Published in Analytics Vidhya

Abhishek Mamdapure Follow

Jun 11, 2020 · 3 min read · Listen

Save

How to read and visualize netCDF(.nc) geospatial files using python?

If you can visualize the data, you can analyze the data.

What are netCDF files?
netCDF (network Common Data Form) is a format of file used for storing multidimensional scientific data (in the form of variables) such as temperature, humidity, pressure, wind speed, and direction. Each of these variables can be displayed through a dimension (such as time). It is commonly used in climatology, meteorology, oceanography where the data is related to weather forecasting, climatic change, and even used for some of the GIS applications.

What are the libraries required to access and visualize netCDF files?

```
from netCDF4 import Dataset
import numpy as np
from mpl_toolkits.basemap import Basemap
```

libraries.py hosted with ❤ by GitHub view raw

To install “Basemap”, please do follow these specific steps:

1. First of all, you must have **anaconda** installed on your computer.

2. Now go to this great link: http://www.lfd.uci.edu/~gohlke/pythonlibs/
   Search for “basemap” and download the file named

“basemap-1.2.1-cp37-cp37m-win_amd64.whl”.
I have downloaded this file as my python version is **3.7.4** and my operating system is **64** bit. Please choose the file as per your system configuration.

3. Move the downloaded ‘**.whl**’ file to directory “**C:\Python27**”. (Again, this is my path where my python is downloaded, do choose your path where the python setup is.)
4. Open anaconda prompt in “Run as administrator” mode and head to the directory where the ‘.whl’ files is been moved.
5. Use this command:

```
pip install basemap-1.2.1-cp37-cp37m-win_amd64.whl
```

After this, you will be ready by importing all the required libraries.

How to read netCDF files in python?

Please click here to download the data for this article. Once the data is downloaded, let’s get started.

```
file = '1966.nc' # mention the path to the downloaded file
data = Dataset(file, mode='r') # read the data
print(type(data)) # print the type of the data
print(data.variables.keys()) # print the variables in the data
```

read.py hosted with ❤ by GitHub view raw

Output:

```
<class 'netCDF4._netCDF4.Dataset'>
dict_keys(['lon', 'lat', 'time', 'tave', 'rstn'])
```

We can see that the type of dataset is netCDF, and there are in all 5 variables (‘lat’, ‘lon’, ‘time’, ‘air’, ‘time_bnds’)in the data.

Now, as we have read the data from any of the variables contained in 'data', for this article we will read the co-ordinates variables along with the 'air' variable. The below code puts the variable into NumPy arrays.

```python
lats = data.variables['lat'][:]
longs = data.variables['lon'][:]
time = data.variables['time'][:]

tave = data.variables['tave'][:]
```

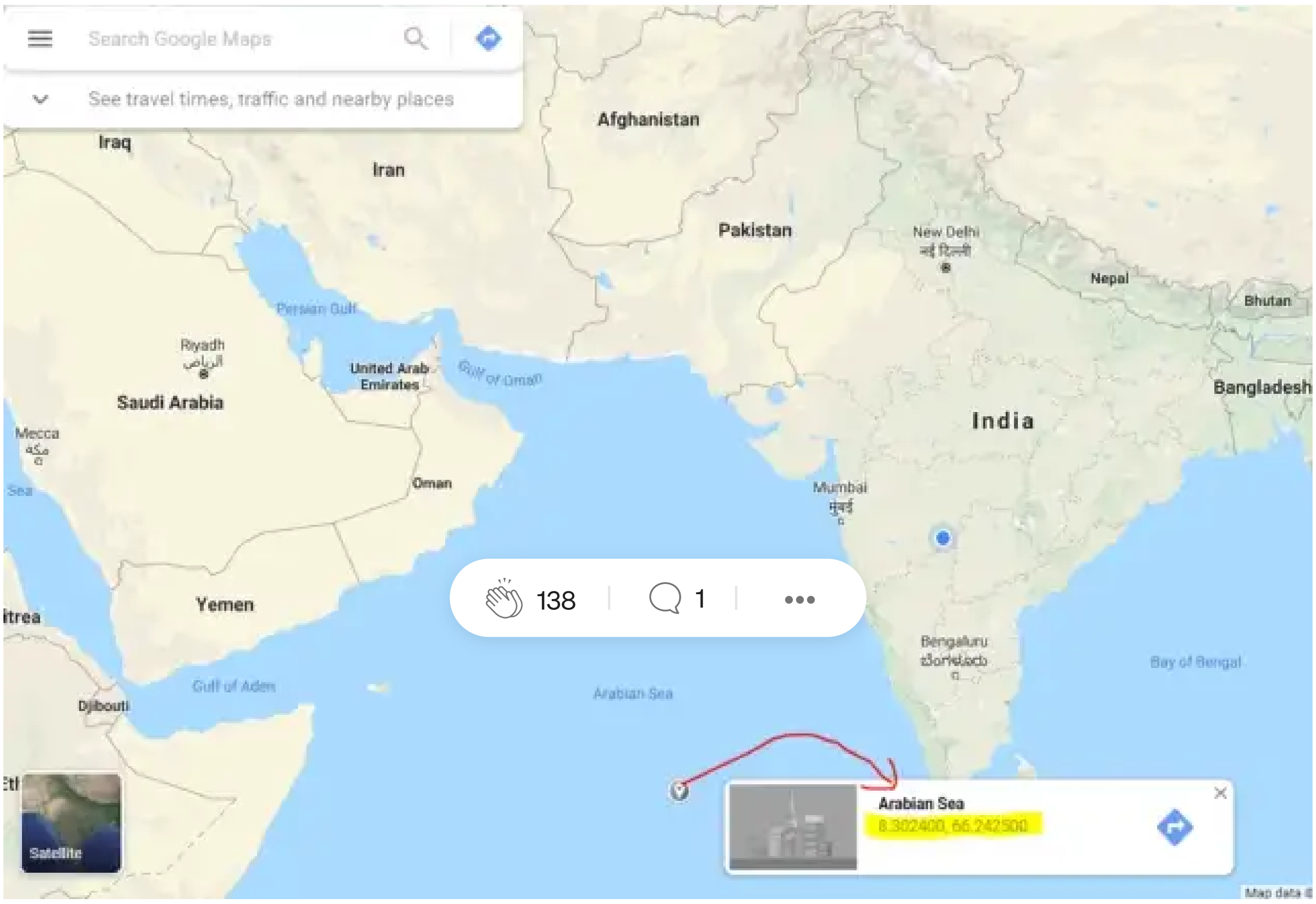
read_variables.py hosted with ❤ by GitHub view raw

positions at lower and upper levels. In the below code, the latitude and longitude points are taken from a google map referencing the boundary of India. You can also check the image below on how to take the points from the google map, highlighted values are of the format(lat, long). Here, basically a Basemap instance is going to be generated which specifies our desired map and projection settings.

```python
mp = Basemap(projection='merc',
             llcrnrlon=68.42,   # lower longitude
             llcrnrlat=7.45,    # lower latitude
             urcrnrlon=99.98,   # uppper longitude
             urcrnrlat=37.78,   # uppper latitude
            resolution = 'i')
```

basemap_instance.py hosted with ❤ by GitHub view raw

You can also check the image below on how to take the points from the google map, highlighted values are of the format(lat, long).



snap from google maps showing coordinates

Once, the Basemap instance is generated then we will proceed for the further part where we plot the data using matplotlib.

```
lon, lat = np.meshgrid(long,lats)  #this converts coordinates into 2D arrray
x,y = mp(lon,lat) #mapping them together
plt.figure(figsize=(6,8)) #figure size
c_scheme = mp.pcolor(x,y,np.squeeze(tave[0,:,:]),cmap = 'jet') # [0,:,:] is for the first day c

# consider this as the outline for the map that is to be created
mp.drawcoastlines()
mp.drawstates()
mp.drawcountries()

cbar = mp.colorbar(c_scheme,location='right',pad = '10%') # map information
plt.title('Average Temperature for: Day 1 of year 2019')
plt.show()

plt.savefig('tave.jpg',dpi=300) #saves the image generated
```

plotting.py hosted with ❤ by GitHub   view raw

The data contains the points for every day for the whole year, and we are now able to see the visualization for only one day, let's try to simulate the process so that all year data can be able to visualize.
Before running the below code, do mention the number of images(days) that should be created, I have mentioned it for the whole year(365)

```
# this will create 365 images in the path given
# please set the path before executing
lon, lat = np.meshgrid(long,lats)
x,y = mp(lon,lat)
plt.figure(figsize=(6,8))

# loop for all the days
days = np.arange(0,365)  # for considering all days of the year

for i in days:
    c_scheme = mp.pcolor(x,y,np.squeeze(tave[i,:,:]),cmap = 'jet')
    mp.drawcoastlines()
    mp.drawstates()
    mp.drawcountries()

    cbar = mp.colorbar(c_scheme,location='right',pad = '10%')
    day = i+1

    plt.title('Average Temperature for the Day ' + str(day) +  ' of year 2019')
    plt.clim(-20,20)

    plt.savefig(r'path\where\to\save\images'+ '\\' + str(day)+'.jpg')
    plt.clf()
```

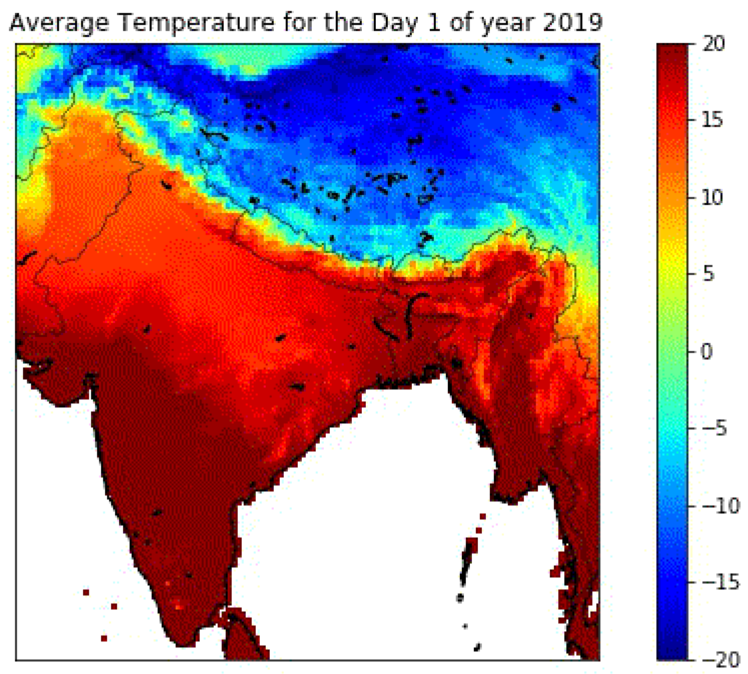
looping_plots.py hosted with ❤ by GitHub view raw

Once the images are created, looping those images to create a '.gif' for better visualization. Here, making the use of 'PIL' library to read the images and make a gif using all of them.

```python
import PIL

image_frames = [] # creating a empty list to be appended later on
days = np.arange(1,365)

for k in days:
    new_fram = PIL.Image.open(r'path\where\images\are\downloaded' + '\\' + str(k) + '.jpg')
    image_frames.append(new_fram)

image_frames[0].save('tave_timelapse.gif',format='GIF',
                    append_images = image_frames[1: ],
                    save_all = True, duration = 100,
                    loop = 0)
```

creating_gif.py hosted with ❤ by GitHub view raw

If all the steps are performed in a proper manner, the ‘tave_timelapse.gif’ generated will look something like this. Have a look. Due to memory constraints for sharing more than 25MB image size on Medium, I have made a simulation for the first 50 days only.



Observation:

We can see how temperature is changing in the upper hemisphere of the map. This type of visualization can be helpful in finding the change of particular factors such as wind, rains over the period of time.

```
References:
https://www.youtube.com/watch?v=r5m_aU5V6oY
http://schubert.atmos.colostate.edu/~cslocum/netcdf_example.html
```

Python Visualization Basemap Netcdf