

Google Earth Engine and machine learning for Earth monitoring

Alexander Uzhinskiy

Joint Institute for Nuclear Research, Joliot-Curie 6, 141980 Dubna, Russia, auzhinskiy@jinr.ru

auzhinskiy@jinr.ru; Tel.: +79057766865

Environmental Concerns

The environmental problems like global warming, acid rain, air pollution, urban sprawl, waste disposal, ozone layer depletion, water pollution, climate change and many more affect every human, animal, and nation on this planet.

Over the last few decades, the exploitation of our planet and the degradation of our environment has gone up at an alarming rate. As our actions have been not in favor of protecting this planet, we have seen natural disasters striking us more often in the form of flash floods, earthquakes, blizzards, tsunamis, and cyclones.

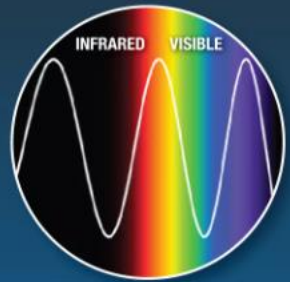


Satellite programs



Transforming Energy Into Imagery

Scans 22 swaths of Earth to create a full disk



Collects particles of light that are turned into a digital signal



Antenna on the ground receives the data



Combinations of red, green and blue create a representation of what the human eye would see from space

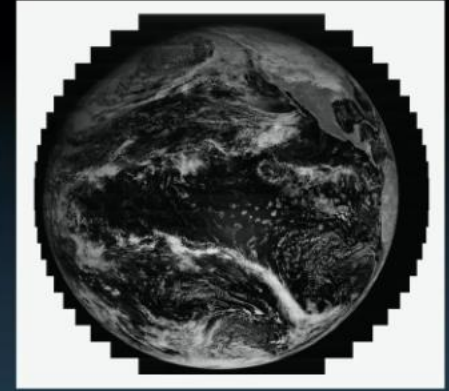
Color is assigned to light collected by the satellite, based on the portion of the electromagnetic spectrum the data represents

Data is sent to a processing center in binary code, a numeric language that uses only 0 and 1, arranged in eight-character strings

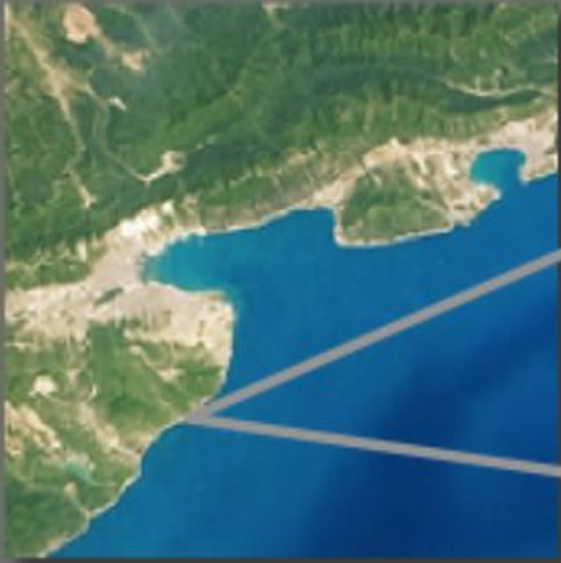


Computers translate the code into a collection of pixels that form a black and white image

Data is further processed and calibrated



Image

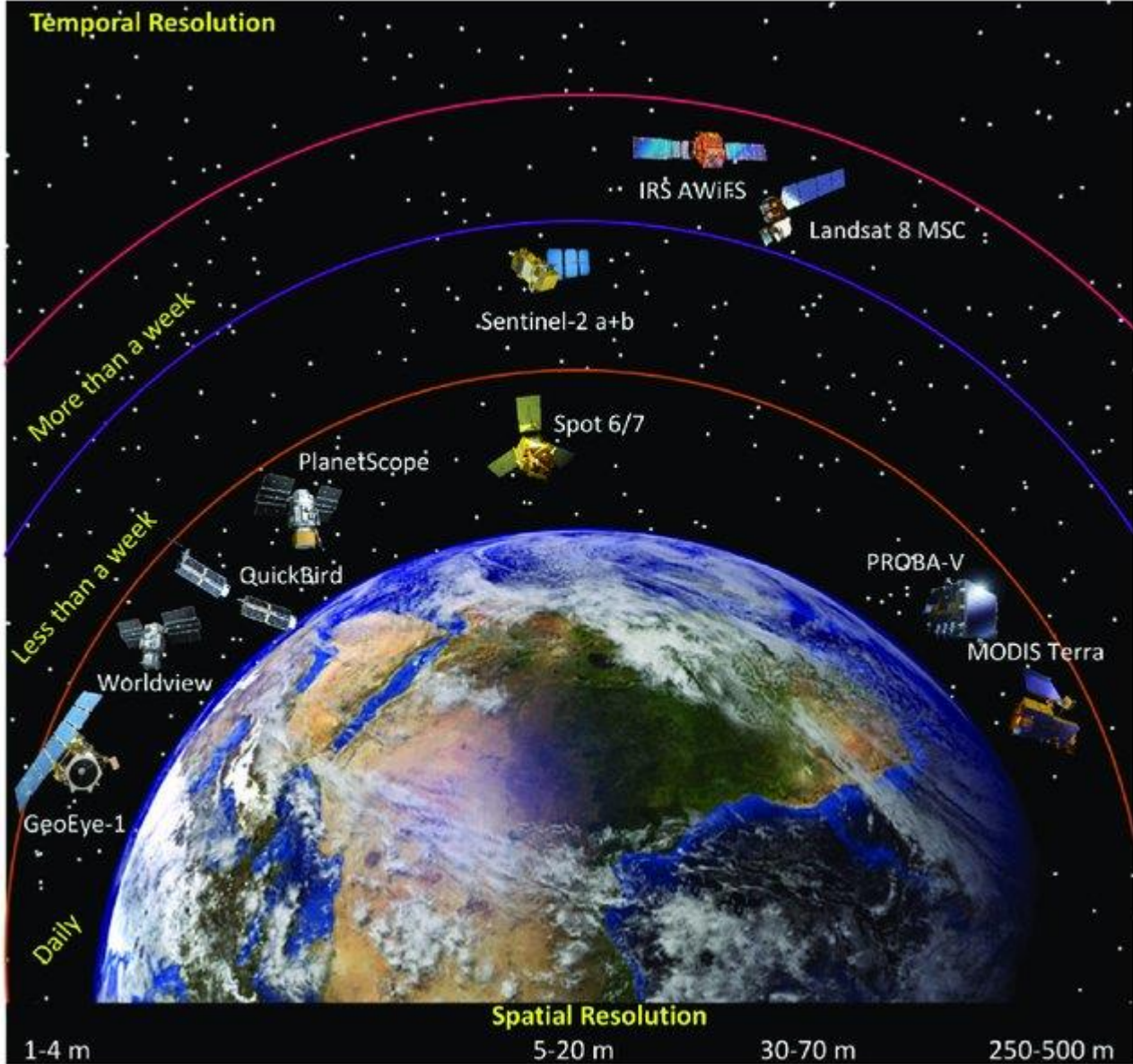


Pixels

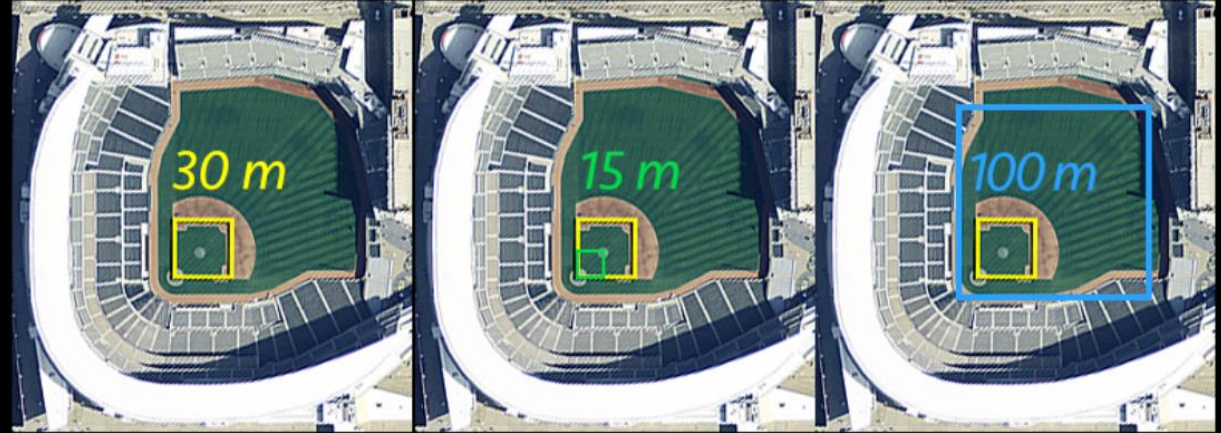


**Digital images are made of pixels,
and pixels are representations of
numeric values**





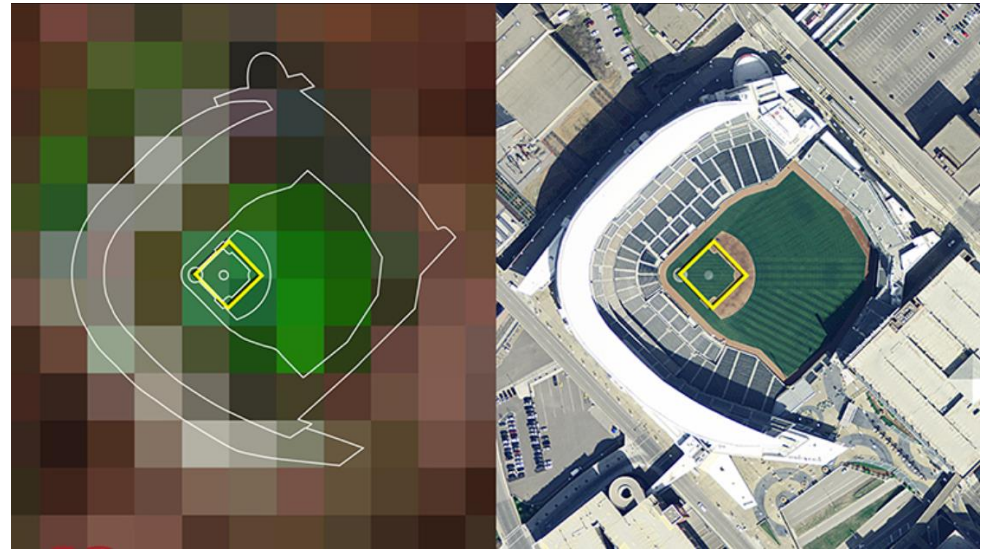
Landsat 8's Spatial Resolution



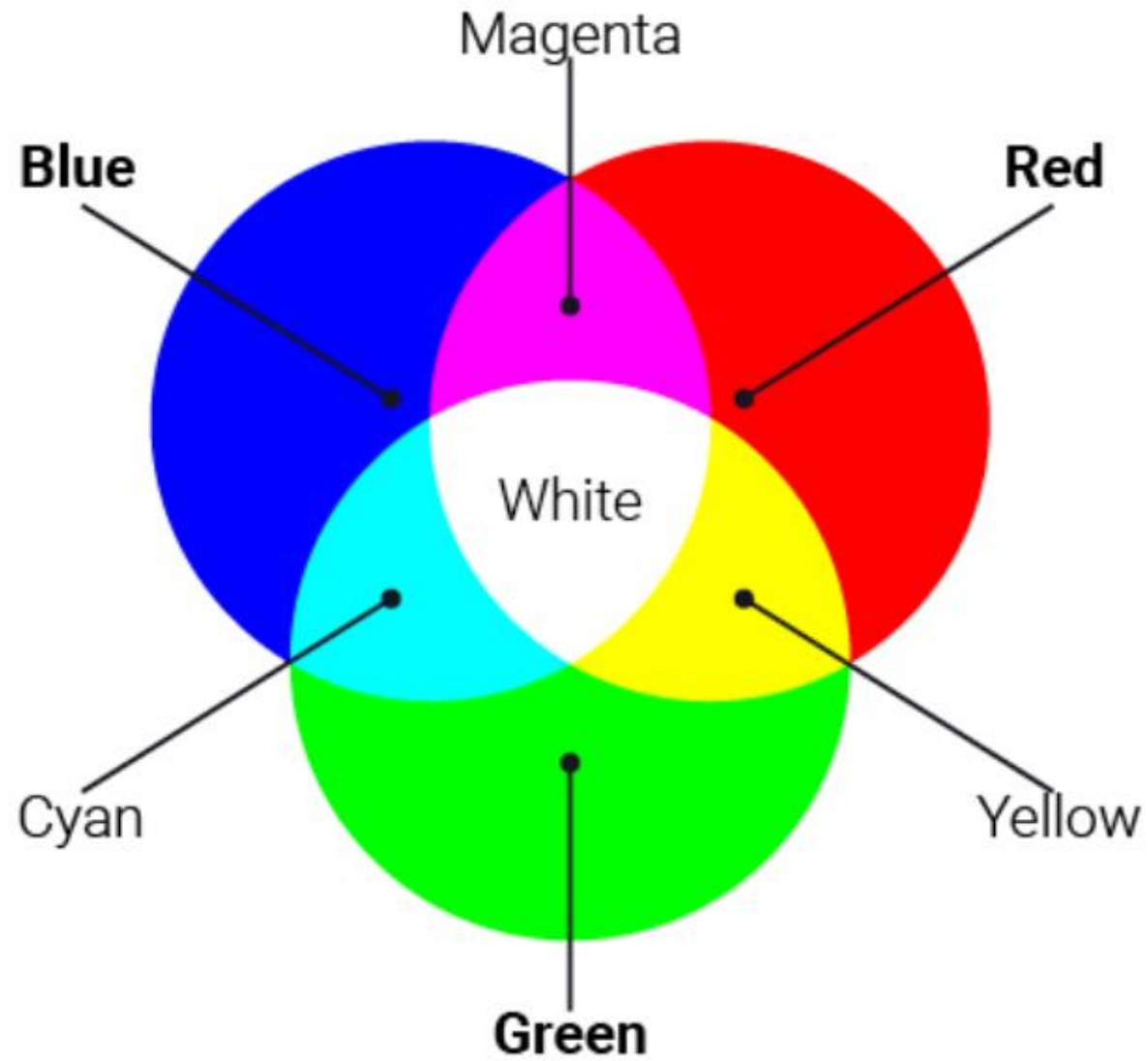
Vis-NIR-SWIR = 30 m

Panchromatic = 15 m

Thermal IR = 100 m
(Resampled to 30 m)

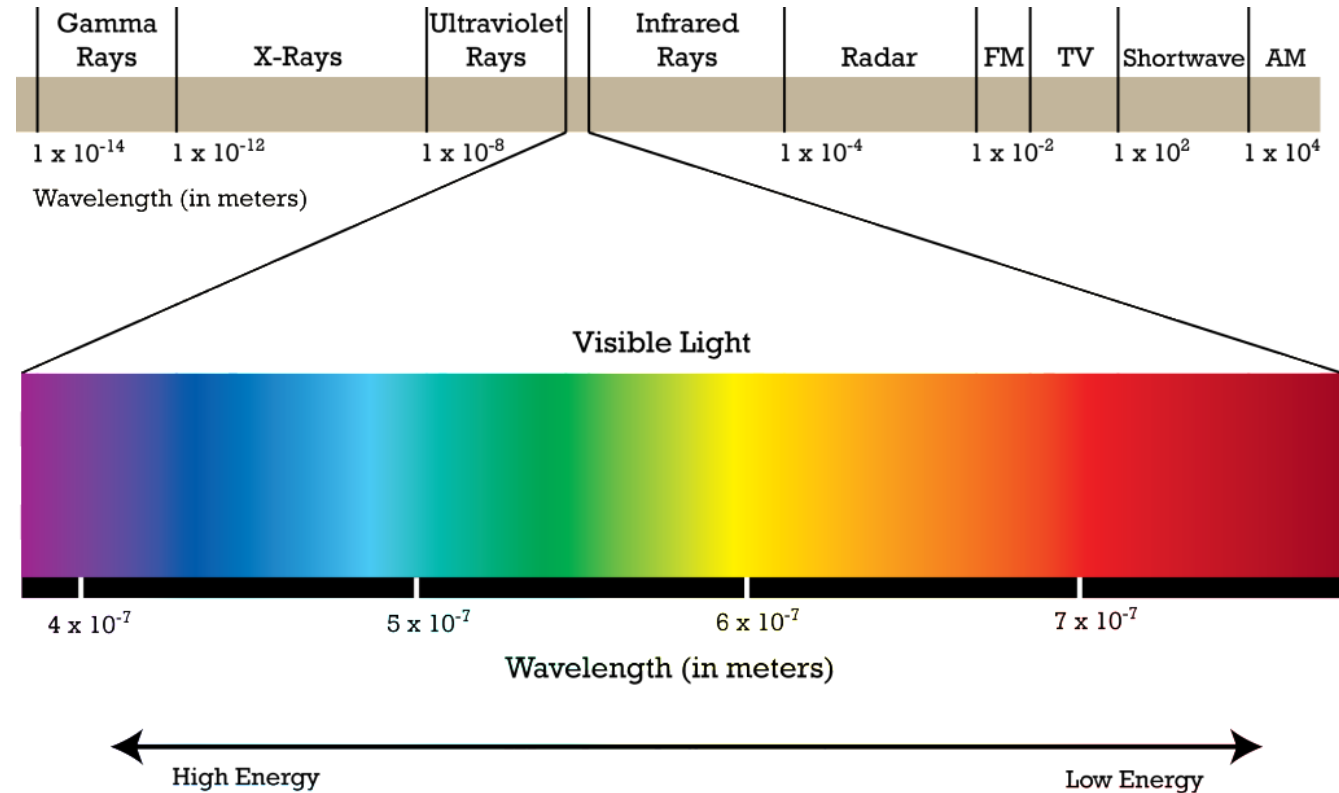


RGB



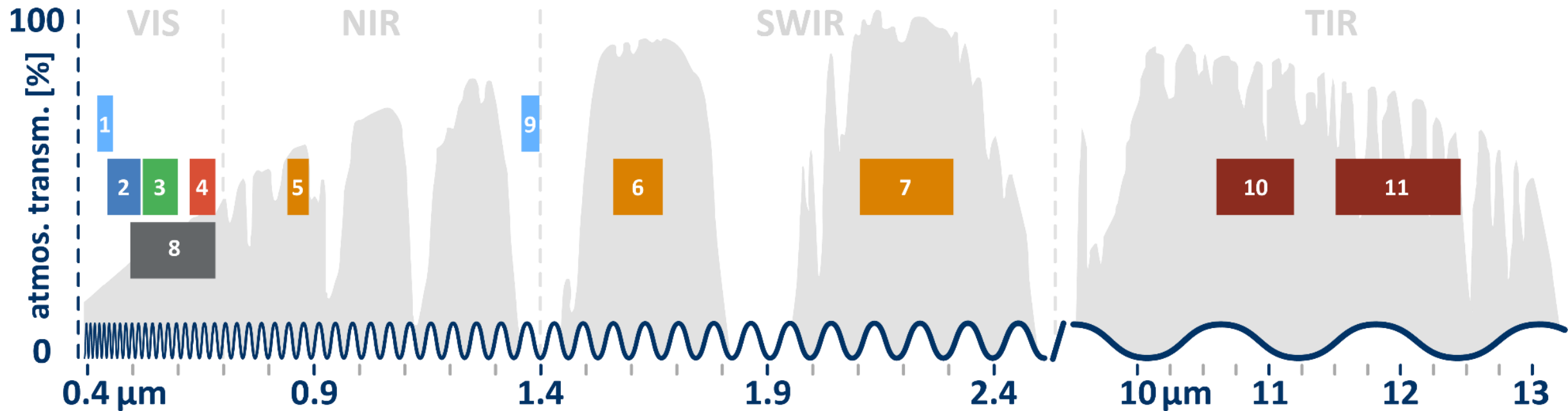
Bands

- RS sensors can collect data in all portions of the EM spectrum
- Multispectral sensors have spectral sensitivity limitations (spectral resolution)
 - The wavelength ranges recorded by sensors are called bands or “channels”, varies with sensor



Bands

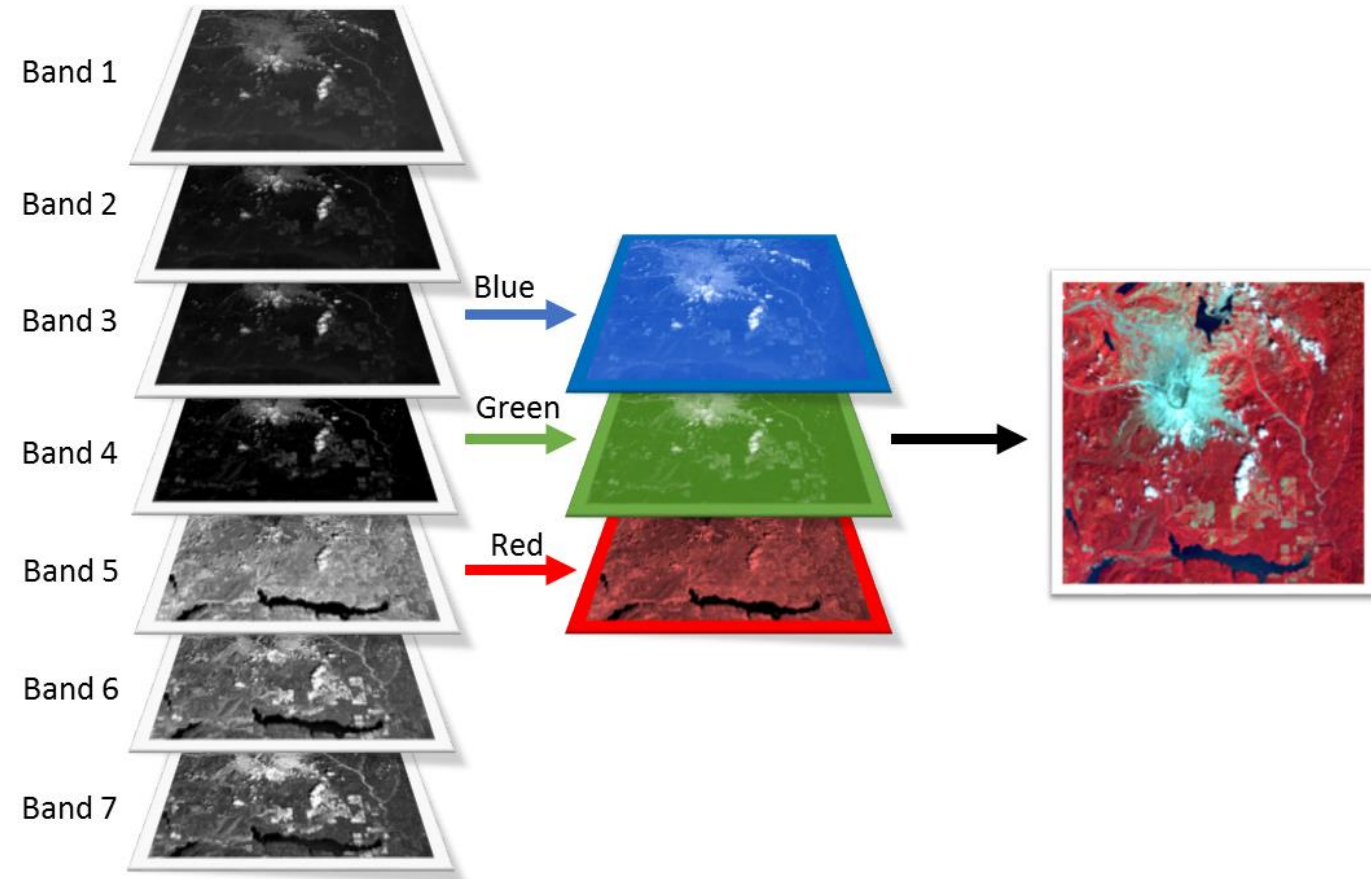
- Landsat collects data in 11 “channels” throughout the EM spectrum

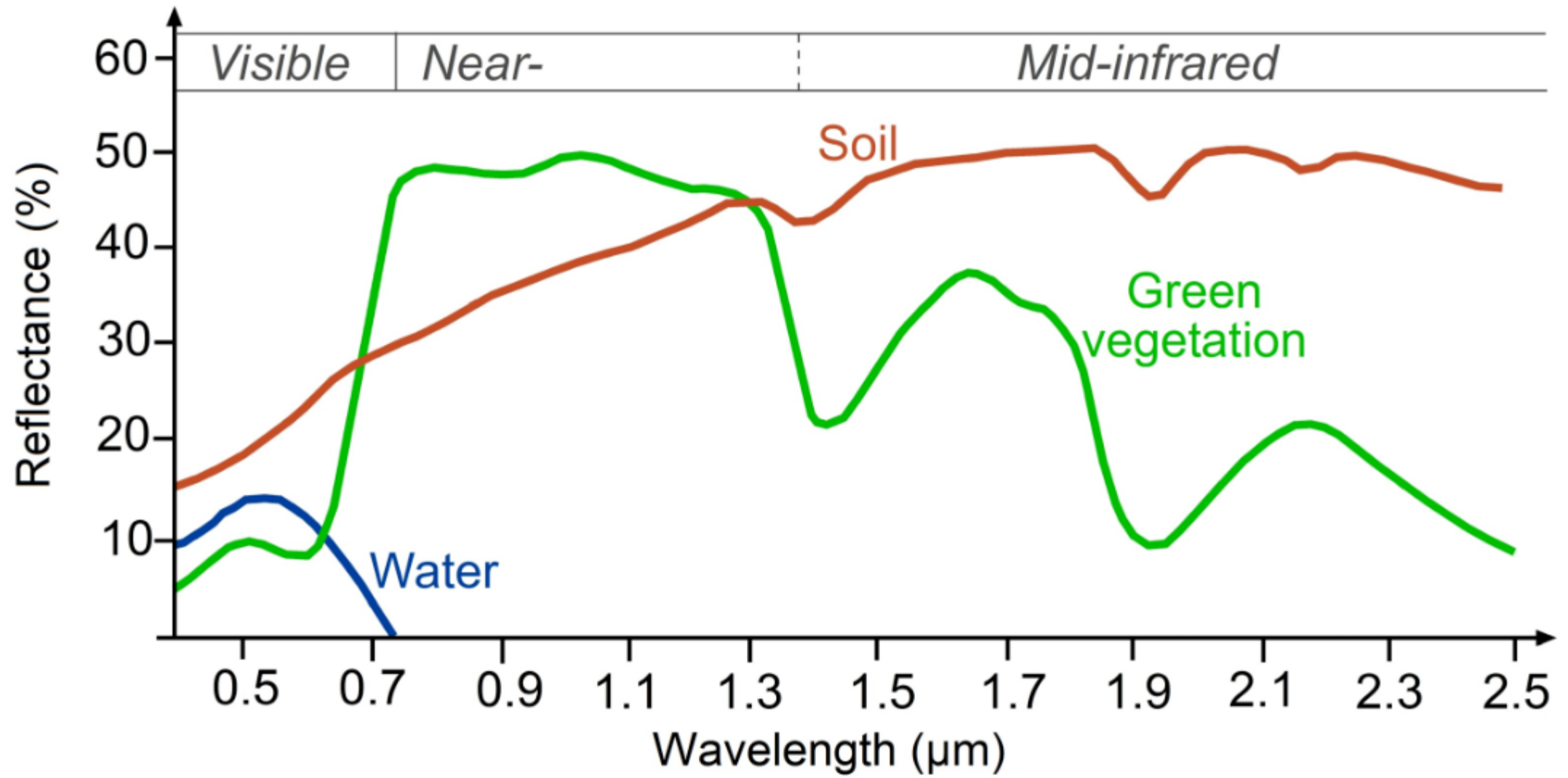


[Figure reference](#)

Band Combinations

- When you display a remote sensing image on a computer, you are limited to 3 bands
- This is what produces true color composites and false color composites

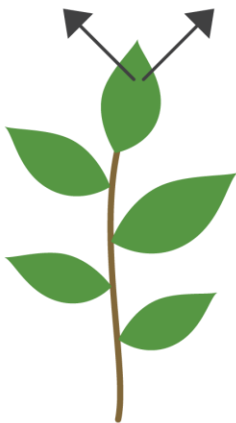




HEALTHY

VEGETATION REFLECTANCE

50% NIR 8% RED



NDVI = 0.72

STRESSED

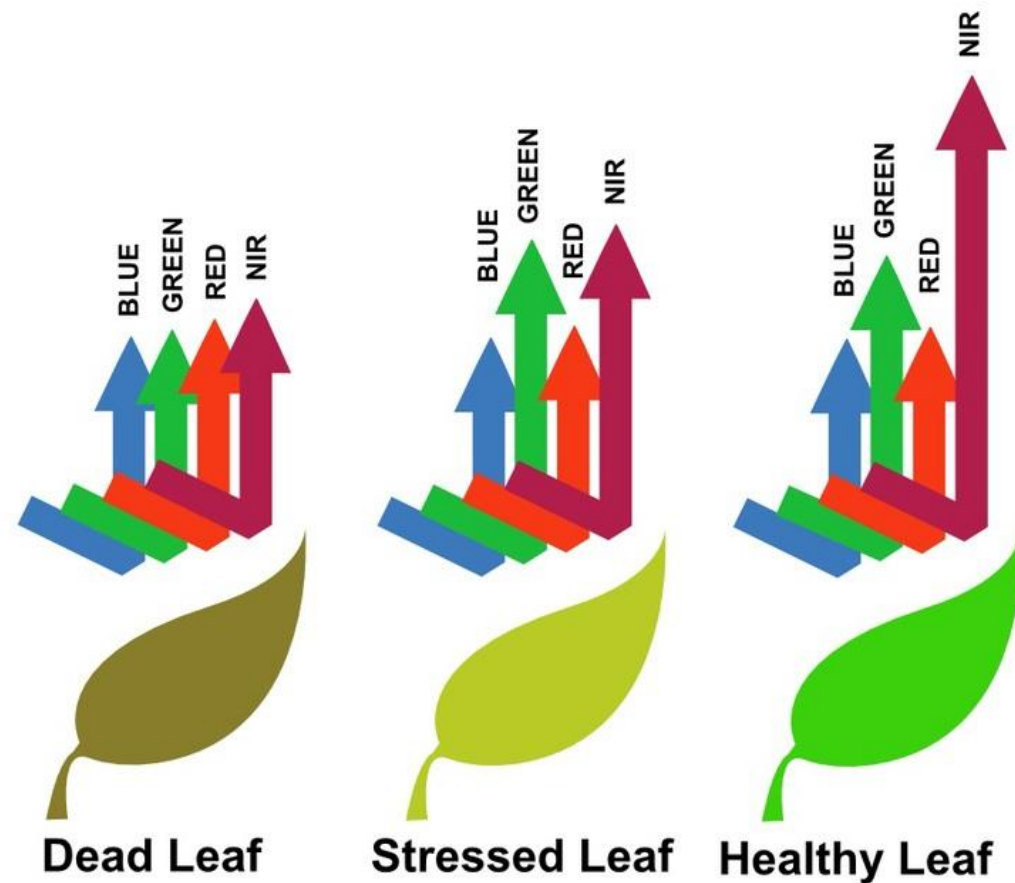
VEGETATION REFLECTANCE

40% NIR 30% RED



NDVI = 0.14

$$\text{NDVI} = \frac{\text{NIR} - \text{RED}}{\text{NIR} + \text{RED}}$$



Old fashion approach

Task: Map all change between 2000 – 2010 over a specific forest

Extent of change

Year of occurrence

Solution:

- Select Area of Interest
- Find WRS path/row(s) – assume an area of 4 scenes

Data Prep:

- Download and store all Landsat during growing season (1 GB / zipped scene)
- Extract and layer stack all Landsat (1.75 GB / scene)
- ~48 scenes per year * 11 years = ~528 scenes, or 924 GB
- Apply atmospheric correction (924 more GB)
- Normalize scenes
- Apply FMASK or similar to remove clouds and shadows
- Create composite and mosaic by year
- Generate vegetation index per year (NDVI and/or NBR)

Analysis:

- Build spatial model in ERDAS to compare year pairs
- Generate change layer
- Classify pixels > certain value as “change”
- Build spatial model to apply year attribute to each pixel
- Repeat for each year pair
- Build spatial model to stack all attributed change layers into a single raster image, with the most recently changed pixel on top
- Apply color ramp visually demonstrating change

GEE approach

The screenshot displays the Google Earth Engine (GEE) interface. At the top, there is a search bar and navigation options. The main workspace is divided into three panels: Scripts, Compositing-method, and Inspector/Console. The Compositing-method panel contains the following JavaScript code:

```
30 ////////////////////////////////////////////////////
31 //Function to mask clouds
32 function bustClouds(img){
33   var cloudScore = ee.Algorithms.Landsat.simpleCloudScore(img).select(['cl
34
35   var notCloudyPixels = cloudScore.lte(cloudThresh).focal_min(cloudBuffer)
36   var pixelsInAllBands = img.mask().reduce(ee.Reducer.min());
37
38
```

The Inspector/Console panel shows the output of the script, displaying a list of 7 band names in JSON format. The main map area shows a visualization of landscape change, with a color scale ranging from 2000 (yellow) to 2010 (red). A 'Change Year visualization parameters' dialog box is open, showing options for 1 band (Grayscale) or 3 bands (RGB), a range from 2000 to 2010, and a palette selection.

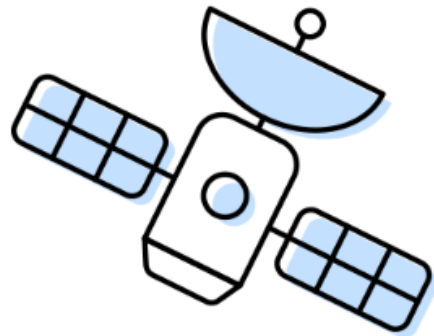
The new method took an experienced geospatial programmer about an hour and ~100 lines of code to generate a raster layer showing the extent of landscape change thematically colored by year of change (in this case, yellow changed closer to the year 2000 and red colors changed closer to 2010).

Users can change the AOI very simply and run this same process anywhere in the world, then export the results to a raster TIFF image.

This represents an extremely dramatic improvement in efficiency. In fact, it allows us to ask new questions.

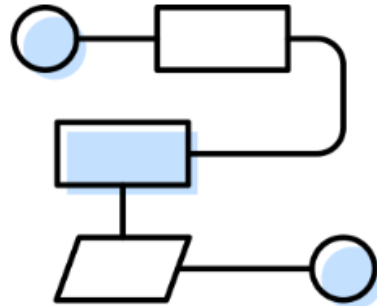
Google Earth Engine

Google Earth Engine combines a multi-petabyte catalog of satellite imagery and geospatial datasets with planetary-scale analysis capabilities. Scientists, researchers, and developers use Earth Engine to detect changes, map trends, and quantify differences on the Earth's surface. Earth Engine is now available for commercial use, and remains free for academic and research use.



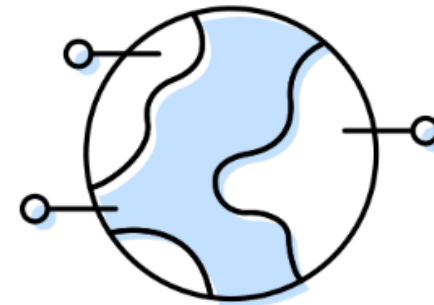
Satellite Imagery

+



Your Algorithms

+



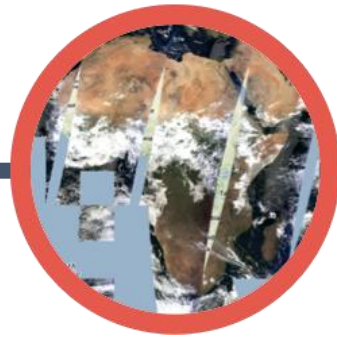
Real World Applications

Earth Engine provides easy, web-based access to an extensive catalog of satellite imagery and other geospatial data in an analysis-ready format. The data catalog is paired with scalable compute power backed by Google data centers and flexible APIs that let you seamlessly implement your existing geospatial workflows. This enables cutting-edge, global scale analysis and visualization.

The Earth Engine Data Catalog



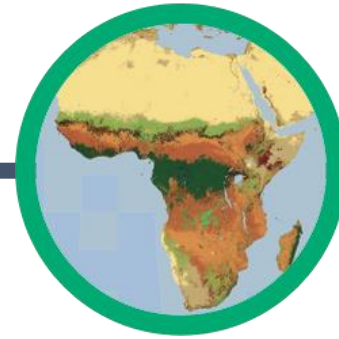
**Landsat & Sentinel 1,
2**
10-30m, weekly



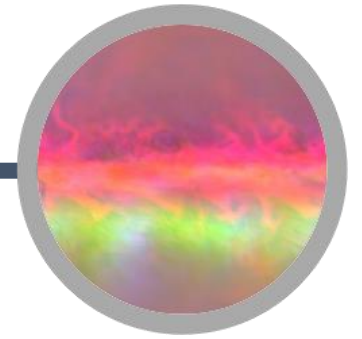
MODIS
250m daily



Vector Data
WDPA, Tiger



**Terrain &
Land Cover**



Weather & Climate
NOAA NCEP, OMI, ...

**... and upload your own vectors and
rasters**

> 200 public datasets

> 5 million images

> 4000 new images every day

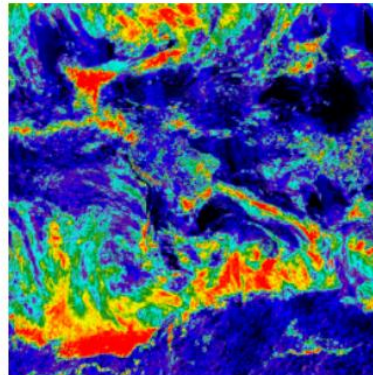
> 60 petabytes of data

The Earth Engine Data Catalog

Datasets tagged climate in Earth Engine

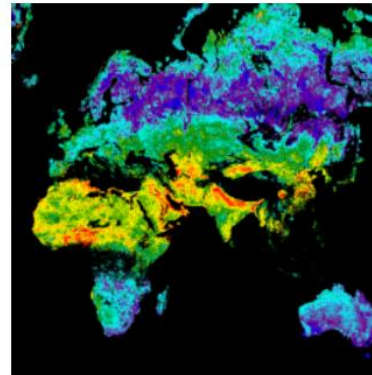
Filter list of datasets

Sentinel-5P NRTI CLOUD: Near Real-Time Cloud



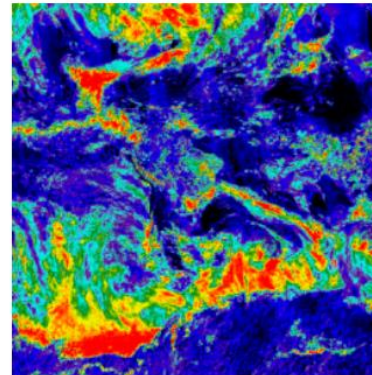
NRTI/L3_CLOUD This dataset provides near real-time high-resolution imagery of cloud parameters. The TROPOMI/S5P cloud properties retrieval is based on the OCRA and ROCINN algorithms currently being used in the operational GOME and GOME-2 products. OCRA retrieves the cloud fraction using measurements in the UV/VIS spectral ...

Sentinel-5P OFFL CH4: Offline Methane



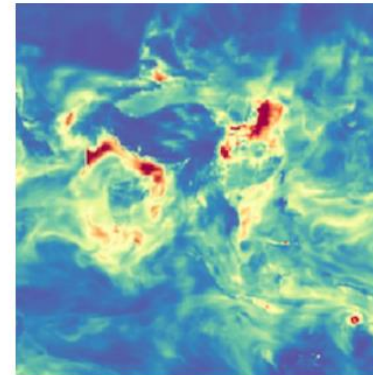
OFFL/L3_CH4 This dataset provides offline high-resolution imagery of methane concentrations. Methane (CH4) is, after carbon dioxide (CO2), the most important contributor to the anthropogenically enhanced greenhouse effect. Roughly three-quarters of methane emissions are anthropogenic and as such it is important to continue

Sentinel-5P OFFL CLOUD: Near Real-Time Cloud



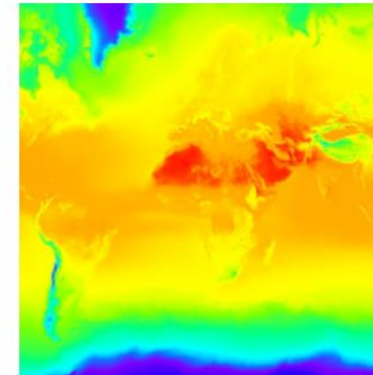
OFFL/L3_CLOUD This dataset provides offline high-resolution imagery of cloud parameters. The TROPOMI/S5P cloud properties retrieval is based on the OCRA and ROCINN algorithms currently being used in the operational GOME and GOME-2 products. OCRA retrieves the cloud fraction using measurements in the UV/VIS spectral regions ...

Copernicus Atmosphere Monitoring Service (CAMS) Global Near-Real-Time



The Copernicus Atmosphere Monitoring Service provides the capacity to continuously monitor the composition of the Earth's atmosphere at global and regional scales. The main global near-real-time production system is a data assimilation and forecasting suite providing two 5-day forecasts per day for aerosols and chemical ...

ERA5 Daily Aggregates - Latest Climate Reanalysis Produced by ECMWF / Copernicus Climate



ERA5 is the fifth generation ECMWF atmospheric reanalysis of the global climate. Reanalysis combines model data with observations from across the world into a globally complete and consistent dataset. ERA5 replaces its predecessor, the ERA-Interim reanalysis. ERA5 DAILY provides aggregated values for each day for ...

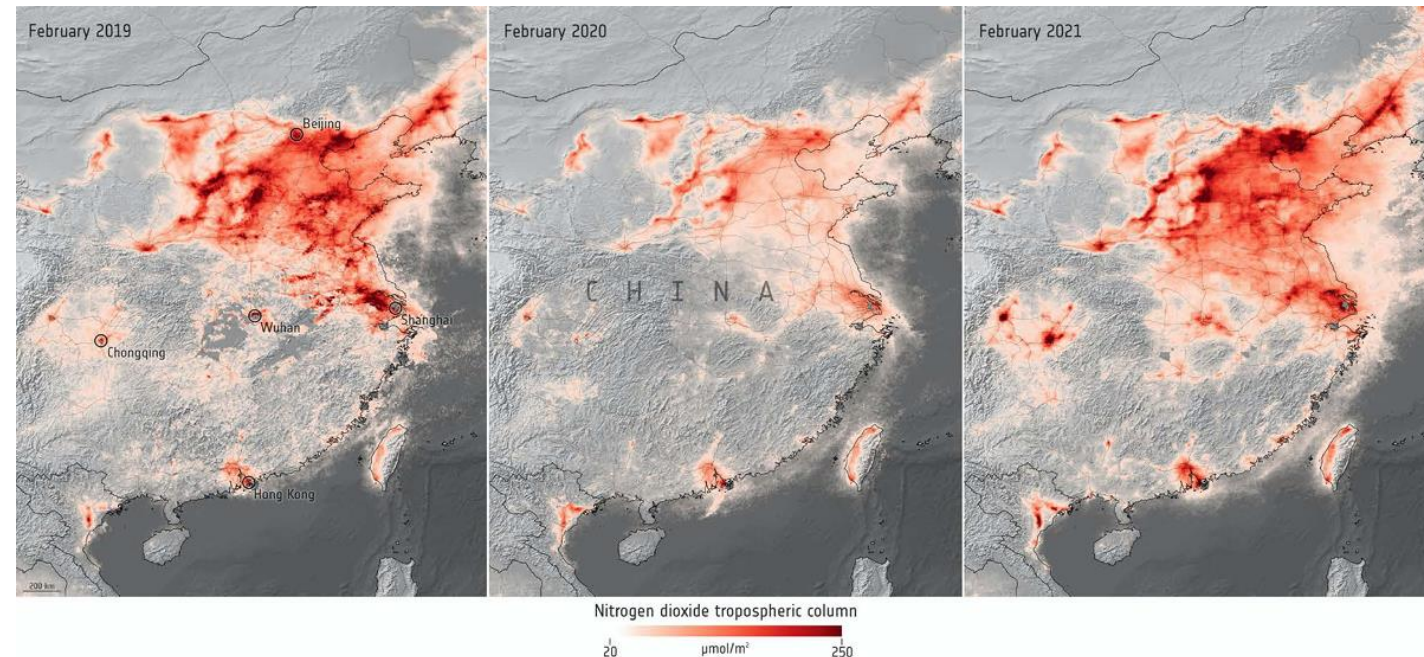
Sentinel-5



Sentinel-5 is focused on air quality and composition-climate interaction with the main data products being O_3 , NO_2 , SO_2 , HCHO, CHOCHO and aerosols. Additionally Sentinel-5 will also deliver quality parameters for CO , CH_4 , and stratospheric O_3 with daily global coverage for climate, air quality, and ozone/surface UV applications.

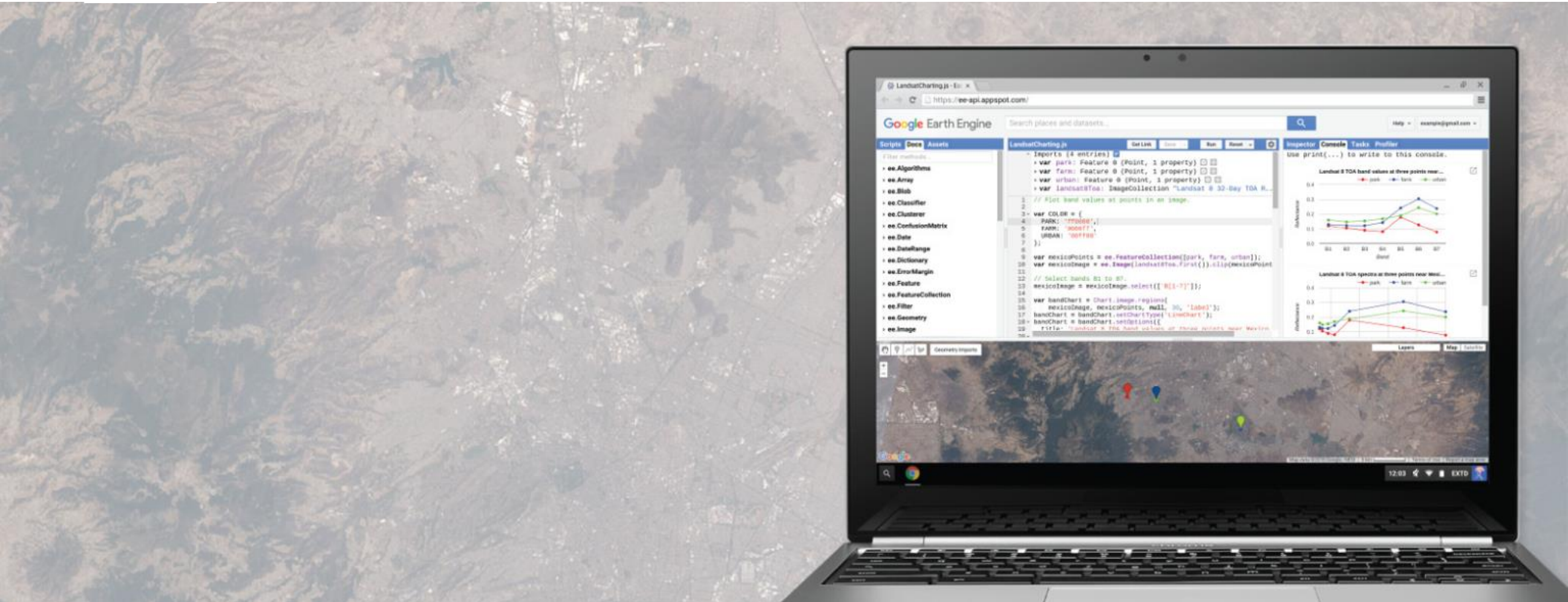
The Sentinel-5 mission consists of high resolution spectrometer system operating in the ultraviolet to shortwave infrared range with 7 different spectral bands: UV-1 (270-300nm), UV-2 (300-370nm), VIS (370-500nm), NIR-1 (685-710nm), NIR-2 (745-773nm), SWIR-1 (1590-1675nm) and SWIR-3 (2305-2385nm).

The Sentinel-5 mission is part of the European Earth Observation Programme "Copernicus" which is a coordinated and managed by the European Commission (EC). The space component of the Copernicus observation infrastructure is developed under the aegis of the European Space Agency (ESA).



contains modified Copernicus Sentinel data (2019-21), processed by ESA

JavaScript code editor <https://code.earthengine.google.com/>



* python too!

Example of map

The screenshot displays the Google Earth Engine web interface. At the top left, the "Google Earth Engine" logo is visible. A search bar contains the text "Search places and datasets...". Below the search bar, there are tabs for "Scripts", "Docs", and "Assets". The "Scripts" tab is active, showing a file tree on the left with folders like "ndvi", "ndvi_calculation", "razm2", "razmeri", "romania", "tutorial", "tutorial (copy)", and "vl". The "tutorial (copy)" folder is selected, and its contents are shown in the main editor area. The script code is as follows:

```
3  
4 Map.setCenter(37.16, 56.73);  
5 Map.setZoom(11);  
6  
7  
8  
9  
10  
11  
12 function maskL8sr(image) {  
13   // Bit 0 - Fill  
14   // Bit 1 - ...
```

Below the script editor, there is a "Writer" section. The main part of the interface is a satellite map showing a forested area with a river. The map includes navigation controls on the left (hand, location pin, zoom in/out, and a square icon) and a "Layers" panel on the right with buttons for "Layers", "Карта", and "Спутник". The bottom of the map shows the Google logo, keyboard shortcuts, copyright information "Картографические данные © 2022 Google", a scale bar for "2 км", and a link to "Условия использования".

Cloud free map

The screenshot displays the Google Earth Engine web interface. At the top left is the "Google Earth Engine" logo and a search bar. Below the logo is a navigation menu with "Scripts", "Docs", and "Assets". The "Assets" panel on the left shows a folder structure: "experiment" containing "Ivanovo", "Poland", "Sweden", "ndvi", "ndvi_calculation", and "razm2". The main script editor shows a file named "tutorial *" with the following code:

```
16 var qaMask = image.select('QA_PIXEL').bitwiseAnd(parseInt('11111', 2)).eq(0);
17 var saturationMask = image.select('QA_RADSAT').eq(0);
18
19 // Apply the scaling factors to the appropriate bands.
20 var opticalBands = image.select('SR_B.*').multiply(0.0000275).add(-0.2);
21 var thermalBands = image.select('ST_B.*').multiply(0.00341802).add(149.0);
22
23 // Replace the original bands with the scaled ones and apply the masks.
24 return image.addBands(opticalBands, thermalBands, qaMask, saturationMask, true);
```

Below the script editor is a satellite map of a forested area with a river. The map includes a toolbar on the left with navigation icons and a zoom control. On the right side of the map, there are buttons for "Layers", "Карта", and "Спутник". The bottom of the interface features a status bar with the Google logo, keyboard shortcuts, copyright information "Картографические данные © 2022 Google", a scale bar for "1 км", and a link to "Условия использования".

Data Types and Geospatial Processing Functions

- **Image** - band math, clip, convolution, neighborhood, selection ...
- **Image Collection** - map, aggregate, filter, mosaic, sort ...
- **Feature** - buffer, centroid, intersection, union, transform ...
- **Feature Collection** - aggregate, filter, flatten, merge, sort ...
- **Filter** - by bounds, within distance, date, day-of-year, metadata ...
- **Reducer** - mean, linearRegression, percentile, histogram ...
- **Join** - simple, inner, outer, inverted ...
- **Kernel** - square, circle, gaussian, sobel, kirsch ...
- **Machine Learning** - CART, random forests, bayes, SVM, kmeans, cobweb ...
- **Projection** - transform, translate, scale ...

over 1000 data types and operators, and growing!

Reduce

Aggregate everything in a collection

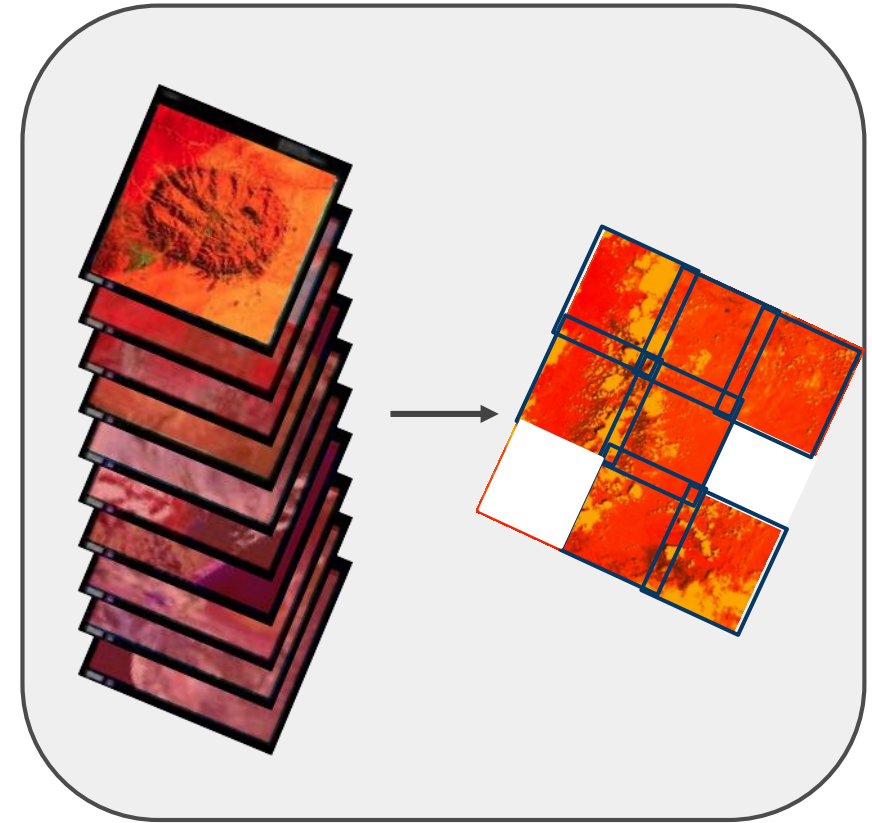
"Reduction"

Examples

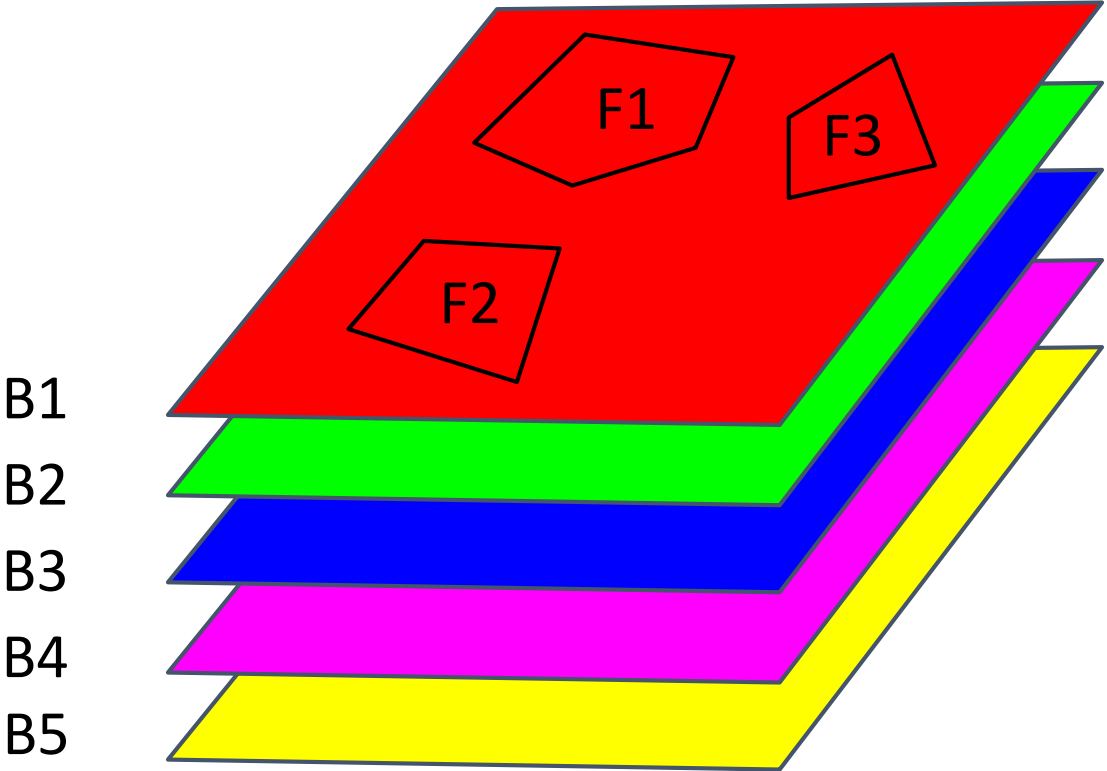
Summed area over all features

Median-pixel composite

Train a classifier



Reduce Regions



FeatureCollection

	B1	B2	B3	B4	B5
F1					
F2					
F3					

Reducer in action

The screenshot displays the Google Earth Engine web interface. The top navigation bar includes the Google Earth Engine logo, a search bar, and user profile icons. The left sidebar shows a project tree with folders like 'tutorial' and 'Writer'. The main editor area contains a script titled 'tutorial *' with the following code:

```
38
39 // Display the results.
40 Map.addLayer(composite, {bands: ['SR_B4', 'SR_B3', 'SR_B2'], min: 0, max: 0.3});
41
42 var region = ee.Geometry.Rectangle(37.16, 56.73, 37.17, 56.735);
43 Map.addLayer(region, {'color': 'red'});
44
45 var mean = composite.reduceRegion({
46   reducer: ee.Reducer.mean(),
47   geometry: region,
48   scale: 30
49 });
```

The Inspector panel on the right shows the output of the script, which is an Object with 19 properties:

```
Object (19 properties)
  QA_PIXEL: 21824
  QA_RADSAT: 0
  SR_B1: 0.029713583044238552
  SR_B2: 0.03616107042660685
  SR_B3: 0.060925221691503134
  SR_B4: 0.05429918944533672
  SR_B5: 0.24582152299513438
  SR_B6: 0.14452542855088027
  SR_B7: 0.09044999080735738
```

The main map area shows a satellite view of a landscape with a red square highlighting a specific region. The bottom of the interface includes a Google logo, a scale bar (1 km), and a copyright notice for 2022 Google.

Editor options

The screenshot displays a web-based GIS editor interface. The top navigation bar includes 'Scripts', 'Docs', and 'Assets'. A sidebar on the left lists various tools and datasets. The main workspace is divided into three panels: a code editor, a map, and an inspector/console.

Code Editor: The code is titled 'Seasonal Temperatures' and contains the following JavaScript code:

```
1 // Plot average seasonal temperatures in US States.
2
3 // Import US state boundaries.
4 var states = ee.FeatureCollection('TIGER/2018/States');
5
6 // Import temperature normals and convert month features to bands.
7 var normClim = ee.ImageCollection('OREGONSTATE/PRISM/Norm81m')
8   .select(['tmean'])
9   .toBands();
10
11 // Calculate mean monthly temperature per state.
12 states = normClim.reduceRegions({
13   collection: states,
14   reducer: ee.Reducer.mean(),
15   scale: 5e4})
16   .filter(ee.Filter.notNull(['01_tmean']));
17
18 // Calculate Jan to Jul temperature difference per state and set as a property.
19 states = states.map(function(state) {
```

Map: The map shows a portion of the United States, including California, Oregon, and parts of Nevada and Idaho. Major cities like San Francisco, San Jose, Stockton, and Modesto are labeled in both Russian and English. The map includes standard navigation controls like zoom in (+) and zoom out (-) buttons.

Inspector/Console: The console shows the instruction 'Use print(...) to write to this console.' Below it, a chart titled 'Average Temperatures in U.S. States' is displayed. The chart is a grouped bar chart comparing the average monthly temperatures for Minnesota (blue bars) and Texas (orange bars) in January, April, July, and October. The y-axis represents 'Temperature (Celsius)' ranging from -20 to 40. The x-axis represents 'Month'.

Month	Minnesota (Celsius)	Texas (Celsius)
January	-10	10
April	5	20
July	20	28
October	5	22

Examples

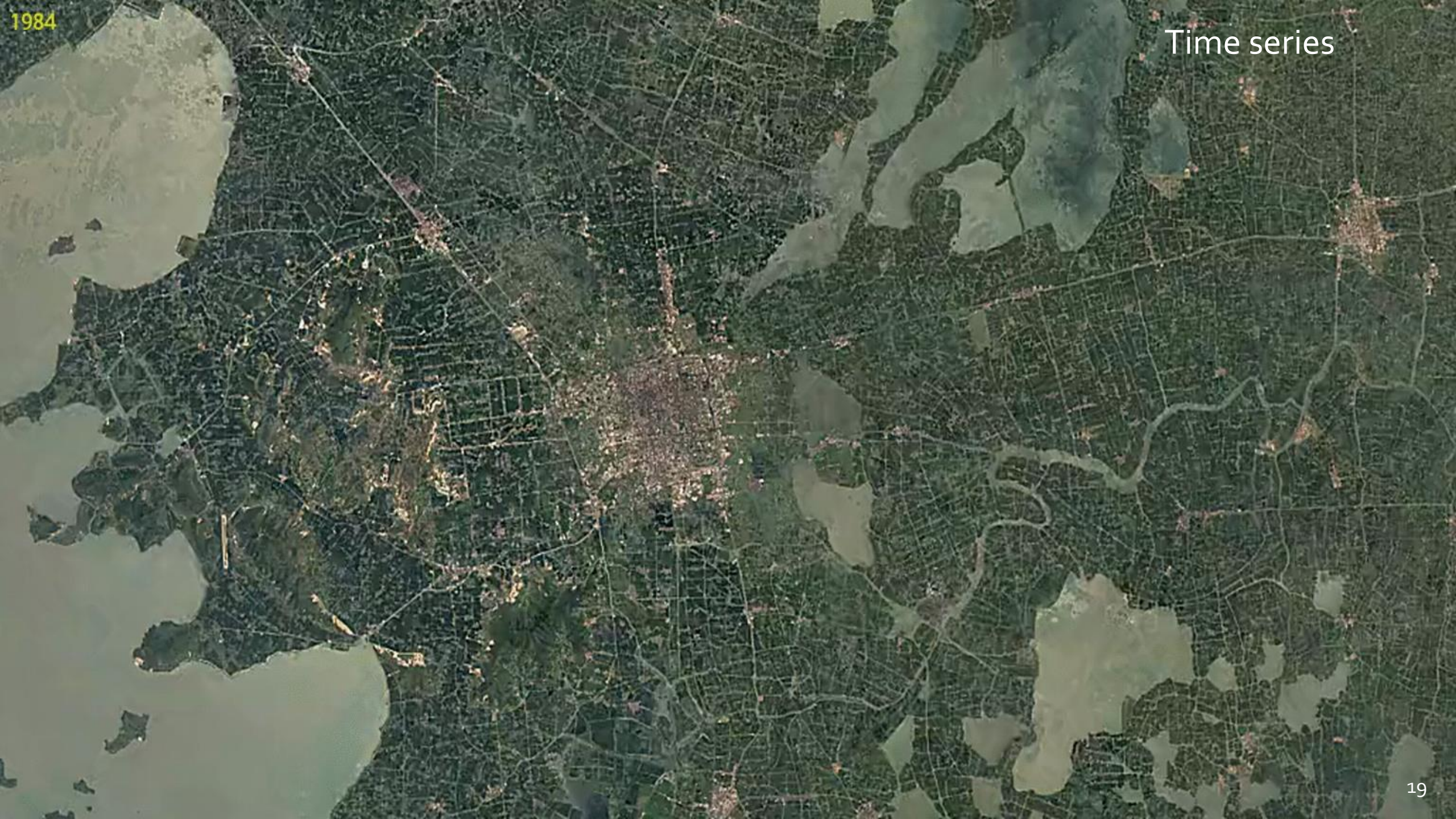
1984

Time series

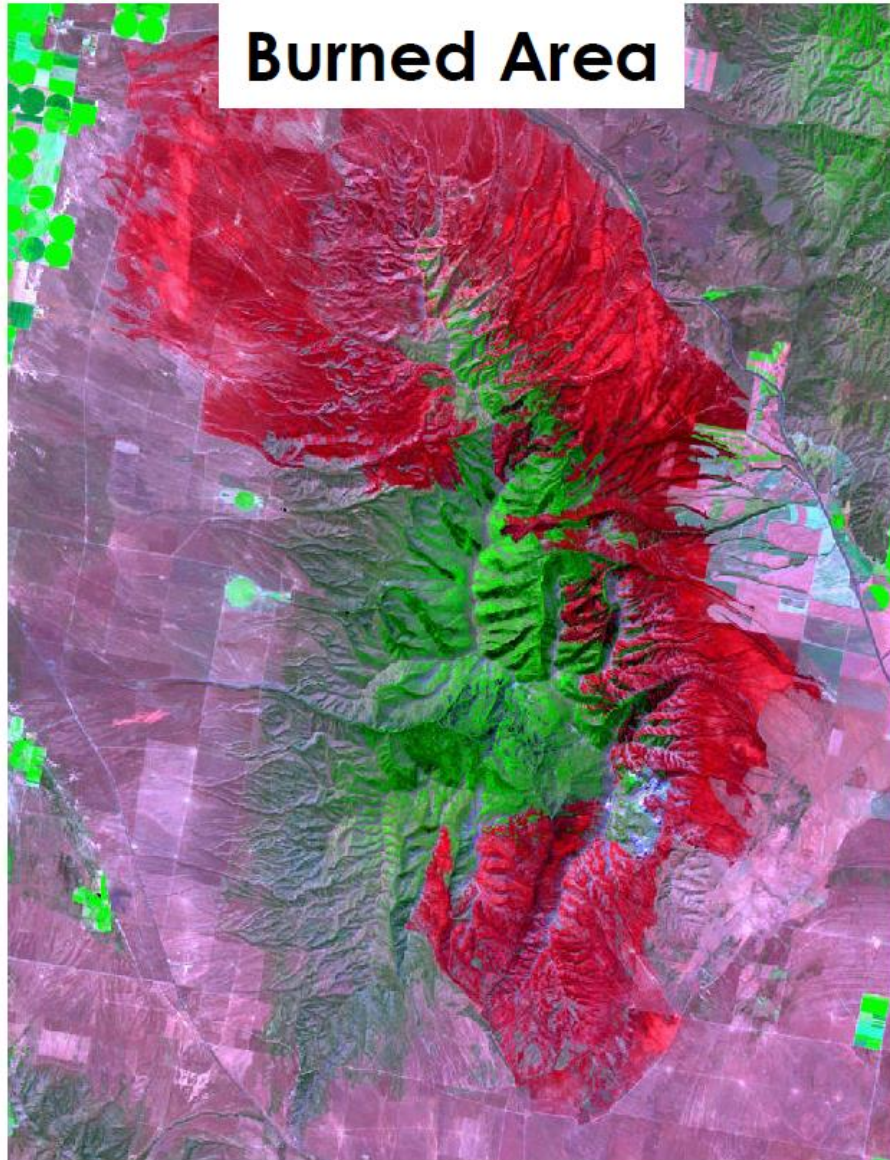


1984

Time series



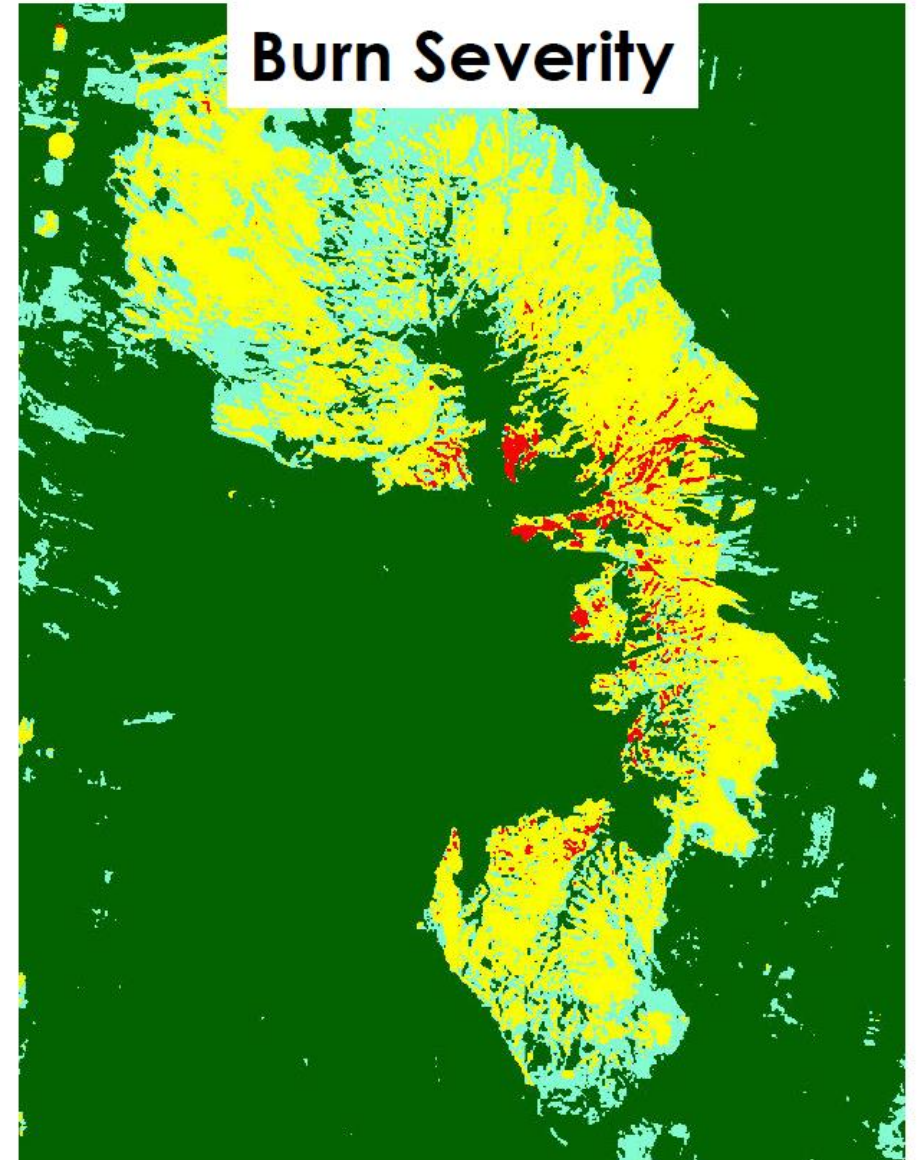
Observations for Pre-and Post-Fire Monitoring



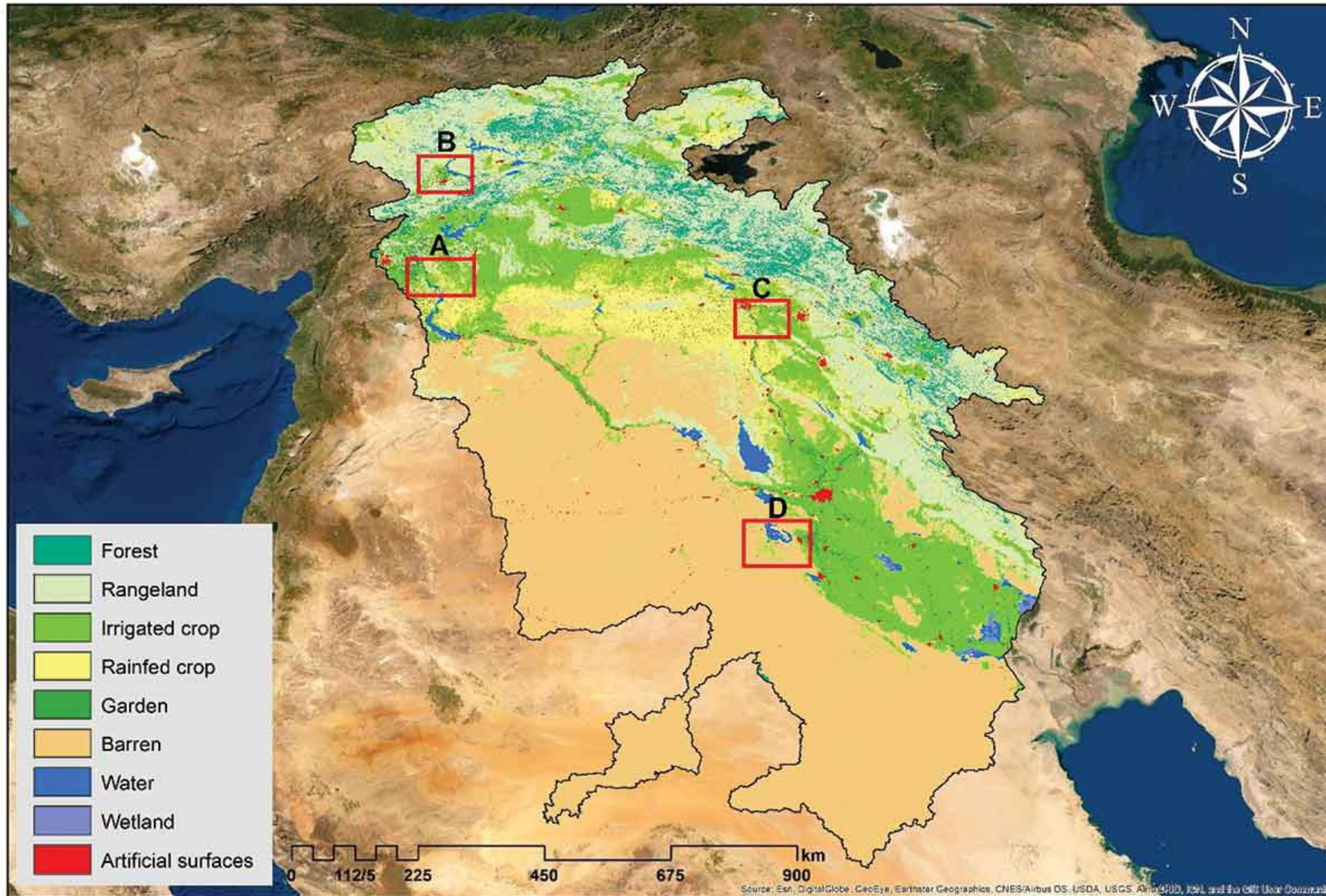
- Burned area uses imagery to assess the extent of impacts on vegetation for a particular fire event.



- Burn severity compares burned area information to pre-fire imagery to assess relative magnitude of burn impacts.

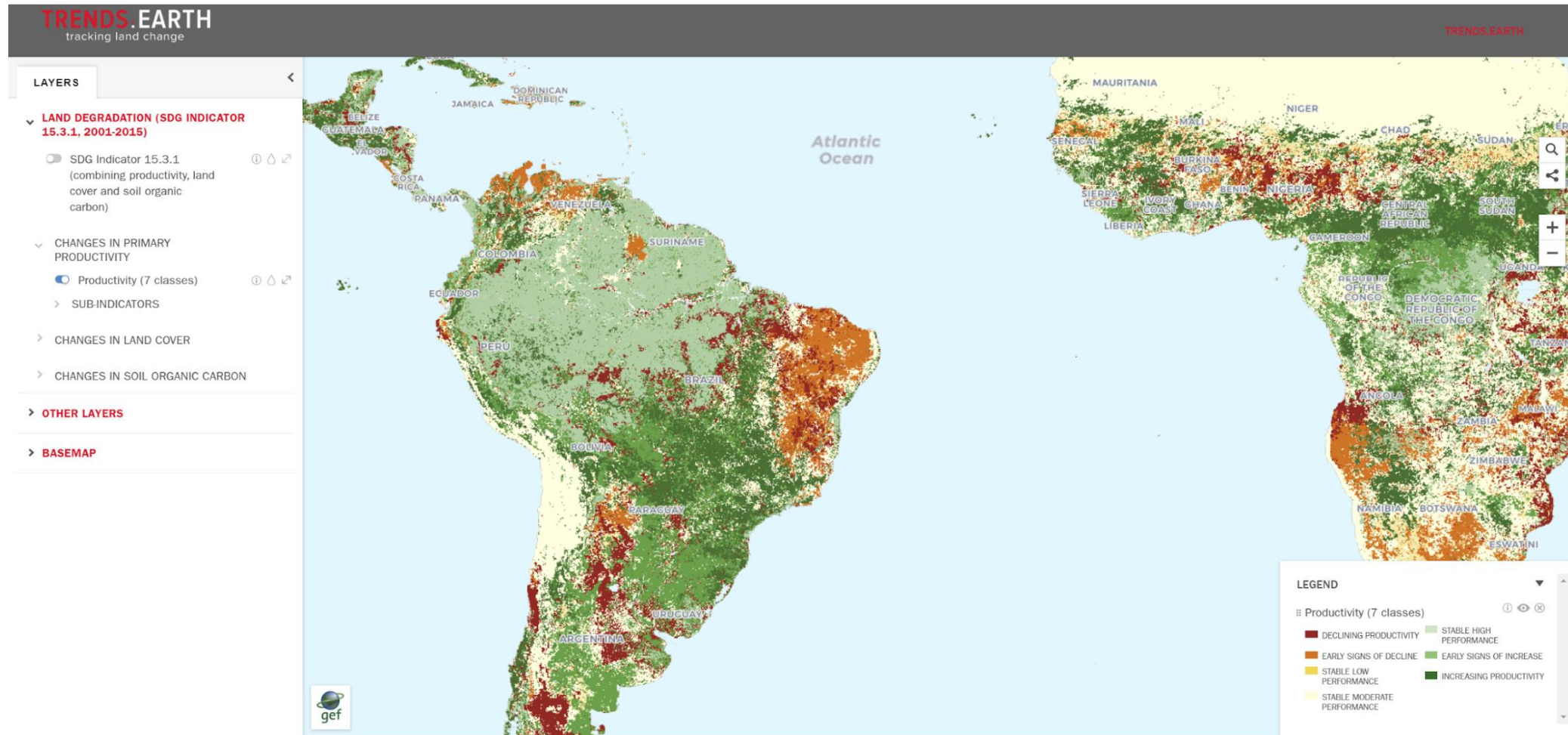


Soil Mapping and Classification



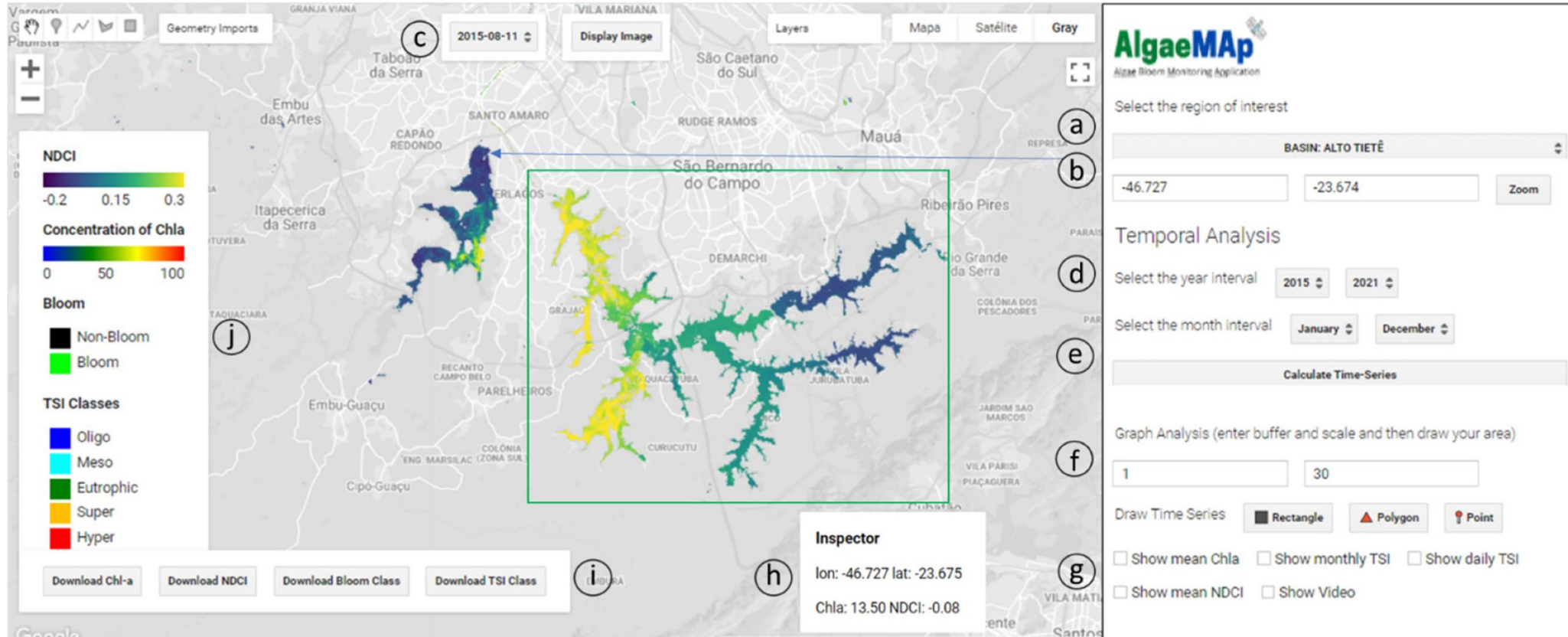
Trends.Earth

Through this project, we will develop a cloud-based platform dedicated to mapping land degradation which identifies potential land restoration opportunities at national to regional scale, allowing communities to prioritize areas to protect, manage, and restore in order to achieve land degradation neutrality.



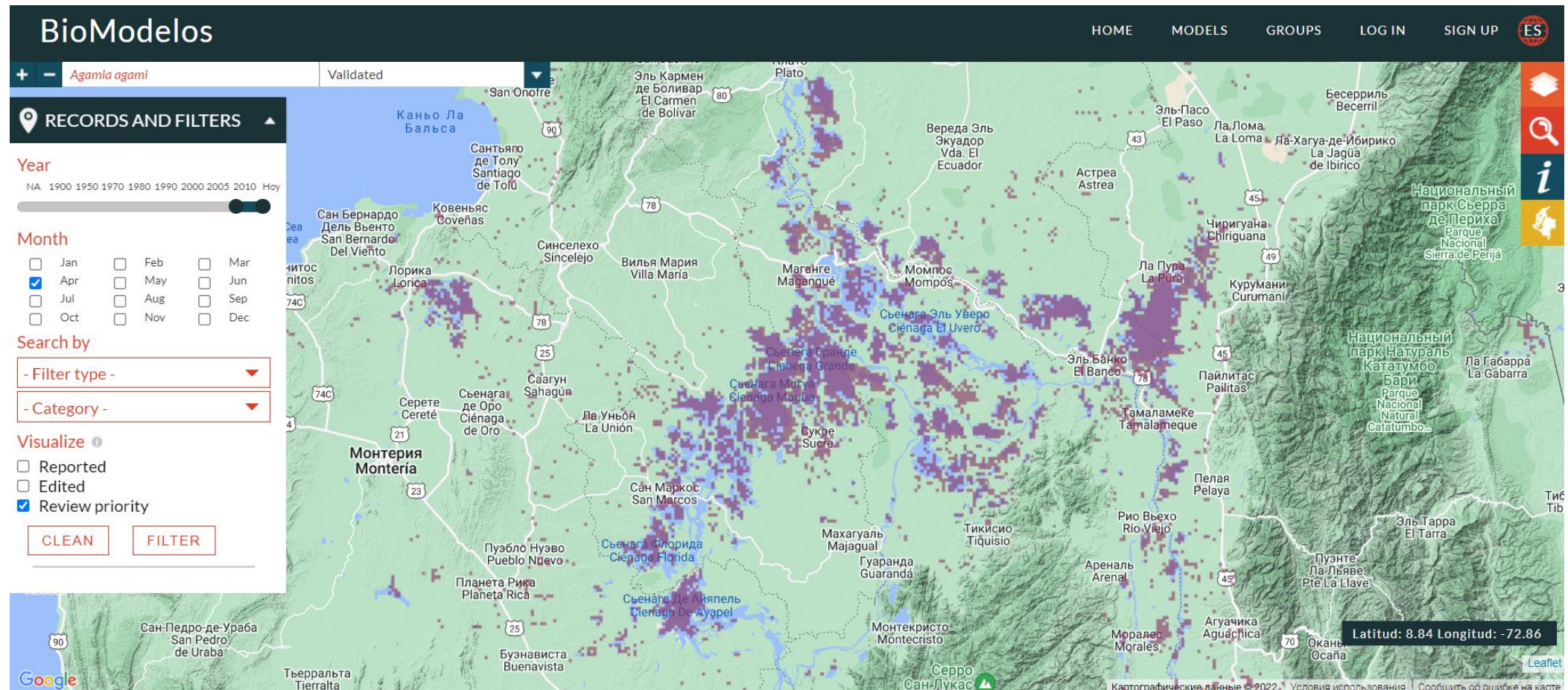
Alert System for Algal Bloom

Algae blooms occur when certain kinds of algae grow very quickly, forming patches, or "blooms," in the water. These blooms can be indicators of water degradation and emit powerful toxins that can endanger human and animal health.



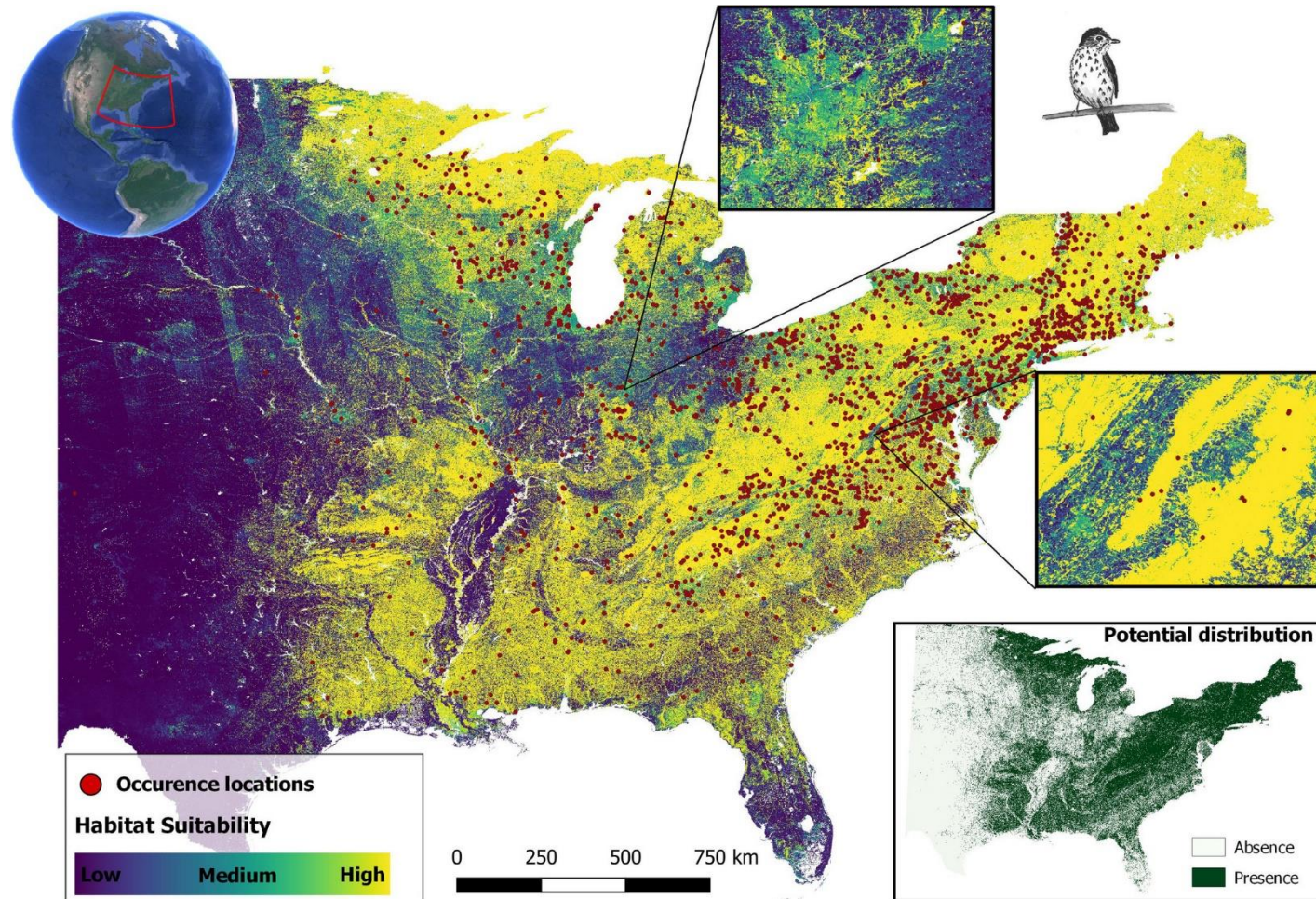
BioModelos: a collaborative online system to map species distributions

BioModelos, a modeling approach supported by an online system and a core team, whereby a network of experts contributes to the development of species distribution models by assessing the quality of occurrence data, identifying potentially limiting environmental variables, establishing species' accessible areas and validating qualitatively modeling predictions. Models developed through BioModelos become publicly available once validated by experts, furthering their use in conservation applications.



Essential Biodiversity Variables - ScaleUp

We implemented a workflow for species distribution modelling in GEE that includes importing species occurrence data into the GEE platform, selecting and preparing predictor variables, and performing model fitting with spatial or temporal split-block cross-validation techniques.

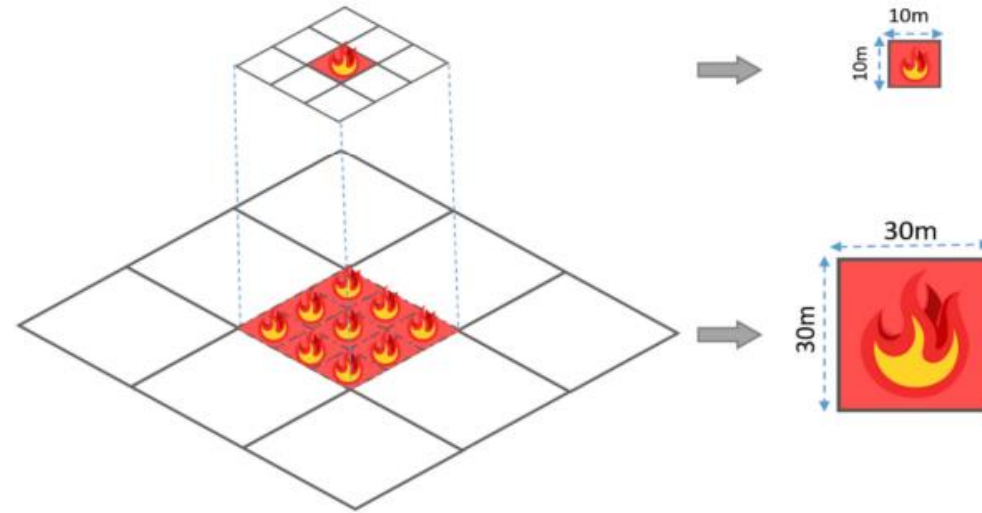
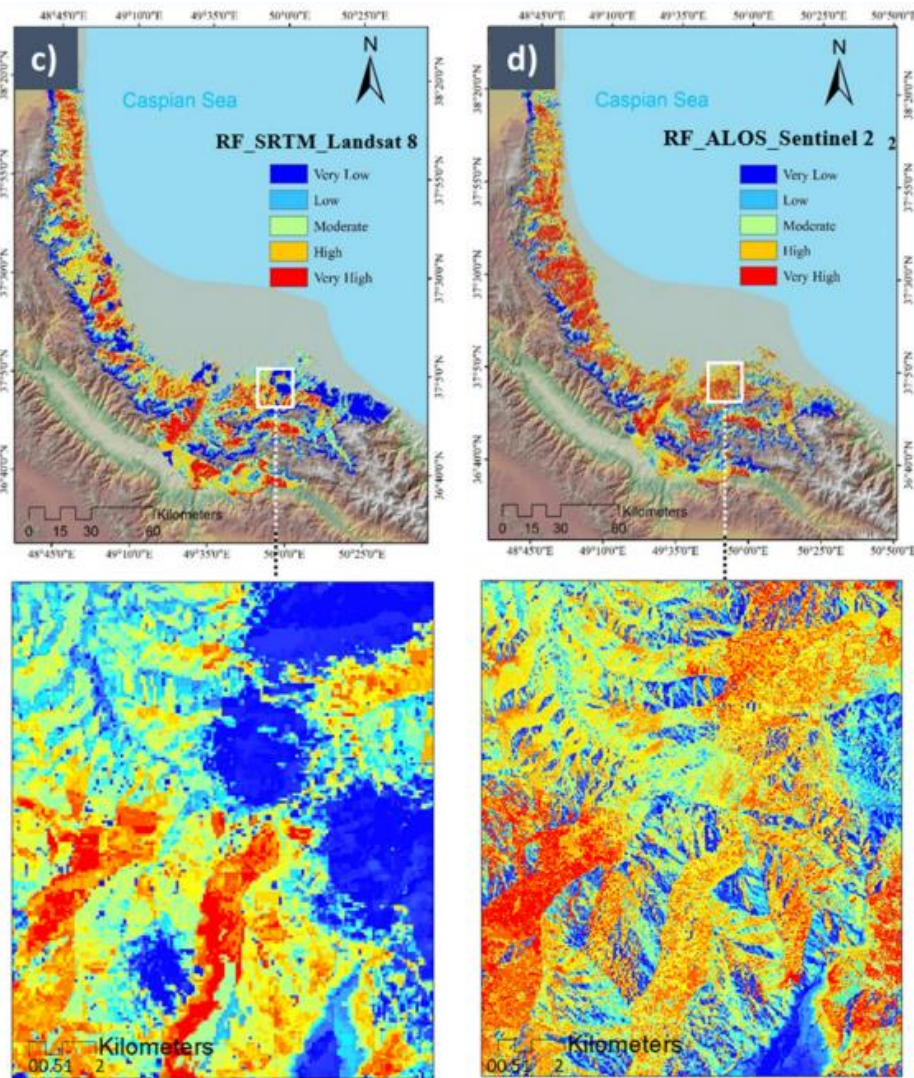


Fire Danger Warning System

This **app** could be used to visualize the **Australia Forest Fire** using Landsat 8 and Sentinel 2 composites



Wildfire Susceptibility Prediction Fusion with Remote Sensing Data of Different Spatial Resolutions

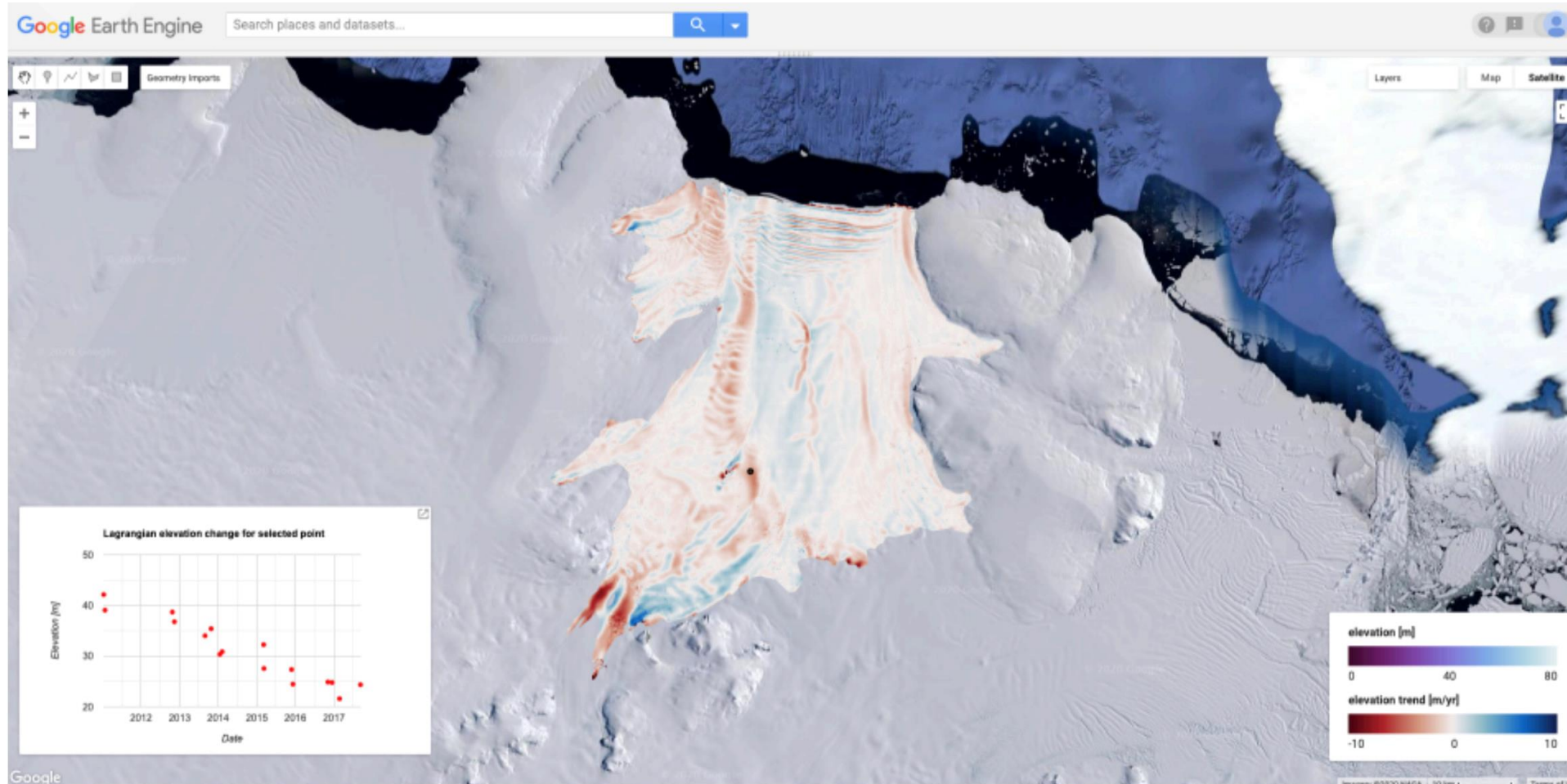


Tavakkoli Piralilou, S.; Einali, G.; Ghorbanzadeh, O.; Nachappa, T.G.; Gholamnia, K.; Blaschke, T.; Ghamisi, P. A Google Earth Engine Approach for Wildfire Susceptibility Prediction Fusion with Remote Sensing Data of Different Spatial Resolutions. *Remote Sens.* **2022**, *14*, 672. <https://doi.org/10.3390/rs14030672>

Figure 10. The resulting wildfire susceptibility prediction (WSP) maps from (a) SVM₃₀, (b) SVM₁₀, (c) RF₁₀, (d) RF₃₀.

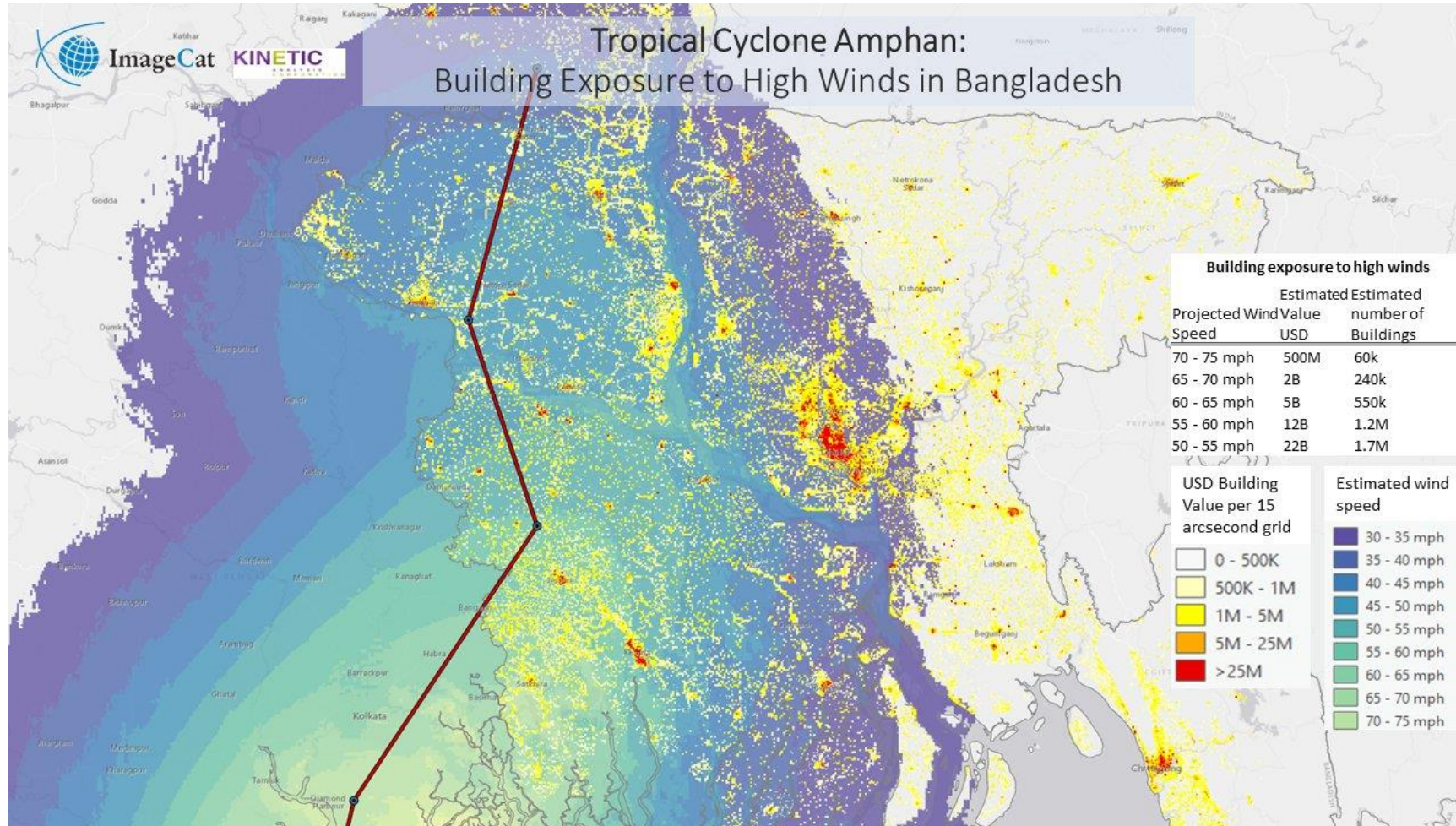
GEE-interface for analyzing basal melt over Dotson ice shelf

A platform to analyze and demonstrate the use of satellite data over Antarctic ice shelves. This will allow insight in the surface, subsurface and basal conditions of these ice shelves, which are major sources of uncertainty in future sea level projections from Antarctica.



ImageCat - Disaster Forecasting, Mitigation and Response

ImageCat, Inc., is a Long Beach, CA. company specializing in innovative solutions to natural hazard risk assessment and management. ImageCat offers solutions towards building a more resilient future using earth observation data and geospatial technologies.



Supporting water and disaster risk management in Mozambique

The project HydroPC focused on co-development and application of innovative data technologies and comprehensive training of beneficiaries to support water and disaster risk management in Mozambique.



Earth Engine Costs

It's free

An excerpt from their website:

"Why is Google working on Earth Engine?"

[Google's mission](#) is to organize the world's information and make it universally accessible and useful. In line with this mission, Earth Engine organizes geospatial information and makes it available for analysis. More generally, Google strives to make the world a better place through the use of technology. Earth Engine's technical infrastructure powers humanitarian, scientific, and environmental initiatives which Google is proud to support."

When to use EE?

EE Benefits:

Good for projects that requires:

- Data coverage for a large region
- Extensive data library
- High speed, intensive processing capacity
- Advanced raster processing tools

EE Limitations:

- Better suited to image analyses than vector-based analyses
- Analysis based on pixel spatial relations are harder to complete (because of the processing on multiple CPU's). Image segmentation and hydrologic modeling options are limited

[https://developers.google.com/earth-engine/guides/playground#:~:text=The%20Earth%20Engine%20\(EE\)%20Code,JavaScript%20code%20editor](https://developers.google.com/earth-engine/guides/playground#:~:text=The%20Earth%20Engine%20(EE)%20Code,JavaScript%20code%20editor)

The image shows a screenshot of the Google Earth Engine playground interface. The interface is divided into several sections:

- Search for datasets or places:** A search bar at the top with the text "Search places and datasets...".
- Script manager:** A sidebar on the left with a "Filter scripts..." field and a "NEW" button. It lists various scripts under categories like "Owner (16)", "Writer", "Reader", and "Examples".
- API documentation:** A link in the top navigation bar.
- Asset manager:** A link in the top navigation bar.
- Code Editor:** A central text area containing JavaScript code for processing Sentinel-2 data. The code includes comments and function definitions for cloud masking and data scaling.
- Inspector:** A panel on the right for inspecting map elements.
- Console:** A panel on the right for viewing output from the code.
- Task manager:** A panel on the right for managing tasks.
- Map:** A satellite map at the bottom with a "Layers" panel on the right and a "Zoom" control on the left.
- Geometry Tools:** A set of icons on the left side of the map for drawing shapes.
- Help button:** A question mark icon in the top right corner.
- Feedback button:** A speech bubble icon in the top right corner.

Arrows point from text labels to these specific interface elements.

Basics

<https://code.earthengine.google.com/6b24f55c3c60991a09352ce552fd310a>

Scripts Docs Assets

Filter scripts... NEW Refresh

Owner (1)

- users/zmeijska/default
 - Image Collection
 - experiment
 - Ivanovo
 - Poland
 - Sweden
 - ndvi
 - ndvi_calculation
 - razm2
 - razmeri
 - romania

New Script * Get Link Save Run Reset Apps Settings

```

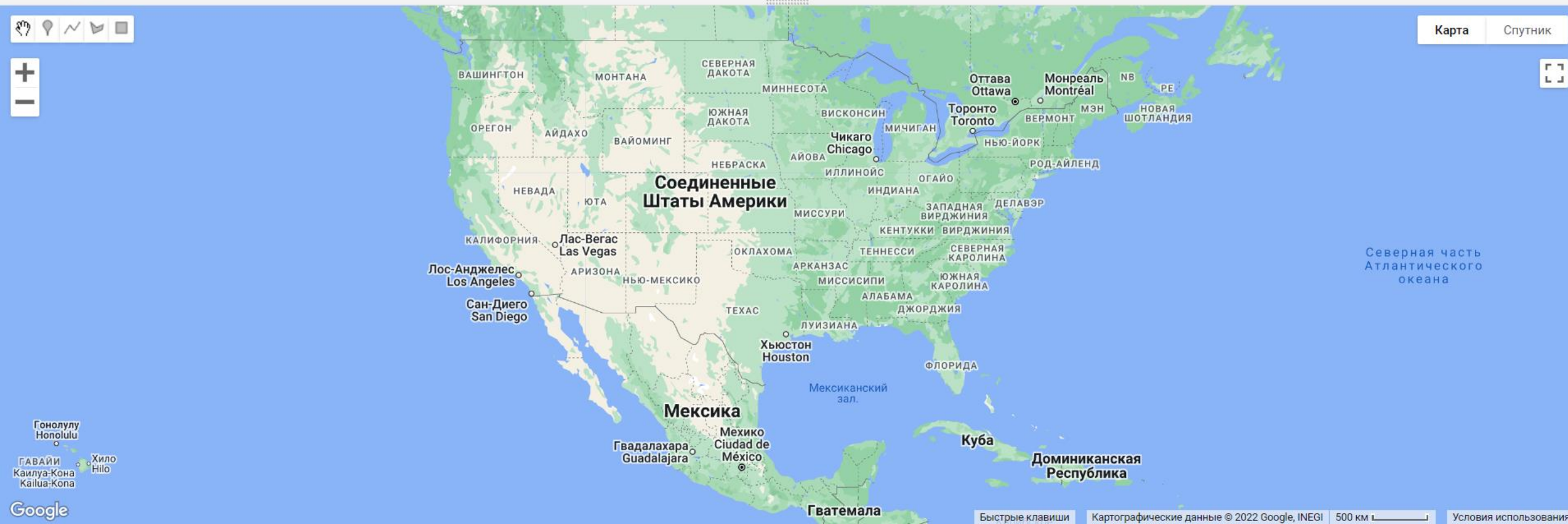
1 // Use single (or double) quotes to make a string.
2 var greetString = 'Ahoy there!';
3 // Use parentheses to pass arguments to functions.
4 print(greetString);
5
6 // Store a number in a variable.
7 var number = 42;
8 print('The answer is:', number);
9
10 // Use square brackets [] to make a list.
11 var listOfNumbers = [0, 1, 1, 2, 3, 5];
12 print('List of numbers:', listOfNumbers);
13
14

```

Inspector Console Tasks

Use print(...) to write to this console.

Ahoy there!	JSON
The answer is: 42	JSON
List of numbers: ▶ [0,1,1,2,3,5]	JSON



Scripts Docs Assets

Filter scripts... NEW Refresh

Owner (1)

- users/zmeijska/default
 - Image Collection
 - experiment
 - Ivanovo
 - Poland
 - Sweden
 - ndvi
 - ndvi_calculation
 - razm2
 - razmeri
 - romania
 - tutorial

New Script * Get Link Save Run Reset Apps Settings

```

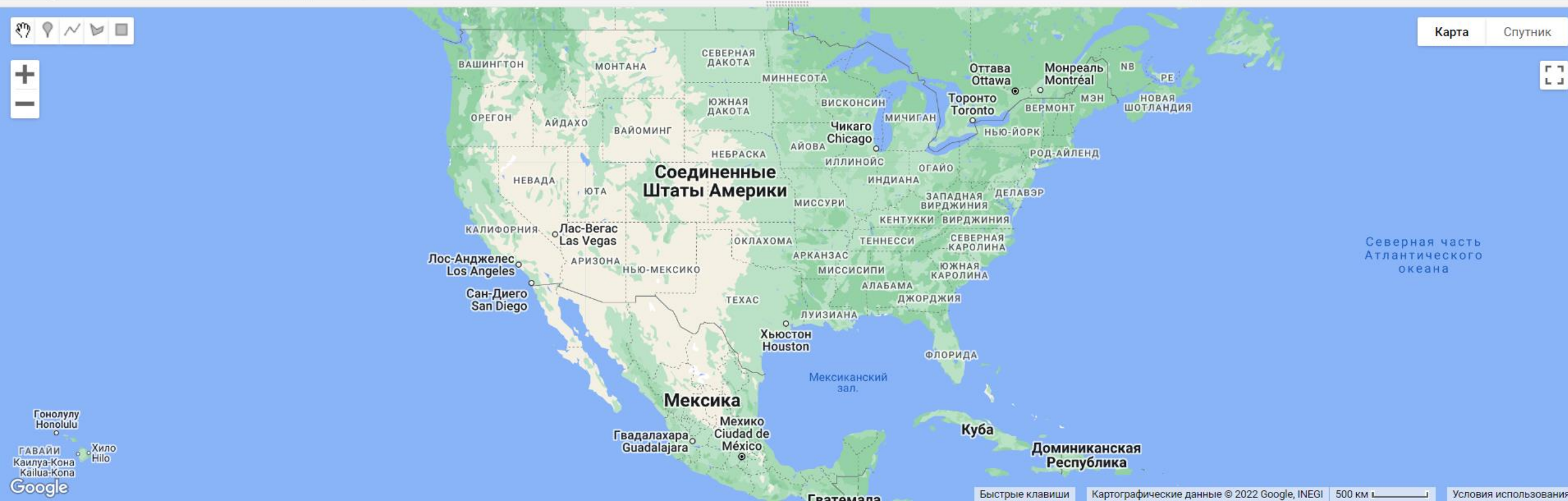
1 // Use curly brackets {} to make a dictionary of key:value pairs.
2 var object = {
3   foo: 'bar',
4   baz: 13,
5   stuff: ['this', 'that', 'the other thing']
6 };
7 print('Dictionary:', object);
8 // Access dictionary items using square brackets.
9 print('Print foo:', object['foo']);
10 // Access dictionary items using dot notation.
11 print('Print stuff:', object.stuff);

```

Inspector Console Tasks

Use print(...) to write to this console.

Dictionary:	JSON
Object (3 properties)	JSON
Print foo:	JSON
bar	JSON
Print stuff:	JSON
["this","that","the other thing"]	JSON



Scripts Docs Assets

Filter scripts... NEW Refresh

Owner (1)

- users/zmeijska/default
 - Image Collection
 - experiment
 - Ivanovo
 - Poland
 - Sweden
 - ndvi
 - ndvi_calculation
 - razm2
 - razmeri
 - romania
 - tutorial

New Script * Get Link Save Run Reset Apps Settings

```

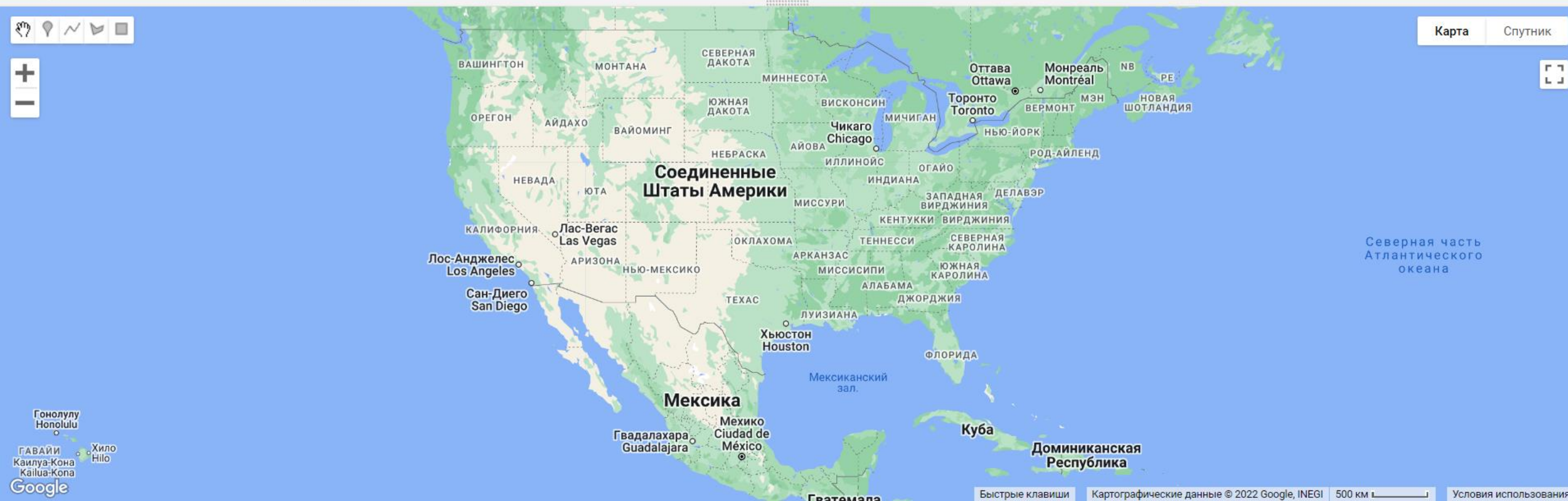
1 // The reflect function takes a single parameter: element.
2 var reflect = function(element) {
3   // Return the argument.
4   return element;
5 };
6 print('A good day to you!', reflect('Back at you!'));
  
```

Inspector Console Tasks

Use print(...) to write to this console.

```

A good day to you!      JSON
Back at you!           JSON
  
```



Scripts Docs Assets

Filter scripts... NEW Refresh

Owner (1)

- users/zmeijska/default
 - Image Collection
 - experiment
 - Ivanovo
 - Poland
 - Sweden
 - ndvi
 - ndvi_calculation
 - razm2
 - razmeri
 - romania
 - tutorial

New Script * Get Link Save Run Reset Apps Settings

```

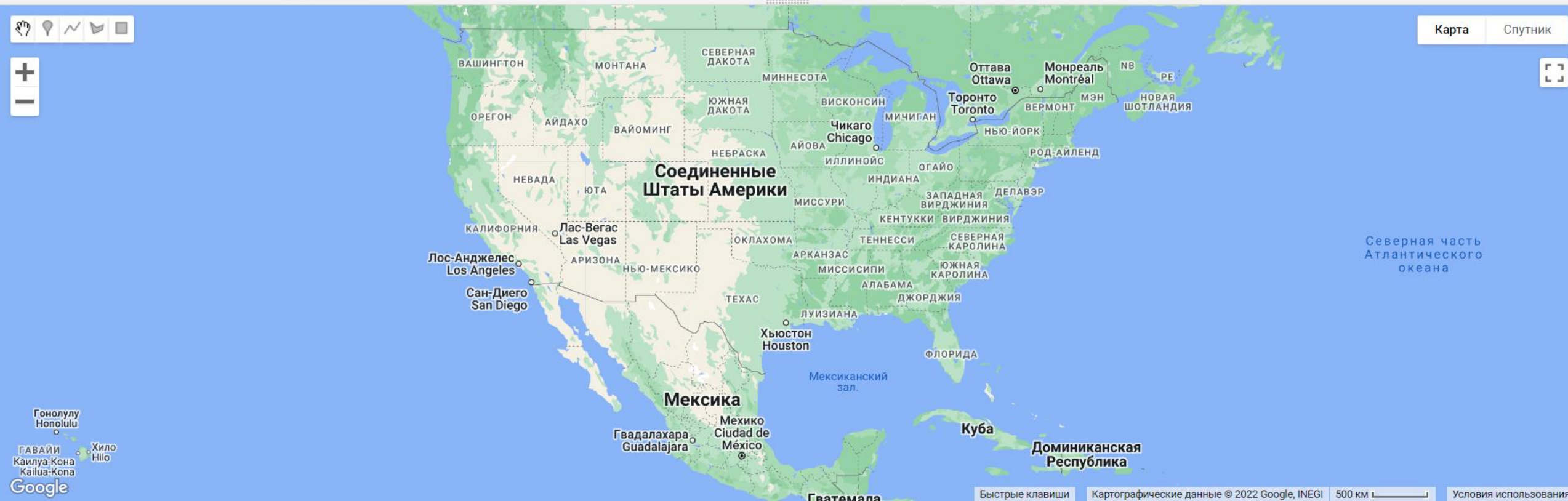
1 // Define a date in Earth Engine.
2 var date = ee.Date('2015-12-31');
3 print('Date:', date);
4
5 // Get the current time using the JavaScript Date.now() method.
6 var now = Date.now();
7 print('Milliseconds since January 1, 1970', now);
8
9 // Initialize an ee.Date object.
10 var eeNow = ee.Date(now);
11 print('Now:', eeNow);

```

Inspector Console Tasks

Use print(...) to write to this console.

Date:	JSON
▶ Date (2015-12-31 00:00:00)	JSON
Milliseconds since January 1, 1970	JSON
1657653540108	
Now:	JSON
▶ Date (2022-07-12 19:19:00)	JSON



Data Catalog

<https://developers.google.com/earth-engine/datasets/catalog/>

Earth Engine Data Catalog

Search Language

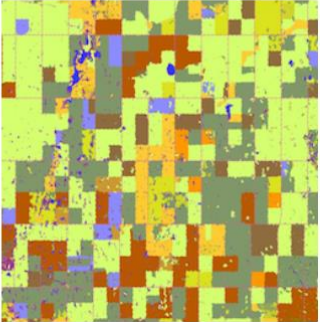

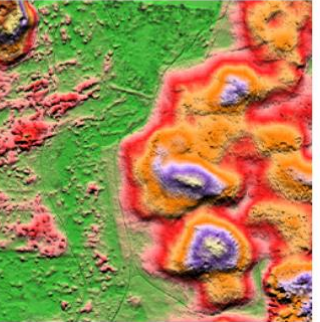
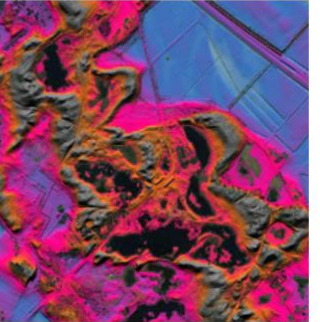

Home View all datasets Browse by tags Landsat MODIS Sentinel API Docs

Earth Engine Data Catalog

Earth Engine's public data catalog includes a variety of standard Earth science raster datasets. You can import these datasets into your script environment with a single click. You can also upload your own [raster data](#) or vector data for private use or sharing in your scripts.

Looking for another dataset not in Earth Engine yet? Let us know by [suggesting a dataset](#).

Filter list of datasets

Canada AAFC Annual Crop Inventory	Allen Coral Atlas (ACA) - Geomorphic Zonation and Benthic Habitat - v1.0	AHN Netherlands 0.5m DEM, Interpolated	AHN Netherlands 0.5m DEM, Non-Interpolated	AHN Netherlands 0.5m DEM, Raw Samples
				
Starting in 2009, the Earth Observation Team of the Science and Technology Branch (STB) at Agriculture and Agri-Food Canada (AAFC) began the process of generating annual crop type digital maps. Focusing on the Prairie Provinces	The Allen Coral Atlas dataset maps the geomorphic zonation and benthic habitat for the world's shallow coral reefs at 5m pixel resolution. The underlying satellite image data are temporal composites of PlanetScope	The AHN DEM is a 0.5m DEM covering the Netherlands. It was generated from LIDAR data taken in the spring between 2007 and 2012. It contains ground level samples with all other items above ground (such as buildings, bridges, trees	The AHN DEM is a 0.5m DEM covering the Netherlands. It was generated from LIDAR data taken in the spring between 2007 and 2012. It contains ground level samples with all other items above ground (such as buildings, bridges, trees	The AHN DEM is a 0.5m DEM covering the Netherlands. It was generated from LIDAR data taken in the spring between 2007 and 2012. This version contains both ground level samples and items above ground level (such as buildings,

Images

https://developers.google.com/earth-engine/datasets/catalog/CGIAR_SRTM90_V4?hl=en

<https://code.earthengine.google.com/31f672aec5b2a3f8bc7d1fbf23450935>

```
// Instantiate an image with the Image constructor.
```

```
var image = ee.Image('CGIAR/SRTM90_V4');
```

```
// Zoom to a location.
```

```
Map.setCenter(37.02540746271644,56.766160331658845, 12);
```

```
Map.addLayer(image);
```

```
//Map.addLayer(image, {min: 50, max: 180});
```

```
//Map.addLayer(image, {min: 50, max: 180, palette: ['blue', 'green', 'red']}, 'custom palette');
```

Scripts Docs Assets

Filter scripts... NEW ↻

Owner (1)

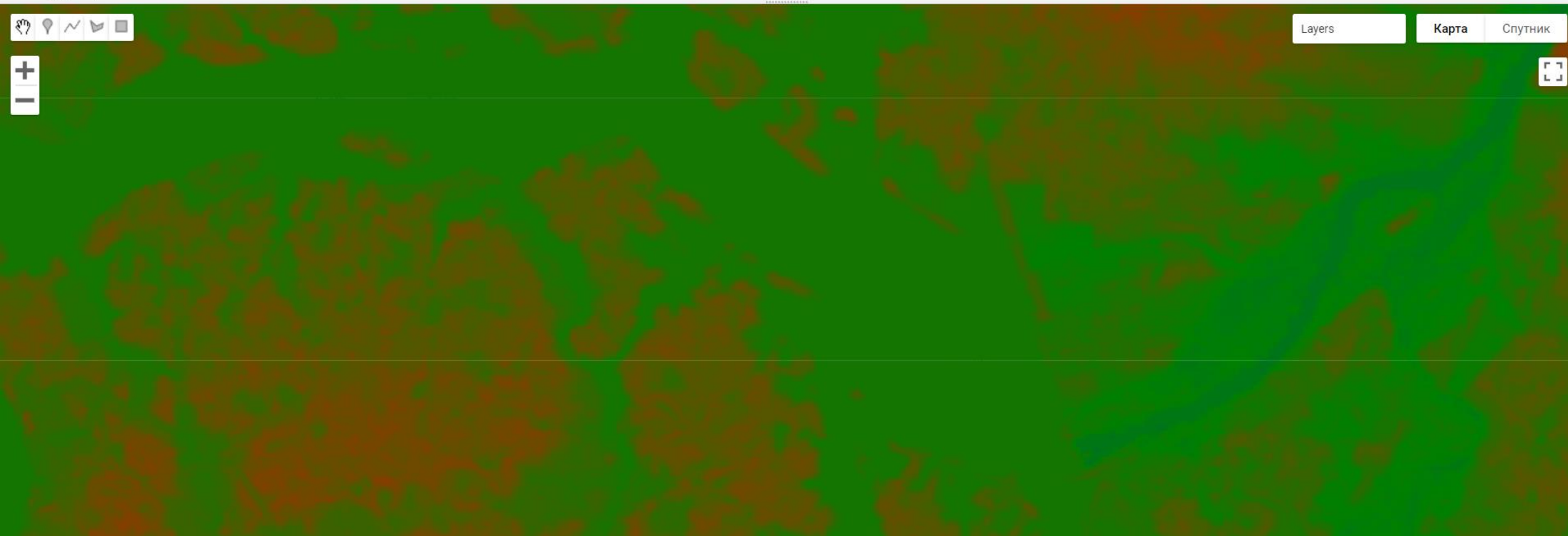
- users/zmeiyska/default
 - Image Collection
 - experiment
 - lip
 - basics
 - image 🔄 ✎ 🗑
 - Ivanovo
 - Poland

lip/image * Get Link Save Run Reset Apps ⚙

```
1 // Instantiate an image with the Image constructor.
2 var image = ee.Image('CGIAR/SRTM90_V4');
3
4 // Zoom to a location.
5 Map.setCenter(37.02540746271644,56.766160331658845, 12);
6
7 //Map.addLayer(image);
8 //Map.addLayer(image, {min: 50, max: 180});
9 Map.addLayer(image, {min: 50, max: 180, palette: ['blue', 'green', 'red']}, 'custom palette');
```

Inspector Console Tasks

Use print(...) to write to this console.



Images computation

<https://code.earthengine.google.com/a4b27505dfd223c7f5878e97ca42d06b>

<https://developers.google.com/earth-engine/apidocs/ee-terrain-slope>

```
// Instantiate an image with the Image constructor.  
var image = ee.Image('CGIAR/SRTM90_V4');  
  
// Apply an algorithm to an image.  
var slope = ee.Terrain.slope(image);  
  
// Zoom to a location.  
Map.setCenter(37.02540746271644,56.766160331658845, 12);  
  
Map.addLayer(slope, {min: 0, max :5}, 'slope');
```

Scripts Docs Assets

image comp *

Get Link

Save

Run

Reset

Apps



Inspector

Console

Tasks

```
1 // Instantiate an image with the Image constructor.
2 var image = ee.Image('CGIAR/SRTM90_V4');
3
4 // Apply an algorithm to an image.
5 var slope = ee.Terrain.slope(image);
6
7 // Zoom to a location.
8 Map.setCenter(37.02540746271644,56.766160331658845, 12);
9
10 Map.addLayer(slope, {min: 0, max :5}, 'slope');
```

Use print(...) to write to this console.

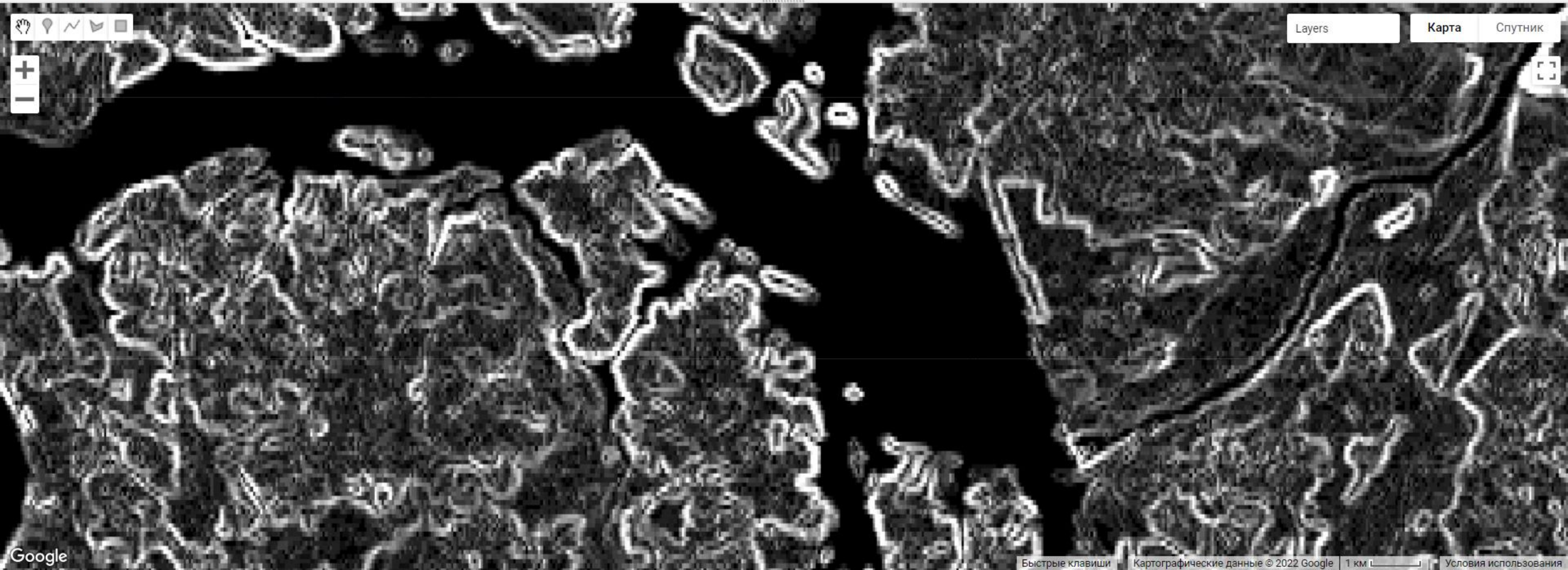


Image statistics

<https://code.earthengine.google.com/37d0c570ad54bf3d0232386d868be168>

```
// Instantiate an image with the Image constructor.
var image = ee.Image('CGIAR/SRTM90_V4');

// Zoom to a location.
Map.setCenter(37.02540746271644,56.766160331658845, 12);
var p1 = ee.Geometry.Point(37.02540746271644,56.766160331658845);
var p2 = ee.Geometry.Point(37.03502049982581,56.770864050772836);

var mean = image.reduceRegion({
  reducer: ee.Reducer.mean(),
  geometry: p1,
  scale: 90
});

print(mean);

var mean = image.reduceRegion({
  reducer: ee.Reducer.mean(),
  geometry: p2,
  scale: 90
});

print(mean);

Map.addLayer(image, {min: 50, max: 180, palette: ['blue', 'green', 'red']}, 'custom palette');
Map.addLayer(p1);
Map.addLayer(p2);
```

Scripts Docs Assets

- experiment
 - lip
 - basics
 - collections
 - image
 - image comp
 - image statistics
 - Ivanovo
 - Poland
 - Sweden
 - ndvi
 - ndvi calculation

```

image statistics *
1 // Instantiate an image with the Image constructor.
2 var image = ee.Image('CGIAR/SRTM90_V4');
3
4 // Zoom to a location.
5 Map.setCenter(37.02540746271644, 56.766160331658845, 12);
6 var p1 = ee.Geometry.Point(37.02540746271644, 56.766160331658845);
7 var p2 = ee.Geometry.Point(37.03502049982581, 56.770864050772836);
8
9 var mean = image.reduceRegion({
10   reducer: ee.Reducer.mean(),
11   geometry: p1,
12   scale: 30
13 });
14

```

Inspector Console Tasks

Use print(...) to write to this console.

- Object (1 property) JSON
 - elevation: 134
- Object (1 property) JSON
 - elevation: 120

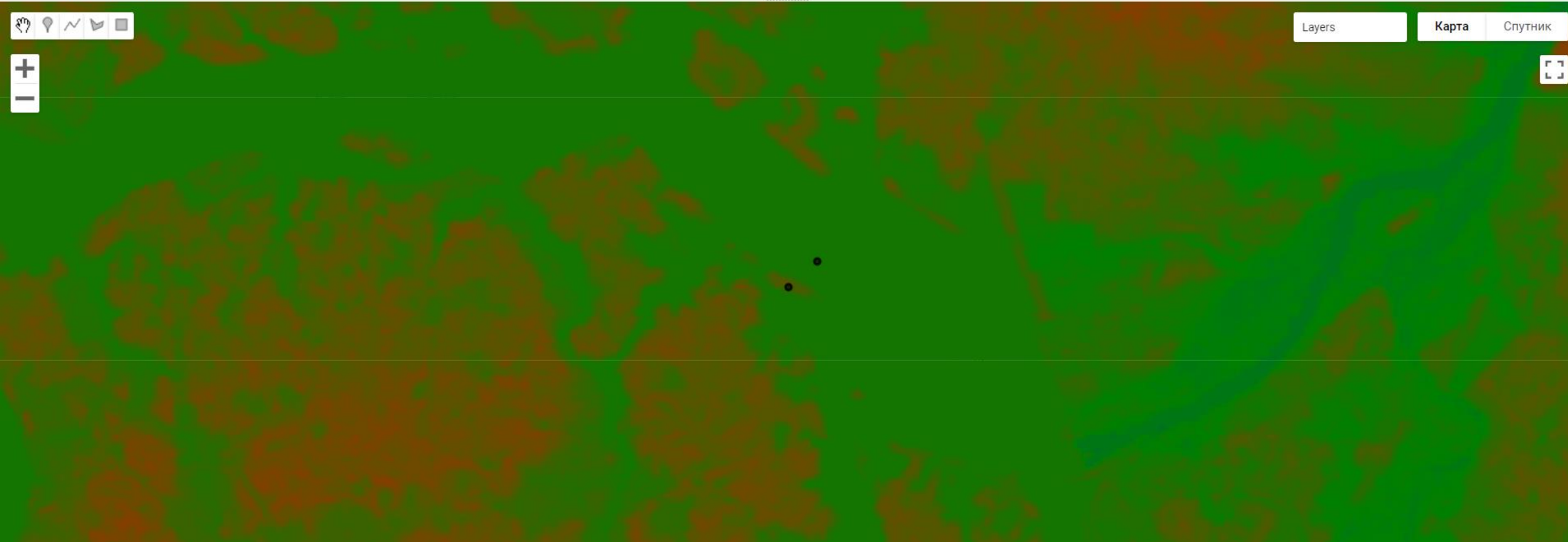


Image Collections

https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C02_T1_TOA

<https://code.earthengine.google.com/92fc966d883b11f9850b36504e6ea5eb>

```
var point = ee.Geometry.Point(37.02540746271644,56.766160331658845);  
var l8 = ee.ImageCollection('LANDSAT/LC08/C02/T1_TOA');
```

```
var spatialFiltered = l8.filterBounds(point);  
print('spatialFiltered', spatialFiltered);
```

```
var temporalFiltered = spatialFiltered.filterDate('2022-06-01', '2022-07-17');  
print('temporalFiltered', temporalFiltered);
```

```
// This will sort from least to most cloudy.  
var sorted = temporalFiltered.sort('CLOUD_COVER');
```

```
// Get the first (least cloudy) image.  
var scene = sorted.first();
```

```
Map.setCenter(37.02540746271644,56.766160331658845, 12);  
var visParams = {bands: ['B4', 'B3', 'B2'], max: 0.3};  
Map.addLayer(scene, visParams, 'true-color composite less clouds');  
Map.addLayer(temporalFiltered, visParams, 'true-color composite');
```


- Scripts
- Docs
- Assets
 - classification
 - collection clouds
 - collection mask
 - collection median
 - collection ndvi
 - collections
 - fires
 - image
 - image comp
 - image statistics
 - timeseries
 - Ivanovo
 - Poland
 - Sweden

```

collections
  Get Link Save Run Reset Apps
1 var point = ee.Geometry.Point(37.02540746271644, 56.766160331658845);
2 var l8 = ee.ImageCollection('LANDSAT/LC08/C02/T1_TOA');
3
4 var spatialFiltered = l8.filterBounds(point);
5 print('spatialFiltered', spatialFiltered);
6
7 var temporalFiltered = spatialFiltered.filterDate('2022-06-01', '2022-07-17');
8 print('temporalFiltered', temporalFiltered);
9
10 // This will sort from least to most cloudy.
11 var sorted = temporalFiltered.sort('CLOUD_COVER');
12
13 // Get the first (least cloudy) image.
14 var scene = sorted.first();
15
16 Map.setCenter(37.02540746271644, 56.766160331658845, 12);
17 var visParams = {bands: ['B4', 'B3', 'B2'], max: 0.3};

```

Inspector Console Tasks

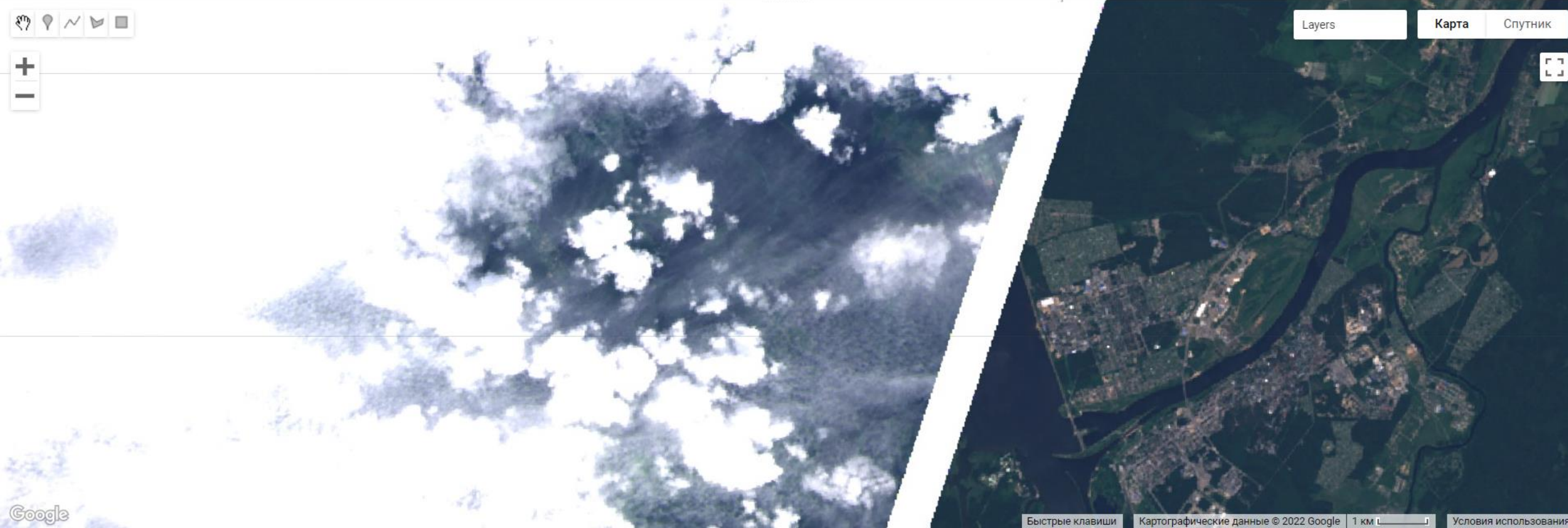
Use print(...) to write to this console.

```

spatialFiltered
  ImageCollection LANDSAT/LC08/C02/T1_TOA (464 elements, 17 band...
temporalFiltered
  ImageCollection LANDSAT/LC08/C02/T1_TOA (5 elements, 17 bands)

```

Map navigation controls: Hand, Location pin, Line graph, Print, Full screen, Zoom in (+), Zoom out (-)



Layers Карта Спутник

Median image

<https://code.earthengine.google.com/e06d18ab4bda8ae5c6500407d8bd97d5>

```
var l8 = ee.ImageCollection('LANDSAT/LC08/C02/T1_TOA').filterDate('2022-06-01', '2022-07-17');
```

```
Map.addLayer(l8, {bands: ['B4', 'B3', 'B2'], max: 0.3}, 'l8 collection');
```

```
// Get the median over time, in each band, in each pixel.
```

```
var median = l8.median();
```

```
Map.addLayer(median, {bands: ['B4', 'B3', 'B2'], max: 0.3}, 'l8 median');
```

```
Map.setCenter(37.02540746271644, 56.766160331658845, 12);
```

Scripts Docs Assets

- lip
 - basics
 - collection composi...
 - collections
 - image
 - image comp
 - image statistics
- Ivanovo
- Poland
- Sweden
- ndvi

collection composite *

Get Link Save Run Reset Apps

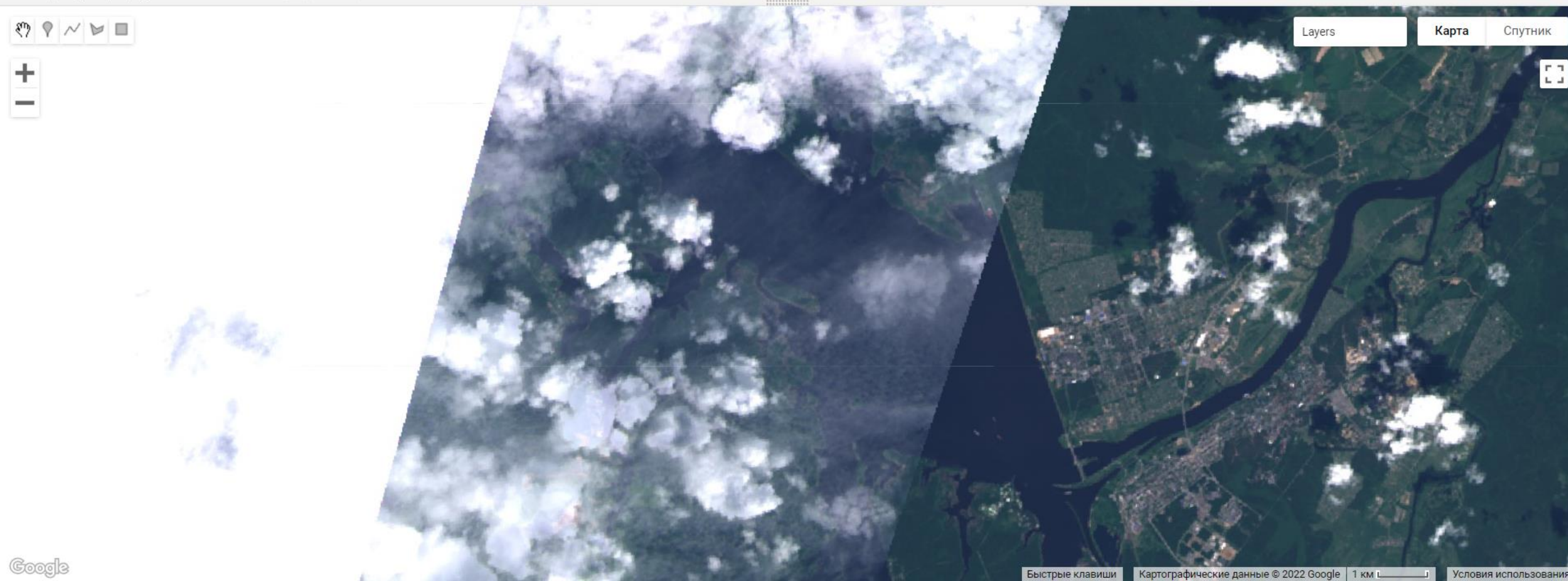
```

1 var l8 = ee.ImageCollection('LANDSAT/LC08/C02/T1_TOA').filterDate('2022-06-01', '2022-07-17');
2
3 Map.addLayer(l8, {bands: ['B4', 'B3', 'B2'], max: 0.3}, 'l8 collection');
4
5 // Get the median over time, in each band, in each pixel.
6 var median = l8.median();
7 Map.addLayer(median, {bands: ['B4', 'B3', 'B2'], max: 0.3}, 'l8 median');
8 Map.setCenter(37.02540746271644, 56.766160331658845, 12);
9

```

Inspector Console Tasks

Use print(...) to write to this console.



Masking

<https://developers.google.com/earth-engine/apidocs/ee-image-updatemask>

https://developers.google.com/earth-engine/datasets/catalog/UMD_hansen_global_forest_change_2021_v1_9?hl=en

<https://code.earthengine.google.com/976753b4bfb128c1cc8293ac60fe1b3f>

```
var l8 = ee.ImageCollection('LANDSAT/LC08/C02/T1_TOA').filterDate('2022-06-01', '2022-07-17');
```

```
// Get the median over time, in each band, in each pixel.
```

```
var median = l8.median();
```

```
Map.addLayer(median, {bands: ['B4', 'B3', 'B2'], max: 0.3}, 'l8 median');
```

```
Map.setCenter(37.02540746271644, 56.766160331658845, 12);
```

```
// Load or import the Hansen et al. forest change dataset.
```

```
var hansenImage = ee.Image('UMD/hansen/global_forest_change_2021_v1_9');
```

```
// Select the land/water mask.
```

```
var datamask = hansenImage.select('datamask');
```

```
// Create a binary mask.
```

```
var mask = datamask.eq(1);
```

```
// Update the composite mask with the water mask.
```

```
var maskedComposite = median.updateMask(mask);
```

```
Map.addLayer(maskedComposite, {bands: ['B4', 'B3', 'B2'], max: 0.3}, 'masked');
```

Scripts Docs Assets

- experiment
 - lip
 - basics
 - collection mask
 - collection median
 - collections
 - image
 - image comp
 - image statistics
 - Ivanovo
 - Poland

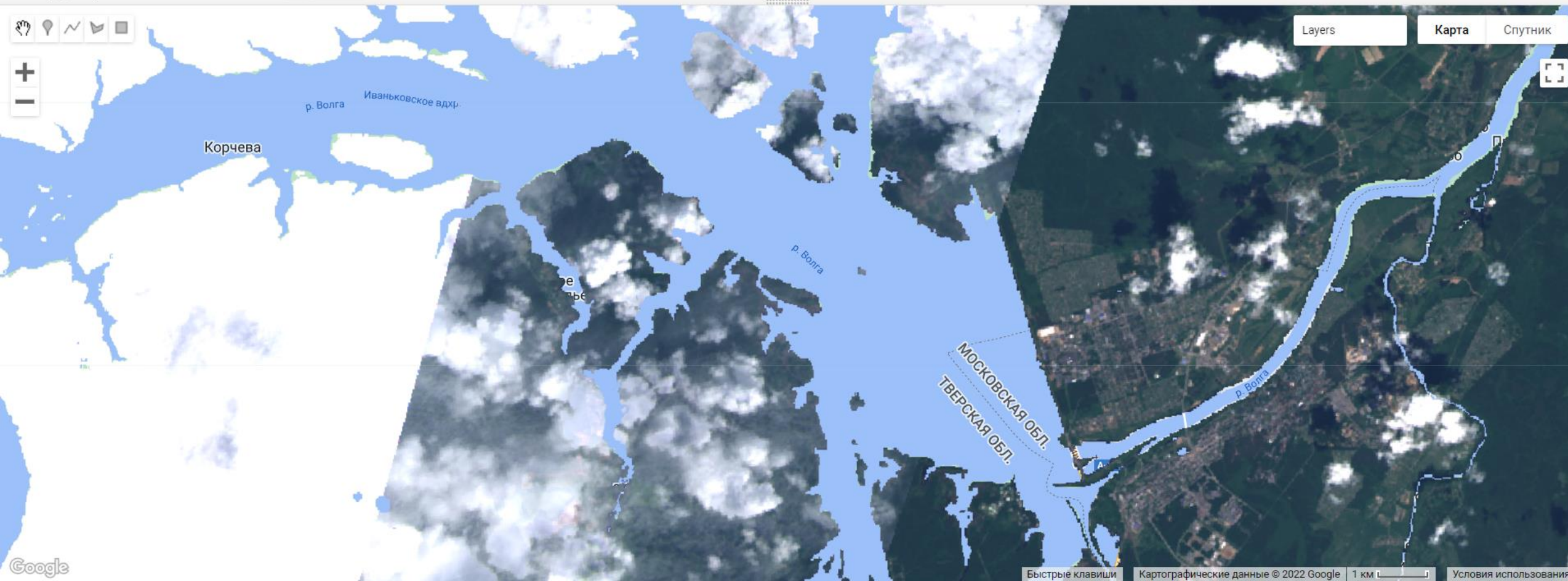
collection mask *

```

6 Map.setCenter(37.02540746271644,56.766160331658845, 12);
7
8 // Load or import the Hansen et al. forest change dataset.
9 var hansenImage = ee.Image('UMD/hansen/global_forest_change_2021_v1_9');
10
11 // Select the land/water mask.
12 var datamask = hansenImage.select('datamask');
13
14 // Create a binary mask.
15 var mask = datamask.eq(1);
16
17 // Update the composite mask with the water mask.
18 var maskedComposite = median.updateMask(mask);
19 Map.addLayer(maskedComposite, {bands: ['B4', 'B3', 'B2'], max: 0.3}, 'masked');
```

Inspector Console Tasks

Use print(...) to write to this console.



Clouds

<https://code.earthengine.google.com/36ce292efce7e367f53aac53ed970fbf>

https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C02_T1_L2#description

```
var dataset = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2')
    .filterDate('2022-05-01', '2022-07-17');

// Applies scaling factors.
function applyScaleFactors(image) {
  var opticalBands = image.select('SR_B.').multiply(0.0000275).add(-0.2);
  var thermalBands = image.select('ST_B.*').multiply(0.00341802).add(149.0);
  return image.addBands(opticalBands, null, true)
    .addBands(thermalBands, null, true);
}

dataset = dataset.map(applyScaleFactors);

var visualization = {
  bands: ['SR_B4', 'SR_B3', 'SR_B2'],
  min: 0.0,
  max: 0.3,
};

Map.setCenter(37.02540746271644, 56.766160331658845, 12);

Map.addLayer(dataset, visualization, 'clouds');
```

```
function maskL8sr(image) {
  // Bit 0 - Fill
  // Bit 1 - Dilated Cloud
  // Bit 2 - Cirrus
  // Bit 3 - Cloud
  // Bit 4 - Cloud Shadow
  var qaMask = image.select('QA_PIXEL').bitwiseAnd(parseInt('11111', 2)).eq(0);
  var saturationMask = image.select('QA_RADSAT').eq(0);

  // Apply the scaling factors to the appropriate bands.
  var opticalBands = image.select('SR_B.').multiply(0.0000275).add(-0.2);

  // Replace the original bands with the scaled ones and apply the masks.
  return image.addBands(opticalBands, null, true)
    .updateMask(qaMask)
    .updateMask(saturationMask);
}

// Map the function over one year of data.
var collection = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2')
    .filterDate('2022-05-01', '2022-07-17')
    .map(maskL8sr);

var composite = collection.median();

// Display the results.
Map.addLayer(composite, {bands: ['SR_B4', 'SR_B3', 'SR_B2'], min: 0, max: 0.3}, 'no clouds');
```

Scripts Docs Assets

- experiment
 - lip
 - basics
 - collection clouds
 - collection mask
 - collection median
 - collections
 - image
 - image comp
 - image statistics
 - Ivanovo

collection clouds *

Get Link Save Run Reset Apps

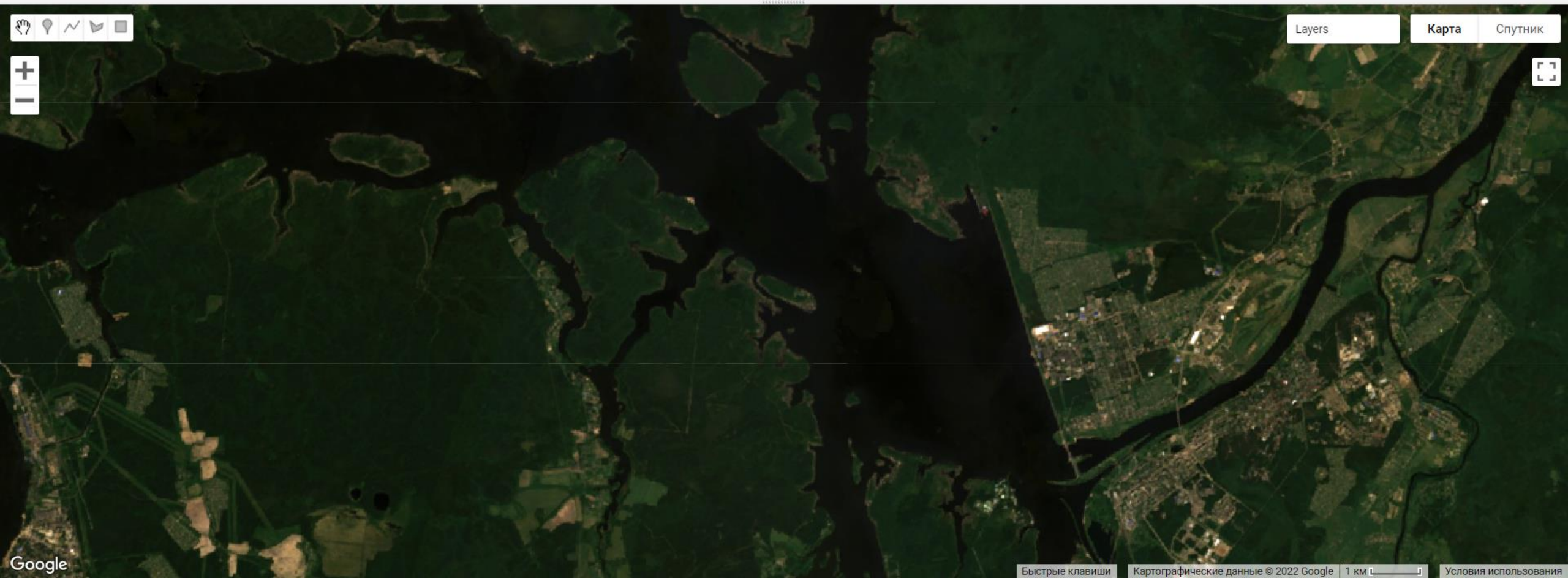
```

1 var dataset = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2')
2   .filterDate('2022-05-01', '2022-07-17');
3
4
5 // Applies scaling factors.
6 function applyScaleFactors(image) {
7   var opticalBands = image.select('SR_B.').multiply(0.0000275).add(-0.2);
8   var thermalBands = image.select('ST_B.*').multiply(0.00341802).add(149.0);
9   return image.addBands(opticalBands, null, true)
10     .addBands(thermalBands, null, true);
11 }
12
13 dataset = dataset.map(applyScaleFactors);
14

```

Inspector Console Tasks

Use print(...) to write to this console.



NDVI

<https://code.earthengine.google.com/ec9e2f81b63ad006475b263f5f0cd661>

```
var point = ee.Geometry.Point(37.02540746271644,56.766160331658845);
var l8 = ee.ImageCollection('LANDSAT/LC08/C02/T1_TOA');

// Get the least cloudy image in 2015.
var image = ee.Image(
  l8.filterBounds(point)
    .filterDate('2022-06-01', '2022-07-17')
    .sort('CLOUD_COVER')
    .first()
);

// Compute the Normalized Difference Vegetation Index (NDVI).
var nir = image.select('B5');
var red = image.select('B4');
var ndvi = nir.subtract(red).divide(nir.add(red)).rename('NDVI');

// Display the result.
Map.setCenter(37.02540746271644,56.766160331658845, 12);
var ndviParams = {min: -1, max: 1, palette: ['blue', 'white', 'green']};
Map.addLayer(ndvi, ndviParams, 'NDVI image');
```


- Scripts
- Docs
- Assets
 - changes
 - classification
 - collection clouds
 - collection mask
 - collection median
 - collection ndvi
 - collections
 - fires
 - image
 - image comp
 - image statistics

```

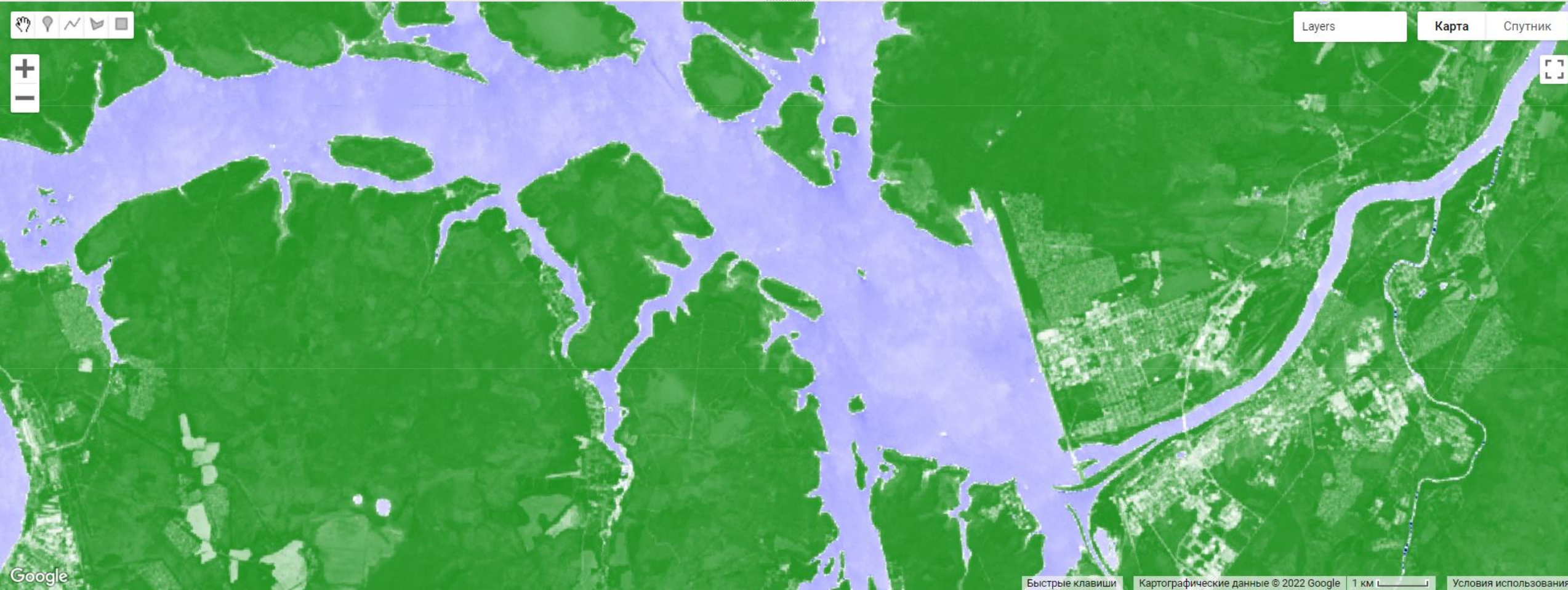
collection ndvi
Get Link Save Run Reset Apps

4 // Get the least cloudy image in 2015.
5 var image = ee.Image(
6   18.filterBounds(point)
7     .filterDate('2022-06-01', '2022-07-17')
8     .sort('CLOUD_COVER')
9     .first()
10 );
11
12 // Compute the Normalized Difference Vegetation Index (NDVI).
13 var nir = image.select('B5');
14 var red = image.select('B4');
15 var ndvi = nir.subtract(red).divide(nir.add(red)).rename('NDVI');
16

```

Inspector Console Tasks

Use print(...) to write to this console.



Charts

<https://code.earthengine.google.com/ec976be7da2129036839e1cf2f21cbbb>

```
//var cloud_perc = 60;//Max cloud percentile per scene.

// Import the Landsat 8 TOA image collection.
var l8 = ee.ImageCollection('LANDSAT/LC08/C01/T1_TOA')
  .filterDate('2021-03-01', '2021-12-01');
//   .filter(ee.Filter.lt('CLOUD_COVER', cloud_perc));

// Map a function over the Landsat 8 TOA collection to add an NDVI band.
var withNDVI = l8.map(function(image) {
  var ndvi = image.normalizedDifference(['B5', 'B4']).rename('NDVI');
  return image.addBands(ndvi);
});

var roi = ee.Geometry.Rectangle(37.01900746271644,56.76360331658845,37.0340746271644,56.76860331658845);
Map.setCenter(37.02540746271644,56.766160331658845, 12);
Map.addLayer(roi, {color: 'blue'}, 'clouds');

// Create a chart.
var chart = ui.Chart.image.series({
  imageCollection: withNDVI.select('NDVI'),
  region: roi,
  reducer: ee.Reducer.median(),
  scale: 30
}).setOptions({title: 'NDVI over time'});

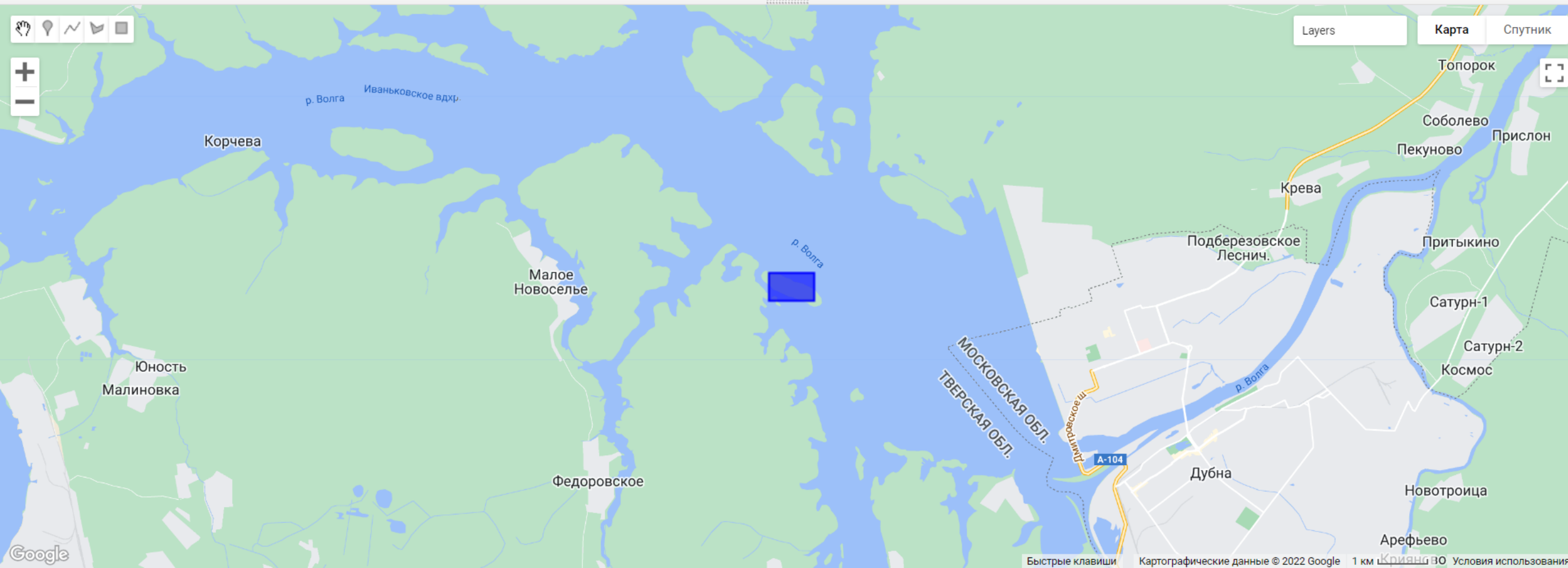
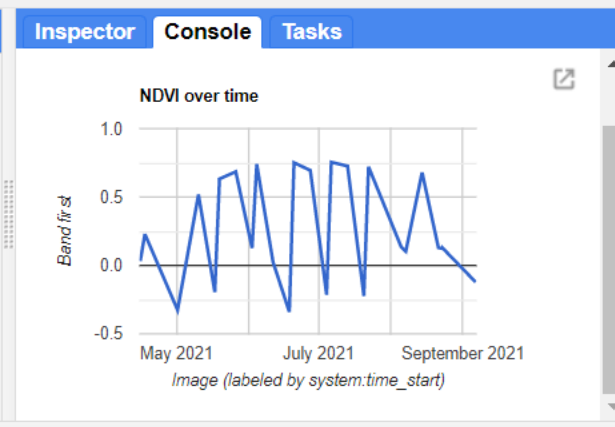
// Display the chart in the console.
print(chart);
```

- Scripts
- Docs
- Assets
- Layer Filters
- Linked Maps
- Manual Legend
- Mobile Friendly UI
- Mosaic Editor
- Ocean Timeseries Investigator
- Population Explorer
- Split Panel
- Two Chart Inspector
- Zoom Box
- Datasets
- Demos

```

lip/Charting
1 //var cloud_perc = 60;//Max cloud percentile per scene.
2
3 // Import the Landsat 8 TOA image collection.
4 var l8 = ee.ImageCollection('LANDSAT/LC08/C01/T1_TOA')
5   .filterDate('2021-03-01', '2021-10-01');
6   //   .filter(ee.Filter.lt('CLOUD_COVER', cloud_perc));
7
8 // Map a function over the Landsat 8 TOA collection to add an NDVI band.
9 var withNDVI = l8.map(function(image) {
10   var ndvi = image.normalizedDifference(['B5', 'B4']).rename('NDVI');
11   return image.addBands(ndvi);
12 });
13
14 var roi = ee.Geometry.Rectangle(37.01900746271644, 56.76360331658845, 37.0340746271644, 56.76860331658845);
15

```



Burn Areas

<https://developers.google.com/earth-engine/datasets/catalog/FIRMS?hl=en>

<https://code.earthengine.google.com/e9c2c3dd7cfb47bb55111557da6102d0>

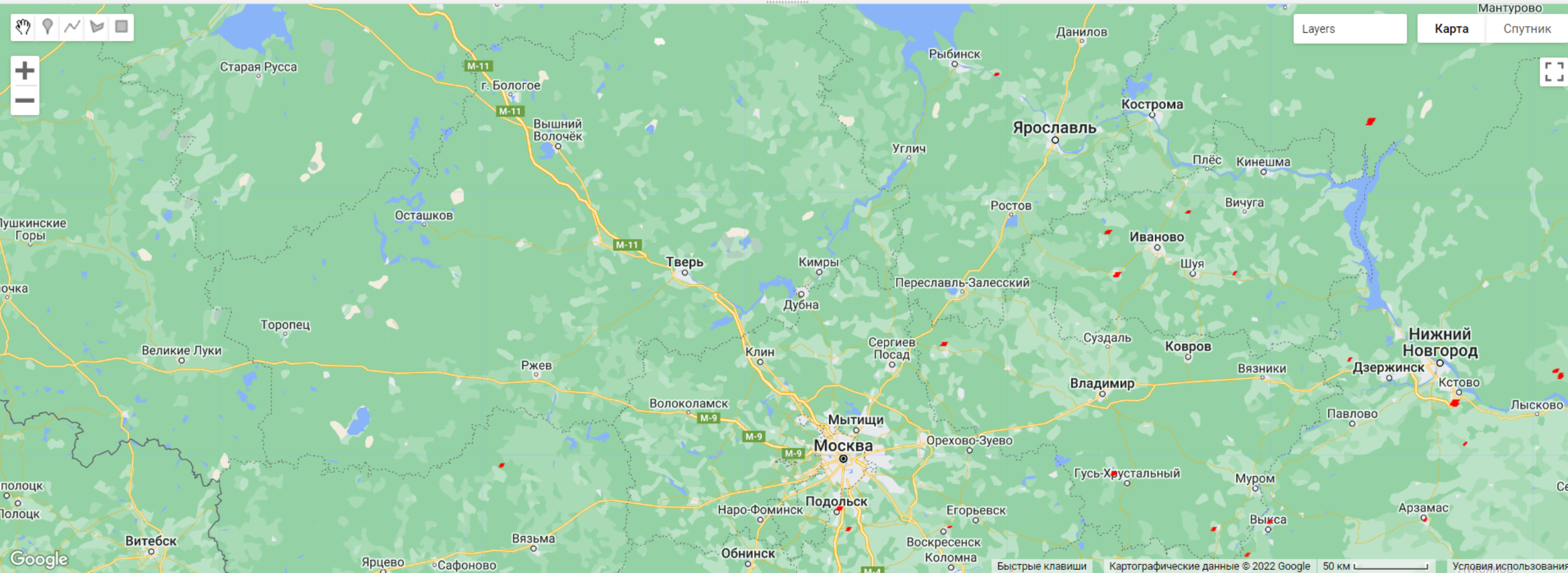
```
var dataset = ee.ImageCollection('FIRMS').filter(
  ee.Filter.date('2022-07-01', '2022-07-16'));
var fires = dataset.select('T21');
var firesVis = {
  min: 325.0,
  max: 400.0,
  palette: [
    'ff0000', 'fd4100', 'fb8200', 'f9c400', 'f2ff00', 'b6ff05',
    '7aff0a', '3eff0f', '02ff15', '00ff55', '00ff99', '00ffdd',
    '00ddff', '0098ff', '0052ff', '0210ff', '3a0dfb', '7209f6',
    'a905f1', 'e102ed', 'ff00cc', 'ff0089', 'ff0047', 'ff0004'
  ]
};
Map.setCenter(37.02540746271644,56.766160331658845, 7);
Map.addLayer(fires, firesVis, 'Fires');
```

- Scripts
- Docs
- Assets
 - collection mask
 - collection median
 - collection ndvi
 - collections
 - fires
 - image
 - image comp
 - image statistics
 - Ivanovo
 - Poland
 - Sweden
 - TOA corrections

```
lip/fires *
3 var fires = dataset.select('T21');
4 var firesVis = {
5   min: 325.0,
6   max: 400.0,
7   palette: [
8     'ff0000', 'fd4100', 'fb8200', 'f9c400', 'f2ff00', 'b6ff05',
9     '7aff0a', '3eff0f', '02ff15', '00ff55', '00ff99', '00ffdd',
10    '00ddff', '0098ff', '0052ff', '0210ff', '3a0dfb', '7209f6',
11    'a905f1', 'e102ed', 'ff00cc', 'ff0089', 'ff0047', 'ff0004'
12  ]
13 };
14 Map.setCenter(37.02540746271644, 56.766160331658845, 7);
15 Map.addLayer(fires, firesVis, 'Fires');
16
```

Inspector Console Tasks

Use print(...) to write to this console.



Timeseries

https://developers.google.com/earth-engine/datasets/catalog/Tsinghua_FROM-GLC_GAIA_v10?hl=en#description

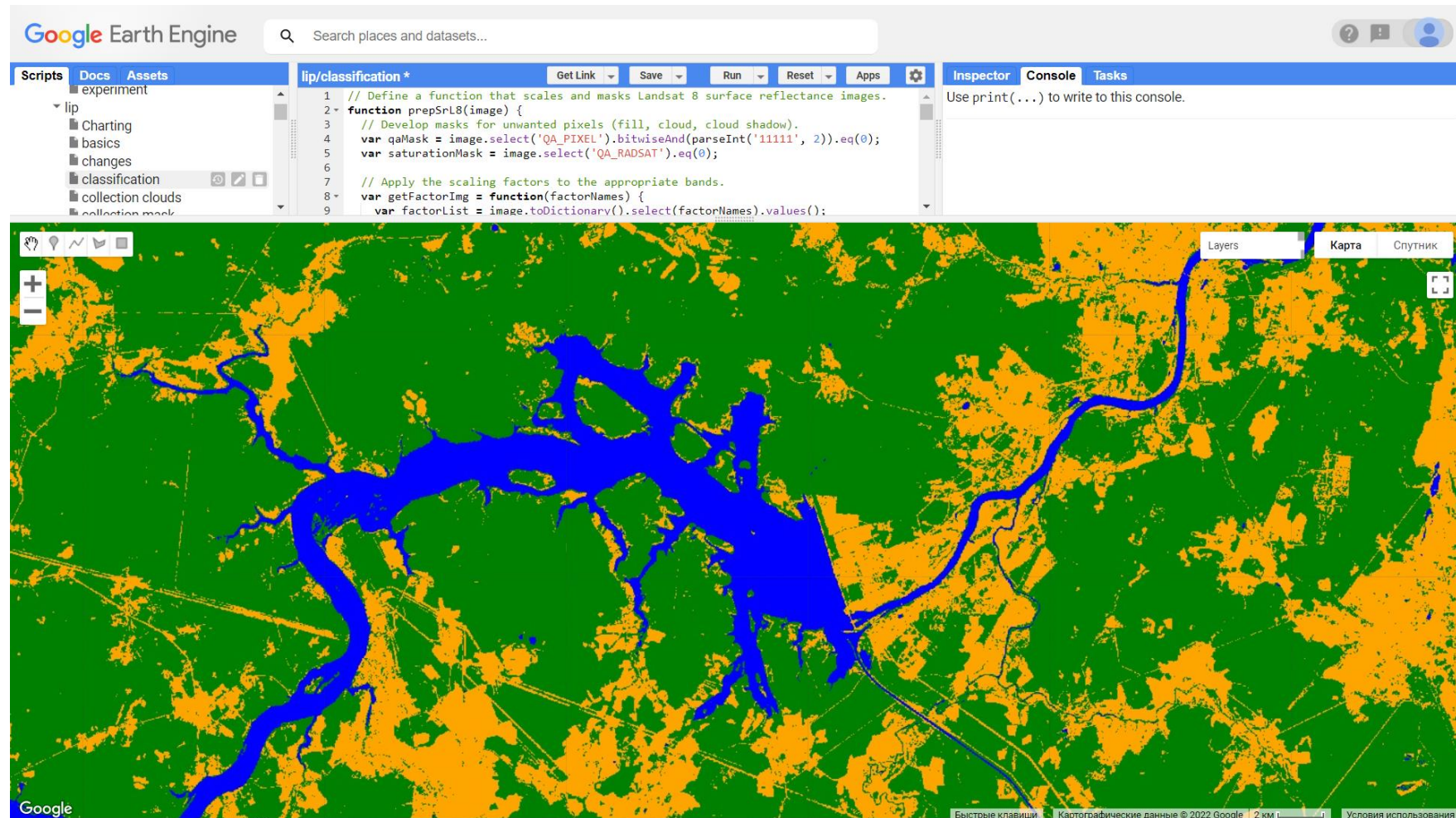
<https://code.earthengine.google.com/cdd574d1d7466208b1eb5bff5fb7c02e>



Land use classification

<https://developers.google.com/earth-engine/apidocs/ee-classifier-libsvm>

<https://code.earthengine.google.com/1828d8cf8bd900884f4a9746ee214252>



The screenshot displays the Google Earth Engine web interface. At the top, there is a search bar and navigation icons. The left sidebar shows a project tree with folders like 'lip/classification'. The main area is split into two panes: a code editor on the left and an inspector/console on the right. The code editor contains a JavaScript script for land use classification. The map below shows a satellite view of a landscape with a river network highlighted in blue. The land is classified into green and yellow/orange areas. The bottom of the interface includes a Google logo, keyboard shortcuts, copyright information, and a scale bar.

```
1 // Define a function that scales and masks Landsat 8 surface reflectance images.
2 function prepSrL8(image) {
3   // Develop masks for unwanted pixels (fill, cloud, cloud shadow).
4   var qaMask = image.select('QA_PIXEL').bitwiseAnd(parseInt('11111', 2)).eq(0);
5   var saturationMask = image.select('QA_RADSAT').eq(0);
6
7   // Apply the scaling factors to the appropriate bands.
8   var getFactorImg = function(factorNames) {
9     var factorList = image.toDictionary().select(factorNames).values();
```

