Satellite Image Analysis with Earth Engine and Python Open Source Packages

This exercise introduces Google Earth Engine and raster satellite image analysis and visualization. Working with data from the Sentinel-2 satellite system:

- Assemble images around a specific location and within a specified time period
    - Query metadata information about the images
- View an image statically and with an interactive map
- Calculate a 'spectral indice'
- Calculate change between two images
- Classify an image into statistical clusters
- Create a time series dataset from single location from the images
    - Visualize the time series with an interactive map

The exercise can be run without any knowledge of coding, but for some background, you might want to run through the following coloboratory notebooks:

1. Earth Engine Python API Colab Setup https://colab.research.google.com/github/google/earthengine-api/blob/master/python/examples/ipynb/ee-api-colab-setup.ipynb
2. Google Earth Engine with Python - Developer's Guide https://colab.research.google.com/github/csaybar/EEwPython/blob/dev/1_Introduction.ipynb#scrollTo=5AkUqloUQpXv

Set up the analysis environment

The basic steps to get things set up, using 'python' scripting jargon, are:

1. Install packages
2. Import modules
3. 'Authenticate' earth engine
4. Define functions

Colaboratory comes with a variety of standard scientific computing 'libraries', and many others can be installed fairly easily. This exercise will use the 'earth engine api', the 'geemap' library for visualizing earth engine data, and a variety of libraries to plot results.

```
# install dependencies into virtual machine
!pip install geemap
#!pip install ee
#!pip install earthengine-api
!pip install geopandas
#!pip install altair
```

```
Collecting geemap
  Downloading https://files.pythonhosted.org/packages/c1/f1/50dc342f3c2844353652d5df007ca
     |████████████████████████████████| 389kB 6.0MB/s
Collecting whitebox
  Downloading https://files.pythonhosted.org/packages/9e/80/72ddfe55b9fad2909c3befb6360c3
     |████████████████████████████████| 71kB 6.6MB/s
Collecting voila
  Downloading https://files.pythonhosted.org/packages/50/7d/ef470f3d3c574162bb8705b264f4c
     |████████████████████████████████| 1.9MB 35.1MB/s
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from geem
Collecting folium>=0.11.0
  Downloading https://files.pythonhosted.org/packages/a4/f0/44e69d50519880287cc41e7c8a6ac
     |████████████████████████████████| 102kB 10.2MB/s
Collecting ipyleaflet>=0.13.3
  Downloading https://files.pythonhosted.org/packages/a0/9d/f411b8a9522f1710f278c60b2cd8d
     |████████████████████████████████| 5.1MB 37.0MB/s
Collecting mss
  Downloading https://files.pythonhosted.org/packages/d7/5f/77dece686b8d08a17430e169e9367
     |████████████████████████████████| 81kB 8.9MB/s
Collecting colour
  Downloading https://files.pythonhosted.org/packages/74/46/e81907704ab203206769dee1385dc
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from gee
Collecting pyshp
  Downloading https://files.pythonhosted.org/packages/ca/1f/e9cc2c3fce32e2926581f8b690583
     |████████████████████████████████| 225kB 42.1MB/s
Collecting geeadd>=0.5.1
  Downloading https://files.pythonhosted.org/packages/53/3e/98e4d21b5d06a670a6692272a89cb
Collecting bqplot
  Downloading https://files.pythonhosted.org/packages/f1/cc/bbb3989709ff5f43a75d8cc9989b0
     |████████████████████████████████| 983kB 32.3MB/s
Collecting geocoder
  Downloading https://files.pythonhosted.org/packages/4f/6b/13166c909ad2f2d76b929a4227c95
     |████████████████████████████████| 102kB 10.1MB/s
Requirement already satisfied: pillow in /usr/local/lib/python3.6/dist-packages (from gee
Collecting ipytree
```

```
  Downloading https://files.pythonhosted.org/packages/36/2f/f93092e485afc384b4cc39f17ff1c
     |████████████████████████████████| 1.1MB 32.6MB/s
Requirement already satisfied: earthengine-api>=0.1.230 in /usr/local/lib/python3.6/dist-
Collecting ipynb-py-convert
  Downloading https://files.pythonhosted.org/packages/43/90/37ff1b9f553583e60c327e1ea8cbf
Requirement already satisfied: click in /usr/local/lib/python3.6/dist-packages (from geem
Collecting ipyfilechooser
  Downloading https://files.pythonhosted.org/packages/f7/4f/e8d949a6395b6079fe81bf728e5cb
Collecting ffmpeg-python
  Downloading https://files.pythonhosted.org/packages/d7/0c/56be52741f75bad4dc6555991fabd
Requirement already satisfied: nbclient<0.6,>=0.4.0 in /usr/local/lib/python3.6/dist-pack
Collecting nbconvert<7,>=6.0.0
  Downloading https://files.pythonhosted.org/packages/13/2f/acbe7006548f3914456ee47f97a20
     |████████████████████████████████| 552kB 33.2MB/s
Collecting jupyter-client<7,>=6.1.3
  Downloading https://files.pythonhosted.org/packages/dc/41/9fa443d5ae8907dd8f7d12146cb00
     |████████████████████████████████| 112kB 35.0MB/s
Collecting jupyter-server<2.0.0,>=0.3.0
  Downloading https://files.pythonhosted.org/packages/7f/0d/1eafbdca04feecdf596382d077660
     |████████████████████████████████| 204kB 37.4MB/s
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from f
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: branca>=0.3.0 in /usr/local/lib/python3.6/dist-packages (f
Collecting traittypes<3,>=0.2.1
  Downloading https://files.pythonhosted.org/packages/9c/d1/8d5bd662703cc1764d986f6908a60
Requirement already satisfied: ipywidgets<8,>=7.5.0 in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (fr
Collecting beautifulsoup4>=4.9.0
  Downloading https://files.pythonhosted.org/packages/d1/41/e6495bd7d3781cee623ce23ea6ac7
     |████████████████████████████████| 122kB 40.7MB/s
Collecting logzero>=1.5.0
  Downloading https://files.pythonhosted.org/packages/7e/f7/369920be5b4e9c1a05f5e10bae62b
```

```
Requirement already satisfied: traitlets>=4.3.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: future in /usr/local/lib/python3.6/dist-packages (from geo
Collecting ratelim
  Downloading https://files.pythonhosted.org/packages/f2/98/7e6d147fd16a10a5f821db6e25f19
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from geocod
Requirement already satisfied: google-auth>=1.4.1 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: google-api-python-client in /usr/local/lib/python3.6/dist-
Requirement already satisfied: google-cloud-storage in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: httplib2shim in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: google-auth-httplib2>=0.0.3 in /usr/local/lib/python3.6/di
Requirement already satisfied: httplib2<1dev,>=0.9.2 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: nest-asyncio in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: nbformat>=5.0 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: async-generator in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: testpath in /usr/local/lib/python3.6/dist-packages (from n
Requirement already satisfied: bleach in /usr/local/lib/python3.6/dist-packages (from nbc
Requirement already satisfied: defusedxml in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: pygments>=2.4.1 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: tornado>=4.1 in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: Send2Trash in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: prometheus-client in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: ipython>=4.0.0; python_version >= "3.3" in /usr/local/lib/
Requirement already satisfied: widgetsnbextension~=3.5.0 in /usr/local/lib/python3.6/dist
Requirement already satisfied: ipykernel>=4.5.1 in /usr/local/lib/python3.6/dist-packages
```

```
Requirement already satisfied: ipykernel>=4.5.1 in /usr/local/lib/python3.6/dist-packages
Collecting soupsieve>1.2; python_version >= "3.0"
  Downloading https://files.pythonhosted.org/packages/6f/8f/457f4a5390eeae1cc3aeab89deb77
Requirement already satisfied: decorator in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3" in /usr/local/lib/pyt
Requirement already satisfied: setuptools>=40.3.0 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: uritemplate<4dev,>=3.0.0 in /usr/local/lib/python3.6/dist-
Requirement already satisfied: google-cloud-core<2.0dev,>=1.0.0 in /usr/local/lib/python3
Requirement already satisfied: google-resumable-media<0.5.0dev,>=0.3.1 in /usr/local/lib/
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: webencodings in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: packaging in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: ptyprocess; os_name != "nt" in /usr/local/lib/python3.6/di
Requirement already satisfied: pickleshare in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/python3.6/d
Requirement already satisfied: pexpect; sys_platform != "win32" in /usr/local/lib/python3
Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: notebook>=4.4.1 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: google-api-core<2.0.0dev,>=1.14.0 in /usr/local/lib/python
Requirement already satisfied: wcwidth in /usr/local/lib/python3.6/dist-packages (from pr
Requirement already satisfied: protobuf>=3.4.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.6.0 in /usr/local/lib/
Building wheels for collected packages: pyshp, ipynb-py-convert
  Building wheel for pyshp (setup.py) ... done
  Created wheel for pyshp: filename=pyshp-2.1.2-cp36-none-any.whl size=36216 sha256=ff3e7
  Stored in directory: /root/.cache/pip/wheels/96/6c/53/4112475adf3b831da97f083163d0f38ee
  Building wheel for ipynb-py-convert (setup.py) ... done
  Created wheel for ipynb-py-convert: filename=ipynb_py_convert-0.4.6-cp36-none-any.whl s
  Stored in directory: /root/.cache/pip/wheels/80/dc/7c/a7279f7726d66951fe48d5afd45247bcf
Successfully built pyshp ipynb-py-convert
ERROR: datascience 0.10.6 has requirement folium==0.2.1, but you'll have folium 0.11.0 wh
Installing collected packages: whitebox, nbconvert, jupyter-client, jupyter-server, voila
  Found existing installation: nbconvert 5.6.1
    Uninstalling nbconvert-5.6.1:
```

```
      Successfully uninstalled nbconvert-5.6.1
  Found existing installation: jupyter-client 5.3.5
    Uninstalling jupyter-client-5.3.5:
      Successfully uninstalled jupyter-client-5.3.5
  Found existing installation: folium 0.8.3
    Uninstalling folium-0.8.3:
      Successfully uninstalled folium-0.8.3
  Found existing installation: beautifulsoup4 4.6.3
    Uninstalling beautifulsoup4-4.6.3:
      Successfully uninstalled beautifulsoup4-4.6.3
Successfully installed beautifulsoup4-4.9.3 bqplot-0.12.19 colour-0.1.5 ffmpeg-python-0.2
WARNING: The following packages were previously imported in this runtime:
  [jupyter_client]
You must restart the runtime in order to use newly installed versions.
```

RESTART RUNTIME

```
Collecting geopandas
  Downloading https://files.pythonhosted.org/packages/f7/a4/e66aafbefcbb717813bf3a355c8c4
     |████████████████████████████████| 972kB 4.3MB/s
Requirement already satisfied: shapely in /usr/local/lib/python3.6/dist-packages (from ge
Requirement already satisfied: pandas>=0.23.0 in /usr/local/lib/python3.6/dist-packages (
Collecting fiona
  Downloading https://files.pythonhosted.org/packages/37/94/4910fd55246c1d963727b03885ead
     |████████████████████████████████| 14.8MB 308kB/s
Collecting pyproj>=2.2.0
  Downloading https://files.pythonhosted.org/packages/e4/ab/280e80a67cfc109d15428c0ec5639
     |████████████████████████████████| 6.5MB 39.8MB/s
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.6/dist-packages (f
Collecting cligj>=0.5
  Downloading https://files.pythonhosted.org/packages/ba/06/e3440b1f2dc802d35f329f299ba96
Collecting click-plugins>=1.0
  Downloading https://files.pythonhosted.org/packages/e9/da/824b92d9942f4e472702488857914
Collecting munch
  Downloading https://files.pythonhosted.org/packages/cc/ab/85d8da5c9a45e072301beb37ad7f8
```

- *You may have now have to `RESTART RUNTIME` (with a button in the output above) to properly import the modules in the next step*

```
Successfully installed click-plugins-1.1.1 cligj-0.7.0 fiona-1.8.18 geopandas-0.8.1 munch
```

Import modules

Plotting, visualization, and data analysis tools

```
# earth engine and statistical analysis
from IPython.display import Image
import ee, datetime
import pandas as pd
from pylab import *
import seaborn as sns

# file management
import os
import glob

# geospatial and plotting modules
#import geopandas as gpd
import geemap.eefolium as geemap

import folium
import matplotlib.pyplot as plt
%matplotlib inline

import altair as alt
```

Authenticate earth engine

Since earth engine requires a login, to use it with colaboratory requires 'authenticating' an analysis session. That is, giving permission via a google account. Rinning the cell below will provide a link to a google authentication website, which will ask your permission and then provide a verification code to paste into a box - then hit enter to finish the process.

```
# Trigger the authentication flow.
ee.Authenticate()
```

```
To authorize access needed by Earth Engine, open the following URL in a web browser and f

    https://accounts.google.com/o/oauth2/auth?client_id=517222506229-vsmmajv00ul0bs7p89v5

The authorization workflow will generate a code, which you should paste in the box below.
Enter verification code: 4/1AY0e-g4yChCeu_CwfOmKWwwvArbXAuoEYs2058J_M4qYtst3PSA3nL54JKU

Successfully saved authorization token.
```

Once it has `Successfully saved authorization token` the session can be 'initialized'

```
# start up the session
ee.Initialize()
```

Define Functions

Functions do something with information. Here are functions to *mask* out clouds and cloud shadows, and to calculate a variety of *spectral indices* from the image data. Sentinel-2 and Landsat 8 satellite image data are used.

- Landsat 8 data is called with `ee.ImageCollection("LANDSAT/LC08/C01/T1_SR")` in the script and described here: https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C01_T1_SR
- The functions for masking are adapted from here: https://code.earthengine.google.com/?scriptPath=Examples:Datasets/LANDSAT_LC08_C01_T1_SR
- Sentinel-2 data is called with `ee.ImageCollection("COPERNICUS/S2_SR")` and described here: https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S2

Sentinel-2 image bands used in the functions are:

- Band 4 = red
- Band 8 = NIR
- at 10m pixel resolution

Landsat 8 bands used are:

- Band 4 = red
- Band 5 = NIR
- at 30m pixel resolution

The exercise uses Sentinel-2 data, and can be adapted to work with Landsat 8.

Cloudmask functions

These will mask clouds from the image collections to help improve analysis results of the time series data.

```
#function to use QA band to filter clouds
#Landsat 8 sr data
def maskL8sr(image):
  #Bits 3 and 5 are cloud shadow and cloud, respectively.
  cloudShadowBitMask = (1 << 3)
  cloudsBitMask = (1 << 5)
  # Get the pixel QA band.
  qa = image.select('pixel_qa')
  # Both flags should be set to zero, indicating clear conditions.
  mask = qa.bitwiseAnd((cloudShadowBitMask).eq(0)).bitwiseAnd((cloudsBitMask).eq(0))
  return image.updateMask(mask)


#function to use QA band to filter clouds
#Sentinel 2 data
def maskS2clouds(image):
  qa = image.select('QA60')

  #Bits 10 and 11 are clouds and cirrus, respectively.
  cloudBitMask = 1 << 10
  cirrusBitMask = 1 << 11

  #Both flags should be set to zero, indicating clear conditions.
  mask = qa.bitwiseAnd(cloudBitMask).eq(0)
  mask2 = qa.bitwiseAnd(cirrusBitMask).eq(0)
  mask3 = qa.bitwiseAnd(cloudBitMask).eq(0).bitwiseAnd(cirrusBitMask).eq(0)
  #cloud_img = image.updateMask(mask)
  #cir_img = cloud_img.updateMask(mask2)
```

```
  return image.updateMask(mask).updateMask(mask2)
  #return image.updateMask(mask)
```

Spectral indices and transformations

Spectral indices are designed to combine different wavelengths of energy in order to highlight some materials or features of interest.

Code for the Landsat 8 indices was adapted from *renelikestacos* on github. There are quite a few here, and many more have been created.

This exercise uses the *NDVI (Normalized Difference Vegetation Index)* as an example. NDVI is described here in the context of the MODIS sensor system - this exercise uses Sentinel-2 imagery which has 10 meter size pixels.

Other indices are described here:

- Landsat 8 Spectral Indices
- Sentinel-2 indices On first look-over I thought there was one for "Macaroni" but I misread the table.

```
# Landsat 8 indices
def NDVI(image):
    return image.normalizedDifference(['B5', 'B4'])

def SAM(image):
    band1 = image.select("B1")
    bandn = image.select("B2","B3","B4","B5","B6","B7","B8","B9");
    maxObjSize = 256;
    b = band1.divide(bandn);
    spectralAngleMap = b.atan();
```

```
    spectralAngleMap_sin = spectralAngleMap.sin();
    spectralAngleMap_cos = spectralAngleMap.cos();
    sum_cos = spectralAngleMap_cos.reduce(ee.call("Reducer.sum"));
    sum_sin = spectralAngleMap_sin.reduce(ee.call("Reducer.sum"));
    return ee.Image.cat(sum_sin, sum_cos, spectralAngleMap_sin, spectralAngleMap_cos);

#Enhanced Vegetation Index
def EVI(image):
    # L(Canopy background)
    # C1,C2(Coefficients of aerosol resistance term)
    # GainFactor(Gain or scaling factor)
    gain_factor = ee.Image(2.5);
    coefficient_1 = ee.Image(6);
    coefficient_2 = ee.Image(7.5);
    l = ee.Image(1);
    nir = image.select("B5");
    red = image.select("B4");
    blue = image.select("B2");
    evi = image.expression(
        "Gain_Factor*((NIR-RED)/(NIR+C1*RED-C2*BLUE+L))",
        {
            "Gain_Factor":gain_factor,
            "NIR":nir,
            "RED":red,
            "C1":coefficient_1,
            "C2":coefficient_2,
            "BLUE":blue,
            "L":l
        }
    )
    return evi

#Atmospherically Resistant Vegetation Index
```

```
#Atmospherically Resistant Vegetation Index
def ARVI(image):
    red = image.select("B4")
    blue = image.select("B2")
    nir = image.select("B5")
    red_square = red.multiply(red)
    arvi = image.expression(
        "NIR - (REDsq - BLUE)/(NIR+(REDsq-BLUE))",{
            "NIR": nir,
            "REDsq": red_square,
            "BLUE": blue
        }
    )
    return arvi

#Leaf Area Index
def LAI(image):
    nir = image.select("B5")
    red = image.select("B4")
    coeff1 = ee.Image(0.0305);
    coeff2 = ee.Image(1.2640);
    lai = image.expression(
        "(((NIR/RED)*COEFF1)+COEFF2)",
        {
            "NIR":nir,
            "RED":red,
            "COEFF1":coeff1,
            "COEFF2":coeff2
        }
    )
    return lai
```

This exercise demonstrates the use of Sentinel-2 image data, so these indices need to be adapted to the different band names in the dataset.

```
# adapted to Sentinel-2

# NDVI for Sentinel-2 bands
def senNDVI(image):
    return image.normalizedDifference(['B8', 'B4'])
```

Create target areas for analysis

A single point will be used to set the initial data search.

Note that this is 'x, y' style (longitude, latitude)

- Hagen Research Natural Area (up the South Fork of Gate Creek): [-122.418960, 44.163945]

```
# use the Gate Creek point
rnapoint = [-122.418960, 44.163945]
```

```
# Create earth engine point
point = {'type':'Point', 'coordinates':rnapoint}
```

Create image collection

An 'image collection' is a group (or filtered group) of related imagery or other data.

- ImageCollection Overview

Images in earth engine are described here.

- Image Overview

Collections can be vast, so commonly they are filtered by a geographic area or time period.

Set time period

This exercise uses data from early 2015 to the October 10, 2020. The Sentinel-2 image collection frequency increases over this time as a second sensor comes into use.

Landsat 8 dataset availability is from April 2013 onwards, so a longer time period could be analysed with this data.

```
# Set start and end date
startTime = datetime.datetime(2015, 3, 28)
# endTime = datetime.datetime.now()
endTime = datetime.datetime(2020, 11, 9)
```

Load Sentinel-2 image collection

```
# Create Sentinel-2 image collection
collection = ee.ImageCollection("COPERNICUS/S2").filterDate(startTime, endTime).filterBounds(p

# Create Landsat 8 image collection
#L8collection = ee.ImageCollection("LANDSAT/LC08/C01/T1_SR").filterDate(startTime, endTime).fi
```

```
#L8collection = ee.ImageCollection('LANDSAT/LC08/C01/T1_SR').filterDate(startTime, endTime).fi
```

Image Collection Information and Metadata

It is usually a good idea to examine the data in the collections created.

```
# Get the number of images.
count = collection.size()
print('Count: ', str(count.getInfo())+'\n')
```

```
    Count:  280
```

```
# Get the date range of images in the collection.
drange = collection.reduceColumns(ee.Reducer.minMax(), ["system:time_start"])
mindate = (ee.Date(drange.get('min')).getInfo())
maxdate = (ee.Date(drange.get('max')).getInfo())
print(f'Sentinel 2 Date range: {mindate} to {maxdate}')
```

```
    Sentinel 2 Date range: {'type': 'Date', 'value': 1445454617457} to {'type': 'Date', 'valu
```

Those dates look a little odd - they are a timestamp (in milliseconds) from Jan 1, 1970 - this is 'UNIX epoch time.' We can convert back to a more human-readable format this way:

```
# divide by 1000 to convert from milliseconds
datetime.datetime.fromtimestamp(0/1000)
```

```
    datetime.datetime(1970, 1, 1, 0, 0)
```

So the date / time of the oldest image in this particular collection is:

```
datetime.datetime.fromtimestamp(1445454617457/1000)
```

```
    datetime.datetime(2015, 10, 21, 19, 10, 17, 457000)
```

There are many other types of information about the images in the metadata.

```
#Get a list of all metadata properties.
properties = collection.propertyNames()
print('Metadata properties: '+str(properties.getInfo())) # ee.List of metadata properties
```

```
    Metadata properties: ['date_range', 'period', 'system:visualization_0_min', 'type_name',
```

Metadata for the first image in the collection

'Metadata' or standarized information about a dataset, is useful both for human-understanding of the usefulness (or not) of a particular dataset. More information about getting metatdata can be found at these links:

- https://developers.google.com/earth-engine/ic_info
- https://samapriya.github.io/gee-py/projects/collection_meta/

```
# sort the collection by time of image acquisition
sorted = collection.sort("system:time_start");
```

```
# Get the first (oldest) image.
scene = sorted.first()


#properties of the image
properties = scene.propertyNames()
print('Metadata properties: '+str(properties.getInfo())) # ee.List of metadata properties

    Metadata properties: ['DATATAKE_IDENTIFIER', 'SPACECRAFT_NAME', 'FORMAT_CORRECTNESS_FLAG'


#Get the 'tile' location identifier
target = 'MGRS_TILE'
meta = scene.get(target).getInfo()
print('{} = {}'.format(target, meta))

    MGRS_TILE = 10TEP


#Get the data id
target = 'GRANULE_ID'
meta = scene.get(target).getInfo()
print('{} = {}'.format(target, meta))

    GRANULE_ID = S2A_OPER_MSI_L1C_TL_MTI__20160419T053054_A001726_T10TEP_N02.01


#Get estimated cloud cover
target = 'CLOUD_COVERAGE_ASSESSMENT'
meta = scene.get(target).getInfo()
print('{} = {}'.format(target, meta))

    CLOUD_COVERAGE_ASSESSMENT = 29.30593076923077
```

```
#Get the timestamp and convert it to a date.
date = ee.Date(scene.get('system:time_start')).getInfo()
# print(date['value'])
x=date['value']
print(datetime.datetime.fromtimestamp(x/1000))
```

```
2015-10-21 19:10:17.457000
```

```
oldest = collection.sort('system:time_start').limit(1)
scene = oldest.first()
#Get a specific metadata property.
target = 'system:time_start'
meta = scene.get(target).getInfo()
print('{} = {}'.format(target, meta))

#Get the timestamp and convert it to a date.
date = ee.Date(scene.get('system:time_start')).getInfo()
# print(date['value'])
x=date['value']
print(datetime.datetime.fromtimestamp(x/1000))
```

```
system:time_start = 1445454617457
2015-10-21 19:10:17.457000
```

Assignment: Geog 485

What does the “MGRS_TILE” metadata mean in the context of the Sentinel-2 image collection - how do we interpret the value ‘10TEP’? Describe how the Sentinel-2 tile system works.

Here is the code cell again...

[ ] ↳ 1 cell hidden

Assignment: Geog 491

What data type are the `mindate` and `maxdate` variables? How could the outputof the code below be made to print the following:

```
Sentinel 2 Date range: 1445454617457 to 1604430749547
```

instead of:

```
# Sentinel 2 Date range: {'type': 'Date', 'value': 1445454617457} to {'type': 'Date', 'value': 1604862
```

```
# this is the code below, mentioned above
# Get the date range of images in the collection.
drange = collection.reduceColumns(ee.Reducer.minMax(), ["system:time_start"])
mindate = (ee.Date(drange.get('min')).getInfo()['value']) # datatype is dict so provide the re
maxdate = (ee.Date(drange.get('max')).getInfo()['value']) # datatype is dict so provide the re
print(f'Sentinel 2 Date range: {mindate} to {maxdate}')
```

```
    Sentinel 2 Date range: 1445454617457 to 1604862746318
```

Assignment: Geog 491

Write a function to convert a timestamp from a date, given a ee image.

Adapt the code below, so that given an image, the output will be the date/time in a more human readable format.

> The function could start something like this:

```
def ts_toDate(scene):
  # your function here
```

> and end up printing output such as:

```
2015-10-21 19:10:17.457000
```

```
def ts_toScene(scene):
    #Get the timestamp and convert it to a date.
    date = ee.Date(scene.get('system:time_start')).getInfo()
    # print(date['value'])
    x=date['value']
    print(datetime.datetime.fromtimestamp(x/1000))

ts_toScene(scene)
```

```
2015-10-21 19:10:17.457000
```

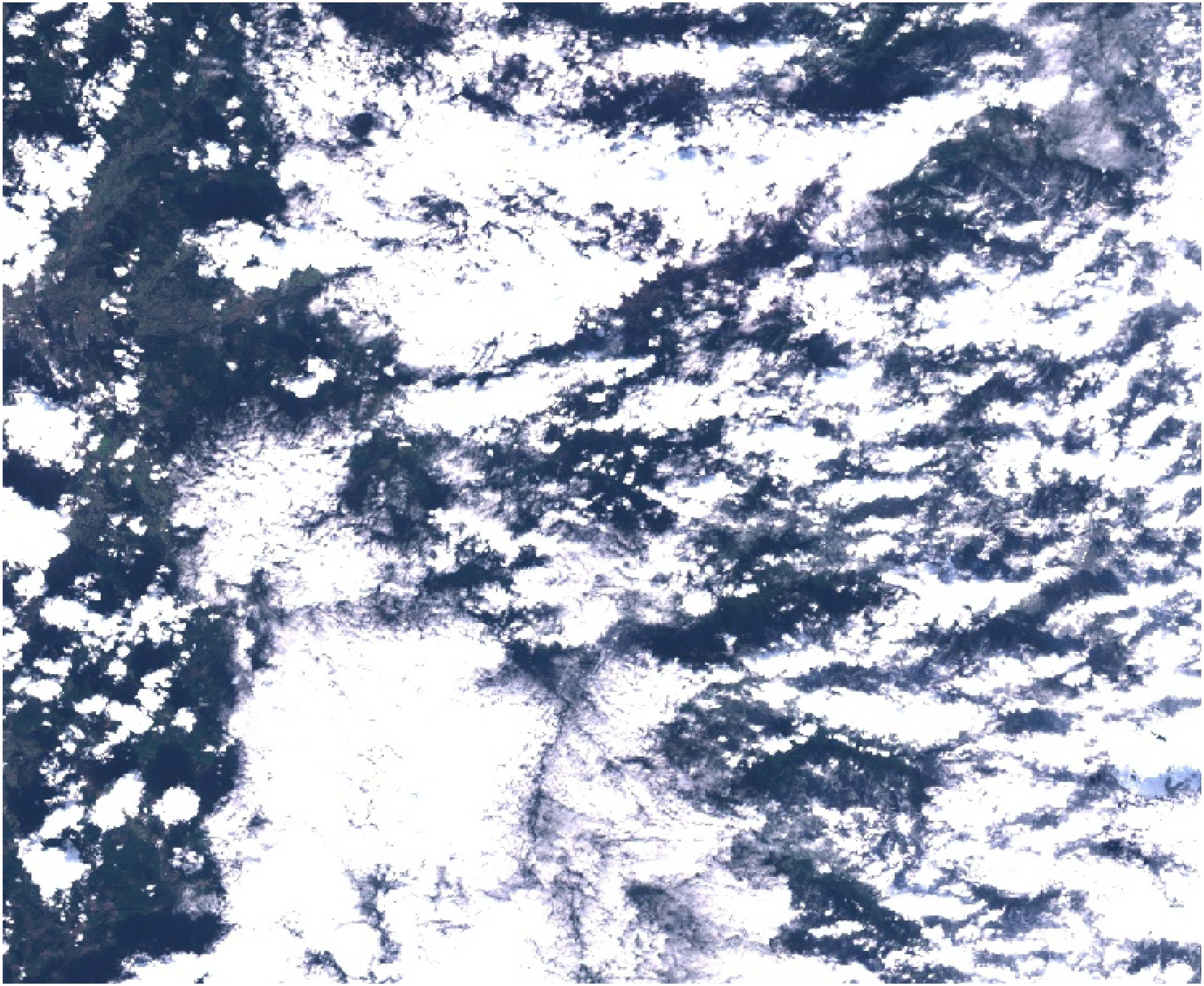
View the most recent image

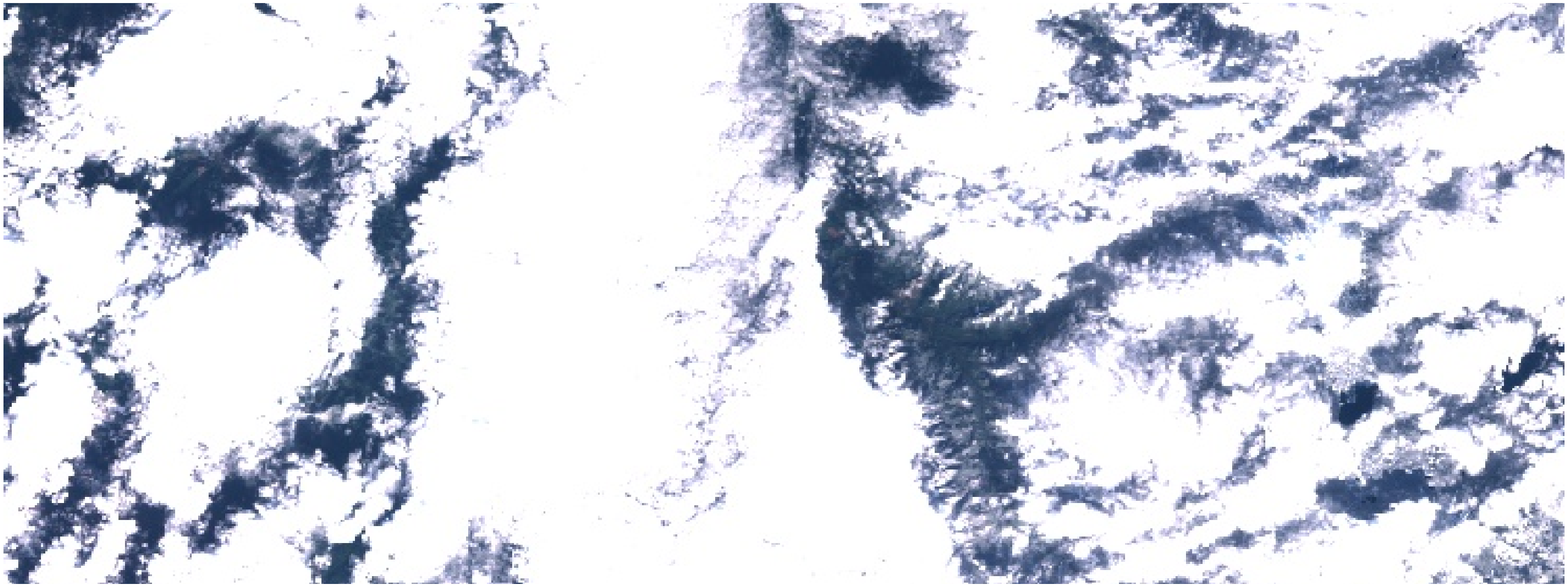
The data can be viewed as a static image or with an interactive map. First, the static image - these are handy because you can copy and paste them directly into another document

```
# Limit the collection to the 1 most recent images.
recent = collection.sort('system:time_start', False).limit(1)
# pull the image from the collection
image = recent.first()
print('Recent images: '+str(recent.getInfo())+'\n')
```

```
Recent images: {'type': 'ImageCollection', 'bands': [], 'id': 'COPERNICUS/S2', 'version':
```

```
# set the visualization parameters for the output
visParams = {'bands': ['B4', 'B3', 'B2'], 'max': 3000, 'dimensions':1000 }

# create image thumbnail in earth engine and view
thumbnail = image.getThumbUrl(visParams)
Image(url=thumbnail)
```





It is a bit cloudy as of this run. The 'least cloudy' images - there are probably many, can be pulled from the collection using the Quality Assurance band

Get the least cloudy image

The metadata for the collection allows the sorting by estimated cloud cover - for now, limit it to the ten least cloudy images

```
sorted = collection.sort('CLOUD_COVERAGE_ASSESSMENT').limit(10)

# Get the first (least cloudy) image.
image = sorted.first();


# Get the number of images.
count = sorted.size()
```

```
print('Count: ', str(count.getInfo())+'\n')

# Get the date range of images in the collection.
drange = sorted.reduceColumns(ee.Reducer.minMax(), ["system:time_start"])
print('Sentinel 2 Date range: ', str(ee.Date(drange.get('min')).getInfo()), str(ee.Date(drange
```

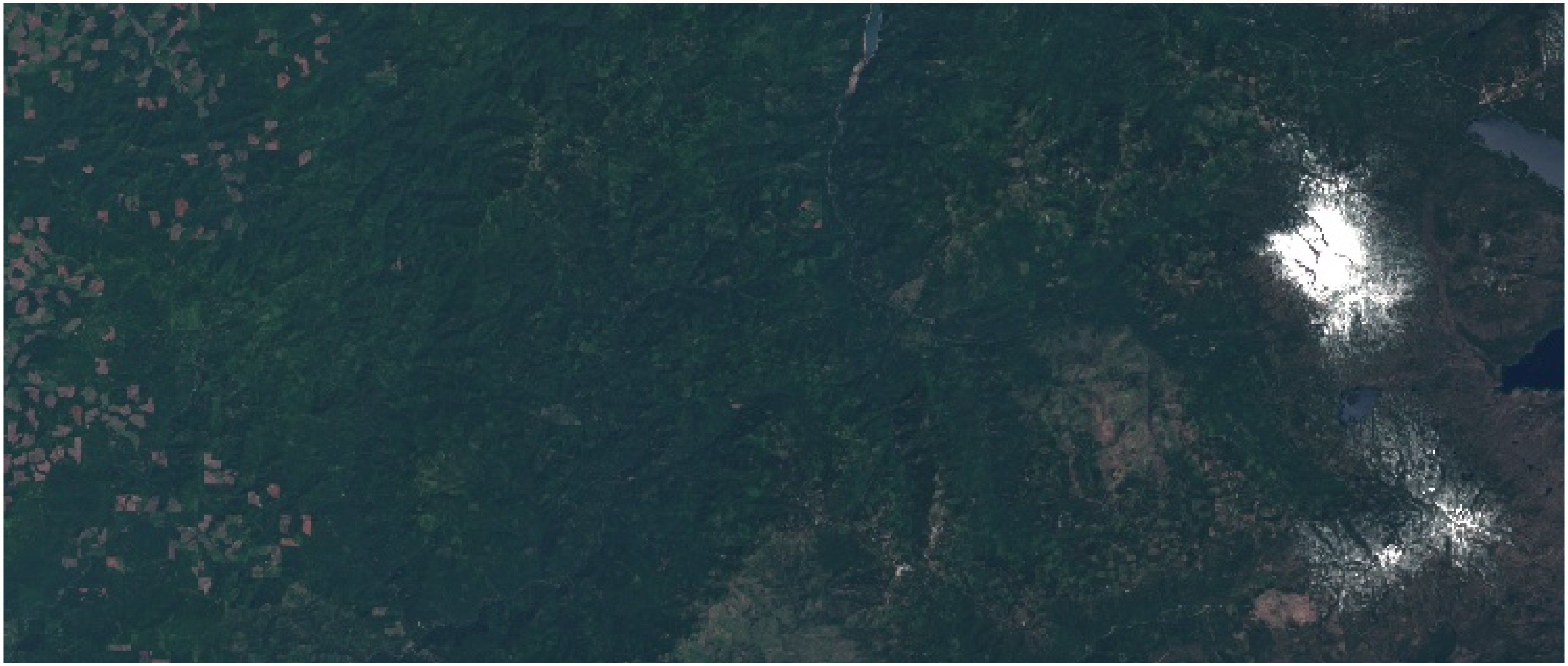
```
Count:  10

Sentinel 2 Date range:  {'type': 'Date', 'value': 1499022428070} {'type': 'Date', 'value'
```

```
#Get the timestamp and convert it to a date.
date = ee.Date(image.get('system:time_start')).getInfo()
# print(date['value'])
x=date['value']
print(datetime.datetime.fromtimestamp(x/1000))
```

```
2017-07-02 19:07:08.070000
```

An image from July 2, 2017. View this 'least cloudy' image.

```
visParams = {'bands': ['B4', 'B3', 'B2'], 'max': 3000, 'dimensions':1000}
thumbnail = image.getThumbUrl(visParams)
Image(url=thumbnail)
```



Calculate and view NDVI

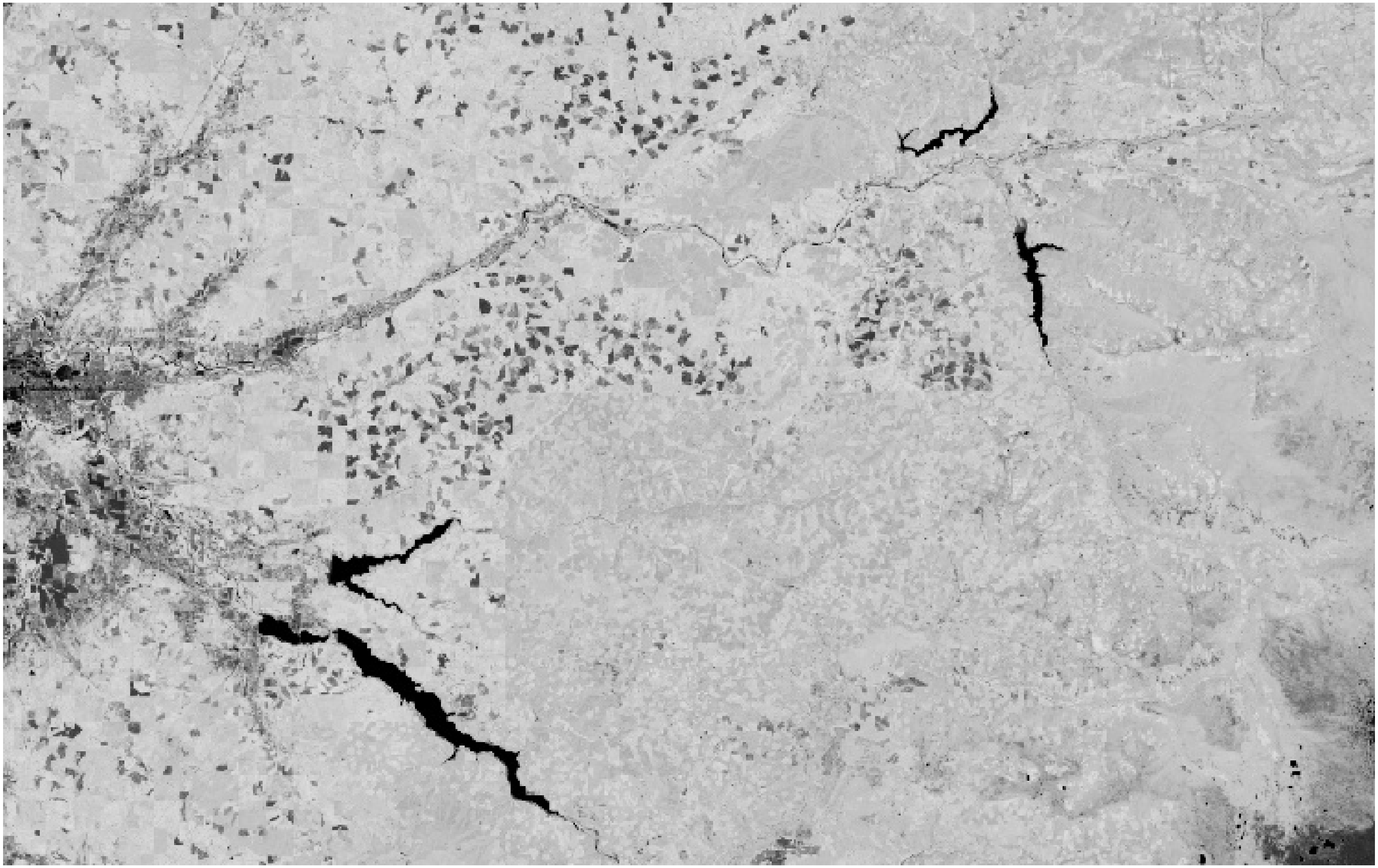
This image is much better.

The NDVI is calculated with a ratio of two wavelenths of reflected energy - 'red' and 'near-infrared'.

Higher NDVI values should correspond to larger amounts of green vegetation. To calculate this, run the Sentinel-2 NDVI function made earlier in the exercise on the existing image.

```
# run the sentinel NDVI function
sndvi = senNDVI(image)


# change visualization setting for the NDVI data
```

```
visParams = {'min':0, 'max': 1, 'dimensions':1000}
thumbnail = sndvi.getThumbUrl(visParams)
Image(url=thumbnail)
```



Brighter areas in the image should represent areas with higher amounts of growing green vegetation.

View scene interactively

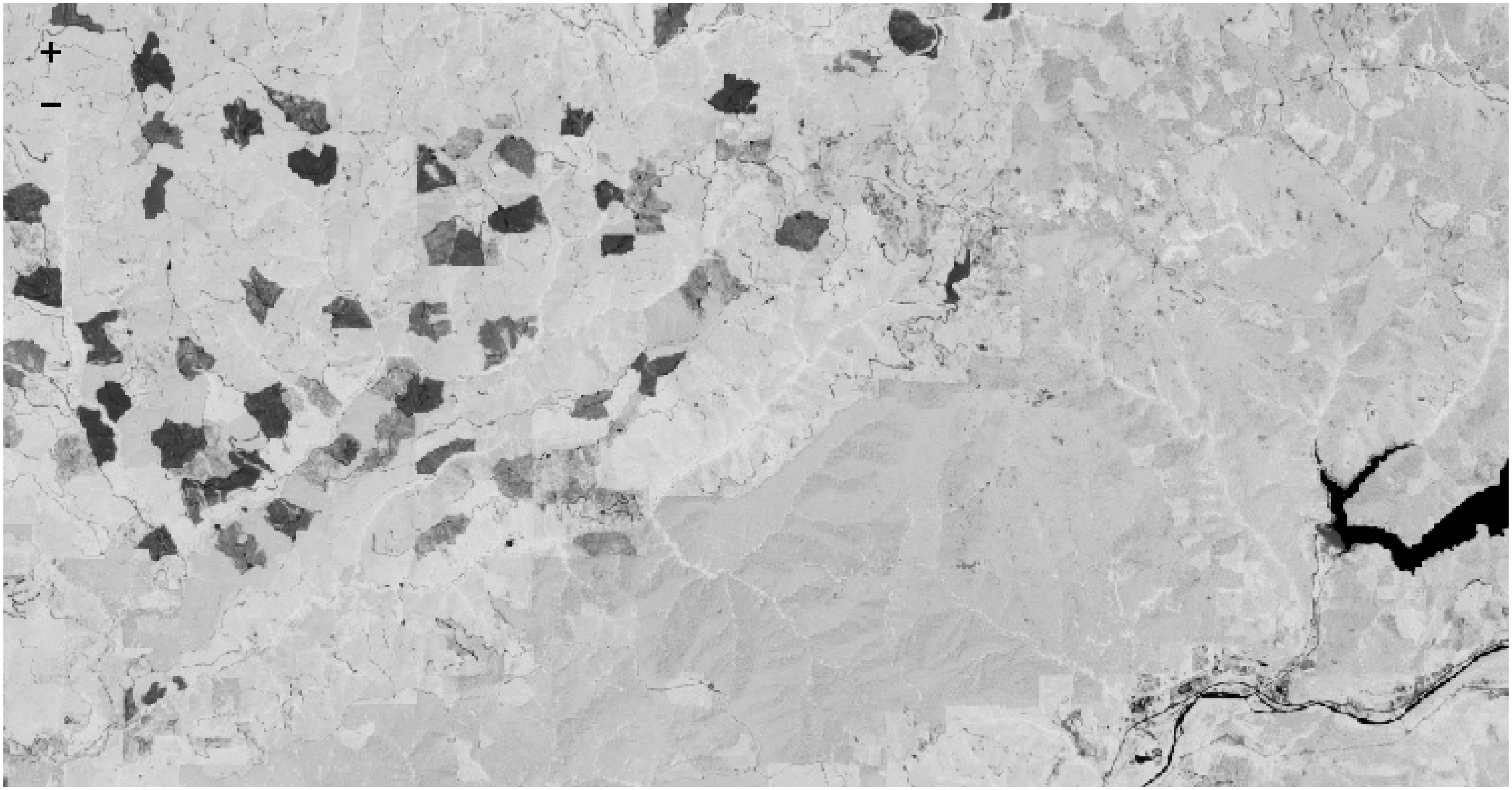
To examine whether NDVI is actually showing growing vegetation more closely, an interactive map is useful.

The *folium* python library can be used to view this same data on an interactive map. Multiple layers and other graphics can also be viewed, ans the visualization saved as an .html file.

First, lets get the parameters for visualization set up.

```
# visualization settings
rgbParams = {'bands': ['B4', 'B3', 'B2'], 'max': 3000, 'gamma': [1.4,1.2, 1.4]}
ndviParams = {'min':0, 'max': 1}
```

```
# flip the point coordinate to be lat, long (y, x) for folium
f = folium.Figure(width=800, height=600)
Map = geemap.Map(center=rnapoint[::-1], zoom= 12).add_to(f)
Map.addLayer(image, rgbParams, 'Natural Color Sentinel-2');
Map.addLayer(sndvi, ndviParams, 'NDVI')
Map.addLayerControl()
f
```



Find the least cloudy recent image and examine

Now, to compare the above image (from 2017) with a more recent, relatively cloud free image.

```
# Limit the collection to the 10 most recent images.
recent = collection.sort('system:time_start', False).limit(10)

sorted = recent.sort('CLOUD_COVERAGE_ASSESSMENT')

# Get the first (least cloudy) image
```

```
# Get the first (least cloudy) image.
recentqa = sorted.first()

recentndvi = senNDVI(recentqa)


#Get the timestamp and convert it to a date.
date = ee.Date(recentqa.get('system:time_start')).getInfo()
# print(date['value'])
x=date['value']
print(datetime.datetime.fromtimestamp(x/1000))
```

```
2020-10-09 19:12:28.376000
```

An image from October 9, 2020

In this run, it is Oct. 9th, 2020.

Lets look at this data, and add the point used to 'filter' the collection geographically as well.

```
# flip the point coordinate to be lat, long (y, x) for folium
f = folium.Figure(width=800, height=600)
Map = geemap.Map(center=rnapoint[::-1], zoom= 12).add_to(f)
Map.addLayer(image, rgbParams, 'Before Natural Color')
Map.addLayer(recentndvi, ndviParams, 'NDVI recently')
Map.addLayer(sndvi, ndviParams, 'NDVI before')
Map.addLayerControl()
folium.Marker(rnapoint[::-1], popup='<i>Research Natural Area</i>').add_to(Map)

f
```


Latitude: 44.157
Longitude: -122.3297
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright)., Google,
Earth Engine

Use image math to compare the images

Compute a difference image of NDVI between the two dates.

Image math is relatively straightforward with earth engine (and it could also be calculated outside earth engine with the 'numpy' and related modules.

```
# create image expression
# the 'subtract() mathmatical operator could also be used
# dif = recentndvi.subtract(sndvi)
dif = image.expression('new-old', {'new': recentndvi, 'old': sndvi})
difParams = {'min':-1, 'max': 1, 'palette': ['red', 'grey', 'green']}
```

Examine the result - redder areas have less green vegation recently compared to 2017, while green areas have more.

```
# plot the difference image along with the other layers
f = folium.Figure(width=800, height=600)
# flip the point coordinate to be lat, long (y, x) for folium
Map = geemap.Map(center=rnapoint[::-1], zoom= 11).add_to(f)
Map.addLayer(image, rgbParams, 'Before Natural Color')
Map.addLayer(recentqa, rgbParams, 'Recent Natural Color')
Map.addLayer(recentndvi, ndviParams, 'NDVI recently')
Map.addLayer(sndvi, ndviParams, 'NDVI before')
Map.addLayer(dif, difParams, 'dif')
Map.addLayerControl()
folium.Marker(rnapoint[::-1], popup='<i>Research Natural Area</i>').add_to(Map)
```

f

Assignment: Geog 485

What explains the changes visible in the NDVI difference image?

Greener areas are estimated to have more growing green vegetation recently, while redder areas are estimated to have less. Think about the time difference between the images as well.

- seasonality,
- atmosphere,
- solar orientation,
- actual LULC change

There is likely more than one factor at play - don't forget to look at the images from the two dates in addition to the NDVI difference image for your answer - and to examine the entire image area using the zoom and pan tools.

[ ] ↳ *34 cells hidden*

The area burned by the Holiday Farm fire seems to be identified, but lots of other areas are marked as different in the image. Basically, it looks like mostly water and shadow is not different classed in the unsupervised signatures.

Assignment: Geog 485

Comment on the classification and the potential issues with this change analysis. Suggest some ways in which the 'classified' change analysis could be improved by modifying the method of classification used, the choice of spectral bands or image dates, or the use of derived bands or other accompanying types of 'ancillary' data.

Assignment Geog 491

Choose 2 of the following three options, and turn in a commented notebook (.ipynb format) and PDF printout of that notebook showing the output for each of the 2 options chosen.

Options:

1. Do the same type of analysis but at another location.
2. Do the same type of analysis, but using the Landsat 8 data collection.
3. Using either Landsat 8 or Sentinel 2 data, add another spectral indice to the time series analysis or change analysis

For each of the 2 options you choose, create a new notebook (you can make a copy of the existing notebook and change the parameters/code). Turn in a copy of each notebook in .ipynb form, and also a PDF printout of each notebook.

Other resources for using Python with Google Earth Engine related to the exercise.

- ImageCollection Overview
- Image Overview
- Google Earth Engine Python API Examples