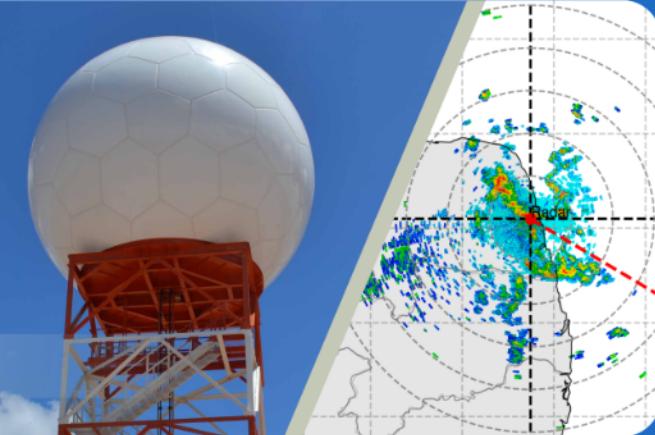


# Radar Meteorológico: Fundamentos e Processamento de Dados com Python

Atividade Pré-Curso:  
Introdução ao Google Colab

## CURSO: RADAR METEOROLÓGICO FUNDAMENTOS E PROCESSAMENTO DE DADOS COM PYTHON

Abril de 2024



Diego Souza  
[diego.souza@inpe.br](mailto:diego.souza@inpe.br)

DISSM - Divisão de Satélites e Sensores Meteorológicos  
CGCT - Coordenação Geral de Ciências da Terra  
INPE - Instituto Nacional de Pesquisas Espaciais

MINISTÉRIO DA  
CIÊNCIA, TECNOLOGIA  
E INOVAÇÃO

GOVERNO FEDERAL  
**BRASIL**  
UNIÃO E RECONSTRUÇÃO

# Executando Scripts Localmente ou na Nuvem

Anaconda Prompt (Miniconda3)

```
(base) D:\Users\dsouza>conda activate workshop
(base) D:\Users\dsouza>cd D:\VLAB\Python
(base) D:\VLAB\Python>python script_07.py
```

```

1  # INPE / CPTEC Training: NWP
2  # Author: Diego Souza
3
4  import pygrib
5  import matplotlib.pyplot as plt
6  import cartopy, cartopy.crs
7  import cartopy.io.shapereader
8  import numpy as np
9  import matplotlib
10
11
12
13  # Open the GRIB file
14  grib = pygrib.open("D:/VLAB/GRIB/ABI_L2_G16_202107060000.grib2")
15
16  # Select the temperature
17  grb = grib.select(name='2 metre temperature')[0]
18
19  # Get information from the file
20  init = str(grb.analDate) # Init date / time
21  run = str(grb.hour).zfill(2) # Run
22  ftime = str(grb.forecastTime) # Forecast hour
23  valid = str(grb.validDate) # Valid date / time
24  print('Init: ' + init + ' UTC')
25  print('Run: ' + run + 'Z')
26  print('Valid: ' + valid + ' UTC')
27
28  # Select the extent [min. lon, min. lat, max. lon, max. lat]
29  extent = [-93.0, -60.0, -25.0, 18.0]
30
31
32  # Read the data for a specific region
33  tmtmp, lata, lons = grb.data(lat1=extent[1],lat2=extent[3],
34  lon1=extent[0],lon2=extent[2])
35
36  # Convert from K to °C
37  tmtmp = tmtmp - 273.15
38
39
40  # Choose the plot size (width x height, in inches)
41  plt.figure(figsize=(8,8))
42
43  # Use the Cylindrical Equidistant projection in cartopy
44  ax = plt.axes(projection=ccrs.PlateCarree())
45
46  # Define the image extent
47  img_extent = [extent[0], extent[2], extent[1], extent[3]]
48
49  # Add a shapefile
50  # https://geofptpb.gov.br/organizacoes-do-territorio/marcadores
51  shapefile = list(shapereader.Reader("BR_UF_2019.shp").geometries())
52  ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor="black",
53  facecolor="none")
54
55  # Add coastlines, borders and gridlines
56  ax.coastlines(resolution='10m', color='black', linewidth=.5)
57  ax.add_feature(cartopy.feature.BORDERS, edgecolor='black',
58  facecolor='none', linewidth=.5)
59  ax.add_feature(cartopy.feature.GRID, ccrs.PlateCarree(), alpha=.5)
60
61  # Define the contour levels
62  data_min = 2
63  data_max = 36
64  interval = 2
65  levels = np.arange(data_min,data_max,interval)

```

CONDA

colab

Curso\_NWP\_202107.ipynb - Colab

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Todas as alterações foram salvas

índice

- Script 4: Adicionando Mapas com a Cartopy
- Script 5: Adicionando Shapefiles
- Script 6: Plotando Contornos e "Labels"
- Script 7: Paletas de Cores Personalizadas
- Script 8: Suavizando os Contornos
- Script 9: Trabalhando com Diversos Arquivos
- Script 10: Criando uma Animação
- Script 11: Médias, Máximos e Mínimos - Múltiplos Plots Simultâneos
- Script 12: Precipitação Instantânea e Precipitação Acumulada
- Script 13: Lendo Campos Especificando o Nível - Linhas de Corrente
- Script 14: Animando as Linhas de Corrente
- Script 15: Vetores de Vento
- Script 16: Barbelas
- Script 17: Plot 2 x 2 - Linhas de Corrente em 250, 500, 700 e 850 hPa
- Script 18: Lendo Diversos Campos em Diversos Níveis - Índice Galvez Davison (GDI)
- Script 19: Plot de Satélite
- Script 20: Plot de Modelo Numérico + Satélite (Exemplo 1)
- Script 21: Plot de Modelo Numérico + Satélite (Exemplo 2)
- Script 22: Plot de Modelo Numérico + Satélite (Exemplo 3)
- Script 23: Plot de Modelo Numérico + Satélite (Exemplo 4)
- Script 24: Plot de Modelo Numérico + Satélite (Exemplo 5)
- Script 25: Plot METAR
- Script 26: METAR + Modelos

```

# plt.colorbar(img3.lines, label='Wind Speed (kt)', extend='both', orientation='vertical', pad=0.03, fraction=0.05)
plt.colorbar(img1, label='Brightness Temperatures (°C)', extend='both', orientation='vertical', pad=0.03, fraction=0.05)

# Extract date
date = (datetime.strptime(datetime, "%Y-%m-%d %H:%M:%S.%fZ"))

# Add a title
plt.title('GOES-16 Band 09 ' + date.strftime('%Y-%m-%d %H:%M') + ' UTC' + ' + GFS Streamlines (250 hPa)', fontweight='bold')

# Save the image
plt.savefig(f'{output}/image_23.png', bbox_inches='tight', pad_inches=0, dpi=300)

# Show the image
plt.show()

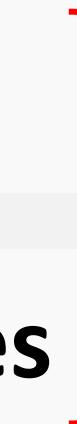
```

Downloading file /content/Samples/OR\_ABI-L2-CMIPF-M6C09\_G16\_s20211870000204\_e20211870009517\_c20211870010014.nc

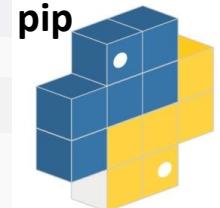
Init: 2021-07-06 00:00:00 UTC  
Run: 00Z  
Forecast: +0  
Valid: 2021-07-06 00:00:00 UTC

GOES-16 Band 09 2021-07-06 00:00 UTC + GFS Streamlines (250 hPa) Reg. [-93.0, -60.0, -25.0, 18.0]

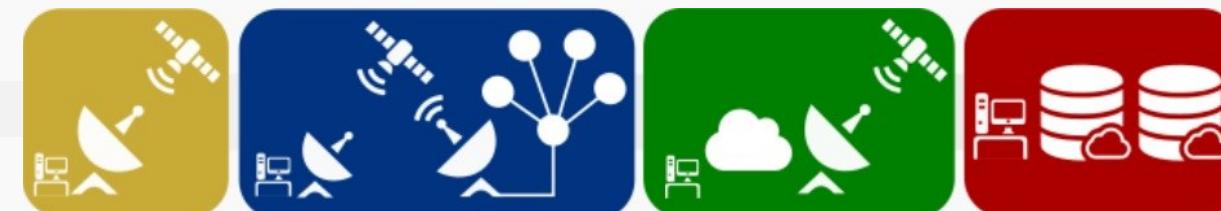
- Python
- Gerenciador de Pacotes
- Gerenciador de Ambientes



# CONDA



- Editor de Texto / IDE
- Dados de Interesse

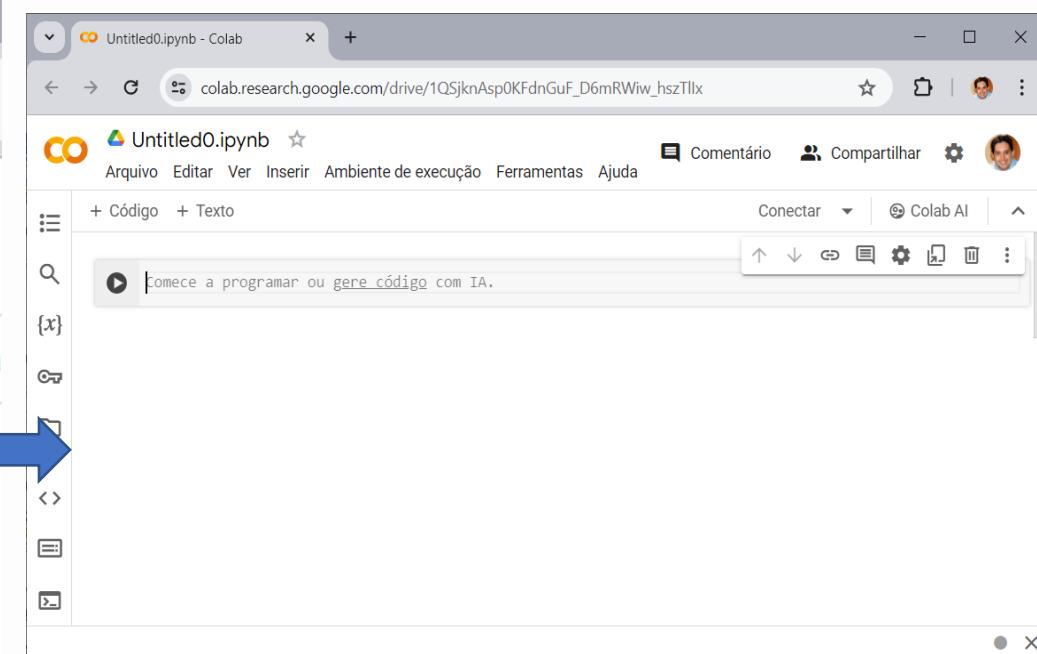
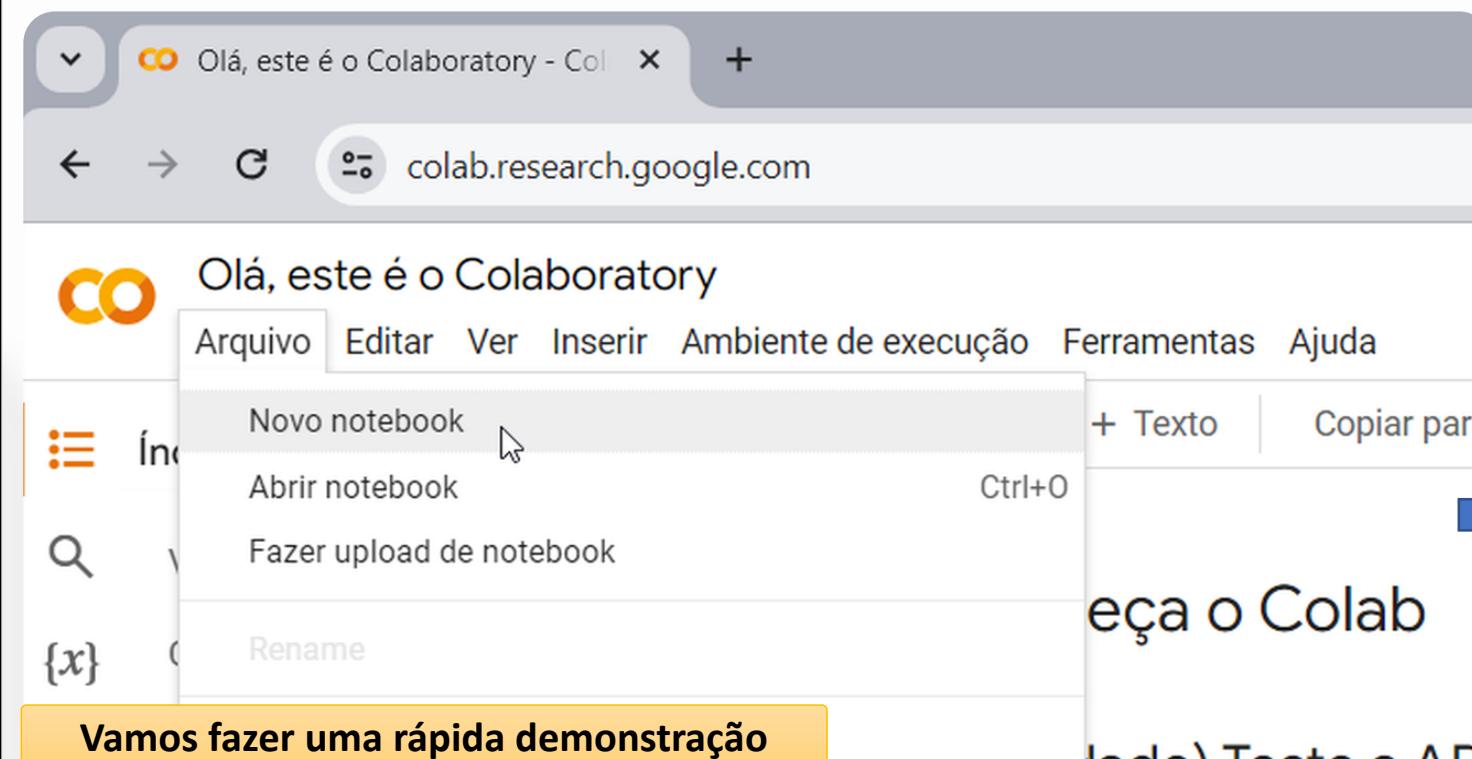


Diversos mecanismos disponíveis

# GOOGLE COLAB: CONCEITOS BÁSICOS

# Primeiros passos com o Google COLAB:

<https://colab.research.google.com/>



# Executando Scripts na Nuvem

**Processamento GOES-R**

GOES-16 DCP RGB 2020-08-05 15:00 UTC  
Reg.: [-100.0, 0.0, -40.0, 40.0]

Contact: Introduction Step 1: Checking the Virtual Machine Configuration Step 2: Installing the Required Libraries Step 3: Downloading Ancillary Files Script 1: Basic Plot / Reading Pixel Values Script 2: Basic Operation / Colorbar / Title / Date Script 3: Overlaying Maps with Cartopy Script 4: Overlaying Maps with Cartopy (Reading the Metadata) Script 5: Reading a Shapefile Script 6: ABI + GLM (Basic Plot) Script 7: Creating RGB Composites Script 8: Custom Colormaps - Enhancing IR Channels Script 9: Downloading Data from the Cloud - Amazon Web Services (AWS) Script 10: Downloading Data from the Cloud - AWS (Using a Function) Script 11: Downloading Data from the Cloud - AWS (Importing a Function from 'utilities.py') Script 12: Cropping a Full Disk Image Script 13: Cropping a Full Disk Image and Preprocessing with GEOTiff Commerciantes

**Dados de Modelos de Previsão Numérica do Tempo**

GOES-16 Band 13 2023-04-10 00:00 UTC + GFS PMSL (hPa) + 1000-500 hPa Thickness  
Reg.: [-100.0, 0.0, -40.0, 40.0]

Contact: Diego Souza - INPE / CGCT / DISMM

Table of contents

- Script 4: Overlaying Maps with Cartopy
- Script 5: Reading a Shapefile
- (x) Script 6: Plotting Contours and Labels
- Script 7: Custom Color Palette
- Script 8: Smoothing Contours
- Script 9: Working with Multiple Files
- Script 10: Creating an Animation
- Averages, Maximums and Minimums - Simultaneous Multiple Plots
- Script 12: Instant Precipitation and Accumulated Precipitation
- Script 13: Reading Fields by Specifying Levels - Streamlines
- Script 14: Animating Streamlines
- Script 15: Wind Vectors
- Script 16: Barbs
- Script 17: 2 x 2 Plot - Streamlines at 250, 500, 700 and 850 hPa
- Script 18: Reading Multiple Fields at Multiple Levels - Galerkin Division Index (ODI)
- Script 19: Satellite Imagery Plot
- Script 20: NWP + Satellite (Example 1)
- Script 21: NWP + Satellite (Example 2)
- Script 22: NWP + Satellite (Example 3)
- Script 23: NWP + Satellite (Example 4)
- Script 24: NWP + Satellite - RGB Composites (Example 5)
- Script 25: METAR (NWP)
- Script 26: METAR + NWP
- Script 27: METAR + NWP + Satellite

**Mais de 100 scripts exemplo!**

**Produtos Derivados**

Sfc - 850 mb  
Reg.: [-100.0, 0.0, -40.0, 40.0]

Contact: Diego Souza - INPE / CGCT / DISMM

Introduction Step 1: Checking the Virtual Machine Configuration Step 2: Installing the Required Python Libraries Step 3: Downloading Samples and Ancillary Files Demonstration 1: GFS4-Advection Layered Persistence (ALPW)  
Ex 1 (ALPW) Step 1: Importing Required Libraries and Making a Basic Plot  
Ex 1 (ALPW) Step 2: Retrieving Pixels for a Specific Region (Min. lon, Min. lat, Max. lon, Max. lat)  
Ex 1 (ALPW) Step 3: Convert the Desired Extent (degrees) to Mercator (using Python)  
Ex 1 (ALPW) Step 4: Retrieving the Time and Date  
Ex 1 (ALPW) Step 5: Final Plot Demonstration 2: NOAA Coral Reef Watch SST (NetCDF)  
Ex 2 (SST) Step 1: Importing Required Libraries and Making a Basic Plot  
Ex 2 (SST) Step 2: Retrieving Pixels for a Specific Region (Min. lon, Min. lat, Max. lon, Max. lat)  
Ex 2 (SST) Step 3: Retrieving the Time and Date  
Ex 2 (SST) Step 4: Final Plot Demonstration 3: NOAA Coral Reef Watch 5 km SST Anomaly (NetCDF)  
Ex 3 (SST Anomaly) Step 1: Importing Required Libraries and Making a Basic Plot  
Ex 3 (SST Anomaly) Step 2: Retrieving Pixels for a Specific Region (Min. lon, Min. lat, Max. lon, Max. lat)  
Ex 3 (SST Anomaly) Step 3: Retrieving the Time and Date

**Dados de Oceanografia**

NOAA Coral Reef Watch Daily 5 km SST - 2022-01-01  
Reg.: [-100.0, 0.0, -40.0, 40.0]

Sea Surface Temperature (°C)

Contact: NOAA Coral Reef Watch

# Show the image  
plt.show()

# Vamos Utilizar o Seguinte Notebook Exemplo

<https://colab.research.google.com/drive/1zg9ugxZ3CjxtxNf4qZwpXtQ43PehG-Cx?usp=sharing>



The screenshot shows a Google Colab notebook interface. The title bar reads "INPE\_GOES-16\_EPGMET\_2023.ipynb". The left sidebar contains a table of contents with sections like "Índice", "Introdução", and various steps for processing GOES-16 satellite data. The main content area features a banner for the "XXII EPGMET" conference, which took place from October 17 to 20, 2023, at CPTEC, INPE - Cachoeira Paulista/SP. The banner includes logos for INPE, VLab, and TATHU, and mentions the use of Python and the TATHU package. Below the banner, text credits Diego Souza for development and notes the notebook version as 16 de outubro de 2023. The notebook content starts with an "Introdução" section, welcoming users to the "Processamento de Dados de Satélites com Python" notebook.

# Executando Scripts na Nuvem

## Crie uma cópia no seu próprio Google Drive

MUITO  
IMPORTANTE  
Crie uma cópia  
no seu Google  
Drive

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Todas as alterações foram salvas

Localizar no Drive Abrir no modo Playground Novo notebook Abrir notebook Fazer upload de notebook Rename Mover Mover para a lixeira Salvar uma cópia no Drive Salvar uma cópia como Gist do GitHub Salvar uma cópia no GitHub Salvar Salvar e fixar revisão Histórico de revisões Fazer download Imprimir Adicionando Mapas com a Cartopy Ctrl+S Ctrl+M S Ctrl+P Ctrl+P



## Execute as instruções em cada célula, sequencialmente

UM BOM ATALHO  
PARA EXECUTAR  
CÉLULAS DE CÓDIGO:  
**Ctrl + Enter**

### Verificando as Configurações da Máquina Virtual

Neste passo verificamos as configurações da máquina virtual.

**OBSERVAÇÃO:**  
Mesmo que você execute várias células, o sistema espera que elas sejam concluídas em sequência

# checking the OS Installed  
!cat /etc/issue  
!uname -a  
print('\n')

# available RAM  
!grep MemTotal /proc/meminfo  
print('\n')

# available HDD Space  
!df -h  
print('\n')

# python Installation Directory  
!which python

# Executando Scripts na Nuvem

INPE\_GOES-16\_EPGMET\_2023.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Todas as alterações foram salvas

Índice

+ Código + Texto

Adiciona células de texto e de código

Instalando as Bibliotecas Necessárias e Download de Arquivos Auxiliares

Neste passo, instalaremos as bibliotecas necessárias para executar os scripts e baixamos algumas amostras de imagens GOES-16:

Bibliotecas Python:

- **NetCDF4**: Leitura dos valores de pixel de arquivos NetCDF4.
- **Cartopy**: Adiciona mapas aos plots.

\*Imagens de amostra:

- OR\_ABI-L2-CMIPF-M6C\_G16\_s\_e\_c.nc: Amostra GOES-16(sensor ABI) - FULL DISK
- OR\_ABI-L2-CMIPC-M6C\_G16\_s\_e\_c.nc: Amostra GOES-16(sensor ABI) - CONUS
- OR\_ABI-L2-CMIPM1-M6C\_G16\_s\_e\_c.nc: Amostra GOES-16(sensor ABI) - MESOESCALA 1

+ Código + Texto

```
[ ] # installing the NetCDF4 library  
!pip install netcdf4  
print('\n')  
  
# installing the Cartopy library  
!pip install cartopy  
!pip install shapely --no-binary shapely --force  
print('\n')  
  
# Create the 'samples' directory
```

# Executando Scripts na Nuvem

INPE\_GOES-16\_EPGMET\_2023.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Todas as alterações foram salvas

Índice

Introdução

Verificando as Configurações da Máquina Virtual

Instalando as Bibliotecas Necessárias e Download de Arquivos Auxiliares

Tabela de Referência: Canais e Comprimentos de onda do Sensor ABI

Importando as Bibliotecas Python Necessárias

Abrindo um Arquivo GOES-16 com a Biblioteca NetCDF4 e Verificando os "Datasets" Disponíveis

Lendo e Plotando o Dataset CMI (Cloud and Moisture Imagery)

Lendo da Data e Hora da Aquisição, Adicionando um Título e uma "Colorbar"

Adicionando Mapas com a Cartopy

Plotando uma imagem CONUS

Plotando uma Imagem MESOESCALA

+ Seção

Índice

## Células de Texto

### Verificando as Configurações da Máquina Virtual

Neste passo verificamos as configurações da máquina virtual e a versão do Python pré instalada.

```
# checking the OS Installed  
!cat /etc/issue  
!uname -a  
print('\n')  
  
# available RAM  
!grep MemTotal /proc/meminfo  
print('\n')  
  
# available HDD Space  
!df -h  
print('\n')  
  
# python Installation Directory  
!which python  
print('\n')  
  
# python Version Installed  
!python --version  
print('\n')
```

- Comandos iniciados com “!”: Comandos Shell (Linux)  
- Todos os outros comandos: Python

## Memória e Armazenamento

## Células de Código

## Saída da Célula de Código

Ubuntu 22.04.3 LTS \n \l  
Linux deb5377749fa 6.1.58+ #1 SMP PREEMPT\_DYNAMIC Sat Nov 18 15:31:17 UTC 2023 x86\_64 x86\_64 x86\_64 GNU/Linux  
MemTotal: 13290480 kB

# Executando Scripts na Nuvem

## Limpar todas as células de saída

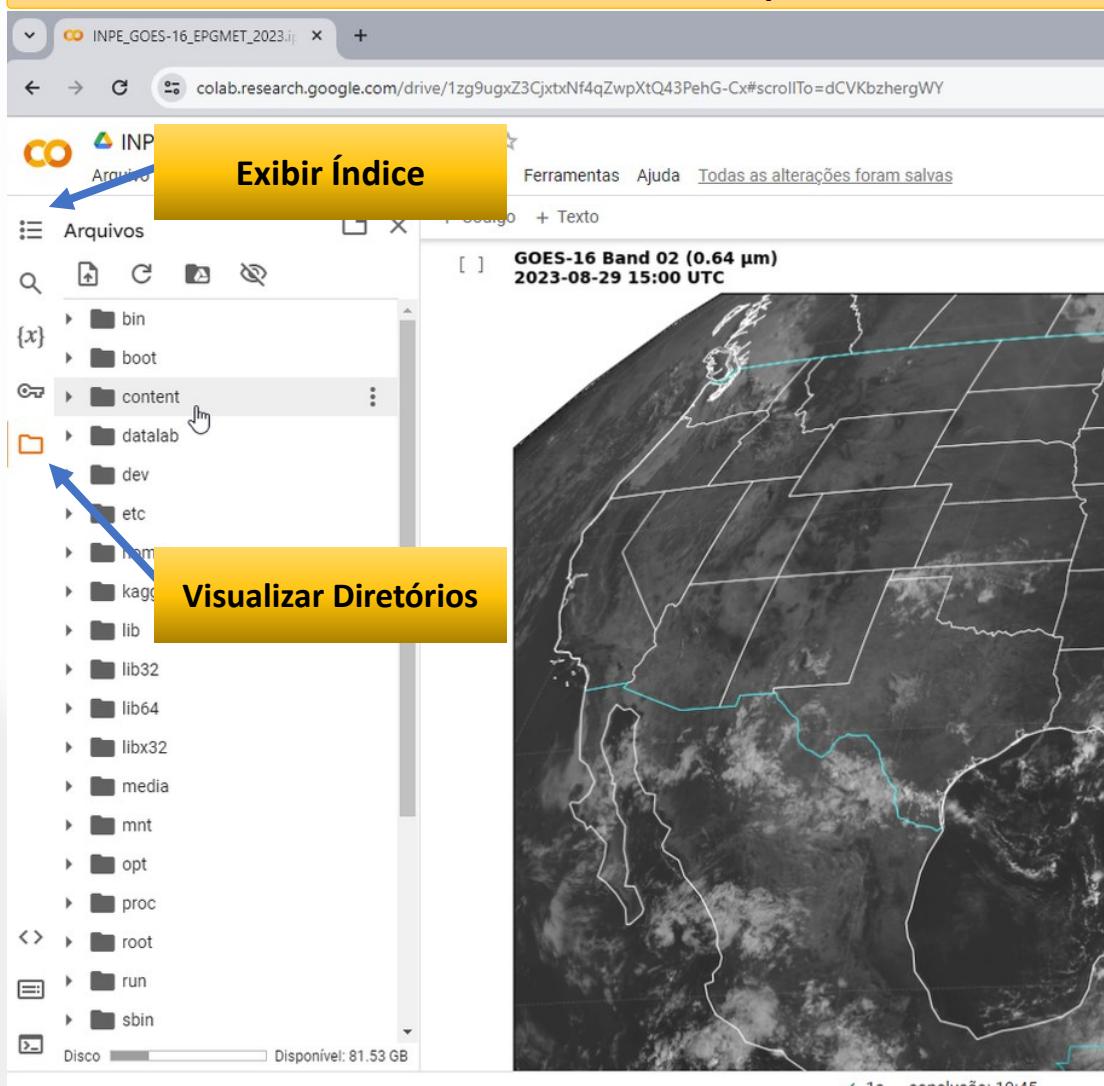
## Reiniciando o ambiente de execução

**Reseta o Python (por exemplo: variáveis), mantendo os outros aspectos da máquina (por exemplo, bibliotecas)**

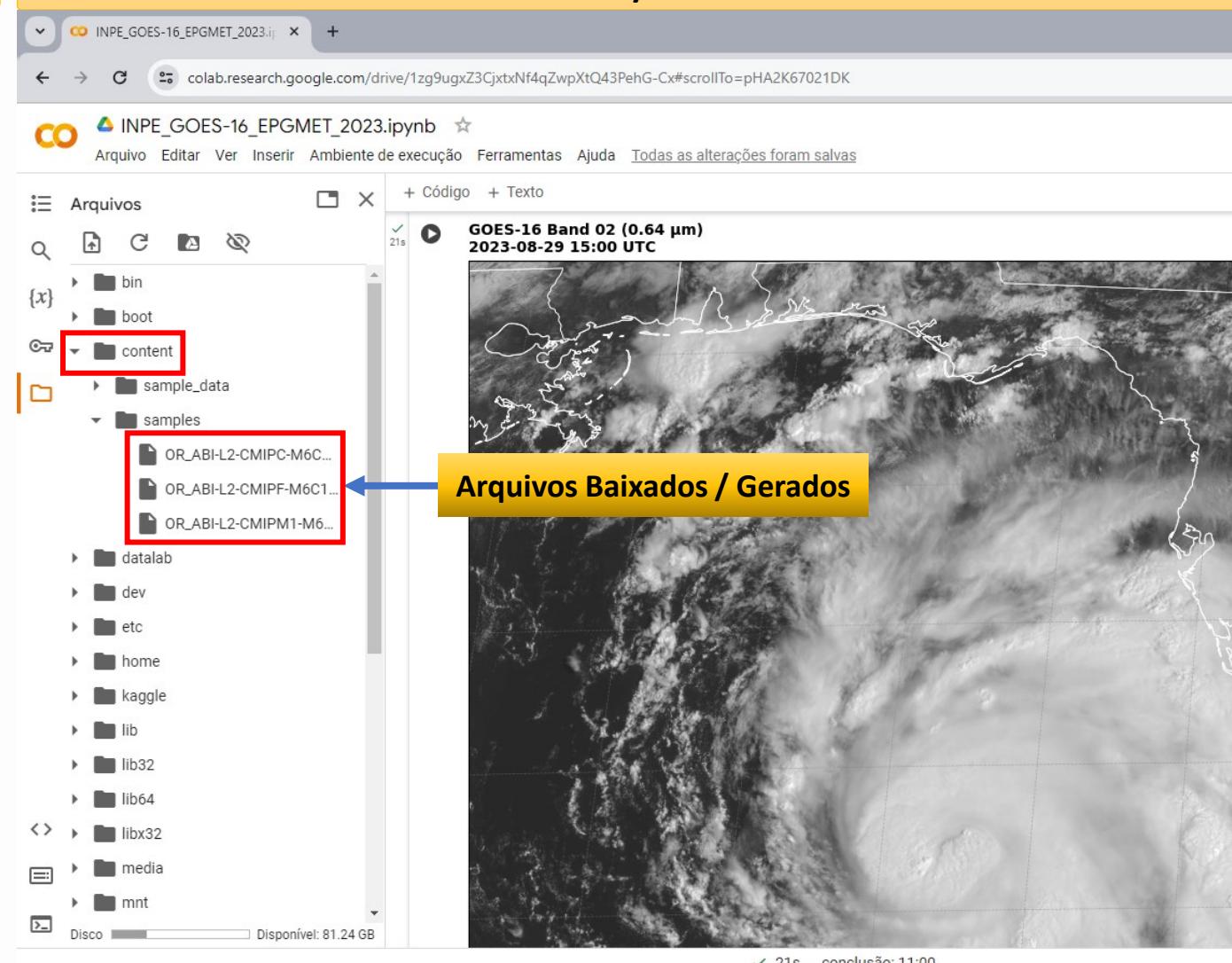
**Reseta a máquina para o estado inicial, incluindo as bibliotecas**

# Executando Scripts na Nuvem

Diretório de Trabalho COLAB: “/content”



Diretório “/content”



**Os slides a seguir são apenas para referência. Não é necessário compreender o processamento de dados de satélites para o treinamento sobre radares.**

# Instalando Bibliotecas e Baixando Amostras

Vamos explorar a versão curta do nosso notebook Colab

<https://colab.research.google.com/drive/1o8YrTMT9pcJFNjSZ9Hrc2iwAEZjqR30q?usp=sharing>

```
# installing the NetCDF4 library
!pip install netcdf4 ← Instala a biblioteca NetCDF4
print('\n')

# installing the Cartopy library
!pip install cartopy ← Instala a biblioteca Cartopy
!pip install shapely --no-binary shapely --force
print('\n')

# Create the 'samples' directory
!mkdir -p samples # where we will store the GOES-R samples ← Cria uma pasta chamada "samples" (para armazenar amostras)
print('\n')

# download some GOES-16 samples from Amazon Web Services (AWS) with wget
# note: the links were taken from the following page: https://home.chpc.utah.edu/~u0553130/Brian\_Blaylock/cgi-bin/goes16\_download.cgi

# Band 13 (10.3μ) - August 29, 15:00 UTC (Full Disk)
!wget -c https://noaa-goes16.s3.amazonaws.com/ABI-L2-CMIPF/2023/241/15/OR\_ABI-L2-CMIPF-M6C13\_G16\_s20232411500206\_e20232411509526\_c20232411509585.nc -P /content/samples/

# Band 02 (0.64μ) - August 29, 15:00 UTC (CONUS)
!wget -c https://noaa-goes16.s3.amazonaws.com/ABI-L2-CMIPC/2023/241/15/OR\_ABI-L2-CMIPC-M6C02\_G16\_s20232411501172\_e20232411503545\_c20232411504026.nc -P /content/samples/

# Band 02 (0.64μ) - August 29, 15:00 UTC (MESOSCALE 1)
!wget -c https://noaa-goes16.s3.amazonaws.com/ABI-L2-CMIPM/2023/241/15/OR\_ABI-L2-CMIPM1-M6C02\_G16\_s20232411500279\_e20232411500336\_c20232411500395.nc -P /content/samples/
```

Baixa dados GOES-R da Amazon Web Services (AWS)

# Informações dos Canais ABI

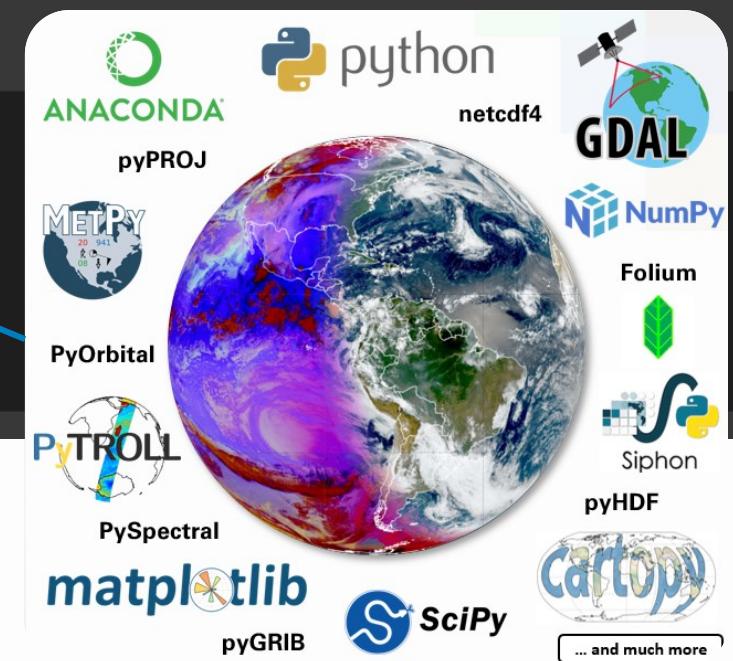
Para sua referência, esta é uma tabela com todas as bandas do sensor ABI, comprimentos de onda centrais, resoluções e descrição:

# Canal	$\lambda$ central ( $\mu\text{m}$ )	Resolução nominal (km)	Faixa Espectral	Descrição/"nome"
1	0,47	1	Visível	Azul
2	0,64	0,5	Visível	Vermelho
3	0,86	1	IV próximo	Vegetação
4	1,37	2	IV próximo	Cirrus
5	1,6	1	IV próximo	Fase
6	2,2	2	IV próximo	Tamanho de Partículas
7	3,9	2	IV onda curta	Janela de Onda Curta
8	6,2	2	IV onda média	Vapor d'água: Níveis Altos
9	6,9	2	IV onda média	Vapor d'água: Níveis Médios
10	7,3	2	IV onda média	Vapor d'água: Níveis Baixos
11	8,4	2	IV onda longa	Fase
12	9,6	2	IV onda longa	Ozônio
13	10,3	2	IV onda longa	Janela de Onda Longa "limpa"
14	11,2	2	IV onda longa	Janela de Onda Longa
15	12,3	2	IV onda longa	Janela de Onda Longa "Suja"
16	13,3	2	IV onda longa	$\text{CO}_2$

# Instalando as Bibliotecas Necessárias

Neste passo vamos importar todas as bibliotecas necessárias para esta demonstração: "netCDF4" para abrir arquivos e ler valores de pixels, "matplotlib" para gerar imagens, "datetime" para ler a data e hora de aquisição, "cartopy" para adicionar mapas e "numpy" para realizar cálculos diversos.

```
[ ] # required modules
from netCDF4 import Dataset      # read / write NetCDF4 files
import matplotlib.pyplot as plt   # plotting library
from datetime import datetime     # basic dates and time types
import cartopy, cartopy.crs as ccrs # plot maps
import numpy as np                # scientific computing with Python
```



## Nomenclatura dos arquivos GOES-R

OR\_ABI-L2-CMIPF-M6C13\_G16\_s20232411500206\_e20232411509526\_c20232411509585.nc

<Operational System Real-Time Data>\_<sensor>-<level>-<short product name>-M<operation mode><channel>-G<GOES>-s<start time of scan:year-julianday-hh:mm:ssUTC>/\_e<end time of scan:year-julianday-hh:mm:ssUTC>/\_c< data creation time ><year-julianday-hh:mm:ssUTC>/nc

Primeiro, vamos criar um objeto chamado "file" (você pode chamá-lo como quiser) e abrir uma imagem GOES-16 com o comando "Dataset" da biblioteca NetCDF4:

```
[ ] # open the GOES-R NetCDF file  
file = Dataset("samples/OR_ABI-L2-CMIPF-M6C13_G16_s20232411500206_e20232411509526_c20232411509585.nc")
```

Como veremos, temos vários datasets dentro de um arquivo NetCDF (com informações muito úteis!). Com o comando "variables.keys()", podemos listar todos os datasets disponíveis.

```
[ ] # listing the key NetCDF variables  
file.variables.keys()
```

<https://www.goes-r.gov/products/baseline-cloud-moisture-imagery.html>

```
dict_keys(['CMI', 'DQF', 't', 'y', 'x', 'time_bounds', 'goes_imager_projection', 'y_image', 'y_image_bounds', 'x_image', 'x_image_bounds',  
'nominal_satellite_subpoint_lat', 'nominal_satellite_subpoint_lon', 'nominal_satellite_height', 'geospatial_lat_lon_extent', 'band_wavelength', 'band_id',  
'total_number_of_points', 'valid_pixel_count', 'outlier_pixel_count', 'min_brightness_temperature', 'max_brightness_temperature',  
'mean_brightness_temperature', 'std_dev_brightness_temperature', 'planck_fk1', 'planck_fk2', 'planck_bc1', 'planck_bc2',  
'algorithm_dynamic_input_data_container', 'percent_uncorrectable_GRB_errors', 'percent_uncorrectable_L0_errors', 'earth_sun_distance_anomaly_in_AU',  
'processing_parm_version_container', 'algorithm_product_version_container', 'esun', 'kappa0', 'focal_plane_temperature_threshold_exceeded_count',  
'maximum_focal_plane_temperature', 'focal_plane_temperature_threshold_increasing', 'focal_plane_temperature_threshold_decreasing',  
'channel_integration_time', 'channel_gain_field'])
```

# Verificando as Características do Dataset “CMI”

Vamos verificar as características do dataset "CMI" (ou "Cloud and Moisture Imagery", como foi atribuído pela NOAA). De acordo com a documentação do produto, através deste dataset podemos ler o "fator de refletância / reflectance factor" (nas bandas visíveis) e as "temperaturas de brilho / brightness temperatures" para os canais infravermelhos:

<https://www.goes-r.gov/products/baseline-cloud-moisture-imagery.html>

```
[ ] # reading a specific variable
file.variables['CMI']

<class 'netCDF4._netCDF4.Variable'>
int16 CMI(y, x)
    _FillValue: -1
    long_name: ABI L2+ Cloud and Moisture Imagery brightness temperature
    standard_name: toa_brightness_temperature
    _Unsigned: true
    sensor_band_bit_depth: 12
    valid_range: [ 0 4095]
    scale_factor: 0.06145332
    add_offset: 89.62
    units: K
    resolution: y: 0.000056 rad x: 0.000056 rad
    coordinates: band_id band_wavelength t y x
    grid_mapping: goes_imager_projection
    cell_methods: t: point area: point
    ancillary_variables: DQF
unlimited_dimensions:
current shape = (5424, 5424)
filling on
```

Neste exemplo (canal 13 -  
10.3 um) temos 5424  
linhas x 5424 colunas e a  
unidade é Kelvin

Note que para esse canal em particular (Banda 13 - 10.3 μm - 2 km - Full Disk) temos 5424 linhas x 5424 colunas. Para bandas de 1 km Full Disk teríamos o dobro, e para a banda de 0.5 km Full Disk teríamos 4 vezes mais (um plot da banda de 0.5 km requer um bom poder computacional). Veja também que a unidade de pixel para este canal em particular é Kelvin.

Considerando isso, vamos ler nosso dataset CMI e criar um plot preliminar.

# Lendo e Plotando o Dado CMI

Primeiro, vamos ler os valores de pixels do dataset CMI, converter os pixels para °C e armazenar todos os valores dessa matriz 2D (5424x5424) na variável "data" (novamente, você pode chamá-la do que quiser). Vamos usar essa variável para plotar uma imagem, mas primeiro precisamos converter os pixels de brilho em uma matriz 2d, portanto para utilizá-la para realizar qualquer operação (cálculo etc).

Nome da variável

Dataset CMI

Toda a matriz 2D

```
[ ] # get the pixel values  
data = file.variables['CMI'][:] - 273.15
```

Converte de K para °C

Agora vamos definir o tamanho da imagem (em polegadas) e plotar a variável "data" em uma escala de cinza chamada "Greys" (que vai de preto para branco, com preto representando valores baixos de temperatura de brilho (BT) e preto para valores altos de temperatura de brilho) usando o comando "imshow".

Vamos passar alguns parâmetros para o comando "imshow":

- "data": a variável que desejamos plotar.
- "vmin" e "vmax": os valores mínimos e máximos da paleta de cores (ou "colormap"). Você pode ajustar estes valores de acordo com suas preferências.
- "cmap": a paleta de cores a ser utilizada. Acesse o seguinte link para verificar as paletas de cores padrão da matplotlib: <https://matplotlib.org/stable/tutorials/colors/colormaps.html>

```
[ ] # choose the plot size (width x height, in inches)  
plt.figure(figsize=(20,20))
```

Tamanho do plot (largura x altura)

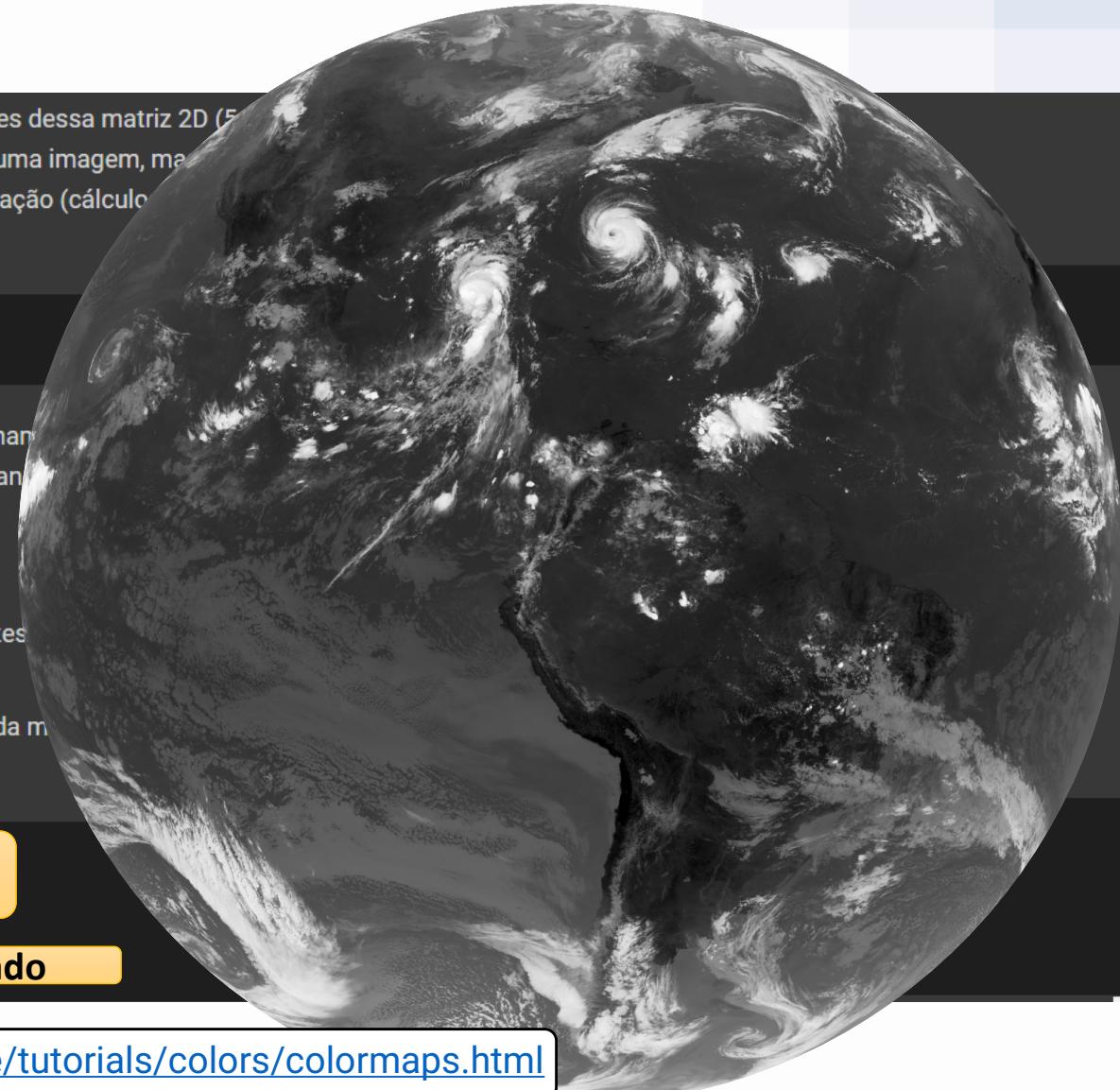
```
# plot the image  
plt.imshow(data, vmin=-80, vmax=40, cmap='Greys')
```

"Colormap" utilizado

Valor mínimo da colormap

Valor máximo da colormap

<https://matplotlib.org/stable/tutorials/colors/colormaps.html>



# Lendo o Período da Aquisição (Início do Scan)

## 1 - Lendo a data e hora do arquivo

```
file.getncattr('time_coverage_start')
```

```
'2023-08-29T15:00:20.6Z'
```

## 2 - Informando a biblioteca “datetime” a formatação da data / hora

```
# read the time/date from the NetCDF file metadata as a string
date_string = file.getncattr('time_coverage_start')

# how this time/date is formatted (using the timedelta convention)
date_format = '%Y-%m-%dT%H:%M:%S.%fZ'

# create the datetime object
date_obj = datetime.strptime(date_string, date_format)
print(date_obj)
```

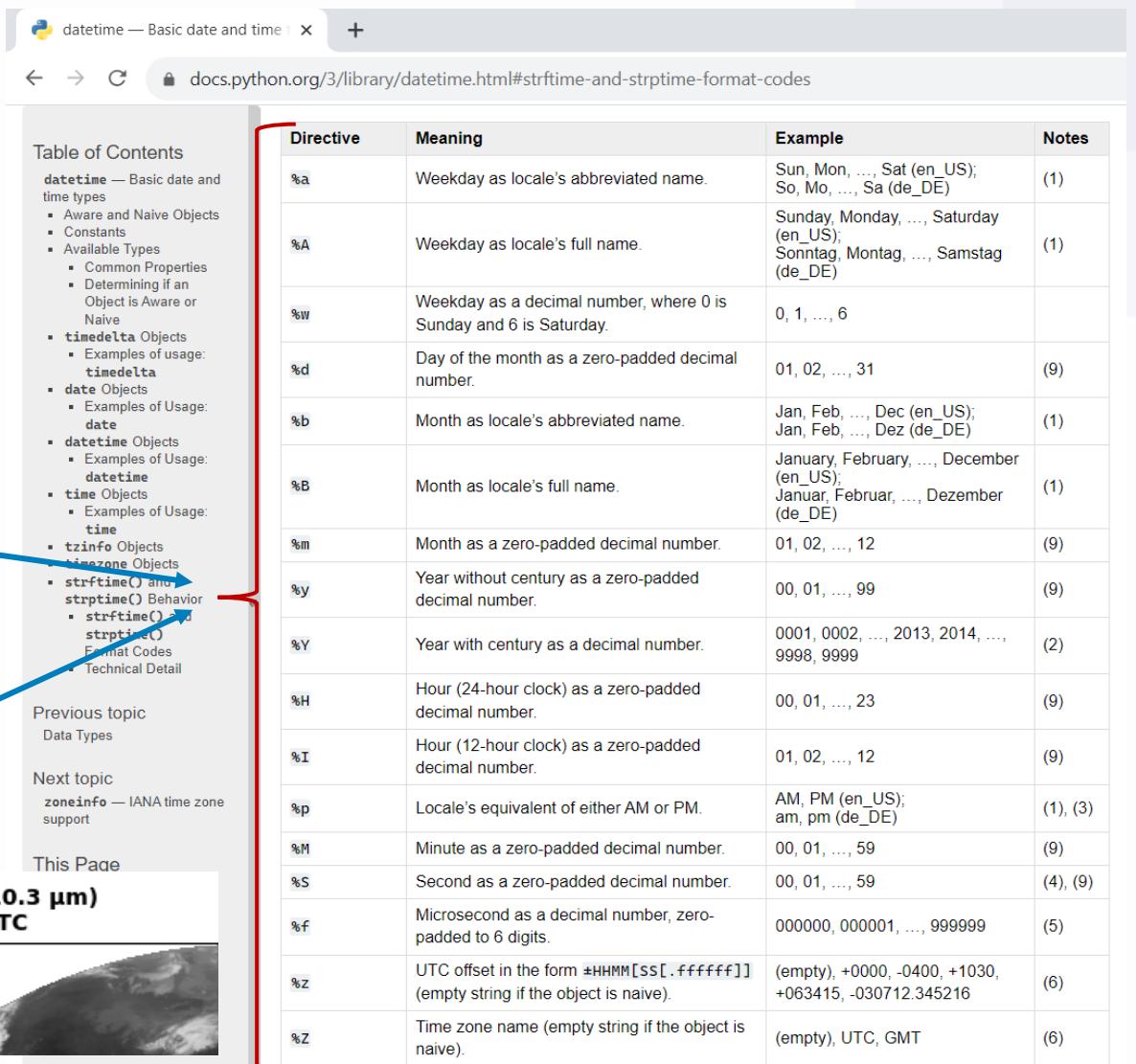
```
2023-08-29 15:00:20.600000
```

## 3 - Como desejamos formatar e mostrar

```
date = date_obj.strftime('%Y-%m-%d %H:%M UTC')
print(date)
```

```
2023-08-29 15:00 UTC
```

<https://docs.python.org/3/library/datetime.html#strftime-and-strptime-format-codes>



The table lists various format codes for the `strftime()` and `strptime()` methods:

Directive	Meaning	Example	Notes
%a	Weekday as locale's abbreviated name.	Sun, Mon, ..., Sat (en_US); So, Mo, ..., Sa (de_DE)	(1)
%A	Weekday as locale's full name.	Sunday, Monday, ..., Saturday (en_US); Sonntag, Montag, ..., Samstag (de_DE)	(1)
%w	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	0, 1, ..., 6	
%d	Day of the month as a zero-padded decimal number.	01, 02, ..., 31	(9)
%b	Month as locale's abbreviated name.	Jan, Feb, ..., Dec (en_US); Jan, Feb, ..., Dez (de_DE)	(1)
%B	Month as locale's full name.	January, February, ..., December (en_US); Januar, Februar, ..., Dezember (de_DE)	(1)
%m	Month as a zero-padded decimal number.	01, 02, ..., 12	(9)
%y	Year without century as a zero-padded decimal number.	00, 01, ..., 99	(9)
%Y	Year with century as a decimal number.	0001, 0002, ..., 2013, 2014, ..., 9998, 9999	(2)
%H	Hour (24-hour clock) as a zero-padded decimal number.	00, 01, ..., 23	(9)
%I	Hour (12-hour clock) as a zero-padded decimal number.	01, 02, ..., 12	(9)
%p	Locale's equivalent of either AM or PM.	AM, PM (en_US); am, pm (de_DE)	(1), (3)
%M	Minute as a zero-padded decimal number.	00, 01, ..., 59	(9)
%S	Second as a zero-padded decimal number.	00, 01, ..., 59	(4), (9)
%f	Microsecond as a decimal number, zero-padded to 6 digits.	000000, 000001, ..., 999999	(5)
%z	UTC offset in the form #HHMM[ss].[fffffff]	(empty), +0000, -0400, +1030, +063415, -030712.345216	(6)
%Z	Time zone name (empty string if the object is naive).	(empty), UTC, GMT	(6)



# Adicionando Títulos e Colorbar

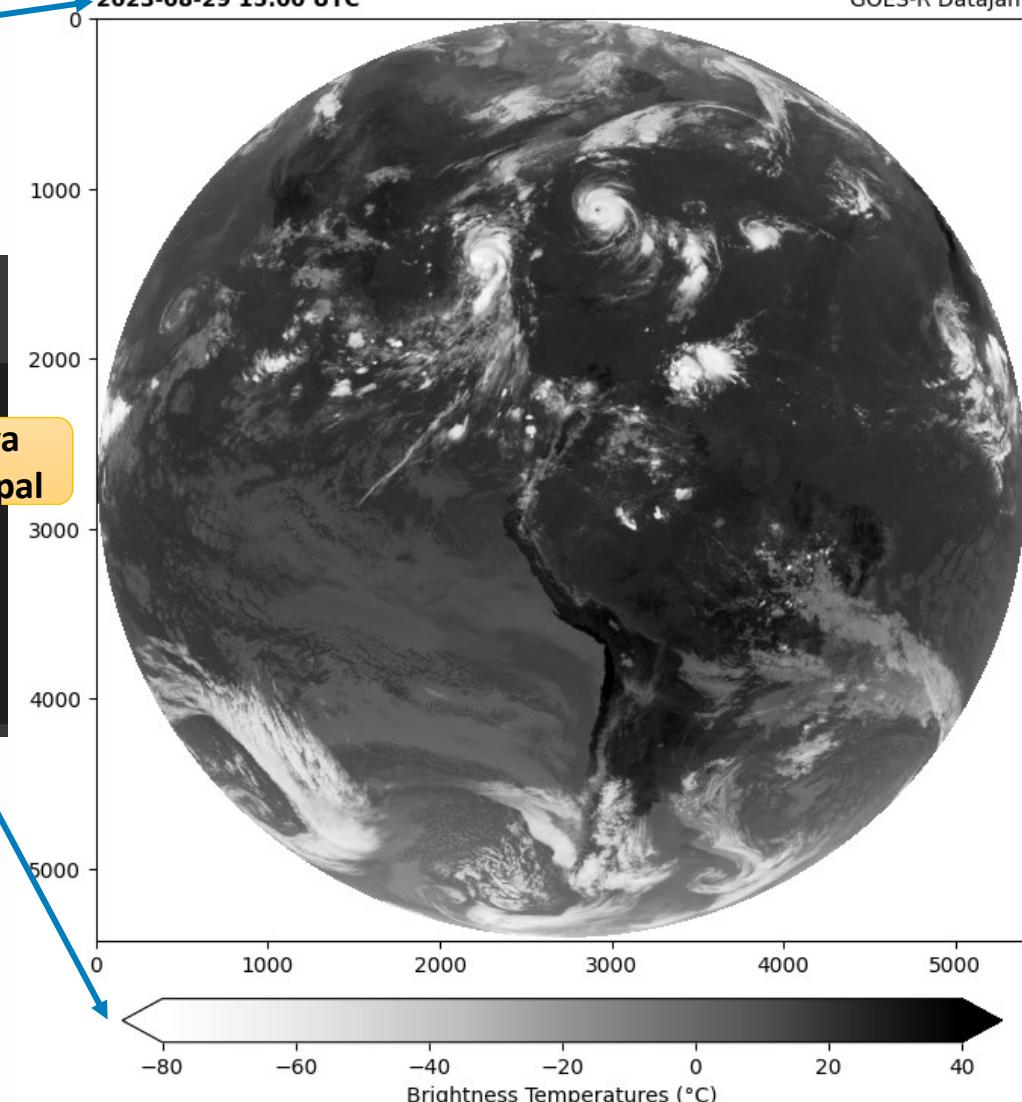
Como desejamos formatar a data e hora

```
date = date_obj.strftime('%Y-%m-%d %H:%M UTC')  
print(date)
```

2023-08-29 15:00 UTC

GOES-16 Band 13 (10.3 µm)  
2023-08-29 15:00 UTC

GOES-R Datajam



Agora que já conhecemos como ler e formatar a data e hora de aquisição, vamos criar um plot adicionando a string com esta informação em um título. Também vamos adicionar uma barra de cores com o comando "colorbar":

```
# choose the plot size (width x height, in inches)  
plt.figure(figsize=(10,10))  
  
# plot the image  
plt.imshow(data, vmin=-80, vmax=40, cmap='Greys')  
  
# add a colorbar  
plt.colorbar(label='Brightness Temperatures (°C)', extend='both', orientation='horizontal', pad=0.05, shrink=0.75)  
  
# add a title (one on the left and one on the right)  
plt.title(f'GOES-16 Band 13 (10.3 µm)\n{date}', fontweight='bold', fontsize=10, loc='left')  
plt.title('GOES-R DataJam', fontsize=10, loc='right')
```

Adiciona um título à esquerda

Adiciona outro título à direita

Título da colorbar

Estende a barra de cores para qual lado?

colorbar

Multiplicador de tamanho

Distância relativa (%) ao plot principal

# Adicionando Mapas com a Cartopy

```
# read the satellite longitude from the metadata  
longitude = file.variables['goes_imager_projection'].longitude_of_projection_origin  
  
# read the satellite altitude from the metadata  
height = file.variables['goes_imager_projection'].perspective_point_height  
  
# use the Geostationary projection in cartopy  
ax = plt.axes(projection=ccrs.Geostationary(central_longitude=longitude, satellite_height=height))  
  
# add coastlines, borders and gridlines  
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)  
ax.coastlines(resolution='110m', color='black', linewidth=0.8)  
ax.gridlines(color='black', alpha=0.5, linestyle='--', linewidth=0.5)
```

```
<cartopy.mpl.gridliner.Gridliner at 0x7879b537e170>
```



Lê a longitude do satélite através dos metadados

Lê a altitude do satélite através dos metadados

Informa à Cartopy a projeção do nosso dado -  
“geostationary” nesse caso, passando a longitude central  
e altitude do satélite como parâmetros

<https://scitools.org.uk/cartopy/docs/v0.15/crs/projections.html>

Adiciona limites dos países com a Cartopy

Adiciona linhas de costa com a Cartopy

Adiciona linhas de referência com a Cartopy

Plot resultante

... agora vamos ver como sobrepor este  
mapa com nossa imagem ABI

# Adicionando Mapas com a Cartopy

Também precisamos informar à Cartopy a extensão “extent” do nosso dado

Também precisamos informar a extensão (“extent”) do nosso dado. De acordo com a documentação do produto (<https://www.goes-r.gov/products/docs/PUG-L2+-vol5.pdf>), a extensão do dado pode ser calculada **multiplicando metade do full disk pelo tamanho do pixel em radianos e pela altitude do satélite**.

Novamente, toda esta informação está disponível nos metadados. Portanto, se quisermos plotar os dados sem precisar de alterar o código, podemos usar os mesmos scripts para plotar os set

Parágrafo 4.2 - ABI Fixed Grid: <https://www.goes-r.gov/products/docs/PUG-L2+-vol5.pdf>

```
[ ] # calculate the extent of the GOES-R full disk in decimal degrees
xmin = file.variables['x'][:].min() * height
xmax = file.variables['x'][:].max() * height
ymin = file.variables['y'][:].min() * height
ymax = file.variables['y'][:].max() * height

# list with the extent
img_extent = (xmin, xmax, ymin, ymax)
print(img_extent)

(-5433892.69232443, 5433892.69232443, -5433892.69232443, 5433892.69232443)
```

De acordo com a documentação GOES-R, a extensão é:  
Número de pixels (metade do full disk) x tamanho do pixel (radianos) x altitude do satélite

Esta primeira multiplicação já está disponível nos metadados, sendo chamada de datasets ‘x’ e ‘y’

Valores necessários pela Cartopy

```
# plot the image
img = ax.imshow(data, vmin=-80, vmax=40, origin='upper', extent=img_extent, cmap='Greys')
```

# Adicionando Mapas com a Cartopy

Todas as instruções necessárias para criar a imagem à direita

```
# open the GOES-R NetCDF file
file = Dataset("samples/OR_ABI-L2-CMIPF-M6C13_G16_s20232411500206_e20232411509526_c20232411509585.nc")

# get the pixel values
data = file.variables['CMI'][:] - 273.15

# choose the plot size (width x height, in inches)
plt.figure(figsize=(15,15))

# use the Geostationary projection in cartopy
ax = plt.axes(projection=ccrs.Geostationary(central_longitude=longitude, satellite_height=height))

# calculate the extent of the GOES-R full disk in decimals (2712*0.000056*35786023.0)
xmin = file.variables['x'][:].min() * height
xmax = file.variables['x'][:].max() * height
ymin = file.variables['y'][:].min() * height
ymax = file.variables['y'][:].max() * height

# list with the extent
img_extent = (xmin, xmax, ymin, ymax)

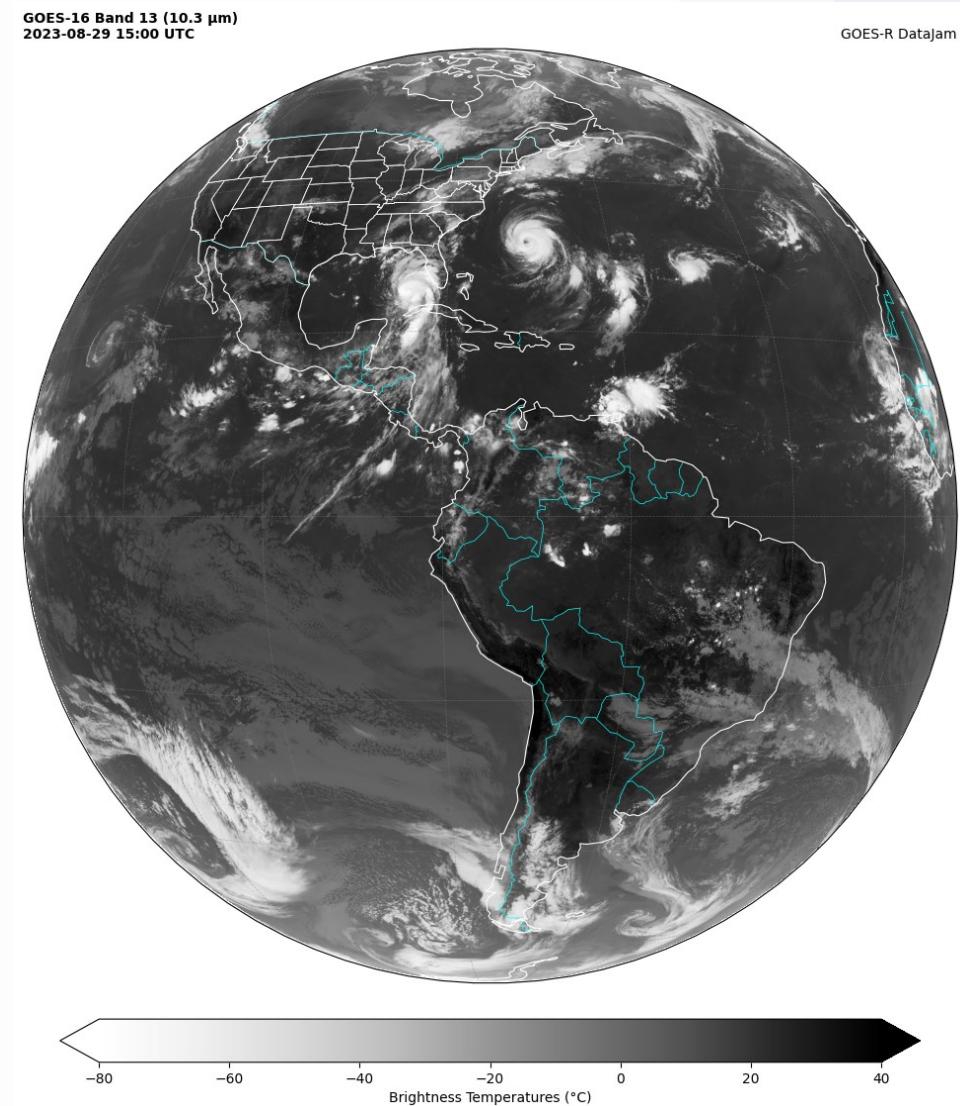
# add coastlines, borders and gridlines
ax.add_feature(cartopy.feature.BORDERS, edgecolor='cyan', linewidth=0.5)
ax.add_feature(cartopy.feature.STATES, edgecolor='white', linewidth=0.5)
ax.coastlines(resolution='110m', color='white', linewidth=0.8)
ax.gridlines(color='gray', alpha=0.5, linestyle='--', linewidth=0.5)

# plot the image
img = ax.imshow(data, vmin=-80, vmax=40, origin='upper', extent=img_extent, cmap='Greys')

# add a colorbar
plt.colorbar(img, label='Brightness Temperatures (°C)', extend='both', orientation='horizontal', pad=0.03, shrink=0.75)

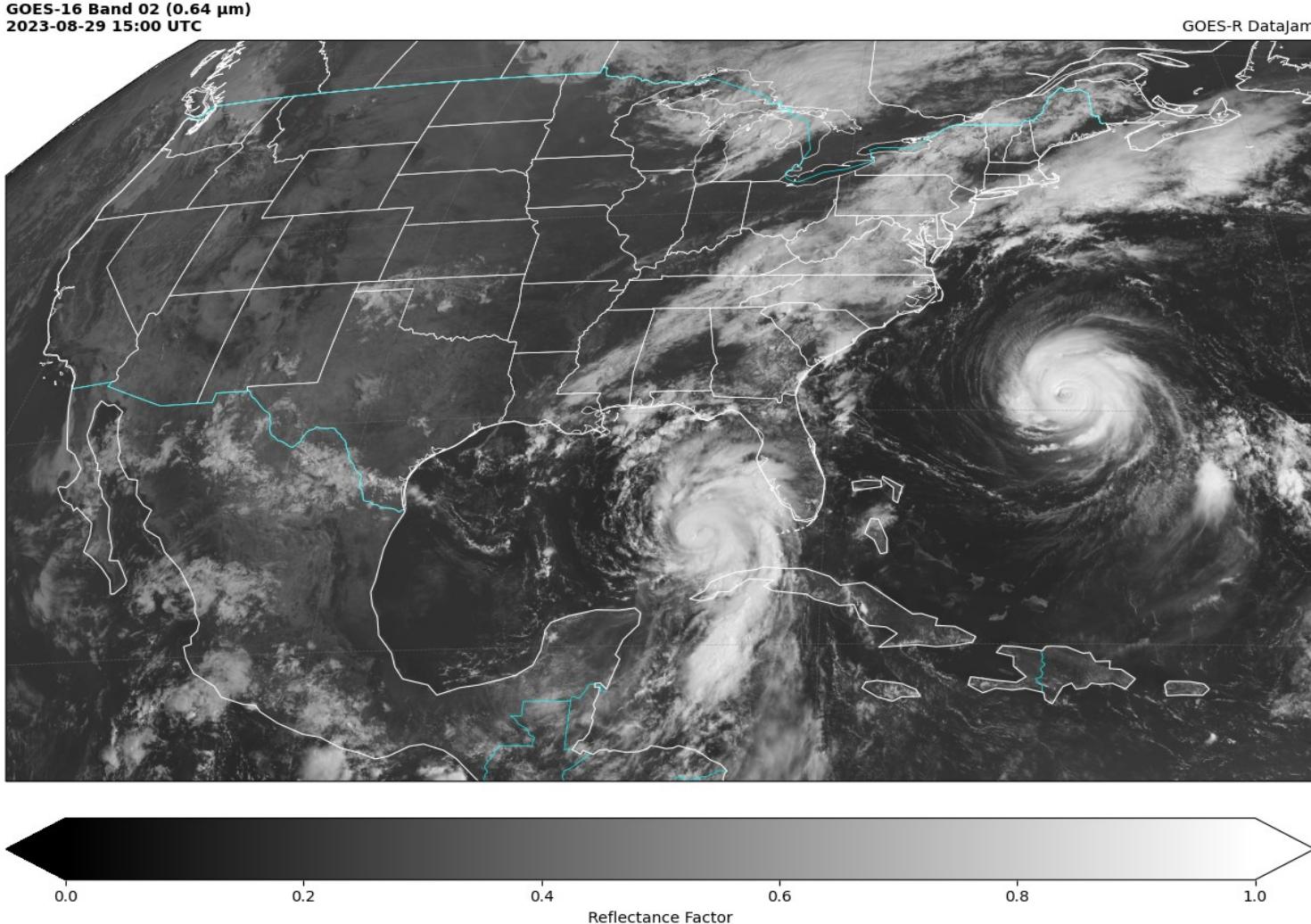
# add a title (one on the left and one on the right)
plt.title(f'GOES-16 Band 13 (10.3 μm)\n{date}', fontweight='bold', fontsize=10, loc='left')
plt.title('GOES-R DataJam', fontsize=10, loc='right')
```

## LEITURA E MANIPULAÇÃO DOS DADOS CONFIGURAÇÃO DO PLOT



# Adicionando Mapas com a Cartopy

Estamos lendo parâmetros dos metadados, portanto Podemos plotar outros setores sem modificar as instruções Cartopy!



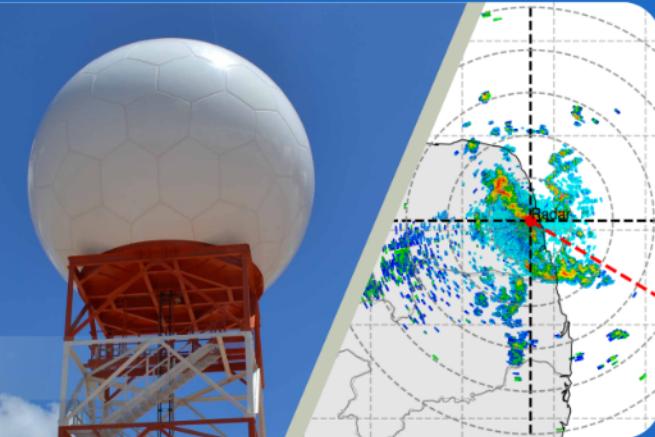
# Radar Meteorológico: Fundamentos e Processamento de Dados com Python

Atividade Pré-Curso:  
Introdução ao Google Colab

## OBRIGADO!

### CURSO: RADAR METEOROLÓGICO FUNDAMENTOS E PROCESSAMENTO DE DADOS COM PYTHON

Abril de 2024



Diego Souza  
[diego.souza@inpe.br](mailto:diego.souza@inpe.br)

DISSM - Divisão de Satélites e Sensores Meteorológicos  
CGCT - Coordenação Geral de Ciências da Terra  
INPE - Instituto Nacional de Pesquisas Espaciais

MINISTÉRIO DA  
CIÊNCIA, TECNOLOGIA  
E INOVAÇÃO

GOVERNO FEDERAL  
**BRASIL**  
UNIÃO E RECONSTRUÇÃO