# Google Earth Engine

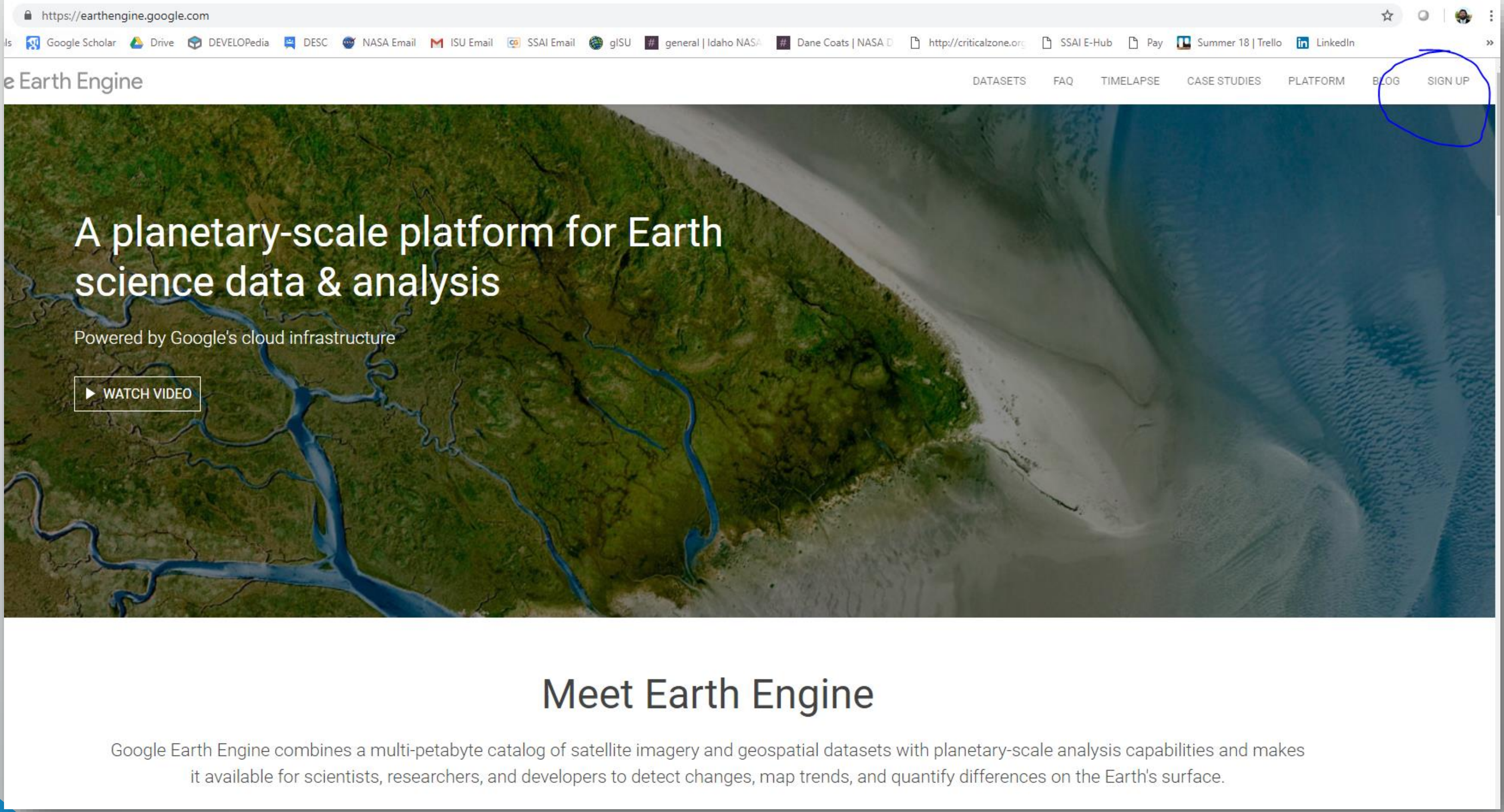
## Taking Geoprocessing into the cloud

A hands-on experience

Dane Coats

# Signing up for Google Earth Engine

## https://earthengine.google.com/



A planetary-scale platform for Earth science data & analysis

Powered by Google's cloud infrastructure

Meet Earth Engine

Google Earth Engine combines a multi-petabyte catalog of satellite imagery and geospatial datasets with planetary-scale analysis capabilities and makes it available for scientists, researchers, and developers to detect changes, map trends, and quantify differences on the Earth's surface.

# Why Google Earth Engine?

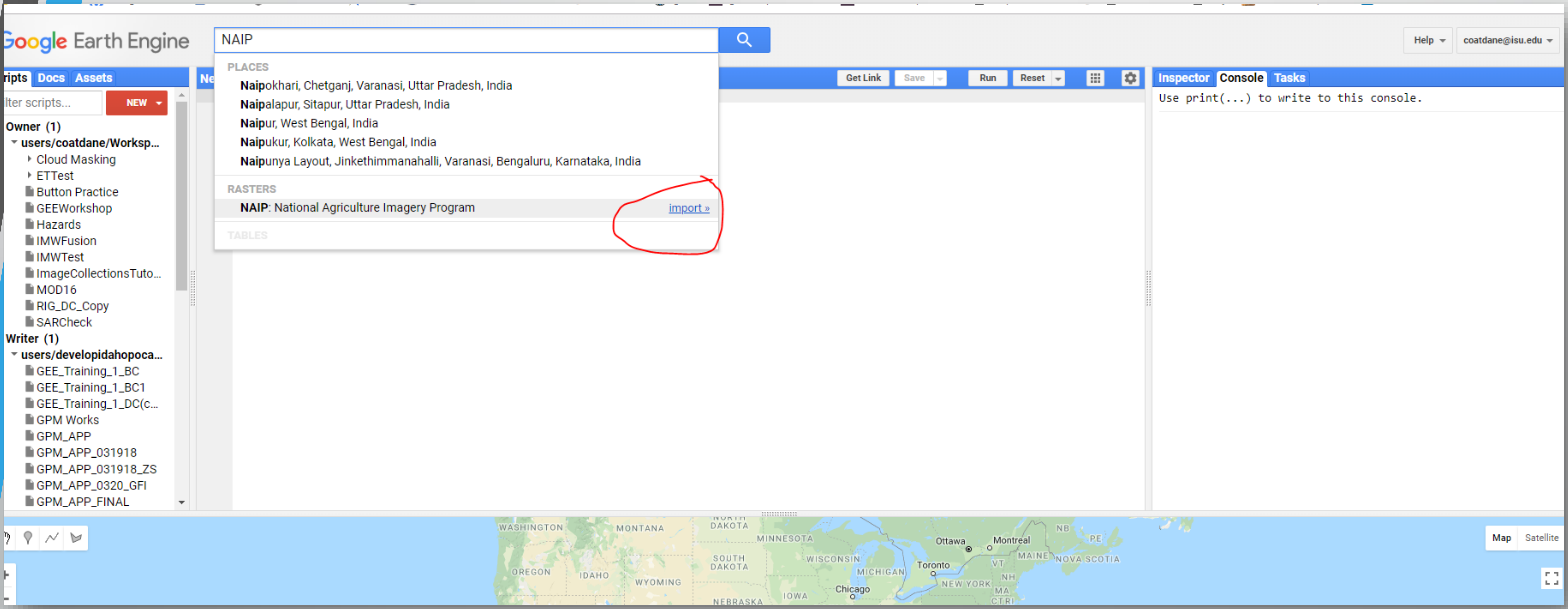
**https://earthengine.google.com/**

- Powerful cloud based computation – built in multiprocessing
- Remotely access data – no download required, most data automatically ingested by googles data servers
- Ability to upload data
- Collaborative environment – work with colleagues remotely with built in git features and version control.
- App interface for end users – in development / beta release RIGHT NOW!

# Agenda

- Import imagery from Google Earth Engine’s cloud servers
- Define area of interest (Points and polygon)
- Display data in interesting True and False Color
- Create training data for supervised classification
- Classify!

# Step 1 – Import imagery



NAIP

PLACES

**Naip**okhari, Chetganj, Varanasi, Uttar Pradesh, India
**Naip**alapur, Sitapur, Uttar Pradesh, India
**Naip**ur, West Bengal, India
**Naip**ukur, Kolkata, West Bengal, India
**Naip**unya Layout, Jinkethimmanahalli, Varanasi, Bengaluru, Karnataka, India

RASTERS

**NAIP**: National Agriculture Imagery Program import »

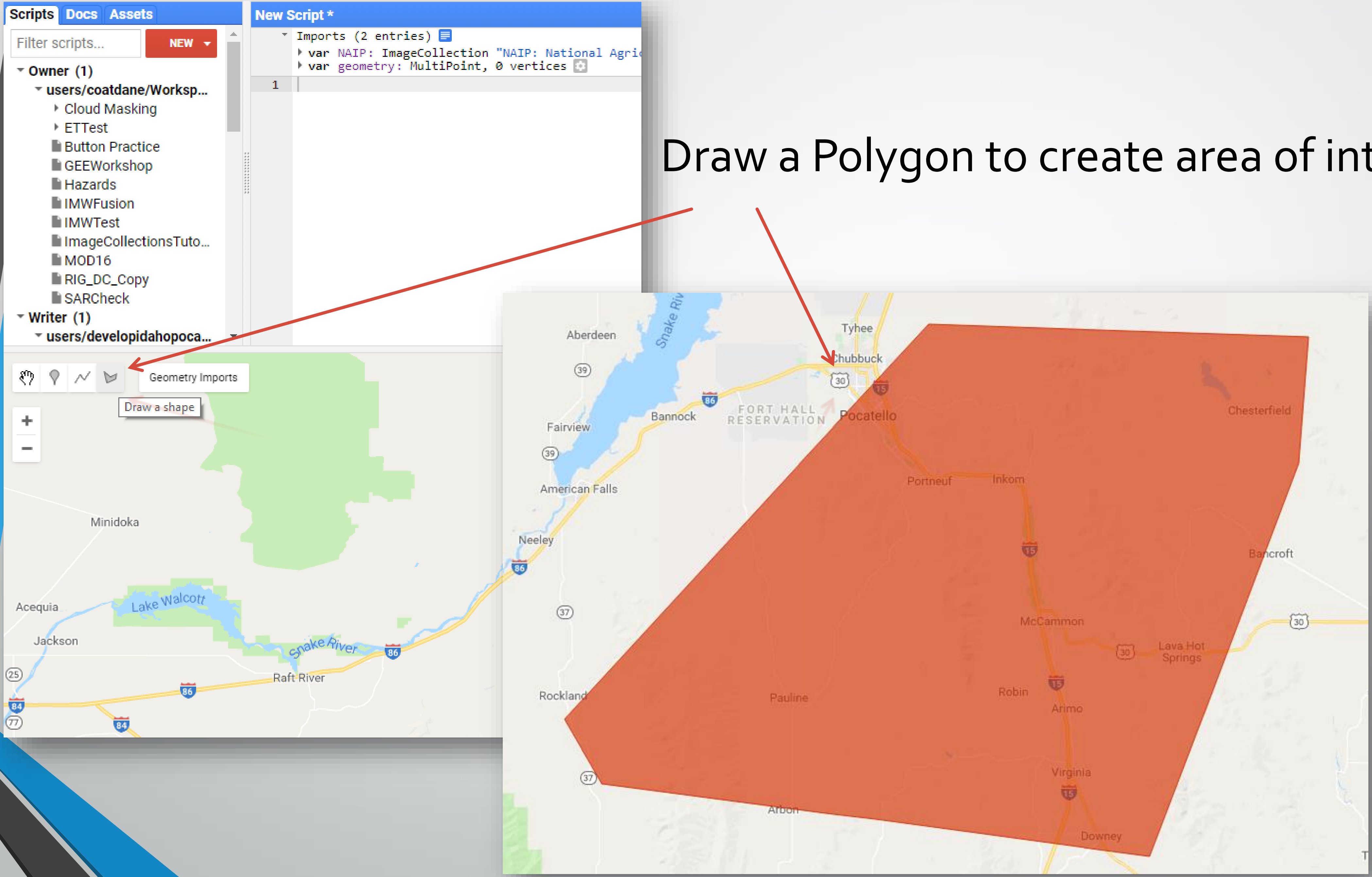
TABLES

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

Owner (1)
users/coatdane/Worksp...
Cloud Masking
ETTest
Button Practice
GEEWorkshop
Hazards
IMWFusion
IMWTest
ImageCollectionsTuto...
MOD16
RIG_DC_Copy
SARCheck
Writer (1)
users/developidahopoca...
GEE_Training_1_BC
GEE_Training_1_BC1
GEE_Training_1_DC(c...
GPM Works
GPM_APP
GPM_APP_031918
GPM_APP_031918_ZS
GPM_APP_0320_GFI
GPM_APP_FINAL

- Fill in search bar - > click **IMPORT**

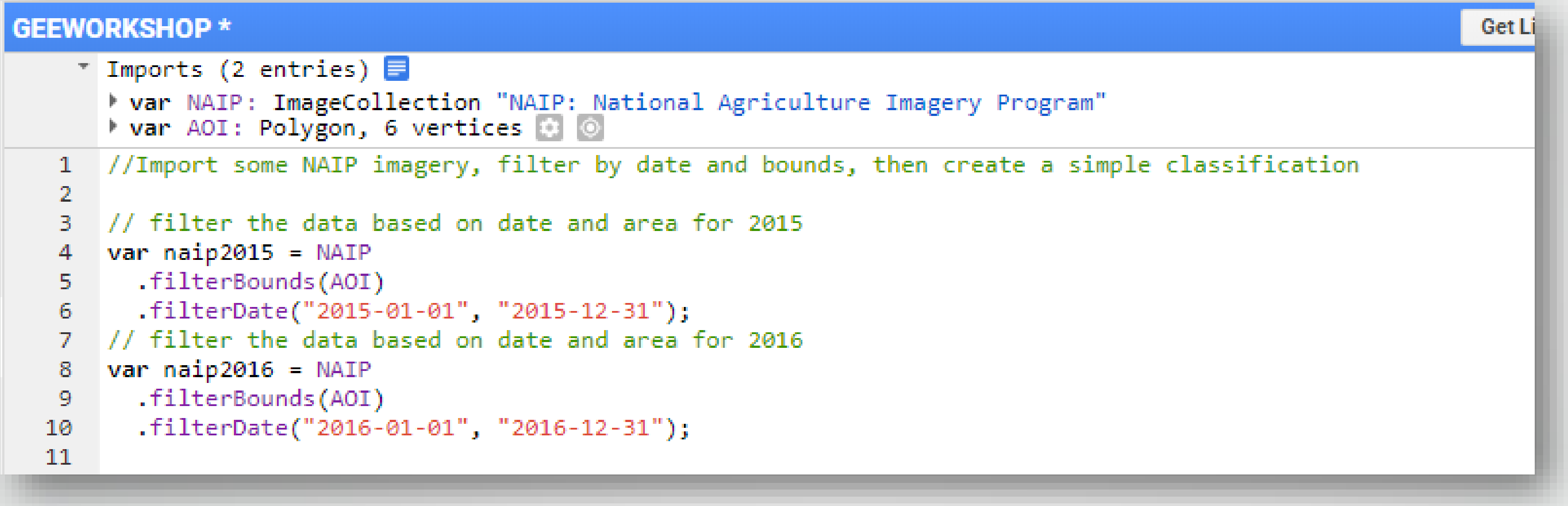
# Draw a Polygon to create area of interest



```
Imports (2 entries)
▸ var NAIP: ImageCollection "NAIP: National Agri
▸ var geometry: MultiPoint, 0 vertices
```

- Owner (1)
  - users/coatdane/Worksp...
    - Cloud Masking
    - ETTest
    - Button Practice
    - GEEWorkshop
    - Hazards
    - IMWFusion
    - IMWTest
    - ImageCollectionsTuto...
    - MOD16
    - RIG_DC_Copy
    - SARCheck
- Writer (1)
  - users/developidahopoca...

# Displaying Maps

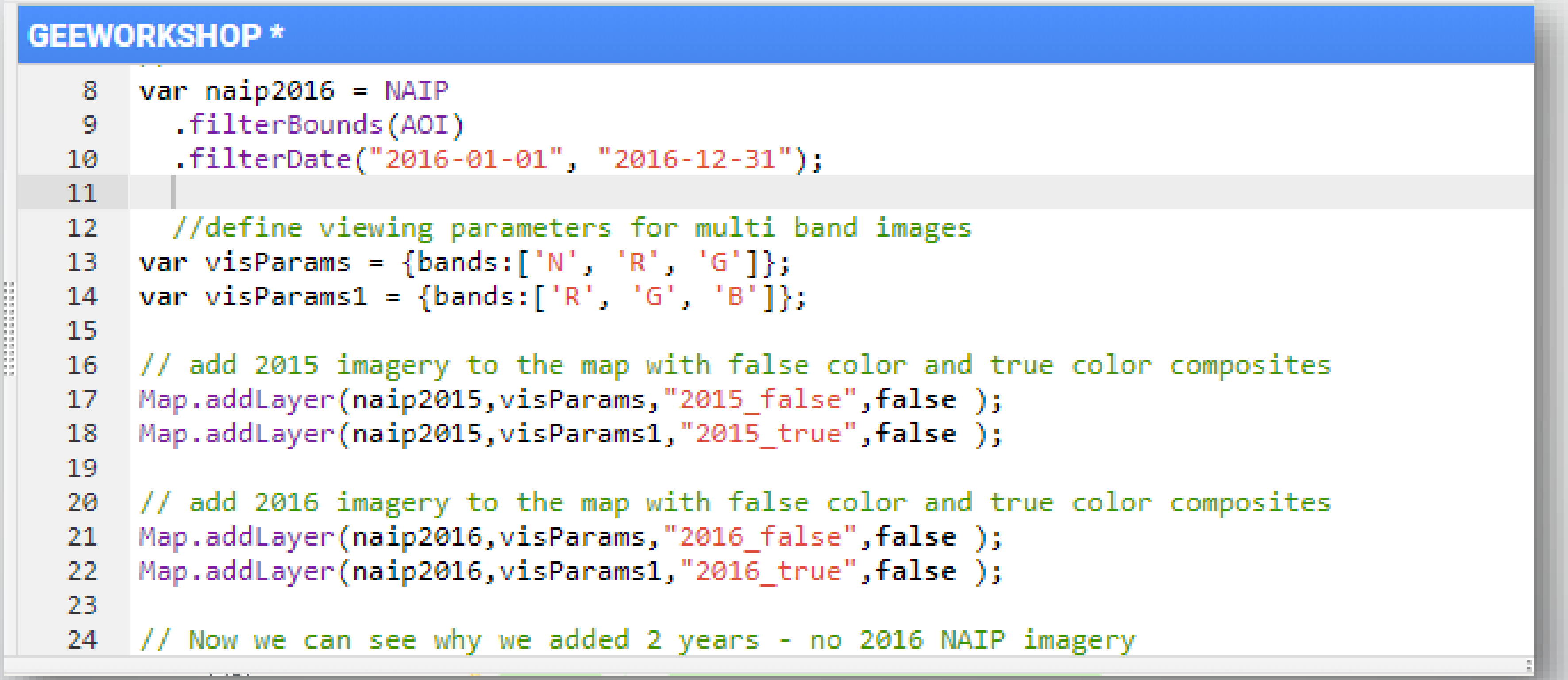


```
Imports (2 entries)
 var NAIP: ImageCollection "NAIP: National Agriculture Imagery Program"
 var AOI: Polygon, 6 vertices
```

```
//Import some NAIP imagery, filter by date and bounds, then create a simple classification

// filter the data based on date and area for 2015
var naip2015 = NAIP
  .filterBounds(AOI)
  .filterDate("2015-01-01", "2015-12-31");
// filter the data based on date and area for 2016
var naip2016 = NAIP
  .filterBounds(AOI)
  .filterDate("2016-01-01", "2016-12-31");

```

- **polygon drawn, image collection imported**
- **Need to FILTER results!**
- **Filter by our area, and filter by date bounds**
- **Can anyone tell me why we are looking at two years of NAIP?**

# Displaying Maps

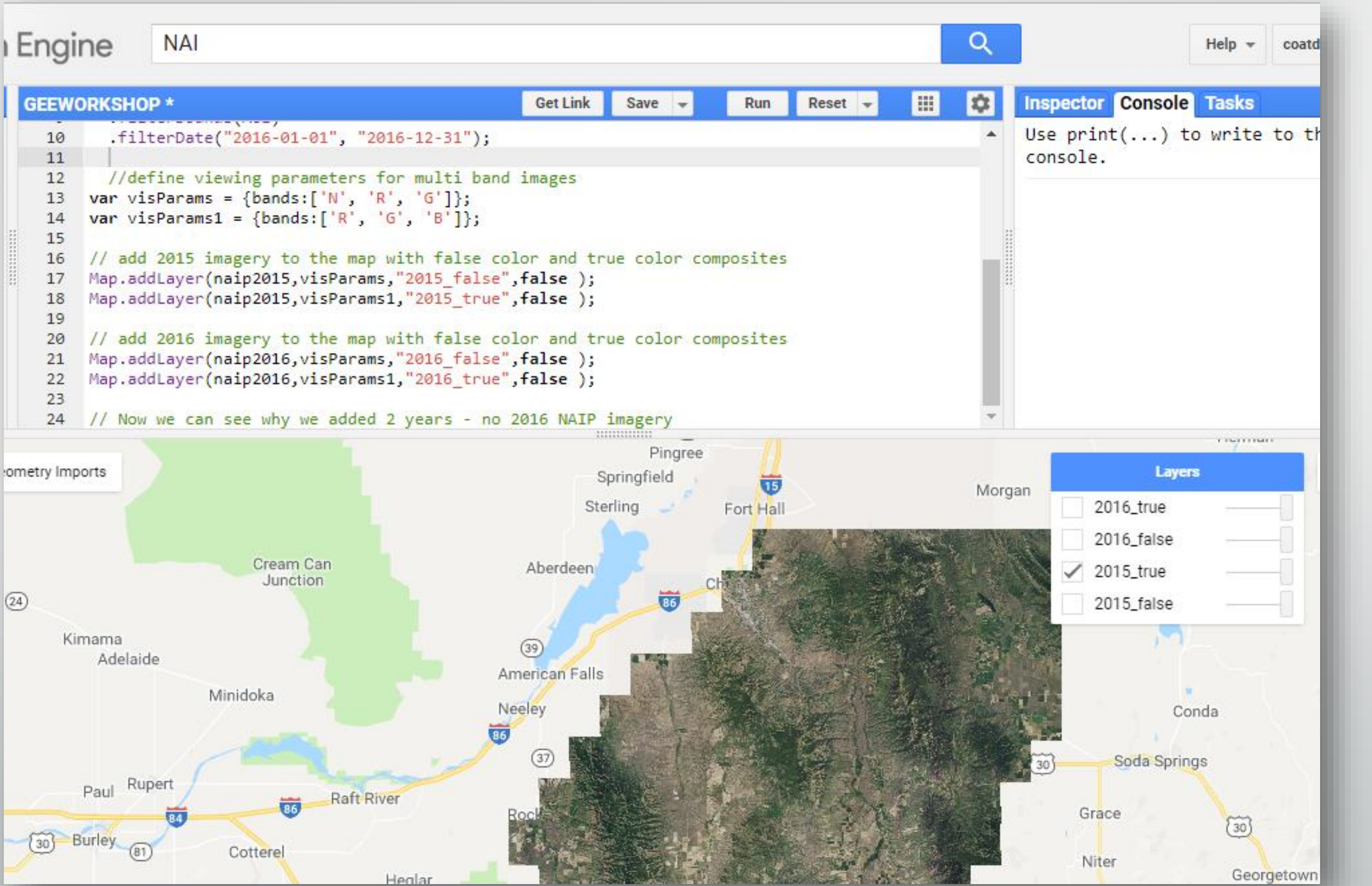


GEEWORKSHOP *

```
var naip2016 = NAIP
  .filterBounds(AOI)
  .filterDate("2016-01-01", "2016-12-31");

  //define viewing parameters for multi band images
var visParams = {bands:['N', 'R', 'G']};
var visParams1 = {bands:['R', 'G', 'B']};

// add 2015 imagery to the map with false color and true color composites
Map.addLayer(naip2015,visParams,"2015_false",false );
Map.addLayer(naip2015,visParams1,"2015_true",false );

// add 2016 imagery to the map with false color and true color composites
Map.addLayer(naip2016,visParams,"2016_false",false );
Map.addLayer(naip2016,visParams1,"2016_true",false );

// Now we can see why we added 2 years - no 2016 NAIP imagery
```

- **First we want to add some bands to visualize Color, and later vegetation –Infrared, Red, and Green, Blue bands relevant for that**
- **Use Map.addLayer**
- **Congratulations you've made a map!**

# Selecting Layers

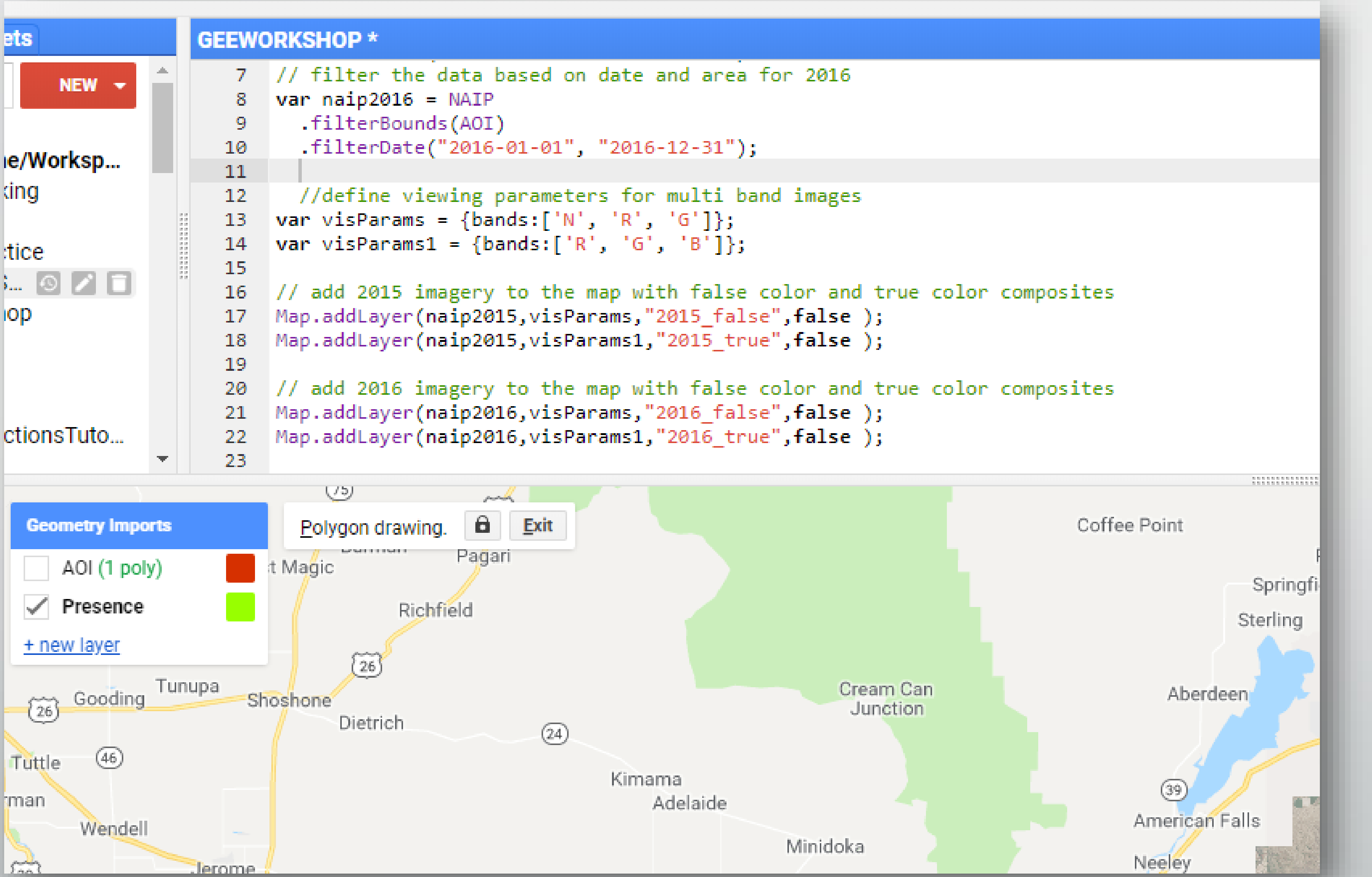


```
  .filterDate("2016-01-01", "2016-12-31");

  //define viewing parameters for multi band images
var visParams = {bands:['N', 'R', 'G']};
var visParams1 = {bands:['R', 'G', 'B']};

// add 2015 imagery to the map with false color and true color composites
Map.addLayer(naip2015,visParams,"2015_false",false );
Map.addLayer(naip2015,visParams1,"2015_true",false );

// add 2016 imagery to the map with false color and true color composites
Map.addLayer(naip2016,visParams,"2016_false",false );
Map.addLayer(naip2016,visParams1,"2016_true",false );

// Now we can see why we added 2 years - no 2016 NAIP imagery
```

```
Use print(...) to write to th
console.
```

Layers
- ☐ 2016_true
- ☐ 2016_false
- ☑ 2015_true
- ☐ 2015_false

# Adding Training Points



GEEWORKSHOP *

```
// filter the data based on date and area for 2016
var naip2016 = NAIP
  .filterBounds(AOI)
  .filterDate("2016-01-01", "2016-12-31");

  //define viewing parameters for multi band images
var visParams = {bands:['N', 'R', 'G']};
var visParams1 = {bands:['R', 'G', 'B']};

// add 2015 imagery to the map with false color and true color composites
Map.addLayer(naip2015,visParams,"2015_false",false );
Map.addLayer(naip2015,visParams1,"2015_true",false );

// add 2016 imagery to the map with false color and true color composites
Map.addLayer(naip2016,visParams,"2016_false",false );
Map.addLayer(naip2016,visParams1,"2016_true",false );
```

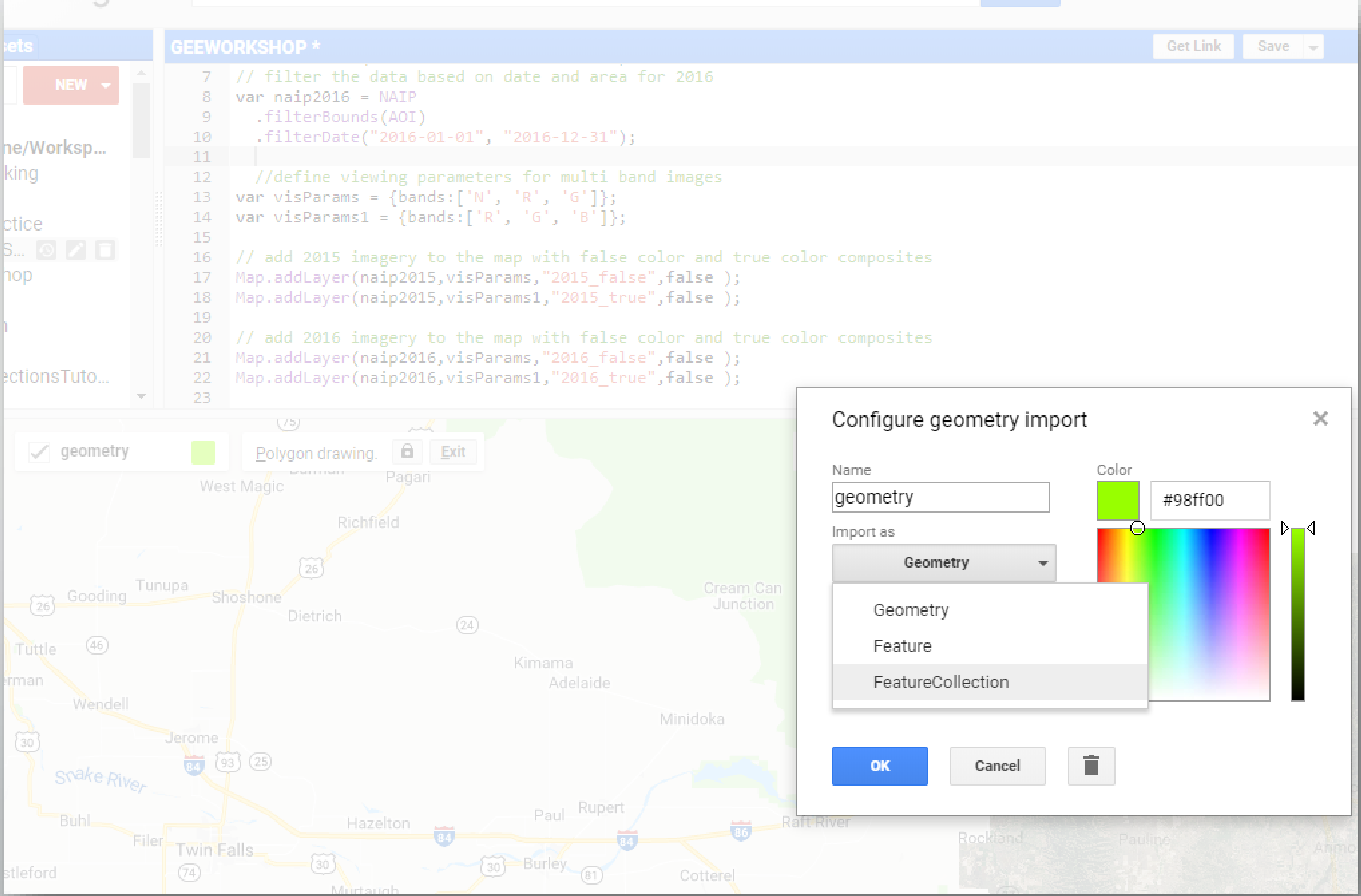
Geometry Imports

| | | |
|---|---|---|
| □ | AOI (1 poly) | 🟥 |
| ☑ | Presence | 🟩 |

+ new layer

Polygon drawing. 🔒 Exit

# Adding Training Points



```
// filter the data based on date and area for 2016
var naip2016 = NAIP
  .filterBounds(AOI)
  .filterDate("2016-01-01", "2016-12-31");

  //define viewing parameters for multi band images
var visParams = {bands:['N', 'R', 'G']};
var visParams1 = {bands:['R', 'G', 'B']};

// add 2015 imagery to the map with false color and true color composites
Map.addLayer(naip2015,visParams,"2015_false",false );
Map.addLayer(naip2015,visParams1,"2015_true",false );

// add 2016 imagery to the map with false color and true color composites
Map.addLayer(naip2016,visParams,"2016_false",false );
Map.addLayer(naip2016,visParams1,"2016_true",false );
```

Configure geometry import

Name: geometry

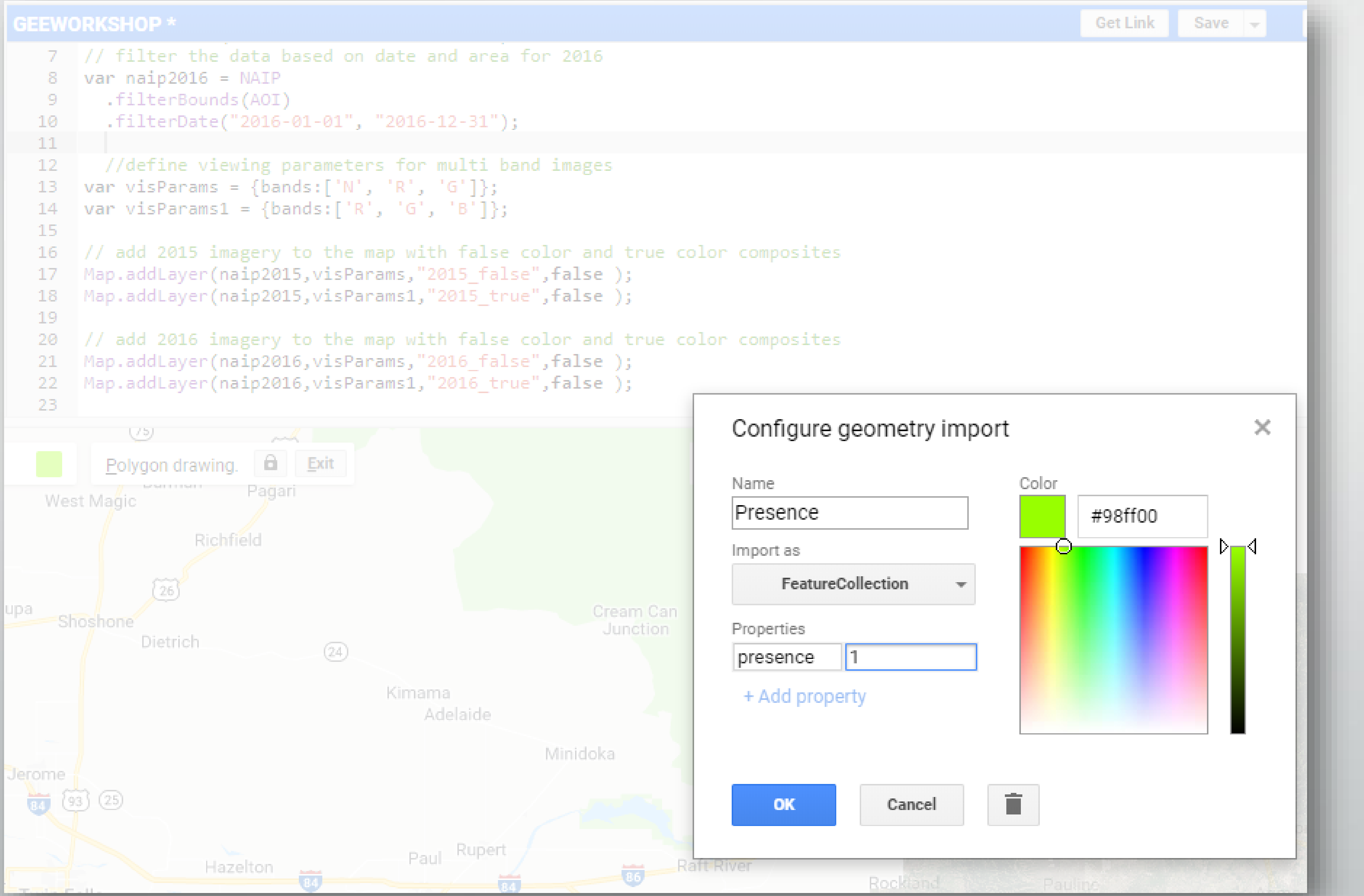
Import as: Geometry

- Geometry
- Feature
- FeatureCollection

Color: #98ff00

OK · Cancel

# Adding Training Points



GEEWORKSHOP *

```
// filter the data based on date and area for 2016
var naip2016 = NAIP
  .filterBounds(AOI)
  .filterDate("2016-01-01", "2016-12-31");

  //define viewing parameters for multi band images
var visParams = {bands:['N', 'R', 'G']};
var visParams1 = {bands:['R', 'G', 'B']};

// add 2015 imagery to the map with false color and true color composites
Map.addLayer(naip2015,visParams,"2015_false",false );
Map.addLayer(naip2015,visParams1,"2015_true",false );

// add 2016 imagery to the map with false color and true color composites
Map.addLayer(naip2016,visParams,"2016_false",false );
Map.addLayer(naip2016,visParams1,"2016_true",false );

```

Configure geometry import

Name: Presence

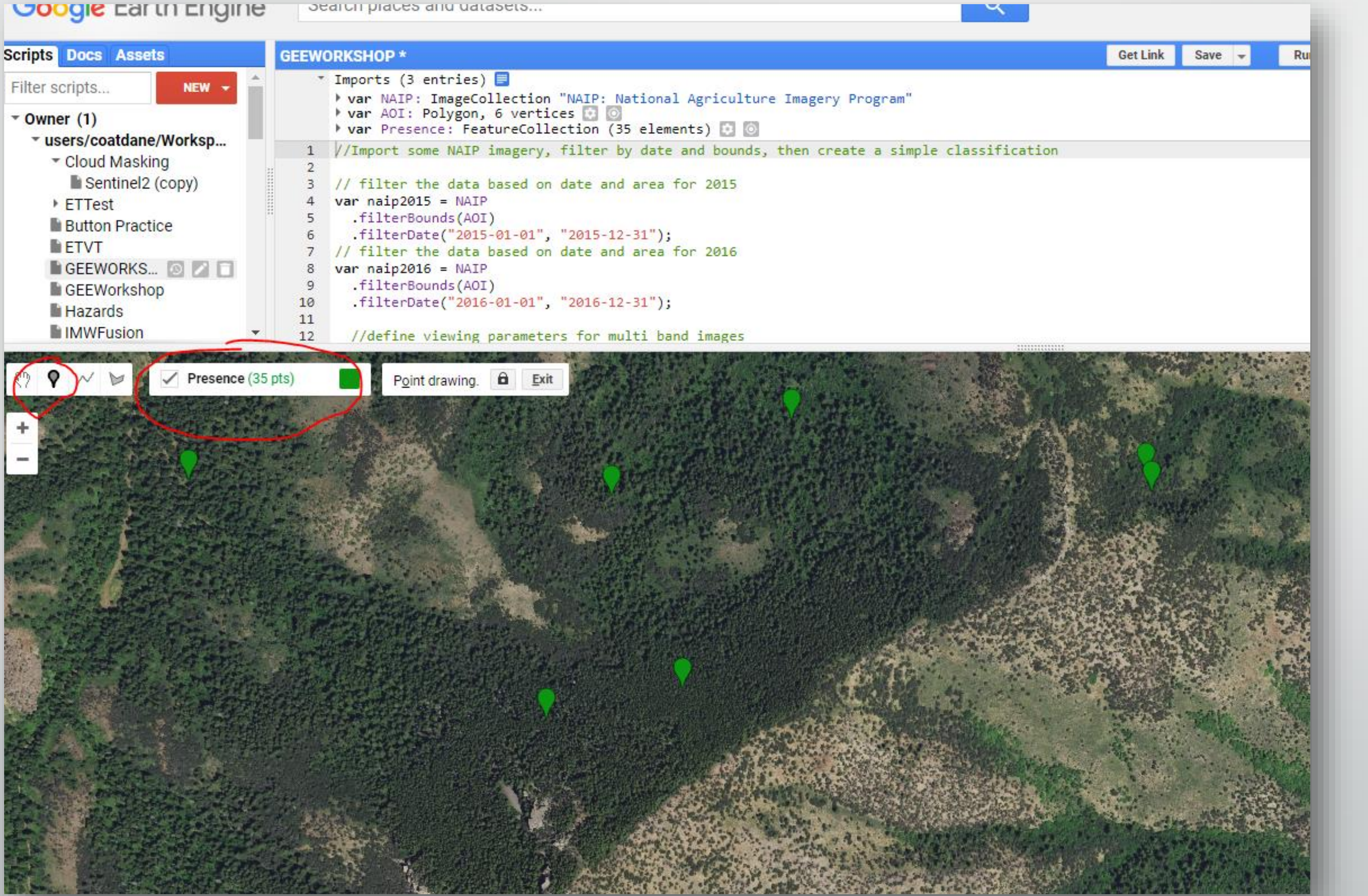
Import as: FeatureCollection

Properties: presence 1

+ Add property

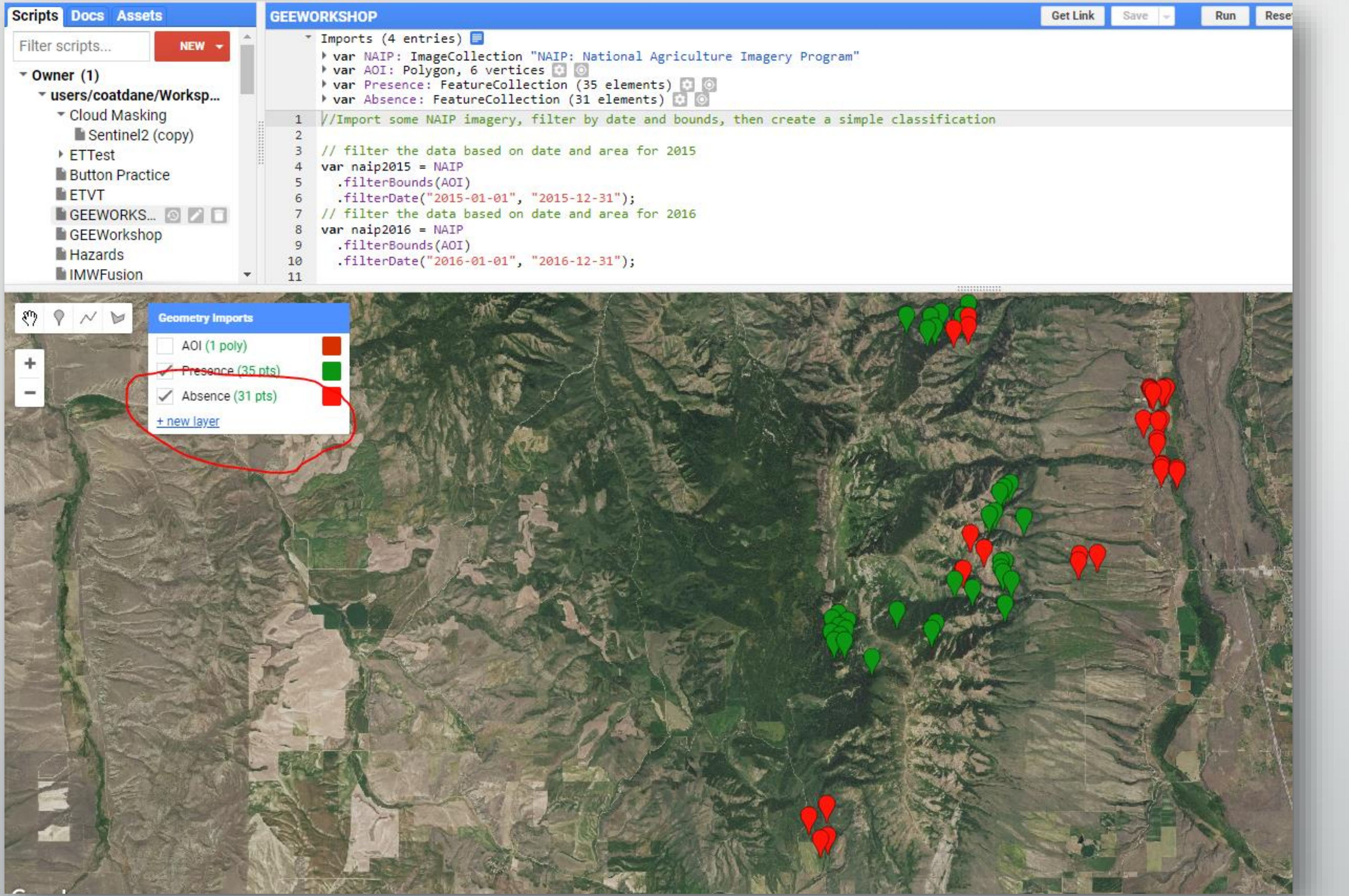
Color: #98ff00

# Adding Training Points



```
Imports (3 entries)
 var NAIP: ImageCollection "NAIP: National Agriculture Imagery Program"
 var AOI: Polygon, 6 vertices
 var Presence: FeatureCollection (35 elements)
//Import some NAIP imagery, filter by date and bounds, then create a simple classification

// filter the data based on date and area for 2015
var naip2015 = NAIP
  .filterBounds(AOI)
  .filterDate("2015-01-01", "2015-12-31");
// filter the data based on date and area for 2016
var naip2016 = NAIP
  .filterBounds(AOI)
  .filterDate("2016-01-01", "2016-12-31");

  //define viewing parameters for multi band images
```

# Adding Training Points



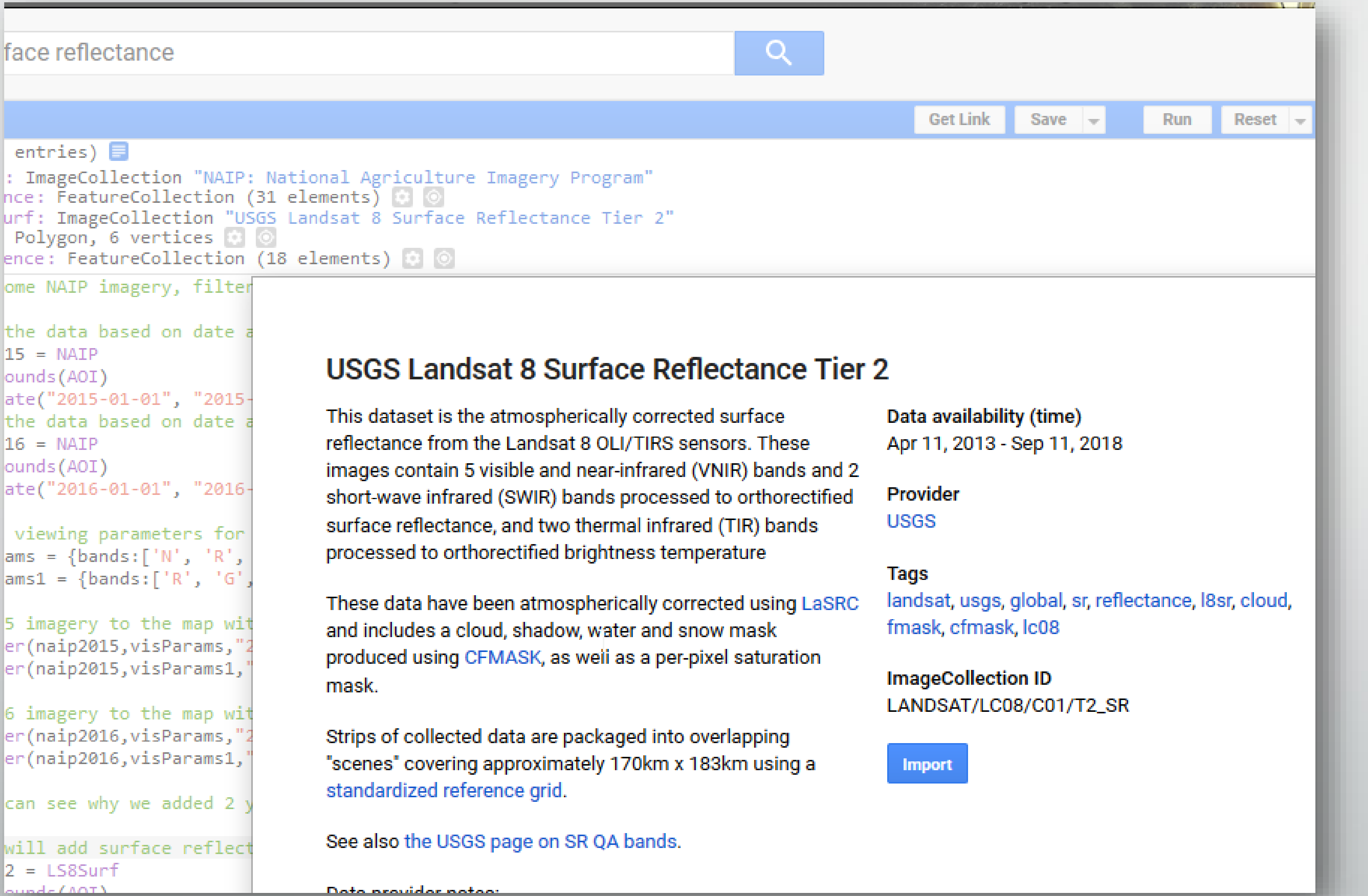
```
Imports (4 entries)
 var NAIP: ImageCollection "NAIP: National Agriculture Imagery Program"
 var AOI: Polygon, 6 vertices
 var Presence: FeatureCollection (35 elements)
 var Absence: FeatureCollection (31 elements)
//Import some NAIP imagery, filter by date and bounds, then create a simple classification

// filter the data based on date and area for 2015
var naip2015 = NAIP
  .filterBounds(AOI)
  .filterDate("2015-01-01", "2015-12-31");
// filter the data based on date and area for 2016
var naip2016 = NAIP
  .filterBounds(AOI)
  .filterDate("2016-01-01", "2016-12-31");
```

Geometry Imports

- ☐ AOI (1 poly) 🟥
- ✓ Presence (35 pts) 🟩
- ✓ Absence (31 pts) 🟥

+ new layer

# Training our future overlords



```
 entries)
: ImageCollection "NAIP: National Agriculture Imagery Program"
nce: FeatureCollection (31 elements)
urf: ImageCollection "USGS Landsat 8 Surface Reflectance Tier 2"
 Polygon, 6 vertices
ence: FeatureCollection (18 elements)
ome NAIP imagery, filter

the data based on date a
15 = NAIP
ounds(AOI)
ate("2015-01-01", "2015-
the data based on date a
16 = NAIP
ounds(AOI)
ate("2016-01-01", "2016-

 viewing parameters for
ams = {bands:['N', 'R',
ams1 = {bands:['R', 'G',

5 imagery to the map wit
er(naip2015,visParams,"2
er(naip2015,visParams1,"

6 imagery to the map wit
er(naip2016,visParams,"2
er(naip2016,visParams1,"

can see why we added 2 y

will add surface reflect
2 = LS8Surf
ounds(AOI)
```

## USGS Landsat 8 Surface Reflectance Tier 2

This dataset is the atmospherically corrected surface reflectance from the Landsat 8 OLI/TIRS sensors. These images contain 5 visible and near-infrared (VNIR) bands and 2 short-wave infrared (SWIR) bands processed to orthorectified surface reflectance, and two thermal infrared (TIR) bands processed to orthorectified brightness temperature

These data have been atmospherically corrected using LaSRC and includes a cloud, shadow, water and snow mask produced using CFMASK, as well as a per-pixel saturation mask.

Strips of collected data are packaged into overlapping "scenes" covering approximately 170km x 183km using a standardized reference grid.

See also the USGS page on SR QA bands.

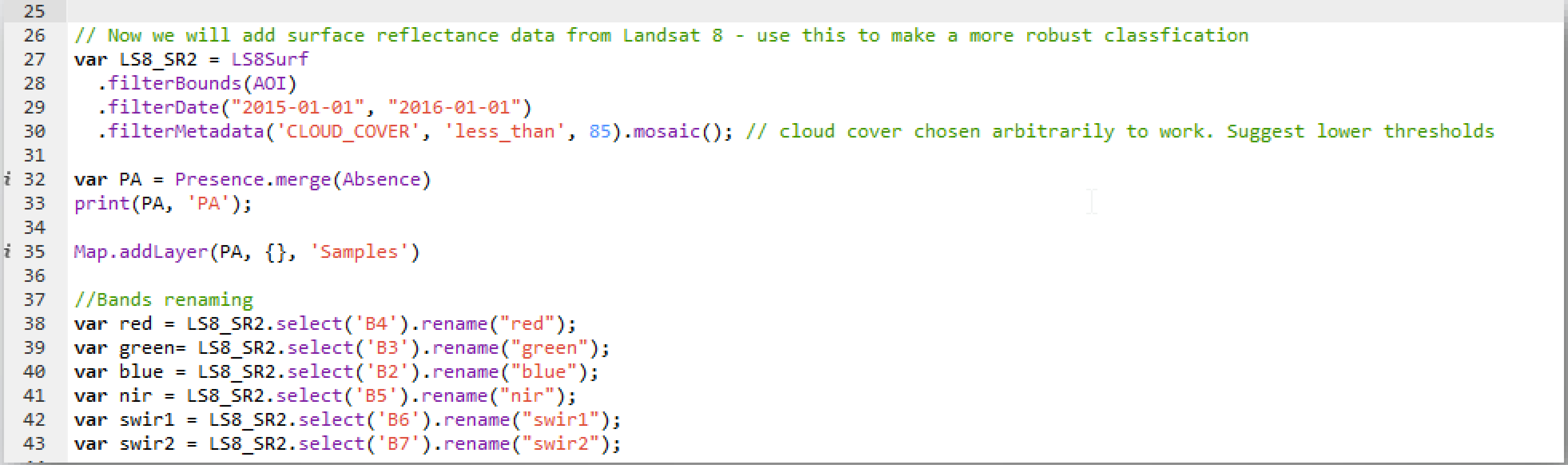
Data provider notes:

**Data availability (time)**
Apr 11, 2013 - Sep 11, 2018

**Provider**
USGS

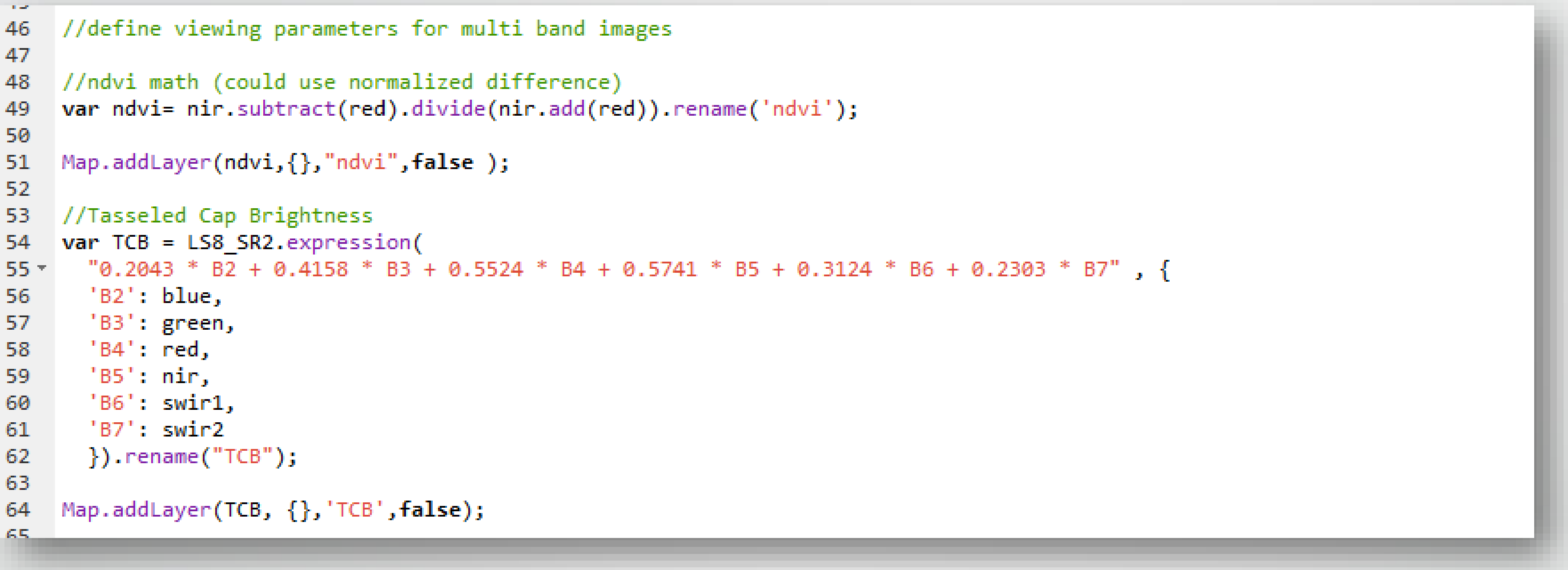
**Tags**
landsat, usgs, global, sr, reflectance, l8sr, cloud, fmask, cfmask, lc08

**ImageCollection ID**
LANDSAT/LC08/C01/T2_SR



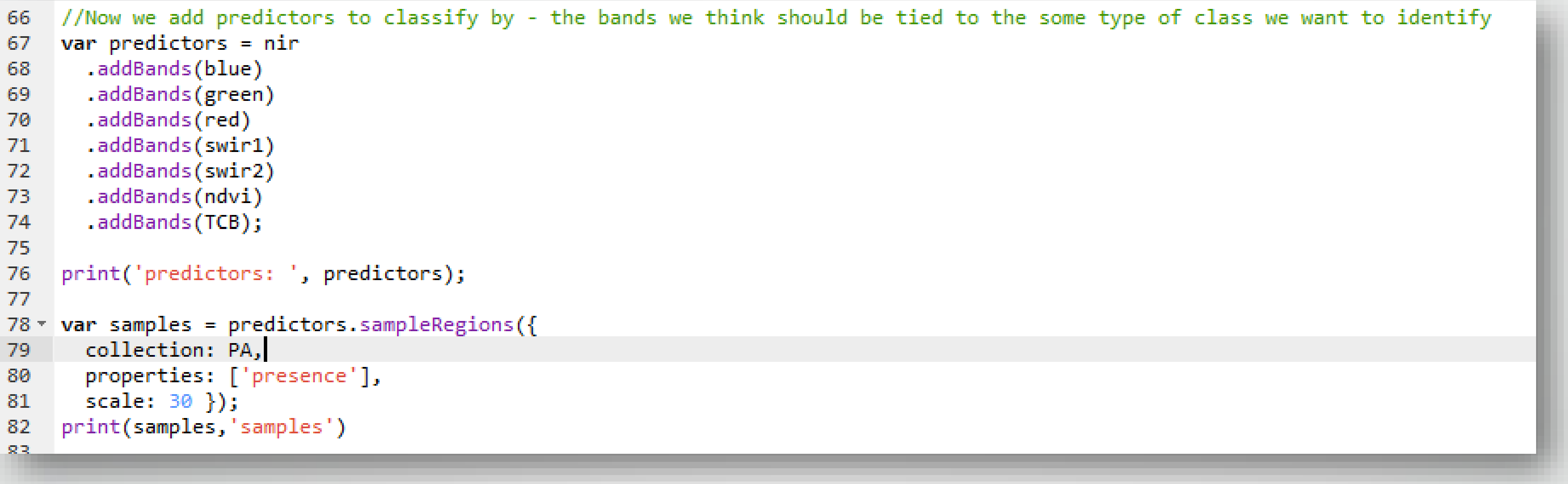
```
// Now we will add surface reflectance data from Landsat 8 - use this to make a more robust classfication
var LS8_SR2 = LS8Surf
  .filterBounds(AOI)
  .filterDate("2015-01-01", "2016-01-01")
  .filterMetadata('CLOUD_COVER', 'less_than', 85).mosaic(); // cloud cover chosen arbitrarily to work. Suggest lower thresholds

var PA = Presence.merge(Absence)
print(PA, 'PA');

Map.addLayer(PA, {}, 'Samples')

//Bands renaming
var red = LS8_SR2.select('B4').rename("red");
var green= LS8_SR2.select('B3').rename("green");
var blue = LS8_SR2.select('B2').rename("blue");
var nir = LS8_SR2.select('B5').rename("nir");
var swir1 = LS8_SR2.select('B6').rename("swir1");
var swir2 = LS8_SR2.select('B7').rename("swir2");
```

- So we have imported surface reflectance data for Landsat 8
- Want to add some common filters – filter for time, and by bounds
- Also filter by % cloud cover!
- Also need to merge our presence and absence points for future use
- Renaming our bands to be easy to remember for band math later



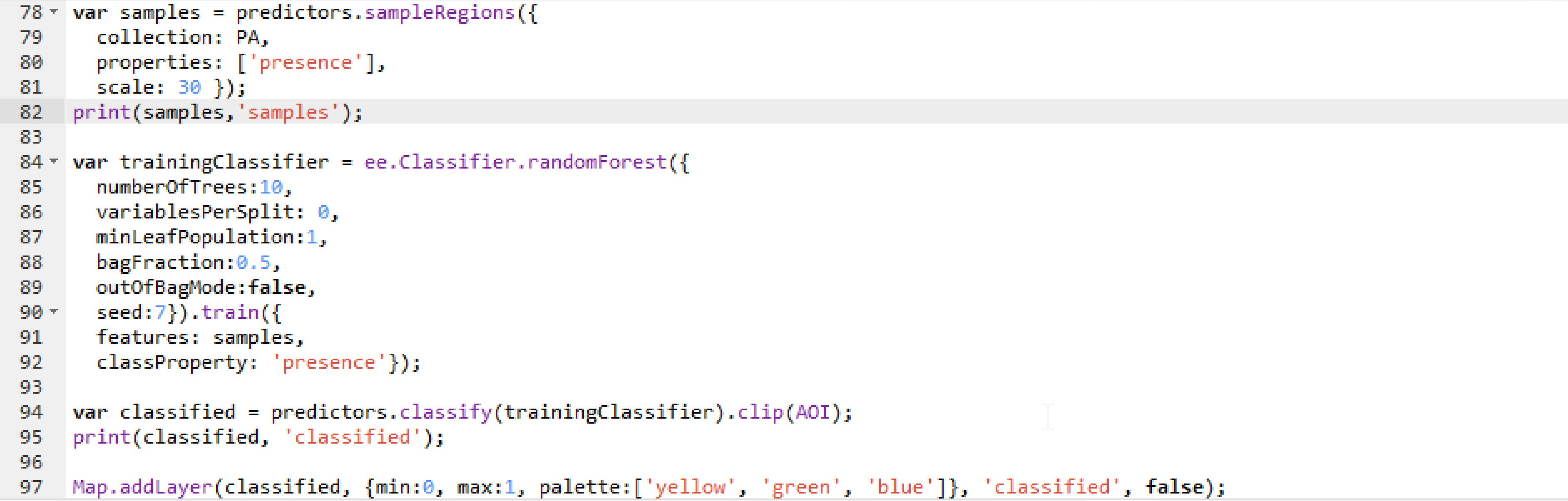
```
//define viewing parameters for multi band images

//ndvi math (could use normalized difference)
var ndvi= nir.subtract(red).divide(nir.add(red)).rename('ndvi');

Map.addLayer(ndvi,{},"ndvi",false );

//Tasseled Cap Brightness
var TCB = LS8_SR2.expression(
  "0.2043 * B2 + 0.4158 * B3 + 0.5524 * B4 + 0.5741 * B5 + 0.3124 * B6 + 0.2303 * B7" , {
  'B2': blue,
  'B3': green,
  'B4': red,
  'B5': nir,
  'B6': swir1,
  'B7': swir2
  }).rename("TCB");

Map.addLayer(TCB, {},'TCB',false);
```

- For veg classification we want to use as many predictive bands as we can think of
- For simplicity, we will use Tasseled Cap Brightness, and NDVI, but this could be anything from plain red color to MSAVI-2.
- Google Earth Engine uses the Normalized Difference function so much they've added that as a built in method.



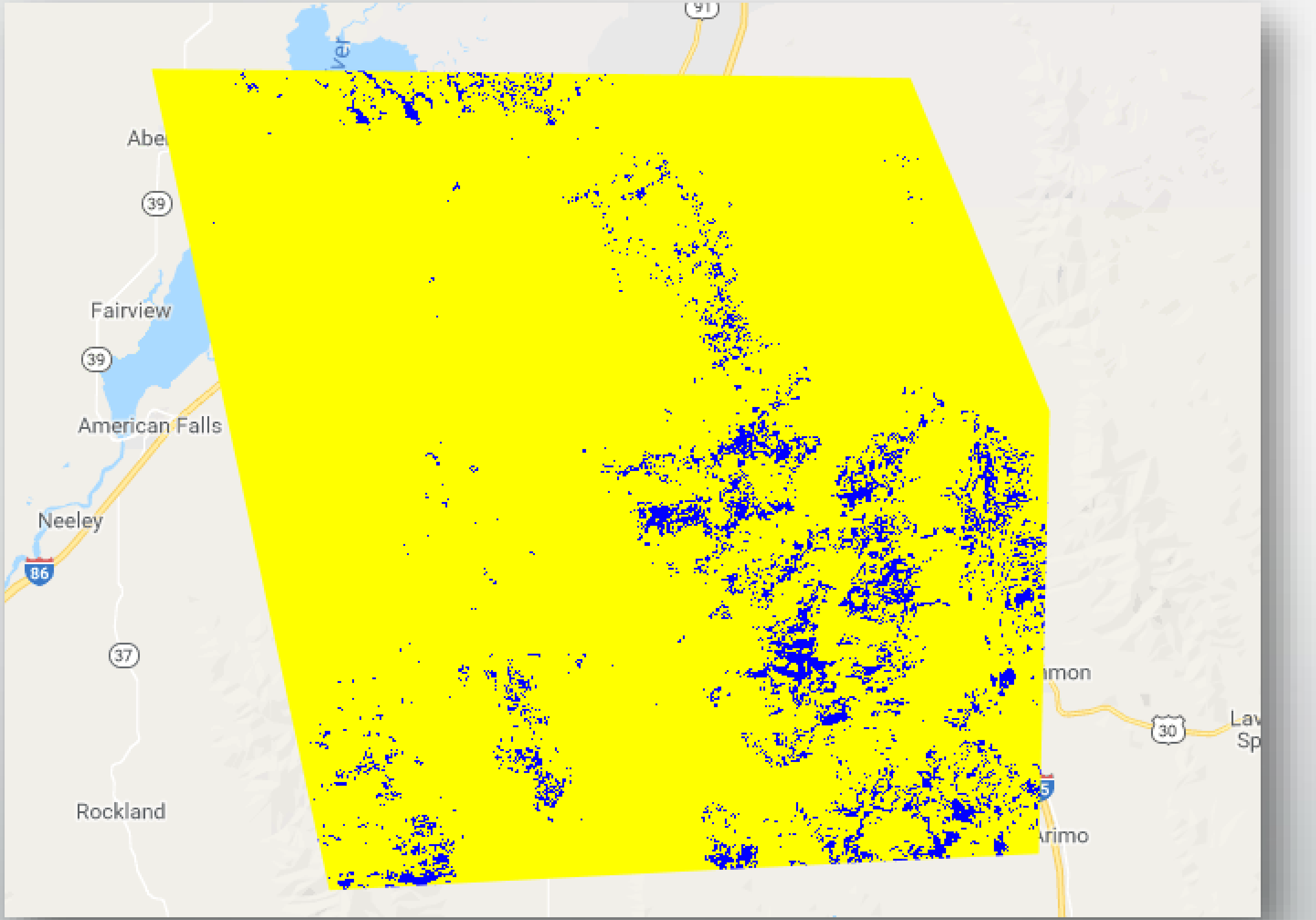
```
//Now we add predictors to classify by - the bands we think should be tied to the some type of class we want to identify
var predictors = nir
  .addBands(blue)
  .addBands(green)
  .addBands(red)
  .addBands(swir1)
  .addBands(swir2)
  .addBands(ndvi)
  .addBands(TCB);

print('predictors: ', predictors);

var samples = predictors.sampleRegions({
  collection: PA,
  properties: ['presence'],
  scale: 30 });
print(samples,'samples')
```

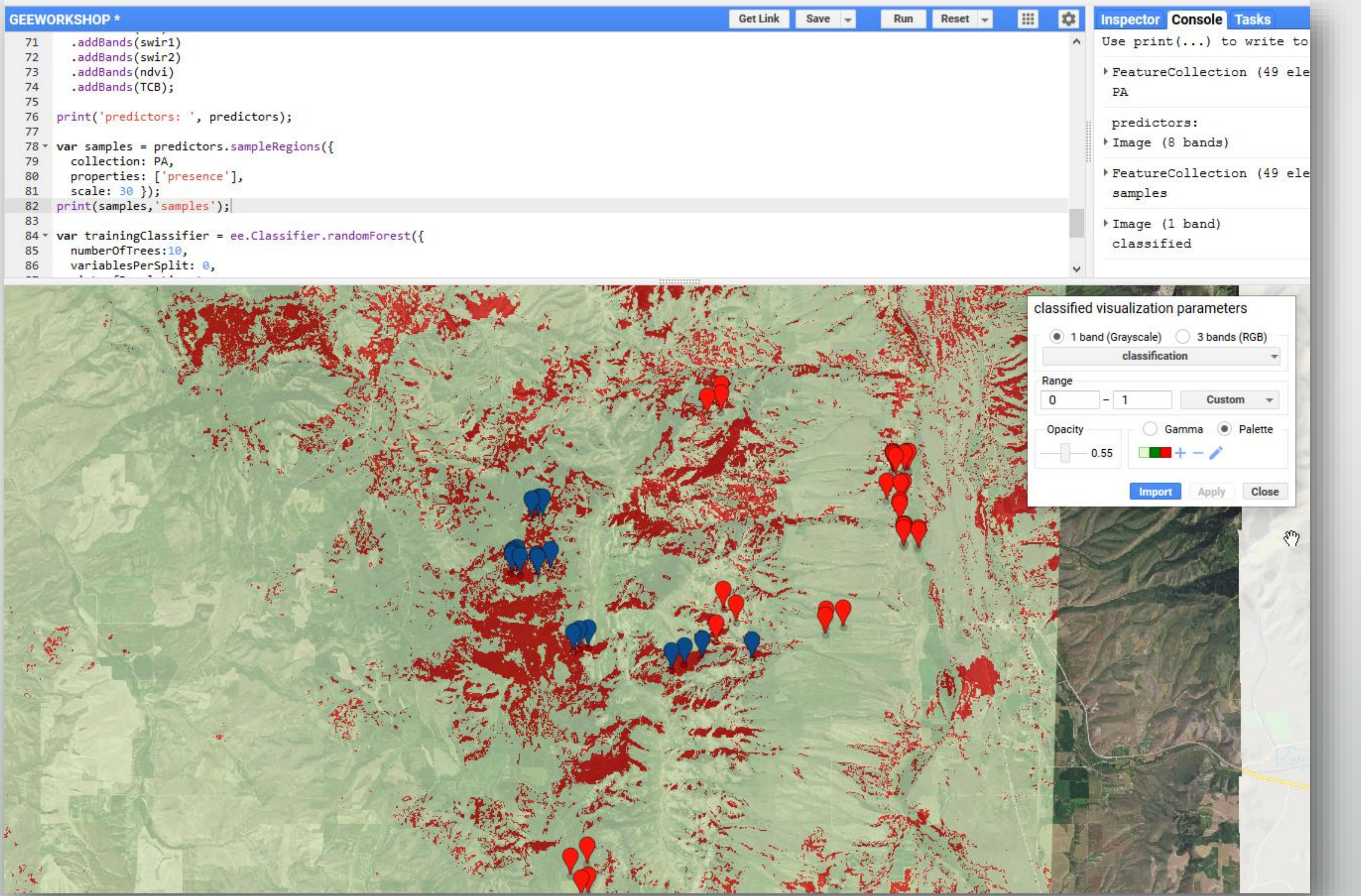
- So to compile the bands we want to use to predict or classify from, we create a variable of a single band, then add all the other bands to the list
- Notice this can be colors (RGB) or false color imagery like NDVI!



```
var samples = predictors.sampleRegions({
  collection: PA,
  properties: ['presence'],
  scale: 30 });
print(samples,'samples');

var trainingClassifier = ee.Classifier.randomForest({
  numberOfTrees:10,
  variablesPerSplit: 0,
  minLeafPopulation:1,
  bagFraction:0.5,
  outOfBagMode:false,
  seed:7}).train({
  features: samples,
  classProperty: 'presence'});

var classified = predictors.classify(trainingClassifier).clip(AOI);
print(classified, 'classified');

Map.addLayer(classified, {min:0, max:1, palette:['yellow', 'green', 'blue']}, 'classified', false);
```

- So now we tell the code to sample from our presence absence  merged point collection, for 30 meter pixels from Landsat imagery, and look at if there is presence = 0 or 1.
- Finally there is the classifier. Google Earth engine has several famous classification schemes; they are all very famous and each could have an entire conference to describe them.
- I chose randomForest because I like it and it gets good results. Google Earth engine has good documentation for the variables you feed a classification scheme, and most of these were chosen arbitrarily
- Lets look at the results!


91
Abe
39
Fairview
39
American Falls
Neeley
86
37
Rockland
mon
30
La
Sp
5
rimo



```
  .addBands(swir1)
  .addBands(swir2)
  .addBands(ndvi)
  .addBands(TCB);

print('predictors: ', predictors);

var samples = predictors.sampleRegions({
  collection: PA,
  properties: ['presence'],
  scale: 30 });
print(samples,'samples');

var trainingClassifier = ee.Classifier.randomForest({
  numberOfTrees:10,
  variablesPerSplit: 0,
```

```
Use print(...) to write to
FeatureCollection (49 ele
PA

predictors:
Image (8 bands)

FeatureCollection (49 ele
samples

Image (1 band)
classified
```

classified visualization parameters

● 1 band (Grayscale) ○ 3 bands (RGB)

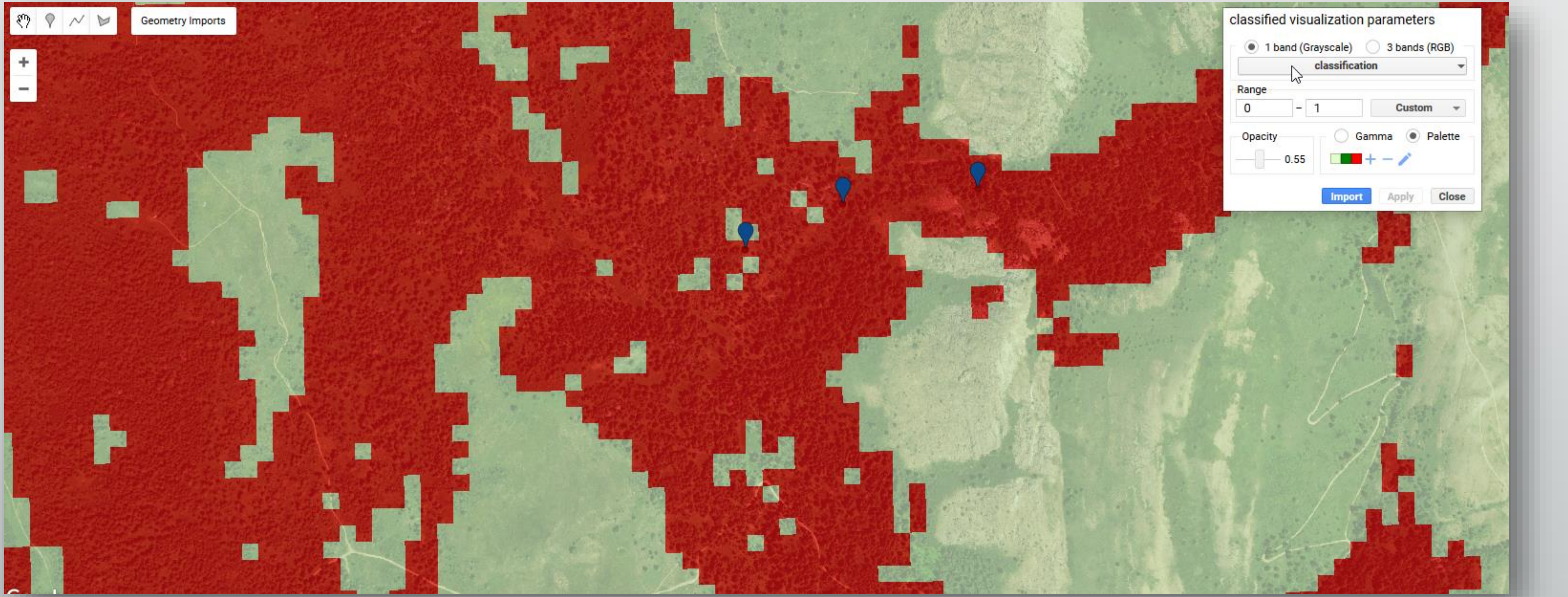
classification

Range 0 – 1 Custom

Opacity 0.55

○ Gamma ● Palette

# For 20-30 points this looks pretty good!



classified visualization parameters

● 1 band (Grayscale) ○ 3 bands (RGB)

classification

Range 0 – 1 Custom

Opacity 0.55

○ Gamma ● Palette

Hopefully you learned something today about Google Earth Engine and some of the interesting and powerful things we can do with it. For more info or to sign up for an account:

https://earthengine.google.com/

If you have any questions or comments, please reach out to me!

Dane Coats:

coatdane@isu.edu

# Acknowledgements

All of this would not be possible without training from the experts:

- Google Earth Engine Tutorials
- IMW Google Earth Engine Crash Course **by Dan Carver**
- Space in the GIS TReC from Keith Weber
- Courtner Ohr for the wonderful website!