Visualize deforestation levels with geospatial images & Amazon SageMaker using Sentinel-2 dataset from ASDI

Abstract:

This article demonstrates geospatial analysis techniques to visualize and quantify deforestation using Sentinel-2 satellite imagery. It leverages Amazon SageMaker and open datasets from the Amazon Sustainability Data Initiative (ASDI) to process time series imagery capturing landscape changes related to wildfires near Paradise, California. After configuring access to the Sentinel-2 data registry, bounding boxes and date ranges isolate relevant pre-treatment and post-treatment scenes bracketing major fire events. Comparative visualization highlights patterns of healthy forest persistence versus zones of more complete canopy removal post-fire. Suggestion is to move such analytical routines into automated pipeline to enable scalable automated deforestation mapping as new Sentinel-2 observations become available over time. Overall, this article demonstrates core techniques for leveraging cloud-based geospatial data and computing tools to derive actionable intelligence maps and indicators pertinent to sustainability challenges like wildfire impacts and climate adaptation.

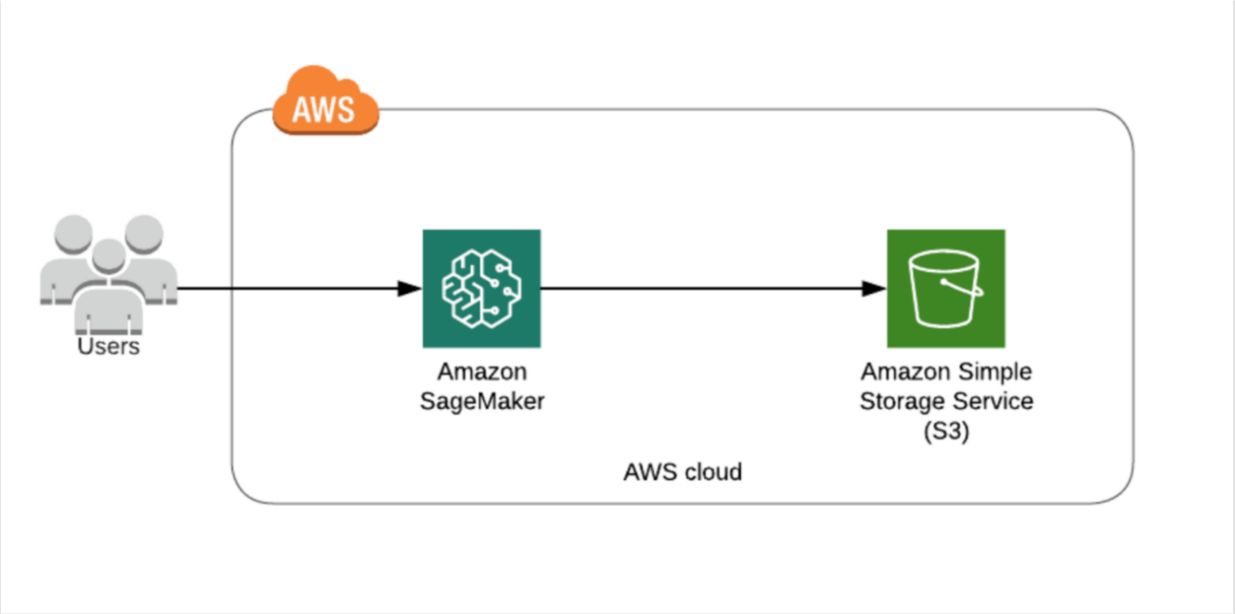
Deforestation has reached critical levels globally, with potentially irreversible consequences for environmental sustainability and exacerbation of climate change. Widespread forest fires occur year-round in regions spanning the Amazon rainforest in Brazil to the western United States, indicating severe ecosystem disruption. Deforestation eliminates natural carbon sinks - one of the few remaining mechanisms buffering anthropogenic climate impacts. Sustainability solutions could benefit from incorporation of relevant datasets, whether customer-generated (e.g. building temperatures, vehicle locations) or external (e.g. weather patterns, satellite imagery). Hosting datasets centrally on platforms like AWS facilitates customer access without transfer/storage burdens, allowing faster development. Landsat 8 imagery serves as one example - while petabyte-scale and requiring significant pre-processing, AWS hosts the catalog publicly, reducing barriers to leveraging Earth observation data. Overall, AWS centralized datasets help customers focus on core sustainability applications rather than data wrangling.

The AWS Open Data Program is the umbrella for all our free data sets, and the Amazon Sustainability Data Initiative (ASDI) manages the subset of those related to sustainability. ASDI also enables data-sharing partnerships aimed at addressing sustainability challenges, such as the recently announced collaboration with Open Source-Climate. This organization leverages open data to evaluate and mitigate climate-related financial risks for investors, corporations, and communities. Through such partnerships, ASDI facilitates access to crucial data for developing innovative solutions to pressing sustainability issues.

Let’s run a sagemaker notebook with advanced visualizations

This notebook provides an introduction to performing geospatial data analysis related to sustainability on SageMaker Studio. We begin by exploring the Sentinel remote sensing dataset from the AWS Open Data Registry. Using Sentinel-2 data, we analyze deforestation by calculating different spectral indices. As an example, we look at Paradise Fire disaster in California - similar methods can be applied to identify areas of forest loss over time. The goal is to demonstrate core concepts in working with Earth observation data for sustainability applications within the SageMaker environment.

S3 serves as the centralized persistent store for images. SageMaker integrates natively with S3 for data access for training and inference workloads, abstracting away much of the heavy lifting of data transfers. This makes it convenient to use images from S3 efficiently for ML development while persistently storing them in a scalable, durable storage layer like S3.



Step 1: Install Packages

Setting up environments in SageMaker Studio is straightforward since Conda is installed by default. Required Python packages can be installed using 'pip install' commands. The packages listed below are required for this workshop - they only need to be installed once when first creating a notebook. Leveraging Conda and pip helps streamline environment configuration for development in SageMaker Studio.

```
%pip install pandas
%pip install numpy
%pip install geopandas
%pip install matplotlib
%pip install sentinelhub
%pip install rasterio
%pip install earthpy
%pip install awswrangler
```

Step 2: Import Packages

Once the appropriate Conda environment has been configured and selected or required packages manually installed in a SageMaker Studio notebook instance, the Python libraries can be directly imported to enable development. The integration of Conda and pip with SageMaker Studio notebooks allows for reproducible, sharable software environments tailored to a project's needs, consistent with research computing best practices. By proactively managing the notebook environment early in development, dependencies can be established to facilitate smoother workflow execution, analysis, collaboration, and review. Overall, conscious creation of computing environments removes technical barriers and enables fuller focus on the research questions at hand within SageMaker Studio.

```
import pandas as pd
import numpy as np
import geopandas as gpd
from shapely.geometry import Point
import matplotlib
import matplotlib.pyplot as plt
import os
import warnings
import datetime
import json
import boto3
import gc
import rasterio as rio
import os
import earthpy.spatial as es
import earthpy.plot as ep
import imageio
import io
import awswrangler as wr
import json

%matplotlib inline
warnings.filterwarnings('ignore')
```

Step 3: Working With Geospatial Images

For geospatial analysis, Sentinel-2 satellite imagery will be utilized as the remote sensing dataset given its public availability through the AWS open data registry. The Sentinel-2 constellation provides ongoing high-resolution multispectral coverage as the continuation of prior SPOT and Landsat Earth observation programs. The sentinelhub Python package facilitates straightforward search and access to Sentinel-2 scenes relevant to the area of interest from the registry. Incorporation of Sentinel-2 facilitates time-series analysis of landscape dynamics pertinent to sustainability topics. The AWSD registry lowers barriers to leveraging Sentinel-2 and similar public domain remotely sensed data.

```
from sentinelhub import (
    MimeType,
    CRS,
    BBox,
    SentinelHubRequest,
    SentinelHubDownloadClient,
    DataCollection,
    bbox_to_dimensions,
    DownloadRequest
)
```

3. 1 - SENTINEL HUB SETUP

Proper configuration of access credentials is required to utilize the Sentinel Hub API for retrieval of remotely sensed data. An optional JSON file stores the authentication parameters to enable programmatic queries while keeping sensitive tokens secure. Specifically, a valid Sentinel Hub instance ID is necessary and suffices for read-only data access. Abstracting the credentials into a separate file facilitates sharing of analysis code without compromising security. More broadly, use of APIs and environmental variables for handling credentials promotes reproducibility and collaboration by avoiding hard-coding of tokens within software projects. With credentials configured appropriately, the analysis can then focus solely on core research questions without impedance from security control permissions.

```
from sentinelhub import SHConfig
config = SHConfig()
```

```
# instance_id - Instance ID from from your Sentinel Hub account
config.instance_id = 'please update with your instance id'

config.save()
# Verify credentials

from sentinelhub import WebFeatureService, BBox, CRS, DataCollection, SHConfig
if config.instance_id == '':
print("Warning! To use WFS functionality, please configure the 'instance_id'.")
```

3.2 - DATA SEARCH

Defining the spatiotemporal domain is an essential initial step for targeted retrieval of remotely sensed data. Our area of interest centers on Paradise, CA, impacted by recent wildfires. We specify geospatial boundaries via a bounding box encompassing Paradise and select a time range spanning pre- and post-fire landscapes. Explicitly delineating the target geographic extent and time window provides precise criteria for the API to identify and retrieve relevant Sentinel satellite observations. Tailored queries focusing on the research aims help filter and collect imagery capturing landscape change dynamics within the region of study. Careful query formulation reduces data volumes for storage and analysis while isolating observations most likely to inform about fire ecology in Paradise. Explicit articulation of retrieval parameters promotes reproducibility and interpretability as well when sharing or publishing analytical findings.

```
# Specify bounding box and time interval for search

#california paradise after fire
search_bbox = BBox(bbox=[-121.666536,39.708771,-121.542266,39.792182],crs=CRS.WGS84)
#before fire
#search_time_interval = ('2018-11-01T00:00:00', '2018-11-01T23:59:59')
#after fire
search_time_interval = ('2019-01-10T00:00:00', '2019-01-10T23:59:59')

wfs_iterator = WebFeatureService(
    search_bbox,
    search_time_interval,
    data_collection=DataCollection.SENTINEL2_L1C,
    maxcc=0.6,
    config=config
)

for tile_info in wfs_iterator:
    print(tile_info)
```

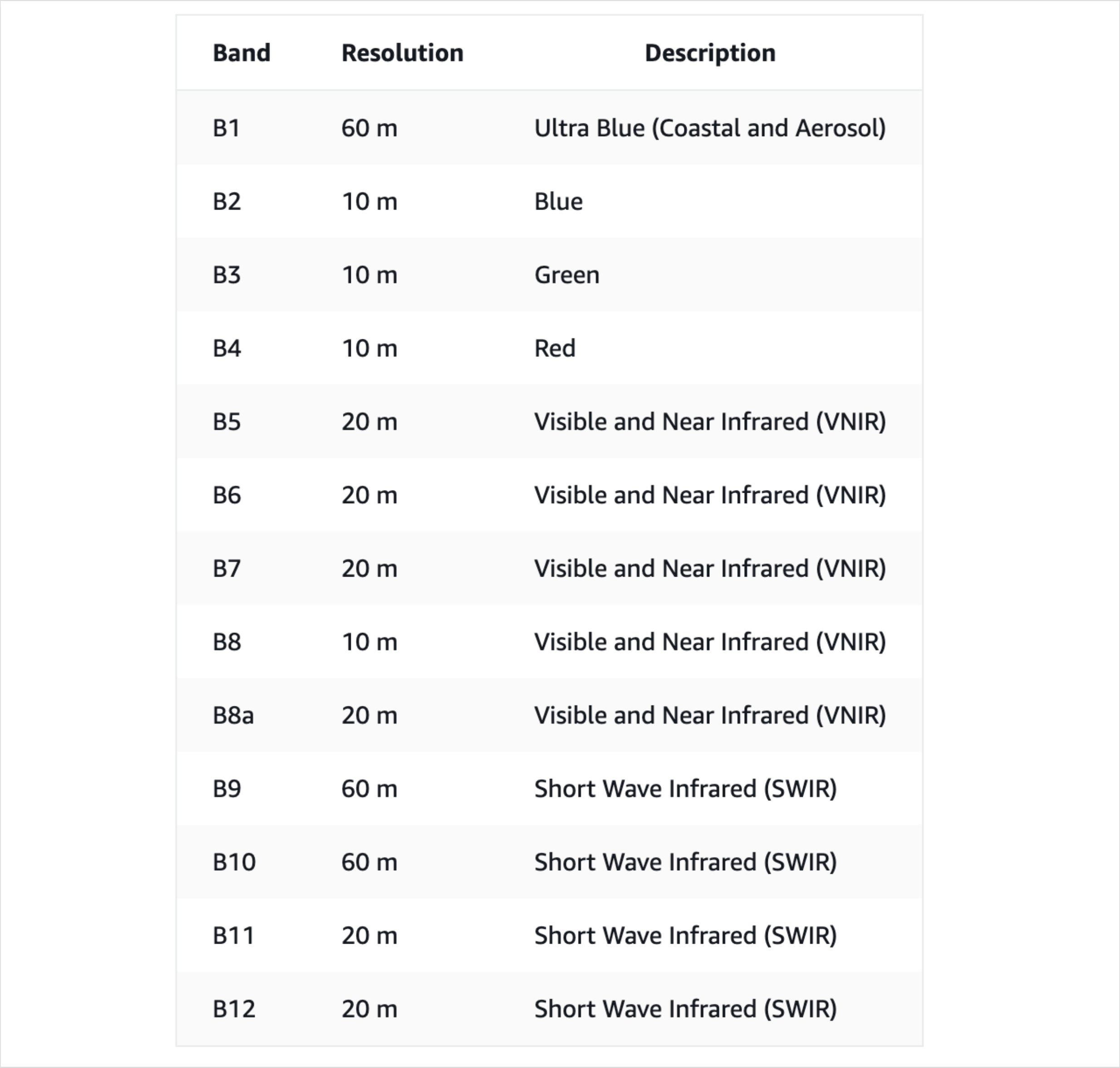
```
output
{'type': 'Feature', 'geometry': {'type': 'MultiPolygon', 'crs': {'type': 'name', 'prop
{'type': 'Feature', 'geometry': {'type': 'MultiPolygon', 'crs': {'type': 'name', 'prop
```

```
##before fire bucket - s3://sentinel-s2-l1c/tiles/10/T/FK/2018/11/1/0
##after fire bucket - s3://sentinel-s2-l1c/tiles/10/T/FK/2019/1/10/0
```

3.3 - WORKING WITH GEOSPATIAL IMAGES

For geospatial analysis in this research, Sentinel-2 satellite imagery is utilized given its public availability through the AWS open data registry. The Sentinel-2 constellation offers ongoing high-resolution multispectral Earth observations across 13 bands spanning the visible, near infrared, and shortwave infrared wavelengths. This continues prior Landsat and SPOT programs' systematic collection of optical remote sensing data useful for environmental monitoring. Specifically, each Sentinel-2 satellite carries a MultiSpectral Instrument sensor designed to capture spectral reflectance signatures of ground surface materials across the specified bands at 10-60 meter pixel resolution. Further technical specifics on the instrument band designations and purposes are available in the accompanying references. When analyzed sequentially or in combination, these spectra facilitate study of landscape dynamics like vegetation growth, burn recovery, and other phenomena pertinent to ecological sustainability challenges. Incorporation of open access Sentinel-2 data thereby enables timely investigation of research questions relying on Earth observation resources.

The Sentinel-2 satellites each carry a single multi-spectral instrument (MSI) with 13 spectral channels in the visible/near infrared (VNIR) and short wave infrared spectral range (SWIR). You can read more about these bands here.



Band	Resolution	Description
B1	60 m	Ultra Blue (Coastal and Aerosol)
B2	10 m	Blue
B3	10 m	Green
B4	10 m	Red
B5	20 m	Visible and Near Infrared (VNIR)
B6	20 m	Visible and Near Infrared (VNIR)
B7	20 m	Visible and Near Infrared (VNIR)
B8	10 m	Visible and Near Infrared (VNIR)
B8a	20 m	Visible and Near Infrared (VNIR)
B9	60 m	Short Wave Infrared (SWIR)
B10	60 m	Short Wave Infrared (SWIR)
B11	20 m	Short Wave Infrared (SWIR)
B12	20 m	Short Wave Infrared (SWIR)

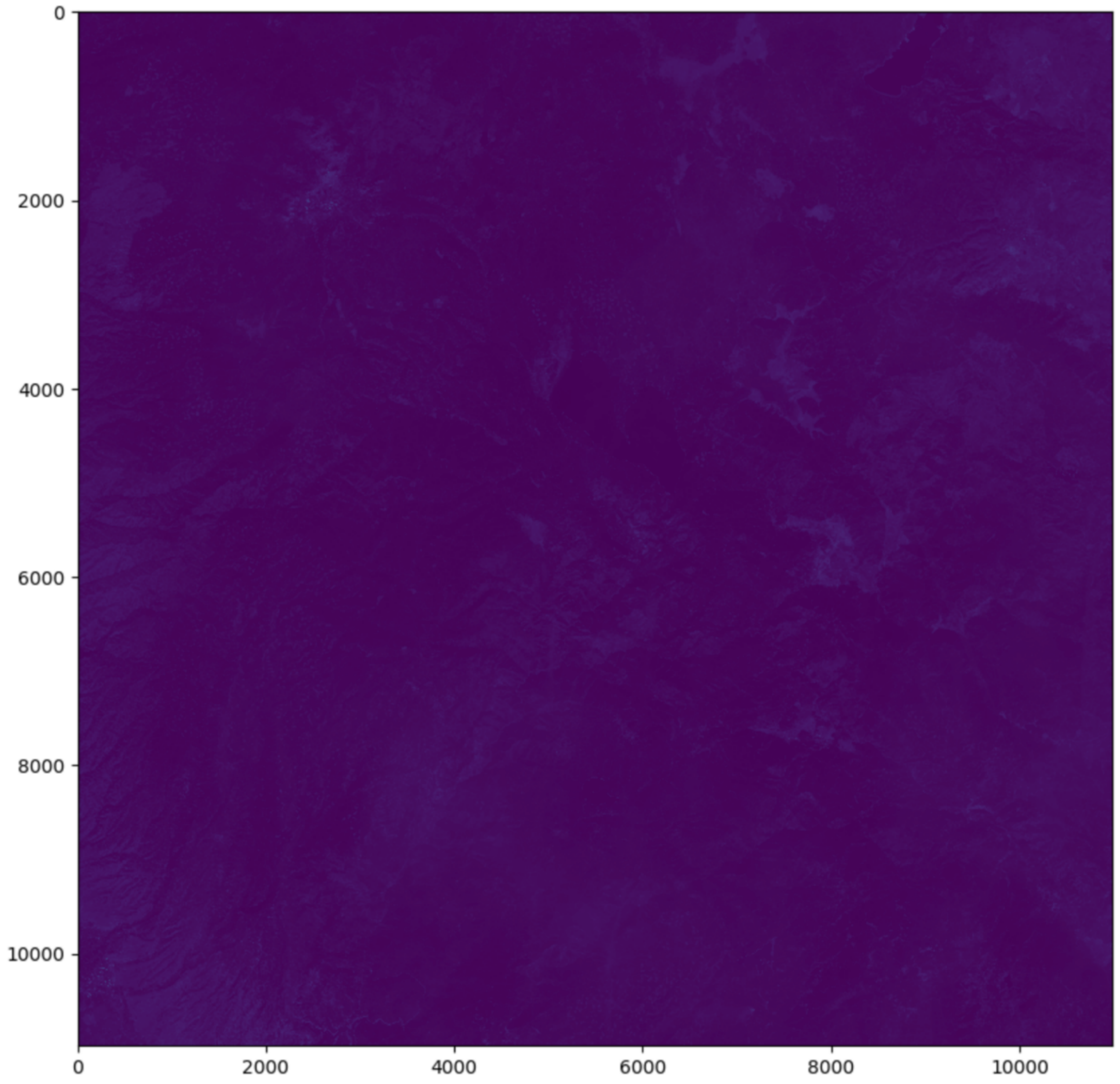
3.4 - WORKING WITH RASTER DATA

Geospatial data consists fundamentally of either raster (gridded) or vector (feature-based) representations. The Sentinel-2 satellite imagery utilized here is stored as GeoTIFFs - raster datasets encoding Earth observation data and digital elevation models. Rasterio provides a Python library for programmatic manipulation of such gridded geospatial data. By reading Sentinel-2 scenes into raster array structures via Rasterio, the red, green, and blue spectral bands can be combined into composite true color images for interpretation and analysis. Working natively with the raster arrays facilitates customizable image processing, indexing to areas of interest, application of filters, etc. The tight integration of Rasterio functionality into the Python geospatial software ecosystem allows researchers to focus more fully on their Earth observation data science tasks rather than data wrangling challenges. Overall, Rasterio enables flexible, reproducible research workflows essential for the advancement of open geospatial techniques applied to urgent sustainability issues.

```
from rasterio import plot
from rasterio.plot import show
from rasterio.session import AWSSession
from rasterio.windows import Window
```

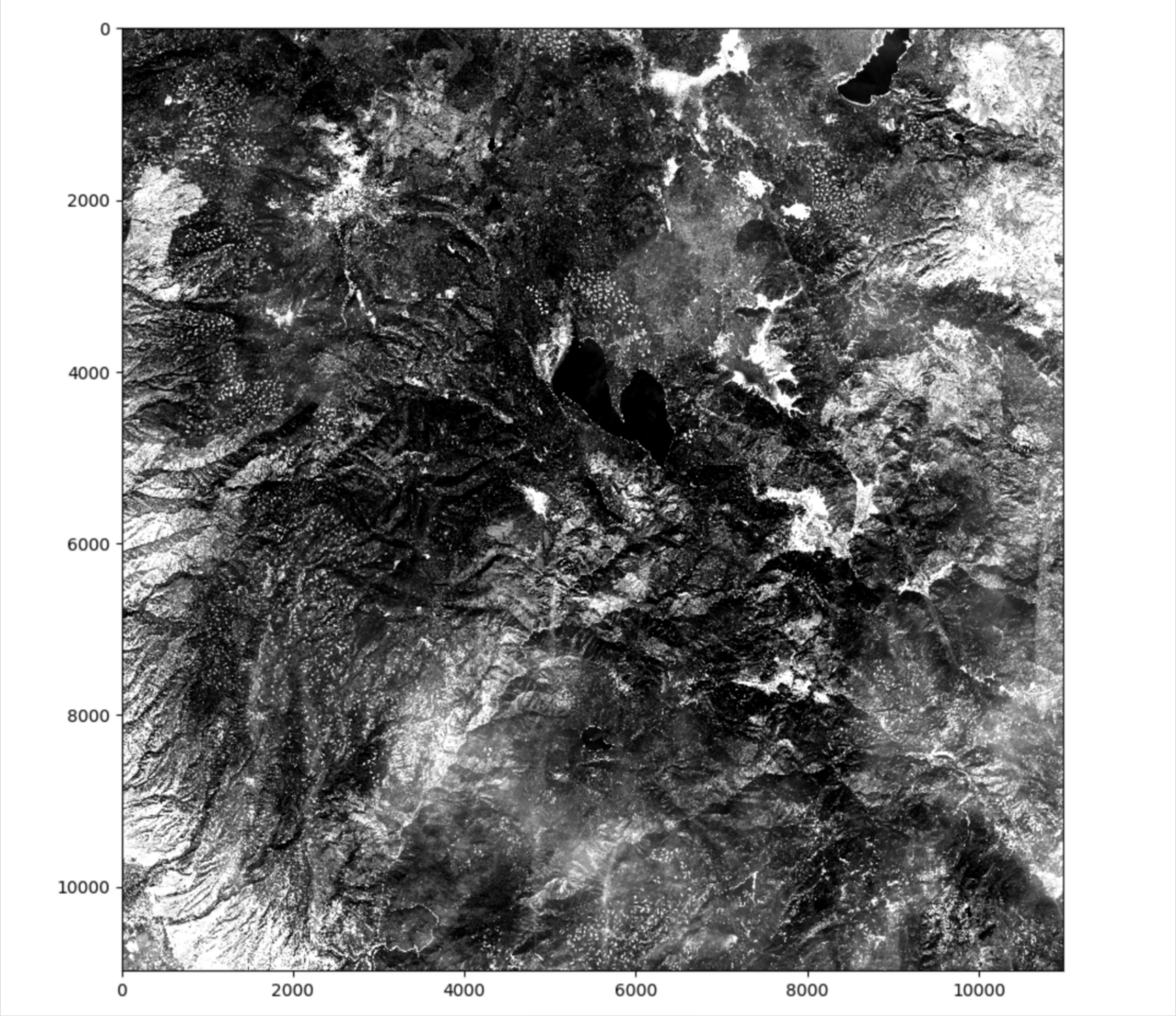
We will use the **AWSSession** object to mark **requester_pays** true to request the file directly from sentinel.

```
aws_session = AWSSession(boto3.Session(), requester_pays=True)
with rio.Env(aws_session):
    with rio.open('s3://sentinel-s2-l1c/tiles/10/T/FK/2018/11/1/0/B04.jp2') as src:
        red = src.read()
    plt.figure(figsize=[10,10],num=1, clear=True)
    show(red) #RdYlGn
    plt.show()
```



The raw Sentinel-2 satellite image appears dark and details are obscured to the human eye. To better visualize and interpret key features, contrast enhancement and color transformation techniques common in remote sensing image processing are applied. Specifically, linear contrast stretching improves the dynamic range devoted to intensities within the region of interest. Additionally, converting the native spectral band combinations into displays more intuitive for human perception based on long-standing color modeling research facilitates more rapid digitization and annotation. Such pre-processing steps are essential to convert the raw sensor observations into derivations by visual analytics. They remove barriers to geospatial time series analysis while retaining the underlying radiometric fidelity. Appropriate image transformations enable the integration of human semantic understanding about landscape change with automated algorithms. This helps advance spatiotemporal detection of phenomena related to sustainability challenges.

```
vmin, vmax = np.nanpercentile(red, (5,95)) # 5-95% contrast stretch
plt.figure(figsize=[10,10])
show(red, cmap='gray', vmin=vmin, vmax=vmax)
plt.show()
```

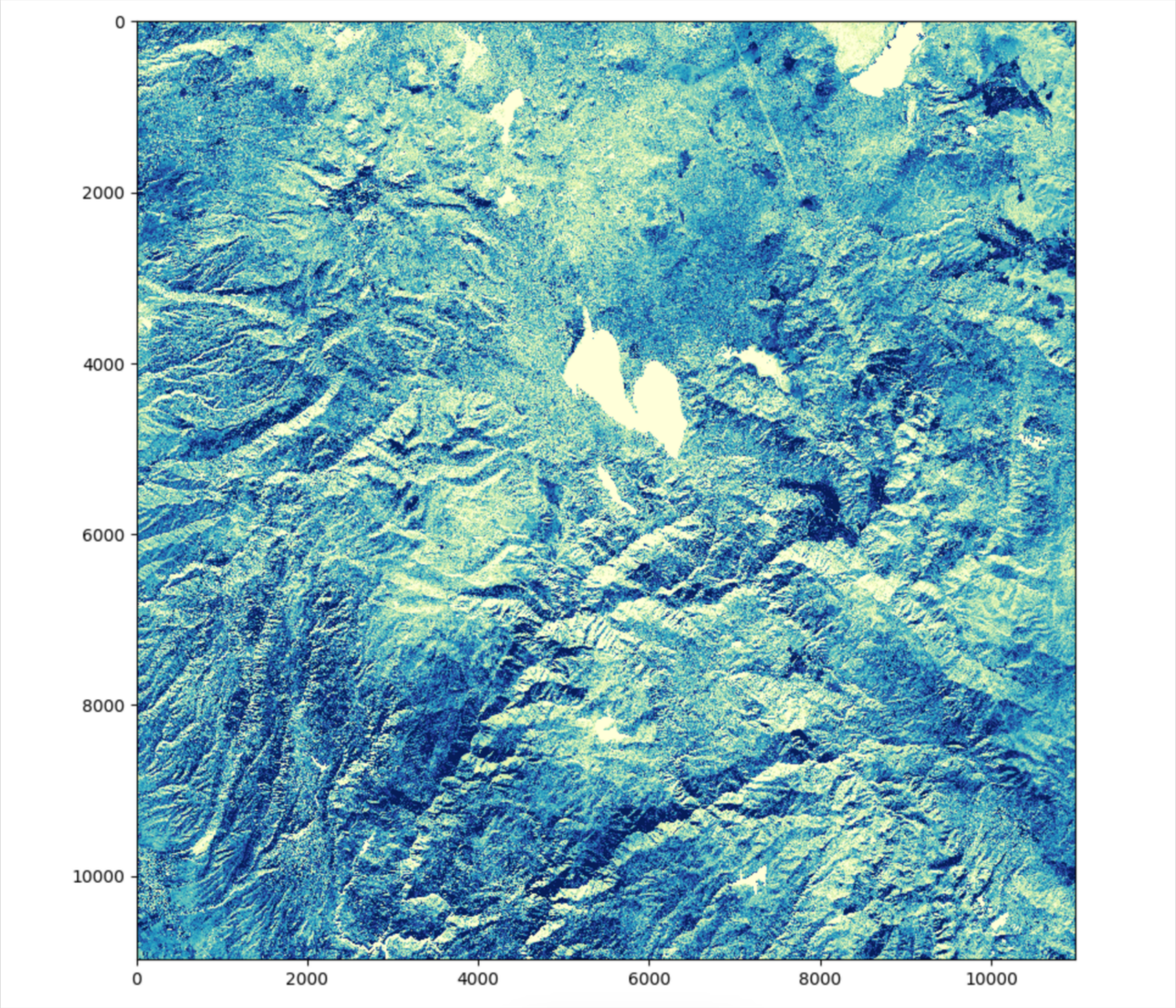


Beyond human-perceived color wavelengths, Sentinel-2 captures reflectance in non-visible near infrared bands useful for vegetation monitoring. By assigning alternative color schemes to these infrared image layers, different aspects of landscape composition and structure are revealed. For example, healthy vegetation appears brighter due to higher near infrared reflectivity than soil or dormant plants. Using such infrared-derived indices facilitates quantitative measurement of biophysical variables like leaf area index and evapotranspiration over time. Appropriate color mappings make the often unfamiliar infrared spectral dimensions more interpretable for visual training data labeling as well. More broadly, leveraging the full spectrum sensing capacities of Sentinel-2 via tailored visualization choices enables generation of scientific insights and derived products not possible from the visible bands alone. Multi-modal visualization approaches which transform the raw sensor observations into more perceptually intuitive formats help translate Earth observation data into actionable intelligence around sustainability challenges.

```
aws_session = AWSSession(boto3.Session(), requester_pays=True)
with rio.Env(aws_session):
    with rio.open('s3://sentinel-s2-l1c/tiles/10/T/FK/2018/11/1/0/B08.jp2') as src:
        nir = src.read()

vmin, vmax = np.nanpercentile(nir, (5,95))  # 5-95% contrast stretch
plt.figure(figsize=[10,10],num=1, clear=True)
show(nir, cmap='YlGnBu', vmin=vmin, vmax=vmax) #RdYlGn #YlGnBu
```

```
plt.show()
```



CALCULATING SPECTRAL INDICES

Spectral indices constitute an essential image processing technique for remote sensing applications. An index combines pixel intensity values across defined bands to accentuate or filter specific ground components based on their spectral reflectance signatures. For instance, common vegetation indices leverage contrast between near infrared brightness and visible wavelength absorption to highlight photosynthetically active regions. Other indices accentuate burn scars, soil exposures, water bodies and snow cover. Carefully constructed spectral indices thus generate derived image layers emphasizing the relative abundance of targeted land cover categories pertinent to sustainability research questions. Over the time series, their quantified pixel values track spatiotemporal dynamics far more specifically than raw broadband intensity alone.

NORMALIZED DIFFERENCE VEGETATION INDEX - NDVI

The normalized difference vegetation index (NDVI) constitutes a widely used spectral index in remote sensing research for quantitative characterization of live green vegetation coverage and condition. Its formulation exploits the distinct reflectance signatures of chlorophyll absorption in the visible red wavelengths compared to high near-infrared reflectivity by leaf mesophyll structures. By difference ratios of these spectral bands, the resulting NDVI values indicate relative density and health status of vegetation canopies imaged by satellite sensors. Values range from -1 to 1 based on the vegetation contrasts, with higher positive indices corresponding to more extensive photosynthesizing foliage. Spatially and temporally, NDVI enables scientific monitoring of subtle changes in crop vigor, forest phenology, drought impacts on landscapes, and numerous other dynamics relevant to ecological sustainability and land use management. The sensitivity yet computational simplicity underpinning spectral indices makes them vital tools for policy-relevant Earth observation analytics. When validated against ground measurements, they provide robust quantifiable indicators of vegetation shifts essential for data-driven environmental decision making over time.

The formula for the normalized difference vegetation index is (B8-B4)/(B8+B4). While high values suggest dense canopy, low or negative values indicate urban and water features. It is calculated as NDVI = (B8 - B4) /(B8 + B4) or (NIR – Red) / (NIR + Red)

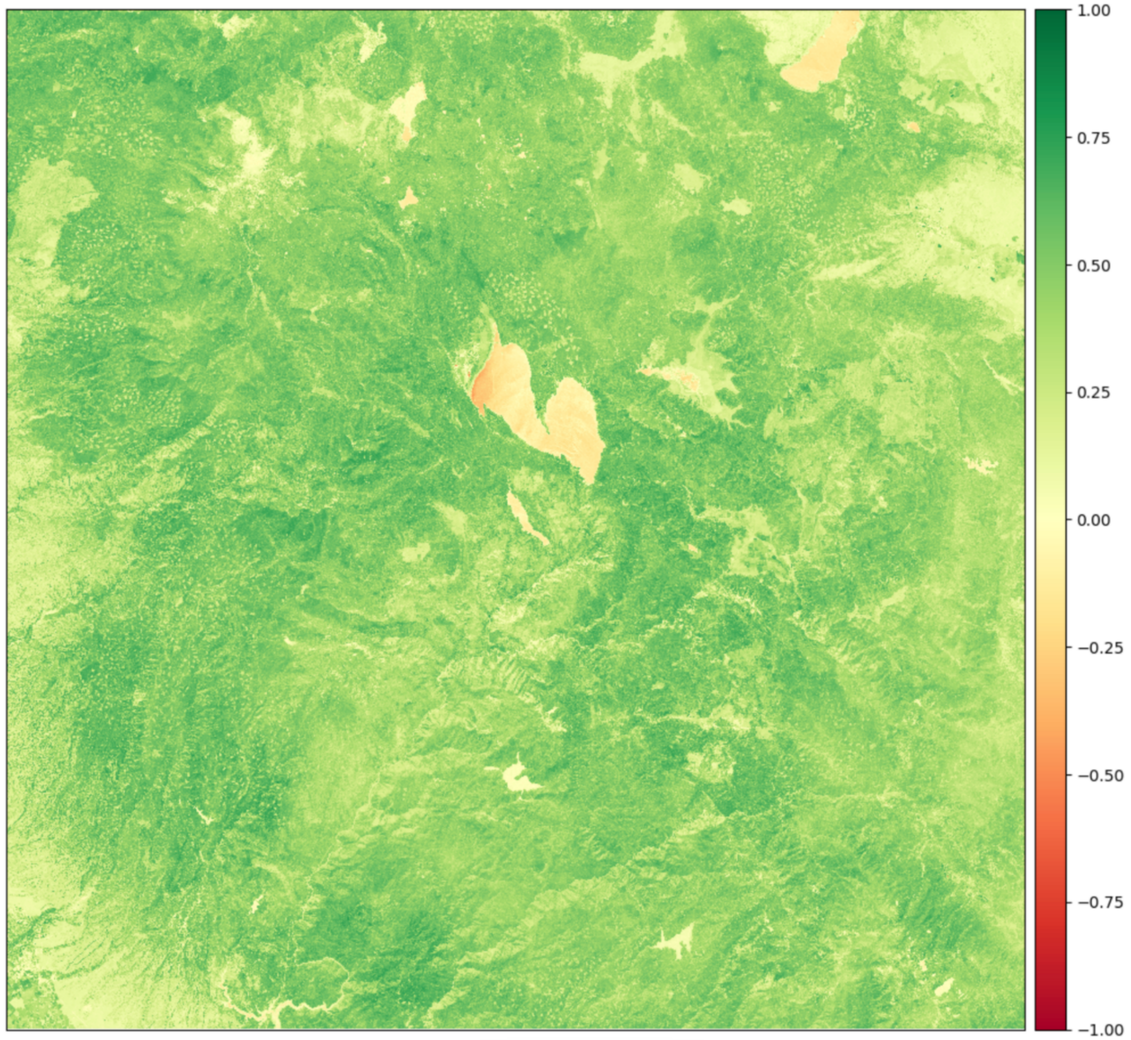
MOISTURE INDEX

The moisture stress index derived from remote sensing constitutes a useful indicator for detecting vegetation water deficits and associated health impacts. Its formulation incorporates contrasting sensitivities in the shortwave infrared domain associated with leaf liquid water absorption features and the moisture-insensitive near-infrared reflectance plateau. Higher index values represent higher inferred moisture content within the imaged vegetation canopies. Conversely, lower index values suggest plants under duress from insufficient available soil and atmospheric moisture, vulnerable to drought impacts. Spatiotemporal tracking enables quantification of moisture stress emergence and recovery cycles across landscapes. When validated against field measurements or climate reanalysis data, the trends quantify changing exposure risks over time that can inform proactive resource sustainability planning and climate adaptation policies specific to affected ecosystems and human communities dependent upon them. Derivation of biophysical variables like moisture stress facilitates translation of raw spectral observations into scientifically actionable intelligence for addressing pressing sustainability challenges.

It is calculated as **(B8A - B11) / (B8A + B11).** We have processed B4 and B8 bands so now we can calculate Normalized Difference Vegetation Index

```
#It calculated as NDVI = (B8 - B4) /(B8 + B4) or (NIR - Red) / (NIR + Red)
ndvisample = (nir.astype(float)-red.astype(float))/(nir.astype(float)+red.astype(floa
```

```
#we are using earthpy to visualize NDVI
ep.plot_bands(ndvisample, cmap="RdYlGn", vmin=-1, vmax=1);
```



```
#clean up memory if processes talking a lot of memory
#del red
#del nir
#del ndvisample
#gc.collect()
```

The visualization depicts a classified map distinguishing intact dense forest areas (dark green) from regions of dead or removed forest cover (lighter shades) within the domain of study. To enhance analytical interpretation, we leverage the EarthPy Python package to transform the continuous Normalized Difference Vegetation Index (NDVI) spectral index into binned categories. EarthPy facilitates straightforward plotting and analysis of remote sensing imagery bands. Here, it enables flexible visualization of the gradient of vegetated conditions derived from the NDVI ranges computed across the Sentinel-2 scene. Quantitative thresholds divide the index values into categorizations interpretable as grades of forest health and disturbance. Translating the continuous imagery into discrete thematic classes in this way simplifies identification of regions undergoing ecological transitions of interest. Over time, aggregation of NDVI-based forest change classifications produced using open access tools like EarthPy can feed indicators to guide reforestation interventions or quantify habitat conservation successes.

```
# Create classes and apply to NDVI resultsndvi_class_bins = [-np.inf, 0, 0.1, 0.25, 0.
ndvi_density_class = np.digitize(ndvisample, ndvi_class_bins)

# Apply the nodata mask to the newly classified NDVI data
ndvi_density_class = np.ma.masked_where(
    np.ma.getmask(ndvisample), ndvi_density_class
)
np.unique(ndvi_density_class)
```

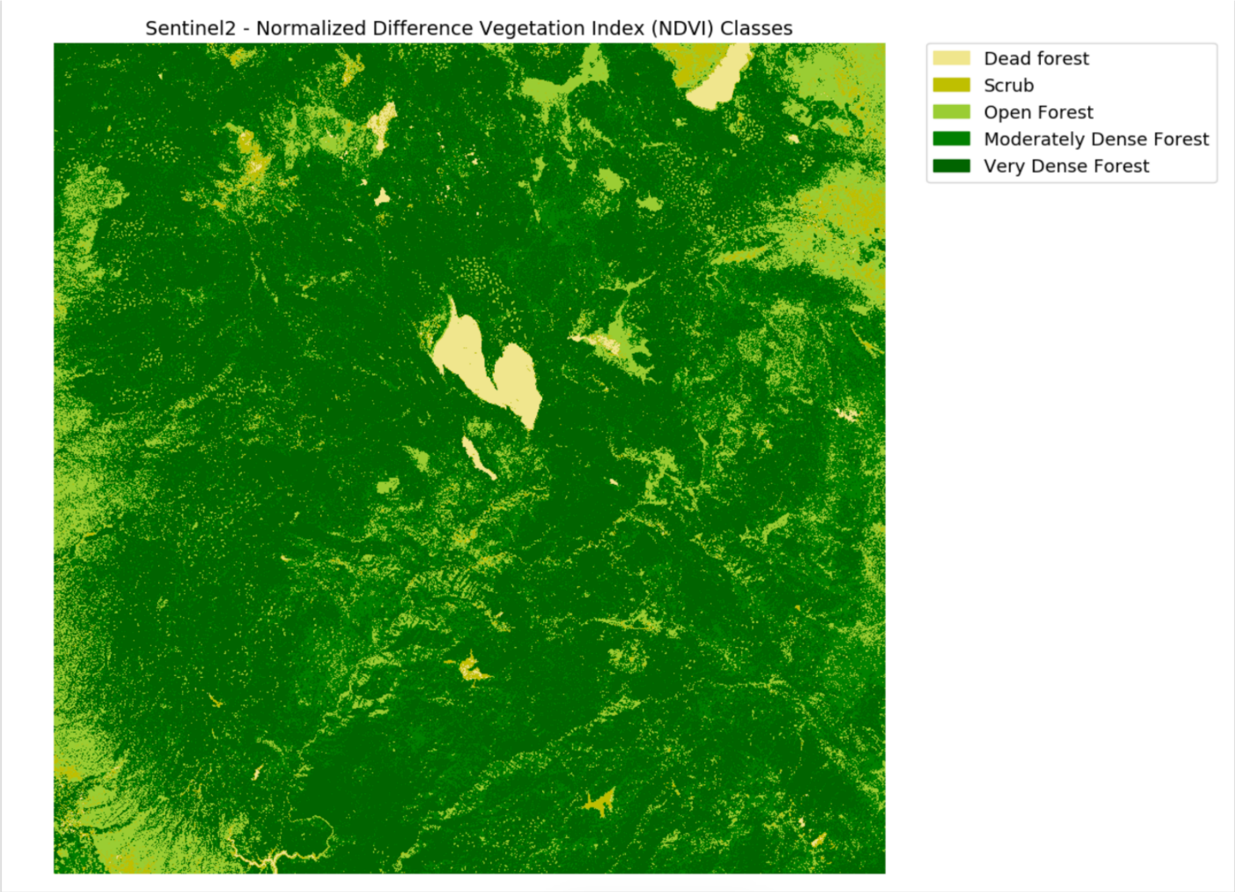
```
from matplotlib.colors import ListedColormap# Define color map
nbr_colors = ["khaki", "y", "yellowgreen", "g", "darkgreen"]
nbr_cmap = ListedColormap(nbr_colors)

# Define class names
ndvi_cat_names = [
    "Dead forest",
    "Scrub",
    "Open Forest",
    "Moderately Dense Forest",
    "Very Dense Forest",
]

# Get list of classes
classes = np.unique(ndvi_density_class)
classes = classes.tolist()
# The mask returns a value of none in the classes. remove that
classes = classes[0:5]
```

```
# Plot your data
fig, (ax1) = plt.subplots(1, figsize=(12, 12), num=1, clear=True)
im1 = ax1.imshow(np.squeeze(ndvi_density_class), cmap=nbr_cmap)
ep.draw_legend(im_ax=im1, classes=classes, titles=ndvi_cat_names)
ax1.set_title(
    "Sentinel2 - Normalized Difference Vegetation Index (NDVI) Classes",
    fontsize=14,
)

ax1.set_axis_off()
plt.tight_layout()
```



In this image we see multiple clusters of forest- '**Very Dense Forest**' to '**Dead forest**'. Now lets process the image one year post the paradise fire (above image).

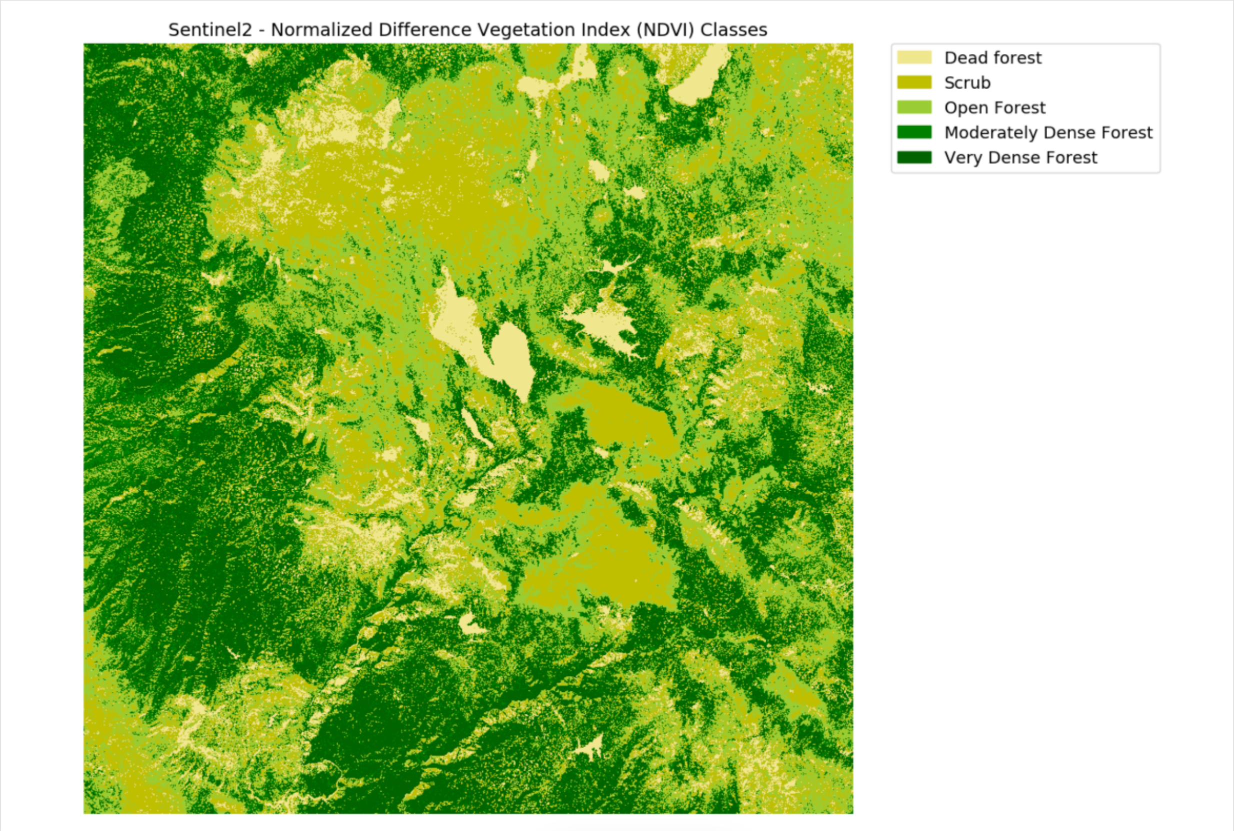
```
aws_session = AWSSession(boto3.Session(), requester_pays=True)with rio.Env(aws_sessio
    with rio.open('s3://sentinel-s2-l1c/tiles/10/T/FK/2019/1/10/0/B04.jp2') as src1:
        red = src1.read()
    with rio.open('s3://sentinel-s2-l1c/tiles/10/T/FK/2019/1/10/0/B08.jp2') as src2:
        nir = src2.read()
    ndvisample = (nir.astype(float)-red.astype(float))/(nir.astype(float)+red.astype(
    #ep.plot_bands(ndvisample, cmap="RdYlGn", vmin=-1, vmax=1);

ndvi_density_class = np.digitize(ndvisample, ndvi_class_bins)
# Apply the nodata mask to the newly classified NDVI data
ndvi_density_class = np.ma.masked_where(
    np.ma.getmask(ndvisample), ndvi_density_class
)
np.unique(ndvi_density_class)

# Get list of classes
classes = np.unique(ndvi_density_class)
classes = classes.tolist()
# The mask returns a value of none in the classes. remove that
classes = classes[0:5]
```

```python
# Plot your data
fig, (ax1) = plt.subplots(1, figsize=(12, 12),num=1, clear=True)
im1 = ax1.imshow(np.squeeze(ndvi_density_class), cmap=nbr_cmap)
ep.draw_legend(im_ax=im1, classes=classes, titles=ndvi_cat_names)
ax1.set_title(
    "Sentinel2 - Normalized Difference Vegetation Index (NDVI) Classes",
    fontsize=14,
)

ax1.set_axis_off()
plt.tight_layout()
```



The foregoing visualization depicts a classified map delineating intact dense forest areas (dark green) and zones of dead or removed forest cover (lighter tones) within the study region. Building on this initial exploration, analogous image processing and spectral index derivation methods can be encoded into serverless Lambda functions as constructed in prior sections. Deploying the algorithms for on-demand execution facilitates scalable automated analysis of new Sentinel-2 scenes as they become available from the geospatial archive. With appropriately tuned categorization thresholds, the Lambda functions will output classified deforestation maps highlighting areas of forest loss over time. These discrete change maps can subsequently inform downstream sustainability policy decisions or conservation actions targeting documented zones of concern. More broadly, wrapping reusable image analysis scripts into cloud-based functions promotes not only scalability but also facilitates standardized products, sharing of best practices across projects, and accessibility to audiences beyond individual researchers.

Sample code available on github

Reference:

Registry of open data: https://registry.opendata.aws/
Sentinel dataset: https://registry.opendata.aws/sentinel-2/
Sentinel hub: https://www.sentinel-hub.com/
Sentinel hub python package: https://sentinelhub-py.readthedocs.io/en/latest/
Sentinel-2 wiki: https://en.wikipedia.org/wiki/Sentinel-2#Spectral_bands
RasterIO python package: https://rasterio.readthedocs.io/en/latest/
EarthPy python package : https://earthpy.readthedocs.io/en/latest/