

Monitoreo de la Capa de Polvo del Sahara utilizando aplicaciones de Satélite.

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

29 September 2019



[Este tutorial es interactivo. Cliquee/presione para operar código!](#)

1 ¿Qué es la capa de aire del Sahara?

La capa de aire del Sahara (SAL, por sus siglas en inglés), es una capa de aire cálido y seco que se origina sobre el desierto del Sahara y que se propaga a través del Atlántico. La SAL se extiende entre 850 y 500 hPa; y se caracteriza por gradientes termales verticales (lapse rates) muy agudos/grandes en la capa 850-500 hPa. Esto atrapa la humedad marítima tropical subyacente. Estos efectos pueden suprimir la formación de ciclones tropicales. Asimismo, la SAL puede constituir un riesgo importante para la salud. Esto porque transporta polvo a través del Atlántico hasta regiones pobladas del Caribe y el este de Estados Unidos. Esto puede incrementar casos de alergias y asma en poblaciones sensibles.

La figura 1 muestra un evento SAL del 15 de setiembre de 2018, usando el instrumento VIIRS del satélite Suomi NPP. Este evento SAL llegó al Caribe el 20 de setiembre de 2018. La SAL puede ser detectada desde el espacio con datos de imágenes visibles e infrarrojas de una variedad de sensores en plataformas de satélite. Entre los sensores se pueden mencionar el ABI (GOES-16), SEVIRI (Meteosat-9/-10), VIIRS (Suomi NPP, NOAA-20), y AVHRR (MetOp-A, -B,-C). La extensión horizontal de la SAL puede ser monitoreada usando productos RGB diseñados para medir polvo, diferencias de bandas (e.g., 10.33 μm – 12.30 μm), y productos de Profundidad Óptica de Aerosoles (AOD). Adicionalmente, sondeos de microondas e infrarrojos (CrIS, ATMS, AMSU, y IASI) son particularmente útiles para monitorear la SAL, debido a que sondeos permiten detectar no sólo la extensión horizontal de la SAL sino la vertical.

2 ¿Qué es NUCAPS?

El NOAA Unique Combined Atmospheric Processing System (NUCAPS), en español “Sistema de Procesamiento Combinado Unico de la NOAA”, produce sondeos atmosféricos de manera operacional a partir de productos de satélites de órbita polar Suomi (Suomi National-Polar-orbiting Partnership o Suomi NPP) y el satélite NOAA- 20. De cada satélite, NUCAPS genera escaneos dos veces por día. Estos están disponibles casi a tiempo real. NUCAPS genera perfiles verticales de temperatura, humedad y concentraciones de gases como el ozono, metano y monóxido de carbono. Los perfiles de humedad de NUCAPS son útiles para estudiar la estructura vertical de la SAL y para verificar predicciones de la propagación de la SAL de modelos.

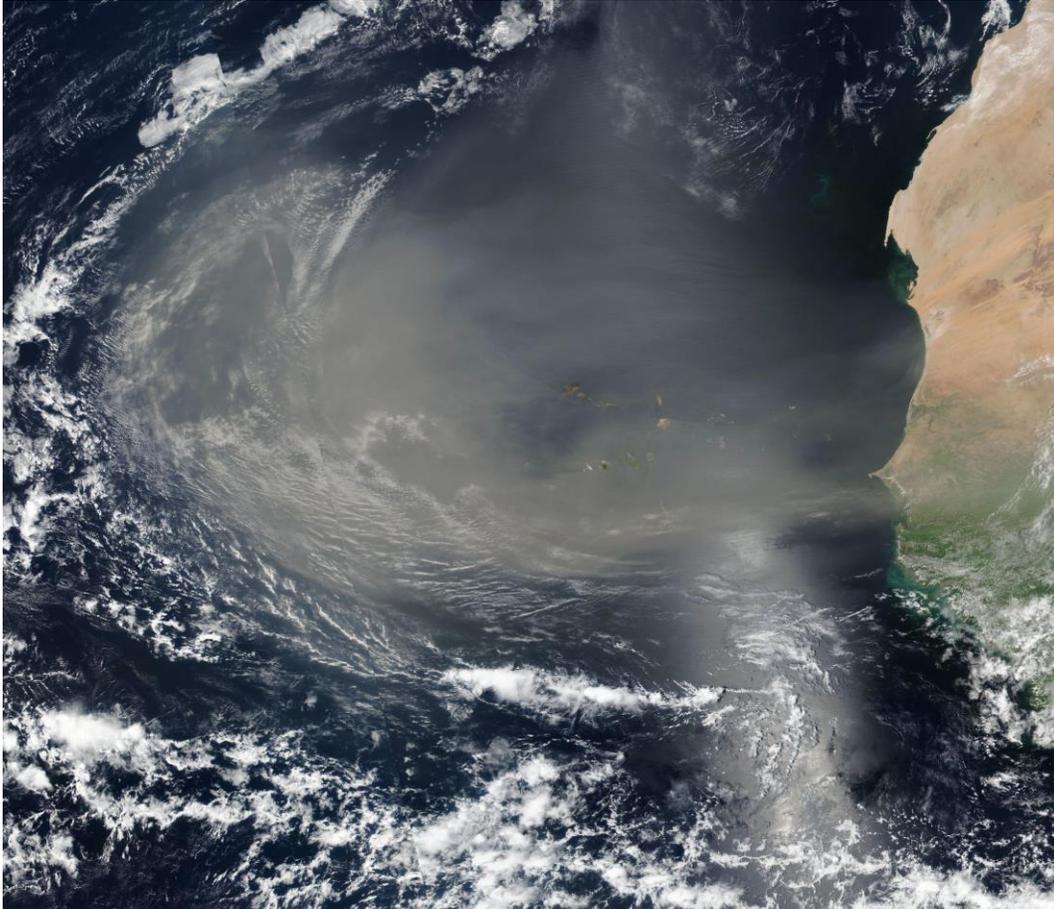


Figura 1: Evento SAL del 15 de setiembre de 2018

3 ¿Dónde se pueden obtener datos de NUCAPS?

Para investigadores, datos de NUCAPS a partir del NOAA-20 y Suomi NPP pueden ser ordenados a través de [NOAA CLASS](#), Bajo la sección del menú productos de sondeos de JPSS (JPSS Sound Products o JPSS_SND). Para pronosticadores operacionales, los sondeos NOAA-20 NUCAPS están disponibles cada 20 minutos a través de AWIPS.

4 ¿Cómo puedo visualizar datos de NUCAPS?

Los pases ascendentes y descendentes diarios pueden ser vistos en línea a través de [NOAA/STAR](#). Adicionalmente, las visualizaciones de sondeos Skew-T son producidas operacionalmente por [NOAA/OSPO](#) para NOAA-20, Suomi NPP, and MetOp series.

A continuación puede encontrar un tutorial que explica como desplegar datos de NUCAPS usando Python y Jupyter Notebooks. Este tutorial ha sido diseñado para ser amigable para el usuario. Así que si no es un programador de Python, es muy fácil replica el tutorial en otro lenguaje de programación.

4.1 Objetivos

- Abrir e inspeccionar un archive de NUCAPS simple NUCAPS file, que contiene una sola franja de datos.
- Crear mapas y secciones verticales.
- Combinar varios archivos en un solo mapa.

Primero, importaremos varias librerías de ayuda para procesar los datos. El símbolo '#' indica comentarios que no van a ser procesados por el código.

```
In [109]: import numpy as np                # Para operaciones de matrices/arreglos.
          import cartopy.crs as ccrs       # Para crear proyecciones.
          import cartopy.feature as cfeature # Para adicionar mapas a los gráficos.
          from glob import glob           # Para buscar archivos en directorios.
          import matplotlib.pyplot as plt  # Librería principal de ploteo.
          import xarray as xr             # Para trabajar con archivos NetCDF.

          plt.rcParams['figure.figsize'] = [15, 10] # Hace que todas las figuras sean 15"x10"
          plt.rcParams.update({'font.size': 21})  # Fija el tamaño de letra a 21pts
```

4.2 El esquema de nombramiento JPSS

El código que sigue lee un archive simple netCDF. Los nombres de los archivos son muy largos!

NUCAPS-EDR_v2r0_npp_s201809201634390_e201809201635090_c201809201739220.nc

Una tendencia recurrente en los nombres de los archivos es separar la información con un guión bajo "_":

- NUCAPS-EDR: producto
- v2r0: algoritmo
- npp: satélite
- s201809201634390: hora de inicio
- e201809201635090: hora de finalización
- c201809201739220: hora de creación

Al considerar los tiempos de inicio y finalización, vemos que el archive contiene 1 minuto de datos de NUCAPS colectados a partir de sensors del satélite Suomi NPP.

4.3 Cómo importar un gránulo simple de datos con xarray

Importemos el archivo usando el comando `open_dataset` en `xarray`. Nótese que necesitamos que la opción `decode_times` sea falsa, porque `xarray` espera guardar los datos en milisegundos; y los archivos de NUCAPS los guardan en nanosegundos.

```
In [49]: # Lee un archivo de NUCAPS simple en format NetCDF
          # Fija el periodo de decodificación a falso (los datos del tiempo no siguen un formato estándar)
          fname = 'sal/NUCAPS-EDR_v2r0_npp_s201809201638230_e201809201638530_c201809201742190.nc '
          nucaps = xr.open_dataset(fname, decode_times=False)
```

```
In [82]: # Puede descomentar para evaluar el contenido de los archivos...
          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
    ...

```

4.4 Cómo crear un mapa simple

Si puede importarse exitosamente, el comando de arriba desplegará las dimensiones, coordenadas y las variables a graficar. Primero, generaremos un mapa que muestra donde se está graficando este gránulo, al graficar las coordenadas de latitud y longitud. Los datos de NUCAPS se guardan como un campo de 120 regards (FORs). Esto se archiva como la coordenada "CrIS_FORs" en el archivo NetCFD de NUCAPS. Cada FOR es de 50km en el nadir y de 150km en el borde de la región de escaneo.

In [114]: *# Iniciar el gráfico*

```

fig = plt.figure(figsize=(15, 10))

# Añadir un mapa al gráfico.
ax=plt.axes(projection=ccrs.PlateCarree())
ax.coastlines('50m')

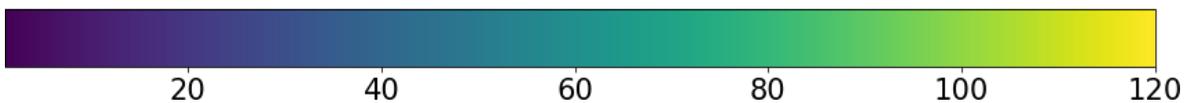
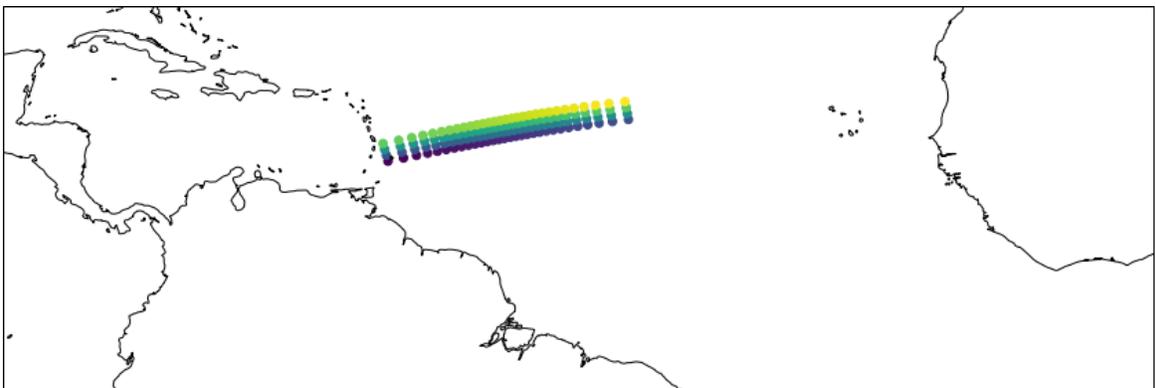
# Graficar la latitud y longitud de los datos de NUCAPS.
plt.scatter(nucaps['Longitude'], nucaps['Latitude'], c=nucaps['CrIS_FORs'])

plt.colorbar(orientation='horizontal')

# Expandir los ejes.
ax.set_ylim(-5, 25)
ax.set_xlim(-90, 0)

# Desplegar el gráfico.
plt.show()

```



Los 120 FORs se numeran de manera lineal, comenzando con uno en la esquina inferior derecho de la franja de muestreo (swath), incrementándose hacia la derecha, y reseteándose a una nueva fila al llegar a la observación #30.

4.5 Graficando un perfil vertical de vapor de agua para examinar la estructura de la SAL.

De la lista de variables que contienen datos, tenemos acceso a perfiles verticales de temperatura, H₂O, O₃, CO, etc. Como la SAL es bastante seca, vamos a poder ver una región seca en cada perfil para cada FOR.

Para generar este gráfico, tenemos que asegurarnos de que todas las variables estén en las mismas dimensiones. Al imprimir, vamos a ver que los datos de CRIS_FORs son uni-dimensionales mientras los de presión y H₂O son bi-dimensionales:

```
In [52]: print(nucaps['Pressure'].shape, nucaps['H2O'].shape, nucaps['CrIS_FORs'].shape)
(120, 100) (120, 100) (120,)
```

Podemos usar el comando repeat para repetir los 100 niveles de presión 120 veces, una vez para cada FOR., once for each FOR. Luego vamos a volver a mostrar este arreglo/matriz de 1D para convertirla en 2D, para poder hacer que coincidan las otras dos 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.]])
```

Ahora, podemos hacer un gráfico con contornos. Esto requiere tres inputs/ingresos: la variable del eje x (FOR, la variable del eje y (presión) y la variable z o código de colores (H₂O). En el gráfico que sigue no se incluye un mapa debido a que estamos viendo un perfil. Pero invertimos el eje y debido a que la presión disminuye con la altura y, por defecto, matplotlib genera el gráfico en orden ascendente.

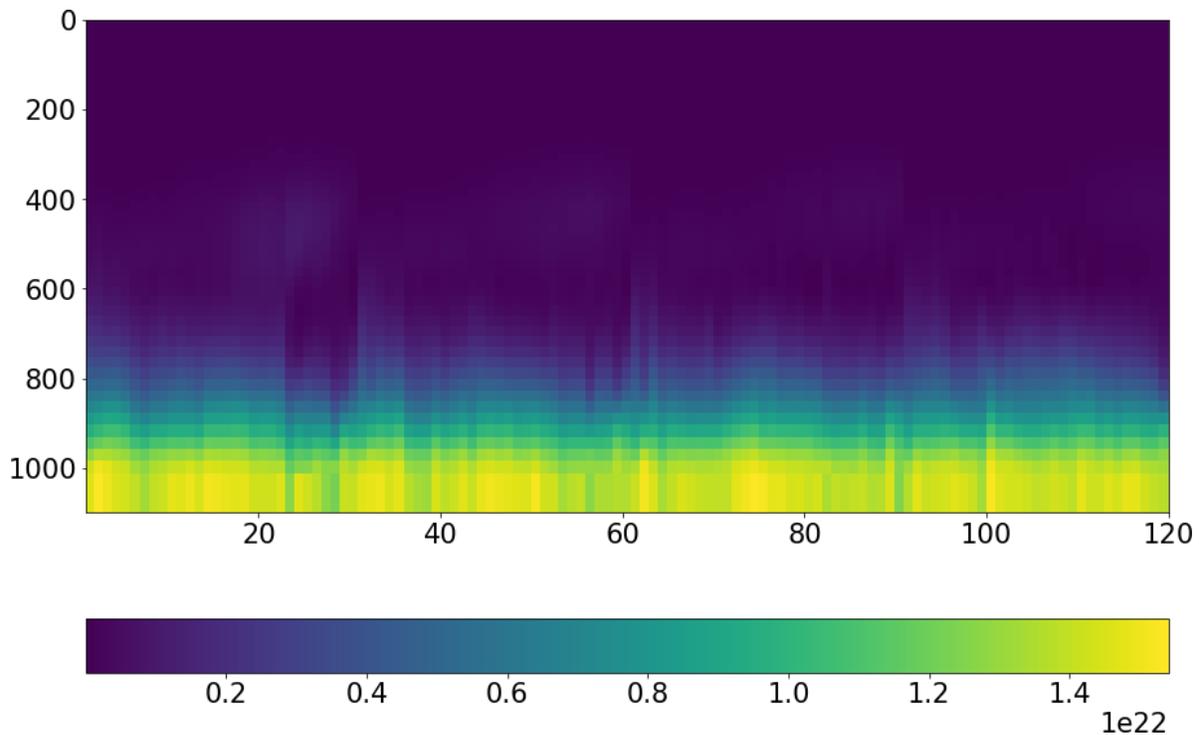
```
In [111]: # Iniciar el gráfico
fig = plt.figure()

# Graficar la latitud y longitud de los sondeos de NUCAPS
plt.pcolormesh(repeatFORs, nucaps['Pressure'], nucaps['H2O'].values)

plt.colorbar(orientation='horizontal')

# Revertir el eje y
plt.gca().invert_yaxis()

# Desplegar el gráfico
plt.show()
```



Puede observar el patrón repetitivo arriba. Esto es porque la línea se resetea después de cada FOR #30. Sin embargo, podemos ver que el aire es bastante seco por encima de los 850 hPa. Como la SAL es típicamente seca entre 850 y 500 hPa, nuestra franja de muestreo debe estar capturando la SAL. En la siguiente sección vamos a ganar mayor conciencia sobre la situación, al examinar la franja de datos.

4.6 Generando una sección horizontal de humedad, para encontrar regiones de aire seco.

Generemos un mapa de la sección horizontal de vapor de agua cerca de los 500 hPa. NUCAPS es generado en grillas de niveles de presión irregulares. Por ello debemos buscar el nivel que sea el más cercano al de 500 hPa. Debajo, Podemos generar una lista de todos los niveles de presión que contiene el archivo. La primera línea imprime en pantalla todos los niveles de presión como números enteros (`dtype='i4'`), desde el primer FOR. Como ellos se repiten, solamente necesitamos uno. Luego lo convierte en un arreglo numpy, `np.array()`. Luego, el loop/animación incorpora todos los niveles de presión y su índice.

```
In [66]: #Genera una lista/diccionario con todos los niveles de presión
         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} )
```

A continuación, se imprimen todos los niveles de presión que Podemos escoger. Podemos ver que el más cercano a 500 hPa es 406 hPa.

```
In [106]: # Imprime en pantalla los niveles.
          PresLevIndex.keys()
```

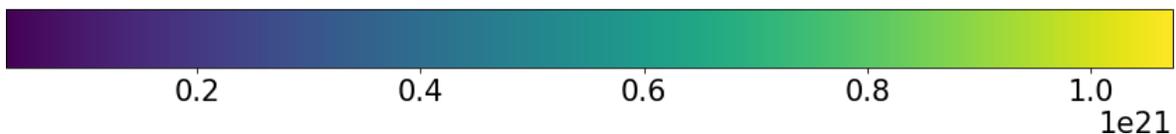
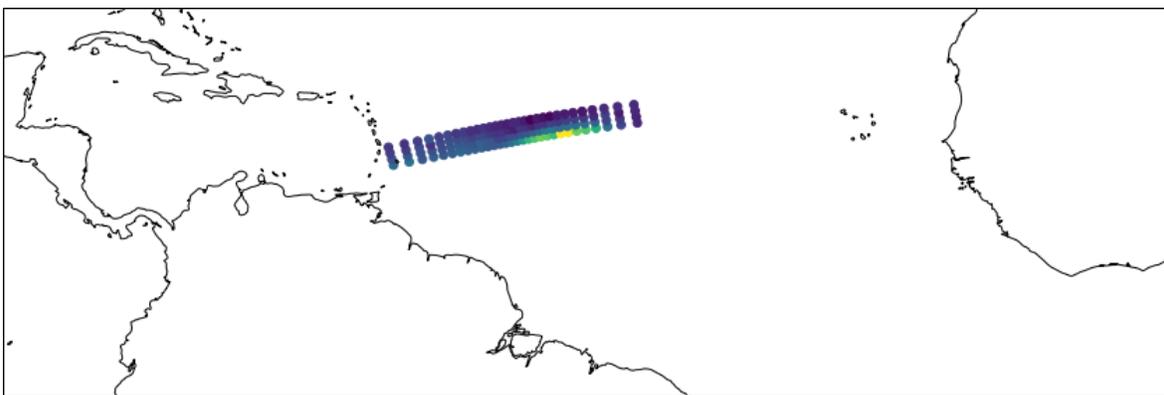
Out[106]: dict_keys([..., 459, 477, 496, 515, 535, 555, 575, 596, 617, ...])

El procedimiento de arriba es bastante útil, porque nos permite utilizar el índice para seleccionar los niveles de presión de los que queremos obtener una sección o franja. Esta franja la colocaremos en una nueva variable, llamada griddedView.

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

A continuación, haremos un mapa mostrando la sección horizontal de vapor de agua.

```
In [124]: # Inicio del gráfico  
         fig = plt.figure()  
  
         # Incorpora un mapa al gráfico.  
         ax=plt.axes(projection=ccrs.PlateCarree())  
         ax.coastlines('50m')  
  
         # Grafica la latitud y la longitud de los datos de NUCAPS  
         plt.scatter(griddedView['Longitude'], griddedView['Latitude'], c=griddedView['H2O'])  
  
         plt.colorbar(orientation='horizontal')  
  
         # Expande los ejes  
         ax.set_ylim(-5, 25)  
         ax.set_xlim(-90, 0)  
  
         # Desplega el gráfico  
         plt.show()
```



4.7 Cómo importar múltiples gránulos de datos usando xarray

El mapa de arriba es interesante, pero solamente representa una región discreta de datos. Al importar un mayor número de gránulos de datos, podemos generar una región más grande. Usando `glob`, podemos realizar una búsqueda de todos los archivos en un directorio. En vez de `open_dataset`, podemos usar `xarrays open_mfdataset` (grupo de datos que contiene varios archivos), todos al mismo tiempo.

```
In [45]: # Importar todos los archivos (esto puede tomar un tiempo)
allfiles = glob("sal/NUCAPS-EDR_v2r0_npp_s*.nc")
nucapsAll = xr.open_mfdataset(allfiles, decode_times=False)
```

4.8 Filtrado de los datos en función a su calidad

Podemos repetir los pasos que realizamos para el gránulo simple, para generar un mapa de vapor de agua a 496 hPa. Sin embargo, hagamos algo antes: Filtramos los datos en función a su calidad. Mientras condiciones parcialmente despejadas pueden producir buenos perfiles, los algoritmos para obtener datos pueden fallar cuando existen sistemas nubosos uniformes. Esto produce valores poco realistas en niveles de presión que se encuentran por debajo de las nubes. Podemos usar el comando “where” para conservar los datos buenos (`Quality_Flag == 0`) y para deshacernos del resto (`Quality_Flag==1`).

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

# Filtrado de datos con las banderas/interruptores de control de calidad:
griddedViewAll = griddedViewAll.where(griddedViewAll['Quality_Flag'] == 0, drop=True)
```

4.9 Cómo graficar secciones horizontales de vapor de agua, que han pasado el control de calidad, para ver la extensión de la SAL.

El código siguiente es idéntico a nuestro ejemplo de gránulo simple. Sin embargo, ahora tenemos un mayor número de gránulos para poder analizar nuestro gráfico:

```
In [29]: # Iniciar el gráfico
fig = plt.figure(figsize=(15, 15))

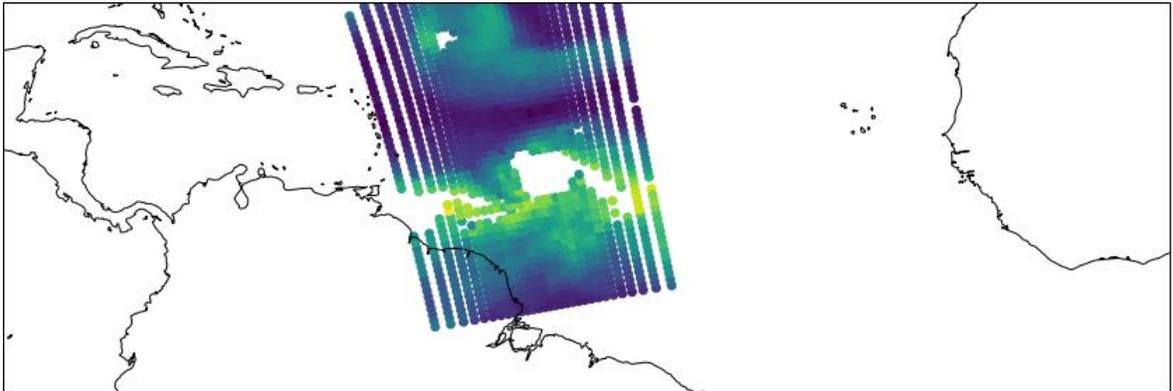
# Incluir el gráfico en el mapa
ax=plt.axes(projection=ccrs.PlateCarree())
ax.coastlines('50m')

# Graficar la latitud y la longitud de los datos de NUCAPS
plt.scatter(griddedViewAll['Longitude'], griddedViewAll['Latitude'], \
            c=griddedViewAll['H2O'])

plt.colorbar(orientation='horizontal')

# Expandir los ejes
ax.set_ylim(-5, 25)
ax.set_xlim(-90, 0)
```

```
# Desplegar el gráfico  
plt.show()
```



Mirando arriba, Podemos ver una capa de aire bastante seco. Podemos confirmar la presencia de la SAL usando la diferencia Split Window del GOES-16 ($10.33\mu\text{m} - 12.30\mu\text{m}$) en la figura 2.

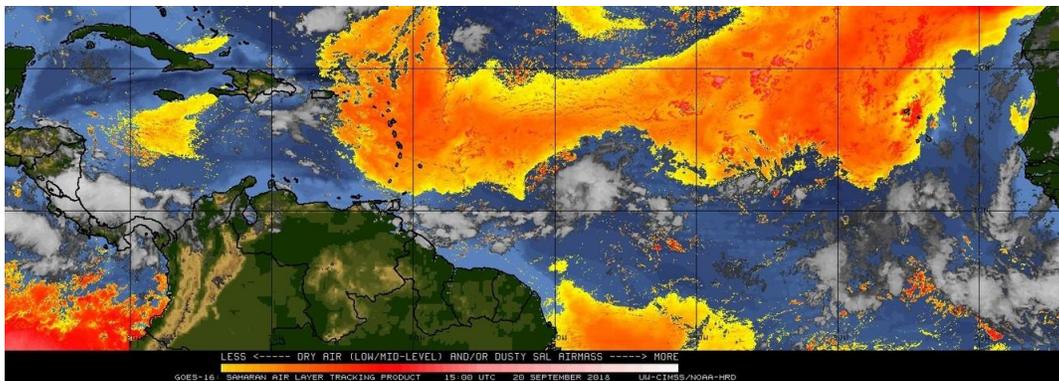


Figura 2: GOES-16 Split Window Difference - 20 de setiembre de 2019

5 Resumen

- La capa de aire del Sahara (SAL) puede ser vista desde el espacio usando imágenes visibles e infrarrojas, además de otros productos generados con ellas. Sondeos hiperspectrales pueden ser utilizados asimismo para ver la extensión horizontal y vertical de la SAL, casi a tiempo real.
- NUCAPS es útil para examinar la estructura termal y de humedad asociada a la SAL.
- Para visualizar la SAL, son útiles tanto las herramientas en línea como el código. Para investigadores, NUCAPS del Suomi NPP y NOAA-20 están disponibles en [NOAA CLASS](#), con una latencia de tres horas. Para pronosticadores operacionales, los NUCAPS del NOAA-20 están disponibles cada 20 minutos en AWIPS.