Reading multidimensional data using open geo tools

Reading multidimensional science data in NetCDF using open geo toos

This notebook reads aerosol index and Tropospheric NO_2 Concentration from Sentinel-5P TROPOMI data using Opengeo Tools

Tutorial data and code are from NASA ARSET program: https://appliedsciences.nasa.gov/join-mission/training/english/high-resolution-no2-monitoring-space-tropomi (https://appliedsciences.nasa.gov/join-mission/training/english/high-resolution-no2-monitoring-space-tropomi)

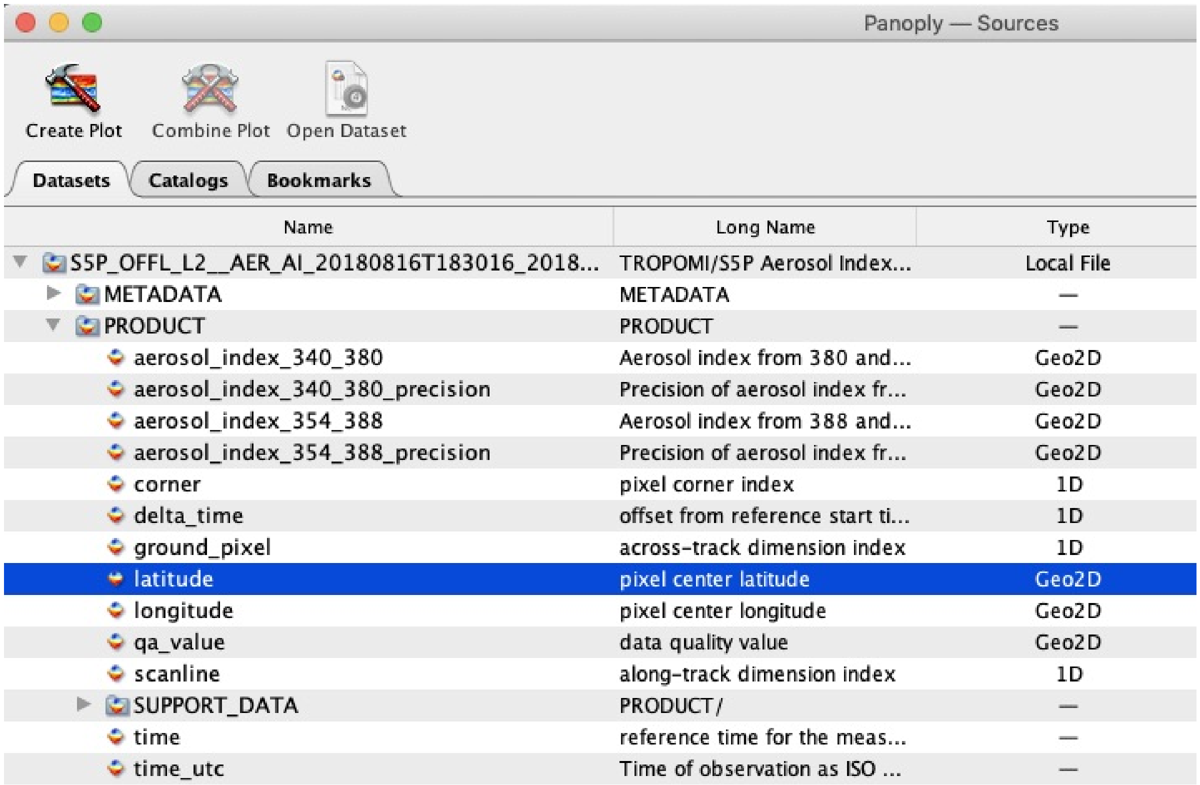
In [1]:
```
import numpy as np
# from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import sys
from netCDF4 import Dataset
from pprint import pprint, pp
import pandas as pd
```

In [2]:
```
ls TROPOMI_PythonCodesAndData/
```

```
S5P_OFFL_L2__AER_AI_20180816T183016_20180816T201146_04361_01_010100_20180822T174822.nc*
S5P_OFFL_L2__CO_____20180816T183016_20180816T201146_04361_01_010100_20180822T174815.nc*
S5P_RPRO_L2__CH4____20180816T182917_20180816T201245_04361_01_010202_20190101T194705.nc*
fileList.txt*
read_and_map_tropomi_no2_ai.py*
read_tropomi_and_list_sds.py*
read_tropomi_no2_ai_and_dump_ascii.py*
read_tropomi_no2_ai_at_a_location.py*
```

Explore NetCDF file for its contents

The NetCDF file is like a folder with multiple sub-folders and files within it. Folders are called as `groups` and files within it are called as `variables`. NASA supplies a cross-platform app called Panoply (https://www.giss.nasa.gov/tools/panoply/) which gives you a UI to query and visualize NetCDF files. Below is a screenshot of Panoply reading the Aerosol Index file.



Screenshot of Panoply software with Aerosol Index NetCDF file opened.

The first step is to read this file as a `netCDF4.Dataset` class.

In [3]:
```
file_path = 'TROPOMI_PythonCodesAndData/S5P_OFFL_L2__AER_AI_20180816T183016_20180816T201146_04361_01_010100_20180822T174822.nc'
ds = Dataset(file_path, mode='r')
type(ds)
```

Out[3]:
```
netCDF4._netCDF4.Dataset
```

In [4]:
```
ds.groups.keys()
```

Out[4]:
```
dict_keys(['PRODUCT', 'METADATA'])
```

Explore the different variables as a DataFrame

In [5]:

```python
v = {'variables':[], 'long_name':[], 'units':[]}
for var in list(ds.groups['PRODUCT'].variables.keys()):
    v['variables'].append(ds.groups['PRODUCT'].variables[var].name)
    v['long_name'].append(ds.groups['PRODUCT'].variables[var].long_name)
    try:
        v['units'].append(ds.groups['PRODUCT'].variables[var].units)
    except:
        v['units'].append(None)

vars_df = pd.DataFrame.from_dict(v)
vars_df
```

Out[5]:

	variables	long_name	units
0	scanline	along-track dimension index	1
1	ground_pixel	across-track dimension index	1
2	time	reference time for the measurements	seconds since 2010-01-01 00:00:00
3	corner	pixel corner index	1
4	latitude	pixel center latitude	degrees_north
5	longitude	pixel center longitude	degrees_east
6	delta_time	offset from reference start time of measurement	milliseconds
7	time_utc	Time of observation as ISO 8601 date-time string	None
8	qa_value	data quality value	1
9	aerosol_index_354_388	Aerosol index from 388 and 354 nm	1
10	aerosol_index_340_380	Aerosol index from 380 and 340 nm	1
11	aerosol_index_354_388_precision	Precision of aerosol index from 388 and 354 nm	1
12	aerosol_index_340_380_precision	Precision of aerosol index from 380 and 340 nm	1

Read Aerosol Index 354 - 388 nm into memory

In [6]:
```
# preview contents of the variable
ds.groups['PRODUCT'].variables['aerosol_index_354_388']
```

Out[6]:
```
<class 'netCDF4._netCDF4.Variable'>
float32 aerosol_index_354_388(time, scanline, ground_pixel)
    units: 1
    proposed_standard_name: ultraviolet_aerosol_index
    comment: Aerosol index from 388 and 354 nm
    long_name: Aerosol index from 388 and 354 nm
    radiation_wavelength: [354. 388.]
    coordinates: longitude latitude
    ancillary_variables: aerosol_index_354_388_precision
    _FillValue: 9.96921e+36
path = /PRODUCT
unlimited dimensions:
current shape = (1, 3245, 450)
filling on
```

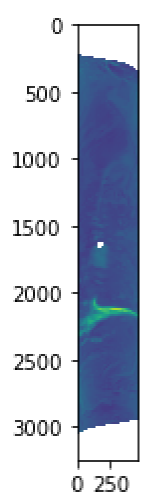
In [7]:
```
ai_data = ds.groups['PRODUCT'].variables['aerosol_index_354_388'][:]
type(ai_data)
```

Out[7]:
```
numpy.ma.core.MaskedArray
```

In [8]:
```
ai_data.shape
```

Out[8]:
```
(1, 3245, 450)
```

In [9]:
```
plt.imshow(ai_data[0]);
```



Reading using `xarray`

See https://github.com/acgeospatial/Sentinel-5P/blob/master/sentinel5p_xarray_blog.ipynb (https://github.com/acgeospatial/Sentinel-5P/blob/master/sentinel5p_xarray_blog.ipynb)

In [28]:
```
import xarray
```

In [31]:
```
xr_data = xarray.open_dataset(file_path, group='PRODUCT',
                              engine='netcdf4', decode_coords=True)
type(xr_data)
```

Out[31]:
```
xarray.core.dataset.Dataset
```

In [32]:
```
xr_data
```

Out[32]:

xarray.Dataset

▶ Dimensions: (**corner**: 4, **ground_pixel**: 450, **scanline**: 3245, **time**: 1)

▼ Coordinates:

scanline	(scanline)	float64	0.0 1.0 2…
ground_pixel	(ground_pixel)	float64	0.0 1.0 2…
time	(time)	datetime64[ns]	2018-08…
corner	(corner)	float64	0.0 1.0 2…
latitude	(time, scanline, ground_pixel)	float32	…
longitude	(time, scanline, ground_pixel)	float32	…

▼ Data variables:

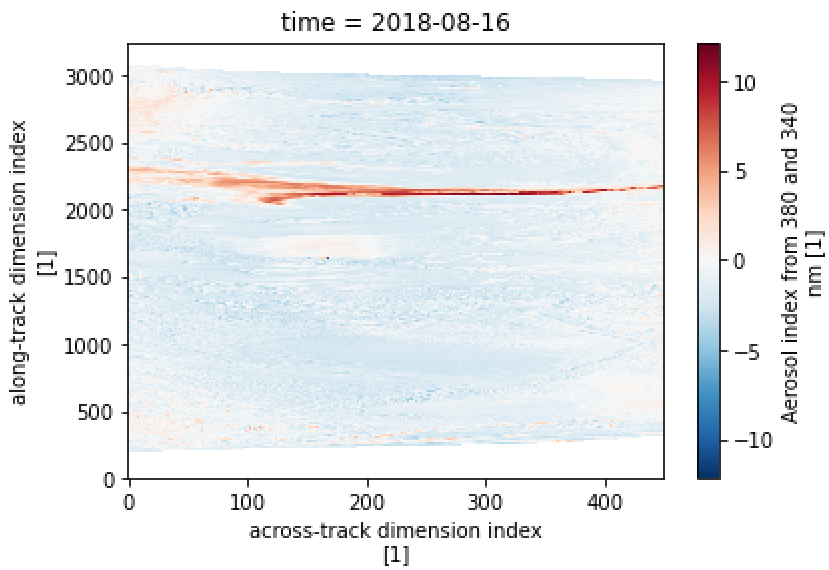
delta_time	(time, scanline)	timedelta64[ns]	…
time_utc	(time, scanline)	object	…
qa_value	(time, scanline, ground_pixel)	float32	…
aerosol_index_…	(time, scanline, ground_pixel)	float32	…
aerosol_index_…	(time, scanline, ground_pixel)	float32	…
aerosol_index_…	(time, scanline, ground_pixel)	float32	…
aerosol_index_…	(time, scanline, ground_pixel)	float32	…

▶ Attributes: (0)

In [33]:
```
xr_data_ai = xr_data['aerosol_index_340_380']
print(type(xr_data_ai))
print(xr_data_ai.shape)
```

```
<class 'xarray.core.dataarray.DataArray'>
(1, 3245, 450)
```

In [34]:
```
xr_data_ai[0].plot();
```

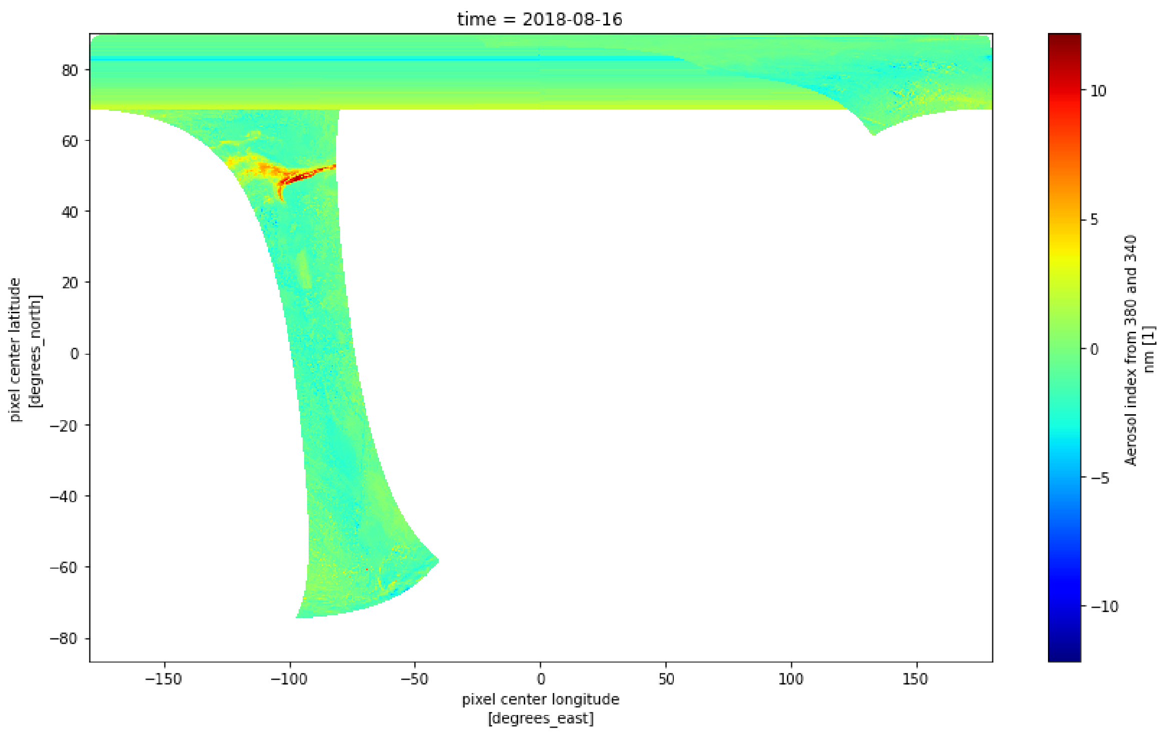


In [37]:
```
(xr_data.latitude.attrs, xr_data.longitude.attrs)
```

Out[37]:
```
({'long_name': 'pixel center latitude',
  'units': 'degrees_north',
  'standard_name': 'latitude',
  'valid_min': -90.0,
  'valid_max': 90.0,
  'bounds': '/PRODUCT/SUPPORT_DATA/GEOLOCATIONS/latitude_bounds'},
 {'long_name': 'pixel center longitude',
  'units': 'degrees_east',
  'standard_name': 'longitude',
  'valid_min': -180.0,
  'valid_max': 180.0,
  'bounds': '/PRODUCT/SUPPORT_DATA/GEOLOCATIONS/longitude_bounds'})
```

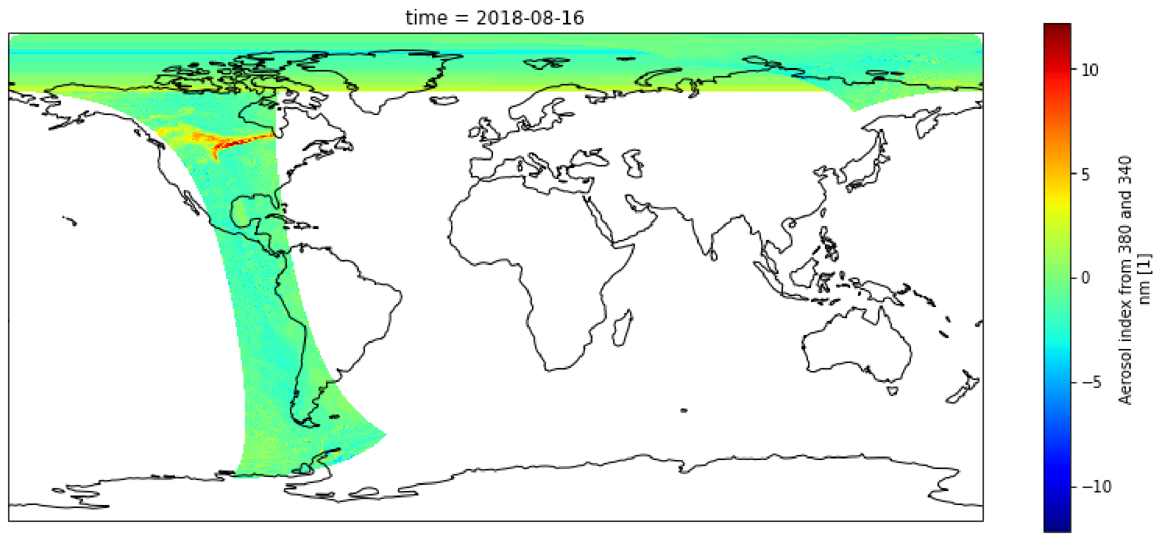
Plot using matplotlib

In [42]:
```
plt.figure(figsize=(14,8))
ax = plt.axes()

xr_data.aerosol_index_340_380[0].plot.pcolormesh(ax=ax, x='longitude',
                                                  y='latitude',
                                                  add_colorbar=True, c
map='jet');
```

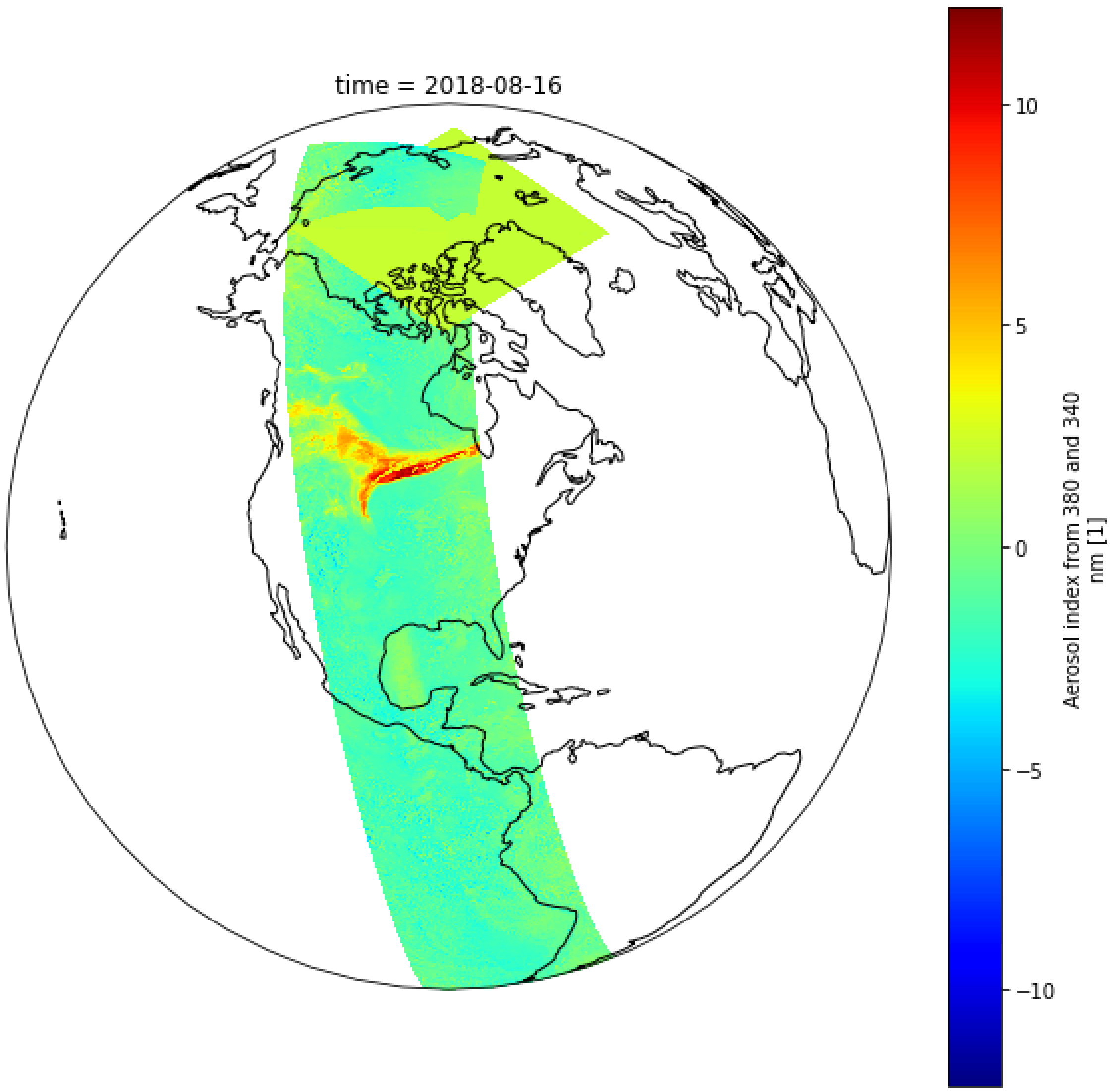


Plotting using `cartopy`

In [44]:
```
import cartopy.crs as ccrs
plt.figure(figsize=(14,6))
ax = plt.axes(projection = ccrs.PlateCarree())

xr_data.aerosol_index_340_380[0].plot.pcolormesh(ax=ax, x='longitude',
                                                  y='latitude',
                                                  add_colorbar=True, c
map='jet')

ax.set_global()
ax.coastlines();
```



In [47]:
```
plt.figure(figsize=(10,10))
ax = plt.axes(projection = ccrs.Orthographic(-88,40))

xr_data.aerosol_index_340_380[0].plot.pcolormesh(ax=ax, x='longitude',
                                                  y='latitude',
                                                  add_colorbar=True, c
map='jet',
                                                  transform=ccrs.PlateCarr
ee())

ax.set_global()
ax.coastlines();
```



In [51]:
```
xr_data_rio = xr_data_ai.rio
type(xr_data_rio)
```

Out[51]: `rioxarray.rioxarray.RasterArray`

In [55]:
```
xr_data.aerosol_index_340_380.rio.to_raster('xr_test.tif')
```

Reading using `rioxarray`

In [60]:
```
import rioxarray
import warnings; warnings.simplefilter('ignore')
```

In [85]:
```
rds = rioxarray.open_rasterio(filename = file_path, parse_coordinates=True,
                                )
type(rds)
```

Out[85]: `list`

In [87]:
```
rds[0]
```

Out[87]:

xarray.Dataset

▶ Dimensions: (**band**: 1, **time**: 1, **x**: 450, **y**: 3245)

▼ Coordinates:

y	(y)	float64	3.244e+03 3.243e+03 … 1.0 0.0
x	(x)	float64	0.0 1.0 2.0 … 447.0 448.0 449.0
time	(time)	int64	272073600
spatial_ref	()	int64	0
band	(band)	int64	1

▶ Data variables: (41)

▶ Attributes: (287)

In [97]:
```
r1 = rds[0]
type(r1)
```

Out[97]: `xarray.core.dataset.Dataset`

In [99]:
```
r1.geolocation_flags
```

Out[99]:

xarray.DataArray 'geolocation_flags' (**band**: 1, **y**: 3245, **x**: 450)

```
array([[[12, 12, ..., 12, 12],
        [12, 12, ..., 12, 12],
        ...,
        [ 8,  8, ...,  8,  8],
        [ 8,  8, ...,  8,  8]]], dtype=uint8)
```

▼ Coordinates:

y	(y)	float64	3.244e+03 3.243e+03 … 1.0 0.0
x	(x)	float64	0.0 1.0 2.0 … 447.0 448.0 449.0
spatial_ref	()	int64	0
band	(band)	int64	1

► Attributes: (13)

In [101]:
```
r1.spatial_ref
```

Out[101]:

xarray.DataArray 'spatial_ref'

```
array(0)
```

▼ Coordinates:

spatial_ref	()	int64	0

▼ Attributes:

GeoTransform : -0.5 1.0 0.0 3244.5 0.0 -1.0

In [102]:
```
r1.spatial_resolution
```

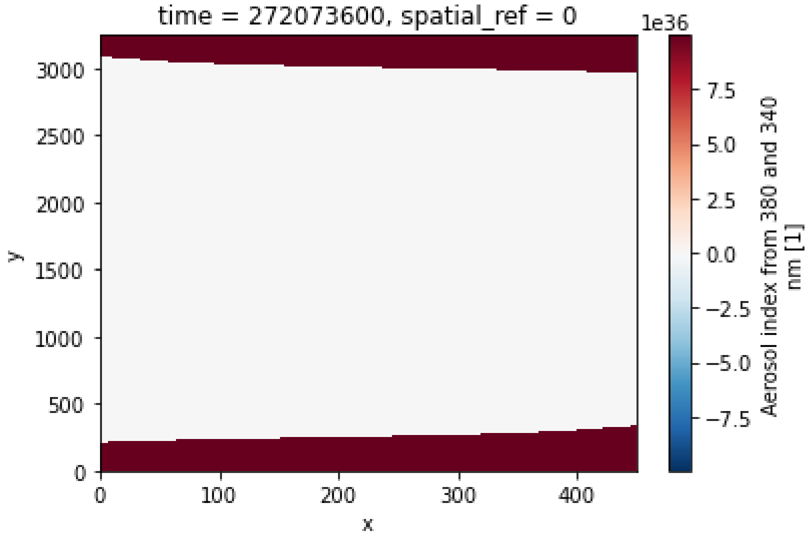
Out[102]:
```
'7x3.5km2'
```

In [103]:
```
(r1.geospatial_lat_max, r1.geospatial_lat_min,
 r1.geospatial_lon_max, r1.geospatial_lon_min)
```

Out[103]:
```
(89.972939, -86.81926, -179.99773, 179.99924)
```

In [112]:
```
r1_ai = r1['aerosol_index_340_380']
r1_ai.plot()
```

Out[112]: <matplotlib.collections.QuadMesh at 0x1965f1520>



In [113]:
```
r1_ai.spatial_ref
```

Out[113]:

xarray.DataArray 'spatial_ref'

array(0)

▼ Coordinates:

spatial_ref () int64 0

▼ Attributes:

GeoTransform : -0.5 1.0 0.0 3244.5 0.0 -1.0

In [202]:
```
# rds[0].rio.set_spatial_dims(x_dim='/PRODUCT/longitude', y_dim='/PRODUCT/latitude')
rds_crs_set = rds[0].rio.set_crs(4326)
rds_crs_set
```

Out[202]:

xarray.Dataset

▶ Dimensions: (**band**: 1, **time**: 1, **x**: 450, **y**: 3245)

▼ Coordinates:

y	(y)	float64	3.244e+03 3.243e+03 … 1.0 0.0
x	(x)	float64	0.0 1.0 2.0 … 447.0 448.0 449.0
time	(time)	int64	272073600
spatial_ref	()	int64	0
band	(band)	int64	1

▶ Data variables: (41)

▶ Attributes: (287)

In [260]:
```
rds_crs_set = rds_crs_set.set_coords(['longitude','latitude'])
rds_crs_set
```

Out[260]:

xarray.Dataset

▶ Dimensions: (**band**: 1, **time**: 1, **x**: 450, **y**: 3245)

▼ Coordinates:

y	(y)	float64	3.244e+03 3.243e+03 … 1.0 0.0
x	(x)	float64	0.0 1.0 2.0 … 447.0 448.0 449.0
time	(time)	int64	272073600
spatial_ref	()	int64	0
latitude	(time, y, x)	float32	53.289627 53.328514 … -68.705986
band	(band)	int64	1
longitude	(time, y, x)	float32	119.45596 119.32921 … 1.9235005

▶ Data variables: (39)

▶ Attributes: (287)

In [206]:
```
rds_crs_set = rds_crs_set.rio.set_crs(4326)
rds_crs_set.rio.crs
```

Out[206]:
```
CRS.from_epsg(4326)
```

In [215]:
```
rds_crs_set = rds_crs_set.rio.write_coordinate_system()
```

In [217]:
```
r1_ai_crs_set = rds_crs_set['aerosol_index_340_380']
r1_ai_crs_set
```

Out[217]:

xarray.DataArray 'aerosol_index_340_380' (**time**: 1, **y**: 3245, **x**: 450)

```
array([[[9.96921e+36, 9.96921e+36, 9.96921e+36, ..., 9.96921e+36,
         9.96921e+36, 9.96921e+36],
        [9.96921e+36, 9.96921e+36, 9.96921e+36, ..., 9.96921e+36,
         9.96921e+36, 9.96921e+36],
        [9.96921e+36, 9.96921e+36, 9.96921e+36, ..., 9.96921e+36,
         9.96921e+36, 9.96921e+36],
        ...,
        [9.96921e+36, 9.96921e+36, 9.96921e+36, ..., 9.96921e+36,
         9.96921e+36, 9.96921e+36],
        [9.96921e+36, 9.96921e+36, 9.96921e+36, ..., 9.96921e+36,
         9.96921e+36, 9.96921e+36],
        [9.96921e+36, 9.96921e+36, 9.96921e+36, ..., 9.96921e+36,
         9.96921e+36, 9.96921e+36]]], dtype=float32)
```

▼ Coordinates:

y	(y)	float64	3.244e+03 3.243e+03 … 1.0 0.0
x	(x)	float64	0.0 1.0 2.0 … 447.0 448.0 449.0
time	(time)	int64	272073600
spatial_ref	()	int64	0
latitude	(time, y, x)	float32	53.289627 53.328514 … -68.705986
longitude	(time, y, x)	float32	119.45596 119.32921 … 1.9235005

► Attributes: (11)

In [221]:
```
r1_ai_crs_set = r1_ai_crs_set.rio.set_crs(4326, True)
```

In [223]:
```
r1_ai_crs_set = r1_ai_crs_set.rio.write_coordinate_system()
```

In [249]:
```
r1_ai_crs_set.longitude.attrs
```

Out[249]:
```
{'bounds': '/PRODUCT/SUPPORT_DATA/GEOLOCATIONS/longitude_bounds',
 'long_name': 'pixel center longitude',
 'standard_name': 'longitude',
 'units': 'degrees_east',
 'valid_max': 180,
 'valid_min': -180,
 '_FillValue': 9.969209968386869e+36,
 'scale_factor': 1.0,
 'add_offset': 0.0,
 'grid_mapping': 'spatial_ref'}
```

In [258]: r1_ai_crs_set.latitude[0][0]

Out[258]: xarray.DataArray 'latitude' (**x**: 450)

```
array([53.289627, 53.328514, 53.366734, 53.404312, 53.44126 , 53.477608,
       53.51336 , 53.54854 , 53.58316 , 53.61724 , 53.65079 , 53.68383 ,
       53.716366, 53.748417, 53.77999 , 53.811104, 53.841766, 53.87199 ,
       53.901783, 53.93116 , 53.96013 , 53.9887  , 54.03083 , 54.085697,
       54.13913 , 54.1912  , 54.24196 , 54.29147 , 54.339783, 54.386948,
       54.433014, 54.478027, 54.522026, 54.565052, 54.607143, 54.648335,
       54.68866 , 54.728153, 54.76684 , 54.80476 , 54.84193 , 54.87838 ,
       54.91413 , 54.949215, 54.983654, 55.01746 , 55.050663, 55.083282,
       55.115334, 55.146835, 55.177803, 55.20826 , 55.238216, 55.26769 ,
       55.29669 , 55.32524 , 55.353348, 55.381027, 55.40829 , 55.435154,
       55.46162 , 55.48771 , 55.513424, 55.53878 , 55.563786, 55.58845 ,
       55.612785, 55.636795, 55.660496, 55.683887, 55.70698 , 55.729786,
       55.752308, 55.774555, 55.79653 , 55.81825 , 55.839714, 55.860928,
       55.881897, 55.902634, 55.923138, 55.943417, 55.963474, 55.98332 ,
       56.002953, 56.022385, 56.041615, 56.06065 , 56.079494, 56.098152,
       56.116627, 56.134926, 56.15305 , 56.171   , 56.18879 , 56.206413,
       56.223877, 56.241188, 56.258347, 56.275352, 56.292213, 56.308933,
       56.325516, 56.341957, 56.358265, 56.374443, 56.39049 , 56.406418,
       56.42222 , 56.437897, 56.45346 , 56.468906, 56.484238, 56.49946 ,
       56.514572, 56.52958 , 56.54448 , 56.559277, 56.573975, 56.588577,
...
       59.069664, 59.082832, 59.09606 , 59.109356, 59.12271 , 59.136127,
       59.149612, 59.163162, 59.17678 , 59.190468, 59.20422 , 59.218044,
       59.231937, 59.2459  , 59.259937, 59.274048, 59.28823 , 59.30249 ,
       59.316826, 59.331238, 59.345726, 59.360294, 59.37494 , 59.389668,
       59.40448 , 59.419373, 59.434345, 59.449406, 59.464554, 59.479782,
       59.495102, 59.51051 , 59.52601 , 59.541595, 59.557274, 59.573044,
       59.588905, 59.604862, 59.620914, 59.637066, 59.65331 , 59.66965 ,
       59.686092, 59.702637, 59.719276, 59.73602 , 59.752865, 59.769814,
       59.786865, 59.80402 , 59.82128 , 59.83865 , 59.856125, 59.873703,
       59.891396, 59.90919 , 59.9271  , 59.945114, 59.96324 , 59.98148 ,
       59.999825, 60.018284, 60.036854, 60.05553 , 60.07432 , 60.093224,
       60.112236, 60.13136 , 60.15059 , 60.16993 , 60.189377, 60.20893 ,
       60.228592, 60.24836 , 60.268227, 60.288197, 60.308266, 60.32843 ,
       60.348686, 60.369038, 60.389473, 60.409996, 60.430595, 60.451267,
       60.47201 , 60.49282 , 60.513687, 60.534603, 60.555557, 60.57655 ,
       60.597565, 60.61859 , 60.63962 , 60.66064 , 60.676384, 60.68687 ,
       60.697346, 60.70781 , 60.718254, 60.728683, 60.739086, 60.74947 ,
       60.759827, 60.770153, 60.780445, 60.790703, 60.800922, 60.8111  ,
       60.821228, 60.831303, 60.841328, 60.851288, 60.861183, 60.871014],
      dtype=float32)
```

▼ Coordinates:

y	()	float64	3.244e+03
x	(x)	float64	0.0 1.0 2.0 … 447.0 448.0 449.0
time	()	int64	272073600
spatial_ref	()	int64	0
latitude	(x)	float32	53.289627 53.328514 … 60.871014

longitude (x) float32 119.45596 119.32921 … 75.9475

▶ Attributes: (10)

In [250]:
```
r1_ai_crs_set.rio.to_raster('ai_crs_try1_rio.tif')
```

Reading using rasterio

In [114]:
```
import rasterio
```

In [172]:
```
base_file_handle = rasterio.open(file_path)
base_file_handle.subdatasets[:5]
```

Out[172]:
```
['netcdf:TROPOMI_PythonCodesAndData/S5P_OFFL_L2__AER_AI_20180816T183016_20180816T20
 'netcdf:TROPOMI_PythonCodesAndData/S5P_OFFL_L2__AER_AI_20180816T183016_20180816T20
 'netcdf:TROPOMI_PythonCodesAndData/S5P_OFFL_L2__AER_AI_20180816T183016_20180816T20
 'netcdf:TROPOMI_PythonCodesAndData/S5P_OFFL_L2__AER_AI_20180816T183016_20180816T20
 'netcdf:TROPOMI_PythonCodesAndData/S5P_OFFL_L2__AER_AI_20180816T183016_20180816T20
```

In [173]:
```
r2 = rasterio.open('netcdf:./'+file_path+":/PRODUCT/aerosol_index_340_380")
type(r2)
```

Out[173]:
```
rasterio.io.DatasetReader
```

In [174]:
```
ai_data = r2.read()
type(ai_data)
```

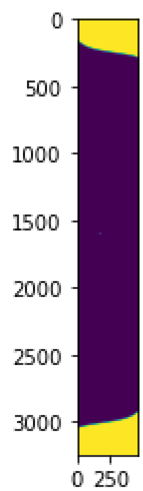
Out[174]:
```
numpy.ndarray
```

In [175]:
```
ai_data.shape
```

Out[175]:
```
(1, 3245, 450)
```

In [176]:
```
plt.imshow(ai_data[0])
```

Out[176]:
```
<matplotlib.image.AxesImage at 0x196e8b8b0>
```



Convert to GeoTIFF using GDAL

In [3]:
```
!gdal_translate NETCDF:"TROPOMI_PythonCodesAndData/S5P_OFFL_L2__AER_AI_20180816T183016_20180816T201146_04361_01_010100_20180822T174822.nc":/PRODUCT/aerosol_index_340_380 try1.tif
```

```
Input file size is 450, 3245
Warning 1: Metadata exceeding 32000 bytes cannot be written into GeoTIFF. Transfer
0...10...20...30...40...50...60...70...80...90...100 - done.
```

In [ ]: