Connecting-Jupyter-to-Google-Earth-engine-the-perform-Linear-Regression-timeseries-NDVI (/github/Amobijones/Connecting-Jupyter-to-Google-Earth-engine-the-perform-Lin
/ Jupyter GEE.ipynb (/github/Amobijones/Connecting-Jupyter-to-Google-Earth-engine-the-perform-Linear-Regression-timeseries-NDVI/tree/main/Jupyter GEE.ipynb)

Arthur: Amobichukwu Amanambu Objectives: 1. This project sought to connect Jupyter notebook with Google earth engine. 2. import shapefiles and convert to jason file then to GEE GEOMETRY 3. PLOT to screen as it were for GEE 4. imoprt NDVI Collection from 1999 to 2017 from two seperate collections Landsat 5 and Landsat 7. 5. Merge the collection then create a monthly time series from them 6. Run a Logistic regression for the enter monthly NDVI

In [1]:

```
# the regulars
import pandas as pd
import geopandas as gpd
import json
import os
import requests

# for earth engine
import ee
import geemap
from geemap import eefolium
from geemap import geojson_to_ee, ee_to_geojson
# allow images to display in the notebook
from IPython.display import Image
from ipyleaflet import GeoJSON
#import require
```

In [2]:

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

# # Initialize the library.
# ee.Initialize()

service_account = 'Your Service account.gserviceaccount.com' # REMEBER TO CHANGE THIS FOR YOUR SPECIFIC ACCOUNT.

#DOWNLAOD THE SERVICE ACCPUNT AS JASON FILE then link the path to the next line of code
credentials = ee.ServiceAccountCredentials(service_account, r"C:\Users\a.amanambu\Downloads\Servivce_account_ID.json")
ee.Initialize(credentials)

#SEE the instructions below to set up your google service account
```

Follow the steps given on this page: https://developers.google.com/earth-engine/guides/service_account (https://developers.google.com/earth-engine/guides/service_account)

Most importantly "Register the service account to use Earth Engine" so what you have to do is register "cloud-service...iam.gserviceaccount.com" (i.e., your service _account) via this link: https://signup.earthengine.google.com/#!/service_accounts (https://signup.earthengine.google.com/#!/service_accounts) by entering the service_account email in your case "cloud-service...iam.gserviceaccount.com" in the Email field

and that's it, you are good to go without having the need to use ee.Authenticate()

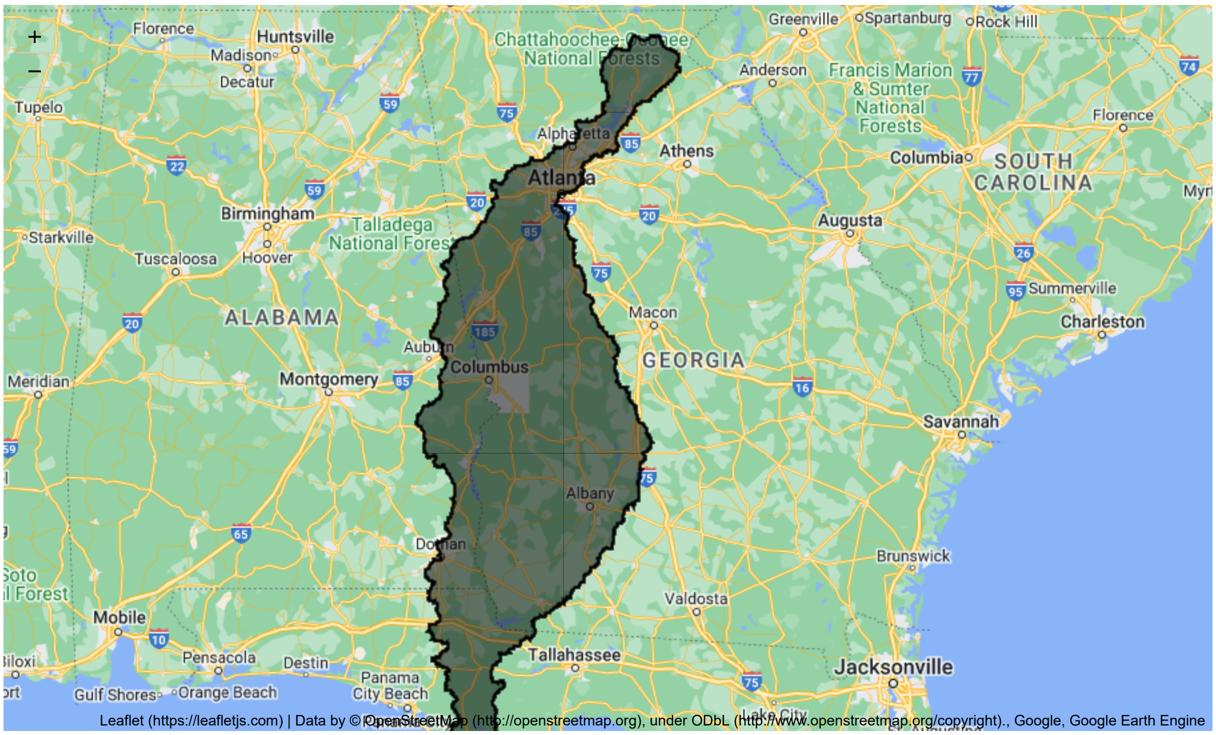
In [ ]:

```
```

In [9]:

```
# iMPORT shape  file which is the boundary
region = gpd.read_file(r"C:\Users\a.amanambu\Desktop\Chapter 3\Apalachicola_boundary\boundary_AC.shp")

#convert to jason file
js = json.loads(region.to_json())

#convert to GEE geometry
geometry = ee.Geometry(ee.FeatureCollection(js).geometry())

#Map GEE interface on Jupyter using eefolium from geemap
Map = eefolium.Map()

#add the geometry layer
Map.addLayer(geometry)

# centralise the map to see the area of interest
Map.setCenter(-81.983246, 31.725767, 6.5) #centralise the map

#plot the map
Map
```

Out[9]:



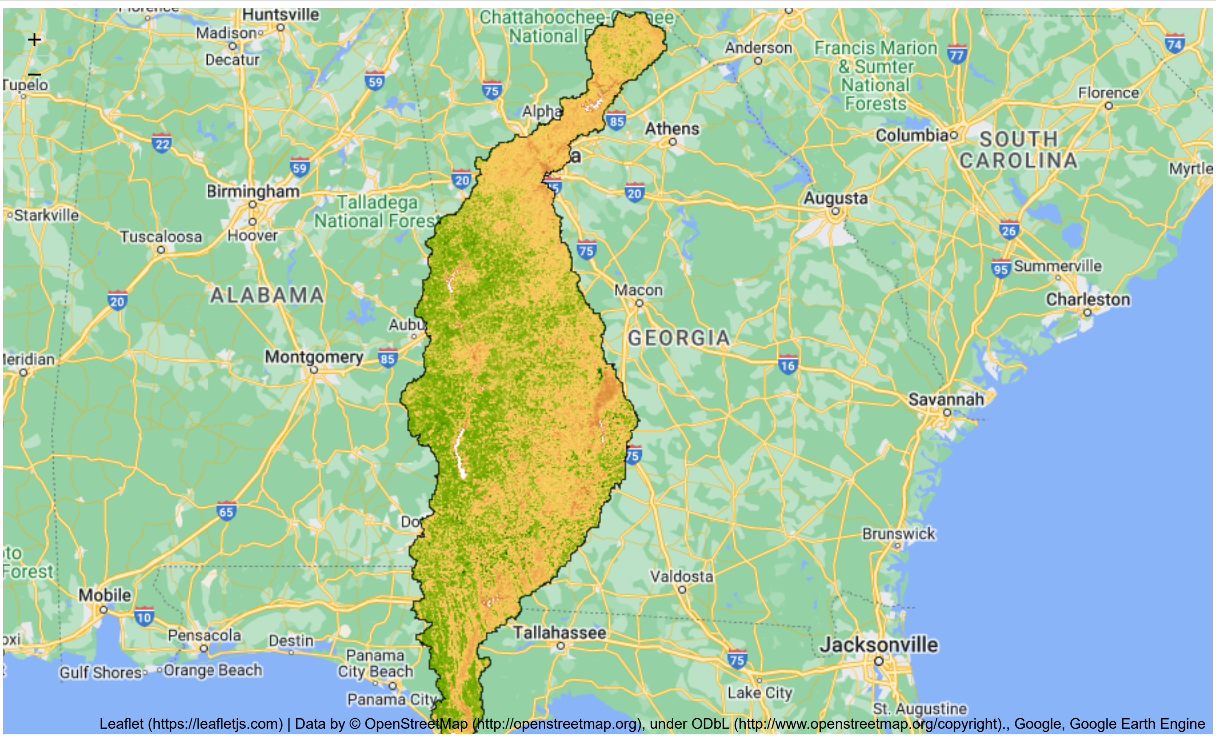
In [4]:

```
#geemap.ee_search()
```

In [5]:

```
# iMPORT shape  file which is the boundary

region = gpd.read_file(r"C:\Users\a.amanambu\Desktop\Chapter 3\Apalachicola_boundary\boundary_AC.shp")
js = json.loads(region.to_json())
geometry = ee.Geometry(ee.FeatureCollection(js).geometry())
Map = eefolium.Map()
Map.addLayer(geometry)
Map

dataset5 = ee.ImageCollection("LANDSAT/LT05/C01/T1_32DAY_EVI")\
                   .filterDate('1999-01-01', '2012-12-31').select('EVI').filterBounds(geometry);


dataset7 = ee.ImageCollection("LANDSAT/LE07/C01/T1_32DAY_EVI")\
                  .filterDate('2013-01-01', '2017-12-31').select('EVI').filterBounds(geometry);

# merge collections
dataset_merged = dataset5.merge(dataset7)

#clip image with lambda function

collection = dataset_merged.map(lambda image: image.clip(geometry))

# create a color code for plotting the NDVI
collectionVis = {
  'min': 0.0,
  'max': 1.0,
  'palette': [
    'FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163', '99B718', '74A901',
    '66A000', '529400', '3E8601', '207401', '056201', '004C00', '023B01',
    '012E01', '011D01', '011301'
  ],
};

#centralise the map to study area
Map.setCenter(-81.983246, 31.725767, 6.5)

#add the NDVI collection to the color code
Map.addLayer(collection, collectionVis, 'EVI');
Map
```

Out[5]:

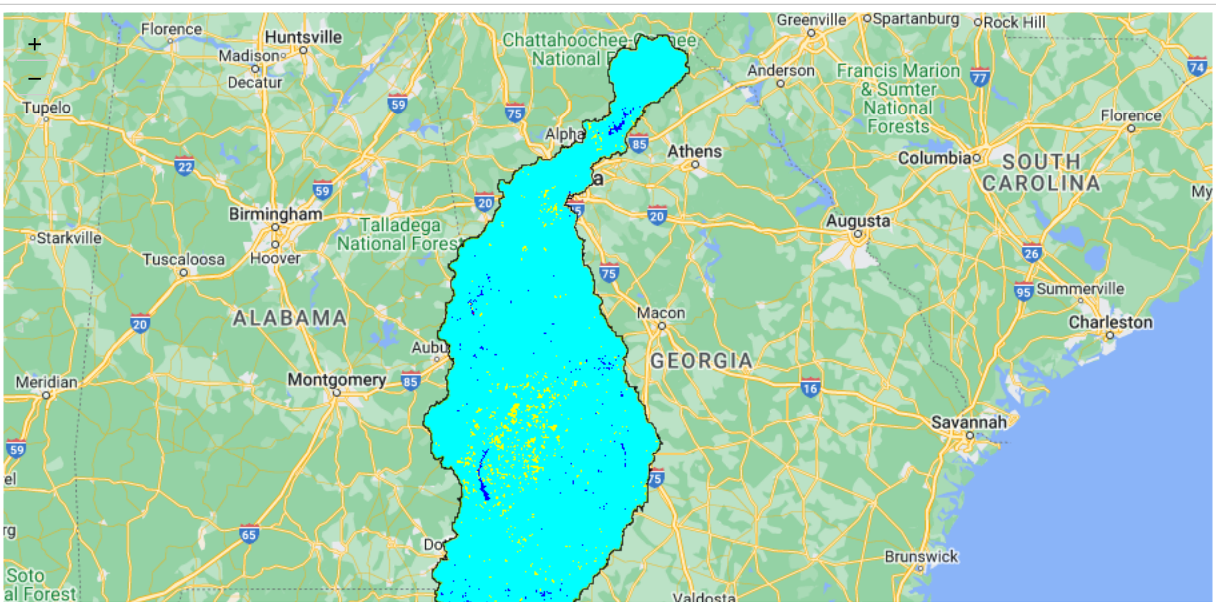


Add a chuck of code to the initial code that calculates Linear regression. note that if you do this seperately it will not work expct build within a cell block

In [6]:

```
# iMPORT shape  file which is the boundary

region = gpd.read_file(r"C:\Users\a.amanambu\Desktop\Chapter 3\Apalachicola_boundary\boundary_AC.shp")
js = json.loads(region.to_json())
geometry = ee.Geometry(ee.FeatureCollection(js).geometry())
Map = eefolium.Map()
Map.addLayer(geometry)
Map

dataset5 = ee.ImageCollection("LANDSAT/LT05/C01/T1_32DAY_EVI")\
                   .filterDate('1999-01-01', '2012-12-31').select('EVI').filterBounds(geometry);


dataset7 = ee.ImageCollection("LANDSAT/LE07/C01/T1_32DAY_EVI")\
                  .filterDate('2013-01-01', '2017-12-31').select('EVI').filterBounds(geometry);

# merge collections
dataset_merged = dataset5.merge(dataset7)

#clip image with lambda function

collection = dataset_merged.map(lambda image: image.clip(geometry))

# create a color code for plotting the NDVI
collectionVis = {
  'min': 0.0,
  'max': 1.0,
  'palette': [
    'FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163', '99B718', '74A901',
    '66A000', '529400', '3E8601', '207401', '056201', '004C00', '023B01',
    '012E01', '011D01', '011301'
  ],
};

#centralise the map to study area
Map.setCenter(-81.983246, 31.725767, 6.5)

#add the NDVI collection to the color code
Map.addLayer(collection, collectionVis, 'EVI');

# This function adds a time band to the image
# Scale milliseconds by a large constant to avoid very small slopes
# in the linear regression output.

createTimeBand = lambda img: img.addBands(img.metadata('system:time_start').divide(1e18));

#create a time band
collection2 = collection.map(createTimeBand);

trend = collection2.select(['system:time_start', 'EVI']).reduce(ee.Reducer.linearFit());

Map.addLayer(trend,
  {'min': 0, 'max': [-0.9, 8e-5, 1], 'bands': ['scale', 'offset', 'scale']}, 'fit');


Map

#Areas projected to decrease in decrease in vegetation are sjhown in yellow.
#This is true as most of these areas are heavily urbanised.
```

Out[6]:




Mobile
Biloxi
Pensacola
Destin
Panama City Beach
Gulf Shores
Orange Beach
Tallahassee
Jacksonville
Lake City
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright)., Google, Google Earth Engine