Merging datasets from two satellite intruments : ISS_LIS and TROPOMI into dataframe and save as CSV

In [1]:
```
import netCDF4 as nc4
from netCDF4 import Dataset
import numpy as np
import pandas as pd
```

STEP-01: READ LIS CSV FILE

In [2]:
```
## Lets read the lis csv file for 16-17th july 2022,region:canada
iss_lis = pd.read_csv(r'C:\Users\Aditya\Desktop\Research2022_Walker\iss_no2_test\iss_lis_16july2022.csv')
```

In [3]:
```
### Change columns name to lat and lon (as per the column name of TROPOMI datafiles lat and lon)

iss_lis.rename(columns={'flash_lat': 'lat', 'flash_lon': 'lon'}, inplace=True)
iss_lis
```

Out[3]:

	lat	lon	flash_radiance	flash_time_loc	smo
0	51.050793	-104.763046	13620.0	2022-07-16 05:07:26 -0400	630
1	51.144081	-104.693893	335410.0	2022-07-16 05:07:29 -0400	631
2	51.175934	-104.811951	145902.0	2022-07-16 05:07:31 -0400	632
3	51.129494	-104.607941	345482.0	2022-07-16 05:07:37 -0400	633
4	51.121143	-104.657196	638645.0	2022-07-16 05:07:41 -0400	634
...	...	...	...	...	...
1353	49.456676	-101.931213	238514.0	2022-07-16 08:21:55 -0400	3023
1354	49.344345	-101.558197	134576.0	2022-07-16 08:21:55 -0400	3024
1355	49.296913	-101.486671	42152.0	2022-07-16 08:22:03 -0400	3025
1356	49.426968	-102.275391	47077.0	2022-07-16 08:21:57 -0400	3026
1357	49.408260	-102.249146	330239.0	2022-07-16 08:22:07 -0400	3027

1358 rows × 5 columns

STEP-02: READ TROPOMI CSV FILE

In [4]:
```
## Lets read the lis csv file for 16-17th july 2022,region:canada
tp_no2 = pd.read_csv(r'C:\Users\Aditya\Desktop\Research2022_Walker\iss_no2_test\tropomi_iss_july16to17_20
```

In [5]:
```
tp_no2
```

Out[5]:

	Unnamed: 0	along_index	across_index	lat	lon	no2_delta_time	no2_val	qa_val
0	139855	310	355	41.000668	-71.981290	2022-07-16 17:33:11 -0400	0.000007	100
1	139856	310	356	41.004734	-71.997700	2022-07-16 17:33:11 -0400	0.000007	100
2	139857	310	357	41.008750	-72.014260	2022-07-16 17:33:11 -0400	0.000007	100
3	139858	310	358	41.012726	-72.030790	2022-07-16 17:33:11 -0400	0.000007	100
4	139859	310	359	41.016647	-72.047330	2022-07-16 17:33:11 -0400	0.000006	100
...	...	...	...	...	...	...	...	...
375568	32489	1943	138	49.631554	-53.248207	2022-07-16 23:02:31 -0400	0.000006	0
375569	32490	1943	139	49.642350	-52.452460	2022-07-16 23:02:31 -0400	0.000007	0
375570	32491	1943	140	49.653100	-51.656254	2022-07-16 23:02:31 -0400	0.000006	0
375571	32492	1943	141	49.663803	-50.859280	2022-07-16 23:02:31 -0400	0.000006	0
375572	32493	1943	142	49.674460	-50.062878	2022-07-16 23:02:31 -0400	0.000006	0

375573 rows × 8 columns

STEP-03: Find NOx profile around a each flash light (in square boxes)

Now as we have successfully read lis and tropomi data from csv, next step is to analysis the Nitrogen oxides(tropomi data) produced due to lightning strike(lis data) within ~100 kilometers around each flash pointFor this we create an imaginary boundary or a box around each flash point

In [6]:
```
### create square boxes with 1 degress (1 degree ~ 111kms) distance from center of each flash
##(note: actual square box dimesion would be 2degx 2deg)

iss_lis['flash_lat_plus'] = iss_lis['lat'] + 1
iss_lis['flash_lat_minus'] = iss_lis['lat'] - 1
iss_lis['flash_lon_plus'] = iss_lis['lon'] + 1
iss_lis['flash_lon_minus'] = iss_lis['lon'] - 1
iss_lis
```

Out[6]:

	lat	lon	flash_radiance	flash_time_loc	smo	flash_lat_plus	flash_lat_minus	flash_lon_plus	flash_lon_mi
0	51.050793	-104.763046	13620.0	2022-07-16 05:07:26 -0400	630	52.050793	50.050793	-103.763046	-105.763(
1	51.144081	-104.693893	335410.0	2022-07-16 05:07:29 -0400	631	52.144081	50.144081	-103.693893	-105.693
2	51.175934	-104.811951	145902.0	2022-07-16 05:07:31 -0400	632	52.175934	50.175934	-103.811951	-105.811
3	51.129494	-104.607941	345482.0	2022-07-16 05:07:37 -0400	633	52.129494	50.129494	-103.607941	-105.607
4	51.121143	-104.657196	638645.0	2022-07-16 05:07:41 -0400	634	52.121143	50.121143	-103.657196	-105.657
...	...	...	...	...	...	...	...	...	
1353	49.456676	-101.931213	238514.0	2022-07-16 08:21:55 -0400	3023	50.456676	48.456676	-100.931213	-102.931
1354	49.344345	-101.558197	134576.0	2022-07-16 08:21:55 -0400	3024	50.344345	48.344345	-100.558197	-102.558
1355	49.296913	-101.486671	42152.0	2022-07-16 08:22:03 -0400	3025	50.296913	48.296913	-100.486671	-102.486
1356	49.426968	-102.275391	47077.0	2022-07-16 08:21:57 -0400	3026	50.426968	48.426968	-101.275391	-103.275
1357	49.408260	-102.249146	330239.0	2022-07-16 08:22:07 -0400	3027	50.408260	48.408260	-101.249146	-103.249

1358 rows × 9 columns

Now our next task is to check the no2 values(from tropomi data) around each flash and eliminate the no2 data outside our box (an imaginary square boundry created around each flash ~100KMS) For doing this we select 2 flashes for now and check condition

In [7]:
```
### flash strikes in given region (this region is subset to the region queried for tropomi_no2 data)

iss_lis_1 = iss_lis.query('-102 <= lon <= -101.5 and 49.9 < lat <= 50').reset_index().round(decimals=6)
iss_lis_1
```

Out[7]:

	index	lat	lon	flash_radiance	flash_time_loc	smo	flash_lat_plus	flash_lat_minus	flash_lon_plus	flash_lon
0	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	-102.
1	1300	49.968143	-101.669922	10764.0	2022-07-16 06:46:16 -0400	2935	50.968143	48.968143	-100.669922	-102.

Next, we add each flash values to whole no2 data i.e. say the tropomi data has 375533 rows so we multiple it by x2 (as 2 flashes are considered). This is done by cross merging two dataframes

In [8]:
```
%%time
### Cross merging (required step to check if given no2 reading within flash range or not)

df_out = iss_lis_1.merge(tp_no2, how='cross')
df_out.head()
```

```
CPU times: total: 188 ms
Wall time: 165 ms
```

Out[8]:

	index	lat_x	lon_x	flash_radiance	flash_time_loc	smo	flash_lat_plus	flash_lat_minus	flash_lon_plus	flash_lon
0	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	-102.
1	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	-102.
2	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	-102.
3	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	-102.
4	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	-102.

In [9]:
```
len(df_out)
```

Out[9]:
```
751146
```

Finally as we have each flash values added to tropomi rows, check conditon such that latitude and longitude of no2 lies inside an imaginary box which we previously creted around each flash point

In [10]:
```
%%time
### Check if longitude of no2 lies in between the square box (ie min and max longitude of flash lon) and
### IF condition satisfied add index value of flash (flash ID) to this new column else NaN

df_out['lon_no2aroundlis'] = df_out.apply(lambda x: x['index'] if x['lon_y'] >= x['flash_lon_minus'] and
```

```
CPU times: total: 17.8 s
Wall time: 18.1 s
```

In [11]:
```
%%time
###Drop rows in DF when new column create by above condition has NaN value (column:'lon_no2aroundlis')

df_out_1 = df_out.dropna(subset = ['lon_no2aroundlis'])
df_out_1
```

```
CPU times: total: 78.1 ms
Wall time: 77.8 ms
```

Out[11]:

	index	lat_x	lon_x	flash_radiance	flash_time_loc	smo	flash_lat_plus	flash_lat_minus	flash_lon_plus	fla
1151	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	
1152	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	
1153	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	
1154	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	
1155	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	
...	...	...	...	...	...	...	...	...	...	
751052	1300	49.968143	-101.669922	10764.0	2022-07-16 06:46:16 -0400	2935	50.968143	48.968143	-100.669922	
751053	1300	49.968143	-101.669922	10764.0	2022-07-16 06:46:16 -0400	2935	50.968143	48.968143	-100.669922	
751054	1300	49.968143	-101.669922	10764.0	2022-07-16 06:46:16 -0400	2935	50.968143	48.968143	-100.669922	
751055	1300	49.968143	-101.669922	10764.0	2022-07-16 06:46:16 -0400	2935	50.968143	48.968143	-100.669922	
751056	1300	49.968143	-101.669922	10764.0	2022-07-16 06:46:16 -0400	2935	50.968143	48.968143	-100.669922	

29212 rows × 19 columns

In [12]:
```
%%time
### Check if latitude of no2 lies in between the square box (ie min and max latitude of flash lon) and ad
### IF condition satisfied add index value of flash (flash ID) to this new column else NaN

df_out['lat_no2aroundlis'] = df_out_1.apply(lambda x: x['index'] if x['lat_y'] >= x['flash_lat_minus'] an
```

```
CPU times: total: 828 ms
Wall time: 865 ms
```

If NO2 value is within flash if both latitude and longitude are within imaginary square box, Previously we already deleted rows when column:'lon_no2aroundlis' was Null now lets do it for column 'lat_no2aroundlis

In [13]:
```
%%time
###Drop rows in DF when new column create by above condition has NaN value (column:'lon_no2aroundlis')

df_out_2 = df_out.dropna(subset = ['lat_no2aroundlis'])
df_out_2
```

```
CPU times: total: 78.1 ms
Wall time: 78.8 ms
```

Out[13]:

	index	lat_x	lon_x	flash_radiance	flash_time_loc	srno	flash_lat_plus	flash_lat_minus	flash_lon_plus	fla
40189	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	
40190	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	
40191	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	
40192	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	
40193	617	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	1375	50.990746	48.990746	-100.663162	
...	...	...	...	...	...	...	...	...	...	
748686	1300	49.968143	-101.669922	10764.0	2022-07-16 06:46:16 -0400	2935	50.968143	48.968143	-100.669922	
748687	1300	49.968143	-101.669922	10764.0	2022-07-16 06:46:16 -0400	2935	50.968143	48.968143	-100.669922	
748688	1300	49.968143	-101.669922	10764.0	2022-07-16 06:46:16 -0400	2935	50.968143	48.968143	-100.669922	
748689	1300	49.968143	-101.669922	10764.0	2022-07-16 06:46:16 -0400	2935	50.968143	48.968143	-100.669922	
748690	1300	49.968143	-101.669922	10764.0	2022-07-16 06:46:16 -0400	2935	50.968143	48.968143	-100.669922	

4900 rows × 20 columns

In [14]:
```
%%time
### Merge dataframe based on Latitude and Longitude columns
### Cross merged dataframe has lis data and tropomi data in different columns but to analyze the final da

df1_iss = df_out_2[['index','lat_x','lon_x','flash_radiance','flash_time_loc']]
df1_iss.rename(columns={'index':'flash_id','lat_x': 'lat', 'lon_x': 'lon'}, inplace=True)
df1_iss = df1_iss.drop_duplicates(keep='first')

df1_no2 = df_out_2[['no2_delta_time','lat_y','lon_y','no2_val','qa_val','lat_no2aroundlis']]
df1_no2.rename(columns={'lat_y': 'lat', 'lon_y': 'lon','lat_no2aroundlis':'flash_id'}, inplace=True)

iss_tp_box = pd.concat([df1_iss,df1_no2]).dropna(subset = ['no2_val','flash_radiance'], how='all')### dro
iss_tp_box
```

```
CPU times: total: 422 ms
Wall time: 423 ms
<timed exec>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.ht
ml#returning-a-view-versus-a-copy
<timed exec>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.ht
ml#returning-a-view-versus-a-copy
```

Out[14]:

	flash_id	lat	lon	flash_radiance	flash_time_loc	no2_delta_time	no2_val	qa_val
40189	617.0	49.990746	-101.663162	10764.0	2022-07-16 06:46:16 -0400	NaN	NaN	NaN
413297	1300.0	49.968143	-101.669922	10764.0	2022-07-16 06:46:16 -0400	NaN	NaN	NaN
40189	617.0	49.805138	-100.737816	NaN	NaN	2022-07-16 17:35:41 -0400	0.000006	100.0
40190	617.0	49.804000	-100.826590	NaN	NaN	2022-07-16 17:35:41 -0400	0.000006	100.0
40191	617.0	49.802660	-100.915690	NaN	NaN	2022-07-16 17:35:41 -0400	0.000006	100.0
...	...	...	...	...	...	...	...	...
748686	1300.0	50.792034	-102.105420	NaN	NaN	2022-07-16 23:02:24 -0400	0.000007	0.0
748687	1300.0	50.801212	-101.796880	NaN	NaN	2022-07-16 23:02:24 -0400	0.000006	0.0
748688	1300.0	50.810383	-101.484080	NaN	NaN	2022-07-16 23:02:24 -0400	0.000006	0.0
748689	1300.0	50.819553	-101.166580	NaN	NaN	2022-07-16 23:02:24 -0400	0.000006	0.0
748690	1300.0	50.828716	-100.843780	NaN	NaN	2022-07-16 23:02:24 -0400	0.000006	0.0

4902 rows × 8 columns

In [24]:
```
#iss_tp_box.to_csv('iss_lis_and_tropomi_no2_merged_box_analysis_two_flashes_v3.csv')
```

cOdE eNdS hErE

Final CSV file shoul look as...

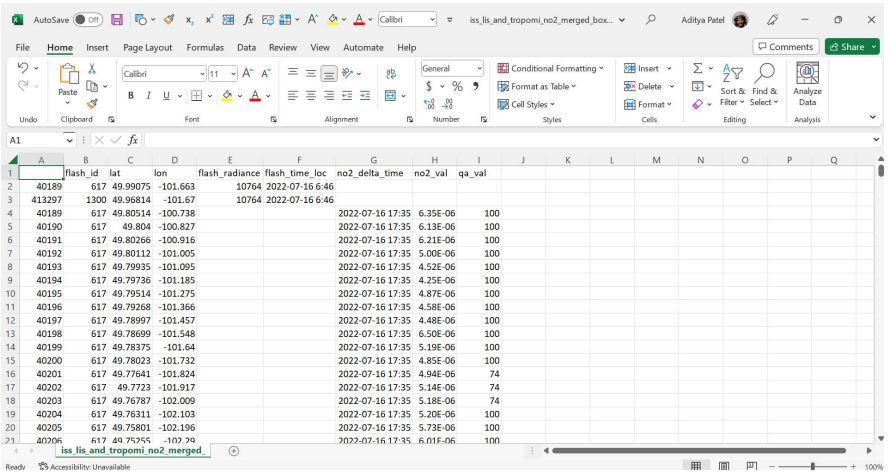
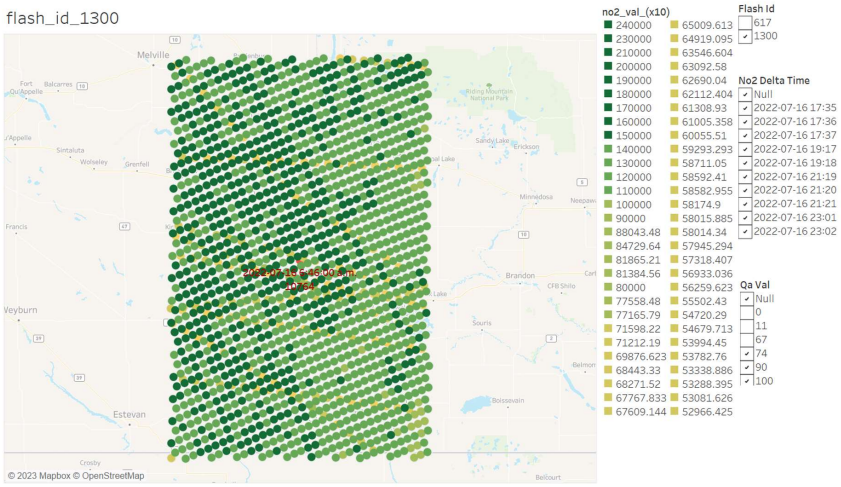


TABLEAU VISUALIZATION: Nitrogen oxide profile around lightning with FLASH ID - 1300 (Region is somewhere in south Saskatchewan,Canada)

1. Red dot represents the highest flash value for flash ID 1300 (represented by NULL mean no2 value is zero)
2. Each NO2 values are multiplied by 10^10 to analyse
3. NO2 data is filtered wrt qa_value > 50
4. Null value for NO2 is filtered
5. Dark green is mean more NOx produced, yellow means least
6. Time lag between NOx and ISS_LIS reading ie NOx is detected ~13hours after the flash was detected by LIS * (ISS_LIS: 16th july 2022, 6:00AM & TROPOMI_NOx: 16th july 2022, 7:00PM)



Issues yet to be addressed

ISSUE: 01:_Major issue

High processing time to check box conditon when analysing more number of flashes

---

Preference:

High priority

Details: The raw data from NETCDF4 file is is in the array form which is converted to pandas dataframe and finally processed data is saved as CSV file. NO2 data is of very high resolution and large approx 3.7lakhs rows We want to checkthe NO2 data point are lying within 100 kilometers of each flash point Currently we add columns to lis data and create an imaginary approx. square box by adding and substracting 1 degrees(approx 111kms) to flash latitude and longitude. The tropomi and lis data frames are cross merged: Each flash is added as whole NO2 data(approx 3.7lakhs rows)

And then a condition whether the NO2 latitude and longitude values are lying in between flash min and max longitude (imagine them as horizontal sides of a square) ae well as flash min and max latitude (imagine them as vertical sides of a square) is checked

Specific Requirement The above code worked good when number of flash points to be analysed are less in number

But as more and more number of flashes are analyzed it hangs up the system and pandas fail here.

Required outcome: to find NO2 values within specfic range of flash point Any improvement/new method which helps in Shortening of processing time when large number of flash points are to be analyzed at once.

Related attempts:[1] *Convert panadas dataframe into dask dataframe (parallelized pandas)

```
#import dask.dataframe as dd
#ddf = dd.from_pandas(df_out_test, npartitions=10
```

```
# def boxlon(df):
#     return df.apply(lambda x: x['srno'] if x['lon_y'] >= x['flash_lon_minus'] and
x['lon_y'] <= x['flash_lon_plus'] else np.nan, axis=1)

# p = ddf.map_partitions(boxlon, meta=(None, 'int64'))
# p.compute()
# #p.to_csv('p_dask_test.csv')
```

ISSUE: 01:_Major issue

High processing time to check box conditon when analysing more number of flashes

---

Preference:

High priority

Related attempts:[2]*Divide a big box into smaller boxes

Currently we have a data queried spatially for a wider region:-140 <= lon <= -50 and 41 < lat <= 56' and so each code checks NO2 value for the whole region whether it is within 100kms of flash point What if? We divide the big box vertically into say 4 and execute the same code to reduce number of iterations Further we can eliminate the NO2 values lying way outside flashes by quering for min and max lat lon for lis inside each box

```
### flash strikes in given region (this region is subset to the region queried for
tropomi_no2 data)

iss_lis_2 = iss_lis.query('-125 <= lon <= -100')#.reset_index().round(decimals=6)
print(iss_lis_2.lat.min()-1) ##min value =48
print(iss_lis_2.lat.max()+1) ##max value =52.26
print(iss_lis_2.lon.min()-1) ##min value =-106
print(iss_lis_2.lon.max()+1) ##max value =-100
tp_no2_2 = tp_no2.query('-106 <= lon <= -100' and '48 <= lat <=
52.26')#.reset_index().round(decimals=6)
print(iss_lis_2)
print(tp_no2_2)
```

*limitation of method[2]: When many flashes are spatially clustered and occuring very close to each other divison of a big box into smaller say into 4 won't help much

ISSUE: 01:_Major issue

High processing time to check box conditon when analysing more number of flashes

---

Preference:

High priority

Related attempts:[3]*Directly process raw satellite data obltained in array form from NETCDF4 files and find NO2 field near each flash point (instead of converting it to raw data pandas dataframe and than creating imaginary box to find it)

```
from scipy.spatial.distance import pdist, squareform
import numpy as np
```

```
points = np.random.uniform(-0.1, 0.1, (5,2))  ### create an array
points
```

Output: array([[0.08997501, 0.06332118], [0.01879947, -0.08414794], [0.02069245, 0.00011062], [-0.057657 , 0.0563319], [0.0869381 , -0.03660487]])

```
# Compute the distance between each different pair of points in X with pdist.
# Then, just for ease of working, convert to a typical symmetric distance matrix
# with squareform.
dists = squareform(pdist(points))
dists

poi = points[1] # point of interest
dist_min = 0.1
close_points = dists[1] < dist_min

print("There are {} other points within a distance of {} from the point "
    "({:.3f}, {:.3f})".format(close_points.sum() - 1, dist_min, *poi))
```

Output: There are 2 other points within a distance of 0.1 from the point (0.019, -0.084)

*Related resources: https://scipy.github.io/devdocs/reference/spatial.distance.html

All new ideas/logic to excecute above code more efficiently and faster are always welcomed and Thank you very much for contributing your precious time here! :)