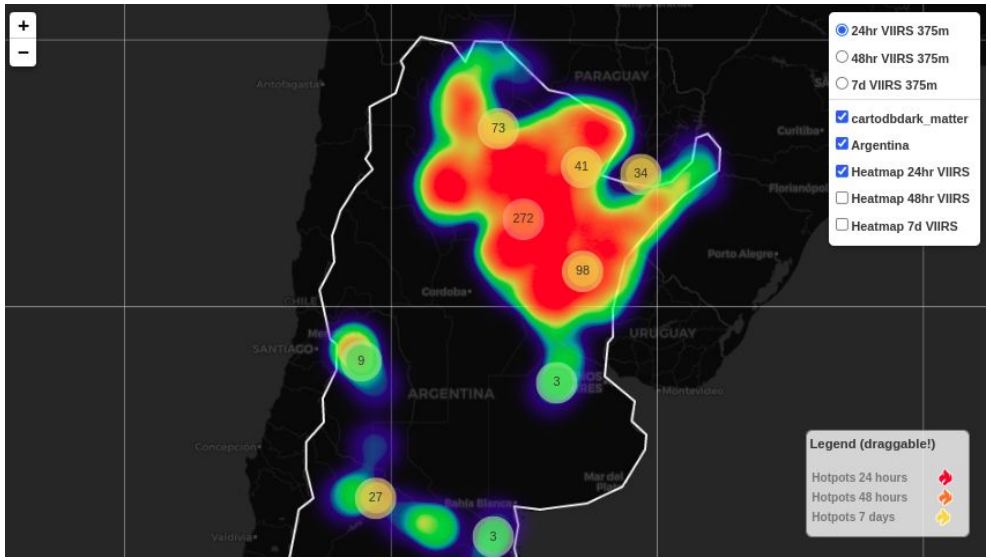
🔥 Mapping VIIRS Active Fires in South America



Abraham Coiman

Agenda

1. Introduction (VIIRS Active fire products)
2. Getting Active fire data
3. Processing Active fire data
4. Visualizing Active fire data
5. Conclusions

Introduction

What is the VIIRS 375 m Active Fire Product?

- Visible Infrared Imaging Radiometer Suite (VIIRS).
- VIIRS sensor aboard the joint NASA/NOAA Suomi National Polar-orbiting Partnership (Suomi NPP) and NOAA-20 satellites.

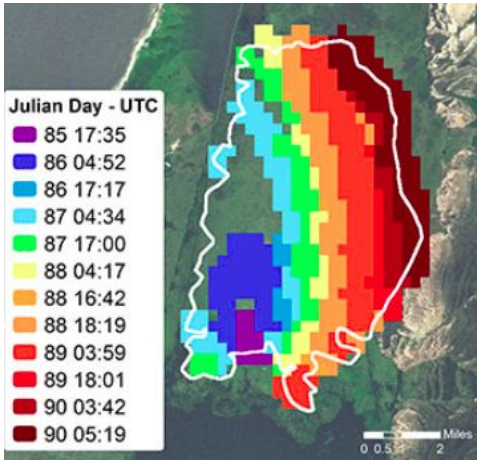
- 375 m thermal anomalies / active fire product

Introduction

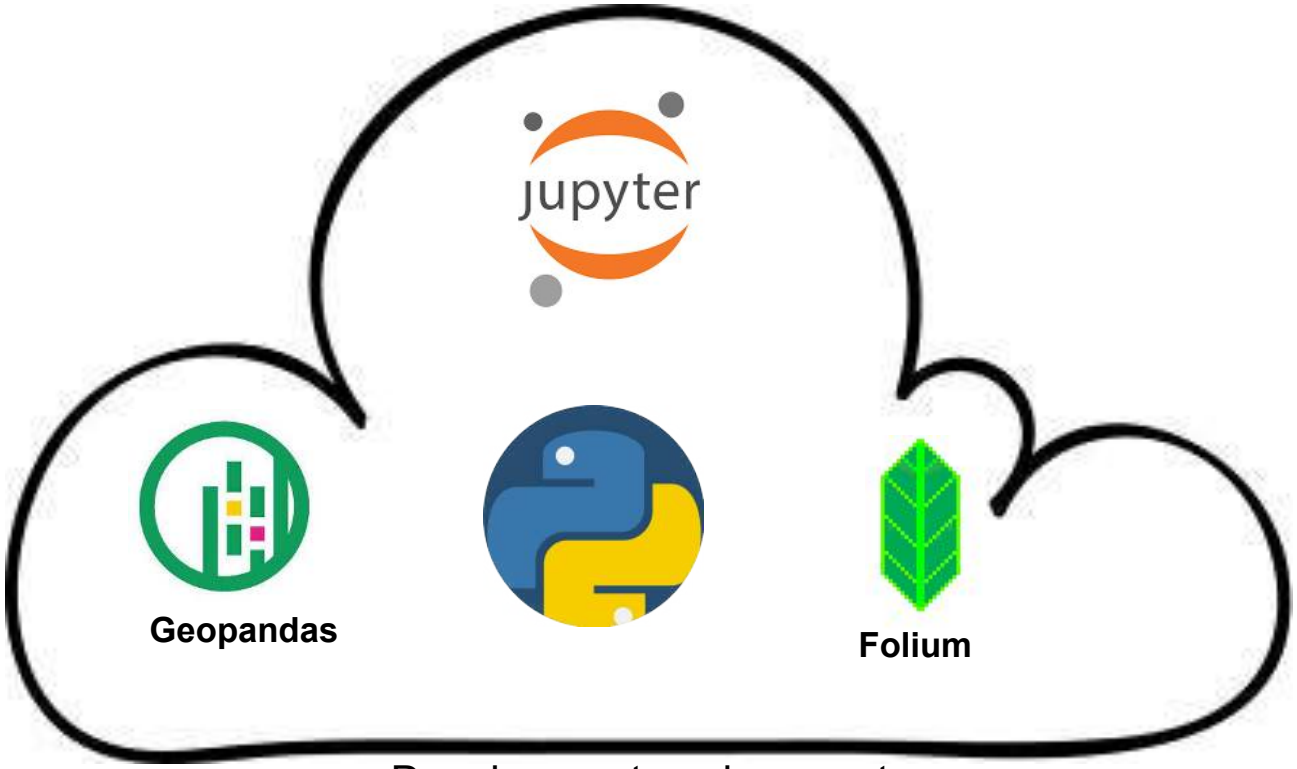
What is the VIIRS 375 m Active Fire Product?

- VIIRS-AF points represent possible active fires called **Hotspots**.

- VIIRS-AF data are very useful to leverage wildfire management and research activities



Getting data



Getting data

Select a Country

```
w = widgets.Dropdown(
    options= countries,
    value='Venezuela'
)
print('Select a country')
display(w)
```

```
Select a country
```

Venezuela

Create folders, download and unzip the VIIRS-AF data

Function to create folders (date and time)

List of dataset links*

* https://firms.modaps.eosdis.nasa.gov/active_fire/

Getting data

Create folders, download and unzip the VIIRS-AF data

```
# list to store folder names
folder_names = []
localtime = time.localtime(time.time())

# save the data into the right spot
for i, name_Link in enumerate(zip(names, links)):
    # download the file content in binary format
    r = requests.get(name_Link[1])

    # open method to open a file on your system and write the contents
    with open(name_Link[0], "wb") as code:
        code.write(r.content)

    # unzip data
    zip_ref = zipfile.ZipFile(name_Link[0], 'r')
    name = make_folder_name(localtime)+'_'+name_Link[0]
    folder_names.append(name)
    zip_ref.extractall(name)
    zip_ref.close()
    # remove zipped file after unzipping
    os.remove(name_Link[0])
```

Getting data

Create folders, download and unzip the VIIRS-AF data

```
# list to store folder names
folder_names = []
localtime = time.localtime(time.time())

# save the data into the right spot
for i, name_Link in enumerate(zip(names, links)):
    # download the file content in binary format
    r = requests.get(name_Link[1])

    # open method to open a file on your system and write the contents
    with open(name_Link[0], "wb") as code:
        code.write(r.content)

    # unzip data
    zip_ref = zipfile.ZipFile(name_Link[0], 'r')
    name = make_folder_name(localtime)+'_'+name_Link[0]
    folder_names.append(name)
    zip_ref.extractall(name)
    zip_ref.close()
    # remove zipped file after unzipping
    os.remove(name_Link[0])
```

Getting data

Create folders, download and unzip the VIIRS-AF data

```
# list to store folder names
folder_names = []
localtime = time.localtime(time.time())

# save the data into the right spot
for i, name_Link in enumerate(zip(names, links)):
    # download the file content in binary format
    r = requests.get(name_Link[1])

    # open method to open a file on your system and write the contents
    with open(name_Link[0], "wb") as code:
        code.write(r.content)

    # unzip data
    zip_ref = zipfile.ZipFile(name_Link[0], 'r')
    name = make_folder_name(localtime)+'_'+name_Link[0]
    folder_names.append(name)
    zip_ref.extractall(name)
    zip_ref.close()
    # remove zipped file after unzipping
    os.remove(name_Link[0])
```

Processing data

Clip each shapefile to our study area (e.g. country boundaries).

```
ptclip = gpd.clip(ptall, study_area)
```

Get latitude and longitude from point GeoDataFrame

```
lats = pts['LATITUDE'].tolist()

lons = pts['LONGITUDE'].tolist()
```

Visualizing data

Customize Folium Popup

Calculate the location parameter

```
centx= study_area['geometry'].centroid.x.tolist()
centy= study_area['geometry'].centroid.y.tolist()
location = [*centy, *centx]
```

Calculate zoom_start parameter

Visualizing data

Customize Folium Popup

```
# select attributes
bt4 = df['BRIGHT_TI4'].tolist()
acd = df['ACQ_DATE'].tolist()
act = df['ACQ_TIME'].tolist()
con = df['CONFIDENCE'].tolist()
bt5 = df['BRIGHT_TI5'].tolist()
frp = df['FRP'].tolist()
dynt = df['DAYNIGHT']

# convert attributes into list
poplist = list(zip(bt4, acd, act, con, bt5, frp, dynt))

# use HTML to customize the popup
popup = ['<div style="width: 250px;">\
            <strong>Brightness temperature I-4 (Kelvin): </strong>{}\
            <br>\
            <strong>Acquisition Date: </strong>{}\
            <br>\
            <strong>Acquisition Time: </strong>{}\
            <br>\
            <strong>Confidence: </strong>{}\
            <br>\
            <strong>Brightness temperature I-5 (Kelvin): </strong>{}\
            <br>\
            <strong>Fire Radiative Power (megawatts): </strong>{}\
            <br>\
            <strong>Day(D) or Night(N): </strong>{}\
        </div>'.format(b4, ad, at, co, b5, fr, dn) for (b4, ad, at, co, b5, fr, dn) in poplist]
```

Visualizing data

Customize Folium Popup

```
# select attributes
b14 = df['BRIGHT_TI4'].tolist()
acd = df['ACQ_DATE'].tolist()
act = df['ACQ_TIME'].tolist()
con = df['CONFIDENCE'].tolist()
b15 = df['BRIGHT_TI5'].tolist()
frp = df['FRP'].tolist()
dynt = df['DAYNIGHT']

# convert attributes into list
poplist = list(zip(b14, acd, act, con, b15, frp, dynt))

# use HTML to customize the popup
popup = ['<div style="width: 250px;">\
            <strong>Brightness temperature I-4 (Kelvin): </strong>{}\
            <br>\
            <strong>Acquisition Date: </strong>{}\
            <br>\
            <strong>Acquisition Time: </strong>{}\
            <br>\
            <strong>Confidence: </strong>{}\
            <br>\
            <strong>Brightness temperature I-5 (Kelvin): </strong>{}\
            <br>\
            <strong>Fire Radiative Power (megawatts): </strong>{}\
            <br>\
            <strong>Day(D) or Night(N): </strong>{}\
        </div>'.format(b4, ad, at, co, b5, fr, dn) for (b4, ad, at, co, b5, fr, dn) in poplist]
```

Visualizing data

Calculate zoom_start parameter

```python
# modified from: https://medium.com/@busybus/rendered-maps-with-python-ffba4b34101c
# script to calculate the zoom start of a Folium Map object
# License: CC0 -- no rights reserved

from math import pi, log, tan, exp, atan, log2, floor

# convert geographical coordinates to pixels
# https://en.wikipedia.org/wiki/Web_Mercator_projection
# note on google API: the world map is obtained with lat=lon=0, w=h=256, zoom=0, therefore:
ZOOM0_SIZE = 512

# Geo-coordinate in degrees => Pixel coordinate
def g2p(lat, lon, zoom):
    return (
        # x
        ZOOM0_SIZE * (2 ** zoom) * (1 + lon / 180) / 2,
        # y
        ZOOM0_SIZE / (2 * pi) * (2 ** zoom) * (pi - log(tan(pi / 4 * (1 + lat / 90))))
    )


# Pixel coordinate => geo-coordinate in degrees
def p2g(x, y, zoom):
    return (
        # lat
        (atan(exp(pi - y / ZOOM0_SIZE * (2 * pi) / (2 ** zoom))) / pi * 4 - 1) * 90,
        # lon
        (x / ZOOM0_SIZE * 2 / (2 ** zoom) - 1) * 180,
    )
```

Visualizing data

Calculate zoom_start parameter

```python
# modified from: https://medium.com/@busybus/rendered-maps-with-python-ffba4b34101c
# script to calculate the zoom start of a Folium Map object
# License: CC0 -- no rights reserved

from math import pi, log, tan, exp, atan, log2, floor

# convert geographical coordinates to pixels
# https://en.wikipedia.org/wiki/Web_Mercator_projection
# note on google API: the world map is obtained with lat=lon=0, w=h=256, zoom=0, therefore:
ZOOM0_SIZE = 512

# Geo-coordinate in degrees => Pixel coordinate
def g2p(lat, lon, zoom):
    return (
        # x
        ZOOM0_SIZE * (2 ** zoom) * (1 + lon / 180) / 2,
        # y
        ZOOM0_SIZE / (2 * pi) * (2 ** zoom) * (pi - log(tan(pi / 4 * (1 + lat / 90))))
    )


# Pixel coordinate => geo-coordinate in degrees
def p2g(x, y, zoom):
    return (
        # lat
        (atan(exp(pi - y / ZOOM0_SIZE * (2 * pi) / (2 ** zoom))) / pi * 4 - 1) * 90,
        # lon
        (x / ZOOM0_SIZE * 2 / (2 ** zoom) - 1) * 180,
    )
```

Visualizing data

Calculate zoom_start parameter

```
def get_zoom_start(bbox):

    # the region of interest in geo-coordinates in degrees
    (left, bottom, right, top) = bbox

    # rendered image map size in pixels
    (w, h) = (1024, 1024)

    # The center point of the region of interest
    (lat, lon) = ((top + bottom) / 2, (left + right) / 2)

    # look for appropriate zoom level to cover the region of interest
    for zoom in range(16, 0, -1):
        # Center point in pixel coordinates at this zoom level
        (x0, y0) = g2p(lat, lon, zoom)

        # the "container" geo-region that the downloaded map would cover
        (TOP, LEFT) = p2g(x0 - w / 2, y0 - h / 2, zoom)
        (BOTTOM, RIGHT) = p2g(x0 + w / 2, y0 + h / 2, zoom)

        # Would the map cover the region of interest?
        if (LEFT <= left < right <= RIGHT):
            if (BOTTOM <= bottom < top <= TOP):
                break
    return zoom
```

Visualizing data

Calculate zoom_start parameter

```python
def get_zoom_start(bbox):

    # the region of interest in geo-coordinates in degrees
    (left, bottom, right, top) = bbox

    # rendered image map size in pixels
    (w, h) = (1024, 1024)

    # The center point of the region of interest
    (lat, lon) = ((top + bottom) / 2, (left + right) / 2)

    # look for appropriate zoom level to cover the region of interest
    for zoom in range(16, 0, -1):
        # Center point in pixel coordinates at this zoom level
        (x0, y0) = g2p(lat, lon, zoom)

        # the "container" geo-region that the downloaded map would cover
        (TOP, LEFT) = p2g(x0 - w / 2, y0 - h / 2, zoom)
        (BOTTOM, RIGHT) = p2g(x0 + w / 2, y0 + h / 2, zoom)

        # Would the map cover the region of interest?
        if (LEFT <= left < right <= RIGHT):
            if (BOTTOM <= bottom < top <= TOP):
                break
    return zoom
```

Visualizing data

create folium.Map object

add heatmaps

add Marker Cluster

```
map = folium.Map(location, zoom_start=get_zoom_start(bbox), \
                 tiles=False,control_scale=True, prefer_canvas=True)
```

```
pt24h_hm = plugins.HeatMap(locations24h,
                           name='Heatmap 24hr VIIRS',
                           overlay=True,
                           control=True,
                           show=False)
```

```
cluster24h = MarkerCluster(name='24hr VIIRS 375m', overlay=False).add_to(map)
cluster48h = MarkerCluster(name='48hr VIIRS 375m', overlay=False).add_to(map)
cluster7d = MarkerCluster(name='7d VIIRS 375m', overlay=False).add_to(map)
```

```
for loc24h, pop24h in zip(locations24h, get_popup(pt24h)):
    folium.Marker(loc24h,
                  icon=folium.features.CustomIcon(pointIcon24h, icon_size=(40, 40)),
                  popup=pop24h,
                 ).add_to(cluster24h)
```

Visualizing data

add map title

```
text = 'VIIRS Active Fires in '
date = str(localtime.tm_year)+'/'+str(localtime.tm_mon)+'/'+str(localtime.tm_mday)
title_html = '''
             <h3 align="center" style="font-size:16px"><b>{}</b></h3>
             '''.format(text + w.value + ' ' + 'on' + ' ' + date)
map.get_root().html.add_child(folium.Element(title_html));
```

Custom module to create a Folium map legend using branca.element class


thank you,
questions?