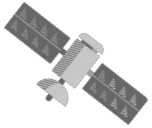
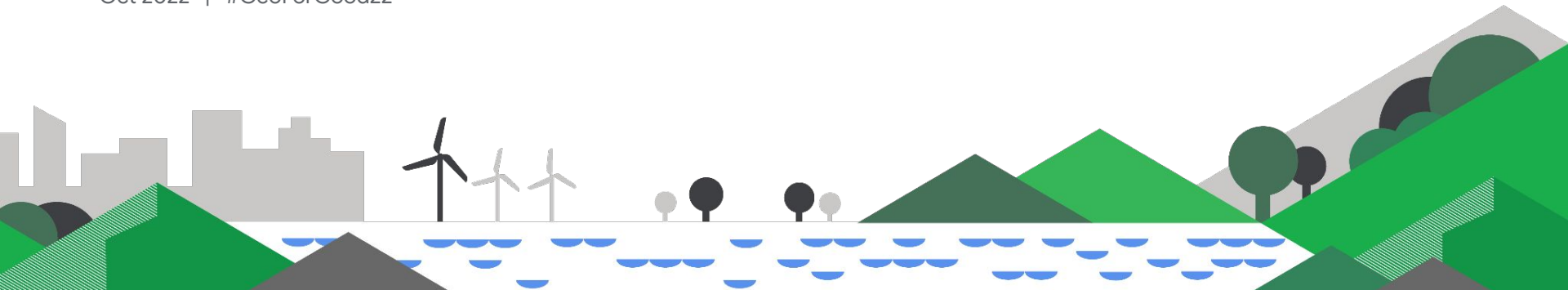


Time Series Visualization

Justin Braaten

Developer Relations Engineer, Google

Oct 2022 | #GeoForGood22



September equinox, 2022

https://code.earthengine.google.com/513dcca0b2a8c16e8650bced28644a04

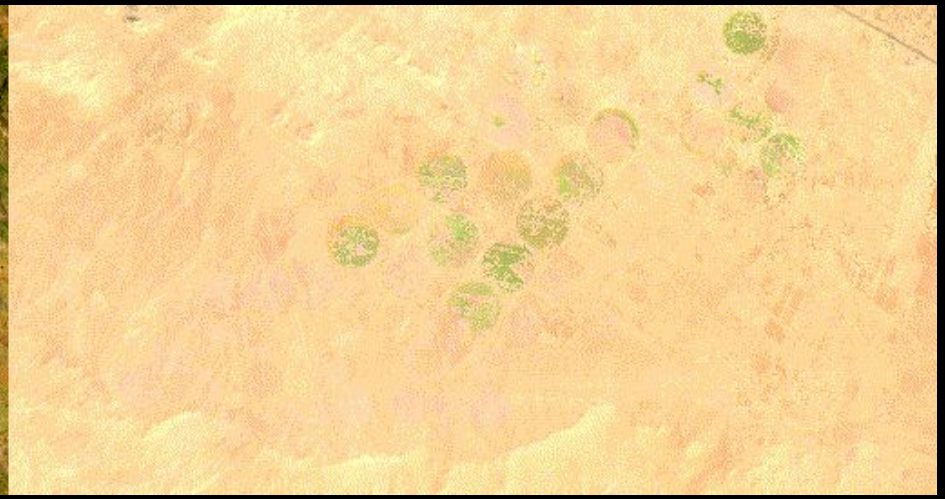
Ucayali River, Peru

Smoke plume over the Pacific from 2019/2020 Australian fires

Coastal erosion, Sagar Island, India

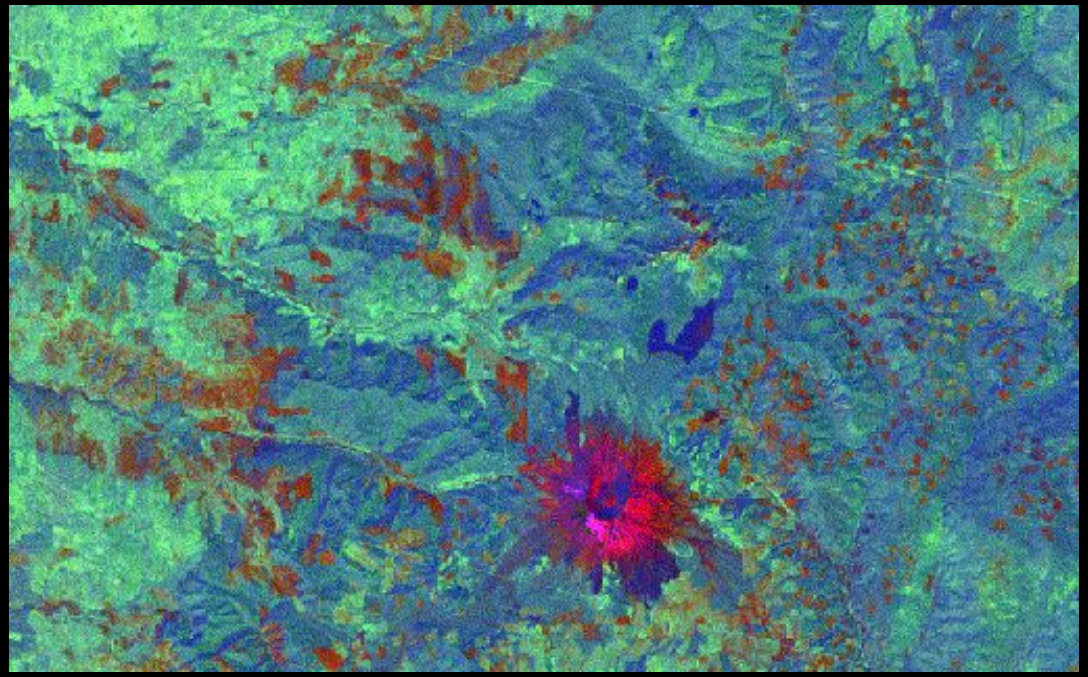
Forest change, Elk Valley, BC

Lake Poopó, Bolivia

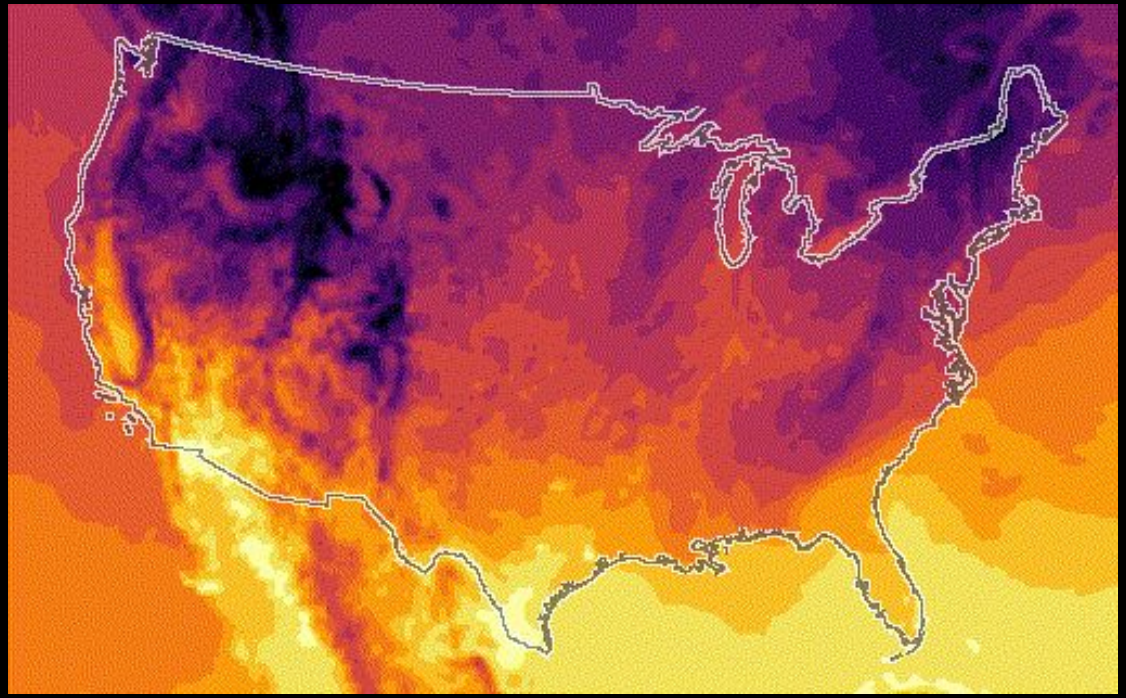


Irrigated fields outside of Nile delta

Mount Saint Helens Eruption, 1980, USA



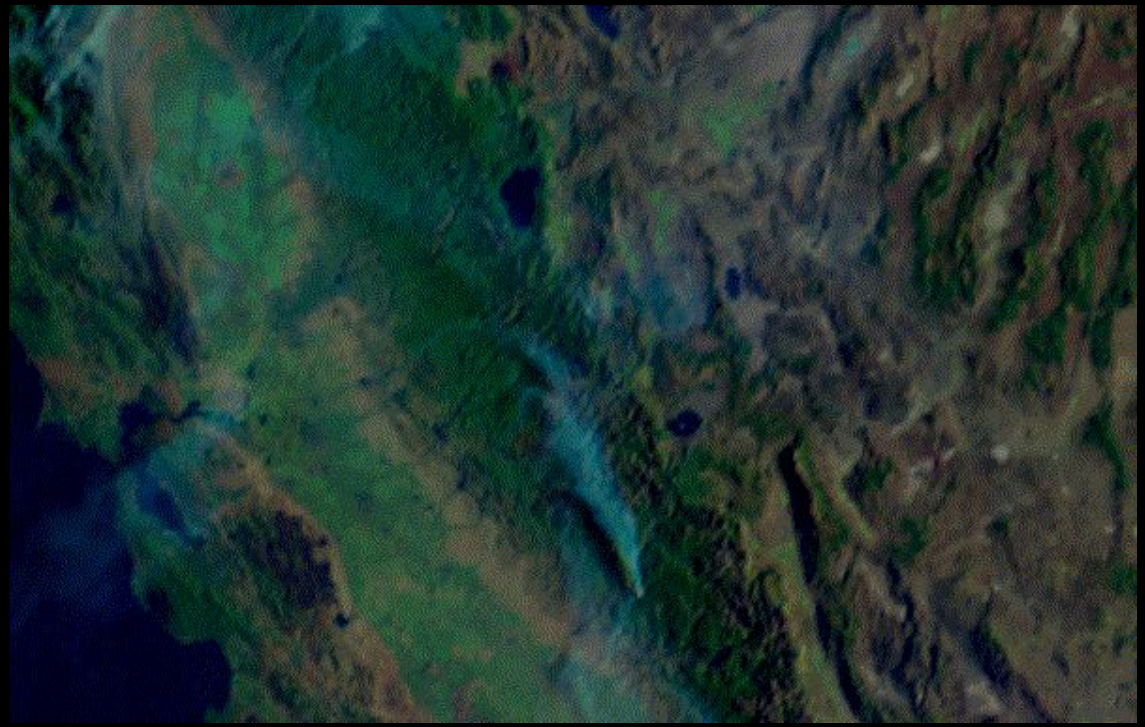
Predicted temperature, 72 hours, USA



#GeoForGood22

https://code.earthengine.google.com/fbc666f9db24fa671384409e63ee91ca

Creek Fire, California, 2020

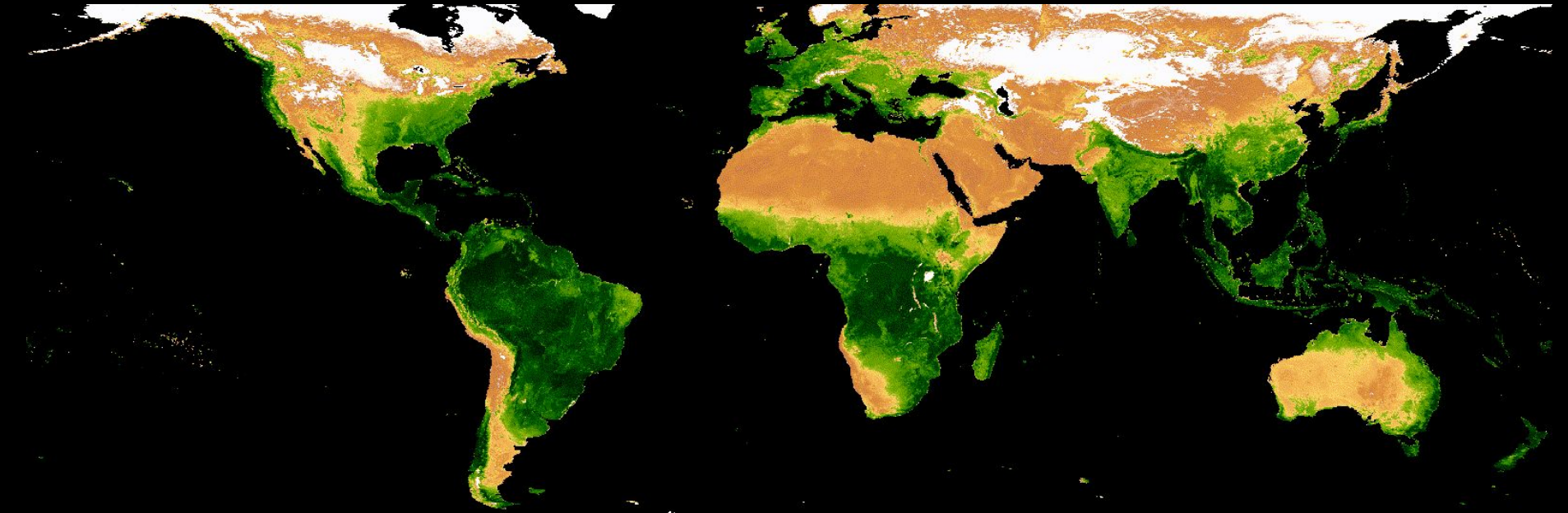


Hurricane Ian, 2022

#GeoForGood22

https://code.earthengine.google.com/9e31da350b394b10bb31e4a181491e47

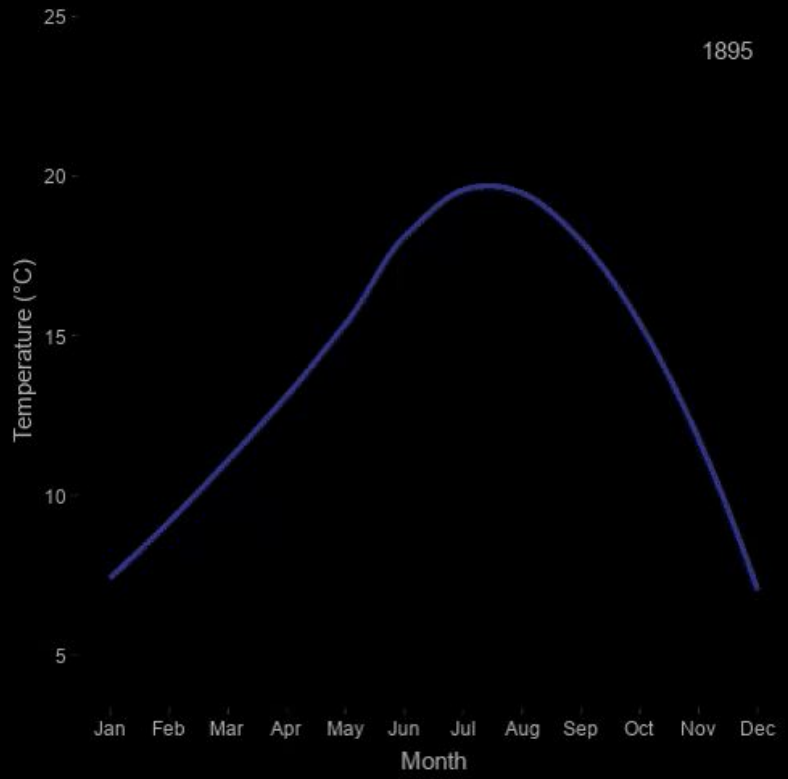
Median monthly NDVI



#GeoForGood22

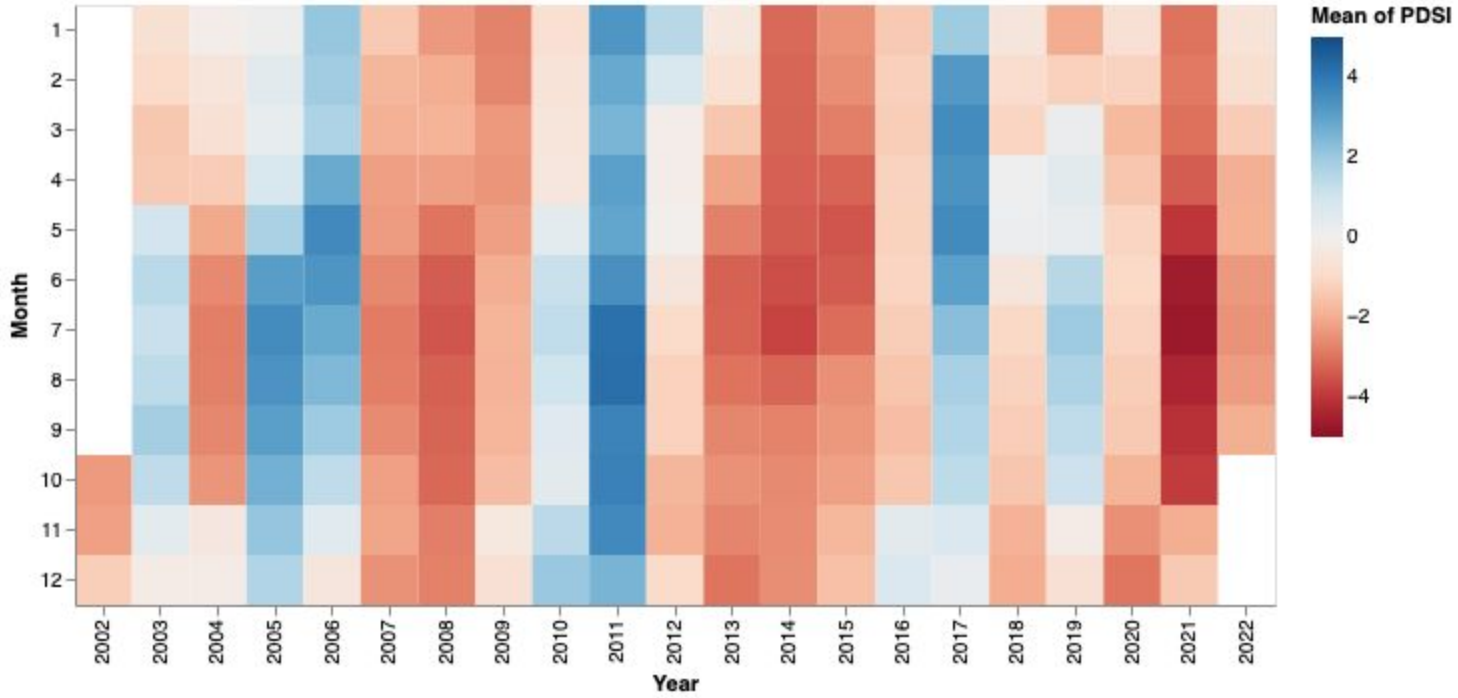
https://code.earthengine.google.com/6e531fcd6efc91ec641d1084ef2ef9ae

Santa Clara County Temperature



#GeoForGood22

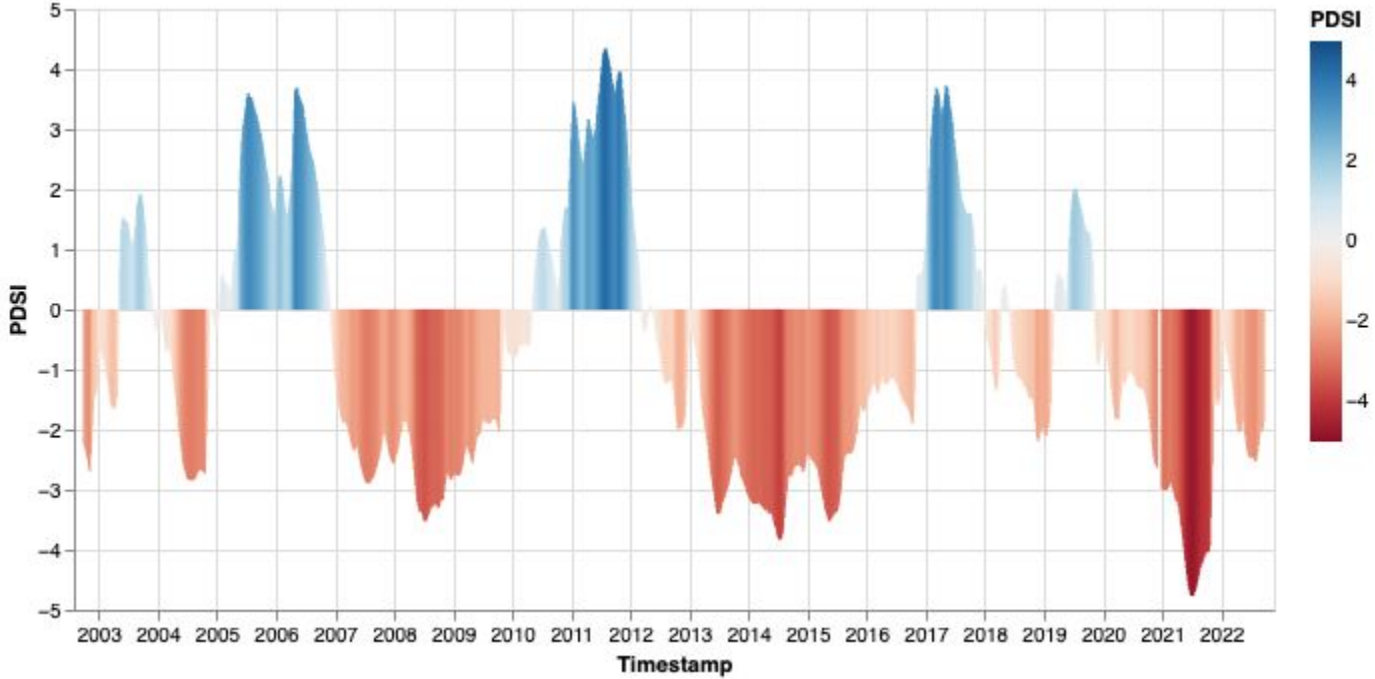
Sierra Nevada ecoregion mean monthly PDSI



#GeoForGood22

https://developers.google.com/earth-engine/tutorials/community/time-series-visualization-with-altair

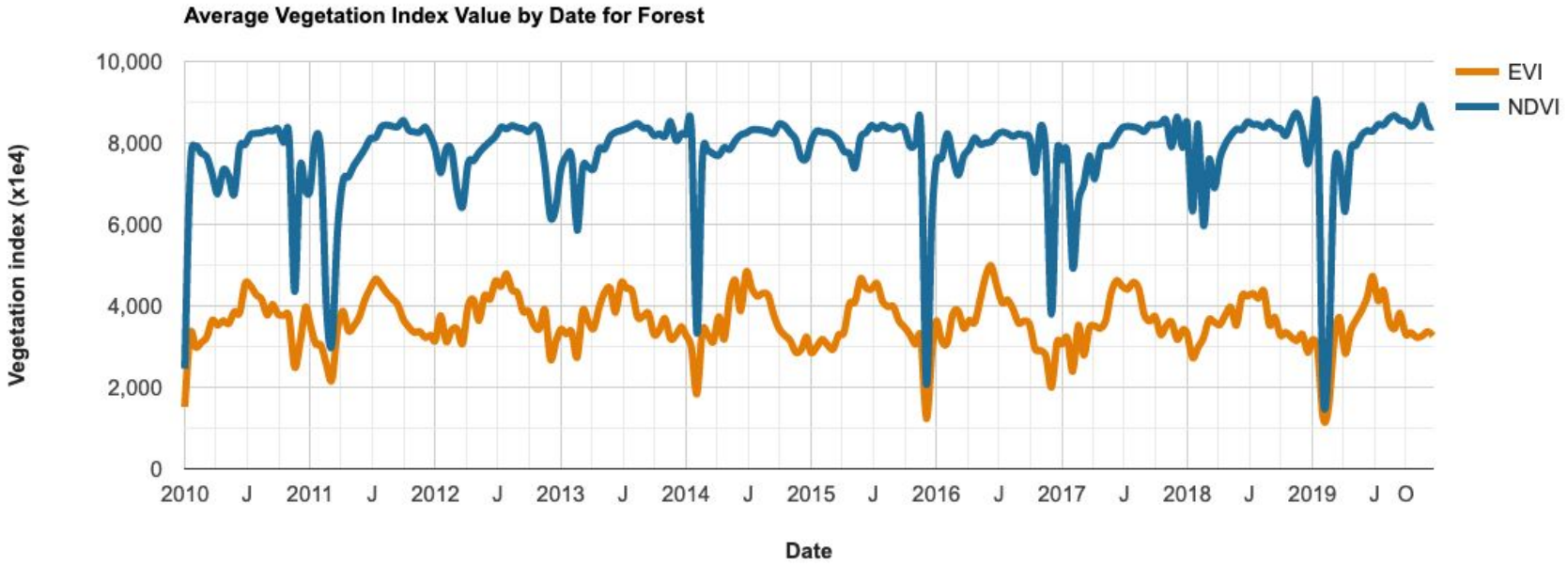
Sierra Nevada ecoregion 5-day PDSI



#GeoForGood22

https://developers.google.com/earth-engine/tutorials/community/time-series-visualization-with-altair

NDVI and EVI for a forest region over time

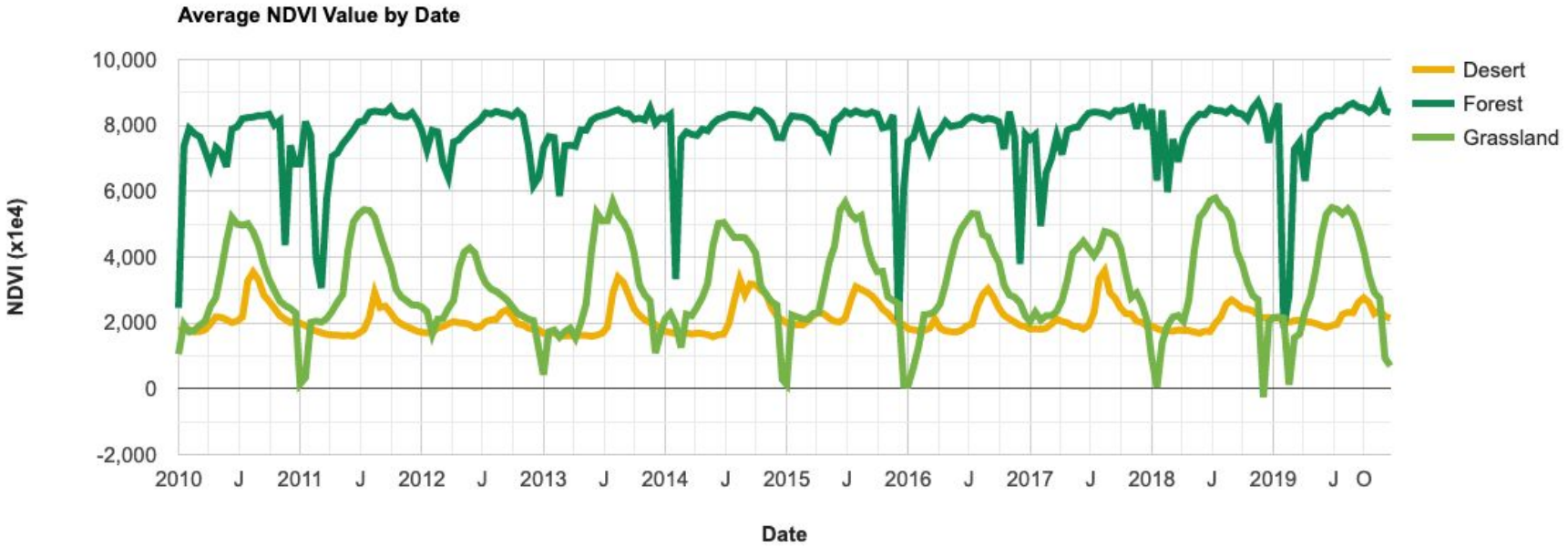



```
ui.Chart.image.series
```

#GeoForGood22

https://developers.google.com/earth-engine/guides/charts_image_collection

NDVI for multiple ecoregions over time

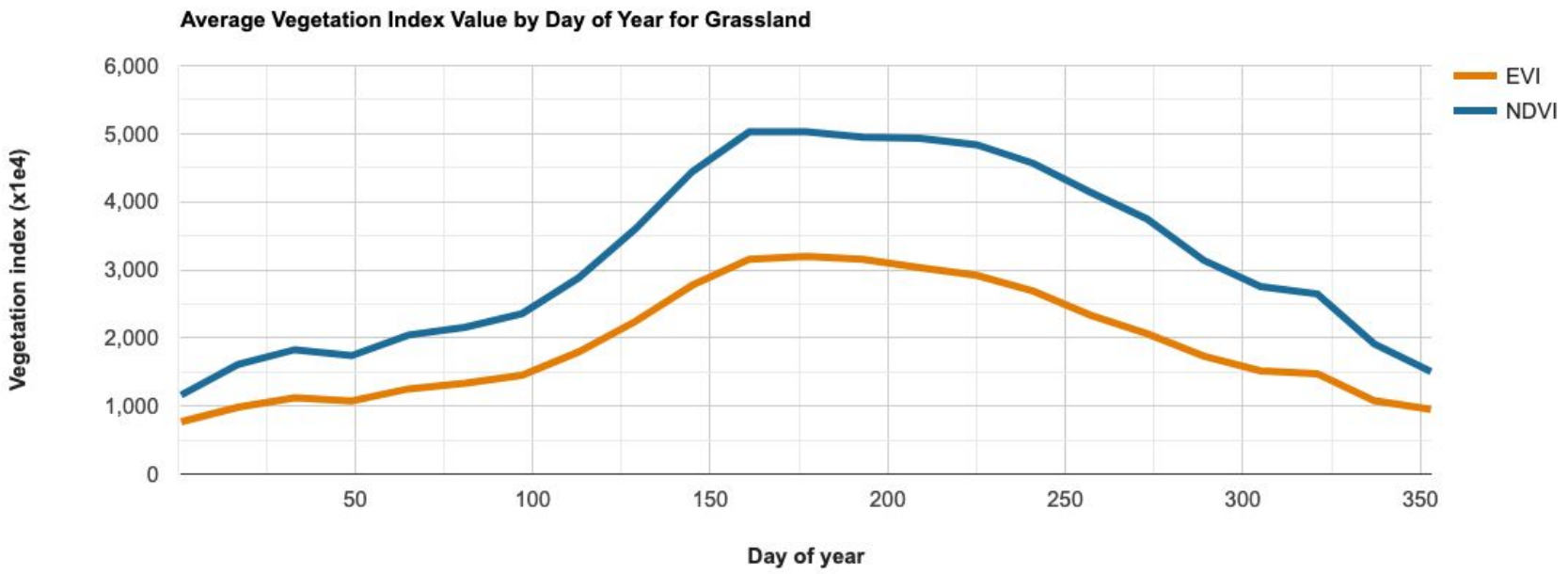


`ui.Chart.image.seriesByRegion`

#GeoForGood22

https://developers.google.com/earth-engine/guides/charts_image_collection

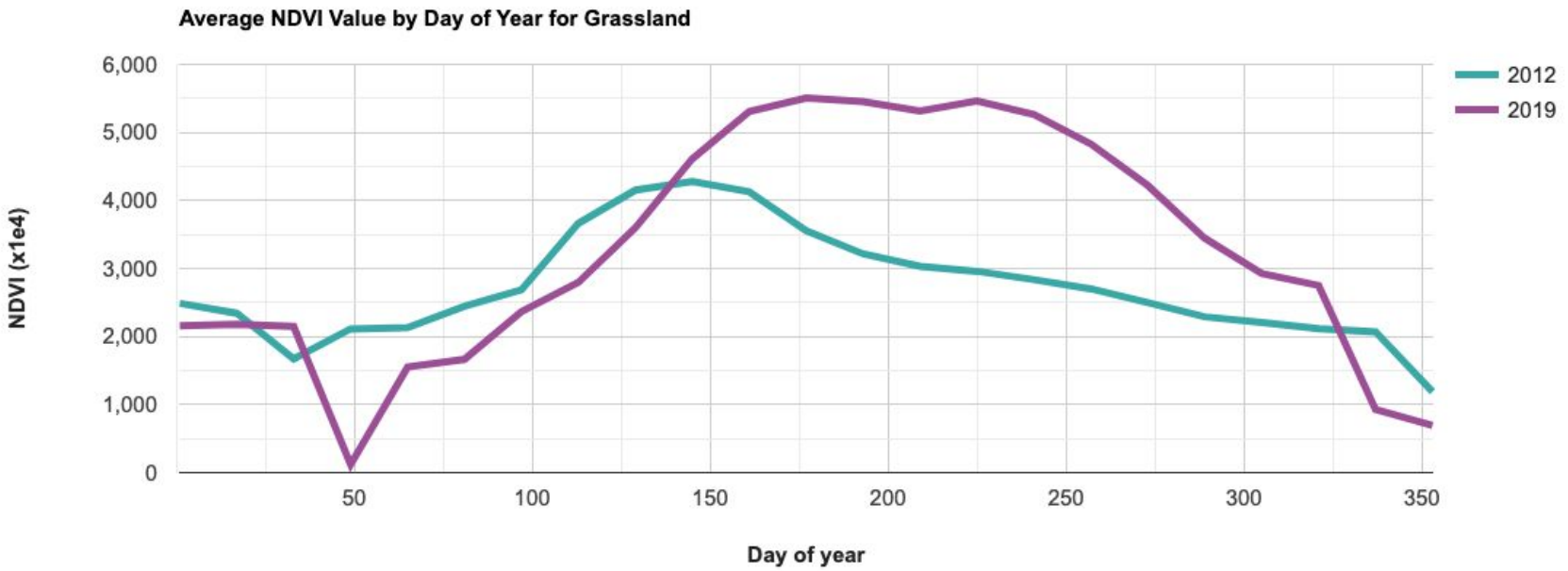
NDVI and EVI by day of year



`ui.Chart.image.doySeries`

https://developers.google.com/earth-engine/guides/charts_image_collection

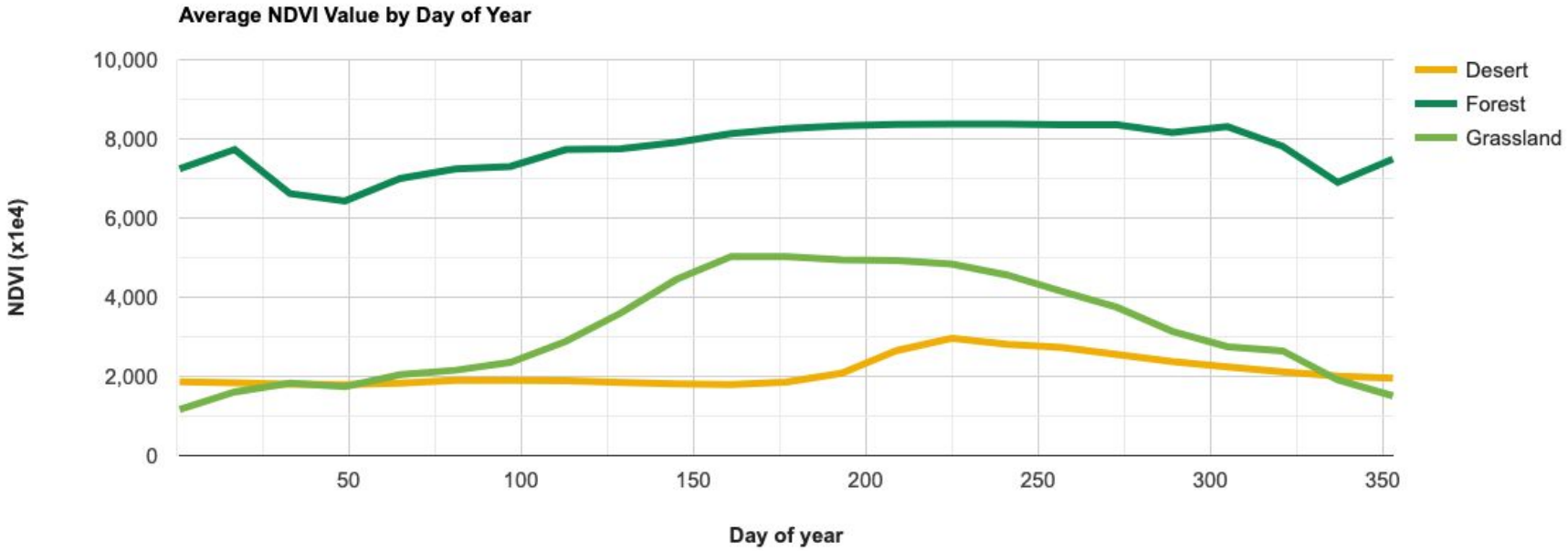
NDVI for two years by day of year




```
ui.Chart.image.doySeriesByYear
```

https://developers.google.com/earth-engine/guides/charts_image_collection

NDVI for multiple ecoregions by day of year

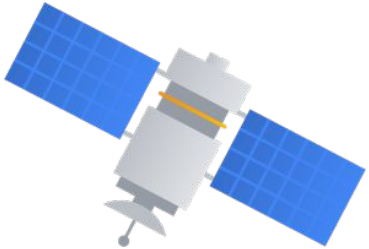


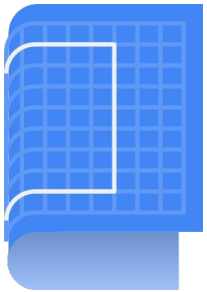

```
ui.Chart.image.doySeriesByRegion
```

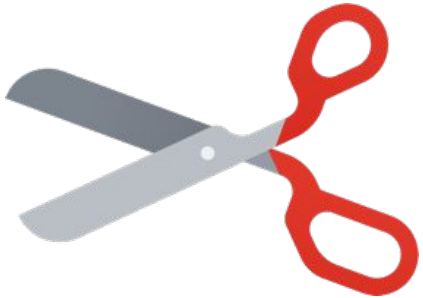
#GeoForGood22

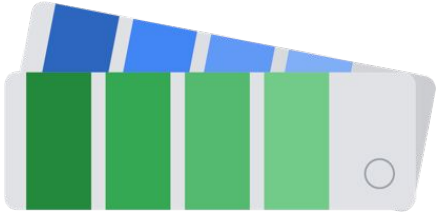
https://developers.google.com/earth-engine/guides/charts_image_collection

Tools of the trade









#GeoForGood22

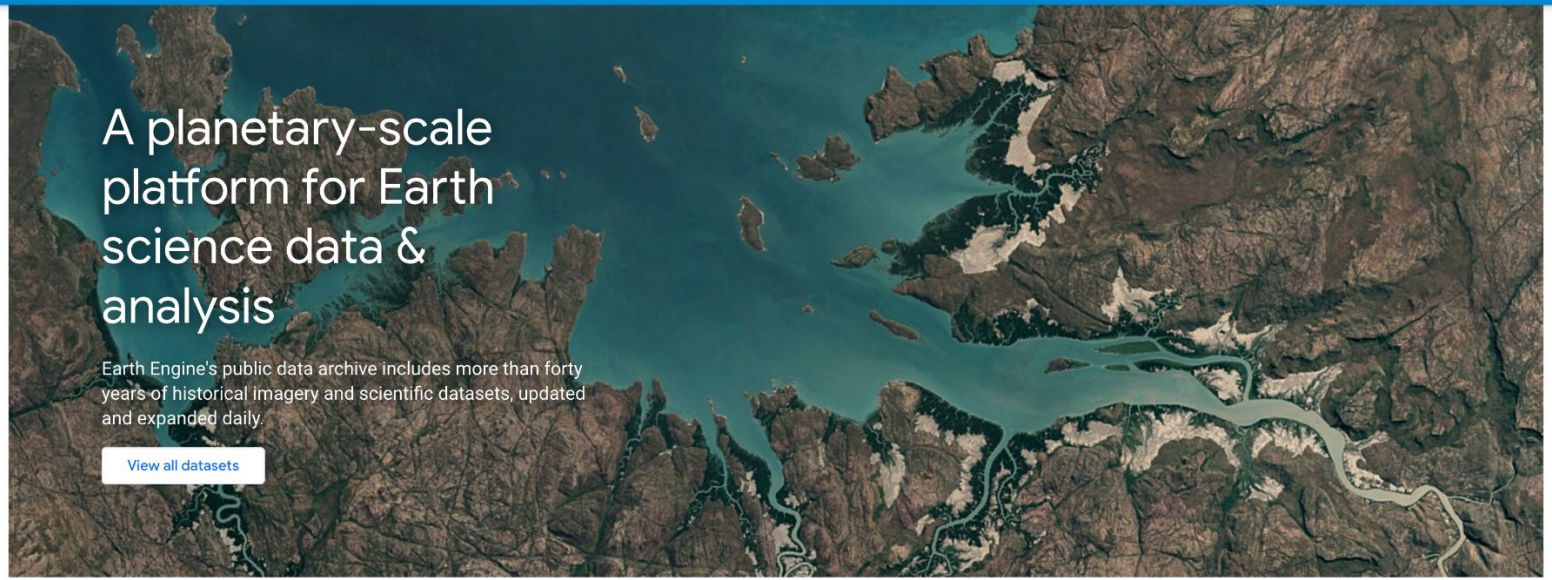
ee.ImageCollection

ee.Image	system:time_start	cloud_cover	quality
	1623355996000	0	100
	1623183196000	20	90

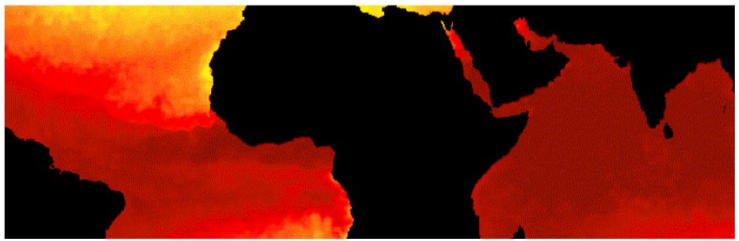
Earth Engine Data Catalog

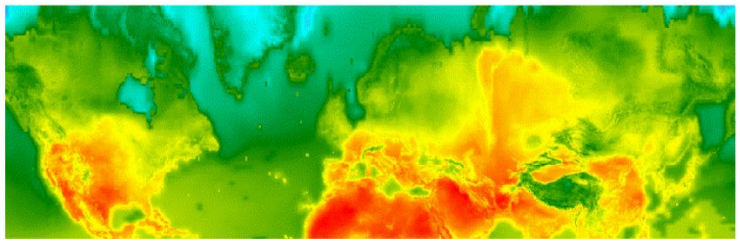
Search | English | Sign in

Home | View all datasets | Browse by tags | Landsat | MODIS | Sentinel | API Docs



Climate and Weather





https://developers.google.com/earth-engine/datasets

Animations

Image collection animation functions

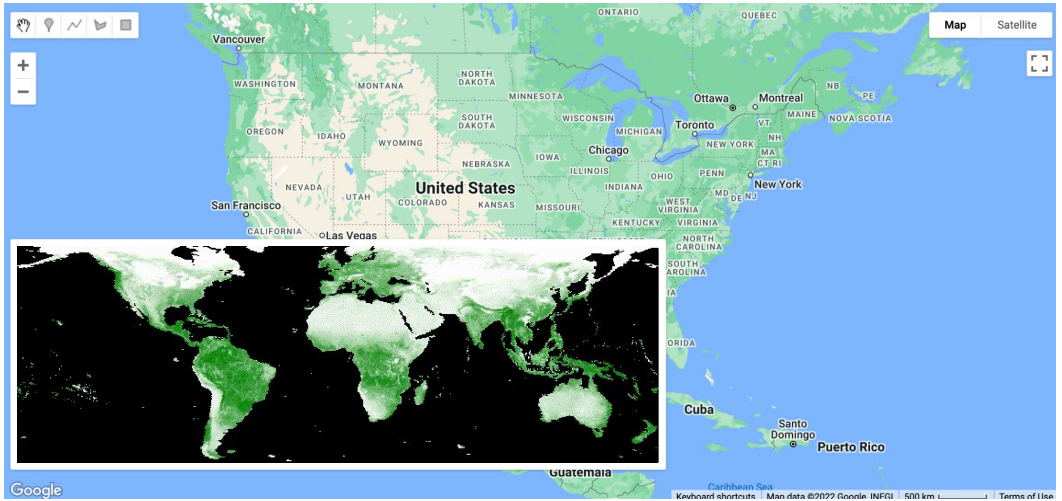
`ui.Thumbnail`	GIF animation as a UI component
`ee.ImageCollection.getVideoThumbURL`	GIF animation downloadable from URL
`Export.video.toDrive`	MP4 animation exported to Google Drive
`Export.video.toCloudStorage`	MP4 animation exported to Cloud Storage bucket
`ee.ImageCollection.getFilmstripThumbURL`	PNG or JPEG collection strip downloadable from URL
`ui.Thumbnail`	PNG or JPEG image as UI component (need to display each frame individually)
`ee.Image.getThumbURL`	PNG or JPEG image as UI component (need to display each frame individually)
`ui.Map.CloudStorageLayer`	Map tile layers with alternating visibility

Basic workflow

1. Import image collection
2. Filter collection by bounds and time
3. Apply styling
4. Create animation

ui.Thumbnail

```
// Fetch a MODIS NDVI collection and select NDVI.
var col = ee.ImageCollection('MODIS/006/MOD13A2')
  .filterDate('2018-01-01', '2019-01-01');

// Define visualization parameters.
var params = {
  bands: ['NDVI'],
  min: 0.0,
  max: 9000.0,
  palette: ['white', 'green'],
  region: ee.Geometry.BBox(-180, -58, 180, 63),
  dimensions: 700,
  crs: 'EPSG:32662',
  framesPerSecond: 10
};

// Render the GIF animation.
var animation = ui.Thumbnail({
  image: col,
  params: params,
  style: {position: 'bottom-left'}
});

// Add the animation to the map.
Map.add(animation);
```



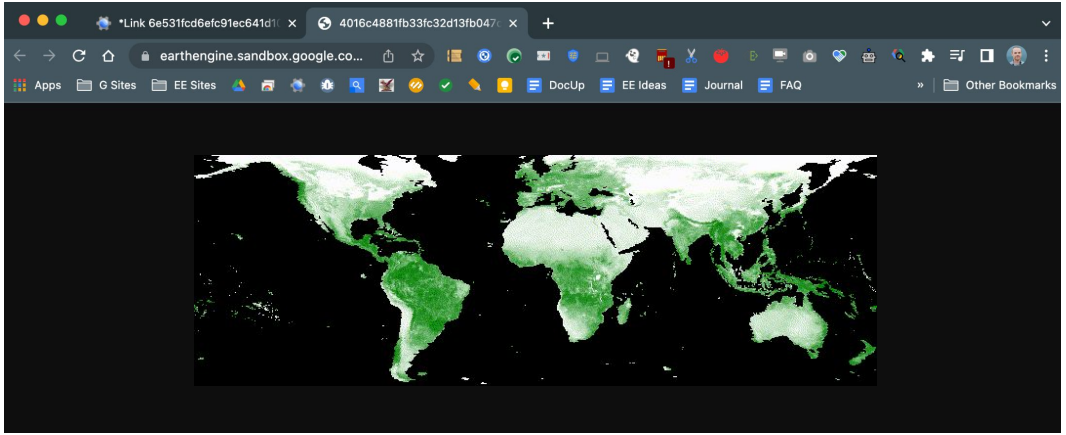
#GeoForGood22

https://code.earthengine.google.com/bb4c726fd3c969e03d2f967008719059

ee.ImageCollection.getVideoThumbURL

```
// Fetch a MODIS NDVI collection.
var col = ee.ImageCollection('MODIS/006/MOD13A2')
  .filterDate('2018-01-01', '2019-01-01');

// Define visualization parameters.
var params = {
  bands: ['NDVI'],
  min: 0.0,
  max: 9000.0,
  palette: ['white', 'green'],
  region: ee.Geometry.BBox(-180, -58, 180, 63),
  dimensions: 700,
  crs: 'EPSG:32662',
  framesPerSecond: 10
};

// Get a URL that will generate a GIF.
var url = col.getVideoThumbURL(params);

// Print the GIF URL to the console.
print(url);
```

https://earthengine.sandbox.google.com/v1alpha/projects/ee-braaten/videoThumbnails/4016c4881fb33fc32d13fb047cb6e656-d7546a9843a469a584eeb26a00218d7a:getPixels



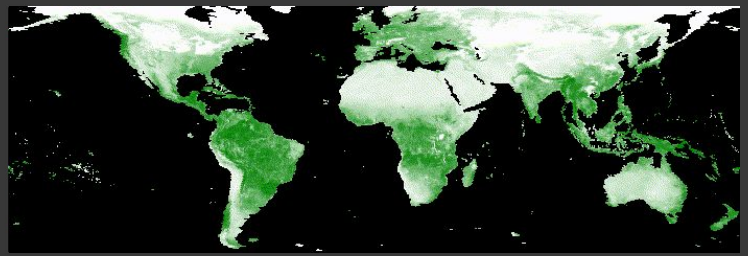
https://code.earthengine.google.com/4ec6fa231ece23fe6320bd6ee7a58cdc

#GeoForGood22

ee.ImageCollection.getVideoThumbURL

…in Colab with Python API

```
[2] from IPython.display import Image

# Fetch a MODIS NDVI collection.
col = (ee.ImageCollection('MODIS/006/MOD13A2')
  .filterDate('2018-01-01', '2019-01-01'))

# Define visualization parameters.
params = {
  'bands': ['NDVI'],
  'min': 0,
  'max': 9000,
  'palette': ['white', 'green'],
  'region': ee.Geometry.BBox(-180, -58, 180, 63),
  'dimensions': 700,
  'crs': 'EPSG:32662',
  'framesPerSecond': 10
}

# Get a URL that will generate a GIF.
url = col.getVideoThumbURL(params)

# Display the NDVI animation.
Image(url=url)
```

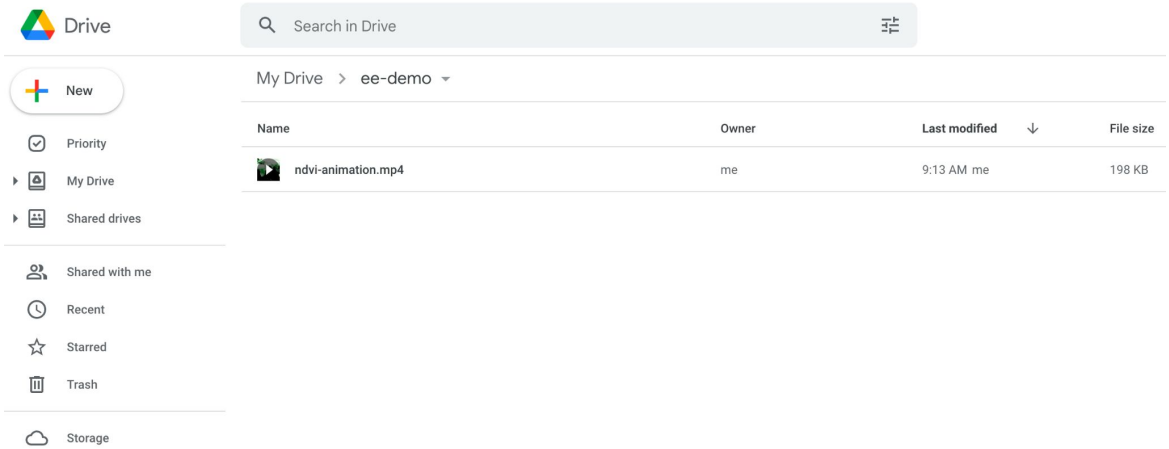


#GeoForGood22

https://gist.github.com/jdbcode/af0c78875e7337ac95821edd93516d26

Export.video.toDrive

```
// Fetch a MODIS NDVI collection.
var col = ee.ImageCollection('MODIS/006/MOD13A2')
  .filterDate('2018-01-01', '2019-01-01');

// Define visualization parameters.
var visParams = {
  bands: ['NDVI'],
  min: 0,
  max: 9000,
  palette: ['white', 'green'],
};

// Make each image an RGB visualization.
var visCol = col.map(function(img) {
  return img.visualize(visParams);
});

// Export animation to Drive.
Export.video.toDrive({
  collection: visCol,
  description: 'ndvi-animation',
  folder: 'ee-demo',
  region: ee.Geometry.BBox(-180, -58, 180, 63),
  dimensions: 700,
  crs: 'EPSG:32662',
  framesPerSecond: 10
});
```

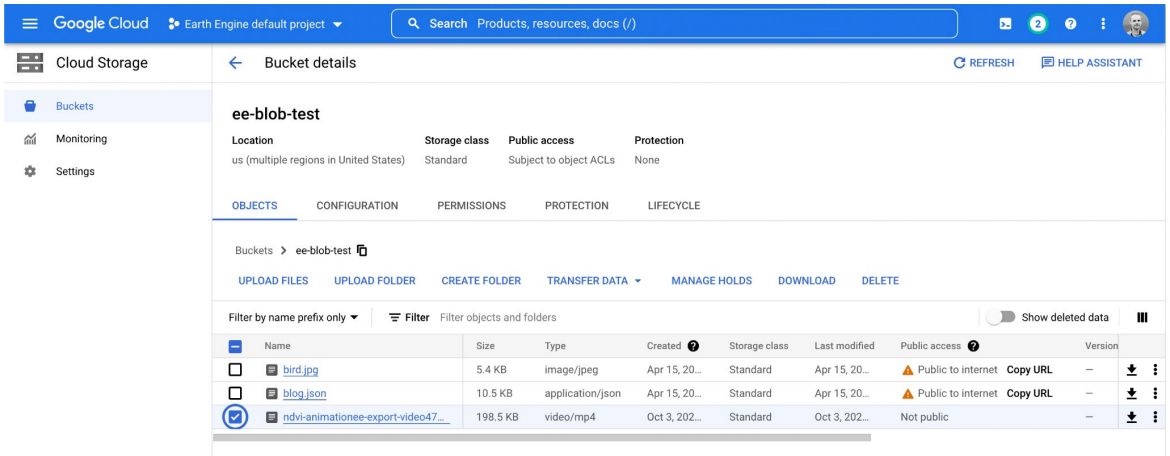


#GeoForGood22

https://code.earthengine.google.com/b8df63e4fa522c4bdc34eeb7231cda04

Export.video.toCloudStorage

```
// Fetch a MODIS NDVI collection.
var col = ee.ImageCollection('MODIS/006/MOD13A2')
  .filterDate('2018-01-01', '2019-01-01');

// Define visualization parameters.
var visParams = {
  bands: ['NDVI'],
  min: 0,
  max: 9000,
  palette: ['white', 'green'],
};

// Make each image an RGB visualization.
var visCol = col.map(function(img) {
  return img.visualize(visParams);
});

// Export animation to Cloud Storage.
Export.video.toCloudStorage({
  collection: visCol,
  description: 'ndvi-animation',
  bucket: 'bucket-name',
  region: ee.Geometry.BBox(-180, -58, 180, 63),
  dimensions: 700,
  crs: 'EPSG:4087',
  framesPerSecond: 10
});
```

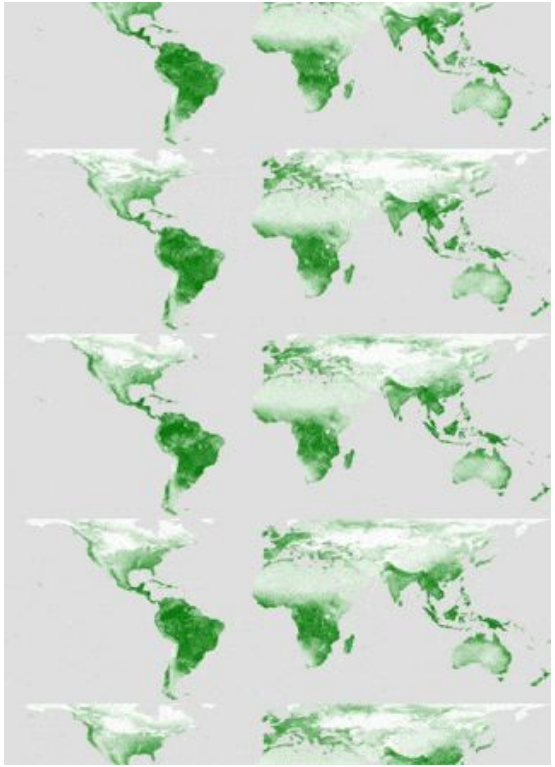


#GeoForGood22

https://code.earthengine.google.com/23e91a44034f8f4aece8fd3ee3b4b49a

ee.ImageCollection.getFilmstripThumbURL

```
// Fetch a MODIS NDVI collection.
var col = ee.ImageCollection('MODIS/006/MOD13A2')
  .filterDate('2018-01-01', '2019-01-01');

// Define visualization parameters.
var params = {
  bands: ['NDVI'],
  min: 0,
  max: 9000,
  palette: ['white', 'green'],
  region: ee.Geometry.BBox(-180, -58, 180, 63),
  dimensions: 700,
  crs: 'EPSG:4087',
  framesPerSecond: 10
};

// Get a URL that will generate a filmstrip image.
var url = col.getFilmstripThumbURL(params);

// Print the filmstrip URL to the console.
print(url);
```



#GeoForGood22

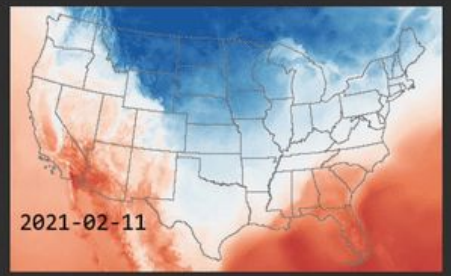
https://code.earthengine.google.com/c69b897a42c0bc3079a418b8a9ec0ae6

ee.ImageCollection.getFilmstripThumbURL

…in Colab with Python API

```
[22] gif_name = 'filmstrip_magick.gif'
     delay = 10

     cmd = f'convert -loop 0 -delay {delay} *_frame.png {gif_name}'
     print(cmd)
     os.system(cmd)

     convert -loop 0 -delay 10 *_frame.png filmstrip_magick.gif
     0
```

```
[23] with open('filmstrip_magick.gif','rb') as f:
         display(ImageGIF(data=f.read(), format='png'))
```



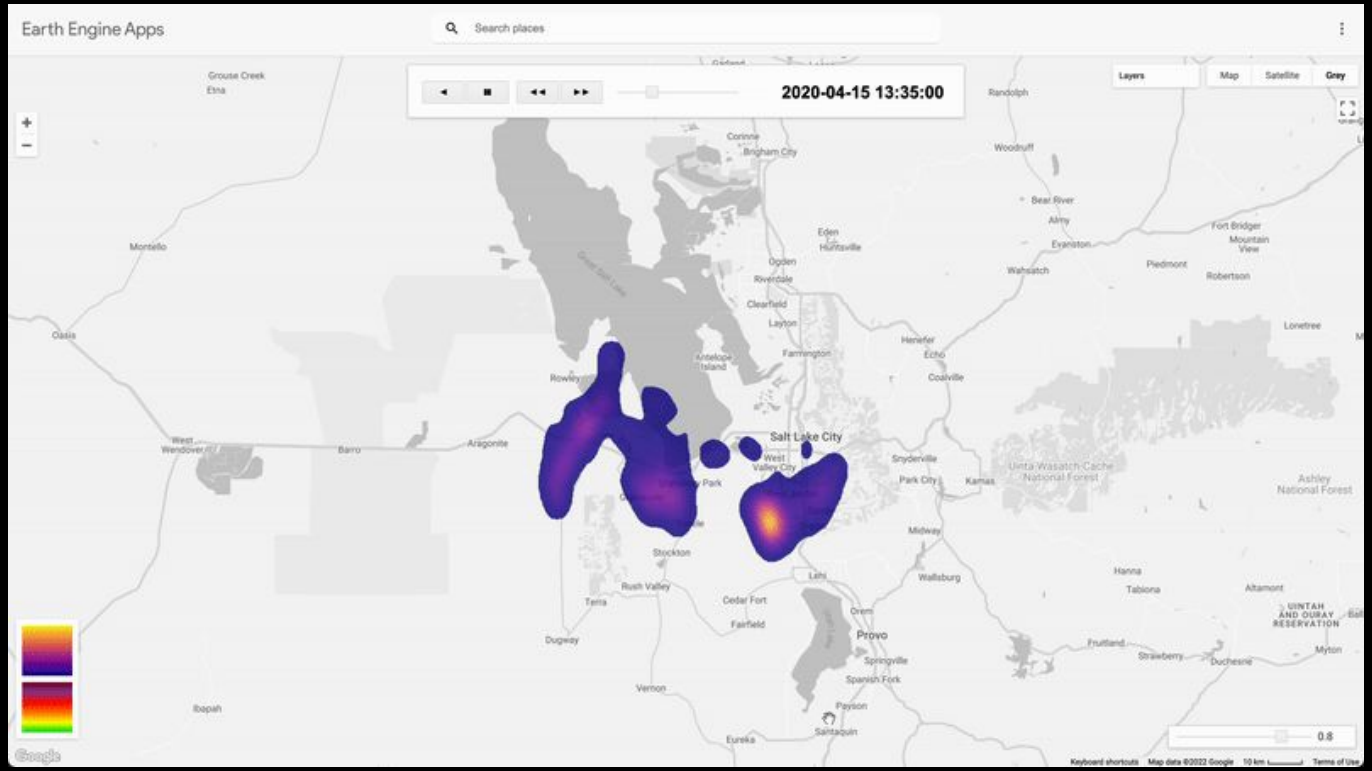
Compress the GIF

```
in_gif = 'filmstrip_magick.gif'
out_gif = 'filmstrip_magick_compress.gif'
fuzz = 5  # percent

cmd = f'convert -fuzz {fuzz}% -layers Optimize {in_gif} {out_gif}'
print(cmd)
os.system(cmd)
```

#GeoForGood22

https://gist.github.com/jdbcode/a46c684c9a22856f2a55ef4a996259c6

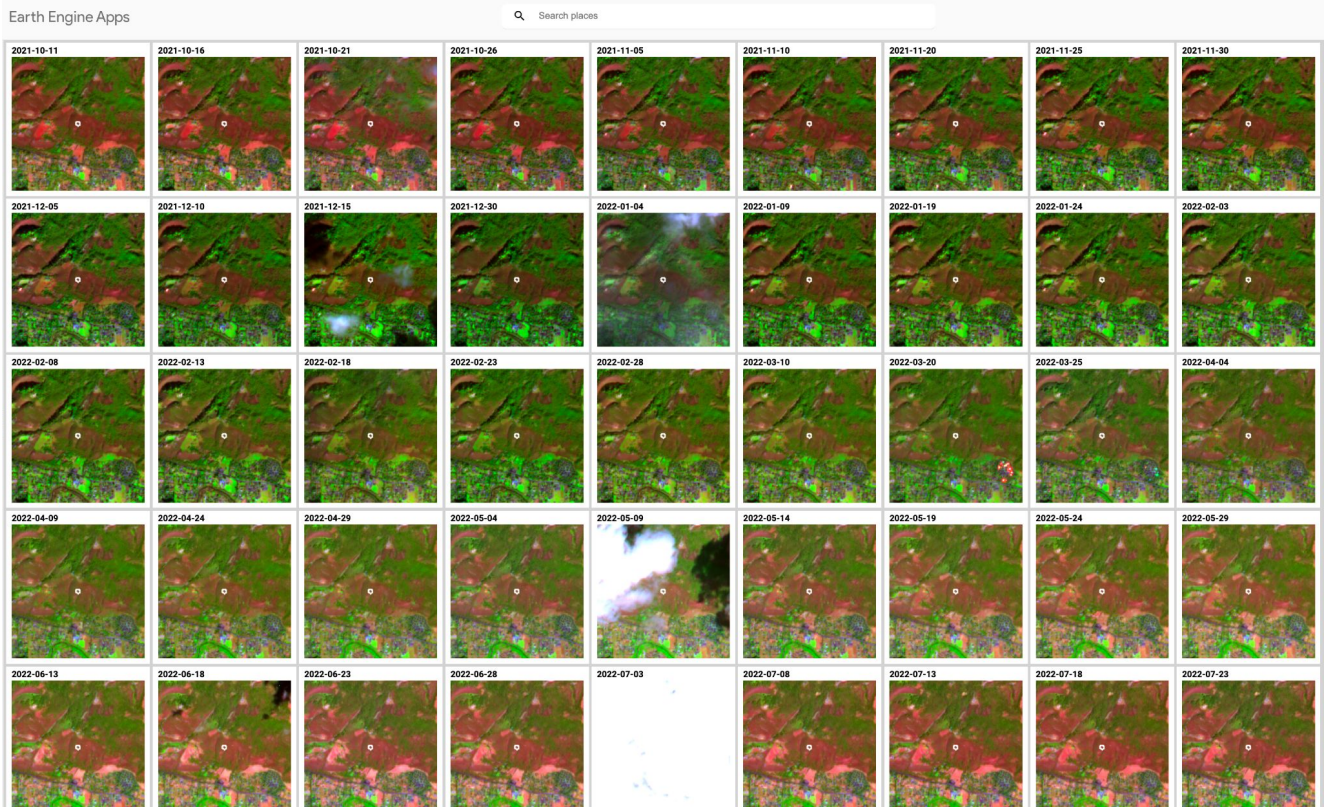
ui.Map.CloudStorageLayer series



#GeoForGood22

Credit: David Tryse

`ui.Thumbnail` gallery



https://github.com/jdbcode/ee-rgb-timeseries/blob/main/eo-timeseries-explorer.js

Image style

Image styling

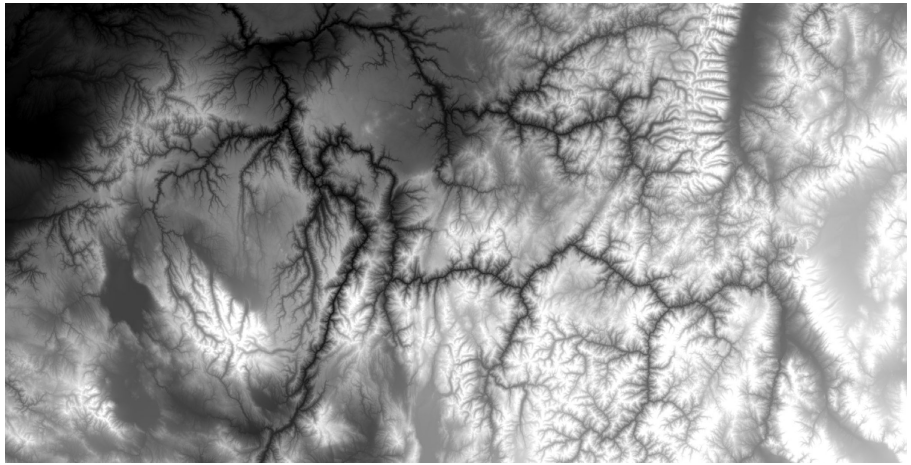
One way or another all image data to be visualized needs to conform to PNG or JPEG format.

- 8-bit (values 0 to 255)
- grayscale, paletted, or RGB(A)

`ee.Image.visualize`

- **`bands`**
  - a list of either a single band or three bands
- **`min`**
  - the data value to associate with 0 in 8-bit image
- **`max`**
  - the data value to associate with 255 in 8-bit image
- **`palette`**
  - a list of color values to scale linearly between
- **`opacity`**
  - value from 0 (transparent) to 1 (opaque)

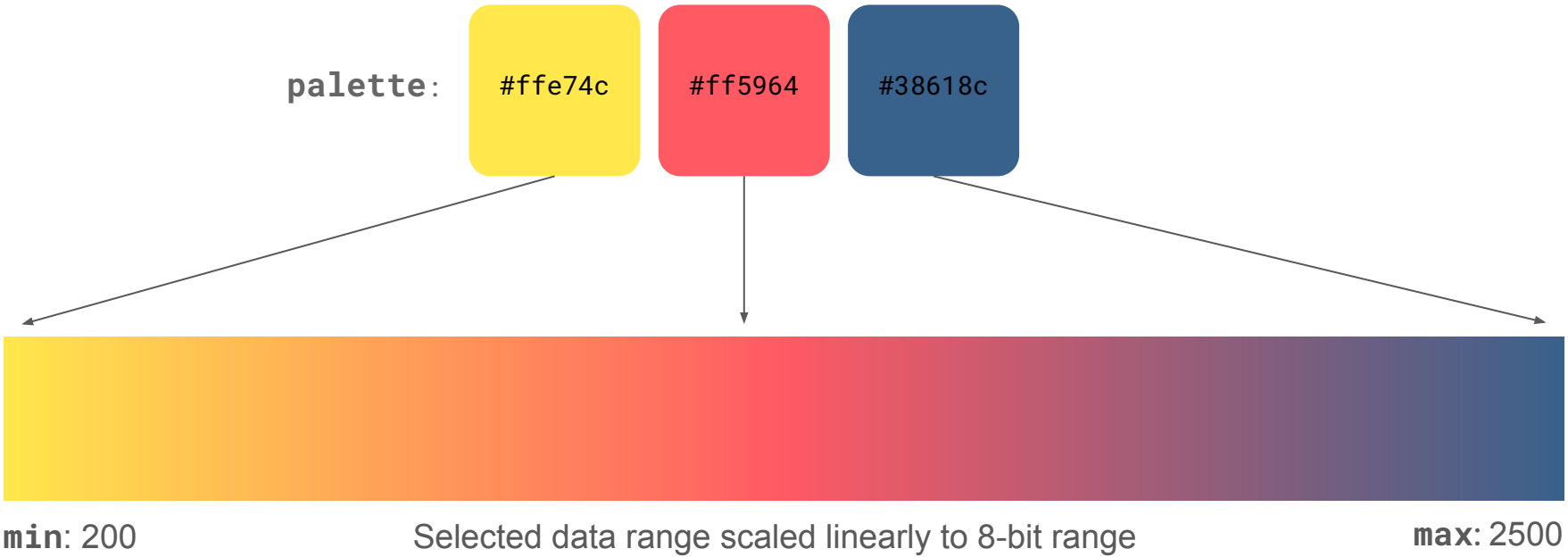
Grayscale



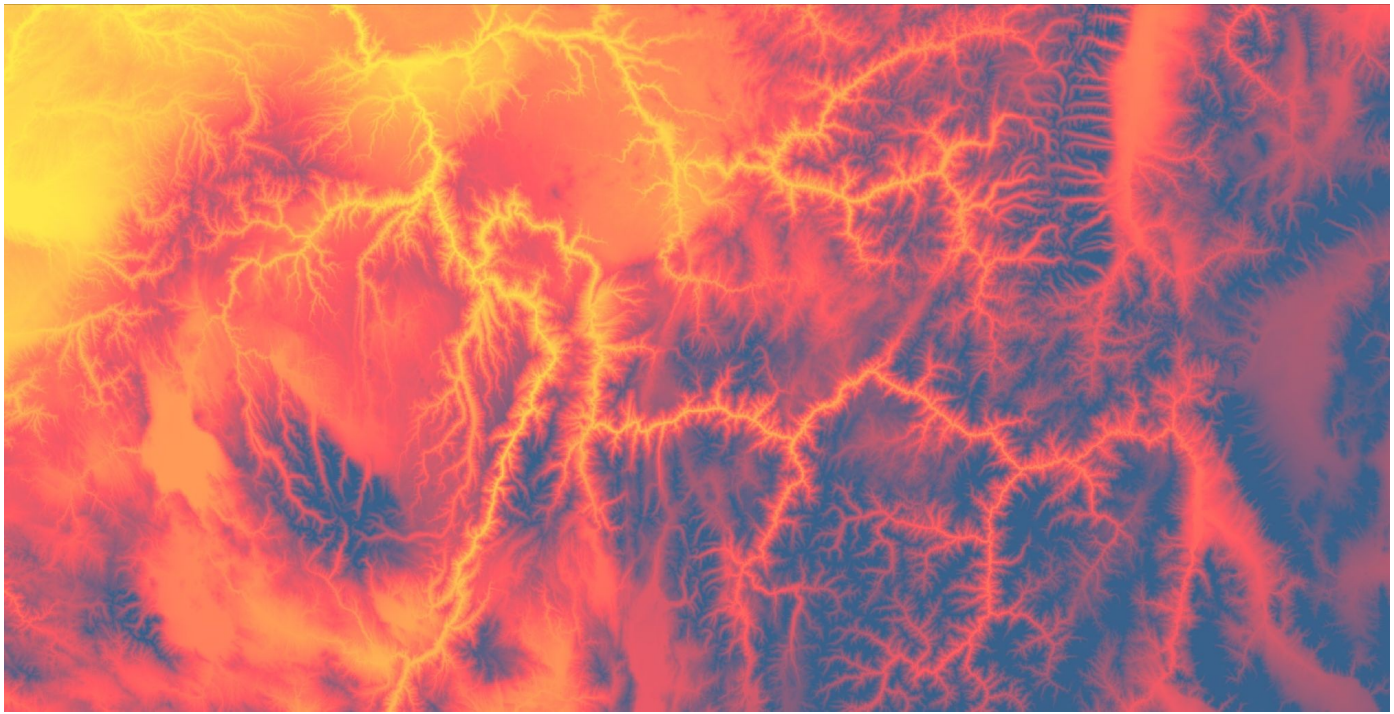
0 8-bit range 255

`min`: 200 Selected data range scaled linearly to 8-bit range **`max`**: 2500

Color palette



Color palette



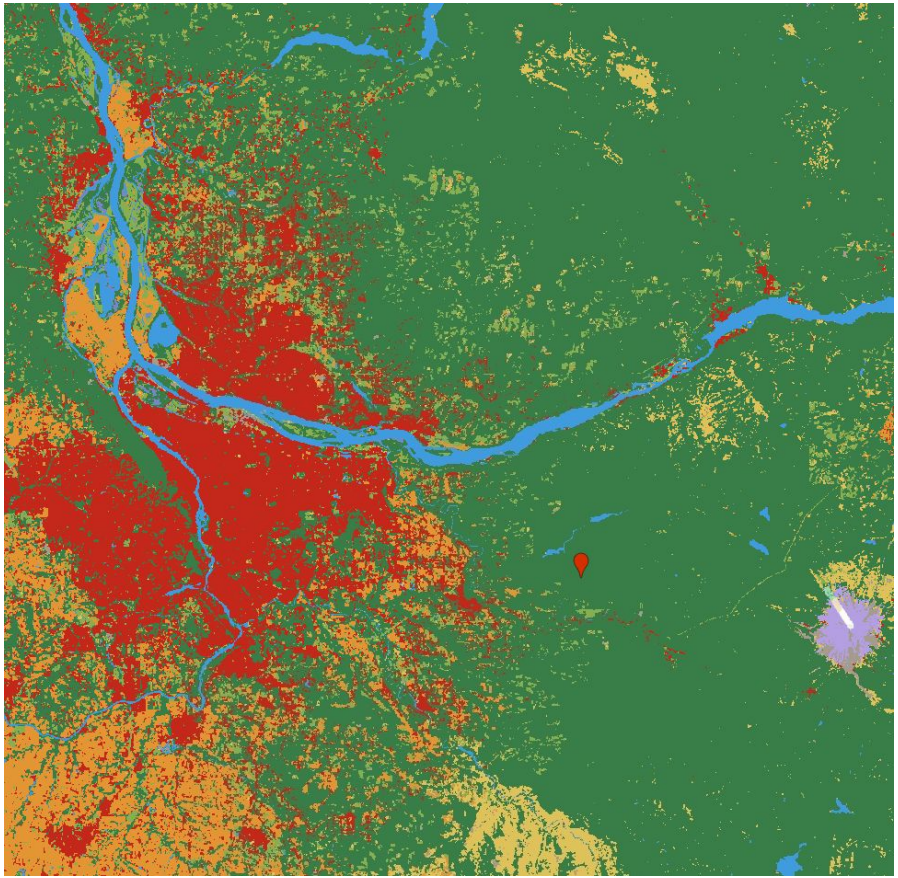
Gray and color palette code

```
var elev = ee.Image('USGS/3DEP/10m');

var grayVis = elev.visualize({
  bands: ['elevation'],
  min: 200,
  max: 2500,
});

var paletteVis = elev.visualize({
  bands: ['elevation'],
  min: 200,
  max: 2500,
  palette: ['ffe74c', 'ff5964', '38618c']
});
```

Categorical palette

- Category values should be sequential 0, 1, 2, …
- Set **`min`** and **`max`** equal to sequence limits
- **`palette`** length equal to n categories

```
var landcoverVis = landcover.visualize({
  bands: ['label'],
  min: 0,
  max: 8,
  palette: ['419BDF', '397D49', '88B053',
            '7A87C6', 'E49635', 'DFC35A',
            'C4281B', 'A59B8F', 'B39FE1']
});
```



RGB

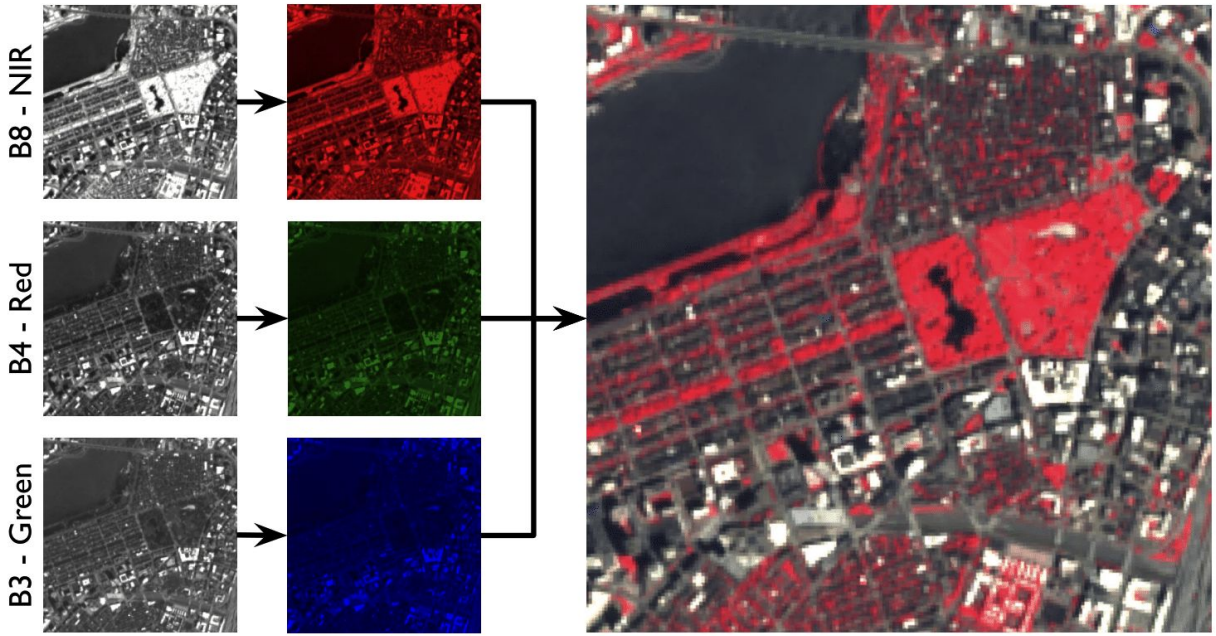


Image Credit: Valerie Pasquarella

RGB code

```
var trueColor = img.visualize({
  bands: ['B4', 'B3', 'B2'],
  min: 100,
  max: 2000
});
```

```
var falseColor = img.visualize({
  bands: ['B11', 'B8', 'B2'],
  min: [100, 75, 50],
  max: [3500, 3400, 3300]
});
```

Applying image style to a collection

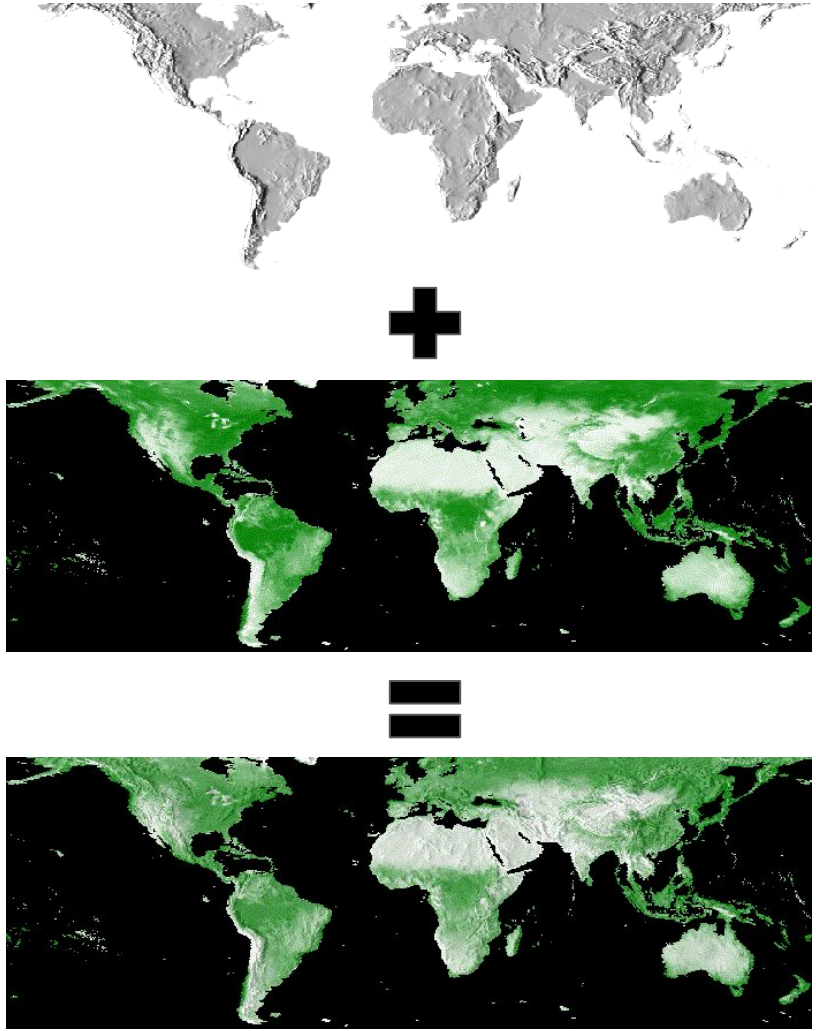
Most of the video and thumbnail functions provide style parameters that run something like **`ee.Image.visualize`** on each image the collection automatically

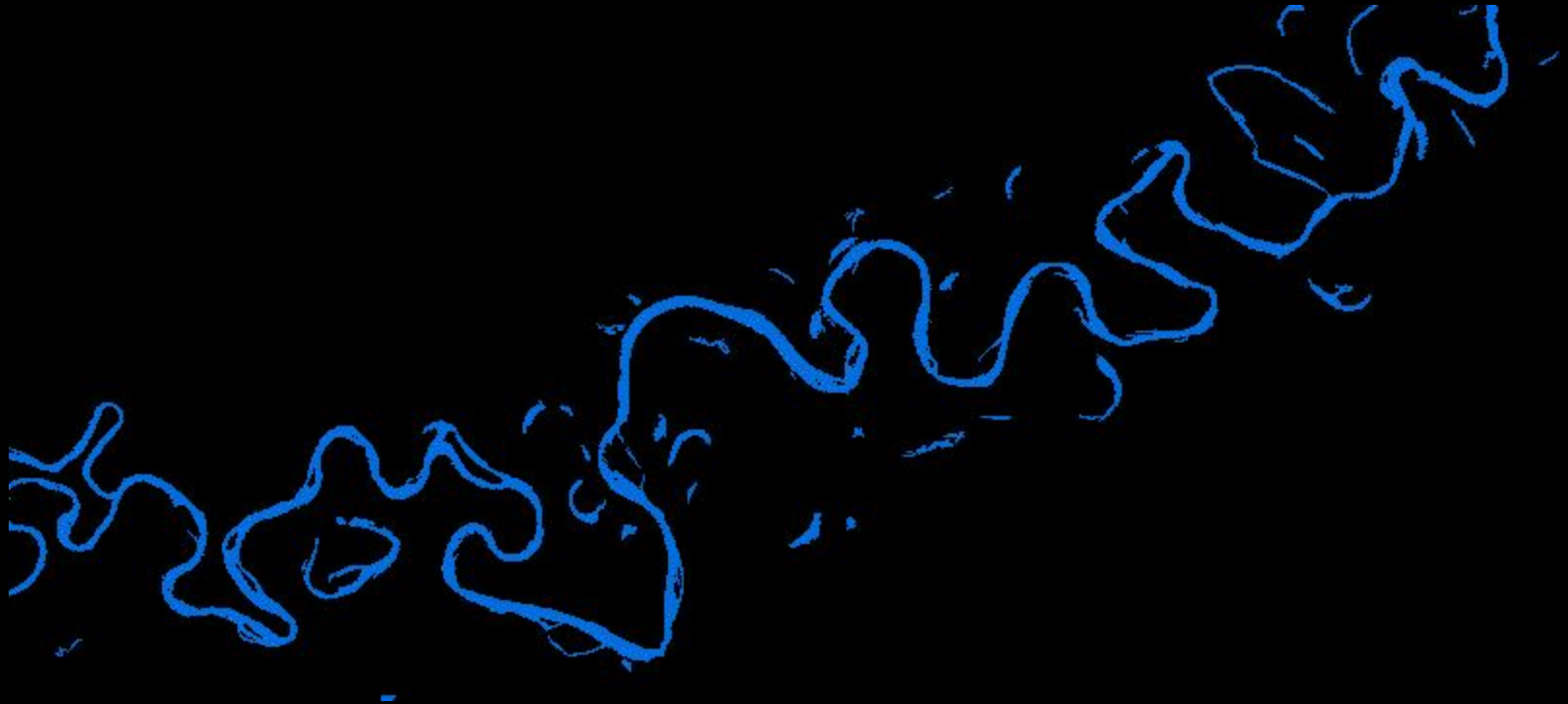
...but... for more control in *fancy* animations, **`map()`** a styling function over the collection yourself.

```
// Fetch a MODIS NDVI collection.
var col = ee.ImageCollection('MODIS/006/MOD13A2')
  .filterDate('2018-01-01', '2019-01-01');

// Define visualization parameters.
var visParams = {
  bands: ['NDVI'],
  min: 0,
  max: 9000,
  palette: ['white', 'green']
};

// Map a visualize function over the collection.
var visCol = col.map(function(img) {
  return img.visualize(visParams)
})
```

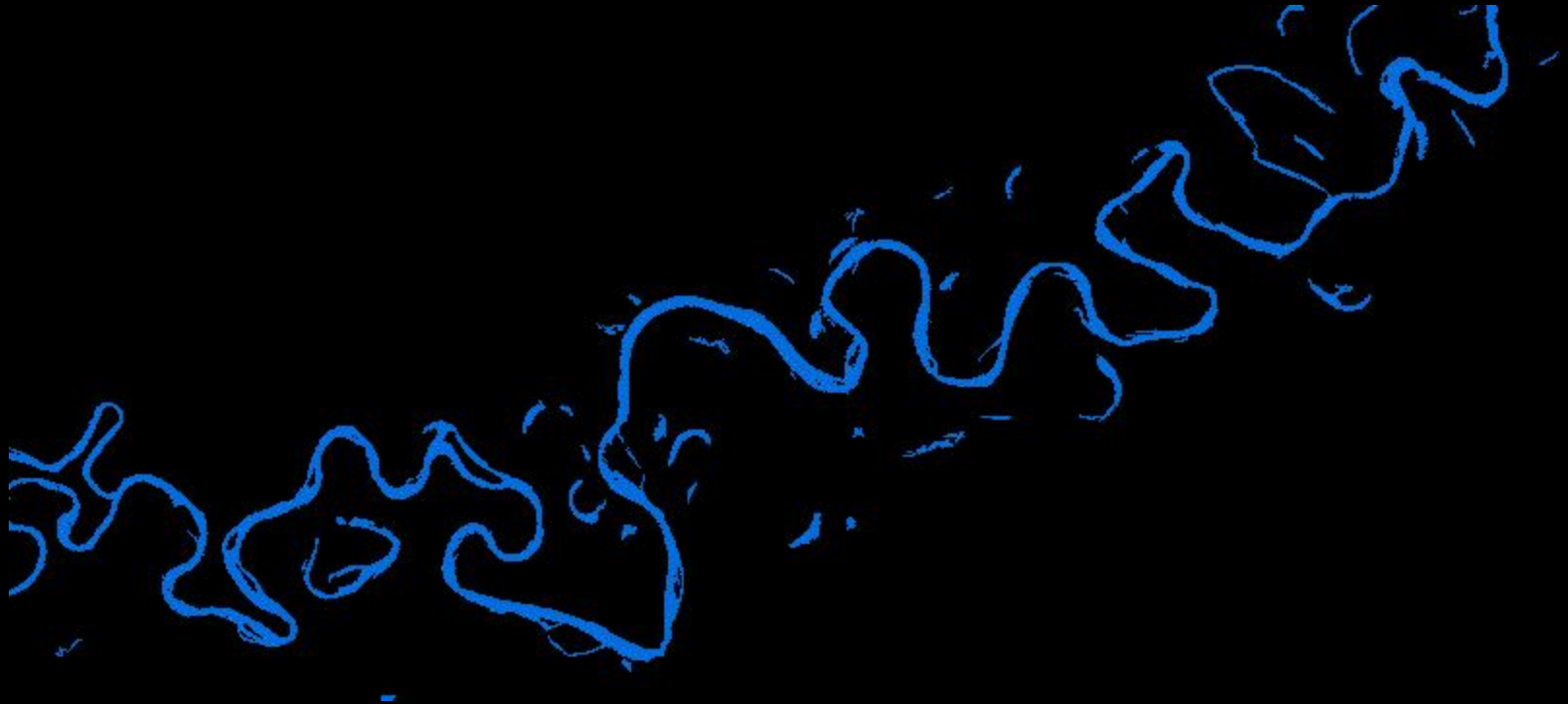
Use `ee.Image.blend` to add overlays

```
// Fetch a MODIS NDVI collection.
var col = ee.ImageCollection('MODIS/006/MOD13A2')
  .filterDate('2018-01-01', '2019-01-01');

// Make a hillshade layer.
var hillshade = ee.Terrain.hillshade(
  ee.Image('MERIT/DEM/v1_0_3').select('dem')
  .multiply(200));

// Define visualization parameters.
var visParams = {
  bands: ['NDVI'],
  min: 0,
  max: 9000,
  palette: ['white', 'green'],
  opacity: 0.7
};

// Map a visualize function over the collection.
var visCol = col.map(function(img) {
  return hillshade.blend(img.visualize(visParams));
});
```





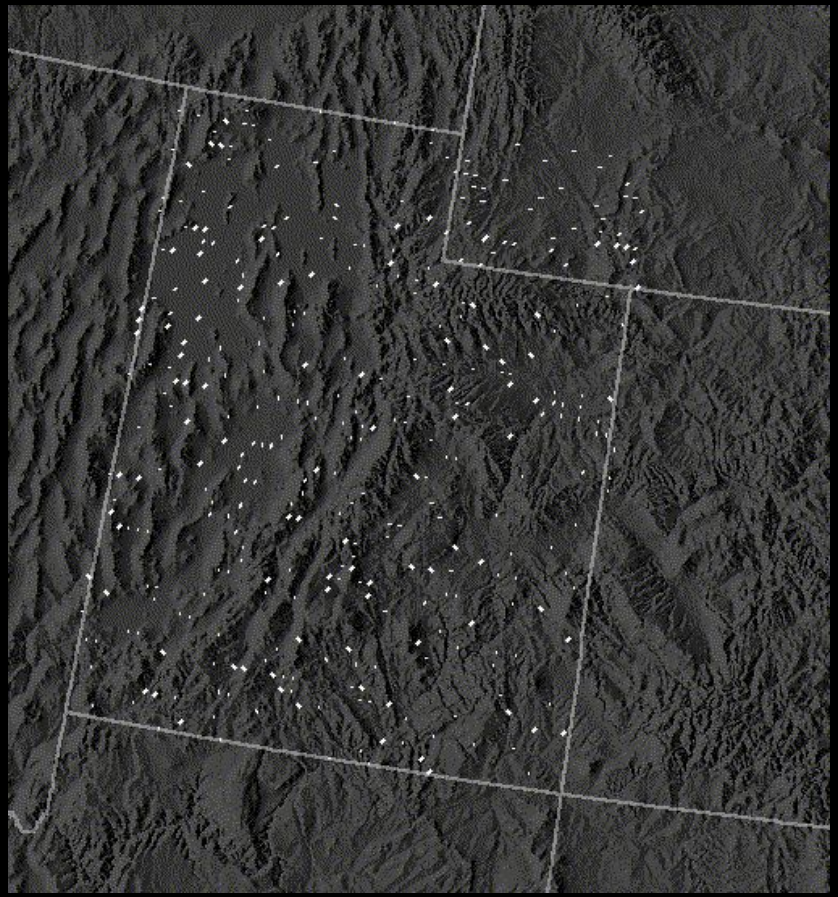
https://medium.com/google-earth/visualizing-changing-landscapes-with-google-earth-engine-b2d502dc02a8



https://medium.com/google-earth/visualizing-changing-landscapes-with-google-earth-engine-b2d502dc02a8

https://medium.com/google-earth/visualizing-changing-landscapes-with-google-earth-engine-b2d502dc02a8

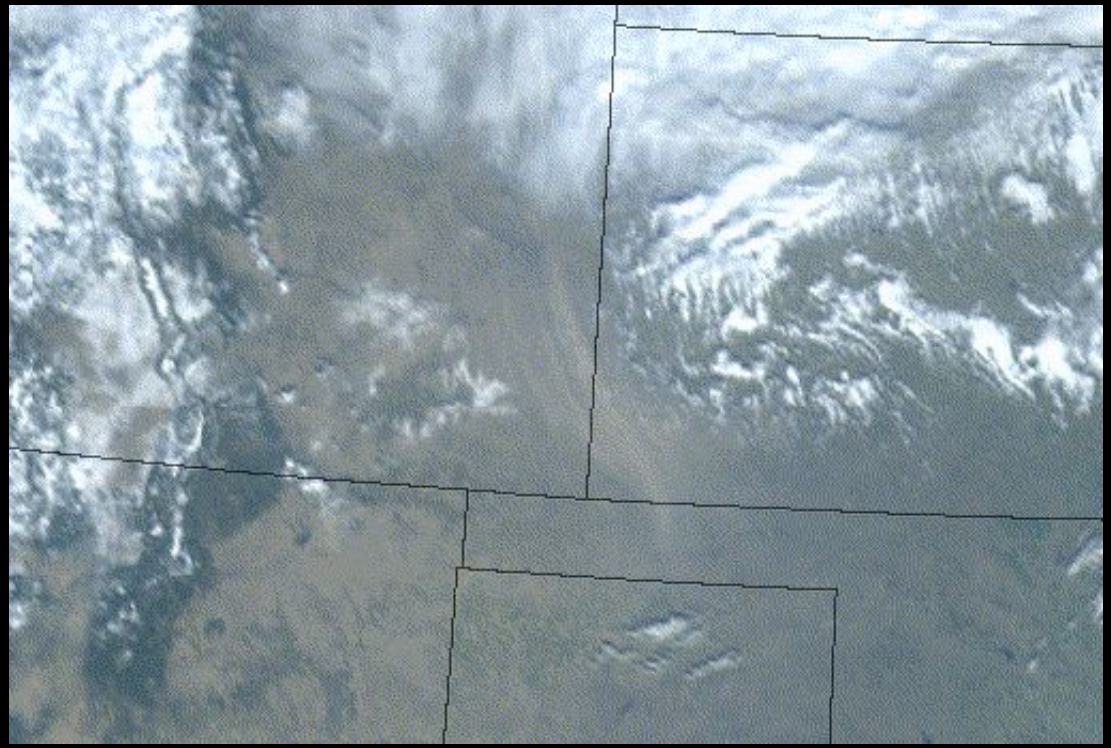
https://developers.google.com/earth-engine/guides/ic_visualization#fade



#GeoForGood22

https://gist.github.com/jdbcode/a6988f20f6f6dd8bc6d43f903c68e6e3

Use `ee.Image.translate` to shift each frame



https://code.earthengine.google.com/6ce73d17596703ee33d4ab7332da9a56

DEMO

https://gist.github.com/jdbcode/79814fb15c9832770761771772df9ab

Charts

https://developers.google.com/earth-engine/guides/charts_image_collection

Google Earth Engine

Search Language

Filter

Objects and Methods
Objects and Methods Overview
Image
ImageCollection
Geometry
Feature & FeatureCollection
FeatureView
Reducer
Join
Array
Chart
Chart Overview
Feature and FeatureCollection Charts
Image Charts
ImageCollection Charts
Array and List Charts
DataTable Charts
Chart Styling

Machine Learning
Overview of ML in Earth Engine
Supervised Classification Algorithms
Unsupervised Classification Algorithms
TensorFlow models
TensorFlow example workflows
TFRecord data format

Apps and User Interfaces
Earth Engine Apps
User Interface API Overview
Widgets
Panels and Layouts

Home > Products > Google Earth Engine > Guides

Was this helpful?

ImageCollection Charts

Send feedback

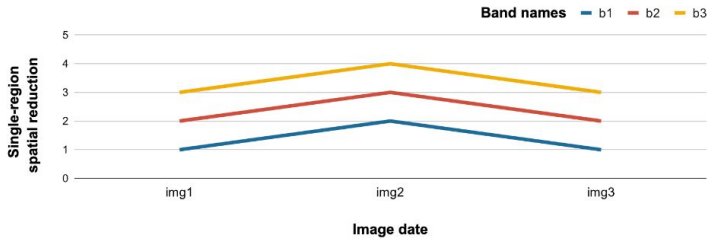
The `ui.Chart.image` module contains a set of functions for rendering charts from the results of spatiotemporal reduction of images within an `ImageCollection`. The choice of function dictates the arrangement of data in the chart, i.e., what defines x- and y-axis values and what defines the series. Use the following function descriptions and examples to determine the best function for your purpose.

Chart functions

Use the following plot diagrams as a visual guide to understand how each function arranges spatiotemporal image collection reduction results in a chart; i.e., what elements define x values, y values, and series. Note that `ui.Chart.image.doySeries*` functions take two reducers: one for region reduction (`regionReducer`) and another for intra-annual coincident day-of-year reduction (`yearReducer`). Examples in the following sections use `ee.Reducer.mean()` as the argument for both of these parameters.

`ui.Chart.image.series`

Image date is plotted along the x-axis according to the `system:time_start` property. Series are defined by image bands. Y-axis values are the reduction of images, by date, for a single region.

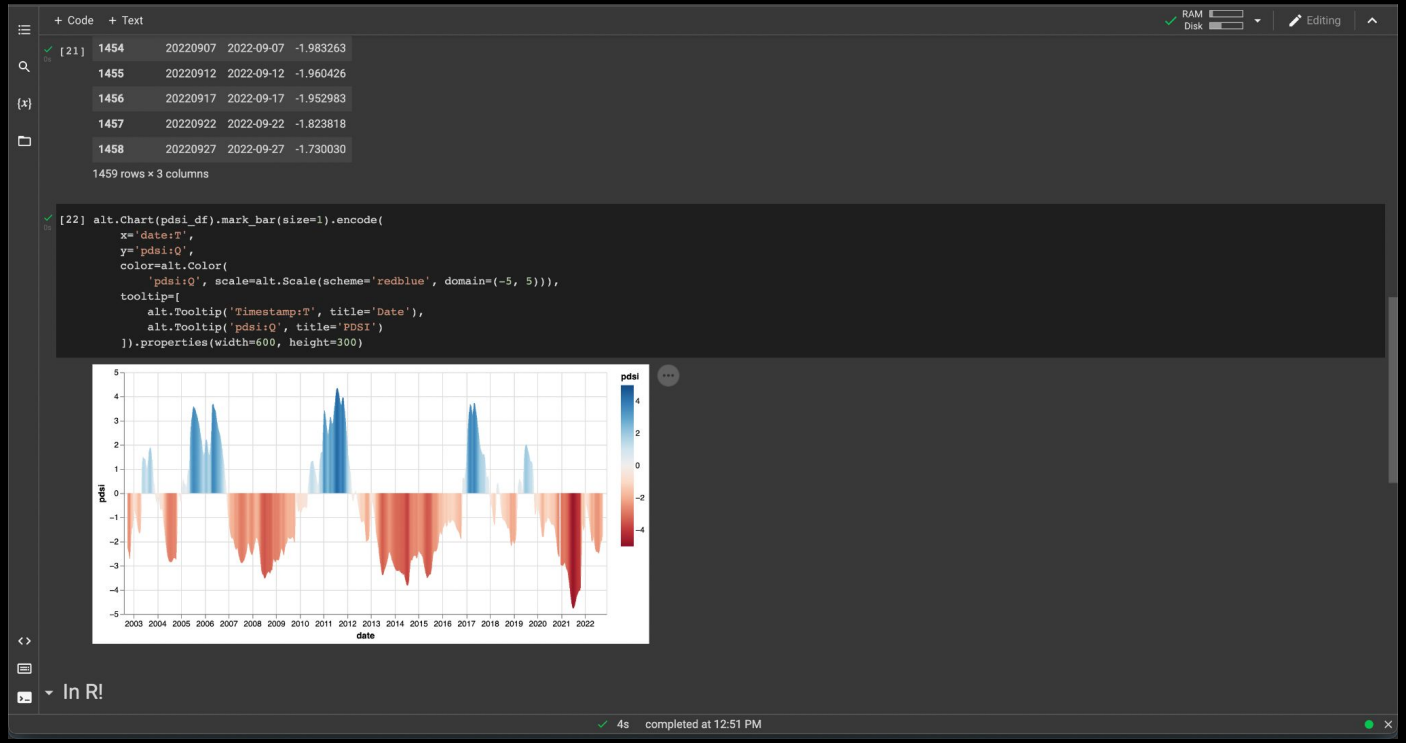


`ui.Chart.image.seriesByRegion`

On this page
Chart functions
Example data
ui.Chart.image.series
ui.Chart.image.seriesByRegion
ui.Chart.image.doySeries
ui.Chart.image.doySeriesByYear
ui.Chart.image.doySeriesByRegion

Recommended for you
Chart Styling
Updated May 26, 2021
Chart Overview
Updated Oct 18, 2021
Feature and FeatureCollection Charts
Updated May 27, 2021

DEMO



https://gist.github.com/jdbcode/490b567812db4c95bbe6a3c19fe30a55


Thank you!