Monitoring the Saharan Air Layer using Satellite Applications

Rebekah Esmaili, rebekah@stcnet.com
Science and Technology Corportation

29 September 2019

launch binder

This tutorial is interactive, click to launch code!

1 What is the Saharan Air Layer?

The Saharan Air Layer (SAL) is a warm and dry pocket of air that originates over the Saharan Desert and propagates over the Atlantic. The SAL extends from 850-500mb and has very steep lapse rates, thus capping the moist, tropical marine air below. This can potentially suppress tropical cyclone formation. The SAL is a potential hazard because it can transport dust to populated regions across the Atlantic, reducing air quality over the Caribbean and eastern United States. This can increase allergies and asthma in sensitive populations.

Figure 1 shows a SAL outflow on September 15, 2018 using the VIIRS instrument from the Suomi NPP satellite. This SAL event reached the Caribbean on September 20, 2018. The SAL can be detected from space by examining visible and infrared imagery from a variety of sensors on satellite platform, such as the ABI (GOES-16), SEVIRI (Meteosat-9/-10), VIIRS (Suomi NPP, NOAA-20), and AVHRR (MetOp-A, -B, -C). The horizontal extent of the SAL can be monitored using dust RGB imagery, channel differencing (e.g., 10.33 μm – 12.30 μm), and Aerosol Optical Depth (AOD) retrievals. Additionally, microwave and infrared sounding (CrIS, ATMS, AMSU, and IASI) are particularly useful for SAL monitoring because sounder products can detect both the horizontal and vertical distribution of the dry air mass.

2 What is NUCAPS?

The NOAA Unique Combined Atmospheric Processing System (NUCAPS) operationally produces atmospheric sounding products from the Suomi National-Polar-orbiting Partnership (Suomi NPP) and NOAA-20 polar orbiting satellites. From each satellite, NUCAPS provides global, twice-daily scans and is available in near real-time. NUCAPS provides vertical profiles of temperature, humidity, and trace gases such as ozone, methane, and carbon monoxide.

NUCAPS humidity profiles is useful for studying the vertical profile of SAL and to verify model predictions for SAL propagation.

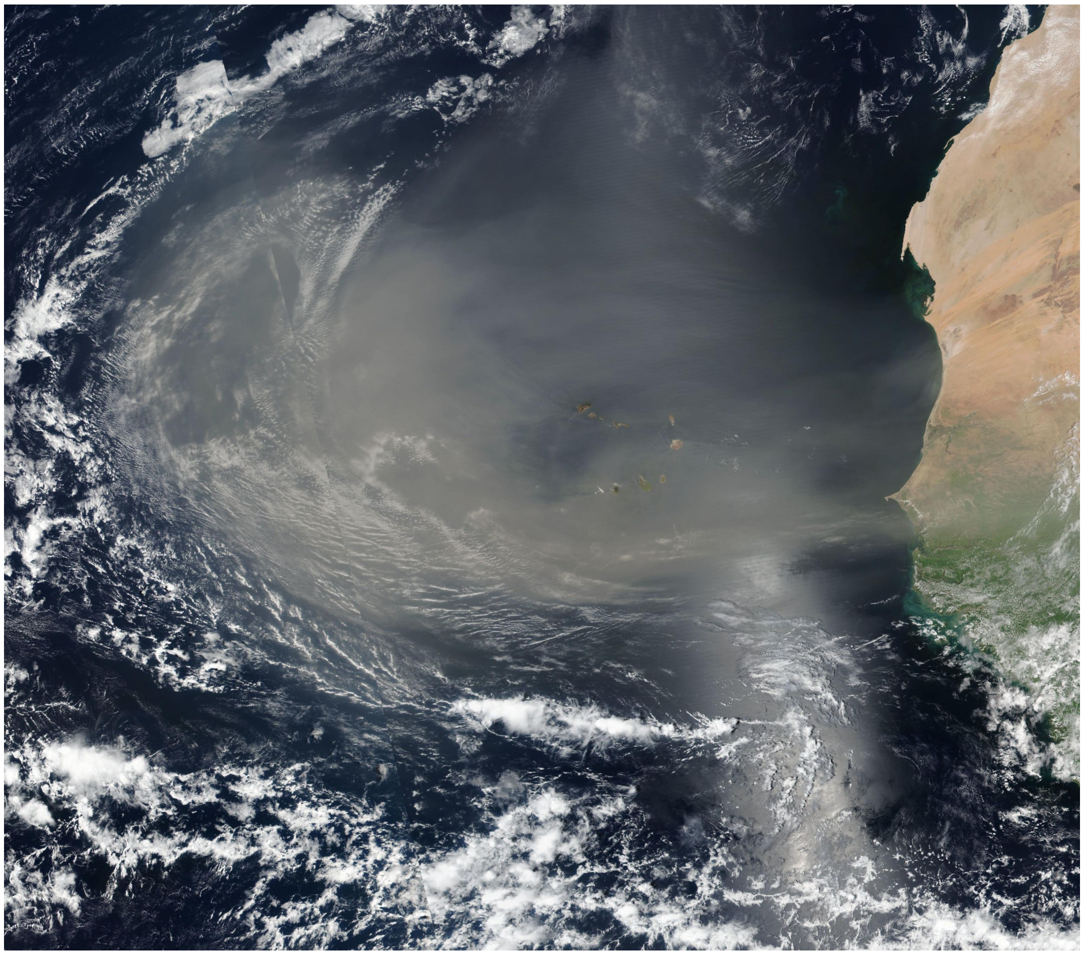


Figure 1: A SAL outflow on September 15, 2019

3 Where can I get NUCAPS datasets?

For researchers, NOAA-20 and Suomi NPP NUCAPS data can be ordered from NOAA CLASS, under the JPSS Sound Products (JPSS_SND) drop down menu. For operational forecasters, NOAA-20 NUCAPS is available within 20 mintes through in AWIPS.

4 How can I visualize NUCAPS datasets?

The daily ascending and descending overpasses can be viewed online via the NOAA/STAR. Additionally, skew-T visualizations are routinely produced by NOAA/OSPO for NOAA-20, Suomi NPP, and MetOp series.

Below is a short tutorial on how to display NUCAPS using Python and Jupyter Notebooks. This tutorial is intended to be a beginner-friendly exercise. So, even if you are not a Python programmer, you may be able to port it to another language of choice.

4.1 Objectives

- Open and inspect a single NUCAPS file, containing one swath of data
- Create maps and vertical cross section plots
- Combine many single files onto a map

First, we will import several 'helper' libraries to process the data. the '#' symbole indicates comments and will not run with the code.

```
In [109]: import numpy as np                            # To perform array operations
          import cartopy.crs as ccrs                    # To create map projections for plots
          import cartopy.feature as cfeature            # To add maps to plots
          from glob import glob                         # Search all files in a directory
          import matplotlib.pyplot as plt               # Main plotting library
          import xarray as xr                           # For working with netCDF files

          plt.rcParams['figure.figsize'] = [15, 10] # Sets all figures in document to be 15"x10"
          plt.rcParams.update({'font.size': 21})        # Sets fontsize in document to 21pts
```

4.2 The JPSS file naming scheme

The code below will read in a single netCDF file. The file names are very long!
NUCAPS-EDR_v2r0_npp_s201809201634390_e201809201635090_c201809201739220.nc
But there's a general trend in the filenames, so using the underscore (_) as a seperator:

- NUCAPS-EDR: product
- v2r0: algorithm
- npp: satellite
- s201809201634390: start time
- e201809201635090: end time
- c201809201739220: creation time

By looking at the start and end times of the name, we see that the file contains 1 minute of NUCAPS data collected from the Suomi NPP sensors.

4.3 Importing a single granule of data using xarray

Using the open_dataset command in xarray, let's import the file. Note that we need the decode_times option to be false, because xarray expects them to be stored in miliseconds and the NUCAPS files stores them in nanoseconds.

```
In [49]: # Read in a single NUCAPS netcdf file
         # Set decode time = false (the time doesnt follow standard formatting)
         fname = 'sal/NUCAPS-EDR_v2r0_npp_s201809201638230_e201809201638530_c201809201742190.nc'
         nucaps = xr.open_dataset(fname, decode_times=False)
```

```
In [82]: # Uncomment to inspect the file contents...
         nucaps
```

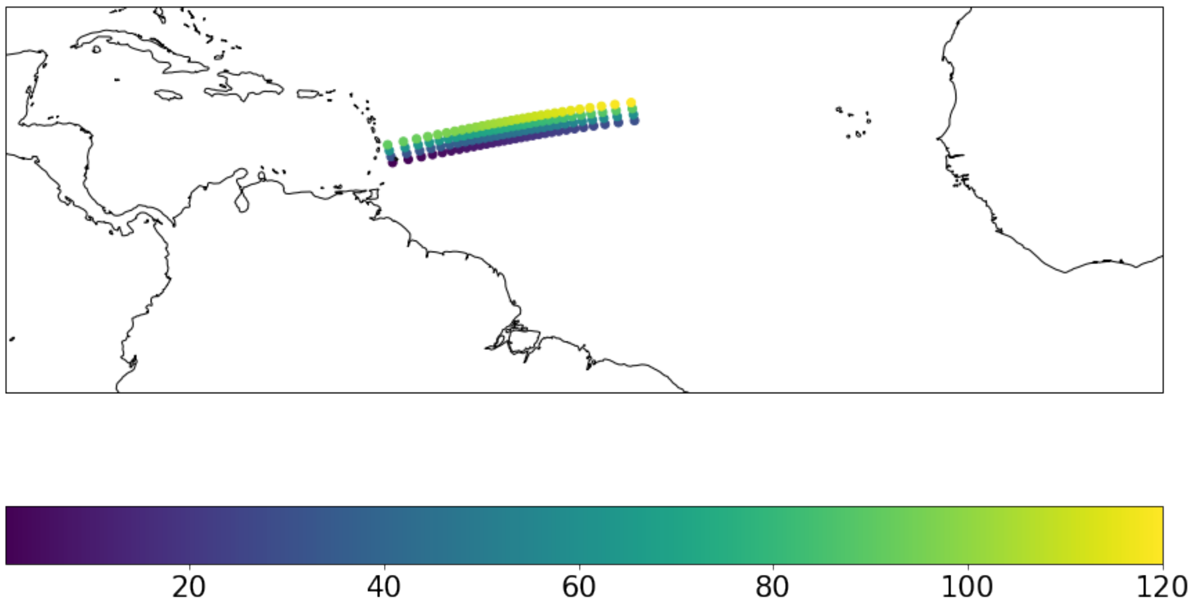
```
Out[82]: <xarray.Dataset>
         Dimensions:                 (Number_of_Cloud_Emis_Hing_Pts: 100, Number_of_Cloud_Layers: 8,
            Number_of_CrIS_FORs: 120, Number_of_Ispares: 129, Number_of_MW_Spectral_Pts: 16,
            Number_of_P_Levels: 100, Number_of_Rspares: 262, Number_of_Stability_Parameters: 16,
            Number_of_Surf_Emis_Hinge_Pts: 100)
         Coordinates:
             Time                    (Number_of_CrIS_FORs) float64 ...
             Latitude                (Number_of_CrIS_FORs) float32 ...
             Longitude               (Number_of_CrIS_FORs) float32 ...
             Pressure                (Number_of_CrIS_FORs, Number_of_P_Levels) float32 ...
             Effective_Pressure      (Number_of_CrIS_FORs, Number_of_P_Levels) float32 ...
         Dimensions without coordinates: Number_of_Cloud_Emis_Hing_Pts, Number_of_Cloud_Layers,
          Number_of_CrIS_FORs, Number_of_Ispares, Number_of_MW_Spectral_Pts,
          Number_of_P_Levels, Number_of_Rspares, Number_of_Stability_Parameters,
          Number_of_Surf_Emis_Hinge_Pts
         Data variables:
             quality_information     |S1 ...
             CrIS_FORs               (Number_of_CrIS_FORs) float64 ...
             View_Angle              (Number_of_CrIS_FORs) float32 ...
             Satellite_Height        (Number_of_CrIS_FORs) float32 ...
             FG_Mean_CO2             (Number_of_CrIS_FORs) float32 ...
             Mean_CO2                (Number_of_CrIS_FORs) float32 ...
             Solar_Zenith            (Number_of_CrIS_FORs) float32 ...
             Ascending_Descending    (Number_of_CrIS_FORs) float32 ...
             Topography              (Number_of_CrIS_FORs) float32 ...
             Land_Fraction           (Number_of_CrIS_FORs) float32 ...
             Surface_Pressure        (Number_of_CrIS_FORs) float32 ...
             Skin_Temperature        (Number_of_CrIS_FORs) float32 ...
             MIT_Skin_Temperature    (Number_of_CrIS_FORs) float32 ...
             FG_Skin_Temperature     (Number_of_CrIS_FORs) float32 ...
             MW_Surface_Class        (Number_of_CrIS_FORs) float32 ...
             ...
             Ispare_Field            (Number_of_CrIS_FORs, Number_of_Ispares) float64 ...
             Rspare_Field            (Number_of_CrIS_FORs, Number_of_Rspares) float32 ...
             Cloud_Top_Pressure      (Number_of_CrIS_FORs, Number_of_Cloud_Layers) float32 ...
```

```
    Cloud_Top_Fraction    (Number_of_CrIS_FORs, Number_of_Cloud_Layers) float32 ...
    Temperature           (Number_of_CrIS_FORs, Number_of_P_Levels) float32 ...
    MIT_Temperature       (Number_of_CrIS_FORs, Number_of_P_Levels) float32 ...
    FG_Temperature        (Number_of_CrIS_FORs, Number_of_P_Levels) float32 ...
    H2O                   (Number_of_CrIS_FORs, Number_of_P_Levels) float32 ...
    ...
```

4.4 Creating a simple map

If successfully imported, the command above will display the dimensions, coordinates, and available variables to plot. First, we will make a map showing where this granule is by plotting the latitude and logitude coordinates. NUCAPS data is saved as a 120 field of regards (FORs). This is stored as the "CrIS_FORs" coordinate in the NUCAPS netCDF file. Each FOR is 50km at nadir and 150km at the scan edge.

```
In [114]: # Initiate the plot
          fig = plt.figure(figsize=(15, 10))

          # Adds a map to the plot
          ax=plt.axes(projection=ccrs.PlateCarree())
          ax.coastlines('50m')

          # Plots the latitude and longitude of the NUCAPS data
          plt.scatter(nucaps['Longitude'], nucaps['Latitude'], c=nucaps['CrIS_FORs'])

          plt.colorbar(orientation='horizontal')

          # Expands axes
          ax.set_ylim(-5, 25)
          ax.set_xlim(-90, 0)

          # Display plot
          plt.show()
```



From above, the 120 FORs are numbered linearly, starting at one in the lower left of the swath, increase to the right, resetting to a new row with every 30th observation.

4.5 Plotting a vertical profile of water vapor to examine the SAL structure

From the list of data variables, we have access to vertical profiles of temperature, H_2O, O_3, CO, etc. Since the SAL is dry, we will see a dry layer when looking at the vertical profiles for each FOR.

To make this plot, we have to make sure all the variables are the same dimensions. By printing below, we'll see that the CRIS_FORs are one-dimensional while pressure and H_2O are two-dimensional:

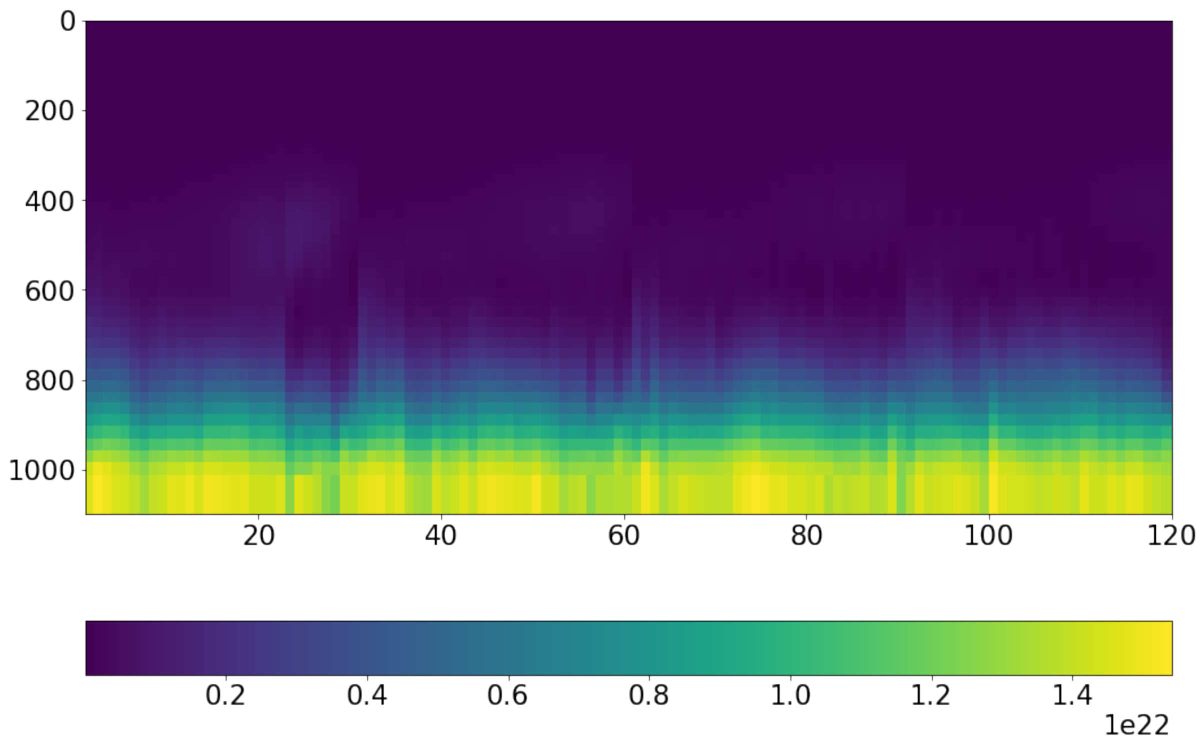
```
In [52]: print(nucaps['Pressure'].shape, nucaps['H2O'].shape, nucaps['CrIS_FORs'].shape)

(120, 100) (120, 100) (120,)
```

We can use the repeat and command to repeat the 100 pressure levels 120 times, once for each FOR. Then we will reshape this 1D array into a 2D array to match the other two variables:

```
In [71]: repeatFORs = np.repeat(nucaps['CrIS_FORs'].values, 100).reshape(120,100)
         repeatFORs

Out[71]: array([[  1.,   1.,   1., ...,   1.,   1.,   1.],
                [  2.,   2.,   2., ...,   2.,   2.,   2.],
                [  3.,   3.,   3., ...,   3.,   3.,   3.],
                ...,
                [118., 118., 118., ..., 118., 118., 118.],
                [119., 119., 119., ..., 119., 119., 119.],
                [120., 120., 120., ..., 120., 120., 120.]])
```

Now, we can make a contour plot, which requires three inputs: the x-axis variable (FOR), the y-axis variable (pressure), and the z variable or color coding (H2O). In the plot below, we do not include a map since it's a profile but we do invert the y axis, because pressure decreases with height and by default, matplotlib will make the plot in ascending order.

```
In [111]: # Initiate the plot
          fig = plt.figure()

          # Plots the latitude and longitude of the NUCAPS data
          plt.pcolormesh(repeatFORs, nucaps['Pressure'], nucaps['H2O'].values)

          plt.colorbar(orientation='horizontal')

          # Reverse the y axes
          plt.gca().invert_yaxis()

          # Display plot
          plt.show()
```



You may observe the repeating pattern above. That's because the row resets after every 30th FOR. However, we can see that the air is quite dry above 850mb. Since the SAL is typically dry between 850-500mb, our swath may be capturing the SAL. In the next section, we will gain greater situational awareness examining a horizontal swath of data.

4.6 Creating a horizontal cross section map of moisture to find pockets of dry air

Let's make a map of the horizontal cross section of water vapor near 500mb. NUCAPS is gridded to an irregular set of pressure levels, so we need to find the closest one to 500mb. Below, we make a dictionary of all the pressure levels contains in the file. The first line prints all the pressure levels as integers (dtype='i4') from the first FOR (since they're repeating, we only need one), converts it to a numpy array using np.array(...). Then, the loop appends all the pressure levels and their index.

```
In [66]: # Make a dictionary with pressure levels and indices ...
         pressureLevs = np.array(nucaps.sel(Number_of_CrIS_FORs=0).Pressure.values, dtype='i4')

         PresLevIndex = {}
         for i, plev in enumerate(pressureLevs):
             PresLevIndex.update( {plev : i} )
```

Below prints out all available pressure levels that we can choose from; we can see that the closest to 500mb is 496mb.

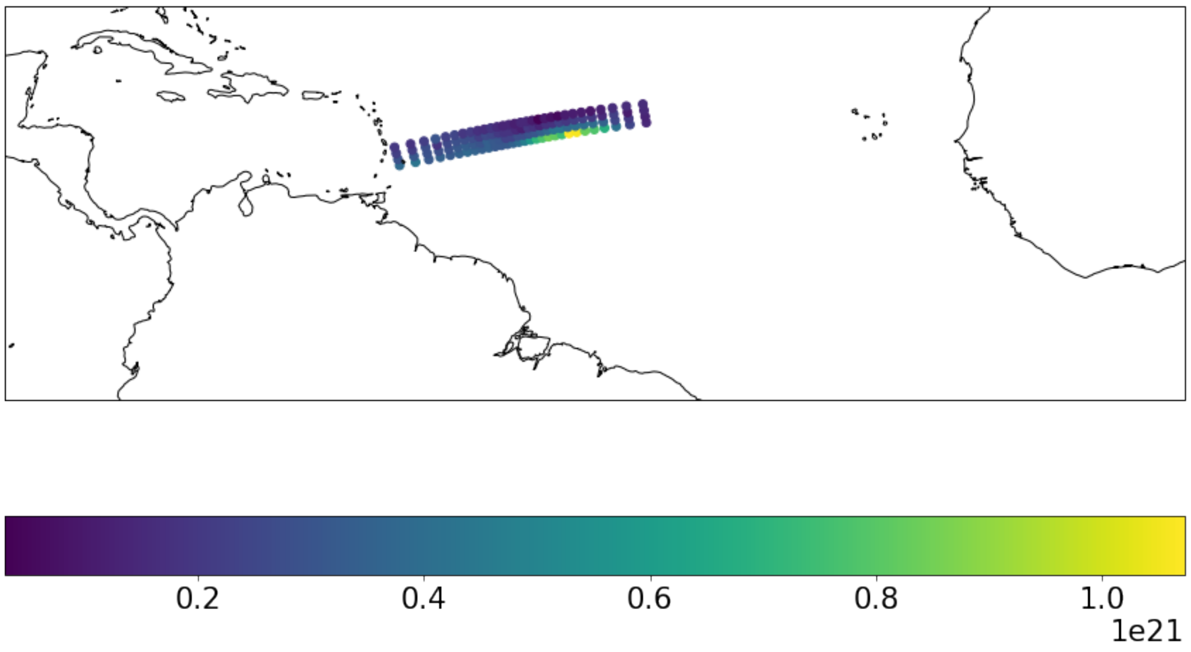
```
In [106]: # Print out all pressure levels in the dictionary
          PresLevIndex.keys()

Out[106]: dict_keys([..., 459, 477, 496, 515, 535, 555, 575, 596, 617, ... ])
```

The procedure the above is handy because we can use the index to select the pressure levels we want to take a slice from, which we'll map to a new variable griddedView.

```
In [123]: griddedView = nucaps.sel(Number_of_P_Levels=PresLevIndex[496])
          #griddedView
```

Below, we make a map showing the horizontal cross section of water vapor.

```
In [124]: # Initiate the plot
          fig = plt.figure()

          # Adds a map to the plot
          ax=plt.axes(projection=ccrs.PlateCarree())
          ax.coastlines('50m')

          # Plots the latitude and longitude of the NUCAPS data
          plt.scatter(griddedView['Longitude'], griddedView['Latitude'], c=griddedView['H2O'])

          plt.colorbar(orientation='horizontal')

          # Expands axes
          ax.set_ylim(-5, 25)
          ax.set_xlim(-90, 0)

          # Display plot
          plt.show()
```



4.7 Importing multiple granules of data using xarray

The above map is interesting, but it only represents a small area. By importing more granules of data, we can construct a larger area. Using glob, we can search for all files in a directory. Instead of open_dataset, we can use xarrays open_mfdataset (multi file dataset) all at the same time.

```
In [45]: # Import all files (may take a momenbt...)
         allfiles = glob("sal/NUCAPS-EDR_v2r0_npp_s*.nc")
         nucapsAll = xr.open_mfdataset(allfiles, decode_times=False)
```

4.8 Filtering data by quality

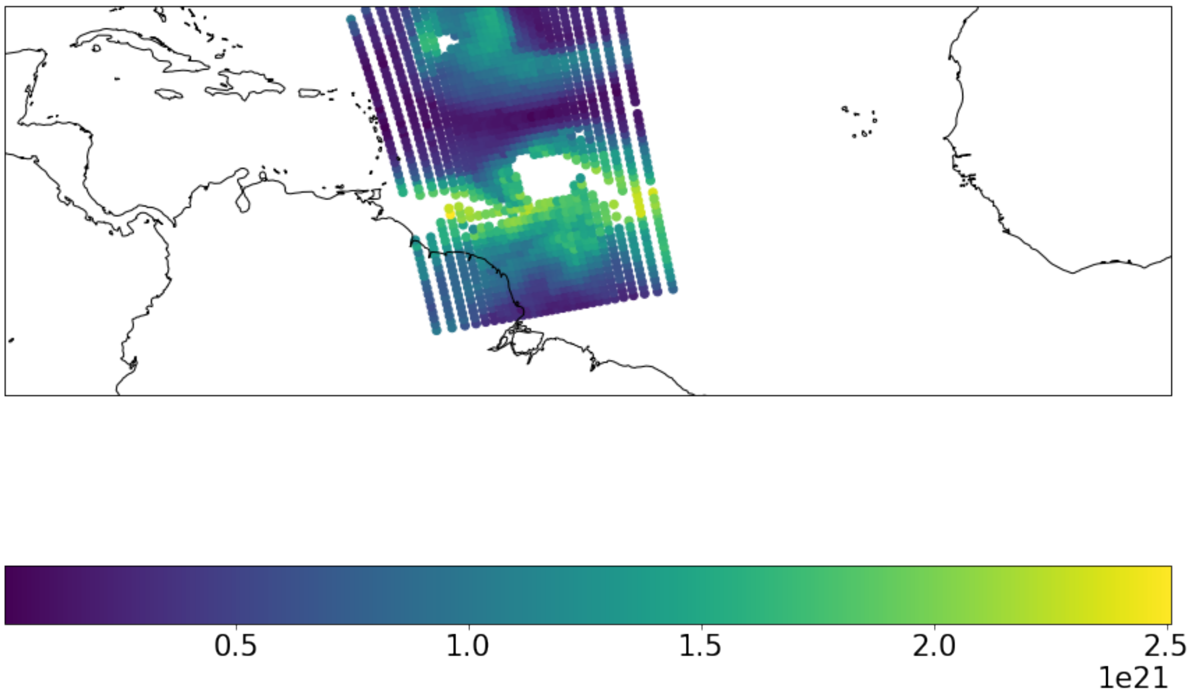
We can repeat the steps we did for the single granule to make a map of water vapor at 496mb. However, let's do one last thing, which is filter by data quality. While partly clear conditions still cna yield good profiles, retrievals can fail within uniform cloud decks, producing unrealistic values at pressure levels below clouds. We can use the where command to keep the good data (Quality_Flag == 0) and drop the rest (Quality_Flag==1).

```
In [118]: griddedViewAll = nucapsAll.sel(Number_of_P_Levels=PresLevIndex[496])

       # To filter by quality flag:
       griddedViewAll = griddedViewAll.where(griddedViewAll['Quality_Flag'] == 0, drop=True)
```

4.9 Plotting multiple, quality filtered water vapor cross sections to see the extent of the SAL

The code below is identical to our single granule example, however, but now we have more granules to look at in our plot:

```
In [29]: # Initiate the plot
         fig = plt.figure(figsize=(15, 15))

         # Adds a map to the plot
         ax=plt.axes(projection=ccrs.PlateCarree())
         ax.coastlines('50m')

         # Plots the latitude and longitude of the NUCAPS data
         plt.scatter(griddedViewAll['Longitude'], griddedViewAll['Latitude'], \
                     c=griddedViewAll['H2O'])

         plt.colorbar(orientation='horizontal')

         # Expands axes
         ax.set_ylim(-5, 25)
         ax.set_xlim(-90, 0)

         # Display plot
         plt.show()
```



Looking above, we can see a very dry layer of air. From the GOES-16 Split Window Difference (10.33 μm – 12.30 μm) in Figure 2, we can confirm the presense of the SAL.

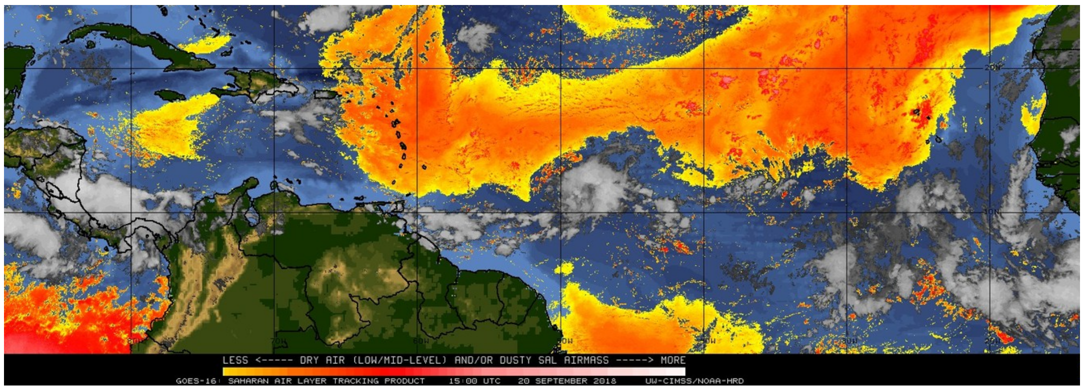

LESS <----- DRY AIR (LOW/MID-LEVEL) AND/OR DUSTY SAL AIRMASS -----> MORE
GOES-16: SAHARAN AIR LAYER TRACKING PRODUCT 15:00 UTC 20 SEPTEMBER 2018 UW-CIMSS/NOAA-HRD

Figure 2: GOES-16 Split Window Difference on September 20, 2019

5 Summary

- The Saharan Air Layer (SAL) can be seen from space using visible and infrared imagery and products. Hyperspectral sounding data can be used to see both the vertical and horizontal extent in near real-time.
- NUCAPS is useful for examining the moisture and temperature structure of SAL events.
- Both code and web-based tools are useful for visualizating the SAL. For researchers, Suomi NPP and NOAA-20 NUCAPS are available via NOAA CLASS with a three-hour latency. For operational forecasters, NOAA-20 NUCAPS is available within 20 minutes in AWIPS.