00.Descarga_GOES-R

November 3, 2022

1 Descargar GOES-R

1.1 Instalar librerias necesarias

Vamos a crear un nuevo ambiente virtual. En mi caso tendrá el nombre goes_r. Además instalaré algunas librerías necesarias para trabajar con datos GOES-R.

```
conda create --name goes_r -c conda-forge matplotlib netcdf4 cartopy xarray boto3 gdal scipy pa
```

Podemos ver varias instrucciones de GOES-R en esta guía de usuario.

1.1.1 Importante

Los codigos que vamos a trabajar en esta secciónn fueron desarrollados el CPTEC/INPE-Brasil. Pueden encontrar un vídeo extenso del curso en la siguiente página link.

1.1.2 Vamos a descargar GOES-R

Descargaremos datos desde el Servicio Web de Amazon (AWS).

1.2 Leer el archivo netCDF

1.2.1 Primero, vamos a leer con xarray

[ ]:
```
import xarray as xr
```

[ ]:
```
ds = xr.open_dataset('data/
  ↪OR_ABI-L2-CMIPF-M6C13_G16_s20223041800203_e20223041809522_c20223041810015.
  ↪nc')
ds
```

[ ]:
```
<xarray.Dataset>
Dimensions:                                                 (y: 5424, x: 5424,
                                                             number_of_time_bounds: 2,
                                                             number_of_image_bounds:
2,                                                           band: 1)

Coordinates:
    t                                                       datetime64[ns] …
```

```
  * y                                                    (y) float64 0.1518 … …
  * x                                                    (x) float64 -0.1518 …
    y_image                                              float32 …
    x_image                                              float32 …
    band_wavelength                                      (band) float32 …
    band_id                                              (band) int32 …
Dimensions without coordinates: number_of_time_bounds, number_of_image_bounds,
                                band
Data variables: (12/35)
    CMI                                                  (y, x) float32 …
    DQF                                                  (y, x) float32 …
    time_bounds                                          (number_of_time_bounds)
datetime64[ns] …
    goes_imager_projection                               int32 …
    y_image_bounds                                       (number_of_image_bounds)
float32 …
    x_image_bounds                                       (number_of_image_bounds)
float32 …
    …                                                   …
    focal_plane_temperature_threshold_exceeded_count  float64 …
    maximum_focal_plane_temperature                   float32 …
    focal_plane_temperature_threshold_increasing      float32 …
    focal_plane_temperature_threshold_decreasing      float32 …
    channel_integration_time                          float64 …
    channel_gain_field                                float64 …
Attributes: (12/29)
    naming_authority:          gov.nesdis.noaa
    Conventions:               CF-1.7
    Metadata_Conventions:      Unidata Dataset Discovery v1.0
    standard_name_vocabulary:  CF Standard Name Table (v35, 20 July 2016)
    institution:               DOC/NOAA/NESDIS > U.S. Department of Commerce,…
    project:                   GOES
    …                         …
    cdm_data_type:             Image
    time_coverage_start:       2022-10-27T00:00:31.9Z
    time_coverage_end:         2022-10-27T00:09:39.6Z
    timeline_id:               ABI Mode 6
    production_data_source:    Realtime
    id:                        919aa206-e73f-4bb1-a8a0-93f8a2c4a05a
```

1.2.2 Ahora, vamos a leer con netCDF4

[ ]:
```
from netCDF4 import Dataset
import matplotlib.pyplot as plt
```

[ ]:
```
file = Dataset('data/
  ↪OR_ABI-L2-CMIPF-M6C13_G16_s20223041800203_e20223041809522_c20223041810015.
  ↪nc')
file
```

[ ]:
```
<class 'netCDF4._netCDF4.Dataset'>
root group (NETCDF4 data model, file format HDF5):
    naming_authority: gov.nesdis.noaa
    Conventions: CF-1.7
    Metadata_Conventions: Unidata Dataset Discovery v1.0
    standard_name_vocabulary: CF Standard Name Table (v35, 20 July 2016)
    institution: DOC/NOAA/NESDIS > U.S. Department of Commerce, National Oceanic
and Atmospheric Administration, National Environmental Satellite, Data, and
Information Services
    project: GOES
    production_site: NSOF
    production_environment: OE
    spatial_resolution: 2km at nadir
    orbital_slot: GOES-West
    platform_ID: G17
    instrument_type: GOES R Series Advanced Baseline Imager
    scene_id: Full Disk
    instrument_ID: FM2
    dataset_name:
OR_ABI-L2-CMIPF-M6C13_G17_s20223000000319_e20223000009396_c20223000009474.nc
    iso_series_metadata_id: 8c9e8150-3692-11e3-aa6e-0800200c9a66
    title: ABI L2 Cloud and Moisture Imagery
    summary: Single emissive band Cloud and Moisture Imagery Products are
digital maps of clouds, moisture and atmospheric windows at IR bands.
    keywords: SPECTRAL/ENGINEERING > INFRARED WAVELENGTHS > BRIGHTNESS
TEMPERATURE
    keywords_vocabulary: NASA Global Change Master Directory (GCMD) Earth
Science Keywords, Version 7.0.0.0.0
    license: Unclassified data.  Access is restricted to approved users only.
    processing_level: National Aeronautics and Space Administration (NASA) L2
    date_created: 2022-10-27T00:09:47.4Z
    cdm_data_type: Image
    time_coverage_start: 2022-10-27T00:00:31.9Z
    time_coverage_end: 2022-10-27T00:09:39.6Z
    timeline_id: ABI Mode 6
    production_data_source: Realtime
    id: 919aa206-e73f-4bb1-a8a0-93f8a2c4a05a
    dimensions(sizes): y(5424), x(5424), number_of_time_bounds(2), band(1),
number_of_image_bounds(2)
    variables(dimensions): int16 CMI(y, x), int8 DQF(y, x), float64 t(), int16
y(y), int16 x(x), float64 time_bounds(number_of_time_bounds), int32
goes_imager_projection(), float32 y_image(), float32
```

```
y_image_bounds(number_of_image_bounds), float32 x_image(), float32
x_image_bounds(number_of_image_bounds), float32
nominal_satellite_subpoint_lat(), float32 nominal_satellite_subpoint_lon(),
float32 nominal_satellite_height(), float32 geospatial_lat_lon_extent(), float32
band_wavelength(band), int32 band_id(band), int32 total_number_of_points(),
int32 valid_pixel_count(), int32 outlier_pixel_count(), float32
min_brightness_temperature(), float32 max_brightness_temperature(), float32
mean_brightness_temperature(), float32 std_dev_brightness_temperature(), float32
planck_fk1(), float32 planck_fk2(), float32 planck_bc1(), float32 planck_bc2(),
int32 algorithm_dynamic_input_data_container(), float32
percent_uncorrectable_GRB_errors(), float32 percent_uncorrectable_L0_errors(),
float32 earth_sun_distance_anomaly_in_AU(), int32
processing_parm_version_container(), int32
algorithm_product_version_container(), float32 esun(), float32 kappa0(), int32
focal_plane_temperature_threshold_exceeded_count(), float32
maximum_focal_plane_temperature(), float32
focal_plane_temperature_threshold_increasing(), float32
focal_plane_temperature_threshold_decreasing(), int32
channel_integration_time(), int32 channel_gain_field()
    groups:
```

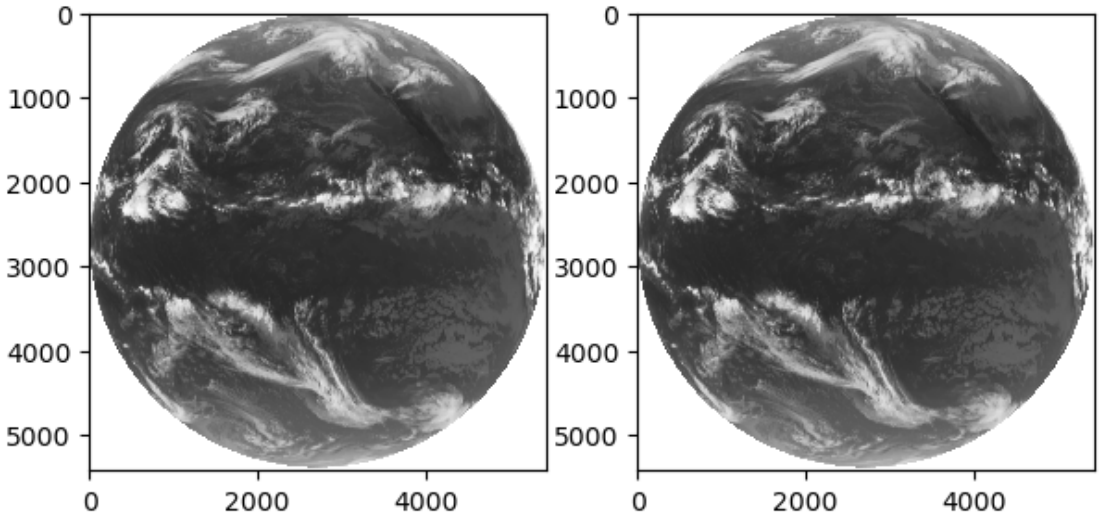
[ ]:
```
# Get the pixel values
data = file.variables['CMI'][:]
data
```

[ ]:
```
masked_array(
  data=[[--, --, --, …, --, --, --],
        [--, --, --, …, --, --, --],
        [--, --, --, …, --, --, --],
        …,
        [--, --, --, …, --, --, --],
        [--, --, --, …, --, --, --],
        [--, --, --, …, --, --, --]],
  mask=[[ True,  True,  True, …,  True,  True,  True],
        [ True,  True,  True, …,  True,  True,  True],
        [ True,  True,  True, …,  True,  True,  True],
        …,
        [ True,  True,  True, …,  True,  True,  True],
        [ True,  True,  True, …,  True,  True,  True],
        [ True,  True,  True, …,  True,  True,  True]],
  fill_value=65535,
  dtype=float32)
```

1.3 Vamos a visualizar y ver si hay alguna diferencia

[ ]:
```
fig, (ax1, ax2) = plt.subplots(1,2 ,figsize=(7,7))
ax1.imshow(data, vmin=193, vmax=313, cmap='Greys')
ax2.imshow(ds['CMI'].values, vmin=193, vmax=313, cmap='Greys')
```

[ ]:
```
<matplotlib.image.AxesImage at 0x7ff7e2f1c0d0>
```



1.3.1 ¿Hay alguna diferencia?

1.3.2 Vamos a darle proyección a la figura usando `cartopy`

[ ]:
```
import cartopy, cartopy.crs as ccrs
```

Convertimos los de Kelvin para Celsus

[ ]:
```
data = file.variables['CMI'][:] - 273.16
```

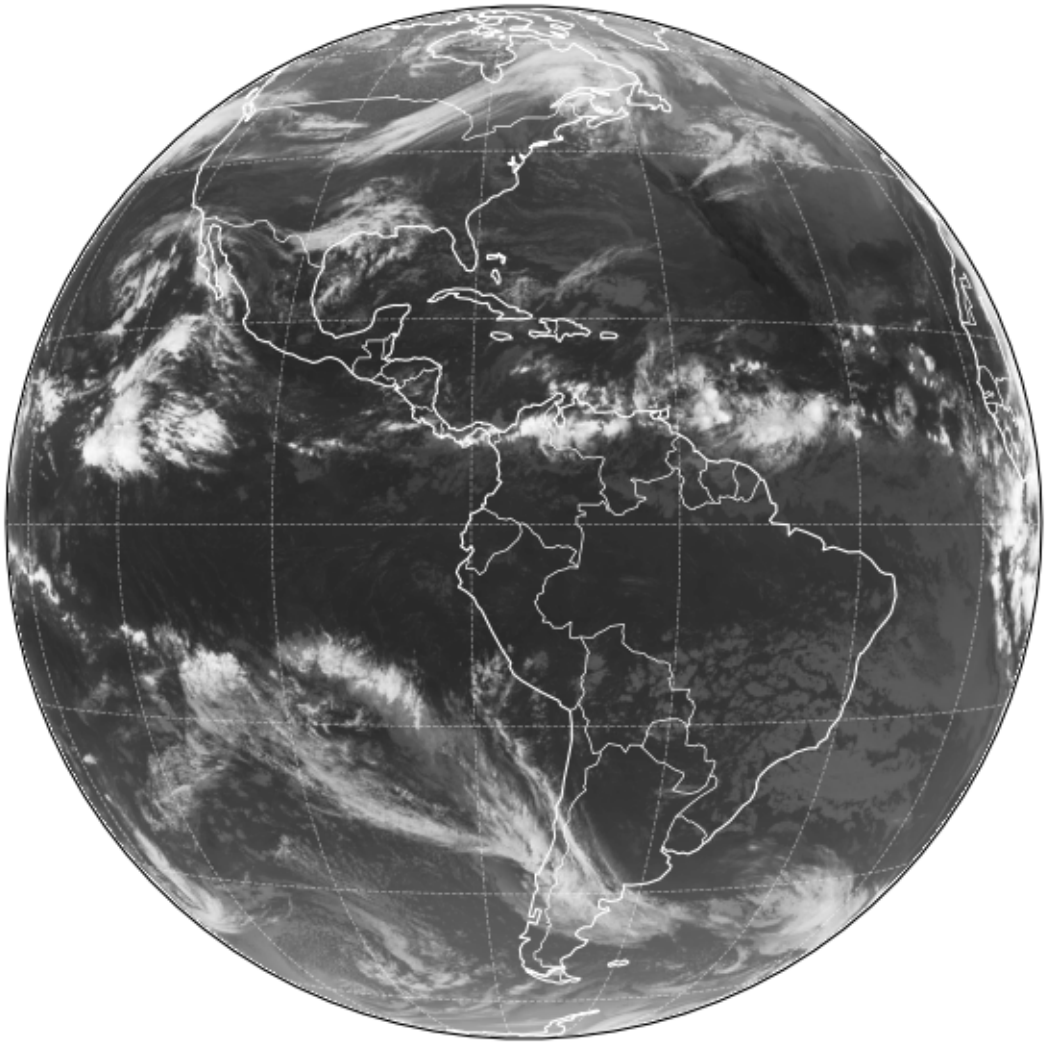
Recuerda que los datos del GOES-R están en una posición geoestacionaria. Los parámetros como longitud central y altura del satélite están en el metadato. Además el área que el satélite observa también están en el metadato.

[ ]:
```
# Elijamos el tamaño de la figura (ancho x alto, en pulgadas)
plt.figure(figsize=(7,7))

# Usamos la proyección geoestacionaria en cartopy
ax = plt.axes(projection=ccrs.Geostationary(central_longitude=-75.0,␣
 ↪satellite_height=35786023.0))
img_extent = (-5434894.67527,5434894.67527,-5434894.67527,5434894.67527)
```

```
# Agregar líneas de costa, bordes y líneas de cuadrícula
ax.coastlines(color='white', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='white', linewidth=0.5)
ax.gridlines(color='white', alpha=0.5, linestyle='--', linewidth=0.5)

# Agregamos la matriz de datos
img = ax.imshow(data, vmin=-80, vmax=40, origin='upper', extent=img_extent,␣
 ↪cmap='Greys')
```



1.4 Retornar al indice