



Coding in Earth Engine

An Introduction using JavaScript in the Code Editor

David Gibson / 2019-09-16

Agenda

- Demo
- Earth Engine Subsystems
- Coding Concepts
- Coding Hands-On
- Next Steps

Next-generation Digital Earth

Michael F. Goodchild^{a,1}, Huadong Guo^b, Alessandro Annoni^c, Ling Bian^d, Kees de Bie^e, Frederick Campbell^f, Max Craglia^g, Manfred Ehlers^h, John van Genderenⁱ, Davina Jackson^b, Anthony J. Lewis^j, Martino Pesaresi^k, Gábor Remetey-Fülöpp^l, Richard Simpson^k, Andrew Skidmore^l, Changlin Wang^d, and Peter Woodgate^m

^aDepartment of Geography, University of California, Santa Barbara, CA 93106; ^bCenter for Earth Observation and Digital Earth, Chinese Academy of Sciences, Beijing 100094, China; ^cJoint Research Centre of the European Commission, 21027 Ispra, Italy; ^dDepartment of Geography, University at Buffalo, State University of New York, Buffalo, NY 14261; ^eFaculty of Geo-Information Science and Earth Observation, University of Twente, 7500 AE, Enschede, The Netherlands; ^fFred Campbell Consulting, Ottawa, ON, Canada K2H 5G8;

^gInstitute for GeoInformatics and Remote Sensing, University of Osnabrück, 49076 Osnabrück, Germany; ^hD_City Network, Newtown 2042 Australia; ⁱDepartment of Geography and Anthropology, Louisiana State University, Baton Rouge, LA 70803; ^jHungarian Association for Geo-Information, H-1122, Budapest, Hungary; ^kNextspace, Auckland 1542, New Zealand; and ^lCooperative Research Center for Spatial Information, Carlton South 3053, Australia

Data >> Bandwidth



“The supply of geographic information from satellite-based and ground-based sensors has expanded rapidly, encouraging belief in a new, fourth, or “big data,” paradigm of science that emphasizes international collaboration, data-intensive analysis, huge computing resources, and high-end visualization.”

-Goodchild et al. (2012)

“Often it turns out to be more efficient to move the questions than to move the data.”

-Jim Gray (1944-2007)



The
FOURTH
PARADIGM
DATA-INTENSIVE SCIENTIFIC DISCOVERY

EDITED BY TONY HEY, STEWART TANSLEY, AND KRISTIN TOLLE

Earth Engine Subsystems



Data

An exhaustive catalog of remote sensing datasets, including multispectral, radar, aerial, climate, land cover, and vector.



Computation

Colocated data storage and computation



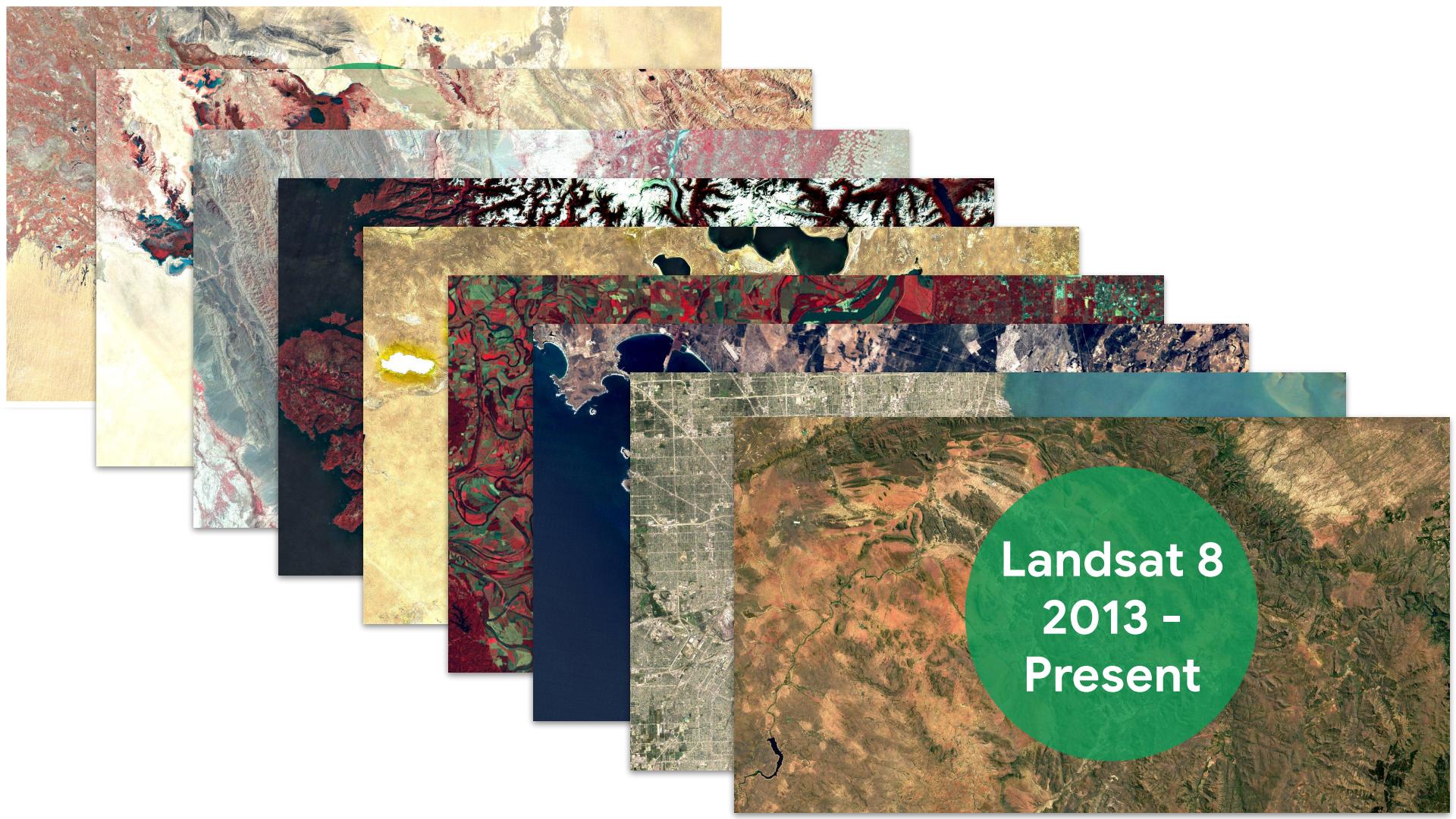
API

Rich JavaScript API, and support for Python, hosted on GitHub.



Web IDE

No software to download or keep up to date. All you need is a modest internet connection.



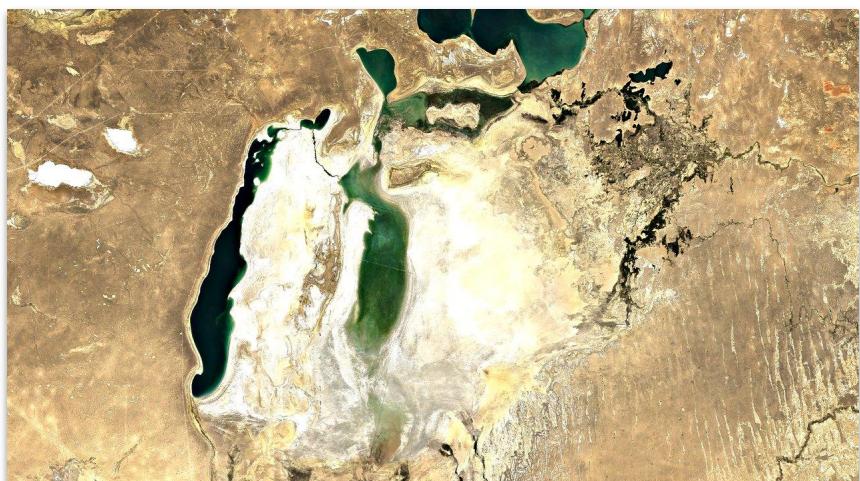
Landsat 8
2013 -
Present

Landsat

Satellite	Date Range	Instruments	Resolution (m/px)
Landsat 1	1972 - 1978	MSS	60
Landsat 2	1975 - 1982	MSS	60
Landsat 3	1978 - 1983	MSS	60
Landsat 4	1982 - 1993	MSS/TM	60 / 30
Landsat 5	1984 - 2012	MSS/TM	30
Landsat 7	1999 - present	ETM+	30 / 15 (pan)
Landsat 8	2013 - present	OLI/TIRS	30 / 15 (pan)



Landsat 4, 1980's



Landsat 8, 2015~



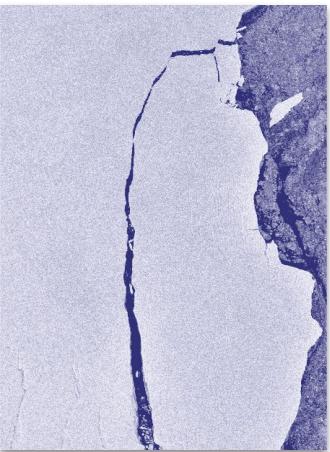
Sentinel-1
2014/04



Sentinel-2
2A: 2015/06
2B: 2017/03



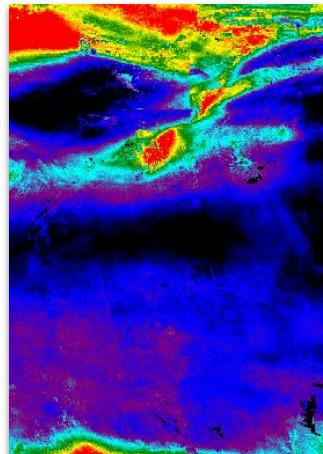
Sentinel-5P
2017/10

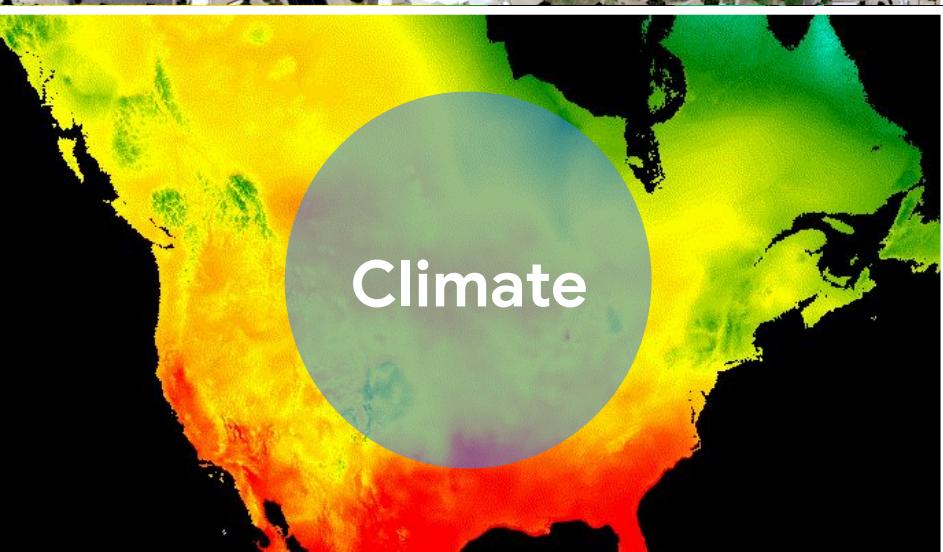
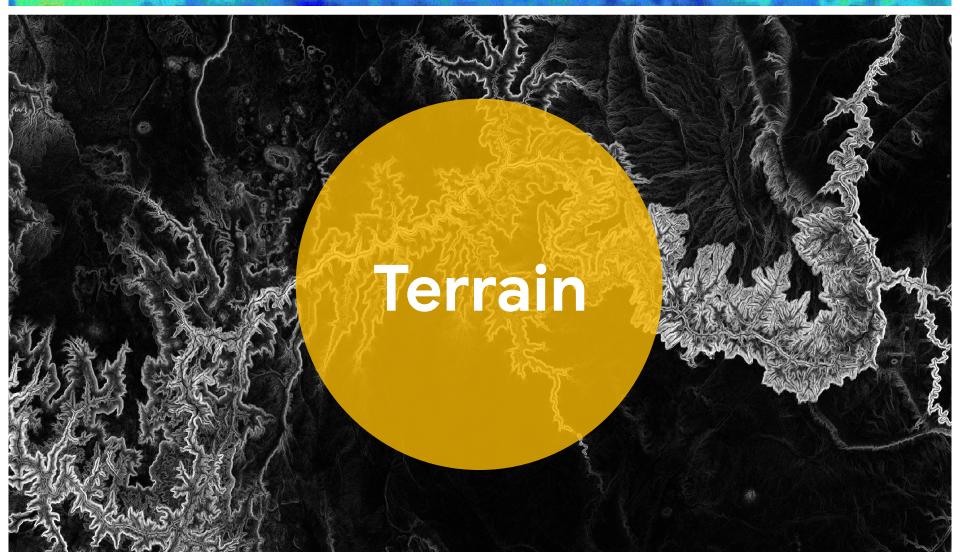
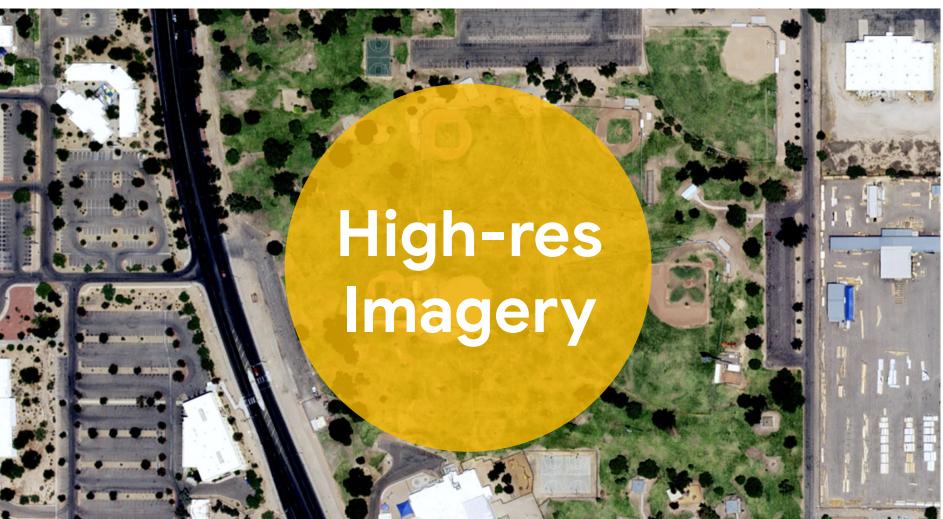
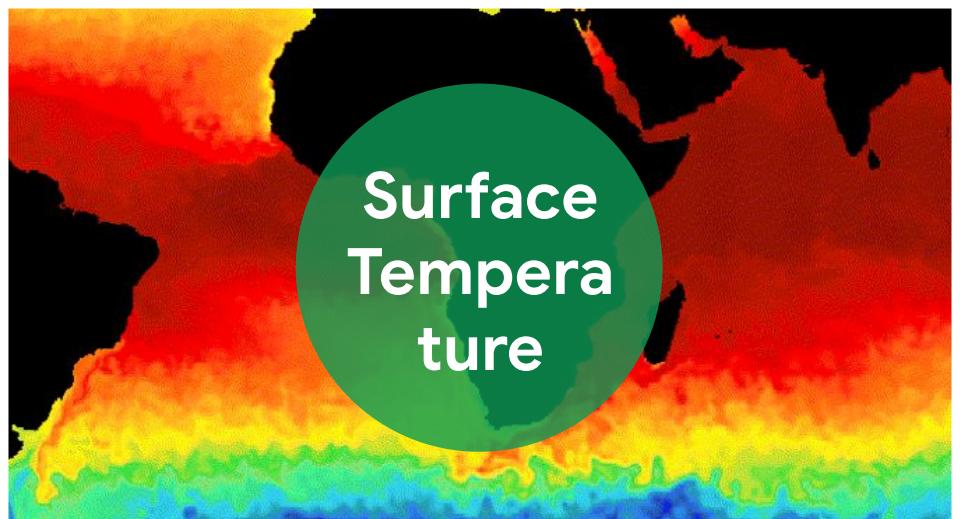


Snow and Ice
Oceans
Oil spills
Maritime activity
Deforestation
Natural disasters
Emergency response

Urban expansion
Land use/change
Agriculture
Water monitoring
Disaster mapping
Infectious diseases

Near Real-Time
[UV Aerosol Index](#),
[HCHO](#), [NO2](#), [O3](#),
[SO2](#)







Data Catalog



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



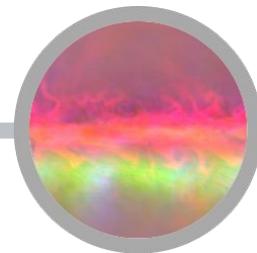
MODIS
250m daily



Vector Data
WDPA, Tiger



Terrain & Land Cover



Weather & Climate
NOAA NCEP, OMI, ...

> 500+ public datasets

> 10s of millions of images

> 25+ petabytes of data

> 1000s of new images every day

... and upload your own vectors and rasters

#GeoForGood19

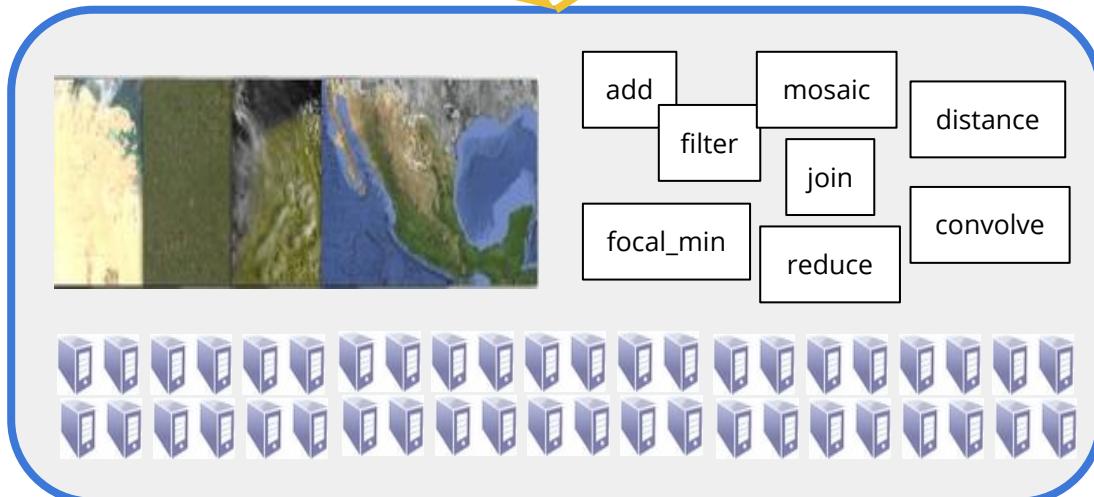


Computation

Geospatial
Datasets

Algorithmic
Primitives

Requests Results



Storage and Compute



API: Requests and Results

Requests

Compute Value

Generate Map

Fetch Map Tile

Perform Import/Export



Results

Numbers, Tables, other Data Structures

Map ID

Image

"OK"

"Error: ..."



Web IDE: the Code Editor

Workspace

for

Investigation

and

Collaboration

The screenshot shows the Google Earth Engine Web IDE interface. At the top, there's a search bar labeled "Search places and datasets..." and a navigation bar with "Google Earth Engine" and a user profile icon. Below the search bar is a menu bar with "Scripts", "Docs", and "Assets". A sidebar on the left lists "Owner (16)", "Writer", "Reader", "Examples" (with sub-options like "Image", "Image Collection", etc.), and "Cloud Masking" (with sub-options like "Landsat457 Surface Reflectance", "Landsat8 Surface Reflectance", etc.). The main area contains a code editor with the following JavaScript code:

```
1 // This example uses the Sentinel-2 QA band to cloud mask
2 // the collection. The Sentinel-2 cloud flags are less
3 // selective, so the collection is also pre-filtered by t
4 // CLOUDY_PIXEL_PERCENTAGE flag, to use only relatively
5 // cloud-free granule.
6
7 // Function to mask clouds using the Sentinel-2 QA band.
8 function maskS2clouds(image) {
9     var qa = image.select('QA60')
10
11     // Bits 10 and 11 are clouds and cirrus, respectively.
12     var cloudBitMask = 1 << 10;
13     var cirrusBitMask = 1 << 11;
14
15     // Both flags should be set to zero, indicating clear c
16     var mask = qa.bitwiseAnd(cloudBitMask).eq(0).and(
17         qa.bitwiseAnd(cirrusBitMask).eq(0))
18
19     // Return the masked and scaled data, without the QA ba
20     return image.updateMask(mask).divide(10000)
21     .select("B,*")
22     .copyProperties(image, ["system:time_start"])
23 }
```

Below the code editor is a satellite map of a coastal urban area, likely San Francisco. The map includes various layers and controls. At the bottom, there's a "Layers" panel with a "Sentinel2" layer selected, and "Map" and "Satellite" buttons. The bottom right corner shows copyright information: "Map data ©2019 Google | 2 km | Terms of Use | Report a map error".

... in 2012

Image > Global roadless area

Get Link Save Run Demo Clear Map

Pick an Example

- Hillshade
- Global roadless area
- Pretty Earth
- Download example

```
1 // Global roadless area
2 // Constants
3 var MAX_DISTANCE = 100000;
4 var MIN_DISTANCE = 10000;
```

The map displays the global distribution of roadless areas, primarily in developing countries where infrastructure is less dense. The green color indicates areas where roads are either non-existent or sparsely developed. The map is overlaid with a network of white lines representing existing roads. Labels on the map include Ireland, Kingdom, Bay of Biscay, North Atlantic Ocean, Portugal, Spain, Mediterranean Sea, Italy, France, Germany, Poland, Russia, Sea of Okhotsk, Japan, South Korea, East China Sea, Philippines, South China Sea, Vietnam, Malaysia, Indonesia, Australia, New Zealand, Coral Sea, Great Barrier Reef, South Pacific Ocean, and various cities like Rio de Janeiro, São Paulo, and Mexico City.

Layers Map Satellite

Google

Map data ©2012 Google, INEGI, MapLink, Tele Atlas - Terms of Use



Coding Concepts: Images

Image bands may have different transforms, data types, and pixel counts.

Image operations usually work independently across bands.

Images have a *mask* band.

Computed output pixels will often correspond to multiple input pixels.

ImageCollection ID, metadata

Image ID, metadata

Band

ID	B1
CRS	EPSG:32601
CRS Transform	[a,b,c,d,e,f]
Data Type	int16
Dimensions	1830 x 1830



Coding Concepts: Filter, Map and Reduce

Get an image

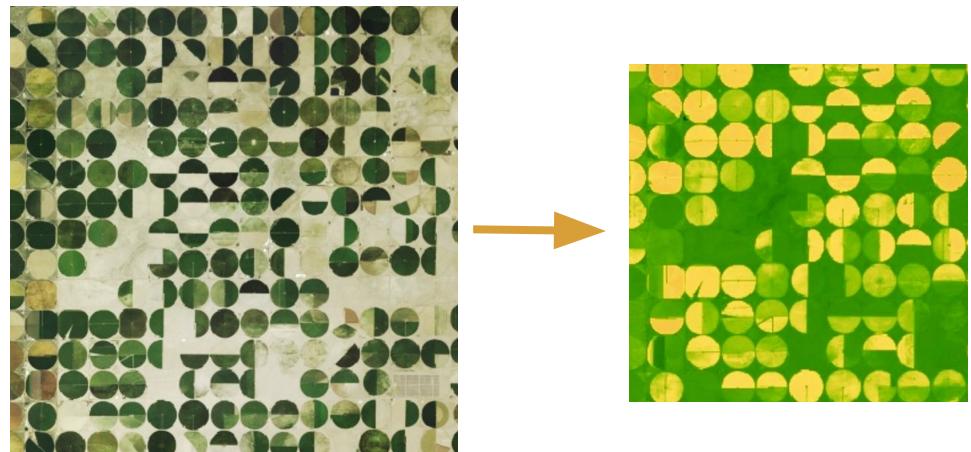


Pick your: projection, resolution,
bands, bounding-box, visualization

Coding Concepts

Apply an algorithm to an image

Use library functions or script your own

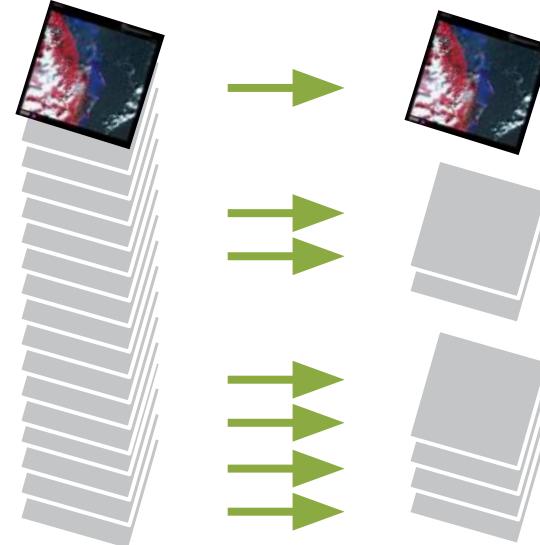


Coding Concepts

Apply an algorithm to an image

Filter a collection

Time, Space & Metadata Search



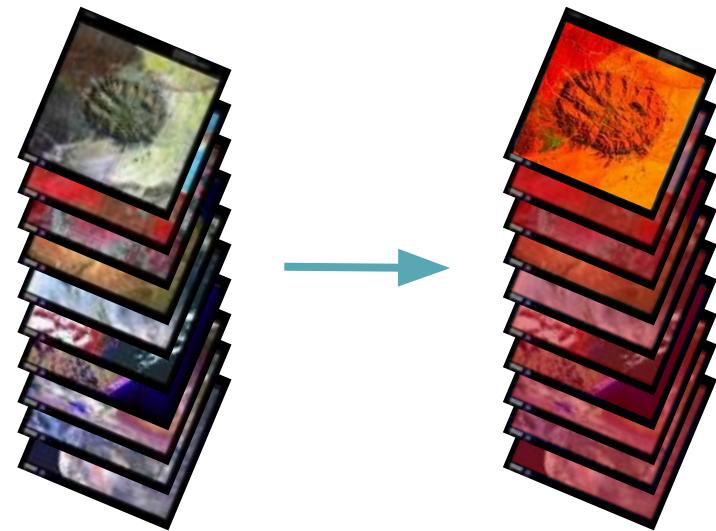
Coding Concepts

Apply an algorithm to an image

Filter a collection

Map an algorithm over a collection

$$N \rightarrow N$$



Coding Concepts

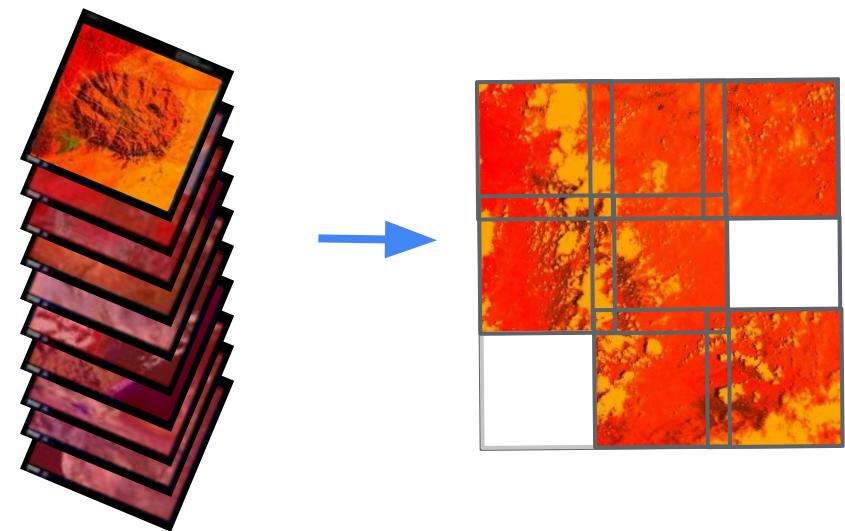
Apply an algorithm to an image

Filter a collection

Map an algorithm over a collection

Reduce a collection

$N \rightarrow 1$ or $N \rightarrow M$



Coding Concepts

Apply an algorithm to an image

Filter a collection

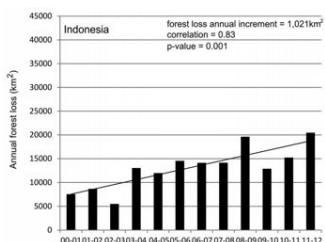
Map an algorithm over a collection

Reduce a collection

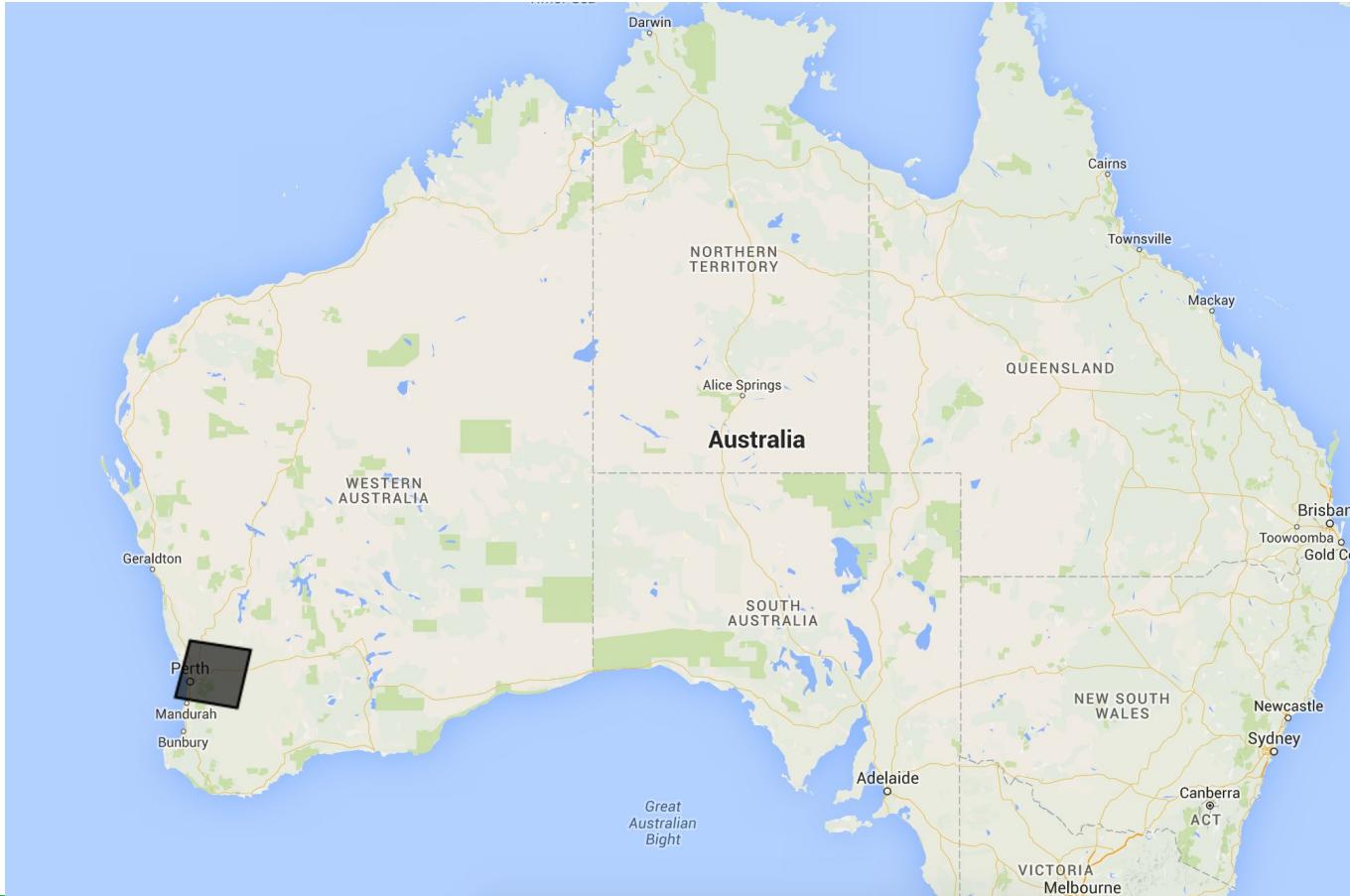
Compute aggregate statistics



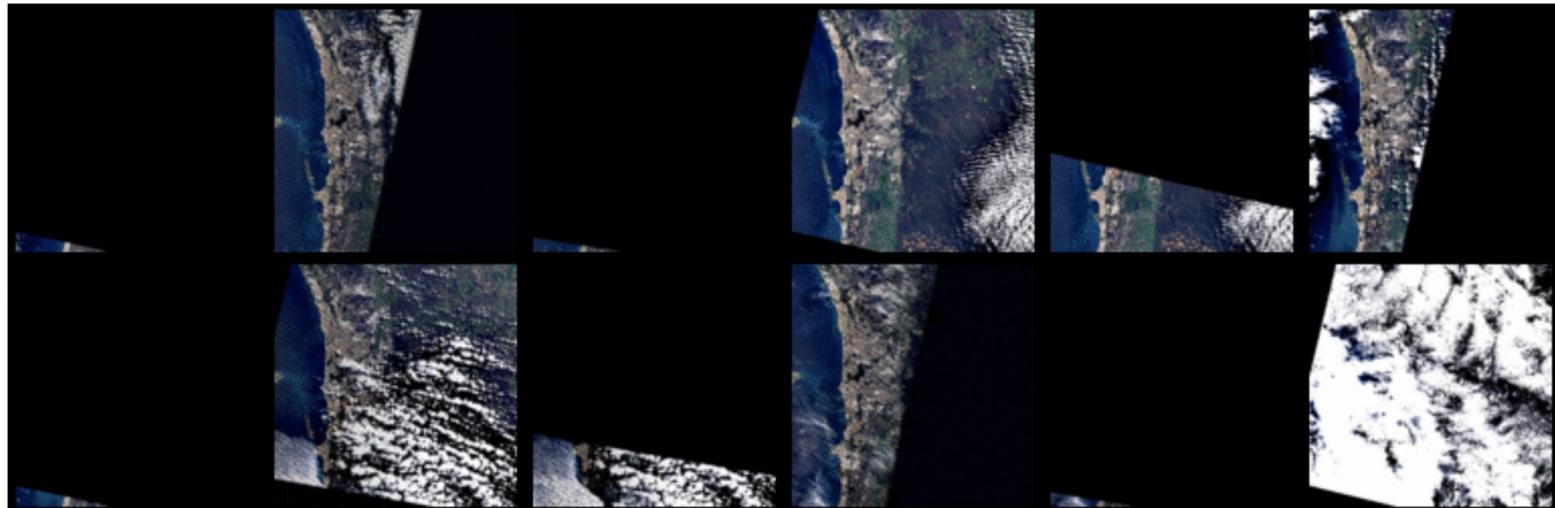
Gabon	1891	391	11898
Lithuania	1845	1226	40296
Cuba	1725	2271	68008
Mali	1694	0	1247103
Costa Rica	1653	382	11327
Czech Republic	1646	1331	46934
South Sudan	1635	38	460581
North Korea	1605	137	67695
Italy	1603	898	201331



Example: Filter, Map, and Reduce



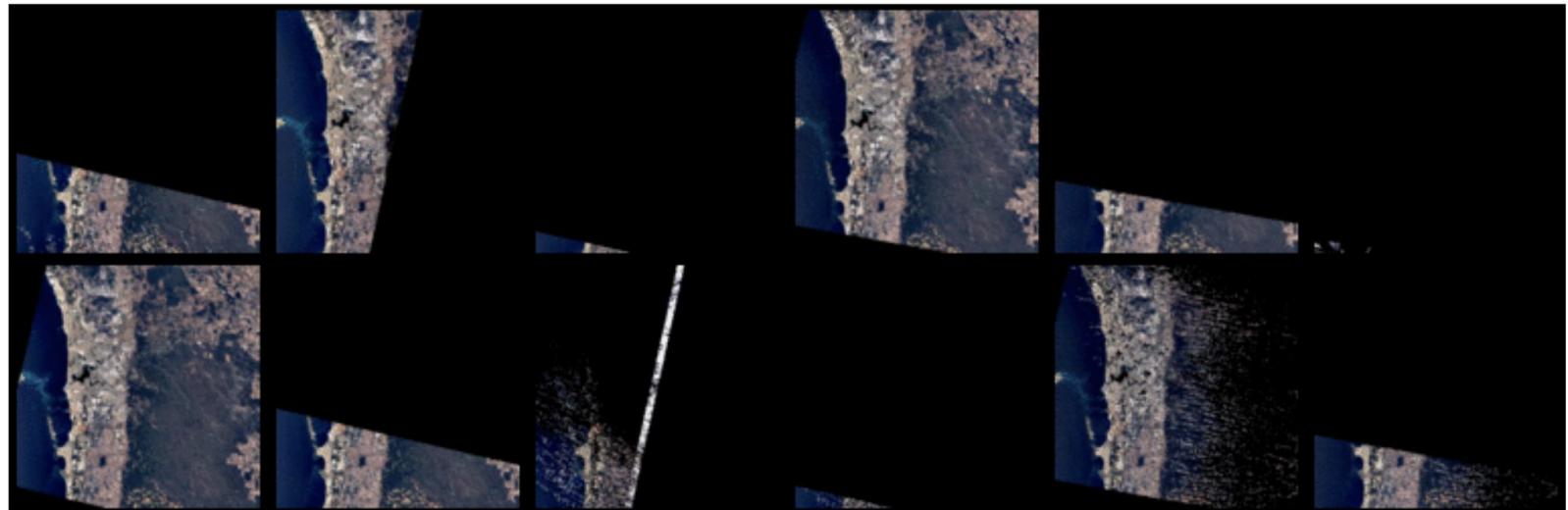
Core concept: Filter



Filter Condition:

"All L7 and L8 images acquired after 2013 that intersect polygon over Perth,
Australia."

Core concept: Map

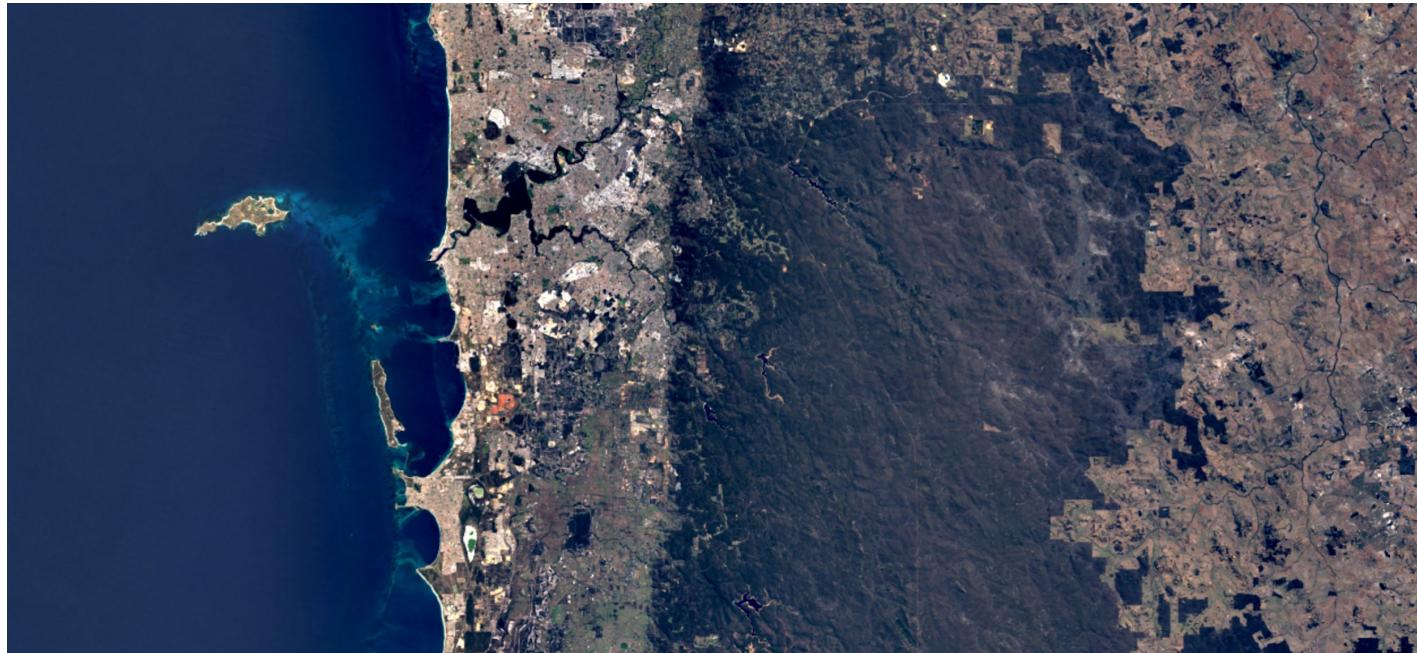


Map Operation: Band values for pixel at location → Cloud score at location

Earth Engine's built-in Landsat cloud score algorithm computes a cloud-likeness score for each pixel, letting you identify the cloudy ones.

#GeoForGood19

Core concept: Reduce



Reduce Operation: All pixel values & cloud scores → Consensus pixel value

Choose the best of the cloud-free pixels and make final mosaic.

#GeoForGood19

Using the Code Editor

[**bit.ly/ee2019ce**](https://bit.ly/ee2019ce)

Setting Up: Home Folder, Default Repository.

Editing and Saving Scripts. Get Link.

Console and Inspector.

Online docs and Help.

Introduction to JavaScript

Definitions and rendering:

```
var variable = expression;  
print(expression);
```

Data Structures:

List: [expression, expression, ...]

Dictionary: {key1: value1, key2: value2, ...}

Syntax highlighting and documentation.

Introduction to JavaScript: Functions

Pieces of computation to execute "later".

REUSE At multiple points in the script

DEFER Until we know the object(s) of interest

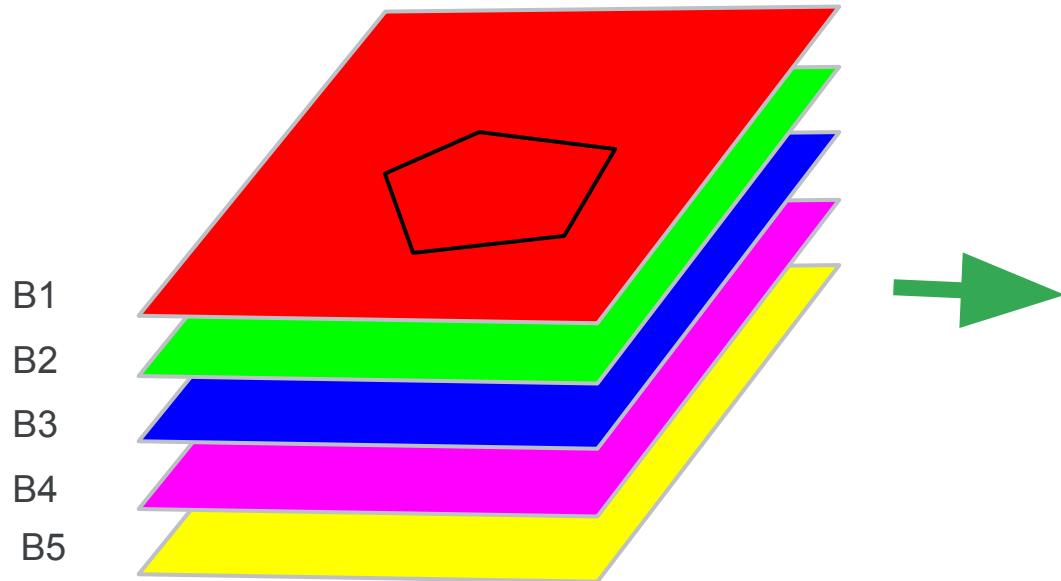
CALLBACK When something happens, eg click on map

Examples:

- Predefined: Earth Engine Library
- Calculation: Normalised Difference
- Transform Image: Add Bands, Remove Clouds, Gather Statistics

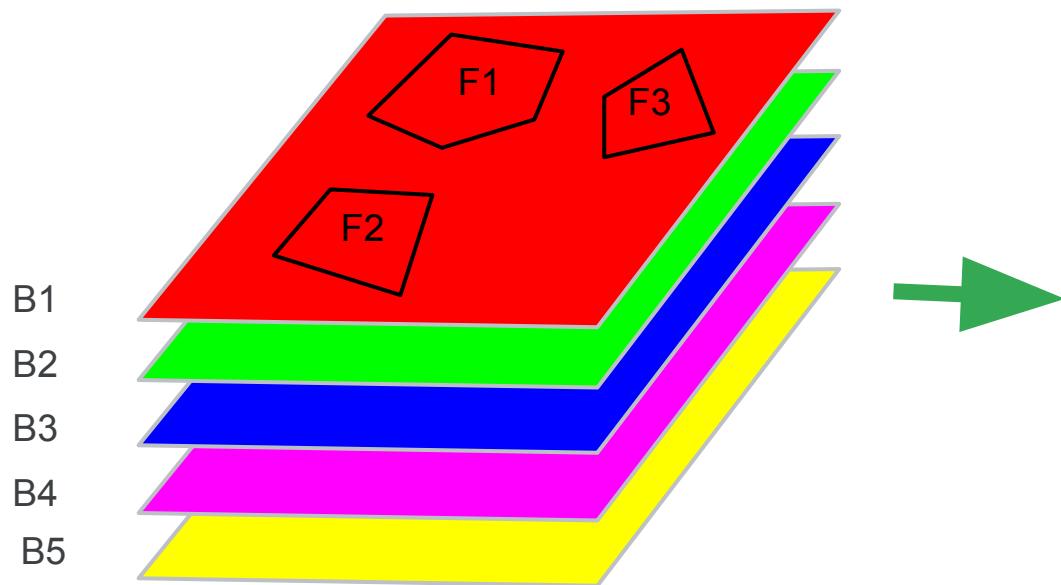
```
function name(a, b) {  
    // do stuff  
    return expression;  
}  
  
// later ...  
  
var result = name(x, y);
```

Reduce Region



Dictionary
{
B1: 8.3,
B2: 14,
B3: 176,
B4: 1.6,
B5: 7
}

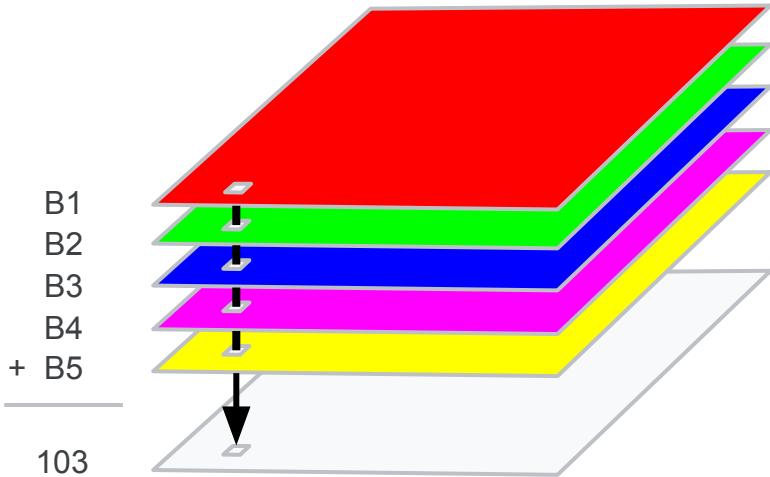
Reduce Regions



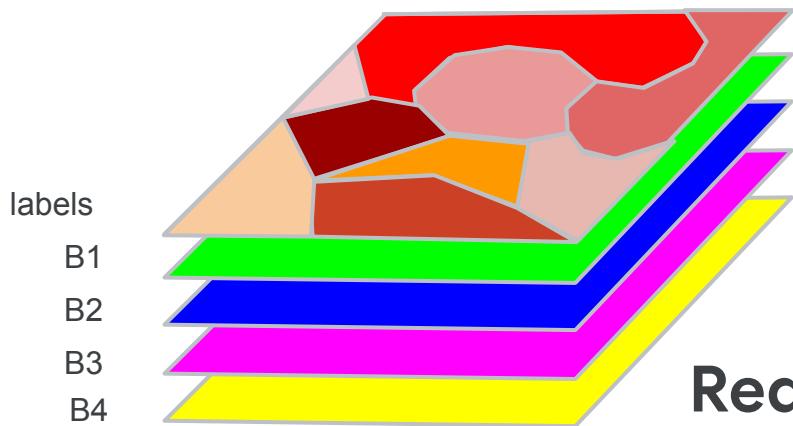
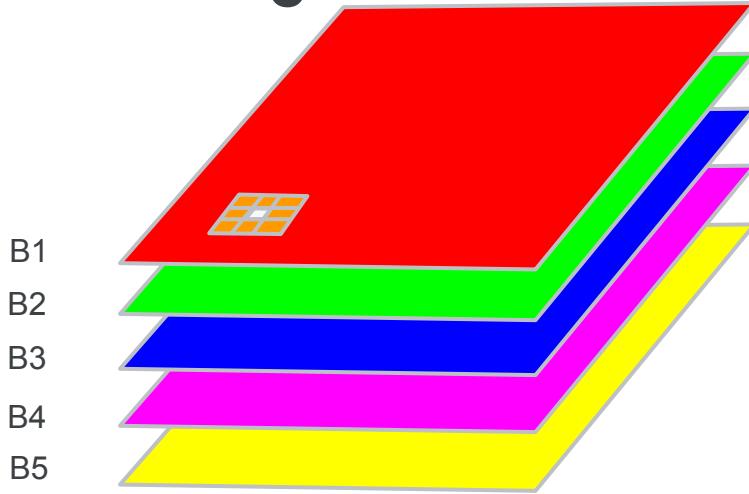
FeatureCollection

	B1	B2	B3	B4	B5
F1					
F2					
F3					

Reduce Bands



Reduce Neighborhood



Reduce To Vectors

FeatureCollection

	B1	B2	B3	B4
F1				
F2				
F3				
...	#GeoForGood19			

Reducers in Earth Engine

8 ways to reduce

- Image.reduce
- Image.reduceNeighborhood
- Image.reduceRegion
- Image.reduceRegions
- Image.reduceToVectors
- ImageCollection.reduce
- FeatureCollection.reduceColumns
- FeatureCollection.ReduceToImage

40+ reducers

- Reducer.allNonZero
- Reducer.and
- Reducer.anyNonZero
- Reducer.count
- Reducer.countEvery
- Reducer.histogram
- Reducer.intervalMean
- Reducer.linearFit
- Reducer.linearRegression
- Reducer.max
- Reducer.mean
- Reducer.median
- Reducer.min
- Reducer.minMax
- Reducer.mode
- Reducer.or
- Reducer.percentile
- Reducer.product
- Reducer.sampleStdDev
- Reducer.sampleVariance
- Reducer.stdDev
- Reducer.sum
- Reducer.toCollection
- Reducer.toList
- #GeoForGood19
- Reducer.variance

Objects and Computations

```
// Construct an object
var eighties = ee.DateRange('1980-01-01', '1990-01-01');

// Objects contain predefined functions
print(eighties.intersection(ee.DateRange('1970'))
      .start()); // "chain" after intersection
```

Not: `intersection(eighties, ee.DateRange('1970'))`

All ee.* objects are *computations* which are sent to the server.

Functions for Filter, Map & Reduce

```
var filtered = L8.filterDate('2016-05-01', '2016-05-15')
    .filterBounds(roi);      // Images -> fewer Images

function addNDVI(image) {
  var ndvi = image.normalizedDifference(['B8', 'B4']); // NIR, Red
  return image.addBands(ndvi);
}

var with_ndvi = filtered.map(addNDVI); // Images -> Images with NDVI
                                         // Images -> single Image

Map.addLayer(filtered.reduce(ee.Reducer.median()), rgb_vis, 'Median');
```

The 20 Most Useful Library Functions

Render print, Map.addLayer

Constructors ee.Array, ee.Dictionary, ee.Number, ee.String, ee.Date(Range)

ee.Image addBands, select, reduceRegion(s), updateMask, add/max/sin/...

ee.Collection filterDate, filterBounds, map, mosaic, first

ee.List iterate, map

← → ⌂ ⌄ 🔒 https://developers.google.com/earth-engine

See the Docs tab, and Guides and Reference:

Google Earth Engine API

GUIDES

REFERENCE

TUTORIALS

EDU

Back to Greenest-Pixel Composite

A few more functions:

- `select` bands from image
- `qualityMosaic` choose pixels by quality score
- `updateMask` choose pixels to display

Recap:

- Functions and deferred execution
- Filter, Map, and Reduce
- Interactive Coding

Structure of a Script

Imports

Definitions: var ..., function ...

Rendering: print, Map.addLayer,
Export, ui.*

Nothing happens without rendering!

New definitions allowed inside functions.

Caution: rendering inside functions.

The screenshot shows a code editor with a script containing 23 numbered lines. The code uses the GEE API to filter a Sentinel-2 collection, add NDVI, mosaic images, and render them. Three green boxes with labels point to specific sections: 'Imports' points to the top section where variables are defined; 'Definitions' points to the middle section where a function is defined; and 'Rendering' points to the bottom section where the final rendering commands are located.

```
1 Imports (2 entries) ┌─────────────────┐
2   ▶ var s2: ImageCollection "Sentinel-2 MSI: Multispectral in
3   ▶ var roi: Point (14.39, 50.05) ⚙ ⚙
4
5 2 var filtered = s2.filterDate('2016-04-01', '2016-07-01');
6
7 4 var rgb_vis = {
8   5   min: 0,
9   6   max: 2500,
10  7   gamma: 1.4,
11  8   bands: ['B4', 'B3', 'B2']
12  9 };
13
14 11 function addNDVI(image) {
15   12   var ndvi = image.normalizedDifference(['B8', 'B4']);
16   13   return image.addBands(ndvi);
17  14 }
18
19 15 var with_ndvi = filtered.map(addNDVI);
20 16 var greenest = with_ndvi.qualityMosaic('nd');
21
22 18 Map.addLayer(greenest, rgb_vis, 'RGB (greenest pixel)');
23 19 // Show a chart of NDVI in the region of interest
24 20 print(ui.Chart.image.series(with_ndvi));
25 21
26 22 Map.setCenter(14.3, 50, 11); // Prague
```

Finding Problems

Check Intermediate Results

- print(), addLayer, Inspector

- Write tests for post-conditions

Delete Code

- Remove half the operations

- Find smallest failing example

Use Revision History

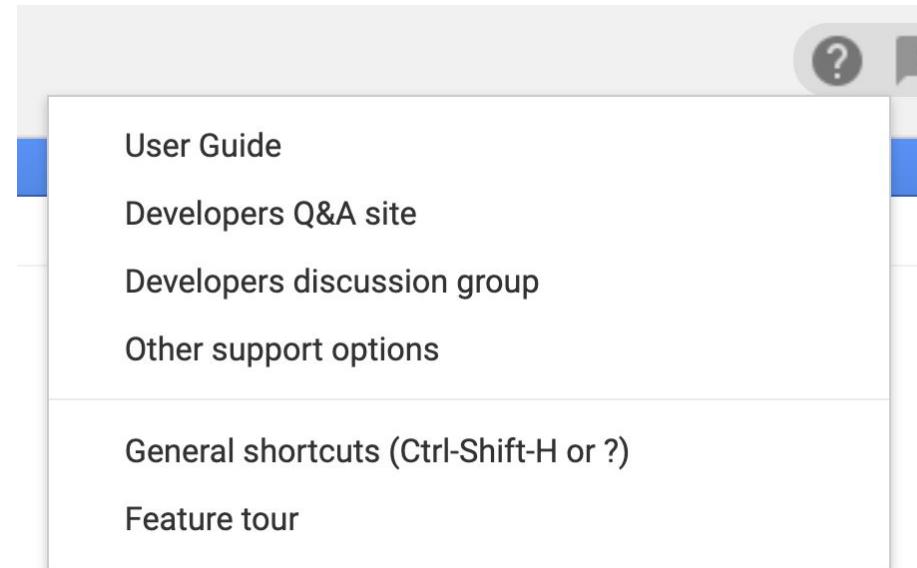
Finding Help

Primary Help Menu:

- User Guide
 - Tutorials
- GIS StackExchange
- Developer Forum

Learn (and steal) from other code:

- Examples
- Google Search



Where Next?

INTRODUCTORY

- Datasets Tour & Data Upload **Mon**
- Make your own Earth Engine App **Mon, Wed**
- Big Data Structures: Images and Features **Tue**
- Import/Export **Wed, Thu**
- Earth Engine Basics through Applied Examples
(Night-time light, Agriculture) **Thu**
- Apps Showcase **Thu**

ANALYSIS

- Reducers **Mon, Tue**
- Python API & Colab **Tue, Wed**
- Tables & Joins **Tue, Wed**
- Classification and Regression **Tue, Wed**
- Collections and map() **Tue**

VISUALIZATION

- Animations in Earth Engine **Tue, Wed**
- User Interface Coding **Thu**
- User Interface and Apps **Wed, Thu**

