

Google Earth and E... · [Follow publication](#)

# Visualizing changing landscapes with Google Earth Engine

5 min read · May 28, 2020



Google Earth

Follow

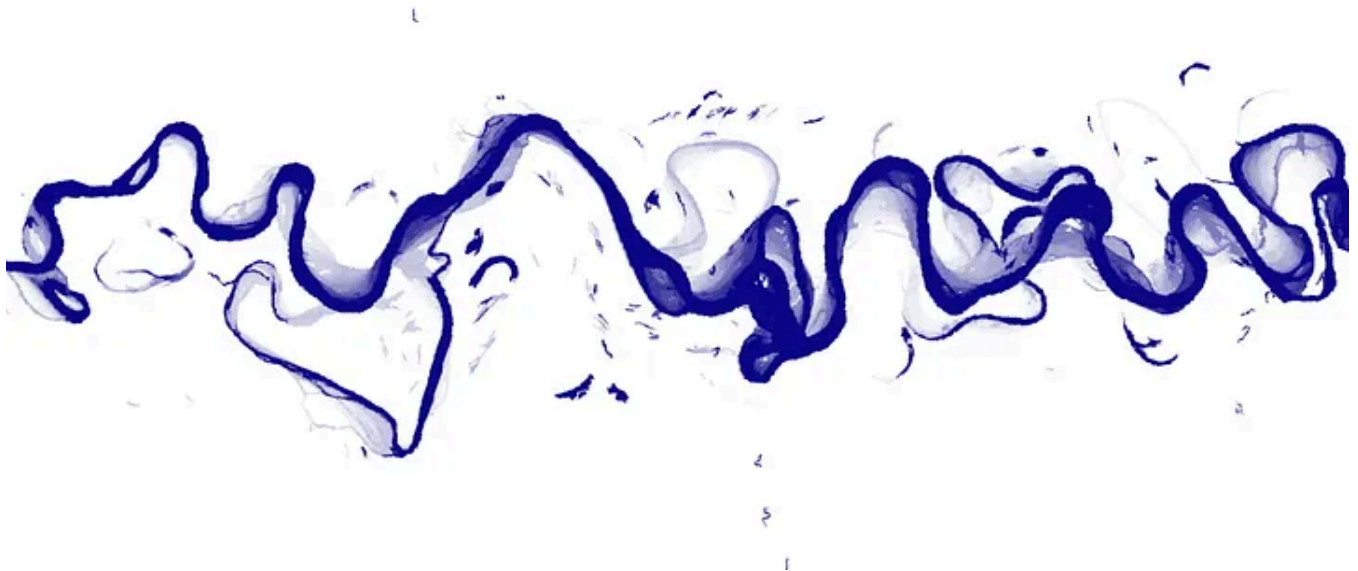


Listen



Share

By *Justin Braaten*, Technical Writer, Earth Engine

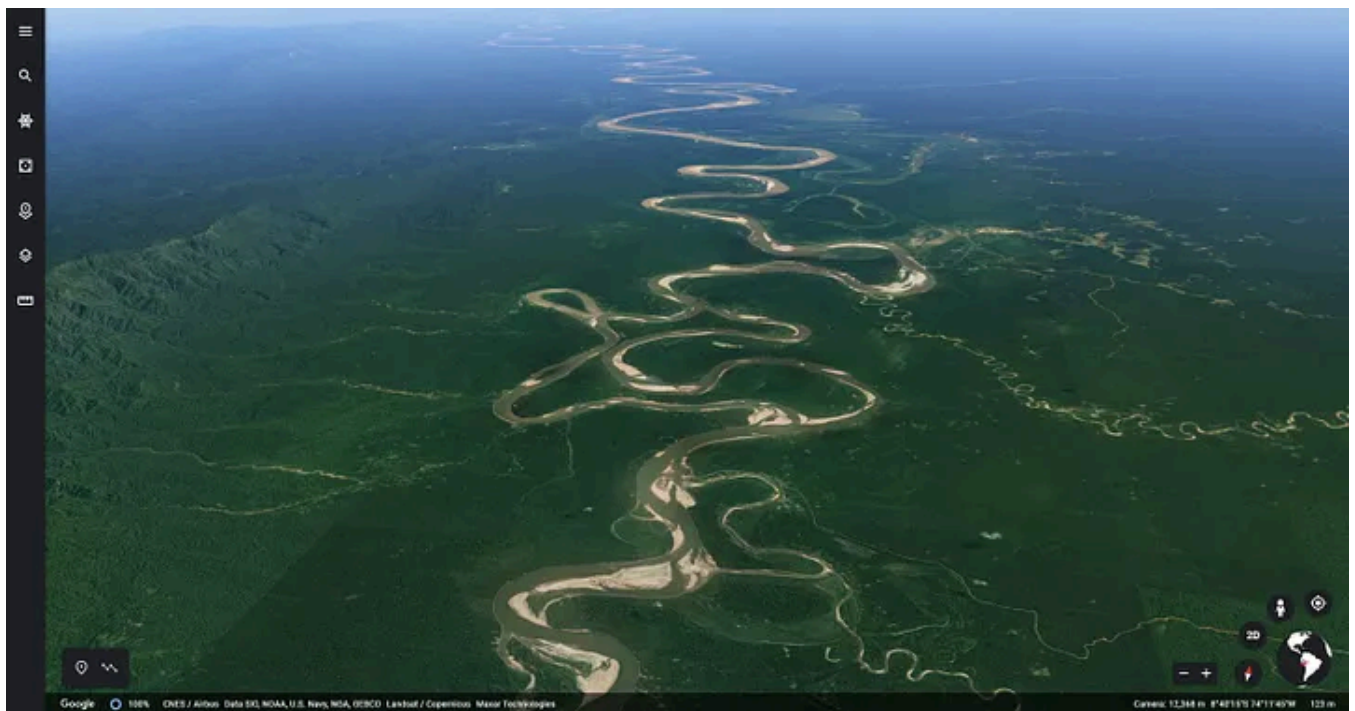


Earth's landscapes are constantly changing. Ephemeral seasonal cycles blanket landscapes in snow, paint hillsides with colorful wildflowers, and dress trees in green. Longer-lasting change is happening too. Forest harvesting and wildfires alter vegetation composition, urban development paves over nature, and erosion sculpts the land. Earth-observing satellites have been a witness to these changes. Now

Google Earth Engine is making it easy to convert the image data into time lapse animations that capture the dynamism of nature and humans.

In this follow-up to a previous post introducing [basic animation techniques in Earth Engine](#), I'll use a section of Peru's wildly dynamic [Ucayali River](#) to demonstrate more advanced animation techniques that emphasize change by imprinting historical context into each frame of the animation using fading and cumulative history effects. A river animation app is provided, so if coding is not your thing, feel free to skip to the end.

Note that while this tutorial uses river dynamics to demonstrate fading and cumulative history animation effects, these techniques are applicable to any thematic map class that changes over space and time. For instance, the same methods can be used to relate forest clearcut history, air pollutant transport, wildfire history, or glacier retreat, to name a few.



Peru's meandering Ucayali River seen from [Google Earth](#).

## Basic animation

First things first — we'll need some data. Taking a look at the [Earth Engine Data Catalog](#), you'll find the [JRC Yearly Water Classification History](#) dataset. These data are global maps of surface water for each year from 1984 through 2018 with 30-meter resolution — just what we need!

## Get Google Earth's stories in your inbox

Join Medium for free to get updates from this writer.

Enter your email

Subscribe

Fire up the [Code Editor](#) and import the dataset. [Map a function](#) over the image collection to set the values representing water pixels in each annual layer to its respective year. These values will be used later to stretch a color palette over the data. Reduce speckle from small ephemeral water bodies in the data by masking out connected pixel groups with less than 15 members. Also, set all non-water pixels as transparent so we can control river channel opacity.

```
var col =  
ee.ImageCollection('JRC/GSW1_1/YearlyHistory').map(function(img) {  
  var year = img.date().get('year');  
  var yearImg = img.gte(2).multiply(year);  
  var despeckle = yearImg.connectedPixelCount(15, true).eq(15);  
  return yearImg.updateMask(despeckle).selfMask().set('year', year);  
});
```

Each image in the collection represents a frame in the animation. Since we're animating flow, let's define a function that will reverse the frame sequence once the end is reached to produce a smooth, flowy transition back to the beginning. This is accomplished by making a copy of the image collection in reverse chronological order and appending it to the original collection.

```
function appendReverse(col) {  
  return col.merge(col.sort('year', false));  
}
```

Next, color water pixels blue and land pixels white to make the subject of the animation more self-evident. Map over the image collection to define visualization properties for each image. Append the reversed frame set to the forward set.

```
var bgColor = 'FFFFFF'; // Assign white to background pixels.
var riverColor = '0D0887'; // Assign blue to river pixels.

var annualCol = col.map(function(img) {
  return img.unmask(0)
    .visualize({min: 0, max: 1, palette: [bgColor, riverColor]})
    .set('year', img.get('year'));
});

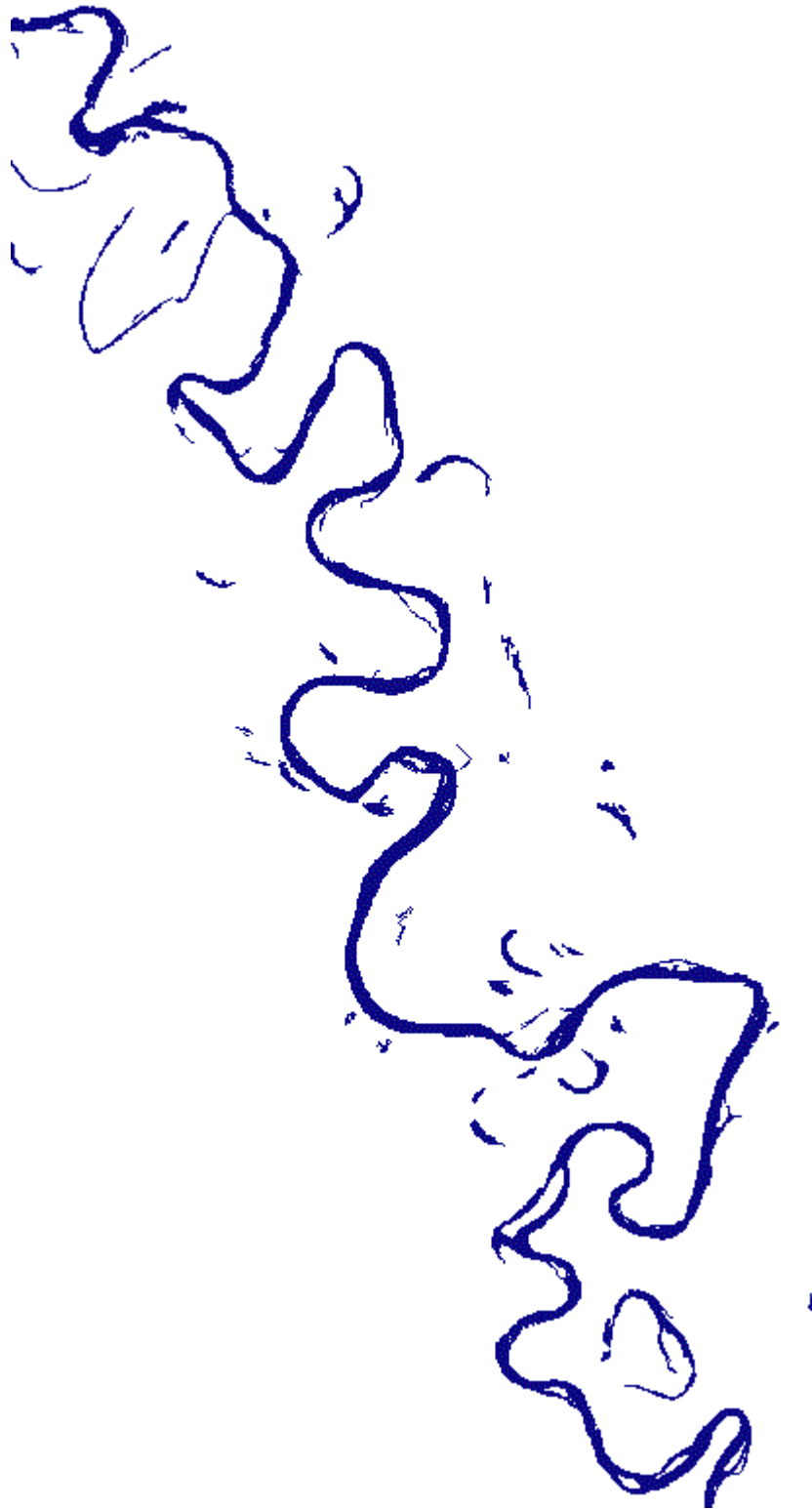
var basicAnimation = appendReverse(annualCol);
```

Finally, define a section of stream to animate, set animation arguments, and render it.

```
var aoi = ee.Geometry.Rectangle(-74.327, -10.087, -73.931, -9.327);

var videoArgs = {
  dimensions:
    600, // Max dimension (pixels), min dimension is proportionally
    scaled.
  region: aoi,
  framesPerSecond: 10
};

print(ui.Thumbnail(basicAnimation, videoArgs));
```



An animation of JRC's Global Surface Water dataset depicting 32 years of channel dynamics for Peru's Ucayali River.

As you can see, this particular section of the river is quite unbounded! The animation looks okay, but it has some unpolished qualities and does not convey channel history very well. Let's add a fading memory of river channels past to smooth out the animation and convey a better sense of movement by providing historical reference.

## Fading history animation

Currently, only a single year's channel is shown per frame. In this next animation, we'll use opacity and compositing to retain and progressively fade all prior years per frame. To do this, add a nearly transparent fade filter overlay to each annual river image, and then construct a collection where each image represents the temporal composite of all previous fade-filtered images overlaid by an unfiltered image from the given year. As more images are included in the composite, the fade filters accumulate making the earliest pixels more opaque, eventually resolving to the color of the fade filter.

```
var bgImg = ee.Image(1).visualize({palette: bgColor});
var fadeFilter = ee.Image(1).visualize({palette: bgColor, opacity:
0.1});

var fadeFilterCol = col.map(function(img) {
  var imgVis = img.visualize({palette: riverColor});
  return imgVis.blend(fadeFilter).set('year', img.get('year'));
});

var yearSeq = ee.List.sequence(1984, 2018);

var fadeCol =
ee.ImageCollection.fromImages(yearSeq.map(function(year) {
  var fadeComp =
  fadeFilterCol.filter(ee.Filter.lte('year',
year)).sort('year').mosaic();
  var thisYearImg = col.filter(ee.Filter.eq('year',
year)).first().visualize({
  palette: riverColor
  });
  return bgImg.blend(fadeComp).blend(thisYearImg).set('year', year);
}));

print(ui.Thumbnail(appendReverse(fadeCol), videoArgs));
```

[Open in app](#) ↗[Sign up](#)[Sign in](#)**Medium**

Search



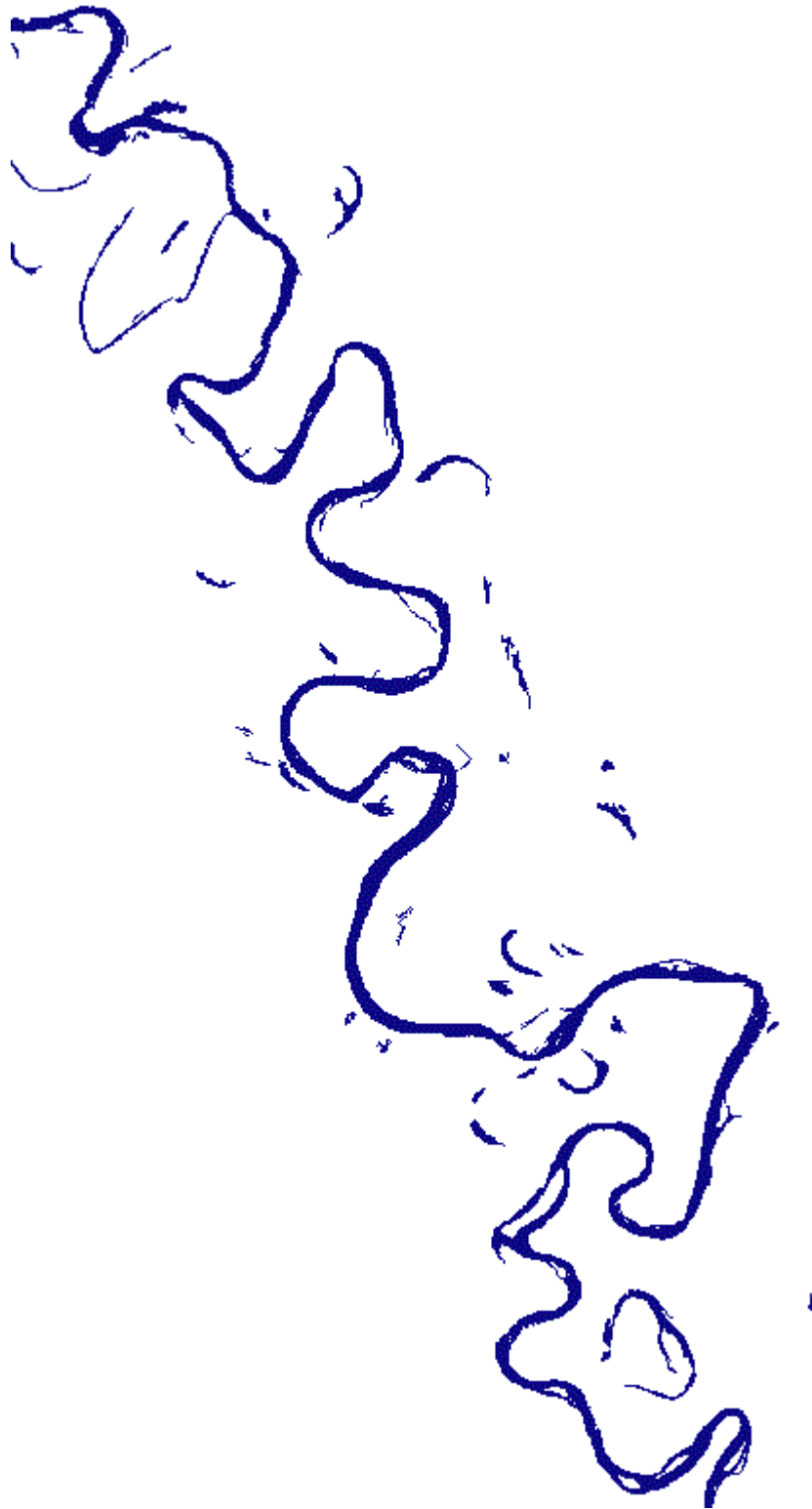
provides historical context to more easily comprehend dynamics.

### **Cumulative history animation**

We can alter the previous animation slightly to show the full river channel history by removing the fade filter image overlay and assigning a distinct color to each

year's river channel. The process is the same as the previous script, except that we don't blend the fade filter image, and instead of assigning the color blue to water pixels, they are colored according to year by stretching a palette across the range of years.

```
var cumulativeVis = {  
  min: 1984,  
  max: 2018,  
  palette: ['0D0887', '5B02A3', '9A179B', 'CB4678', 'ED1E79']  
};  
  
var cumulativeFilterCol = col.map(function(img) {  
  return img.visualize(cumulativeVis).set('year', img.get('year'));  
});  
  
var cumulativeCol =  
ee.ImageCollection.fromImages(yearSeq.map(function(year) {  
  var cumulativeComp =  
cumulativeFilterCol.filter(ee.Filter.lte('year', year))  
  .sort('year')  
  .mosaic();  
  return bgImg.blend(cumulativeComp).set('year', year);  
})));  
  
print(ui.Thumbnail(appendReverse(cumulativeCol), videoArgs));
```

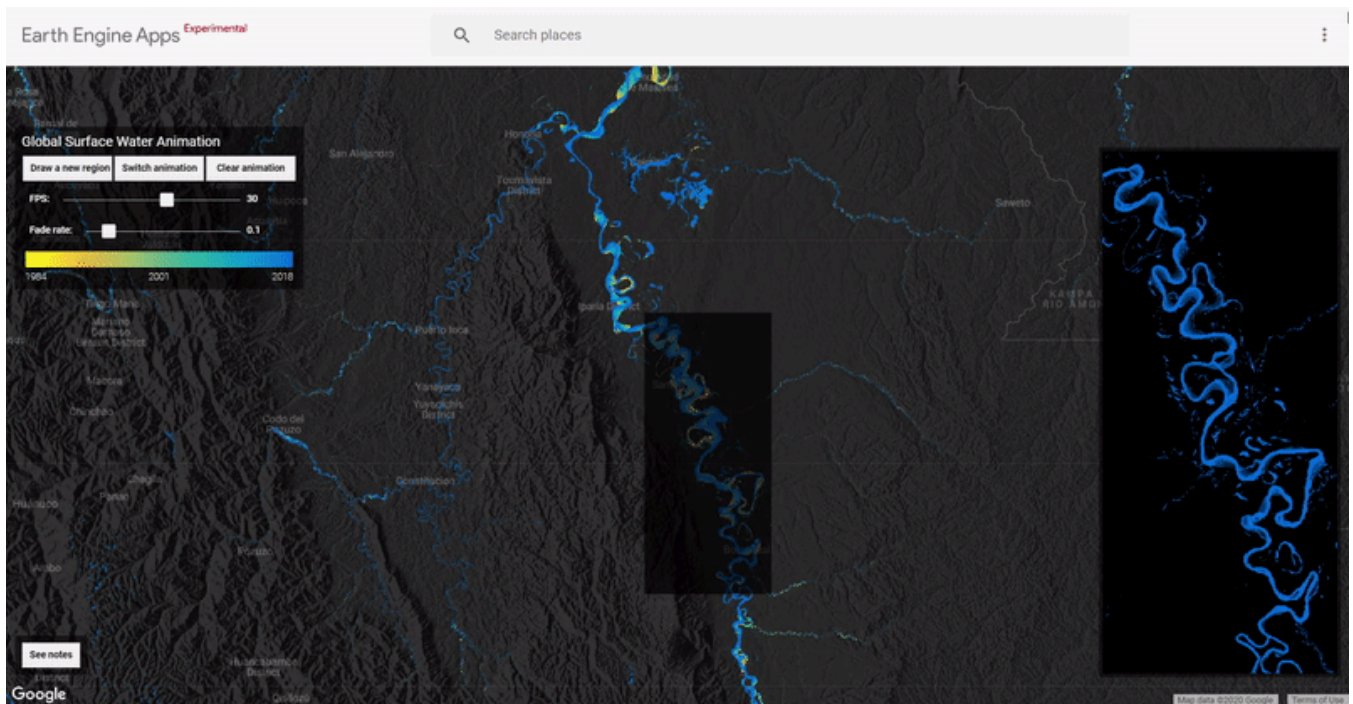


Animation of Peru's Ucayali River demonstrating a cumulative history technique to smooth the animation and provide historical context.

There you have it: 32 years of river channel history in a few seconds! I hope you've enjoyed this code walkthrough demonstrating image overlay and compositing techniques to customize time series animations with Earth Engine. If you're interested in learning more, please see the Earth Engine [ImageCollection Visualization Guide](#).

## Surface water history app

I was intrigued by how active the Ucayali River is and was curious to know how many other rivers meander so freely, so I built an [Earth Engine App](#) to make exploring easier. It features a thematic map that displays the most recent channel path year and allows you to draw a rectangle over a region to generate an animation anywhere on Earth. Customize the frame and history fade rate before you download the animated GIF image.



Interactive JRC Surface Water animation app built using Google Earth Engine Apps.

Post your animations to Twitter and let us know what you discover by tagging:  
@googleearth #RiverGIF

## Earth Engine

## Data Visualization

## Landsat

Jrc

## Earth Observation



Follow

**Published in Google Earth and Earth Engine**

82K followers · Last published Oct 1, 2025

For geospatial professionals, scientists, developers and storytellers

[Follow](#)

## Written by Google Earth

10.4K followers · 554 following

Put the globe to work

### Responses (4)



Write a response

What are your thoughts?



Emailmaleki

Aug 22, 2021



Excellent!

how can we insert the year of each frame in animation?



[Reply](#)



Meenumeena

Apr 9, 2021



Can I get its full code please?



[Reply](#)



Asanka Liyadpita

Jul 13, 2020



Hi..... it's very interesting and awesome.

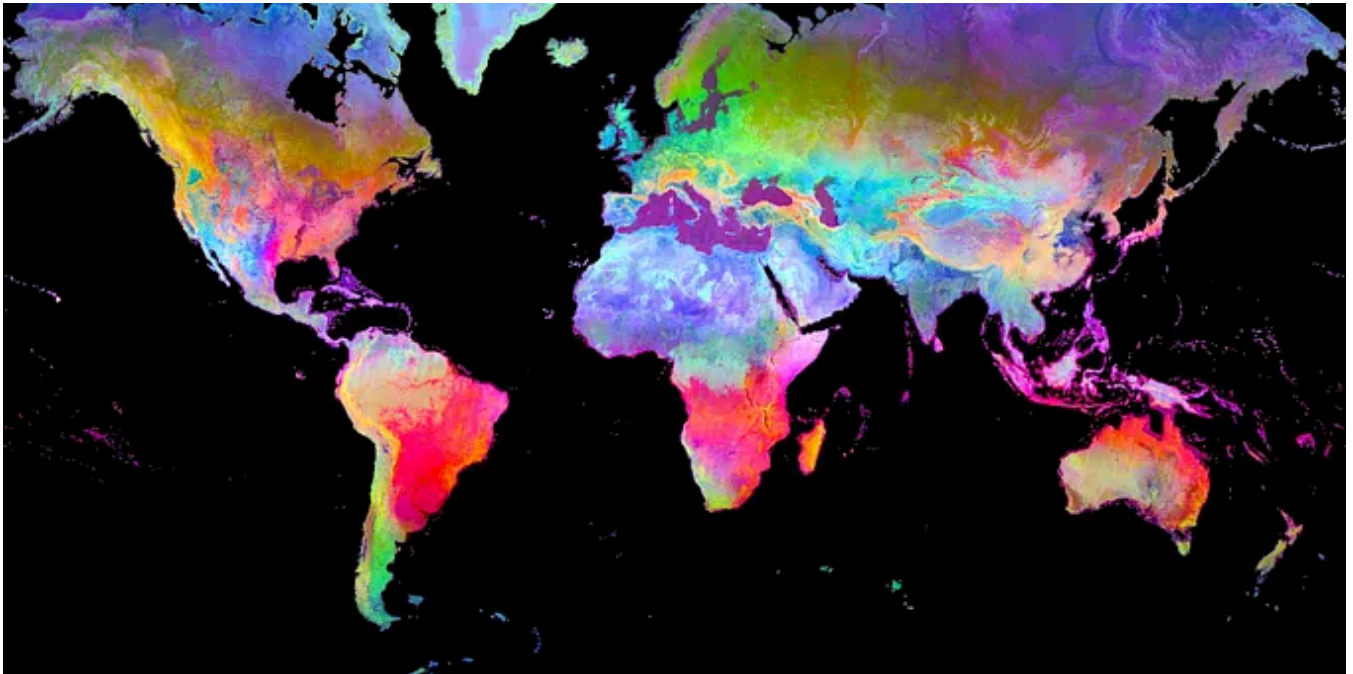
how can I contact this technical writer



[Reply](#)

[See all responses](#)

## More from Google Earth and Google Earth and Earth Engine



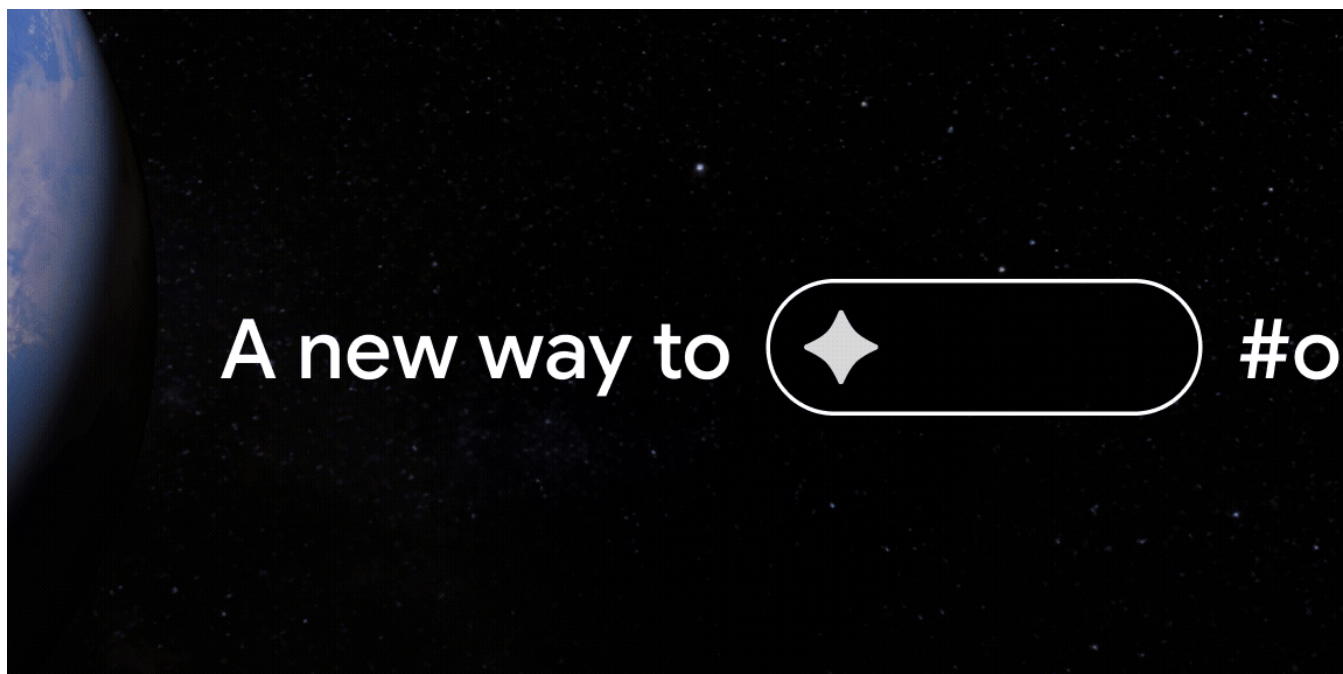
In Google Earth and Earth Engine by Google Earth

### **AI-powered pixels: Introducing Google's Satellite Embedding dataset**

By Valerie Pasquarella, Research Scientist and Emily Schechter, Product Manager, Google

Jul 30 571 13



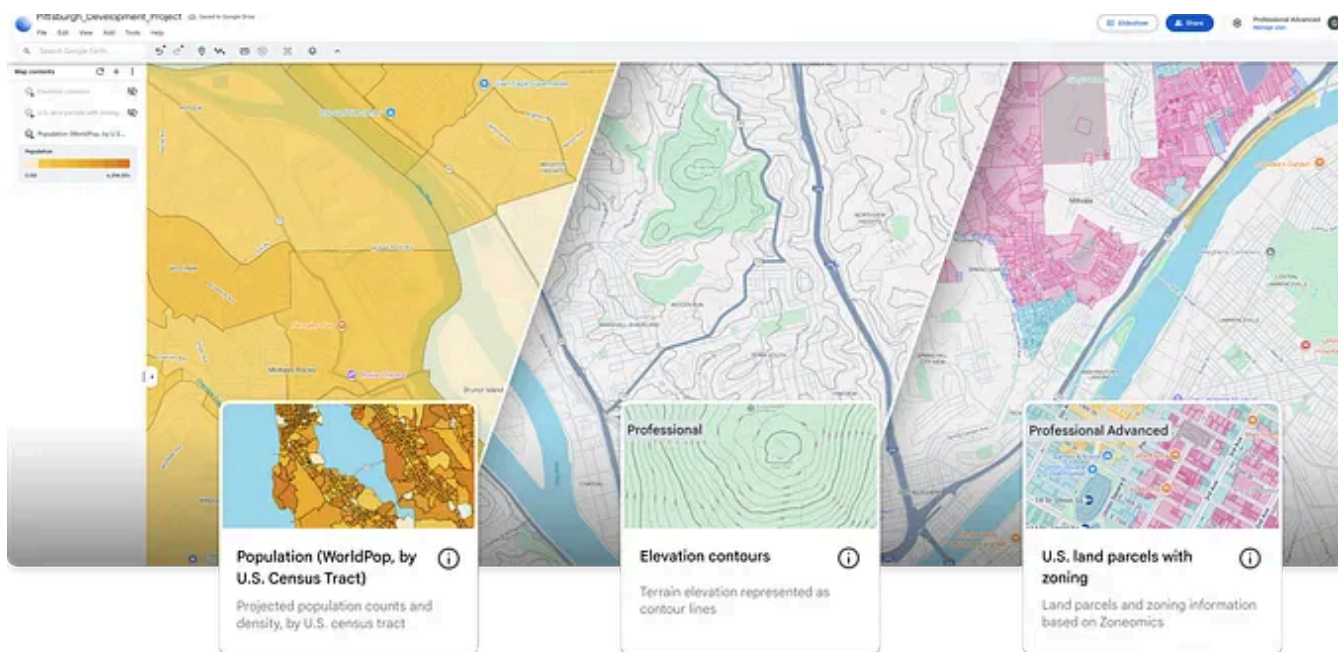


 In Google Earth and Earth Engine by Google Earth

## A whole new way to work #onEarth: Introducing Google Earth's new professional plans

By Brian Ho & Patrik Blohmé, Google Earth Product Managers

Oct 1  110  1

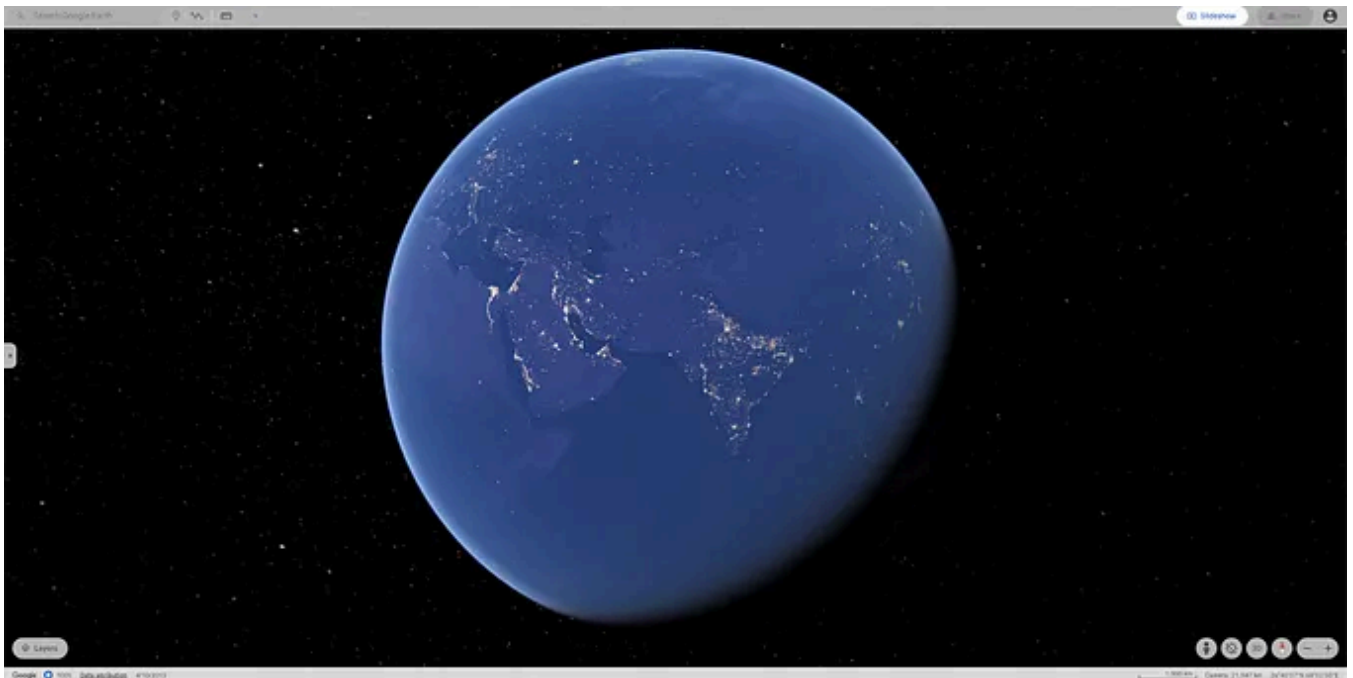


 In Google Earth and Earth Engine by Google Earth

## New in Google Earth: Advanced Data Layers to Power Your Professional Projects #onEarth

By Alaina Adams and Jenica Rawtani, Google Earth Product Managers

Sep 16 🖱️ 64



In Google Earth and Earth Engine by Google Earth

## Tile Overlays in Google Earth: The Missing Manual

By Brian Ellis, Software Engineer, Google Earth

Jun 5, 2024 🖱️ 61 💬 4

[See all from Google Earth](#)[See all from Google Earth and Earth Engine](#)

## Recommended from Medium



Codastra

## BigQuery GIS, Big Maps

How to run spatial analytics over billions of rows—without leaving your data warehouse.

★ 6d ago 🖱 23

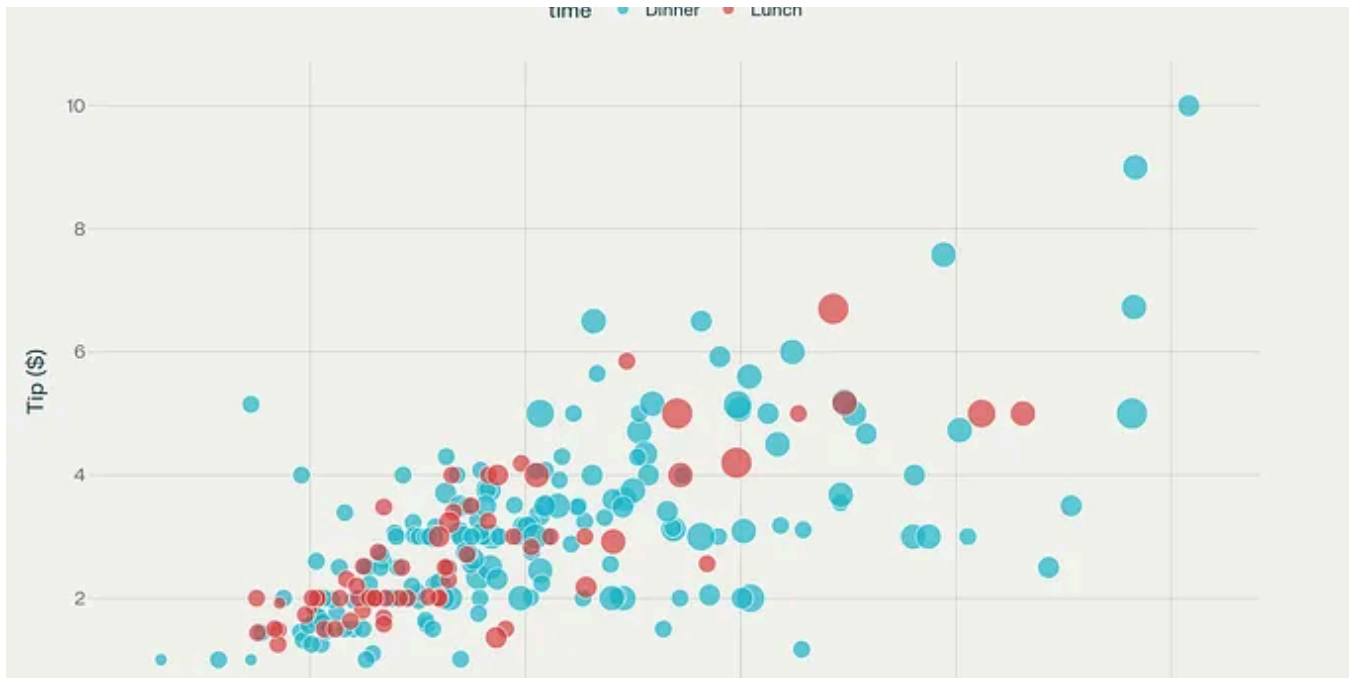


In Geospatial Intelligence by Jan Tschada

## From Search to Spatial Insight: Prototyping Problem-Solving Agents with GeoAI

In classical AI, problem-solving agents transform goals into search problems, compute solutions, and then execute them in the world. In...

★ Oct 13

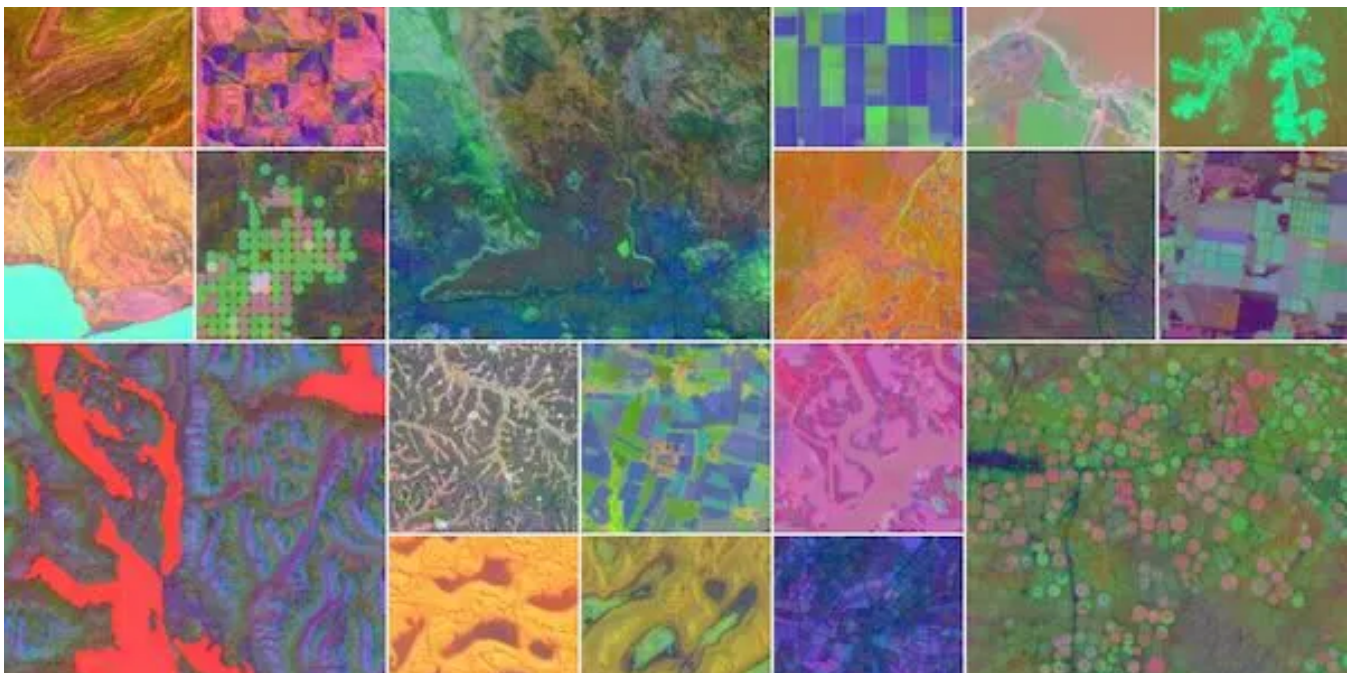


Gaurav Garkoti

## The Complete Guide to Seaborn: Master Statistical Data Visualization in Python

Seaborn represents the pinnacle of statistical data visualization in Python

★ Oct 8 🖱 33 💬 1



'Wine' Roland Mucciarelli

## AlphaEarth Foundation, the new model to map the environment

Having an environmental map of the Earth is a fundamental tool for monitoring its state of health. A few days ago, the news of the release...

★ Aug 7 🤝 22



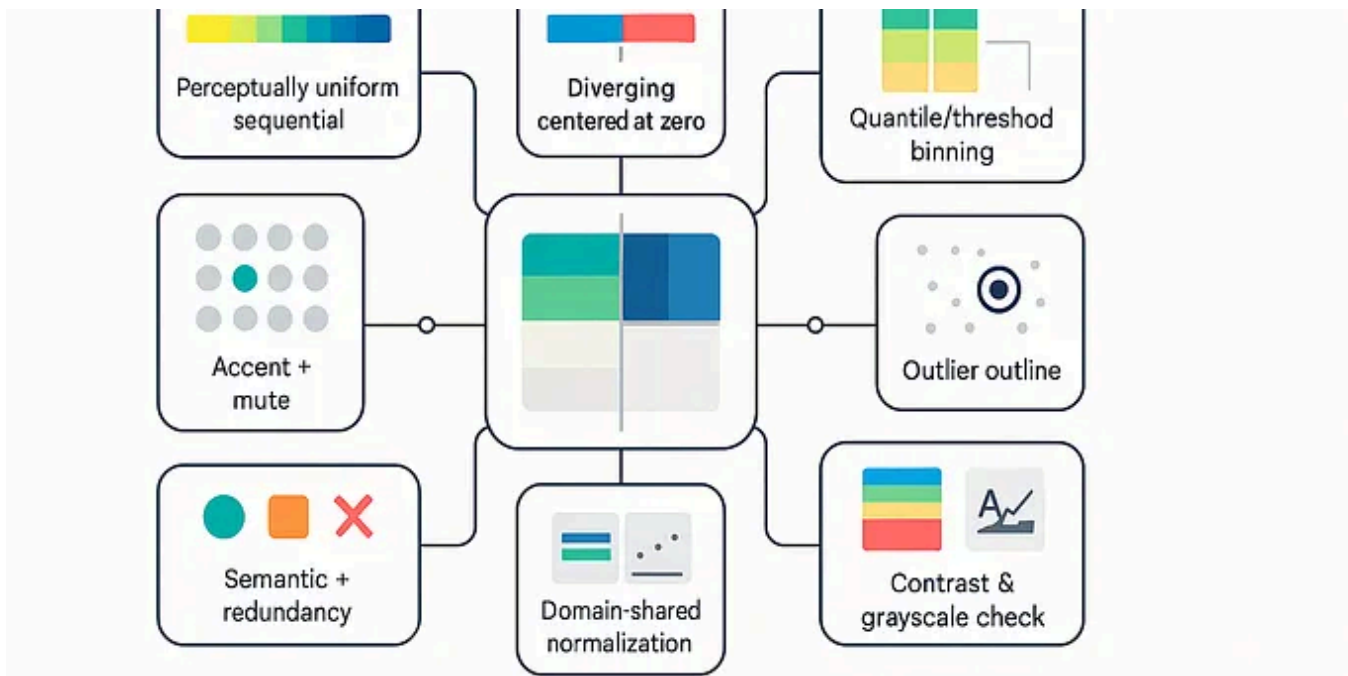
In Google Earth and Earth Engine by Google Earth

## From global to local, how Google's Environmental Insights Explorer is enabling climate action in...

By Willa Ng, Product Strategy, Environmental Insights Explorer; Sofia Reis, Product Go-to-Market Lead, Environmental Insights Explorer and...

Jul 31 🤝 18





 Bhagya Rana

## 8 Color Mapping Tricks That Make Data Pop

Practical ways to spotlight outliers, emphasize thresholds, and guide the eye—without misleading your audience.

★ Sep 15 🖱 13



See more recommendations