2022 EY Challenge - Sentinel-2 Pre-computed Median Mosaic

To assist with the challenge, our partners at NASA have created a pre-computed median mosaic from the Sentinel-2 Level-2A data that you are free to access. This will allow you to use the Sentinel-2 dataset effectively without having to expend costly computational resources and time creating your own mosaics. The product contains only four bands: red, green, blue, and near-infrared (NIR) and was calculated over Australia, South Africa, and Costa Rica for the year of 2019. Here are some statistics about the product:

- Our regions in Australia, South Africa and Costa Rica cover 75-million square kilometers.
- There are nearly 80,000 Sentinel-2 scenes and 64-Terrabytes of data over these regions in one year.
- Our 2019 median mosaic is 434 GB of data which is 150x smaller than the baseline scene data.
- The mosaic product is stored in about 11,300 tiles covering 0.25-degree square latitude/longitude blocks.
- The mosaic took 20 hours to produce using 800 parallel virtual machines (VM). It would take one free Azure VM nearly 2 years to create the same product!

The full Sentinel-2 data is still available, so you may wish to create your own custom mosaics that cover additional bands captured by Sentinel-2. We have extensive documentation on how best to query the Sentinel-2 data and manage the computational load required to compute your own mosaics.

In [1]:
```
# Supress Warnings
import warnings
warnings.filterwarnings('ignore')

# Import common GIS tools
import numpy as np
import xarray as xr
import matplotlib.pyplot as plt
import rioxarray as rio
import rasterio.features

# Import Planetary Computer tools
import stackstac
import pystac_client
import planetary_computer
import xrspatial.multispectral as ms

# Import function to obtain the precomputed mosaics
from sentinel_mosaic import get_mosaic
```

Discover and load the data for analysis

First, we define our area of interest using latitude and longitude coordinates. Our sample region is on hte southern end of Costa Rica. The first line defines the lower-left corner of the bounding box and the second line defines the upper-right corner of the bounding box. GeoJSON format uses a specific order: (longitude, latitude), so be careful when entering the coordinates.

In [2]:
```
min_lon, min_lat = (-83.20, 8.00)  # Lower-left corner
max_lon, max_lat = (-82.80, 8.40)  # Upper-right corner
bbox = (min_lon, min_lat, max_lon, max_lat)
```

In [3]:
```
# This step finds the pixels in our bounding box and stores them in memory
# The output below will show the 0.25-degree tiles that are intersected by our bounds
median = get_mosaic(bbox)
```

```
(-83.0, 8.0, -82.75, 8.25)
(-83.0, 8.25, -82.75, 8.5)
(-83.25, 8.0, -83.0, 8.25)
(-83.25, 8.25, -83.0, 8.5)
4 scenes intersect
```

In [4]:
```
# This will report the pixel dimensions of our mosaic data.
# Recall that pixel resolution will impact the dimensions.
median.sel(band="red").shape
```

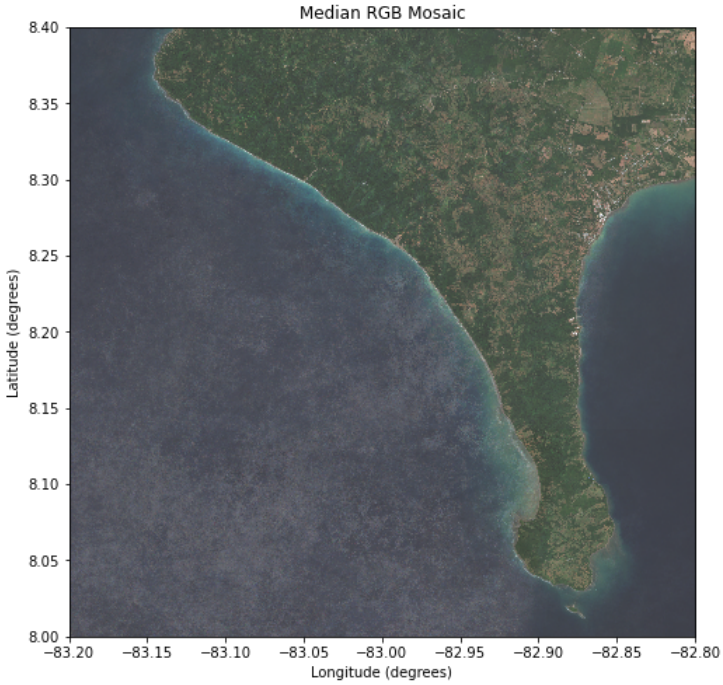
Out[4]:
```
(4000, 4000)
```

To visualize the data, we'll use xarray-spatial's `true_color` and `ndvi` functions to display the results. The RGB image is what we would see with our eye if looking from space at the land.

In [5]:
```
image = ms.true_color(median.sel(band="red"), median.sel(band="green"), median.sel(band="blue"))
```

In [6]:
```
image.plot.imshow(figsize=(8,8))
plt.title("Median RGB Mosaic")
plt.xlabel("Longitude (degrees)")
```

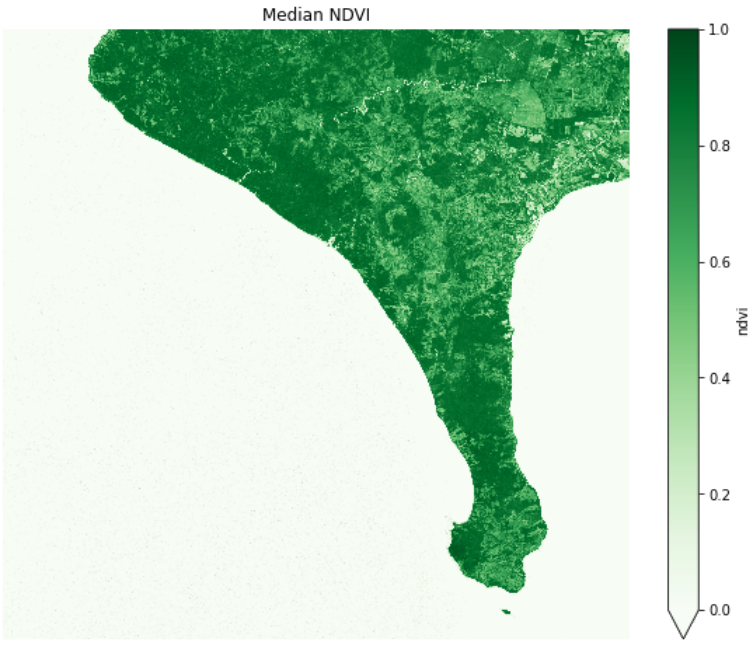
```
plt.ylabel("Latitude (degrees)")
plt.show()
```



The **Normalized Difference Vegetation Index (NDVI)** is used to measure the "greenness" of vegetation and has a range of 0.0 to 1.0. Low values (0.0 to 0.25) reflect a lack of vegetation (bare soil, urban, water), middle values (0.25 to 0.6) reflect grasslands or croplands in their growing state, and high values (0.6 to 1.0) reflect dense vegetation such as trees or croplands at their peak vegetation state. The equation uses two spectral bands where: **NDVI = (NIR-Red) / (NIR+Red).**

In [7]:
```
ndvi_median = ms.ndvi(median.sel(band="nir"), median.sel(band="red"))
```

In [8]:
```
ndvi_median.plot.imshow(cmap="Greens", vmin=0.0, vmax=1.0, figsize=(10,8))
plt.title("Median NDVI")
plt.axis('off')
plt.show()
```



How will the participants use this data?

The mosaic data contains the Lat-Lon coordinates of each pixel and the median band values (Red, Green, Blue, NIR) for each pixel. These band values can be easily used to calculate indices such as NDVI (vegetation) or NDWI (water). Since the frog species training data also uses Lat-Lon position, it is possible to find the closest Sentinel-2 mosaic pixel using code similar to what is demonstrated

below. This process can be repeated for each frog species sample in the training data. In the end, the Sentinel-2 spectral data from this mosaic can be used for modeling species distribution.

Here are some other things to consider as you apply this data to your models. The closest pixel to any frog location may not be the "best" representation of the surrounding vegetation or land conditions. For example, the frog location may have been acquired near a building or road that could skew the spectral data. So, participants may want to use methods that consider spectral data in the "proximity" to any frog location as this might better reflect the surrounding vegetation or land conditions.

Another possible issue is exceeding cloud computing memory limits. This may happen if you load large amounts of mosaic data prior to searching for corresponding frog locations. So, participants may want to calculate their desired spectral information for each frog location and only store that final result for later use in their model.

In [9]:
```
# This is an example for a specific Lon-Lat location randomly selected within our sample region.
values = median.sel(x=-83.00, y=8.00, method="nearest").values
print("These are the band values (R,G,B,NIR) for the closest pixel: ", values)
```

```
These are the band values (R,G,B,NIR) for the closest pixel:  [426. 458. 598. 363.]
```