TEMPO via the Python Interface to Remote Sensing Information Gateway

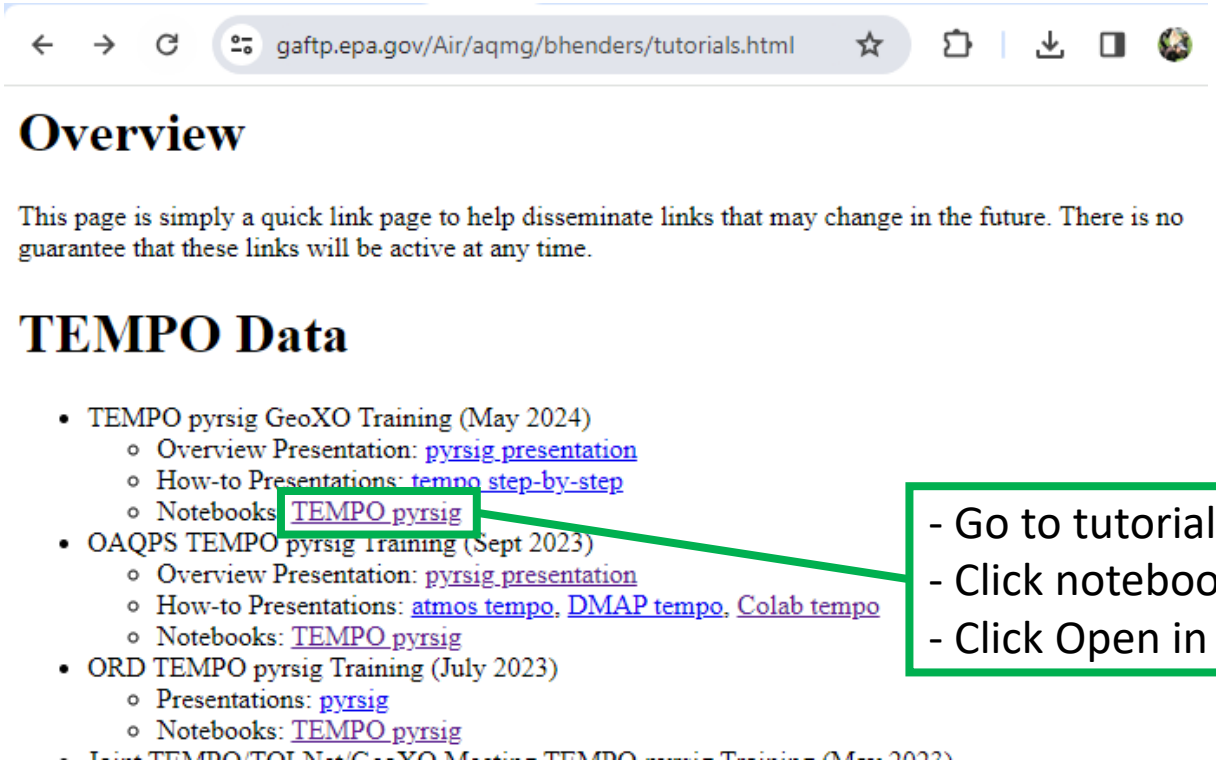
Presented: Barron H. Henderson

Date: 2024-05-09

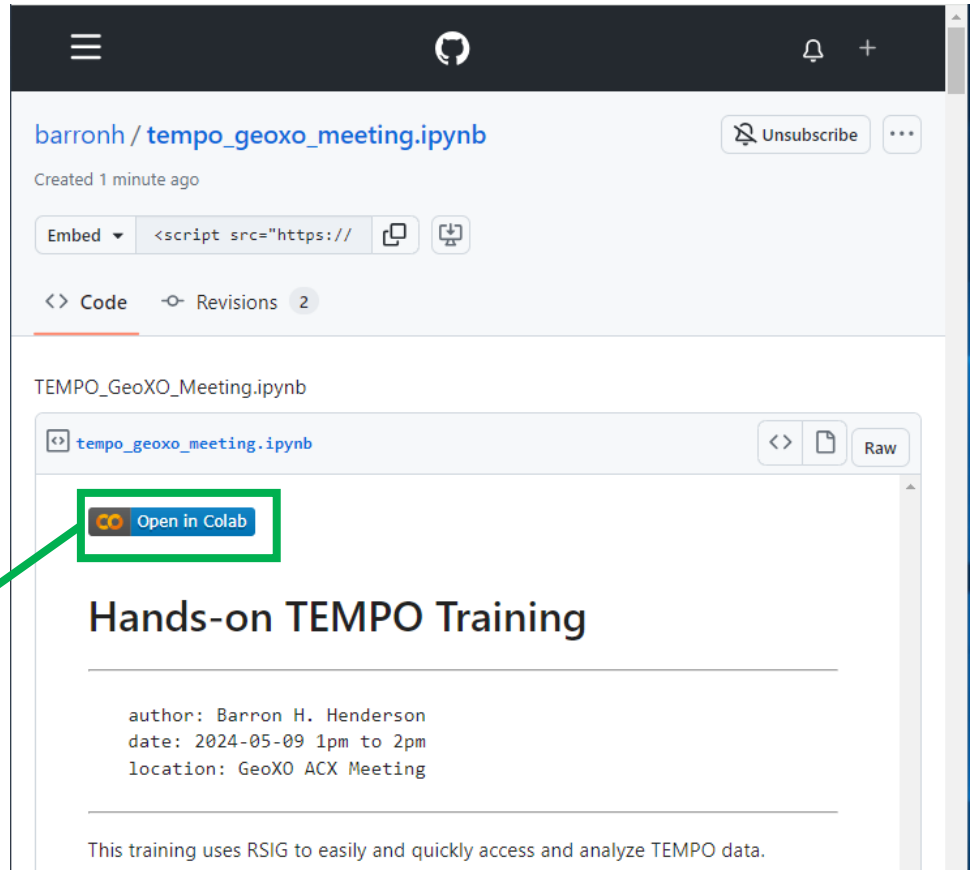
RSIG Team: Jim J. Szykman, Luke Valin, Todd Plessel, Matt Freeman

Step-by-Step Visuals

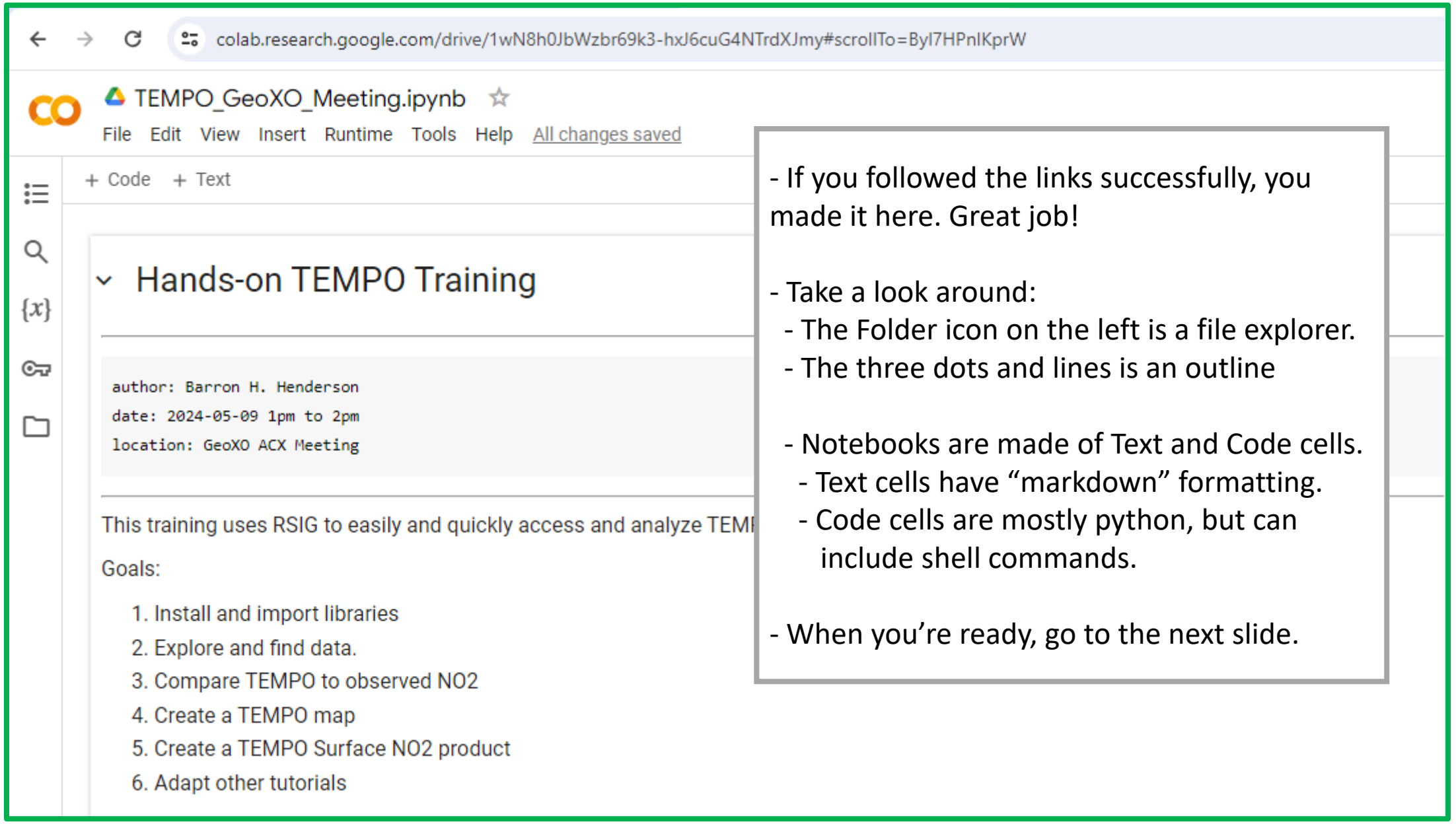
- Materials are all available
  - https://gaftp.epa.gov/Air/aqmg/bhenders/tutorials.html



- Go to tutorials
- Click notebook link
- Click Open in Colab



https://gist.github.com/barronh/63f8c0956bbe9b31f6d693b465cd659d/

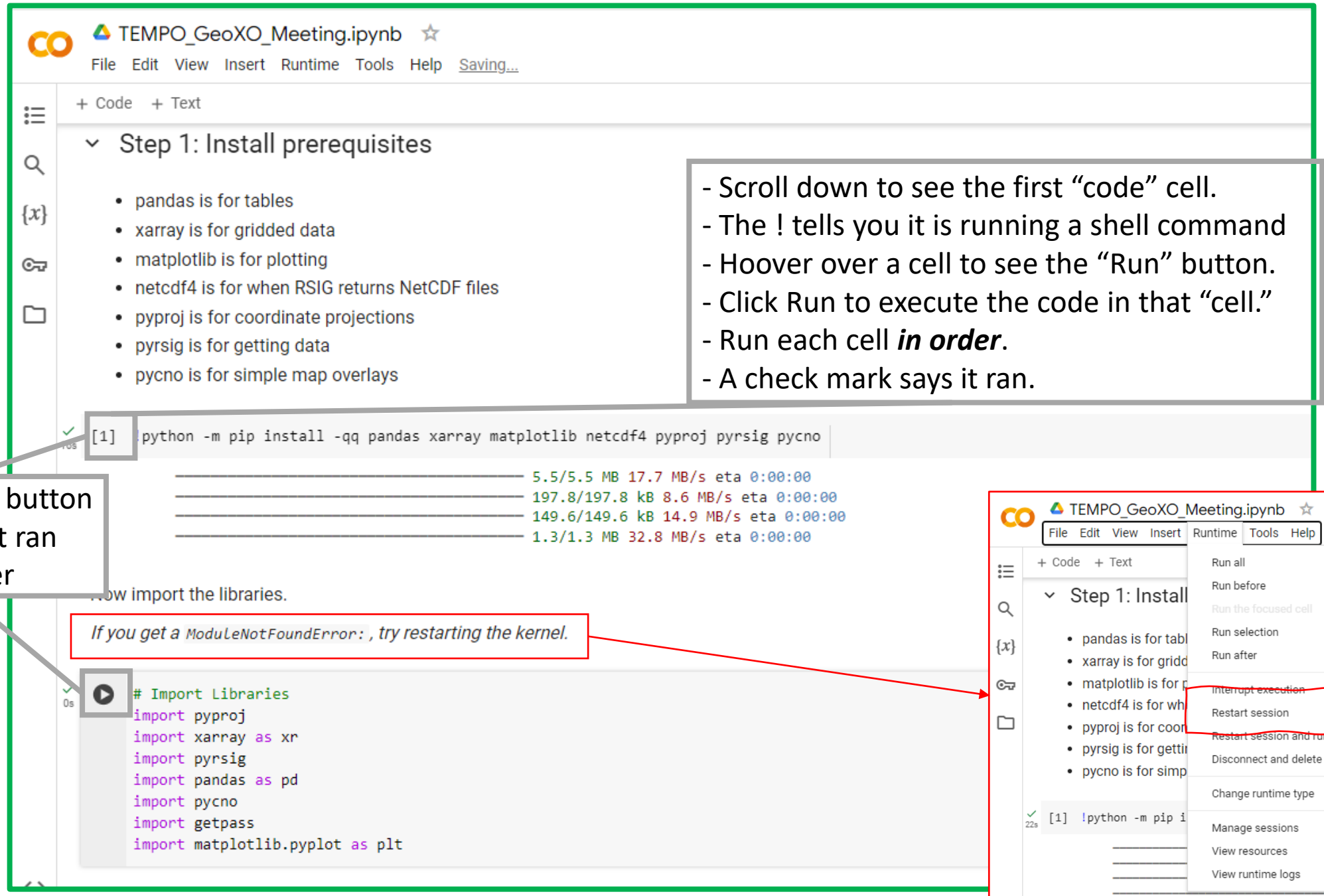


- If you followed the links successfully, you made it here. Great job!

- Take a look around:
  - The Folder icon on the left is a file explorer.
  - The three dots and lines is an outline

  - Notebooks are made of Text and Code cells.
    - Text cells have “markdown” formatting.
    - Code cells are mostly python, but can include shell commands.

- When you’re ready, go to the next slide.



- Scroll down to see the first “code” cell.
- The ! tells you it is running a shell command
- Hoover over a cell to see the “Run” button.
- Click Run to execute the code in that “cell.”
- Run each cell ***in order***.
- A check mark says it ran.

- Click “Run” button
- [#] means it ran
- Run in order

Step 2: Exploring Data

- Import libraries
- Prepare a pyrsig object

- Here we're choosing a beginning date of Dec 18.
- A bounding box in the northeast.
- For more options, run ?pyrsig.RsigApi

```
[3] # Choosing a Northeast domain for 2023 December 18th
    # Dec 18th is in the public data.
    locname = 'nyc'
    bbox = (-74.8, 40.32, -71.43, 41.4)
    bdate = '2023-12-18'
```

```
[4] api = pyrsig.RsigApi(bdate=bdate, bbox=bbox, workdir=locname, gridfit=True)
    # api_key = getpass.getpass('Enter TEMPO key (anonymous if unknown):')
    api_key = 'anonymous'  # using public, so using anonymous
    api.tempo_kw['api_key'] = api_key
```

```
# after the cell runs, click on the table button.
# Then use filters to find tempo data producs by names that start with tempo
api.descriptions()
```

	name	label	description	bbox_str	beginPosition	timeResolution	endPosition	prefix
0	airnow.pm25	pm25(ug/m3)	UTC hourly mean surface measured particulate m...	-157 21 -51 59	2003-01-02T00:00:00Z	PT1H	now	airnow
1	airnow.pm10	pm10(ug/m3)	UTC hourly mean surface measured particulate m...	-157 21 -51 59	2003-01-02T00:00:00Z	PT1H	now	airnow
2	airnow.ozone	ozone(ppb)	UTC hourly mean surface measured ozone concent...	-157 21 -51 64	2003-01-02T00:00:00Z	PT1H	now	airnow
3	airnow.no	no(ppb)	UTC hourly mean surface measured nitric oxide ...	-157 21 -51 64	2003-01-02T00:00:00Z	PT1H	now	airnow
4	airnow.no2	no2(ppb)	UTC hourly mean surface measured nitrogen diox...	-157 21 -51 64	2003-01-02T00:00:00Z	PT1H	now	airnow

Step 2: Exploring Data

- Import libraries
- Prepare a pyrsig object

```
[3]  # Choosing a Northeast domain for 2023 December 18th
     # Dec 18th is in the public data.
     locname = 'nyc'
     bbox = (-74.8, 40.32, -71.43, 41.4)
     bdate = '2023-12-18'
```

```
[4]  api = pyrsig.RsigApi(bdate=bdate, bbox=bbox, workdir=locname, gridfit=True)
     # api_key = getpass.getpass('Enter TEMPO key (anonymous if unknown):')
     api_key = 'anonymous'  # using public, so using anonymous
     api.tempo_kw['api_key'] = api_key
```

more later

- Public release date, so api_key is anonymous.
- API Key's are only necessary for PurpleAir and password protected TEMPO.

```
     # after the cell runs, click on the table button.
     # Then use filters to find tempo data producs by names that start with tempo
     api.descriptions()
```

	name	label	description	bbox_str	beginPosition	timeResolution	endPosition	prefix
0	airnow.pm25	pm25(ug/m3)	UTC hourly mean surface measured particulate m...	-157 21 -51 59	2003-01-02T00:00:00Z	PT1H	now	airnow
1	airnow.pm10	pm10(ug/m3)	UTC hourly mean surface measured particulate m...	-157 21 -51 59	2003-01-02T00:00:00Z	PT1H	now	airnow
2	airnow.ozone	ozone(ppb)	UTC hourly mean surface measured ozone concent...	-157 21 -51 64	2003-01-02T00:00:00Z	PT1H	now	airnow
3	airnow.no	no(ppb)	UTC hourly mean surface measured nitric oxide ...	-157 21 -51 64	2003-01-02T00:00:00Z	PT1H	now	airnow
4	airnow.no2	no2(ppb)	UTC hourly mean surface measured nitrogen diox...	-157 21 -51 64	2003-01-02T00:00:00Z	PT1H	now	airnow

Step 2: Exploring Data

- Import libraries
- Prepare a pyrsig object

```
[3]  # Choosing a Northeast domain for 2023 December 18th
     # Dec 18th is in the public data.
     locname = 'nyc'
     bbox = (-74.8, 40.32, -71.43, 41.4)
     bdate = '2023-12-18'
```

```
[4]  api = pyrsig.RsigApi(bdate=bdate, bbox=bbox, workdir=locname, gridfit=Tr
     # api_key = getpass.getpass('Enter TEMPO key (anonymous if unknown):')
     api_key = 'anonymous'  # using public, so using anonymous
     api.tempo_kw['api_key'] = api_key
```

```
     # after the cell runs, click on the table button.
     # Then use filters to find tempo data producs by names that start with tempo
     api.descriptions()
```

- Some cells have output
- Descriptions is a DataFrame of most RSIG data products.
- Click the table button, then click filter.
- Allows finding the name, which we'll call key

	name	label	description	bbox_str	beginPosition	timeResolution	endPosition	prefix
0	airnow.pm25	pm25(ug/m3)	UTC hourly mean surface measured particulate m...	-157 21 -51 59	2003-01-02T00:00:00Z	PT1H	now	airnow
1	airnow.pm10	pm10(ug/m3)	UTC hourly mean surface measured particulate m...	-157 21 -51 59	2003-01-02T00:00:00Z	PT1H	now	airnow
2	airnow.ozone	ozone(ppb)	UTC hourly mean surface measured ozone concent...	-157 21 -51 64	2003-01-02T00:00:00Z	PT1H	now	airnow
3	airnow.no	no(ppb)	UTC hourly mean surface measured nitric oxide ...	-157 21 -51 64	2003-01-02T00:00:00Z	PT1H	now	airnow
4	airnow.no2	no2(ppb)	UTC hourly mean surface measured nitrogen diox...	-157 21 -51 64	2003-01-02T00:00:00Z	PT1H	now	airnow

Step 2: Exploring Data

- Import libraries
- Prepare a pyrsig object

```
[6]  # Choosing a Northeast domain for 2023 December 18th
     # Dec 18th is in the public data.
     bbox = (-74.8, 40.32, -71.43, 41.4)
     bdate = '2023-12-18'
```

```
[7]  api = pyrsig.RsigApi(bdate=bdate, bbox=bbox, gridfit=True)
     # api_key = getpass.getpass('Enter TEMPO key (anonymous if unknown):')
     api_key = 'anonymous'  # using public, so using anonymous
     api.tempo_kw['api_key'] = api_key
```

```
     # after the cell runs, click on the table button.
     # Then use filters to find tempo data producs by names that start with tempo
     api.descriptions()
```

- Filter for tempo products.
- Find the Level 2 tropospheric vertical column

1 to 25 of 221 entries

index: to

name: tempo

label:

description:

bbox_str:

beginPosition:

timeResolution:

endPosition:

prefix:

Search by all fields:

index	name	label	description	bbox_str	beg
1309	tempo.l2.no2.solar_zenith_angle	l2.no2.solar_zenith_angle(deg)	Solar zenith angle at pixel center.	-180 -90 180 90	2023 17T(
1310	tempo.l2.no2.solar_azimuth_angle	l2.no2.solar_azimuth_angle(deg)	Solar azimuth angle at pixel center.	-180 -90 180 90	2023 17T(
1311	tempo.l2.no2.viewing_zenith_angle	l2.no2.viewing_zenith_angle(deg)	Viewing zenith angle at pixel center.	-180 -90 180 90	2023 17T(
1312	tempo.l2.no2.viewing_azimuth_angle	l2.no2.viewing_azimuth_angle(deg)	Viewing azimuth angle at pixel center.	-180 -90	2023

TEMPO_GeoXO_Meeting.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[9]  tempokey = 'tempo.l2.no2.vertical_column_troposphere'
```

```
[10] # By default, the pyrsig uses 'ascii' backend, but 'xdr' is faster;
     # both look the same in python, but the files are very different.
     # I'm using xdr here for speed
     df = api.to_dataframe(tempokey, backend='xdr')
     df
```

- You should have found this name
- We use this name to get a DataFrame of

- I'm requesting xdr format for speed,
- ascii is slower, but returns tab delimited text.
- For other options, see run ?api.to_dataframe

	Timestamp(UTC)	LONGITUDE(deg)	LATITUDE(deg)	no2_vertical_column_troposphere(molecules/cm2)	Longitude_SW(deg)	Longitude_SE(deg)	Longitude_NW(deg)	Longitude_NE(deg)	Latitude_SW(deg)
0	2023-12-18T13:11:00+0000	-73.261124	41.397552	9.600535e+14	-73.226740	-73.288809	-73.233614	-73.295652	41.408867
1	2023-12-18T13:11:00+0000	-72.579437	41.399281	9.477854e+14	-72.544632	-72.607487	-72.551695	-72.614491	41.411222
2	2023-12-18T13:11:00+0000	-72.642540	41.397034	1.068233e+15	-72.607487	-72.669970	-72.614491	-72.676945	41.408752
3	2023-12-18T13:11:00+0000	-72.704453	41.395725	2.464205e+15	-72.669970	-72.732351	-72.676945	-72.739336	41.407031
4	2023-12-18T13:11:00+0000	-72.767273	41.393372	1.179887e+15	-72.732351	-72.794975	-72.739336	-72.801992	41.405197
...	...	...	...	...	...	...	...	...	...
1082	2023-12-18T20:18:00+0000	-74.128571	40.326485	2.761060e+15	-74.095238	-74.155880	-74.101330	-74.161970	40.337756
1083	2023-12-18T20:18:00+0000	-74.189255	40.323994	2.565168e+15	-74.155880	-74.216782	-74.161970	-74.222843	40.335640
1084	2023-12-18T20:18:00+0000	-74.250313	40.322124	3.123098e+15	-74.216782	-74.277761	-74.222843	-74.283768	40.333426
1085	2023-12-18T20:18:00+0000	-74.311188	40.320011	3.012529e+15	-74.277761	-74.338547	-74.283768	-74.344549	40.331434
1086	2023-12-18T20:18:00+0000	-73.706215	40.320427	4.572452e+15	-73.672052	-73.734129	-73.678310	-73.740360	40.331833

23579 rows × 12 columns

TEMPO_GeoXO_Meeting.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

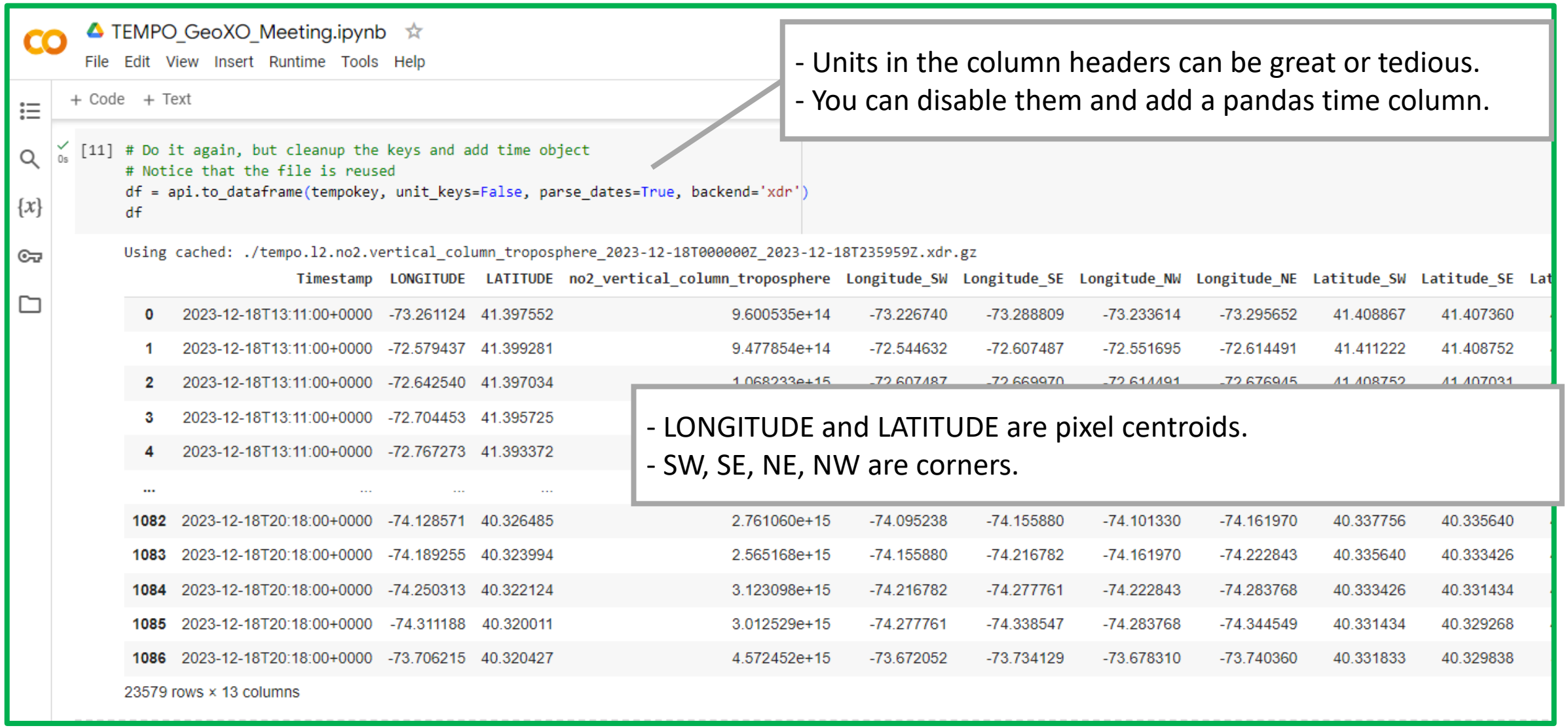
```
[9] tempokey = 'tempo.l2.no2.vertical_column_troposphere'
```

```
[10] # By default, the pyrsig uses 'ascii' backend, but 'xdr' is faster;
     # both look the same in python, but the files are very different.
     # I'm using xdr here for speed
     df = api.to_dataframe(tempokey, backend='xdr')
     df
```

	Timestamp(UTC)	LONGITUDE(deg)	LATITUDE(deg)	no2_vertical_column_troposphere(molecules/cm2)	Longitude_SW(deg)	Longitude_SE(deg)	Longitude_NW(deg)	Longitude_NE(deg)	Latitude_SW(deg)
0	2023-12-18T13:11:00+0000	-73.261124	41.397552	9.600535e+14	-73.226740	-73.288809	-73.233614	-73.295652	41.408867
1	2023-12-18T13:11:00+0000	-72.579437	41.399281	9.477854e+14	-72.544632	-72.607487	-72.551695	-72.614491	41.411222
2	2023-12-18T13:11:00+0000	-72.642540	41.397034	1.068233e+15	-72.607487	-72.669970	-72.614491	-72.676945	41.408752
3	2023-12-18T13:11:00+0000	-72.704453	41.395725	2.464205e+15	-72.669970	-72.732351	-72.676945	-72.739336	41.407031
4	2023-12-18T13:11:00+0000	-72.767273	41.393372	1.179887e+15	-72.732351	-72.794975	-72.739336	-72.801992	41.405197
...	...	...	...	...	...	...	...	...	...
1082	2023-12-18T20:18:00+0000	-74.128571	40.326485	2.761060e+15	-74.095238	-74.155880	-74.101330	-74.161970	40.337756
1083	2023-12-18T20:18:00+0000	-74.189255	40.323994	2.565168e+15	-74.155880	-74.216782	-74.161970	-74.222843	40.335640
1084	2023-12-18T20:18:00+0000	-74.250313	40.322124						
1085	2023-12-18T20:18:00+0000	-74.311188	40.320011						
1086	2023-12-18T20:18:00+0000	-73.706215	40.320427						

23579 rows × 12 columns

- Congratulations, you just queried TEMPO data
- If you used the same data/bbox, you should have gotten 20k+ pixels.
- If you change the bounding box, you need to delete any downloaded files.



	Timestamp	LONGITUDE	LATITUDE	no2_vertical_column_troposphere	Longitude_SW	Longitude_SE	Longitude_NW	Longitude_NE	Latitude_SW	Latitude_SE	Lat
0	2023-12-18T13:11:00+0000	-73.261124	41.397552	9.600535e+14	-73.226740	-73.288809	-73.233614	-73.295652	41.408867	41.407360	
1	2023-12-18T13:11:00+0000	-72.579437	41.399281	9.477854e+14	-72.544632	-72.607487	-72.551695	-72.614491	41.411222	41.408752	
2	2023-12-18T13:11:00+0000	-72.642540	41.397034	1.068233e+15	-72.607487	-72.669970	-72.614491	-72.676945	41.408752	41.407031	
3	2023-12-18T13:11:00+0000	-72.704453	41.395725								
4	2023-12-18T13:11:00+0000	-72.767273	41.393372								
...	...	...	...								
1082	2023-12-18T20:18:00+0000	-74.128571	40.326485	2.761060e+15	-74.095238	-74.155880	-74.101330	-74.161970	40.337756	40.335640	
1083	2023-12-18T20:18:00+0000	-74.189255	40.323994	2.565168e+15	-74.155880	-74.216782	-74.161970	-74.222843	40.335640	40.333426	
1084	2023-12-18T20:18:00+0000	-74.250313	40.322124	3.123098e+15	-74.216782	-74.277761	-74.222843	-74.283768	40.333426	40.331434	
1085	2023-12-18T20:18:00+0000	-74.311188	40.320011	3.012529e+15	-74.277761	-74.338547	-74.283768	-74.344549	40.331434	40.329268	
1086	2023-12-18T20:18:00+0000	-73.706215	40.320427	4.572452e+15	-73.672052	-73.734129	-73.678310	-73.740360	40.331833	40.329838	

23579 rows × 13 columns

TEMPO_GeoXO_Meeting.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
[11] # Do it again, but cleanup the keys and add time object
     # Notice that the file is reused
     df = api.to_dataframe(tempokey, unit_keys=False, parse_dates=True, backend='xdr')
     df
```

- Units in the column headers can be great or tedious.
- You can disable them and add a pandas time column.

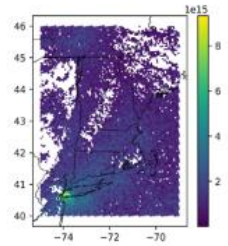
```
Using cached: ./tempo.l2.no2.vertical_column_troposphere_2023-12-18T000000Z_2023-12-18T235959Z.xdr.gz
```

	Timestamp	LONGITUDE	LATITUDE	no2_vertical_column_troposphere	Longitude_SW	Longitude_SE	Longitude_NW	Longitude_NE	Latitude_SW	Latitude_SE	Lat
0	2023-12-18T13:11:00+0000	-73.261124	41.397552	9.600535e+14	-73.226740	-73.288809	-73.233614	-73.295652	41.408867	41.407360	
1	2023-12-18T13:11:00+0000	-72.579437	41.399281	9.477854e+14	-72.544632	-72.607487	-72.551695	-72.614491	41.411222	41.408752	
2	2023-12-18T13:11:00+0000	-72.642540	41.397034	1.068233e+15	-72.607487	-72.669970	-72.614491	-72.676945	41.408752	41.407031	
3	2023-12-18T13:11:00+0000	-72.704453	41.395725								
4	2023-12-18T13:11:00+0000	-72.767273	41.393372								
...	...	...	...								
1082	2023-12-18T20:18:00+0000	-74.128571	40.326485	2.761060e+15	-74.09			970	40.337756	40.335640	
1083	2023-12-18T20:18:00+0000	-74.189255	40.323994	2.565168e+15	-74.15			843	40.335640	40.333426	
1084	2023-12-18T20:18:00+0000	-74.250313	40.322124	3.123098e+15	-74.21			768	40.333426	40.331434	
1085	2023-12-18T20:18:00+0000	-74.311188	40.320011	3.012529e+15	-74.27			549	40.331434	40.329268	
1086	2023-12-18T20:18:00+0000	-73.706215	40.320427	4.572452e+15	-73.67			360	40.331833	40.329838	

23579 rows × 13 columns

- LONGITUDE and LATITUDE are pixel centroids.
- SW, SE, NE, NW are corners.

Making shapefiles for ArcGIS is described in a tutorial.



GIS TropOMI Processing

Step 3: Compare to observations

- Make an hourly average product.
- Make a simple time-series plot
- Do the same with airnow to compare

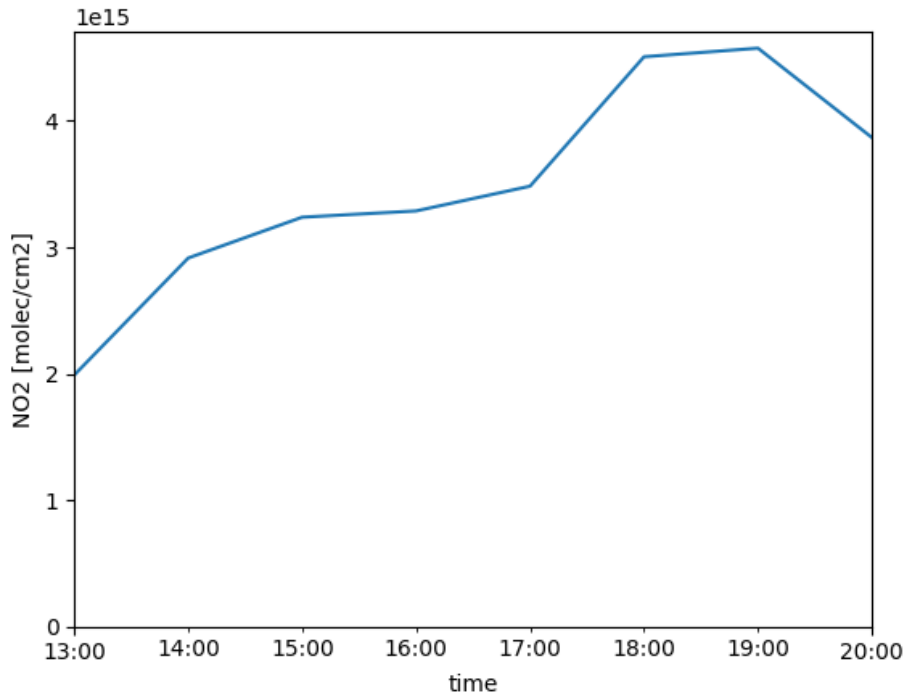
- Time averaging is easy
- Try another frequency

```
[ ]  # Make an hourly average
     hdf = df.groupby(pd.Grouper(key='time', freq='1h')).mean(numeric_only=True)
     hdf
```

time	LONGITUDE	LATITUDE	no2_vertical_column_troposphere	Longitude_SW	Longitude_SE	Longi
2023-12-18 13:00:00+00:00	-73.120932	40.853867	1.989354e+15	-73.086563	-73.148597	-7
2023-12-18 14:00:00+00:00	-73.212463	40.846213	2.914695e+15	-73.178142	-73.240112	-7
2023-12-18 15:00:00+00:00	-73.214949	40.854656	3.237612e+15	-73.180620	-73.242615	-7
2023-12-18 16:00:00+00:00	-73.147281	40.852352	3.287196e+15	-73.112882	-73.174985	-7
2023-12-18 17:00:00+00:00	-73.133614	40.856752	3.483012e+15	-73.099205	-73.161332	-7
2023-12-18 18:00:00+00:00	-73.153252	40.868111	4.506504e+15	-73.118870	-73.180939	-7
2023-12-18 19:00:00+00:00	-73.178307	40.830798	4.574339e+15	-73.143952	-73.205990	-7
2023-12-18 20:00:00+00:00	-73.140368	40.851870	3.867439e+15	-73.105964	-73.168078	-7

Next steps: Generate code with hdf | View recommended plots

```
# Plot a data column selected from the names above
tempocol = 'no2_vertical_column_troposphere'
ax = hdf[tempocol].plot(ylim=(0, None), ylabel='NO2 [molec/cm2]')
```



- plotting using pandas.
- pandas uses matplotlib.
- both are extremely well documented, so you can change anything you want.

```
airnowkey = 'airnow.no2'
adf = api.to_dataframe(airnowkey, unit_keys=False, parse_dates=True)
adf
```

	Timestamp	LONGITUDE	LATITUDE	STATION	no2	SITE_NAM	
0	2023-12-18T00:00:00-0000	-73.3369	41.1189	1	5.0	840090019003;42602	2023-12-18 00:00:00+00:00
1	2023-12-18T00:00:00-0000	-72.9027	41.3013	2	16.0	840090090027;42602	2023-12-18 00:00:00+00:00
2	2023-12-18T00:00:00-0000	-73.9661	40.8535	3	15.0	840340030010;42602	2023-12-18 00:00:00+00:00
3	2023-12-18T00:00:00-0000	-74.1261	40.6703	4	12.0	840340170006;42602	2023-12-18 00:00:00+00:00
4	2023-12-18T00:00:00-0000	-74.0666	40.7317	5	20.0	840340171002;42602	2023-12-18 00:00:00+00:00
...	...	...	...	...	...	...	
211	2023-12-18T23:00:00-0000	-74.0666	40.7317	5	18.0	840340171002;42602	20
212	2023-12-18T23:00:00-0000	-74.4294	40.4622	6	3.0	840340230011;42602	20
213	2023-12-18T23:00:00-0000	-74.6763	40.7876	7	2.0	840340273001;42602	20
214	2023-12-18T23:00:00-0000	-74.2084	40.6414	8	12.0	840340390004;42602	20
215	2023-12-18T23:00:00-0000	-73.1391	40.9610	9	4.0	840361030044;42602	20

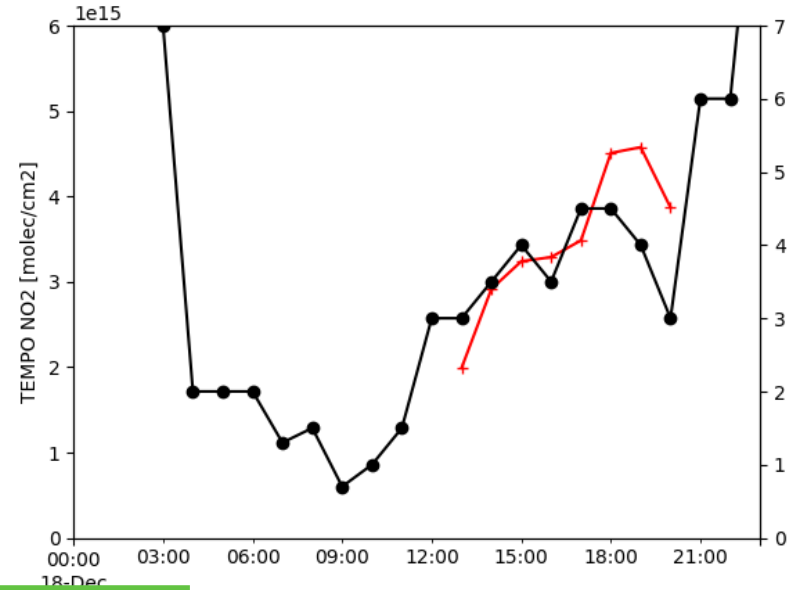
216 rows × 7 columns

Next steps: Generate code with adf | View recommended plots

- Look at descriptions for airnow or aqs.
- airnow is near-realtime; aqs is definitive.
- You should see no2, ozone, pm25, etc.

```
airnowno2 = adf['no2'].groupby(adf['time']).median()
ax = hdf[tempocol].plot(ylabel='TEMPO NO2 [molec/cm2]', color='r', marker='+', ylim=(0, 6e15))
airnowno2.plot(ax=ax.twinx(), color='k', marker='o', ylim=(0, 7), label='AirNow NO2 [ppb]')
```

```
<Axes: xlabel='time'>
```



For plotting, we plot AirNow on a secondary y-axis

Step 4: Create a TEMPO custom map

- Here we will request similar data, but pregridded.
- This is a custom L3 file on a CMAQ 12km grid.

```
[ ]  api.grid_kw
```

```
{'GDNAM': '12US1',
 'GDTYP': 2,
 'NCOLS': 21,
 'NROWS': 17,
 'XORIG': 1848000.0,
 'YORIG': 252000.0,
 'XCELL': 12000.0,
 'YCELL': 12000.0,
 'P_ALP': 33.0,
 'P_BET': 45.0,
 'P_GAM': -97.0,
 'XCENT': -97.0,
 'YCENT': 40.0,
 'VGTYP': 7,
 'VGTOP': 5000.0,
 'NLAYS': 35,
 'earth_radius': 6370000.0,
 'g': 9.81,
 'R': 287.04,
 'A': 50.0,
 'T0': 290,
 'P0': 100000.0,
 'REGRID_AGGREGATE': 'None'}
```

FYI – earlier, gridfit=True told pyrsig use only cells in our bbox.

```
# Now retrieve a NetCDF file with IOAPI coordinates (like CMAQ)
ds = api.to_ioapi(tempokey)
ds
```

xarray.Dataset

▸ Dimensions: (**TSTEP**: 24, VAR: 4, DATE-TIME: 2, **LAY**: 1, **ROW**: 17, **COL**: 21)

▾ Coordinates:

TSTEP	(TSTEP)	datetime64[ns]	2023-12-18 ... 2023-12-18T23:0...
LAY	(LAY)	float32	0.9975
ROW	(ROW)	float64	0.5 1.5 2.5 3.5 ... 14.5 15.5 16.5
COL	(COL)	float64	0.5 1.5 2.5 3.5 ... 18.5 19.5 20.5

▾ Data variables:

TFLAG	(TSTEP, VAR, DATE-TIME)	int32	...
LONGITUDE	(TSTEP, LAY, ROW, COL)	float32	...
LATITUDE	(TSTEP, LAY, ROW, COL)	float32	...
COUNT	(TSTEP, LAY, ROW, COL)	int32	...
NO2_VERTICA...	(TSTEP, LAY, ROW, COL)	float32	...

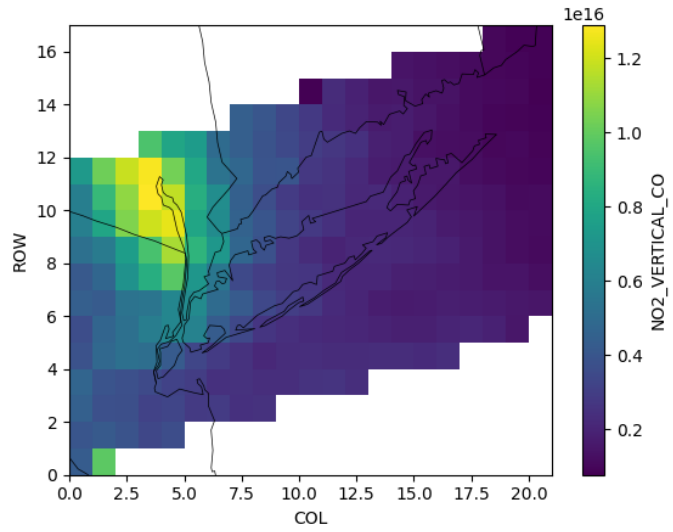
▸ Indexes: (4)

▸ Attributes: (34)

```
# Choose a column from above, notice that names are truncated, so they can be weird
tempoikey = 'NO2_VERTICAL_CO'
```

```
# Now plot a map
cno = pycno.cno(ds.crs_proj4)
qm = ds[tempoikey].where(lambda x: x>0).mean(('TSTEP', 'LAY')).plot()
cno.drawstates(resnum=1)
```

```
<matplotlib.collections.LineCollection at 0x7ce362e9eaa0>
```



Step 5: Make a surface NO2 map

- Download CMAQ EQUATES surface and
- Extract CMAQ to gridded TEMPO
- Calculate a transfer function [^1]
- Estimate surface NO2
- Average and make a plot

[1] No warranty expressed or implied!

```
[18] # Get a column and surface estimate f
     cmaqcolkey = 'cmaq.equates.conus.inte
     qids = api.to_ioapi(cmaqcolkey, bdate=
     cmaqsfckey = 'cmaq.equates.conus.acon
     qsds = api.to_ioapi(cmaqsfckey, bdate=
```

- Use your CMAQ files:
- qids = pyrsig.open_ioapi(colpath)
- qsds = pyrsig.open_ioapi(concpath)

Align Grids

To align the grids, we have to convert between lambert projections. This is a little complicated, but pyrsig gives you all the tools you need.

1. get 2d x/y for TEMPO L3 cell centroids
2. get 2d x/y for TEMPO L3 cell centroids on CMAQ grid
3. store for later use
4. pretend the EQUATES data is 2023

```
[19] # 1. get 2d x/y for TEMPO L3 cell
     y, x = xr.broadcast(ds.ROW, ds.COL
     # 2. get 2d x/y for TEMPO L3 cell
     dstproj = pyproj.Proj(ds.crs_proj4
     srcproj = pyproj.Proj(qids.crs_pro
     X, Y = srcproj(*dstproj(x.values,
     # 3. store the result for later us
     ds['CMAQX'] = ('COL',), X.mean(0)
     ds['CMAQY'] = ('ROW',), Y.mean(1)
     # 4. here we pretend that the CMAQ
     ds['CMAQT'] = ('TSTEP',), qsds.TST
```

Extract CMAQ to TEMPO custom L3

- We'll extract data using the CMAQ coordinates
- And, add the data to the TEMPO dataset

```
[24] # Now we extract the CMAQ at the TEMPO locations
     # all extractions will output time, y, x data
     dims = ('TSTEP', 'ROW', 'COL')
     # all extractions use the same coordinates
     selopts = dict(TSTEP=ds['CMAQT'], COL=ds['CMAQX'], ROW=ds['CMAQY'], method='nearest')
     # 1 atm is the surface
     selopts['LAY'] = 1
     # Get CMAQ surface NO2 (NO2), and tropospheric column (NO2_COLUMN)
     ds['CMAQ_NO2_SFC'] = dims, qsds['NO2'].sel(**selopts).data, {'units': 'ppb'}
     ds['CMAQ_NO2_COL'] = dims, qids['NO2_COLUMN'].sel(**selopts).data * 1e15, {'units': 'm
```

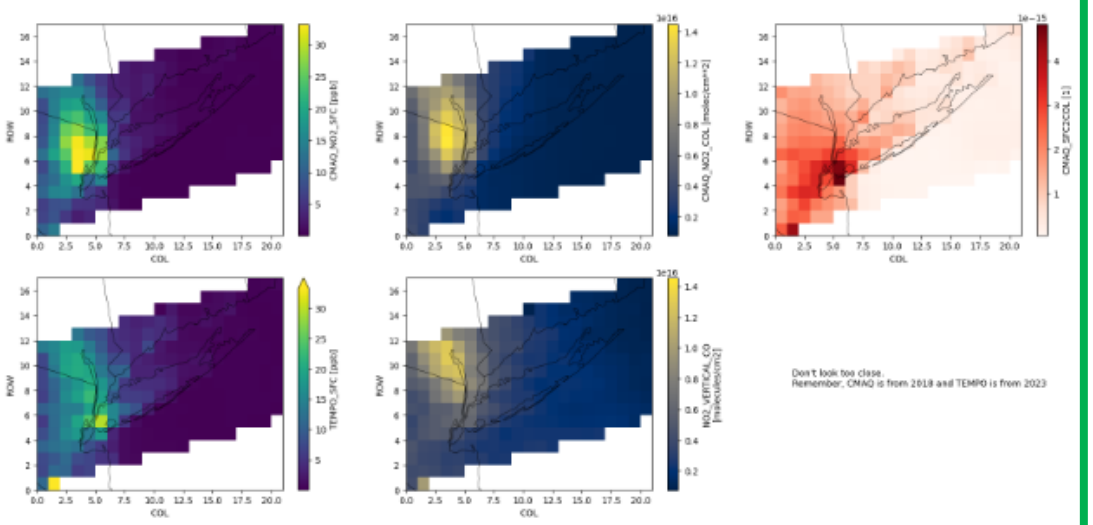
```
[25] # Calculate the transfer function
     ds['CMAQ_SFC2COL'] = ds['CMAQ_NO2_SFC'] / ds['CMAQ_NO2_COL']
     ds['CMAQ_SFC2COL'].attrs.update(units='1')
     # Calculate the estimate surface NO2
     ds['TEMPO_SFC'] = ds['NO2_VERTICAL_CO'] * ds['CMAQ_SFC2COL']
     ds['TEMPO_SFC'].attrs.update(units='ppb')
```

Now Plot TEMPO-based Surface NO2

```
# Now plot the time average
pds = ds.where(ds['NO2_VERTICAL_CO'] > 0).mean(('TSTEP', 'LAY'), keep_attrs=True)

# Controlling figure canvas use
gskw = dict(left=0.05, right=0.95, bottom=0.05, top=0.95)
fig, axx = plt.subplots(2, 3, figsize=(18, 8), gridspec_kw=gskw)

# Put CMAQ on top row : columns 0, 1, and 2
qmsfc = pds['CMAQ_NO2_SFC'].plot(ax=axx[0, 0], cmap='viridis')
qmcol = pds['CMAQ_NO2_COL'].plot(ax=axx[0, 1], cmap='cividis')
pds['CMAQ_SFC2COL'].plot(ax=axx[0, 2], cmap='Reds')
# Put TEMPO on bottom row and use the same colorscales as CMAQ
pds['TEMPO_SFC'].plot(ax=axx[1, 0], norm=qmsfc.norm, cmap=qmsfc.cmap)
pds['NO2_VERTICAL_CO'].plot(ax=axx[1, 1], norm=qmcol.norm, cmap=qmcol.cmap)
# add state overlays (alternatively)
cno.drawstates(ax=axx, resnum=1)
# hide the unused axes
axx[1, 2].set(visible=False)
# add a reminder
_ = fig.text(0.7, 0.25, 'Don\'t look too close.\nRemember, CMAQ is from 2018 and TEMPO is
```



- This is a simplistic example:
  - assumes that the CMAQ ratio of surface to column is “true”.
  - We used a different year single day…
  - I chose that day because the winds were in the same general direction…
  - CMAQ NO2_COLUMN and TEMPO do not share an averaging kernel…
- ***This is just an example of the mechanics***

Optional Exercises

- Change beginning date (bdate) to another date in the mini-public release.
- Extend the date range (?api.to_datafame)
- Change the bounding box
  - Warning: by default, files are reused that match the original time.
  - you must delete downloaded files, or
  - Change workdir='location_name’
- Change the comparison between AirNow and TEMPO:
  - TropOMI and TEMPO, or
  - Pandora and TEMPO
- Try extracting formaldehyde or ozone

Questions?

pyrsig: henderson.barron@epa.gov

RSIG: rsig@epa.gov