 m = 2, and >500 m = 3.

```
// Initialize image with zeros and define elevation zones.
var zones = ee.Image(0)
    .where(elevation.gt(100), 1)
    .where(elevation.gt(200), 2)
    .where(elevation.gt(500), 3);

// Mask pixels below sea level (<= 0 m) to retain only land areas.
// Name the band with values 0-3 as 'zone'.
zones = zones.updateMask(elevation.gt(0)).rename('zone');

Map.addLayer(zones, {
    min: 0,
    max: 3,
    palette: ['white', 'yellow', 'lime', 'green'],
    opacity: 0.7
}, 'Elevation zones');
```

We will convert this zonal elevation image in Colombia to polygon shapes, which is a vector format (termed a `FeatureCollection` in Earth Engine), using the `ee.Image.reduceToVectors` method. This will create polygons delineating connected pixels with the same value. In doing so, we will use the same projection and spatial resolution as the image. Please note that loading the vectorized image in the native

resolution (231.92 m) takes time to execute. For faster visualization, we set a coarse scale of 1,000 m.

```
var projection = elevation.projection();
var scale = elevation.projection().nominalScale();

var elevationVector = zones.reduceToVectors({
    geometry: colombia.geometry(),
    crs: projection,
    scale: 1000, // scale
    geometryType: 'polygon',
    eightConnected: false,
    labelProperty: 'zone',
    bestEffort: true,
    maxPixels: 1e13,
    tileScale: 3 // In case of error.
});

print(elevationVector.limit(10));

var elevationDrawn = elevationVector.draw({
    color: 'black',
    strokeWidth: 1
});
Map.addLayer(elevationDrawn, {}, 'Elevation zone polygon');
```

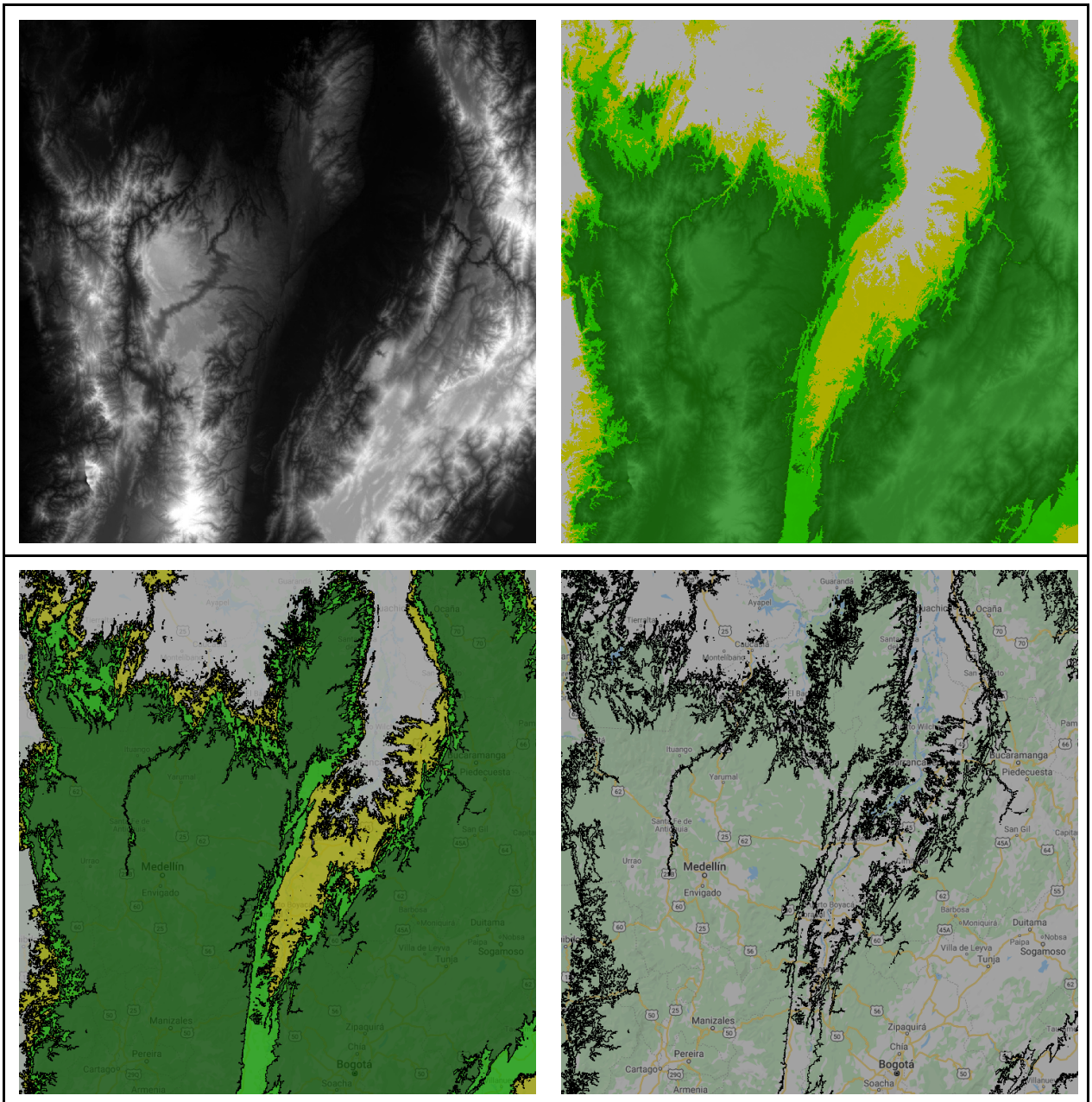


Fig. F5.1.1 Raster-based elevation (top left) and zones (top right), vectorized elevation zones overlaid on the raster (bottom-left) and vectorized elevation zones only (bottom-right)

You may have realized that polygons consist of complex lines, including some small polygons with just one pixel. That happens when there are no surrounding pixels of the same elevation zone. You may not need a vector map with such details—if, for instance, you want to produce a regional or global map. We can use a morphological reducer `focalMode` to simplify the shape by defining a neighborhood size around a pixel. In this

example, we will set the kernel radius as four pixels. This operation makes the resulting polygons look much smoother, but less precise (Fig. F5.1.2).

```
var zonesSmooth = zones.focalMode(4, 'square');

zonesSmooth = zonesSmooth.reproject(projection.atScale(scale));

Map.addLayer(zonesSmooth, {
    min: 1,
    max: 3,
    palette: ['yellow', 'lime', 'green'],
    opacity: 0.7
}, 'Elevation zones (smooth)');

var elevationVectorSmooth = zonesSmooth.reduceToVectors({
    geometry: colombia.geometry(),
    crs: projection,
    scale: scale,
    geometryType: 'polygon',
    eightConnected: false,
    labelProperty: 'zone',
    bestEffort: true,
    maxPixels: 1e13,
    tileScale: 3
});

var smoothDrawn = elevationVectorSmooth.draw({
    color: 'black',
    strokeWidth: 1
});
Map.addLayer(smoothDrawn, {}, 'Elevation zone polygon (smooth)');
```

We can see now that the polygons have more distinct shapes with many fewer small polygons in the new map (Fig. F5.1.2). It is important to note that when you use methods like `focalMode` (or other, similar methods such as `connectedComponents` and `connectedPixelCount`), you need to reproject according to the original image in order to display properly with zoom using the interactive Code Editor.

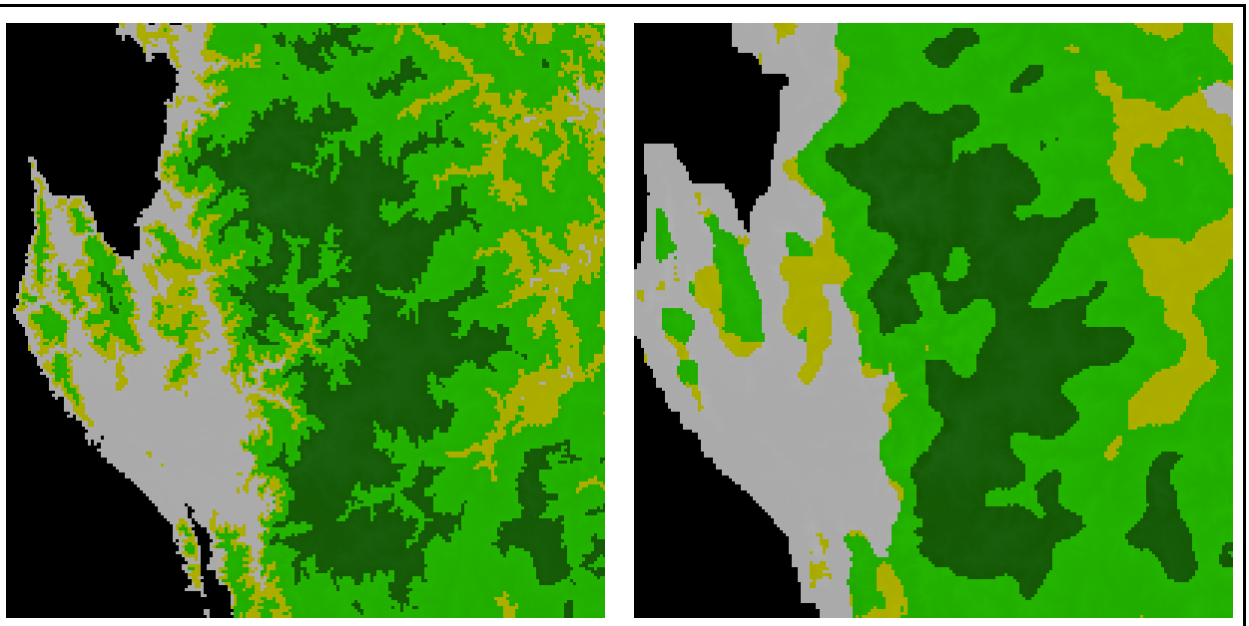
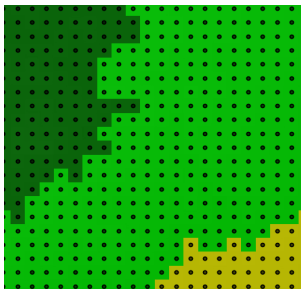


Fig. F5.1.2 Before (left) and after (right) applying `focalMode`

Section 1.2. Raster to Points

Lastly, we will convert a small part of this elevation image into a point vector dataset. For this exercise, we will use the same example and build on the code from the previous subsection. This might be useful when you want to use geospatial data in a tabular format in combination with other conventional datasets such as economic indicators (Fig. F5.1.3).



	A	B	C	D	E
	system:index	Elevation	Lat	Long	.geo
0	0	129	-0.8678472222	-89.55118056	{"type":"Multil
1	1	140	-0.8678472222	-89.54909722	{"type":"Multil
2	2	161	-0.8678472222	-89.54701389	{"type":"Multil
3	3	171	-0.8678472222	-89.54493056	{"type":"Multil
4	4	174	-0.8678472222	-89.54284722	{"type":"Multil
5	5	192	-0.8678472222	-89.54076389	{"type":"Multil

Fig. F5.1.3 Elevation point values with latitude and longitude

The easiest way to do this is to use `sample` while activating the geometries parameter. This will extract the points at the centroid of the elevation pixel.

```
var geometry = ee.Geometry.Polygon([
    [-89.553, -0.929],
```

```
    [-89.436, -0.929],
    [-89.436, -0.866],
    [-89.553, -0.866],
    [-89.553, -0.929]
]);

// To zoom into the area, un-comment and run below
// Map.centerObject(geometry,12);
Map.addLayer(geometry, {}, 'Areas to extract points');

var elevationSamples = elevation.sample({
    region: geometry,
    projection: projection,
    scale: scale,
    geometries: true,
});

Map.addLayer(elevationSamples, {}, 'Points extracted');

// Add three properties to the output table:
// 'Elevation', 'Longitude', and 'Latitude'.
elevationSamples = elevationSamples.map(function(feature) {
    var geom = feature.geometry().coordinates();
    return ee.Feature(null, {
        'Elevation': ee.Number(feature.get(
            'elevation')),
        'Long': ee.Number(geom.get(0)),
        'Lat': ee.Number(geom.get(1))
    });
});

// Export as CSV.
Export.table.toDrive({
    collection: elevationSamples,
    description: 'extracted_points',
    fileFormat: 'CSV'
});
```

We can also extract sample points per elevation zone. Below is an example of extracting 10 randomly selected points per elevation zone (Fig. F5.1.4**)**. You can also set different

values for each zone using `classValues` and `classPoints` parameters to modify the sampling intensity in each class. This may be useful, for instance, to generate point samples for a validation effort.

```
var elevationSamplesStratified = zones.stratifiedSample({
    numPoints: 10,
    classBand: 'zone',
    region: geometry,
    scale: scale,
    projection: projection,
    geometries: true
});

Map.addLayer(elevationSamplesStratified, {}, 'Stratified samples');
```

Fig. F5.1.4 Stratified sampling over different elevation zones

Code Checkpoint F51a. The book's repository contains a script that shows what your code should look like at this point.

Section 1.3. A More Complex Example

In this section we'll use two global datasets, one to represent raster formats and the other vectors:

- The Global Forest Change (GFC) dataset: a raster dataset describing global tree cover and change for 2001–present.
- The World Protected Areas Database: a vector database of global protected areas.

The objective will be to combine these two datasets to quantify rates of deforestation in protected areas in the "arc of deforestation" of the Colombian Amazon. The datasets can be loaded into Earth Engine with the following code:

```
// Read input data.
// Note: these datasets are periodically updated.
// Consider searching the Data Catalog for newer versions.
var gfc = ee.Image('UMD/hansen/global_forest_change_2020_v1_8');
var wdpa = ee.FeatureCollection('WCMC/WDPA/current/polygons');

// Print assets to show available layers and properties.
print(gfc);
print(wdpa.limit(10)); // Show first 10 records.
```

The GFC dataset (first presented in detail in Chap. F1.1) is a global set of rasters that quantify tree cover and change for the period beginning in 2001. We'll use a single image from this dataset:

- `'lossyear'`: a categorical raster of forest loss (1–20, corresponding to deforestation for the period 2001–2020), and 0 for no change

The World Database on Protected Areas (WDPA) is a harmonized dataset of global terrestrial and marine protected area locations, along with details on the classification and management of each. In addition to protected area outlines, we'll use two fields from this database:

- `'NAME'`: the name of each protected area
- `'WDPA_PID'`: a unique numerical ID for each protected area

To begin with, we'll focus on forest change dynamics in 'La Paya', a small protected area in the Colombian Amazon. We'll first visualize these data using the `paint` command, which is discussed in more detail in Chap. F5.3:

```
// Display deforestation.
var deforestation = gfc.select('lossyear');

Map.addLayer(deforestation, {
    min: 1,
    max: 20,
    palette: ['yellow', 'orange', 'red']
}, 'Deforestation raster');

// Display WDPA data.
var protectedArea = wdpa.filter(ee.Filter.equals('NAME', 'La Paya'));

// Display protected area as an outline (see F5.3 for paint()).
var protectedAreaOutline = ee.Image().byte().paint({
    featureCollection: protectedArea,
    color: 1,
    width: 3
});

Map.addLayer(protectedAreaOutline, {
    palette: 'white'
}, 'Protected area');

// Set up map display.
Map.centerObject(protectedArea);
Map.setOptions('SATELLITE');
```

This will display the boundary of the La Paya protected area and deforestation in the region (Fig. F5.1.5).

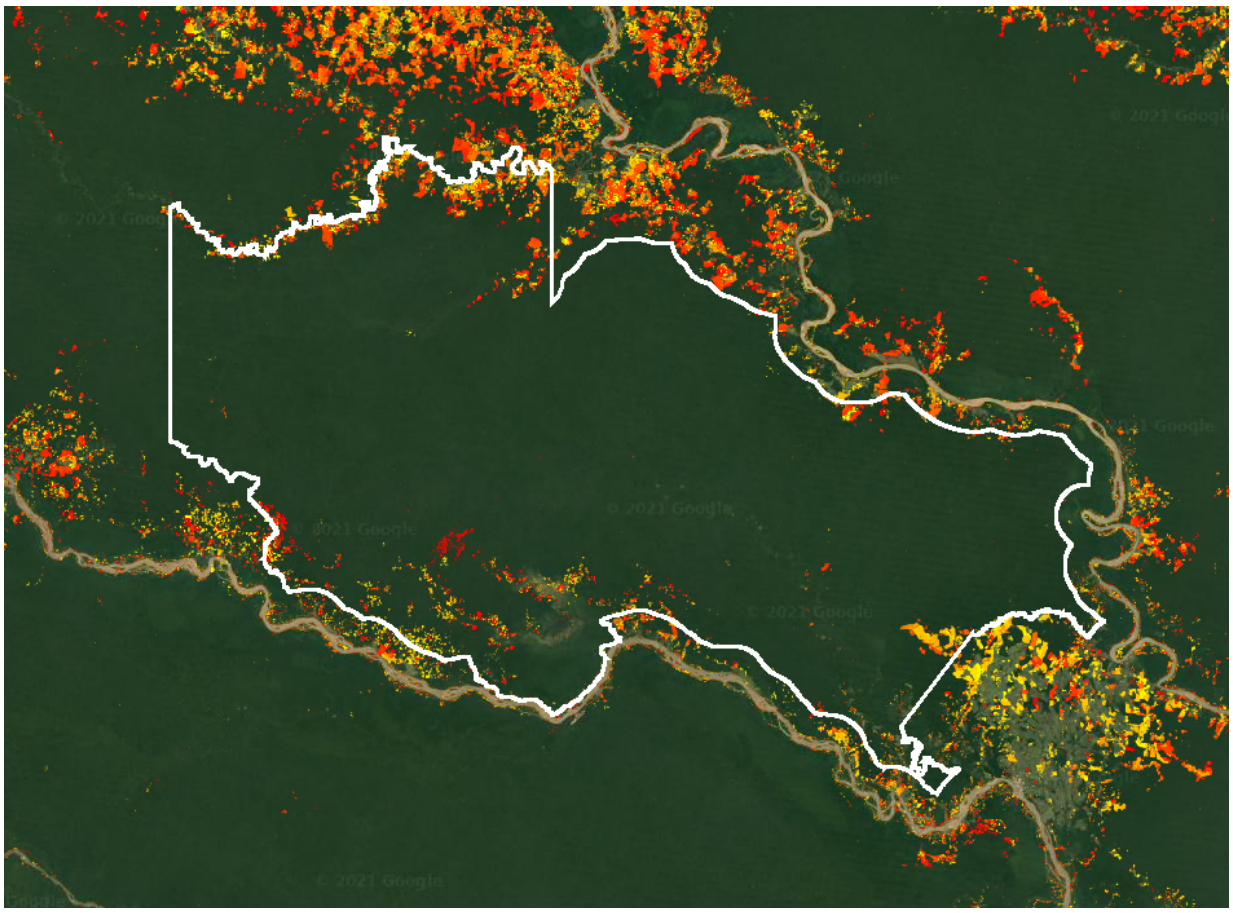


Fig. F5.1.5 View of the La Paya protected area in the Colombian Amazon (in white), and deforestation over the period 2001–2020 (in yellows and reds, with darker colors indicating more recent changes)

We can use Earth Engine to convert the deforestation raster to a set of polygons. The deforestation data are appropriate for this transformation as each deforestation event is labeled categorically by year, and change events are spatially contiguous. This is performed in Earth Engine using the `ee.Image.reduceToVectors` method, as described earlier in this section.

```
// Convert from a deforestation raster to vector.
var deforestationVector = deforestation.reduceToVectors({
    scale: deforestation.projection().nominalScale(),
    geometry: protectedArea.geometry(),
    labelProperty: 'lossyear', // Label polygons with a change year.
    maxPixels: 1e13
});
```

```
// Count the number of individual change events
print('Number of change events:', deforestationVector.size());

// Display deforestation polygons. Color outline by change year.
var deforestationVectorOutline = ee.Image().byte().paint({
    featureCollection: deforestationVector,
    color: 'lossyear',
    width: 1
});

Map.addLayer(deforestationVectorOutline, {
    palette: ['yellow', 'orange', 'red'],
    min: 1,
    max: 20
}, 'Deforestation vector');
```

Fig. F5.1.6 shows a comparison of the raster versus vector representations of deforestation within the protected area.

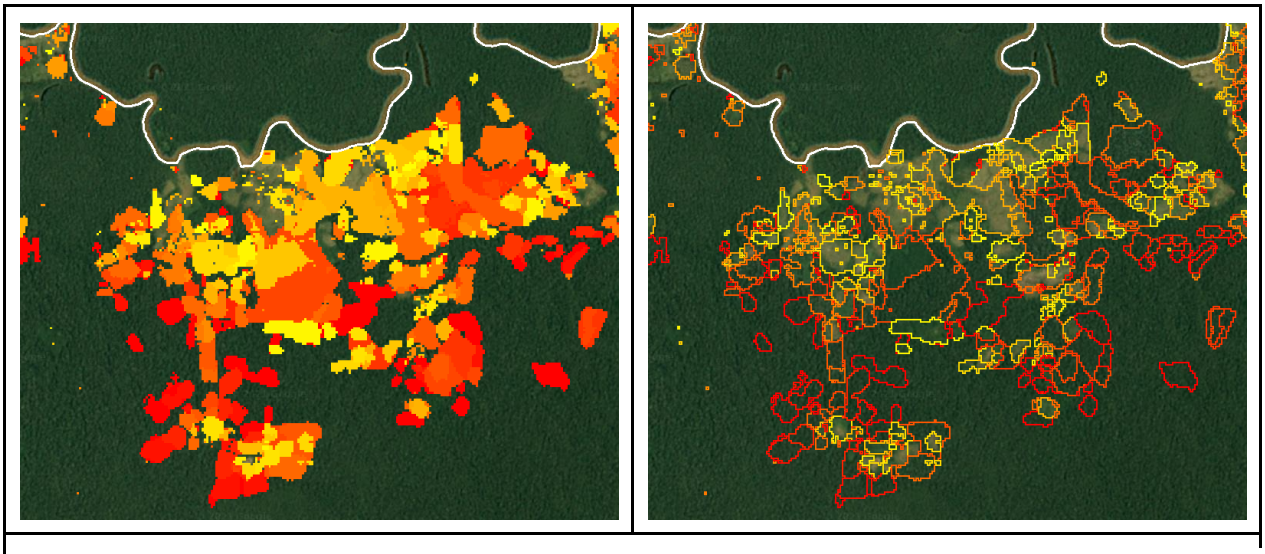


Fig. F5.1.6 Raster (left) versus vector (right) representations of deforestation data of the La Paya protected area

Having converted from raster to vector, a new set of operations becomes available for post-processing the deforestation data. We might, for instance, be interested in the number of individual change events each year (Fig. F5.1.7):

```
var chart = ui.Chart.feature
    .histogram({
        features: deforestationVector,
        property: 'lossyear'
    })
    .setOptions({
        hAxis: {
            title: 'Year'
        },
        vAxis: {
            title: 'Number of deforestation events'
        },
        legend: {
            position: 'none'
        }
    });

print(chart);
```

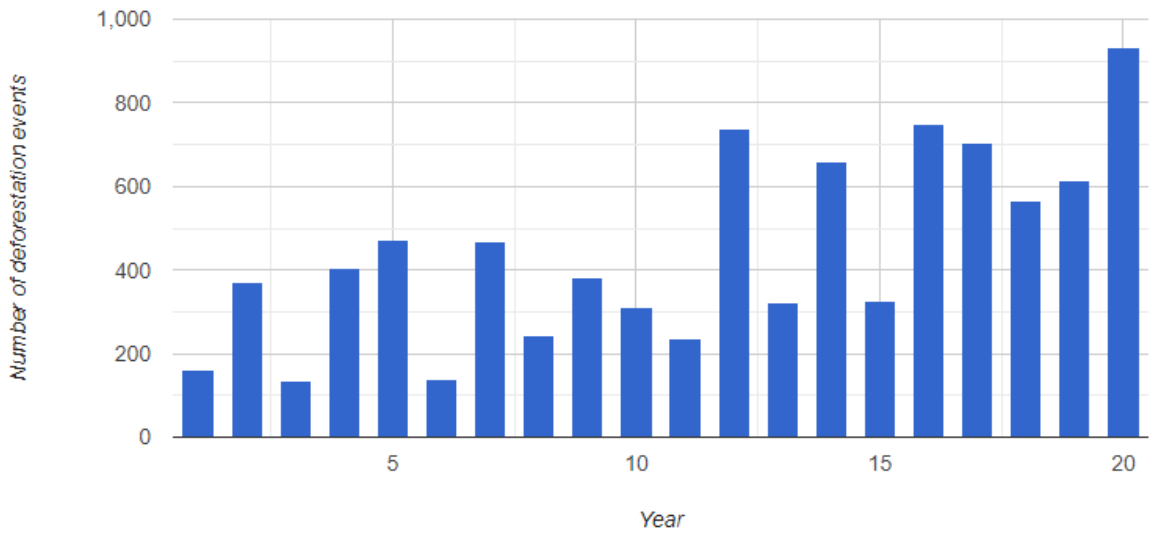


Fig. F5.1.7 Plot of the number of deforestation events in La Paya for the years 2001–2020

There might also be interest in generating point locations for individual change events (e.g., to aid a field campaign):

```
// Generate deforestation point locations.
```

```
var deforestationCentroids = deforestationVector.map(function(feat) {
    return feat.centroid();
});

Map.addLayer(deforestationCentroids, {
    color: 'darkblue'
}, 'Deforestation centroids');
```

The vector format allows for easy filtering to only deforestation events of interest, such as only the largest deforestation events:

```
// Add a new property to the deforestation FeatureCollection
// describing the area of the change polygon.
deforestationVector = deforestationVector.map(function(feat) {
    return feat.set('area', feat.geometry().area({
        maxError: 10
    }).divide(10000)); // Convert m^2 to hectare.
});

// Filter the deforestation FeatureCollection for only large-scale
(>10 ha) changes
var deforestationLarge = deforestationVector.filter(ee.Filter.gt(
    'area', 10));

// Display deforestation area outline by year.
var deforestationLargeOutline = ee.Image().byte().paint({
    featureCollection: deforestationLarge,
    color: 'lossyear',
    width: 1
});

Map.addLayer(deforestationLargeOutline, {
    palette: ['yellow', 'orange', 'red'],
    min: 1,
    max: 20
}, 'Deforestation (>10 ha)');
```

Code Checkpoint F51b. The book's repository contains a script that shows what your code should look like at this point.

Section 1.4: Raster Properties to Vector Fields

Sometimes we want to extract information from a raster to be included in an existing vector dataset. An example might be estimating a deforestation rate for a set of protected areas. Rather than perform this task on a case-by-case basis, we can attach information generated from an image as a property of a feature.

The following script shows how this can be used to quantify a deforestation rate for a set of protected areas in the Colombian Amazon.

```
// Load required datasets.
var gfc = ee.Image('UMD/hansen/global_forest_change_2020_v1_8');
var wdpa = ee.FeatureCollection('WCMC/WDPA/current/polygons');

// Display deforestation.
var deforestation = gfc.select('lossyear');

Map.addLayer(deforestation, {
    min: 1,
    max: 20,
    palette: ['yellow', 'orange', 'red']
}, 'Deforestation raster');

// Select protected areas in the Colombian Amazon.
var amazonianProtectedAreas = [
    'Cordillera de los Picachos', 'La Paya', 'Nukak',
    'Serrania de Chiribiquete',
    'Sierra de la Macarena', 'Tinigua'
];

var wdpaSubset = wdpa.filter(ee.Filter.inList('NAME',
    amazonianProtectedAreas));

// Display protected areas as an outline.
var protectedAreasOutline = ee.Image().byte().paint({
    featureCollection: wdpaSubset,
    color: 1,
    width: 1
});
```

```
Map.addLayer(protectedAreasOutline, {
    palette: 'white'
}, 'Amazonian protected areas');

// Set up map display.
Map.centerObject(wdpaSubset);
Map.setOptions('SATELLITE');

var scale = deforestation.projection().nominalScale();

// Use 'reduceRegions' to sum together pixel areas in each protected
area.
wdpaSubset = deforestation.gte(1)
    .multiply(ee.Image.pixelArea().divide(10000)).reduceRegions({
        collection: wdpaSubset,
        reducer: ee.Reducer.sum().setOutputs([
            'deforestation_area']),
        scale: scale
    });

print(wdpaSubset); // Note the new 'deforestation_area' property.
```

The output of this script is an estimate of deforested area in hectares for each reserve. However, as reserve sizes vary substantially by area, we can normalize by the total area of each reserve to quantify rates of change.

```
// Normalize by area.
wdpaSubset = wdpaSubset.map(
    function(feat) {
        return feat.set('deforestation_rate',
            ee.Number(feat.get('deforestation_area'))
            .divide(feat.area().divide(10000)) // m2 to ha
            .divide(20) // number of years
            .multiply(100)); // to percentage points
    });

// Print to identify rates of change per protected area.
// Which has the fastest rate of loss?
```

```
print(wdpaSubset.reduceColumns({
    reducer: ee.Reducer.toList().repeat(2),
    selectors: ['NAME', 'deforestation_rate']
}));
```

Code Checkpoint F51c. The book's repository contains a script that shows what your code should look like at this point.

Section 2. Vector-to-Raster Conversion

In Sect. 1, we used the protected area feature collection as its original vector format. In this section, we will rasterize the protected area polygons to produce a mask and use this to assess rates of forest change.

Section 2.1. Polygons to a Mask

The most common operation to convert from vector to raster is the production of binary image masks, describing whether a pixel intersects a line or falls within a polygon. To convert from vector to a raster mask, we can use the `ee.FeatureCollection.reduceToImage` method. Let's continue with our example of the WDPA database and Global Forest Change data from the previous section:

```
// Load required datasets.
var gfc = ee.Image('UMD/hansen/global_forest_change_2020_v1_8');
var wdpa = ee.FeatureCollection('WCMC/WDPA/current/polygons');

// Get deforestation.
var deforestation = gfc.select('lossyear');

// Generate a new property called 'protected' to apply to the output
mask.
var wdpa = wdpa.map(function(feat) {
    return feat.set('protected', 1);
});

// Rasterize using the new property.
// unmask() sets areas outside protected area polygons to 0.
var wdpaMask = wdpa.reduceToImage(['protected'], ee.Reducer.first())
    .unmask();

// Center on Colombia.
```

```
Map.setCenter(-75, 3, 6);

// Display on map.
Map.addLayer(wdpaMask, {
    min: 0,
    max: 1
}, 'Protected areas (mask)');
```

We can use this mask to, for example, highlight only deforestation that occurs within a protected area using logical operations:

```
// Set the deforestation layer to 0 where outside a protected area.
var deforestationProtected = deforestation.where(wdpaMask.eq(0), 0);

// Update mask to hide where deforestation layer = 0
var deforestationProtected = deforestationProtected
    .updateMask(deforestationProtected.gt(0));

// Display deforestation in protected areas
Map.addLayer(deforestationProtected, {
    min: 1,
    max: 20,
    palette: ['yellow', 'orange', 'red']
}, 'Deforestation protected');
```

In the above example we generated a simple binary mask, but `reduceToImage` can also preserve a numerical property of the input polygons. For example, we might want to be able to determine which protected area each pixel represents. In this case, we can produce an image with the unique ID of each protected area:

```
// Produce an image with unique ID of protected areas.
var wdpaId = wdpa.reduceToImage(['WDPAID'], ee.Reducer.first());

Map.addLayer(wdpaId, {
    min: 1,
    max: 100000
}, 'Protected area ID');
```

This output can be useful when performing large-scale raster operations, such as efficiently calculating deforestation rates for multiple protected areas.

Code Checkpoint F51d. The book's repository contains a script that shows what your code should look like at this point.

Section 2.2. A More Complex Example

The `reduceToImage` method is not the only way to convert a feature collection to an image. We will create a distance image layer from the boundary of the protected area using `distance`. For this example, we return to the La Paya protected area explored in Sect. 1.

```
// Load required datasets.
var gfc = ee.Image('UMD/hansen/global_forest_change_2020_v1_8');
var wdpa = ee.FeatureCollection('WCMC/WDPA/current/polygons');

// Select a single protected area.
var protectedArea = wdpa.filter(ee.Filter.equals('NAME', 'La Paya'));

// Maximum distance in meters is set in the brackets.
var distance = protectedArea.distance(1000000);

Map.addLayer(distance, {
    min: 0,
    max: 20000,
    palette: ['white', 'grey', 'black'],
    opacity: 0.6
}, 'Distance');

Map.centerObject(protectedArea);
```

We can also show the distance inside and outside of the boundary by using the rasterized protected area (Fig. F5.1.8).

```
// Produce a raster of inside/outside the protected area.
var protectedAreaRaster = protectedArea.map(function(feat) {
    return feat.set('protected', 1);
}).reduceToImage(['protected'], ee.Reducer.first());
```

```
Map.addLayer(distance.updateMask(protectedAreaRaster), {
    min: 0,
    max: 20000
}, 'Distance inside protected area');

Map.addLayer(distance.updateMask(protectedAreaRaster.unmask()
.not()), {
    min: 0,
    max: 20000
}, 'Distance outside protected area');
```

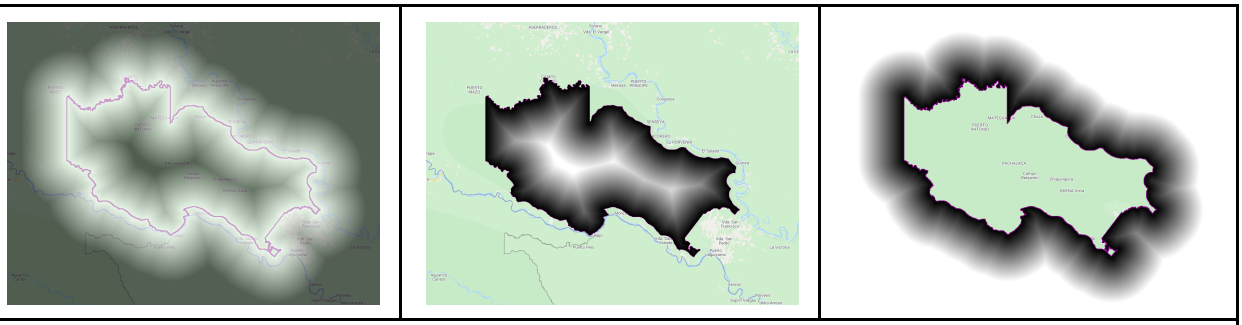


Fig. F5.1.8 Distance from the La Paya boundary (left), distance within the La Paya (middle), and distance outside the La Paya (right)

Sometimes it makes sense to work with objects in raster imagery. This is an unusual case of vector-like operations conducted with raster data. There is a good reason for this where the vector equivalent would be computationally burdensome.

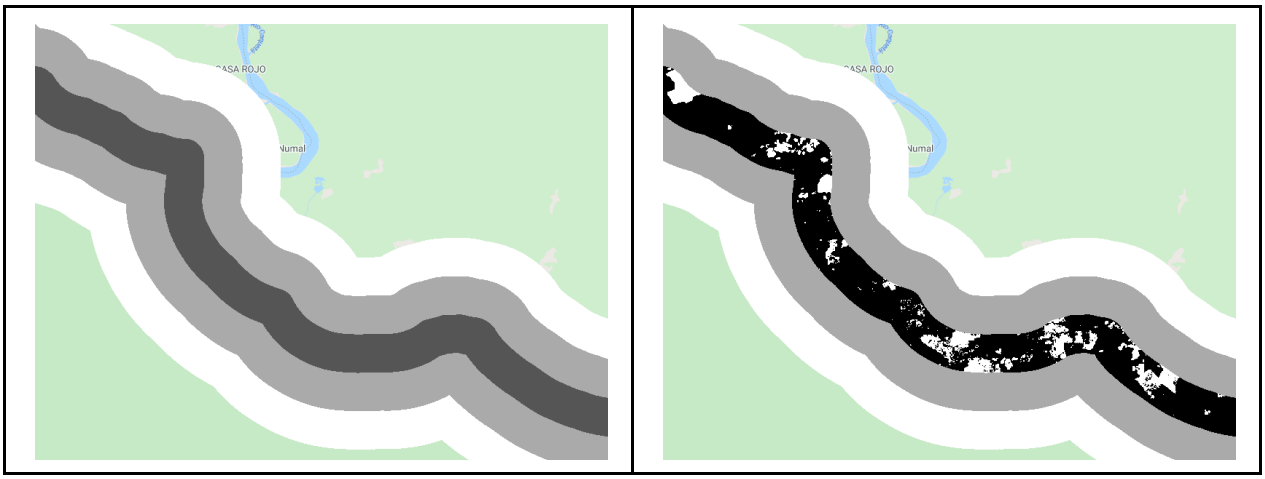
An example of this is estimating deforestation rates by distance to the edge of the protected area, as it is common that rates of change will be higher at the boundary of a protected area. We will create a distance raster with three zones from the La Paya boundary (>1 km, >2 km, >3 km, and >4 km) and to estimate the deforestation by distance from the boundary (Fig. F5.1.9).

```
var distanceZones = ee.Image(0)
    .where(distance.gt(0), 1)
    .where(distance.gt(1000), 2)
    .where(distance.gt(3000), 3)
    .updateMask(distance.lte(5000));
```

```
Map.addLayer(distanceZones, {}, 'Distance zones');

var deforestation = gfc.select('loss');
var deforestation1km = deforestation.updateMask(distanceZones.eq(1));
var deforestation3km = deforestation.updateMask(distanceZones.lte(2));
var deforestation5km = deforestation.updateMask(distanceZones.lte(3));

Map.addLayer(deforestation1km, {
    min: 0,
    max: 1
}, 'Deforestation within a 1km buffer');
Map.addLayer(deforestation3km, {
    min: 0,
    max: 1,
    opacity: 0.5
}, 'Deforestation within a 3km buffer');
Map.addLayer(deforestation5km, {
    min: 0,
    max: 1,
    opacity: 0.5
}, 'Deforestation within a 5km buffer');
```



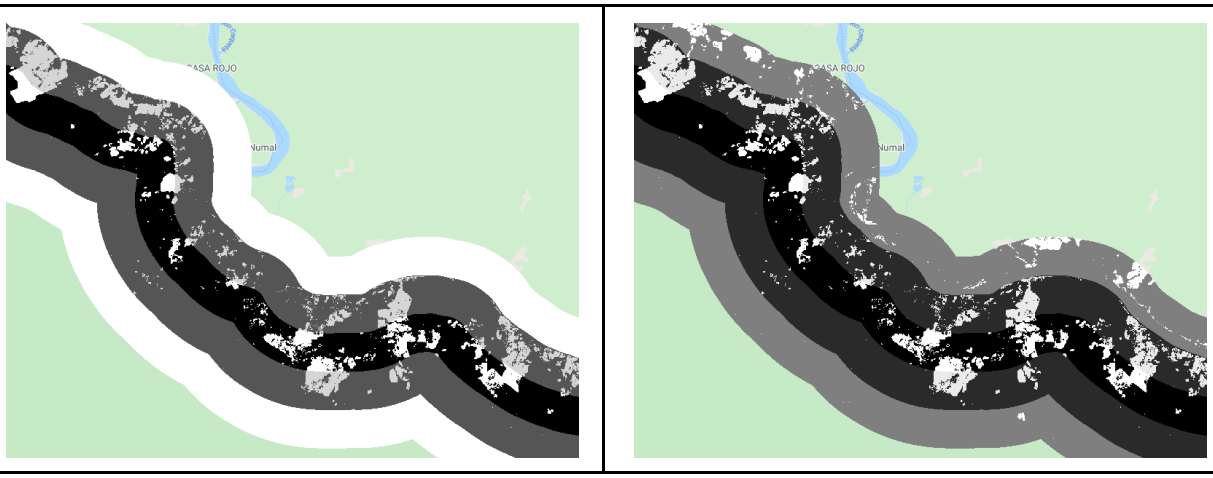


Fig. F5.1.9 Distance zones (top left) and deforestation by zone (<1 km, <3 km, and <5 km)

Lastly, we can estimate the deforestation area within 1 km of the protected area but only outside of the boundary.

```
var deforestation1kmOutside = deforestation1km
    .updateMask(protectedAreaRaster.unmask().not());

// Get the value of each pixel in square meters
// and divide by 10000 to convert to hectares.
var deforestation1kmOutsideArea = deforestation1kmOutside.eq(1)
    .multiply(ee.Image.pixelArea()).divide(10000);

// We need to set a larger geometry than the protected area
// for the geometry parameter in reduceRegion().
var deforestationEstimate = deforestation1kmOutsideArea
    .reduceRegion({
        reducer: ee.Reducer.sum(),
        geometry: protectedArea.geometry().buffer(1000),
        scale: deforestation.projection().nominalScale()
    });

print('Deforestation within a 1km buffer outside the protected area
(ha)',
    deforestationEstimate);
```

Code Checkpoint F51e. The book's repository contains a script that shows what your code should look like at this point.

Synthesis

Question 1. In this lab, we quantified rates of deforestation in La Paya. There is another protected area in the Colombian Amazon named Tinigua. By modifying the existing scripts, determine how the dynamics of forest change in Tinigua compare to those in La Paya with respect to:

- the number of deforestation events
- the year with the greatest number of change events
- the mean average area of change events
- the total area of loss

Question 2. In Sect. 1.4, we only considered losses of tree cover, but many protected areas will also have increases in tree cover from regrowth (which is typical of shifting agriculture). Calculate growth in hectares using the Global Forest Change dataset's gain layer for the six protected areas in Sect. 1.4 by extracting the raster properties and adding them to vector fields. Which has the greatest area of regrowth? Is this likely to be sufficient to balance out the rates of forest loss? Note: The gain layer shows locations where tree cover has increased for the period 2001–2012 (0 = no gain, 1 = tree cover increase), so for comparability use deforestation between the same time period of 2001–2012.

Question 3. In Sect. 2.2, we considered rates of deforestation in a buffer zone around La Paya. Estimate the deforestation rates inside of La Paya using buffer zones. Is forest loss more common close to the boundary of the reserve?

Question 4. Sometimes it's advantageous to perform processing using raster operations, particularly at large scales. It is possible to perform many of the tasks in Sect. 1.3 and 1.4 by first converting the protected area vector to raster, and then using only raster operations. As an example, can you display only deforestation events >10 ha in La Paya using only raster data? (Hint: Consider using `ee.Image.connectedPixelCount`. You may also want to also look at Sect. 2.1).

Conclusion

In this chapter, you learned how to convert raster to vector and vice versa. More importantly, you now have a better understanding of why and when such conversions are useful. Our examples should give you practical applications and ideas for using these techniques.

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.