TROPOMI – downloading and analysis

RD[1,2] and KPM[1,2]

[1]National Institute of Environmental Health Sciences, Division of Translational Toxicology
[2]National Institute of Environmental Health Sciences, Biostatistics and Computational Biology Branch

August 1, 2023

Abstract

Here I'll put the download descriptions and basic analysis part. I may include some initial spatial modeling results.

Keywords: Sentinel

1 Download using `Python` APIU

Currently, I am using the `python` API `sentinelsat` to download the Sentinel-5p data.

Note 1.1. If we end up using this, we have to cite "NOAA/NESDIS/STAR Aerosols and Atmospheric Composition Science Team" as suggested here.

Note 1.2. Some files are failing to download. Need to list them and re-download. Not done in the jupyter notebook so far.

Ignore the next 4 pages. They are the rendered version of the jupyter notebook that is downloading the data. Rather, see the `Github` version [Not linked yet].

Go to the next Section 2 for alternate download. Go to Section 3 for analyzed data.

Detailed instruction taken from this link

Load libraries. SentinelAPI to access s5phub and ipywidgets to create the interactive cell. May need to run ``jupyter nbextension enable --py widgetsnbextension'' from the console to activate widgets.

In [8]:
```
# Import Python packages

# Module to connect to the Copernicus Open Access Hubs
from sentinelsat import SentinelAPI

# Module for manipulating dates and times
import datetime

# Module to set filesystem paths appropriate for user's operating system
from pathlib import Path

# Modules to create interactive menus in Jupyter Notebook
from IPython.display import display
import ipywidgets as widgets
```

Do not run it twice. Run once and then select drop-down menus after that.

In [11]:
```
# Enter product, data latency, observation start/end dates and domain boundaries for fil
# Selections are made using interactive Jupyter Notebook widgets
# Run this block *once* to generate menus
# When main function is run, it reads ".value" of each menu selection
# Do NOT re-run block if you change menu selections (re-running block resets menus to de

# Formatting settings for drop-down menus
style = {'description_width':'140px'}
layout = widgets.Layout(width='300px')

# Create drop-down menus using widgets
product = widgets.Dropdown(options=[('Aerosol Index', 'AI'), ('Aerosol Layer Height', 'A
latency = widgets.Dropdown(options=[('Near real time'), ('Offline'), ('Reprocessing') ],
start_year = widgets.Dropdown(options=[('2018'), ('2019'), ('2020'), ('2021'), ('2022'),
start_month = widgets.Dropdown(options=[('Jan', '01'), ('Feb', '02'), ('Mar', '03'), ('A
start_day = widgets.Dropdown(options=[('01'), ('02'), ('03'), ('04'), ('05'), ('06'), ('
end_year = widgets.Dropdown(options=[('2018'), ('2019'), ('2020'), ('2021'), ('2022'), (
end_month = widgets.Dropdown(options=[('Jan', '01'), ('Feb', '02'), ('Mar', '03'), ('Apr
end_day = widgets.Dropdown(options=[('01'), ('02'), ('03'), ('04'), ('05'), ('06'), ('07

# Caption for map domain boundaries
domain_caption = widgets.Label(value='ENTER LATITUDE/LONGITUDE BOUNDARIES FOR SEARCH ARE

# Format observation start/end dates menus to display side-by-side
start_date = widgets.HBox([start_year, start_month, start_day])
end_date = widgets.HBox([end_year, end_month, end_day])

# Create numerical (float) text entry widgets for map boundary corners
west_lon_float = widgets.BoundedFloatText(description='Western-most Longitude:', value=0
east_lon_float = widgets.BoundedFloatText(description='Eastern-most Longitude:', value=0
lon_label = widgets.Label(value='(use negative values to indicate °W, e.g., 100 °W = -10
lon_box = widgets.HBox([west_lon_float, east_lon_float, lon_label])
north_lat_float = widgets.BoundedFloatText(description='Northern-most Latitude:', value=
south_lat_float = widgets.BoundedFloatText(description='Southern-most Latitude:', value=
```

```
lat_label = widgets.Label(value='(use negative values to indicate °S, e.g., 30 °S = -30)
north_lat_box = widgets.HBox([north_lat_float, lat_label])
south_lat_box = widgets.HBox([south_lat_float, lat_label])

# Display drop-down menus
print('If you change menu selections (e.g., to run another search), do NOT re-run this b
display(product, latency)
display(start_date, end_date)
display(domain_caption, north_lat_box, lon_box, south_lat_box)
```

```
If you change menu selections (e.g., to run another search), do NOT re-run this block!
Re-running will re-set all menus to their defaults!
Dropdown(description='Product:', layout=Layout(width='300px'), options=(('Aerosol Inde
x', 'AI'), ('Aerosol Lay…
Dropdown(description='Data Latency:', layout=Layout(width='300px'), options=('Near real
time', 'Offline', 'Rep…
HBox(children=(Dropdown(description='Start Year:', layout=Layout(width='300px'), options
=('2018', '2019', '202…
HBox(children=(Dropdown(description='End Year:', layout=Layout(width='300px'), options=
('2018', '2019', '2020'…
Label(value='ENTER LATITUDE/LONGITUDE BOUNDARIES FOR SEARCH AREA (use up/down arrows or
type in value)', layou…
HBox(children=(BoundedFloatText(value=0.0, description='Northern-most Latitude:', layout
=Layout(height='30px',…
HBox(children=(BoundedFloatText(value=0.0, description='Western-most Longitude:', layout
=Layout(height='30px',…
HBox(children=(BoundedFloatText(value=0.0, description='Southern-most Latitude:', layout
=Layout(height='30px',…
```

Few functions. No need to change

In [12]:
```
# Convert user-entered date format to that used by Sentinel API
# "year", "month", "day": parameter variables from widget menu, set in main function

def convert_date_sentinel_api_format(year, month, day):

    # Add dashes b/w year/month and month/day
    formatted_date = year + '-' + month + '-' + day

    return formatted_date
```

In [13]:
```
# Get product abbrevation used in TROPOMI file name
# "product": parameter variable from widget menu, set in main function

def get_tropomi_product_abbreviation(product):
    if product == 'CO':
        product_abbreviation = 'L2__CO____'
    elif product == 'NO2':
        product_abbreviation = 'L2__NO2___'
    elif product == 'SO2':
        product_abbreviation = 'L2__SO2___'
    elif product == 'HCHO':
        product_abbreviation = 'L2__HCHO__'
    elif product == 'AI':
        product_abbreviation = 'L2__AER_AI'
    elif product == 'ALH':
        product_abbreviation = 'L2__AER_LH'

    return product_abbreviation
```

In [14]:
```
# Create list of TROPOMI data file names for user-entered product, latency, search regio
# "product_abbreviation": parameter variable from "get_tropomi_product_abbreviation(prod
# "start_date", "end_date": parameter variables from "convert_date_sentinel_api_format(y
# "west_lon", "east_lon", "south_lat", "north_lat", "latency": parameter variables from
```

```
def tropomi_list_files(west_lon, east_lon, south_lat, north_lat, start_date, end_date, p

    # Access S5P Data Hub using guest login credentials
    api = SentinelAPI('s5pguest', 's5pguest', 'https://s5phub.copernicus.eu/dhus')

    # Query API for specified region, start/end dates, data product
    footprint = 'POLYGON((' + west_lon + ' ' + south_lat + ',' + east_lon + ' ' + south_
    try:
        products = api.query(area=footprint, date=(start_date + 'T00:00:00Z', end_date +
    except:
        print('Error connecting to SciHub server. This happens periodically. Run code ag

    # Convert query output to pandas dataframe (df) (part of Sentinelsat library)
    products_df = api.to_dataframe(products)

    # Extract data file names from dataframe to list
    if len(products_df) > 0:
        file_name_list = products_df['filename'].tolist()
        file_size_list = products_df['size'].tolist()
    else:
        file_name_list = []
        file_size_list = []

    return file_name_list, file_size_list, products
```

In [15]:
```
# Download TROPOMI data files
# "save_path": parameter variable set in main function
# "products": parameter variable from "tropomi_list_files( )" function

def tropomi_download_files(products, save_path):

    # Query S5P Data Hub using guest login credentials
    api = SentinelAPI('s5pguest', 's5pguest', 'https://s5phub.copernicus.eu/dhus')

    # Download data files to specified subdirectory
    # Note: Sentinelsat library includes tqdm download progress bar
    try:
        api.download_all(products, save_path)
    except KeyboardInterrupt:
        print('\nDownload was interrupted by user.')
```

In [16]:
```
# Print available TROPOMI data files that match user specifications, with option to down
# "save_path": parameter variable set in main function
# "product_abbreviation": parameter variable from "get_tropomi_product_abbreviation(prod
# "start_date", "end_date": parameter variables from "convert_date_sentinel_api_format(d
# "west_lon", "south_lat", "east_lon", "north_lat", "latency": parameter variables from

def get_tropomi_files(west_lon, east_lon, south_lat, north_lat, start_date, end_date, pr

    # Query S5P Data Hub and list file names matching user-entered info
    file_name_list, file_size_list, products = tropomi_list_files(west_lon, east_lon, so

    # Print list of available file names/sizes
    if len(file_name_list) > 0:
        print('\nList of available data files (file size):')
        for file, size in zip(file_name_list, file_size_list):
            print(file, ' (', size, ')', sep='')

        # Print directory where files will be saved
        print('\nData files will be saved to:', save_path)

        # Ask user if they want to download the available data files
        # If yes, download files to specified directory
        download_question = 'Would you like to download the ' + str(len(file_name_list))
```

```
        ask_download = input(download_question)
        if ask_download in ['yes', 'YES', 'Yes', 'y', 'Y']:
            tropomi_download_files(products, save_path)
        else:
            print('\nFiles are not being downloaded.')
    else:
        print('\nNo files retrieved.  Check settings and try again.')
```

Main. Do the query and download.

In [ ]:
```
# Execute search to find available TROPOMI L2 data files, with option to download files
# Get values from widget menus (search parameters) using ".value"

# Main function

# Set directory to save downloaded files (as pathlib.Path object)
# Use current working directory for simplicity
save_path = Path.cwd()

# Get TROPOMI product abbreviation used in file name
product_abbreviation = get_tropomi_product_abbreviation(product.value)

# Change user-entered observation year/month/day for observation period to format used b
start_date = convert_date_sentinel_api_format(start_year.value, start_month.value, start
end_date = convert_date_sentinel_api_format(end_year.value, end_month.value, end_day.val

# Convert latitude/longitude values entered as floats to string format used by Sentinel
west_lon = str(west_lon_float.value)
east_lon = str(east_lon_float.value)
south_lat = str(south_lat_float.value)
north_lat = str(north_lat_float.value)

# Execute script
get_tropomi_files(west_lon, east_lon, south_lat, north_lat, start_date, end_date, produc
```

```
Querying products:  50%|####9     | 100/202 [00:00<?, ?product/s]
List of available data files (file size):
S5P_OFFL_L2__NO2____20230717T034611_20230717T052741_29833_03_020500_20230720T143035.nc
(554.27 MB)
S5P_OFFL_L2__NO2____20230717T002312_20230717T020442_29831_03_020500_20230720T140859.nc
(581.08 MB)
S5P_OFFL_L2__NO2____20230717T222237_20230718T000406_29844_03_020500_20230721T110429.nc
(573.65 MB)
S5P_OFFL_L2__NO2____20230717T185937_20230717T204107_29842_03_020500_20230721T105019.nc
(580.3 MB)
S5P_OFFL_L2__NO2____20230717T204107_20230717T222237_29843_03_020500_20230721T105329.nc
(576.16 MB)
S5P_OFFL_L2__NO2____20230717T171808_20230717T185937_29841_03_020500_20230721T091037.nc
(585.09 MB)
S5P_OFFL_L2__NO2____20230717T153638_20230717T171808_29840_03_020500_20230721T084843.nc
(574.34 MB)
S5P_OFFL_L2__NO2____20230717T121339_20230717T135509_29838_03_020500_20230721T050121.nc
(577.47 MB)
S5P_OFFL_L2__NO2____20230717T103210_20230717T121339_29837_03_020500_20230721T021736.nc
(575.41 MB)
S5P_OFFL_L2__NO2____20230717T135509_20230717T153638_29839_03_020500_20230721T064825.nc
(554.57 MB)
S5P_OFFL_L2__NO2____20230717T052741_20230717T070910_29834_03_020500_20230721T015449.nc
(583.61 MB)
S5P_OFFL_L2__NO2____20230717T020442_20230717T034611_29832_03_020500_20230720T141325.nc
(583.56 MB)
S5P_OFFL_L2__NO2____20230716T224143_20230717T002312_29830_03_020500_20230720T140959.nc
(574.38 MB)
S5P_OFFL_L2__NO2____20230716T210013_20230716T224143_29829_03_020500_20230720T140514.nc
```

2 Unnecessary currently: Alternate Download from GES DISC

Link of the dataset can be found here. The variables of interest are – Geolocation data: Latitude, Longitude, and UTC; Science data : ColumnAmountNO2. The description (i.e., the readme file at the same location) says that it is the NO_2 vertical column density.

To download the data, go through the following:

1. To download in batch, I need an Earthdata account and then need to link GES DISC with this account.
2. Use the GES DISC sampler to subset the data. I sampled to choose the above 4 variables.
3. It takes us to the window that contains a text file which contains a set of links to the data in netcdf4 format.
4. Manual download is possible by clicking the links, but there are > 5000 files.
5. The next step is to download the text file with the set of links. Call it "links.txt".
6. To download on batch, we need to create 3 pre-requisite files as below:
   (a) Create the ".netrc" file in Mac by typing the following in the command prompt:

   ```
   cd $HOME
   touch .netrc
   echo ''machine urs.earthdata.nasa.gov login <uid> password <password>
   chmod 0600
   ```

   Note: Remove "<". <uid> is user-id, not email id.

   (b) Create the urs_cookies" file in Mac by typing the following in the command prompt:

   ```
   cd $HOME
   touch .urs_cookies
   ```

   (c) Create the "dodsrc" file in Mac by typing the following in the command prompt:

   ```
   cd $HOME
   touch .dodsrc
   ```

   Then edit the file with text editor and write the following:

   ```
   HTTP.NETRC=< YourHomeDirectory >/.netrc
   HTTP.COOKIEJAR=< YourHomeDirectory >/.urs_cookies
   ```

   Note: < YourHomeDirectory > is where the previous two files are saved. I may need to do cmd+shift+. to see the hidden files, or "ls -a" in the terminal.
7. Once they are created, I run the following in the terminal:

   ```
   cat links.txt | tr -d '\r' | xargs -n 1 curl -LJO -n -c ~/.urs_cookie
   ```

3 Analysis of Data on July 1, 2023

3.1 Scatterplot

There are 11 `.nc` files on this date and the total number of unique lon-lat combination is more than $20m$. Therefore, even plotting looks not possible (or at least time consuming). For now, the following procedure is used to plot the data :

(i.) First, a set of locations with extreme 5% values of NO_2 have been left out from the plotting. With these extremes, the histograms (and plots) looked meaningless.

(ii.) A random subsample of $500,000$ points have been randomly sampled from the rest.

(iii.) Plots have been done using `PyPlot`.

Following is the scatterplot of a random subsample – ***.

3.2 Concerns

I.) Too large data, some upscaling needed.

II.) If we are going to use covariates, MERRA data at GET DISC was available at a $0.25° \times 0.25°$ resolution, which may force us upscale the TROPOMI data to the same resolution.

III.) Instead of outright removal of the extreme 5% NO_2 data, might use a kernel smoother based on the remaining 95% and thus impute the outliers.

IV.) *** Github PUSH codes and data.

3.3 Trial with Uber h3

`Python` code available at this link.