



Visualizing and Analyzing GOES-R Data Using Google Colab



Diego Souza

diego.souza@inpe.br

Engineer - INPE (Brazil)

National Institute for Space Research

Marcial Garbanzo

marcial.garbanzo@ucr.ac.cr

Professor - UCR

University of Costa Rica



Presentation Outline

- Satellite Data Access and Processing
- Introduction to Google Colab
- GOES-R Data Processing Examples
- Additional Resources



Satellite Data Access and Processing

GRB



HRIT/EMWIN



HRD



GEONETCast



INTERNET / CLOUD SERVICES



CCC
Environmental
Data Commons



Google Cloud Platform



LONG-TERM ARCHIVE



EUMETSAT
Data Centre



WEB PAGES



netcdf4



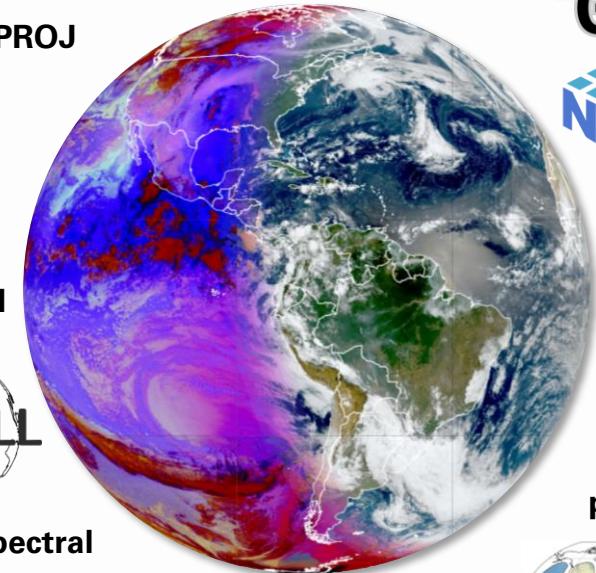
Folium



pyHDF



pyGRIB



pyPROJ



PySpectral

matplotlib

... and much more

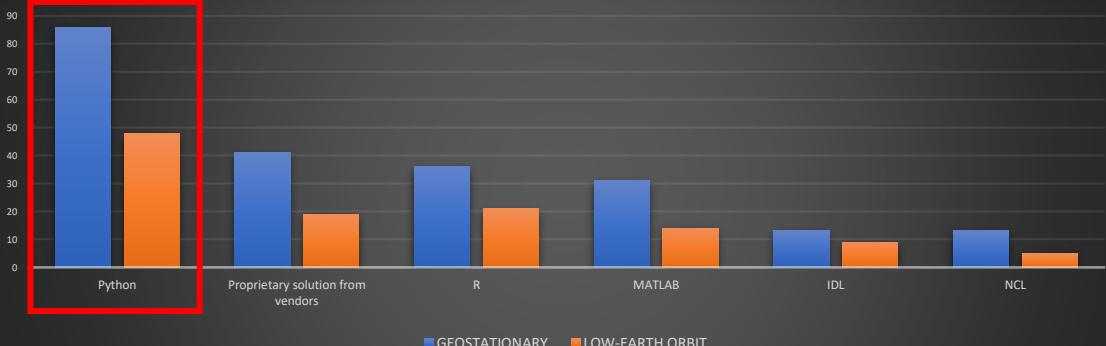
The Python Programming Language



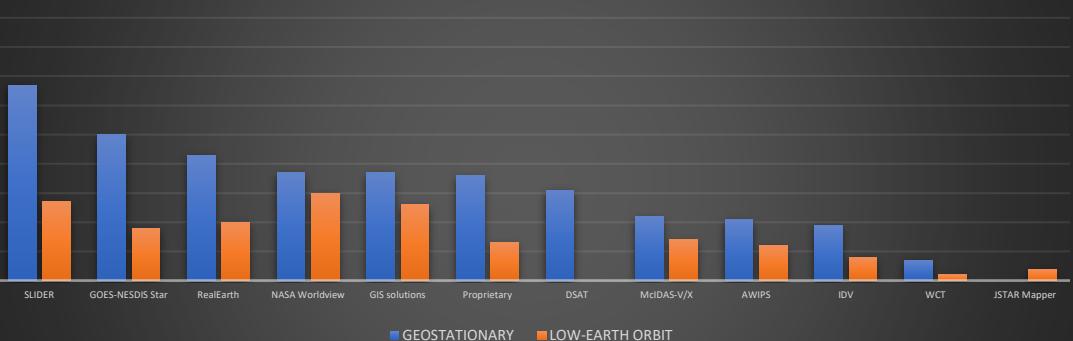
- Freely Available
- Easy to Get Started
- Gradual Learning Curve
- (Very) Active Community
- Many (Many!) Libraries / Modules
- Multiple Input and Output Formats
- Works on Different Operating Systems
- Integration with Other Software / Applications

Use of Python by the Satellite Community

9. Data processing - Please indicate the tools you currently use for GEO and LEO satellites



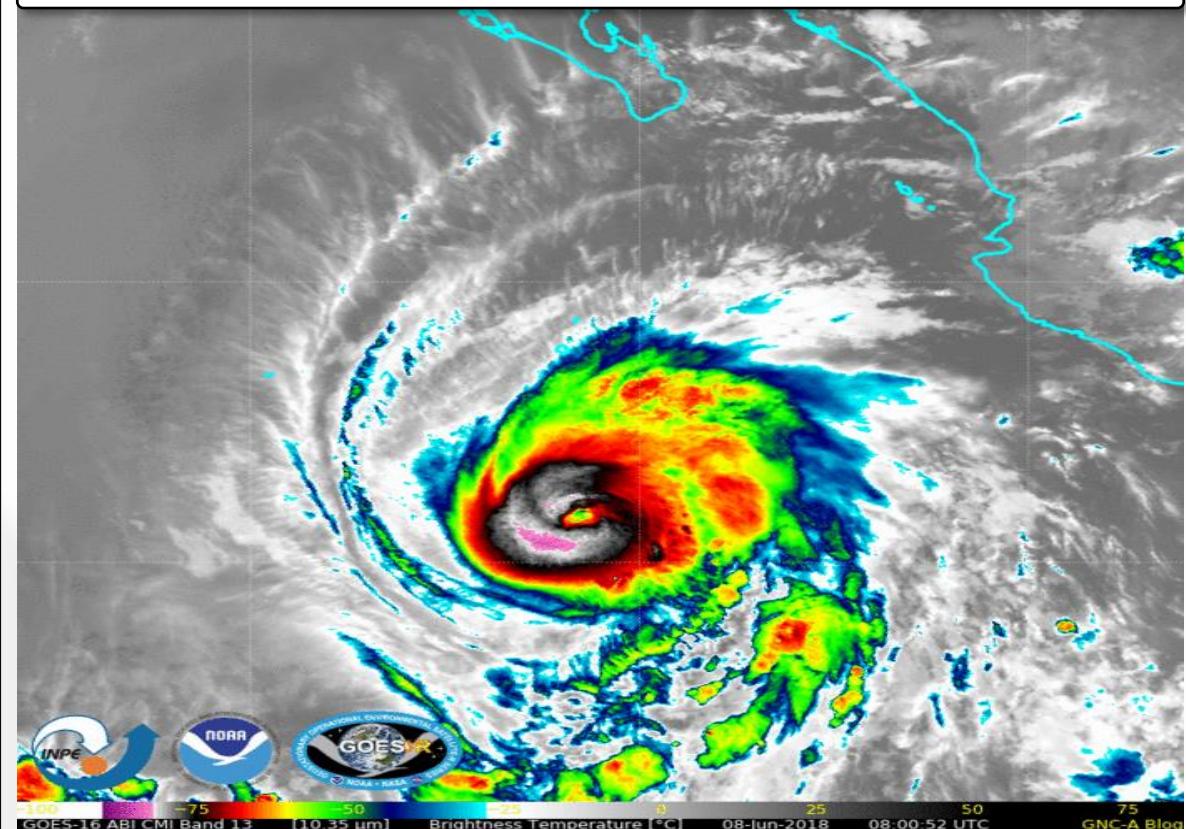
10. Data visualization - Please indicate the tools you currently use for GEO and LEO satellites



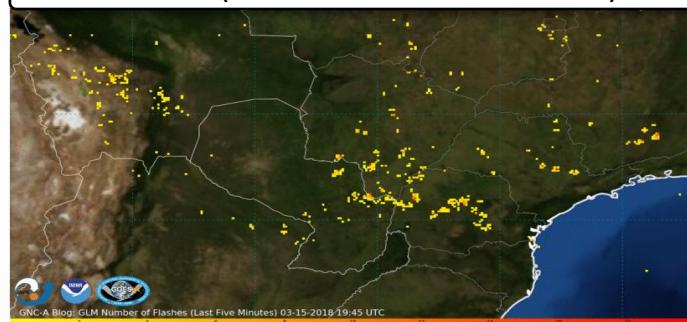
The results of the WMO Regional Survey can be seen at the following link: [Link](#)

What Can You Do With Python? Some Examples

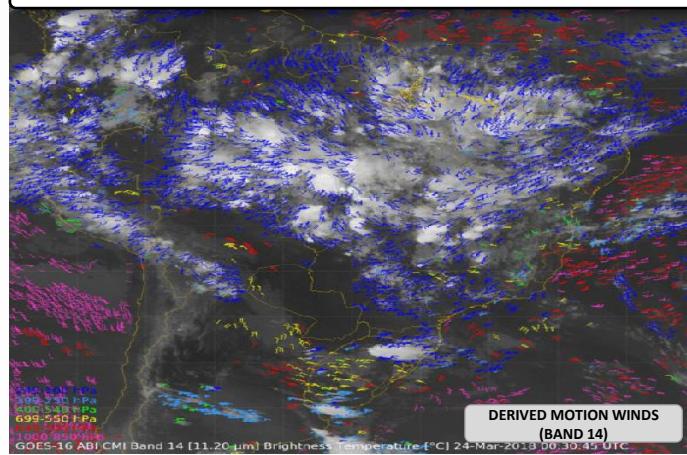
GOES-R ABI BANDS



GOES-R GLM (GEOSTATIONARY LIGHTNING MAPPER)

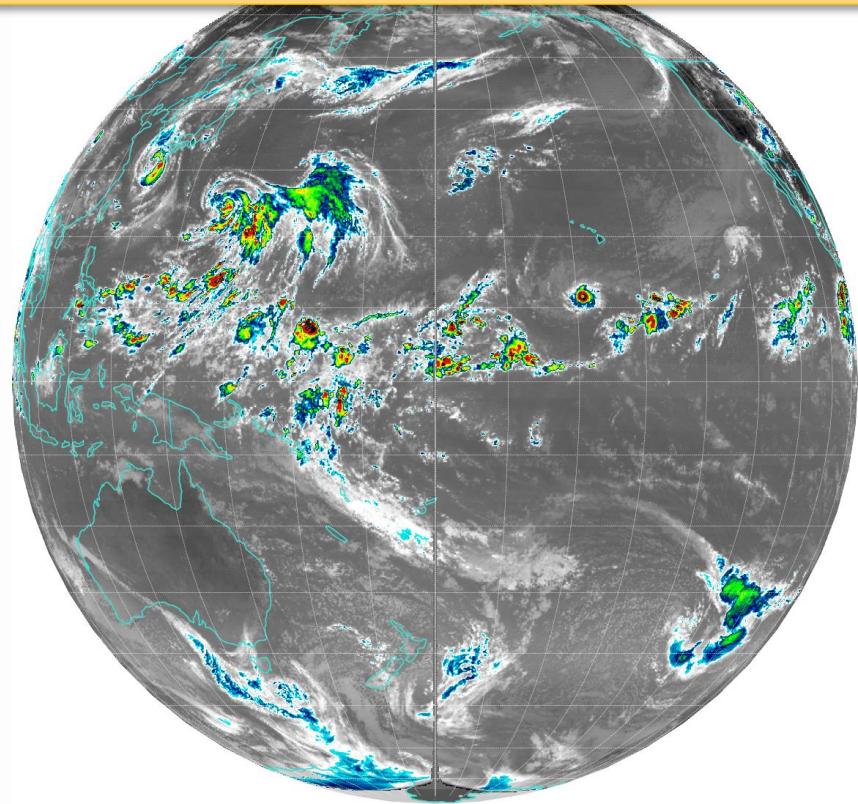
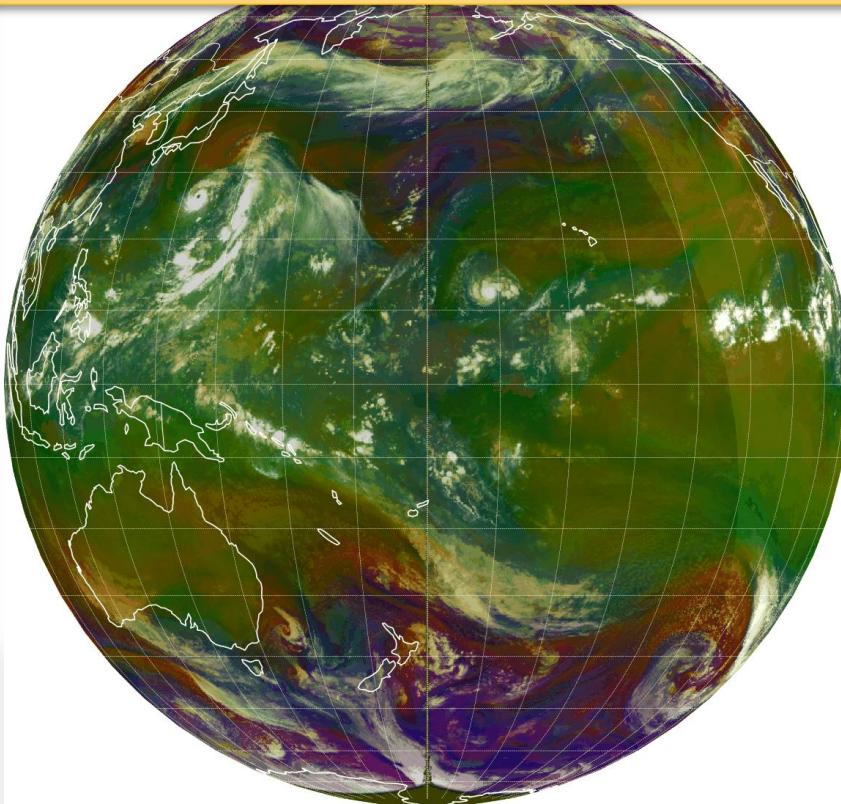


GOES-R DERIVED PRODUCTS



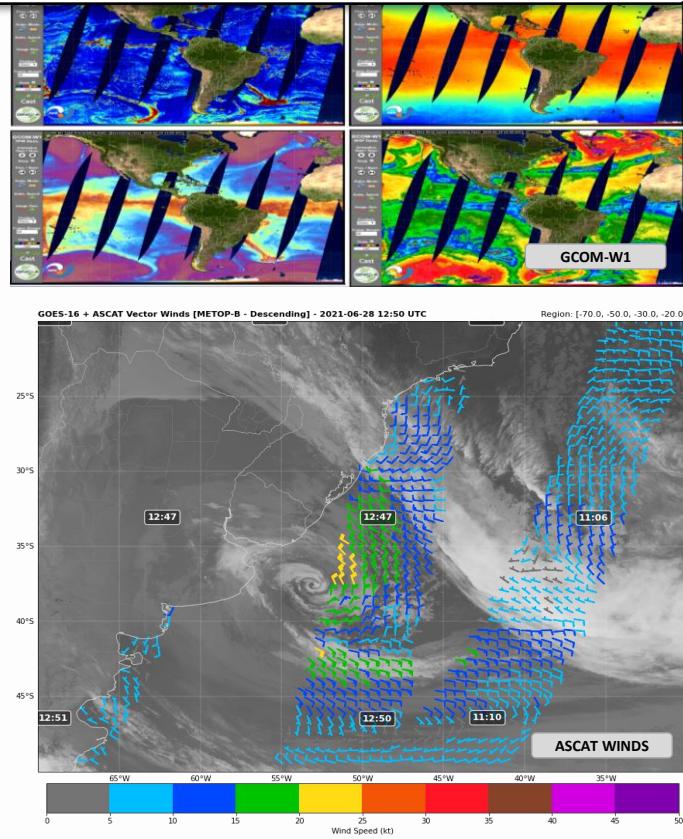
What Can You Do With Python? Some Examples

GOES-East / GOES-West (NOAA) , Meteosat 0° / Meteosat 45.5°E (EUMETSAT) and Himawari-8 (JMA)



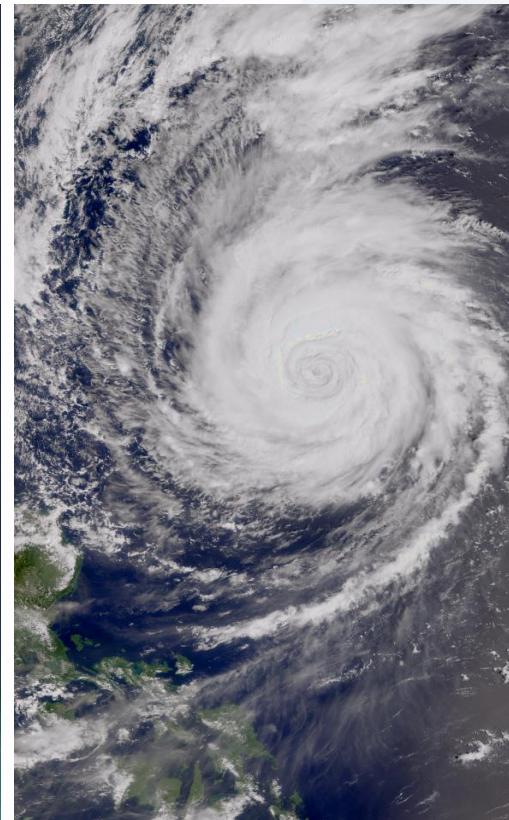
What Can You Do With Python? Some Examples

LOW-EARTH ORBIT (LEO) SATELLITES - IMAGERY AND PRODUCTS

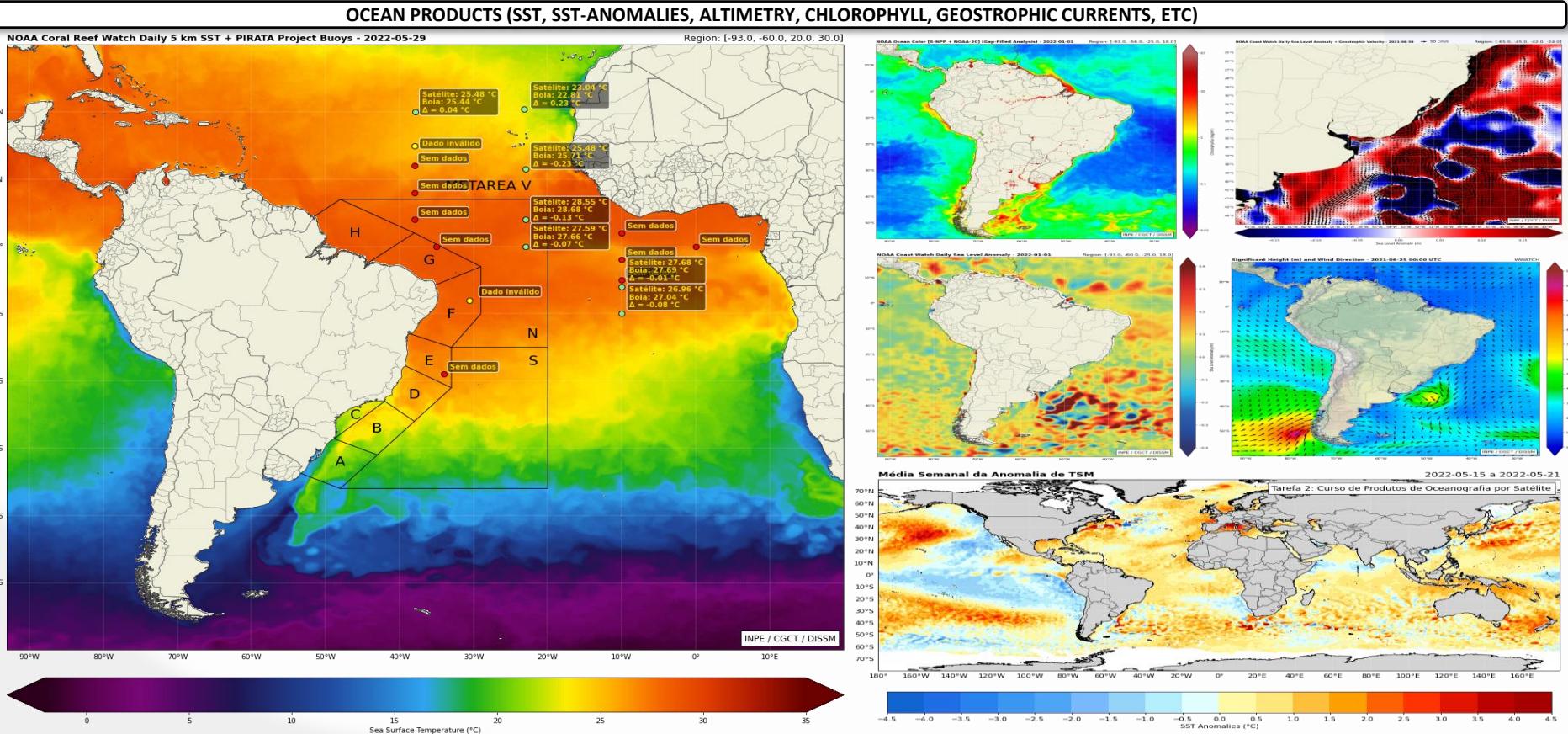


What Can You Do With Python? Some Examples

HIGH RESOLUTION IMAGERY (IN THE EXAMPLES BELOW, SENTINEL-2 AND SENTINEL-3 IMAGERY)

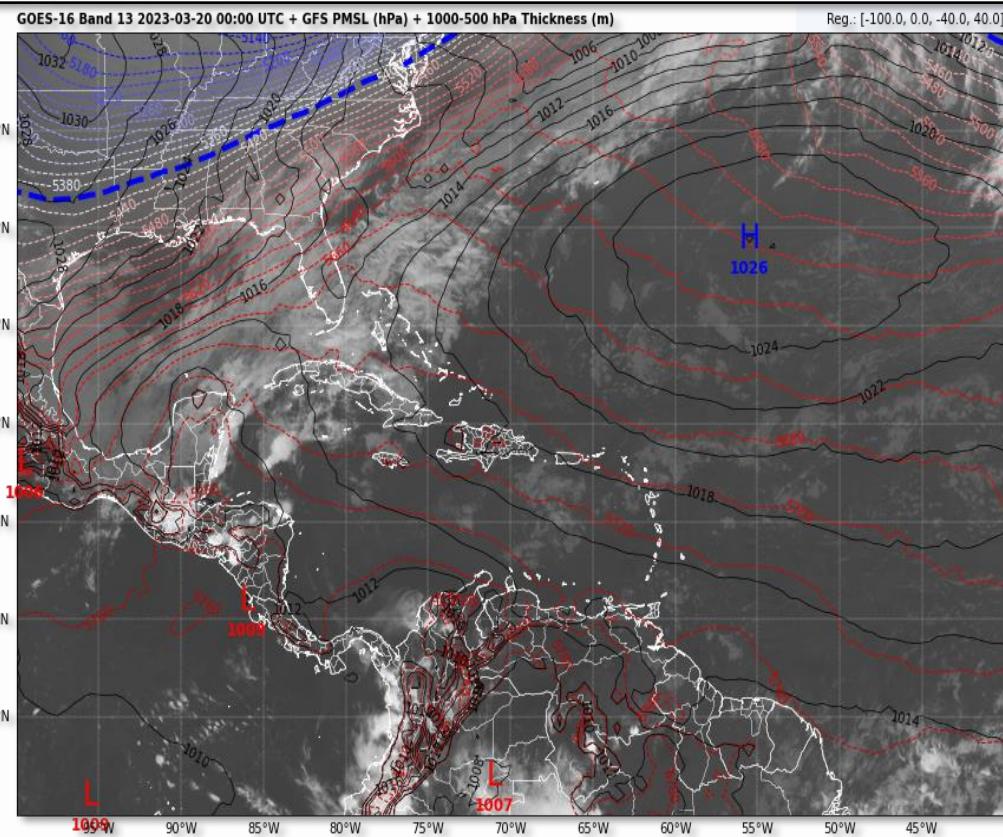
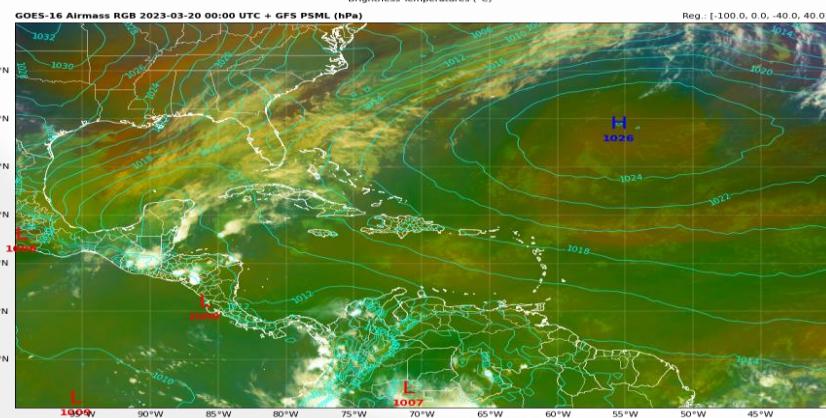
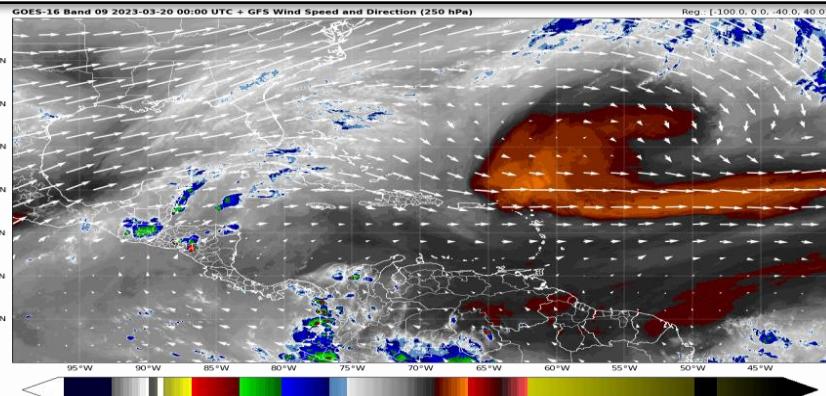


What Can You Do With Python? Some Examples

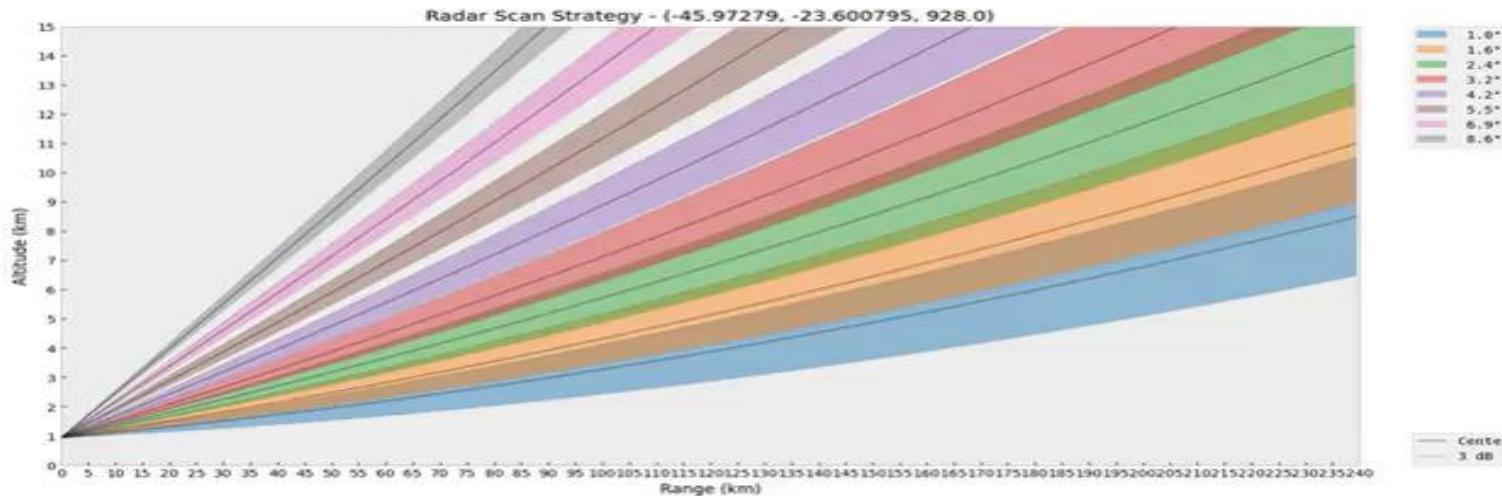


What Can You Do With Python? Some Examples

NWP OUTPUT + SATELLITE IMAGERY



WEATHER RADAR - THIS CONETENT IS BEING PREPARED FOR AN UPCOMING TRAINING ORGANIZED IN BRAZIL

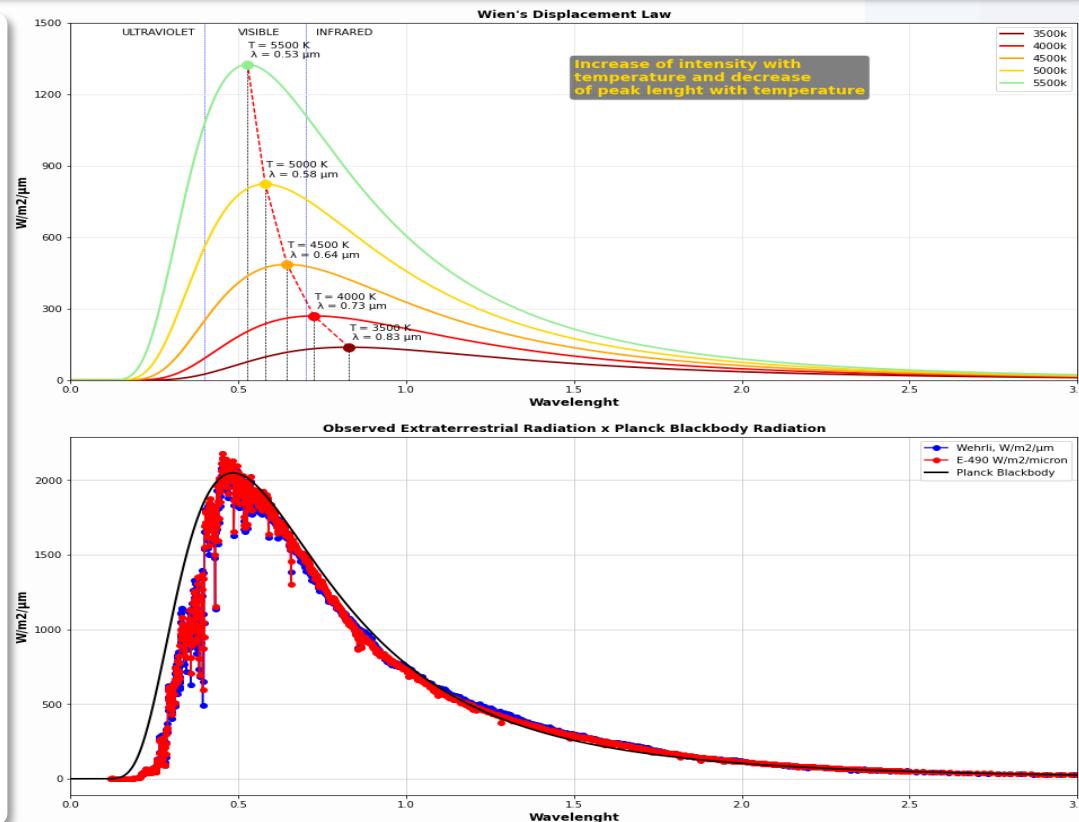
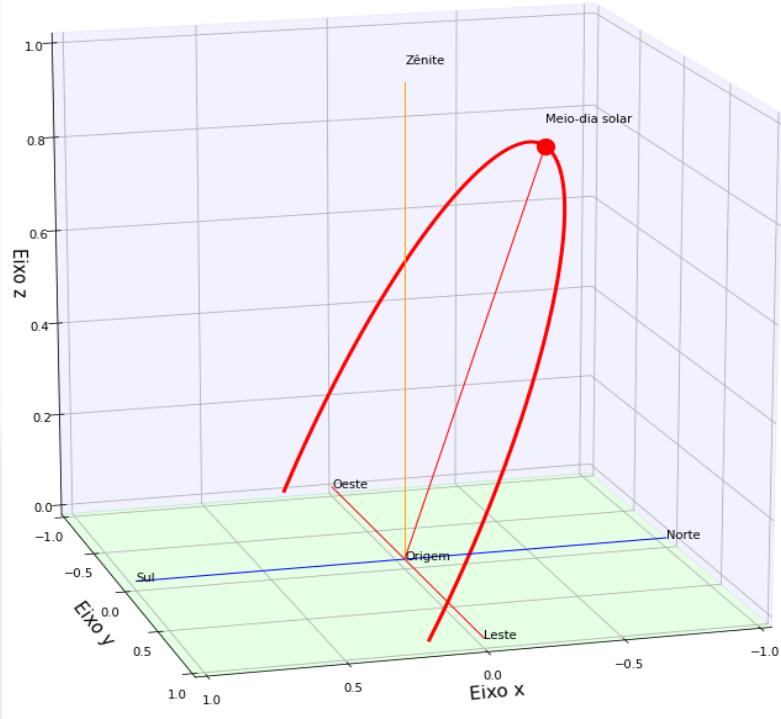


What Can You Do With Python? Some Examples

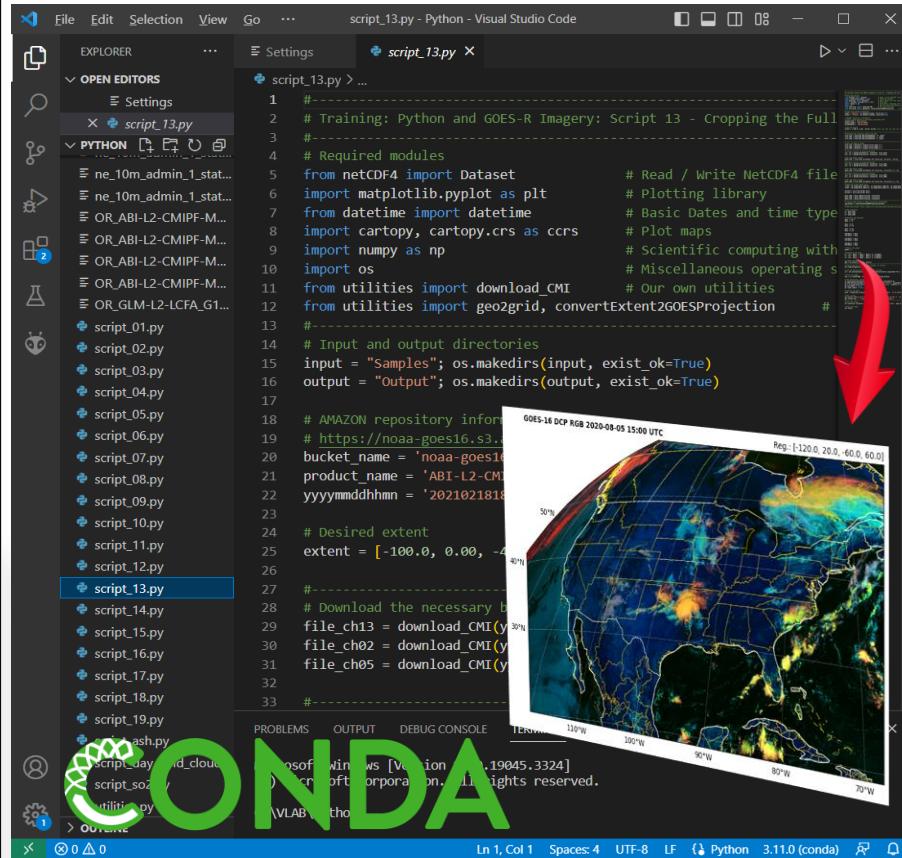
GENERAL GRAPHS - IN THIS EXAMPLE, SOME PLOTS CREATED FOR AN UPCOMING TRAINING ON RADIATION PRODUCTS

Trajetória Solar $\Phi = -40.16^\circ$ e $dn = 289$ (2022-10-16)

Nascer do Sol = 5:30:42 h
Pôr do Sol = 18:29:17 h



Running Python Scripts Locally or in the Cloud



File Edit Selection View Go ...

EXPLORER OPEN EDITORS Settings

script_13.py > ...

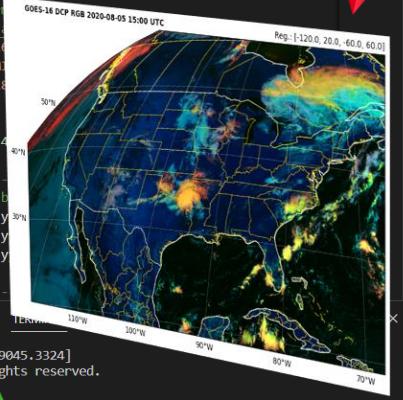
```

1  # Training: Python and GOES-R Imagery: Script 13 - Cropping the Full
2  # ---#
3  # Required modules
4
5  from netCDF4 import Dataset           # Read / Write NetCDF4 file
6  import matplotlib.pyplot as plt       # Plotting library
7  from datetime import datetime         # Basic Dates and time type
8  import cartopy, cartopy.crs as ccrs   # Plot maps
9  import numpy as np                   # Scientific computing with
10 import os                           # Miscellaneous operating system
11 from utilities import download_CMI  # Our own utilities
12 from utilities import geo2grid, convertExtent2GOESProjection  #
13
14 # Input and output directories
15 input = "Samples"; os.makedirs(input, exist_ok=True)
16 output = "Output"; os.makedirs(output, exist_ok=True)
17
18 # AMAZON repository information
19 # https://noaa-goes16.s3.amazonaws.com/
20 bucket_name = 'noaa-goes16'
21 product_name = 'ABI-L2-CMIPF-M..._G16'
22 yyyymmddhhmm = '20210218181500'
23
24 # Desired extent
25 extent = [-100.0, 0.0, -45.0, 55.0]
26
27 # Download the necessary bands
28 file_ch13 = download_CMI(y=35, p=13, extent=extent)
29 file_ch02 = download_CMI(y=2, p=2, extent=extent)
30 file_ch05 = download_CMI(y=5, p=5, extent=extent)
31
32 # ---#
33

```

GOES-16 DCP RGB 2020-08-05 15:00 UTC

Reg.: [-120.0, 20.0, -60.0, 60.0]

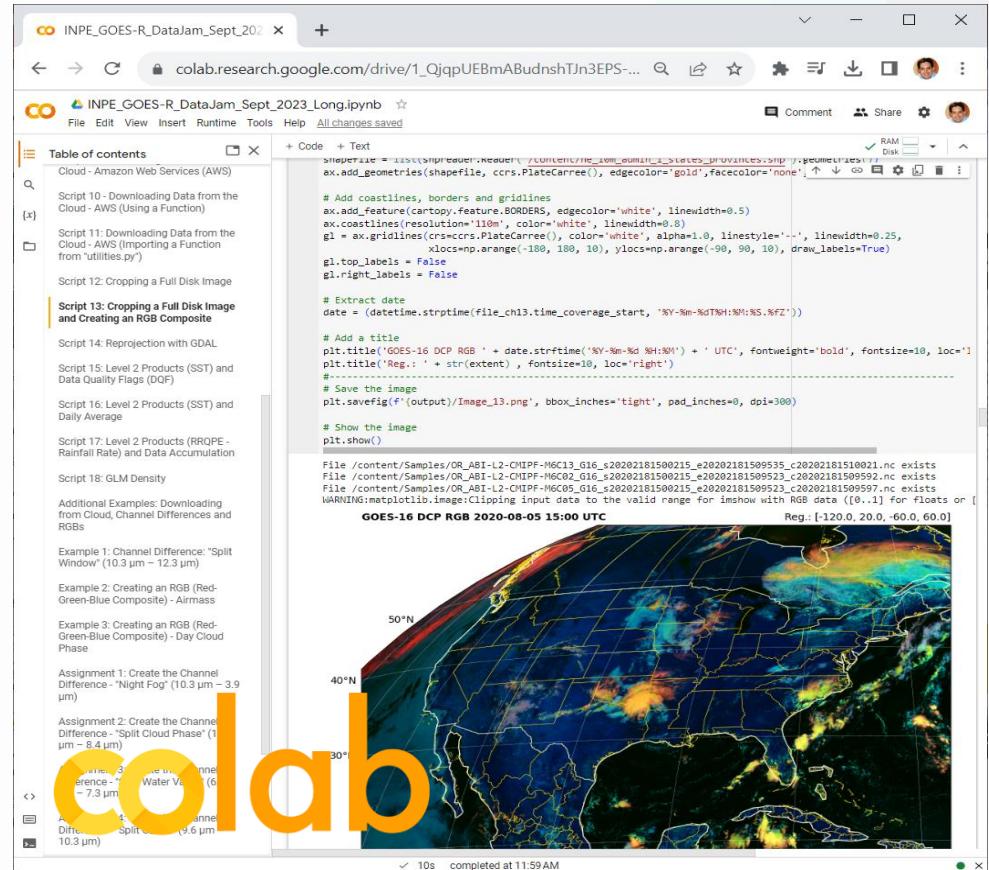


PROBLEMS OUTPUT DEBUG CONSOLE

Windows [Version 10.0.19045.3324] Microsoft Corporation. All rights reserved.

CONDA

Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.11.0 (conda)



File Edit View Insert Runtime Tools Help All changes saved

Table of contents

- 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 Creating an RGB Composite**
 - Script 14: Reprojection with GDAL
 - Script 15: Level 2 Products (SST) and Data Quality Flags (DQF)
 - Script 16: Level 2 Products (SST) and Daily Average
 - Script 17: Level 2 Products (RROPE - Rainfall Rate) and Data Accumulation
 - Script 18: GLM Density
 - Additional Examples: Downloading from Cloud, Channel Differences and RGBs
 - Example 1: Channel Difference: "Split Window" (10.3 µm - 12.3 µm)
 - Example 2: Creating an RGB (Red-Green-Blue Composite) - Alarms
 - Example 3: Creating an RGB (Red-Green-Blue Composite) - Day Cloud Phase
 - Assignment 1: Create the Channel Difference - "Night Fog" (10.3 µm - 3.9 µm)
 - Assignment 2: Create the Channel Difference - "Split Cloud Phase" (1 µm - 8.4 µm)
 - Assignment 3: Create the Channel Difference - "Water Vapour" (6.7 µm - 7.3 µm)
 - Assignment 4: Create the Channel Difference - "Split Cloud Phase" (10.3 µm - 9.6 µm)

INPE_GOES-R_DataJam_Sept_2023.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Table of contents

- 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 Creating an RGB Composite**
 - Script 14: Reprojection with GDAL
 - Script 15: Level 2 Products (SST) and Data Quality Flags (DQF)
 - Script 16: Level 2 Products (SST) and Daily Average
 - Script 17: Level 2 Products (RROPE - Rainfall Rate) and Data Accumulation
 - Script 18: GLM Density
 - Additional Examples: Downloading from Cloud, Channel Differences and RGBs
 - Example 1: Channel Difference: "Split Window" (10.3 µm - 12.3 µm)
 - Example 2: Creating an RGB (Red-Green-Blue Composite) - Alarms
 - Example 3: Creating an RGB (Red-Green-Blue Composite) - Day Cloud Phase
 - Assignment 1: Create the Channel Difference - "Night Fog" (10.3 µm - 3.9 µm)
 - Assignment 2: Create the Channel Difference - "Split Cloud Phase" (1 µm - 8.4 µm)
 - Assignment 3: Create the Channel Difference - "Water Vapour" (6.7 µm - 7.3 µm)
 - Assignment 4: Create the Channel Difference - "Split Cloud Phase" (10.3 µm - 9.6 µm)

GOES-16 DCP RGB 2020-08-05 15:00 UTC

Reg.: [-120.0, 20.0, -60.0, 60.0]

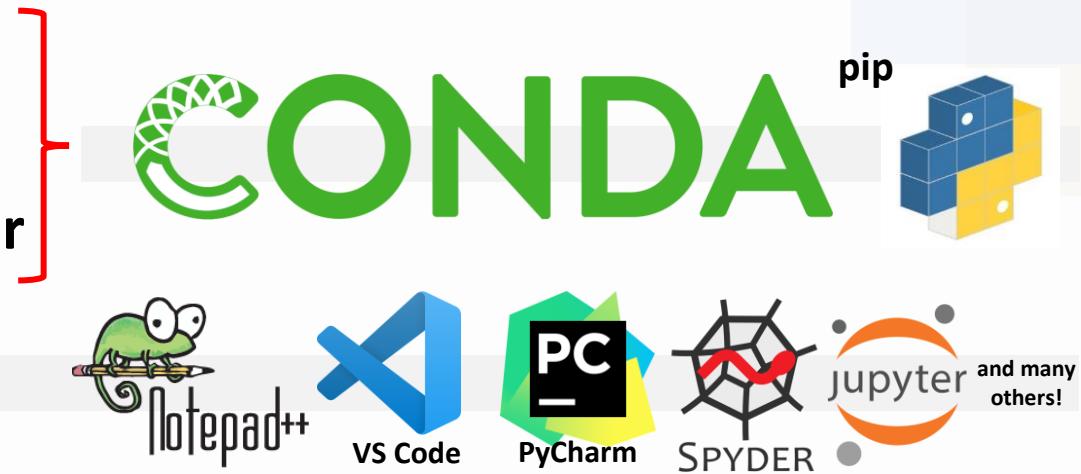
colab

10s completed at 11:59 AM

Running Python Scripts Locally

This is what you need in your computer or server: Conda (or pip), and editor to write your scripts (or IDE) and some sample imagery

- Python
- Package Manager
- Virtual Environment Manager



- Text Editor / IDE (Integrated Development Environment)

- Sample Data

We will not see it in this video but there's a link in the end showing the steps needed



A variety of data access mechanisms available

Running Python Scripts in the Cloud

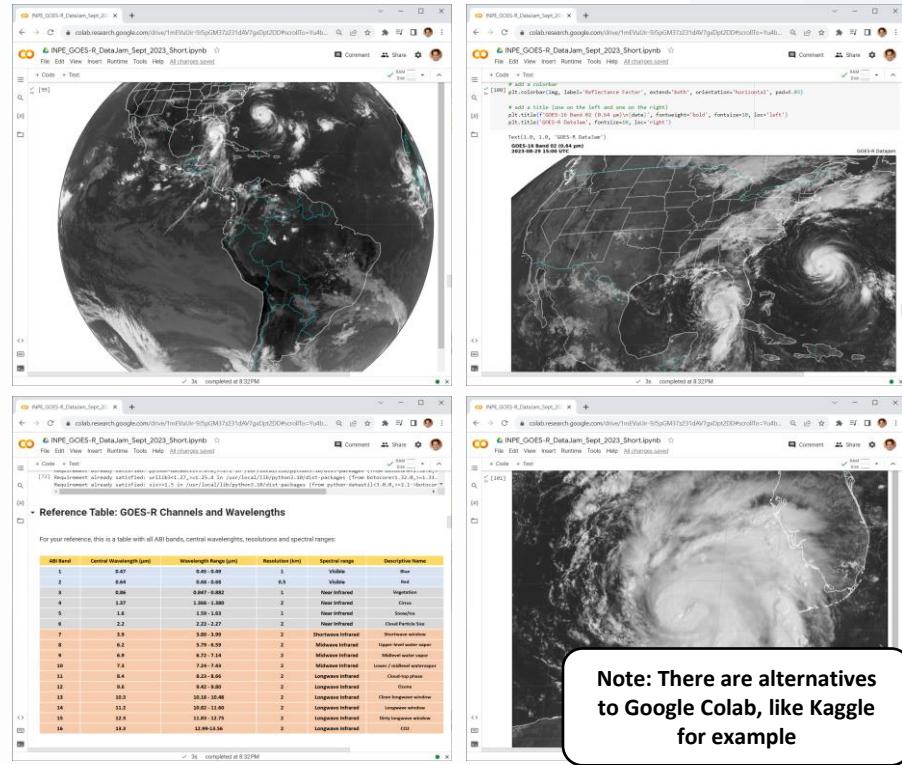
You only need a Colab “Notebook”! We have prepared some examples.

<https://colab.research.google.com/drive/1mEVuUir-9J5pGM37z231dAV7gxDpt2DD?usp=sharing>



Developed by: Diego Souza - INPE / CGCT / DISSM - Brazil
COLAB Notebook Version: September 07, 2023

Contact:
Diego Souza - INPE / CGCT / DISSM - (diego.souza@inpe.br)

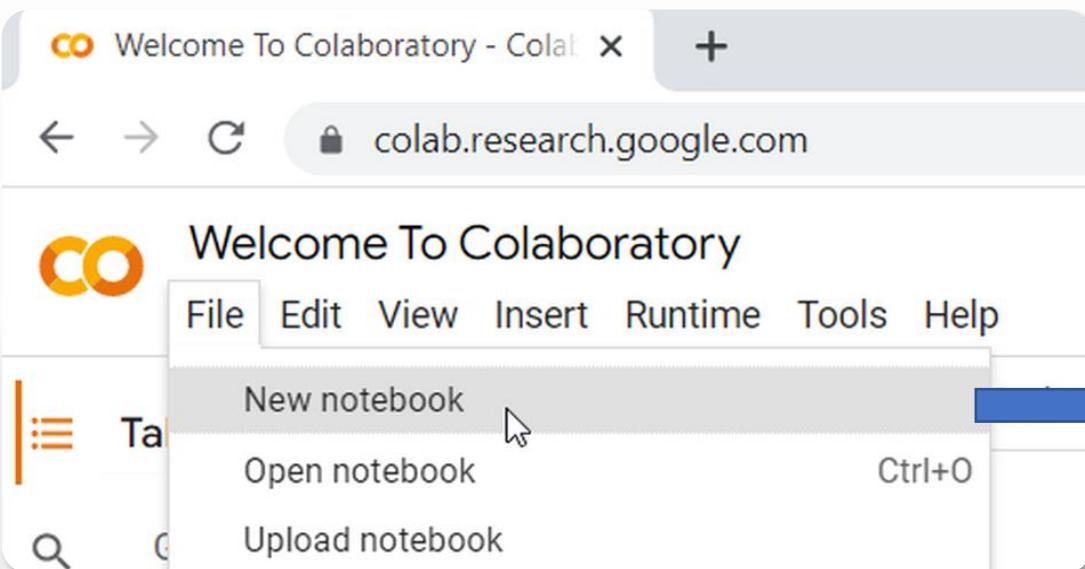


Alt Band	Central Wavelength (μm)	Wavelength Range (μm)	Resolution (km)	Spectral range	Descriptive Name
1	0.67	0.64 - 0.89	8.5	Visible	Red
2	0.96	0.87 - 1.02	1	Near Infrared	Vegetation
4	1.37	1.26 - 1.49	2	Near Infrared	Cancer
5	1.61	1.50 - 1.72	2	Near Infrared	Bands
6	2.21	2.01 - 2.7	2	Near Infrared	Cloud Particle Size
7	3.9	3.80 - 3.99	2	Shortwave Infrared	Water vapor
8	5.79 - 5.9		2	Midwave Infrared	Upper level cloud cover
9	6.7	6.50 - 6.95	2	Midwave Infrared	Midlevel cloud cover
10	7.3	7.20 - 7.43	2	Midwave Infrared	Cloud-top temperature
11	8.4	8.23 - 8.65	2	Longwave Infrared	Cloud-top phase
12	9.6	9.40 - 9.84	2	Longwave Infrared	Ozone
13	10.5	10.30 - 10.84	2	Longwave Infrared	Cloud temperature
14	11.2	10.80 - 11.90	2	Longwave Infrared	Temperature
15	13.3	11.80 - 12.79	2	Longwave Infrared	Very longwave window
16	13.8	12.99 - 13.56	2	Longwave Infrared	CO ₂

Note: There are alternatives to Google Colab, like Kaggle for example

GOOGLE COLAB: BASIC CONCEPTS

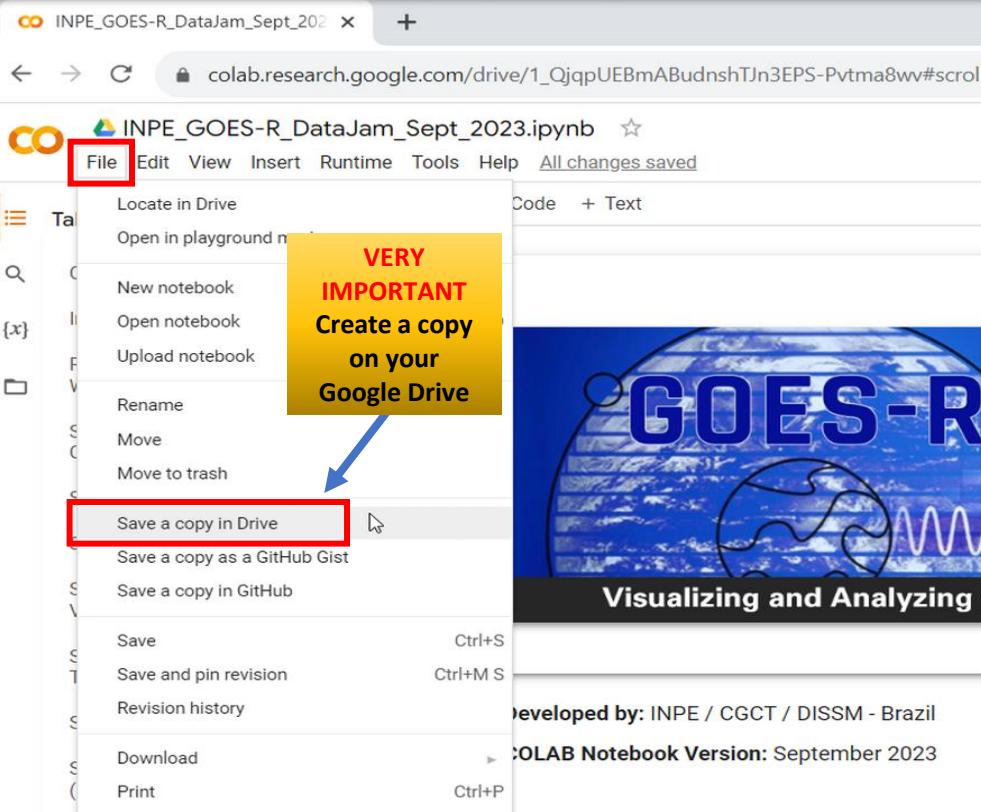
Getting started with Google COLAB: <https://colab.research.google.com/>



A screenshot of the Google Colab interface. On the left, a browser window shows the URL `colab.research.google.com`. On the right, the Colab interface is open with a notebook titled "Untitled1.ipynb". A blue arrow points from the "New notebook" option in the "File" menu towards the right side of the screen, where a callout box says "This will create a blank Colab notebook". Another callout box at the bottom right says "... let's make a quick demonstration".

Creating a Copy and Running Code Cells

Create a copy on your Google Drive



INPE_GOES-R_DataJam_Sept_2023.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Locate in Drive Open in playground New notebook Open notebook Upload notebook Rename Move Move to trash Save a copy in Drive Save a copy as a GitHub Gist Save a copy in GitHub Save Ctrl+S Save and pin revision Ctrl+M S Revision history Download Ctrl+P Print Ctrl+P

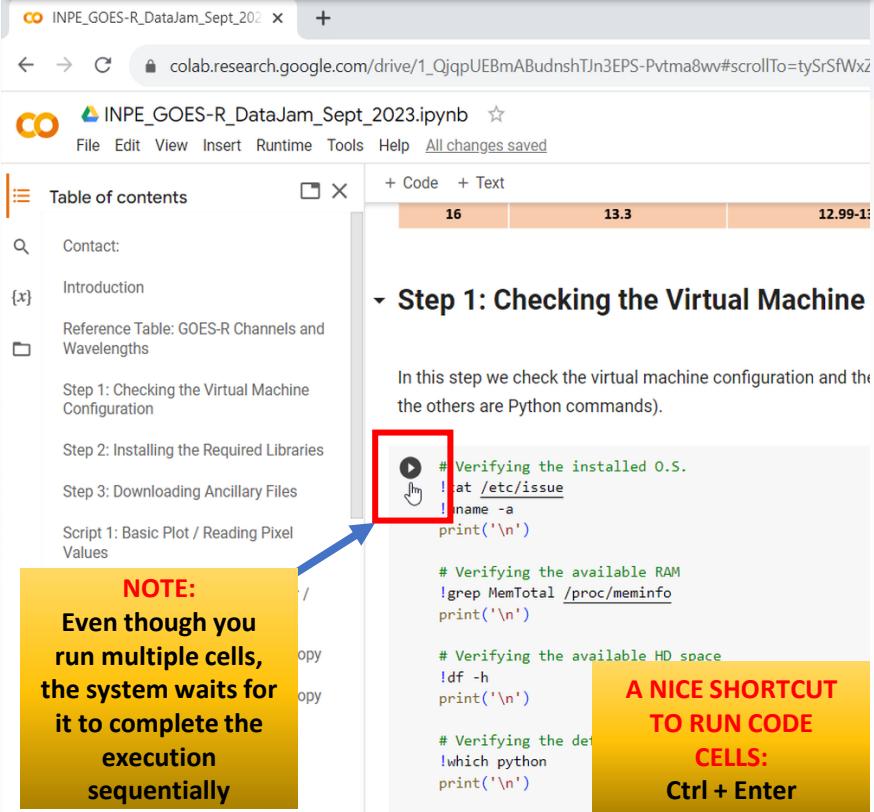
Code + Text

Very Important Create a copy on your Google Drive

INPE GOES-R DataJam Sept 2023.ipynb

Developed by: INPE / CGCT / DISSM - Brazil COLAB Notebook Version: September 2023

Run the instructions in each cell, sequentially



INPE_GOES-R_DataJam_Sept_2023.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Table of contents

Contact: Introduction Reference Table: GOES-R Channels and Wavelengths 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

NOTE: Even though you run multiple cells, the system waits for it to complete the execution sequentially

Verifying the installed O.S.
!cat /etc/issue
!uname -a
print('\n')

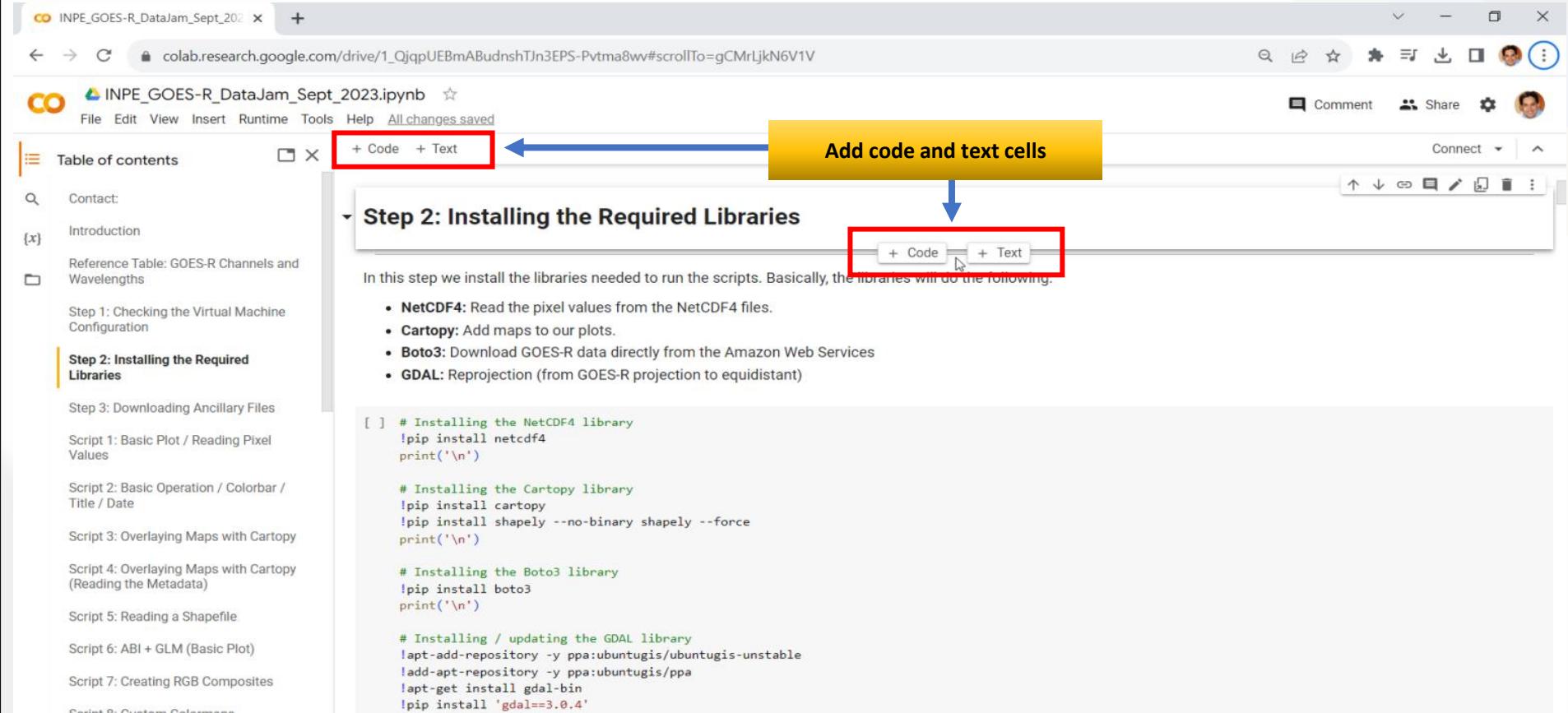
Verifying the available RAM
!grep MemTotal /proc/meminfo
print('\n')

Verifying the available HD space
!df -h
print('\n')

Verifying the default Python
!which python
print('\n')

A NICE SHORTCUT TO RUN CODE CELLS: Ctrl + Enter

Adding Code and Text Cells



The screenshot shows a Google Colab notebook titled "INPE_GOES-R_DataJam_Sept_2023.ipynb". The left sidebar contains a table of contents with several items, including "Step 2: Installing the Required Libraries", which is currently selected. The main area displays a section titled "Step 2: Installing the Required Libraries" with the following text: "In this step we install the libraries needed to run the scripts. Basically, the libraries will do the following." Below this text is a bulleted list of four libraries and their functions:

- **NetCDF4:** Read the pixel values from the NetCDF4 files.
- **Cartopy:** Add maps to our plots.
- **Boto3:** Download GOES-R data directly from the Amazon Web Services
- **GDAL:** Reprojection (from GOES-R projection to equidistant)

At the top of the main area, there are two buttons: "+ Code" and "+ Text". A yellow callout box with the text "Add code and text cells" has arrows pointing to both of these buttons. Another red box highlights the "+ Code" button, and a blue arrow points from the "Add code and text cells" callout to this button. The code block below shows the commands used to install these libraries:

```
[ ] # 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')

# Installing the Boto3 library
!pip install boto3
print('\n')

# Installing / updating the GDAL library
!apt-add-repository -y ppa:ubuntugis/ubuntugis-unstable
!add-apt-repository -y ppa:ubuntugis/ppa
!apt-get install gdal-bin
!pip install 'gdal==3.0.4'
```

INPE_GOES-R_DataJam_Sept_202 x + colab.research.google.com/drive/1_QjqpUEBmA BudnshTJn3EPS-Pvtma8wv#scrollTo=pMkaEVMM4sn9

File Edit View Insert Runtime Tools Help All changes saved

Comment Share RAM Disk

Table of contents

Contact: Introduction Reference Table: GOES-R Channels and Wavelengths

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)

Table of Contents

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)

Step 1: Checking the Virtual Machine Configuration

In this step we check the virtual machine configuration and the Python version installed (the commands starting with "!" are shell commands, all the others are Python commands).

```
[ ] # Verifying the installed O.S.  
!cat /etc/issue  
!uname -a  
print('\n')  
  
# Verifying the available RAM  
!grep MemTotal /proc/meminfo  
print('\n')  
  
# Verifying the available HD space  
!df -h  
print('\n')  
  
# Verifying the default Python installation directory  
!which python  
print('\n')  
  
# Verifying the Python version  
!python --version  
print('\n')
```

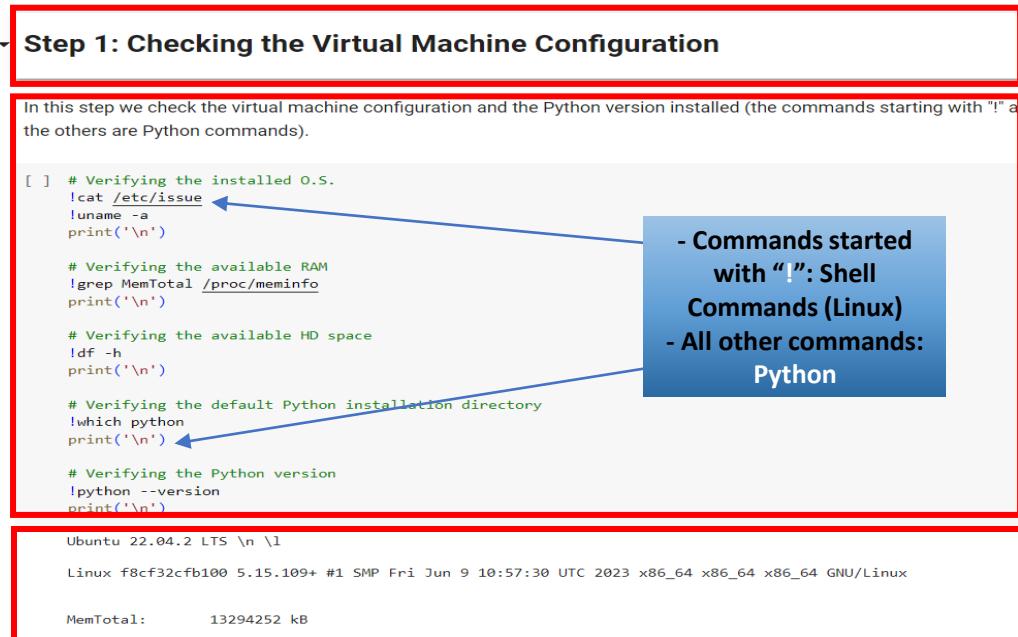
Ubuntu 22.04.2 LTS \n \1
Linux f8cf32cfb100 5.15.109+ #1 SMP Fri Jun 9 10:57:30 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
MemTotal: 13294252 kB

Text Cells

Memory and Disk

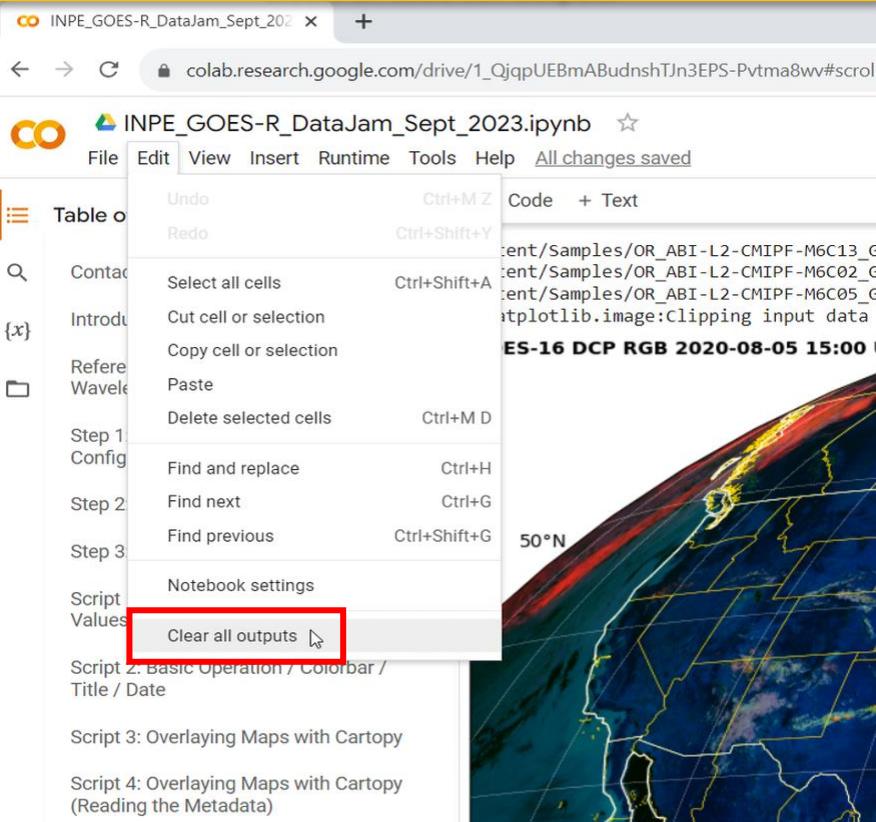
Code Cells

Code Cell Execution Output



Clearing Outputs and Restarting the Runtime

Clears all code cell outputs



INPE_GOES-R_DataJam_Sept_2023.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Table of contents

Undo Ctrl+M Z
Redo Ctrl+Shift+Y

Select all cells Ctrl+Shift+A
Cut cell or selection
Copy cell or selection
Paste
Delete selected cells Ctrl+M D

Find and replace Ctrl+H
Find next Ctrl+G
Find previous Ctrl+Shift+G

Notebook settings

Script Values

Clear all outputs

Script 2: Basic Operation / Colorbar / Title / Date

Script 3: Overlaying Maps with Cartopy

Script 4: Overlaying Maps with Cartopy (Reading the Metadata)

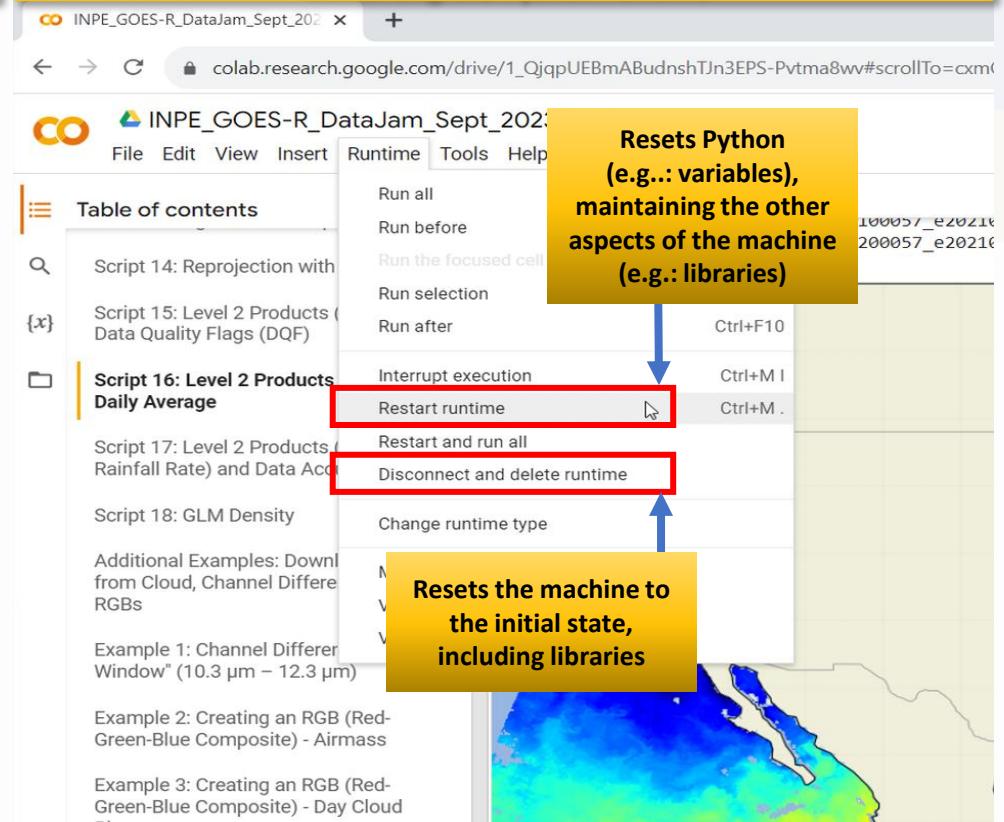
Code + Text

sent/Samples/OR_ABI-L2-CMIPF-M6C13_C
sent/Samples/OR_ABI-L2-CMIPF-M6C02_C
sent/Samples/OR_ABI-L2-CMIPF-M6C05_C
matplotlib.image:Clipping input data

ES-16 DCP RGB 2020-08-05 15:00

50°N

Restarting the Python runtime



INPE_GOES-R_DataJam_Sept_2023.ipynb

File Edit View Insert Runtime Tools Help

Table of contents

Script 14: Reprojection with

{x} Script 15: Level 2 Products (Data Quality Flags (DQF))

Script 16: Level 2 Products Daily Average

Script 17: Level 2 Products (Rainfall Rate) and Data Acc

Script 18: GLM Density

Additional Examples: Downl from Cloud, Channel Differe RGBs

Example 1: Channel Differen Window" (10.3 µm – 12.3 µm)

Example 2: Creating an RGB (Red-Green-Blue Composite) - Airmass

Example 3: Creating an RGB (Red-Green-Blue Composite) - Day Cloud

Run all
Run before
Run the focused cell
Run selection
Run after
Interrupt execution
Restart runtime
Restart and run all
Disconnect and delete runtime

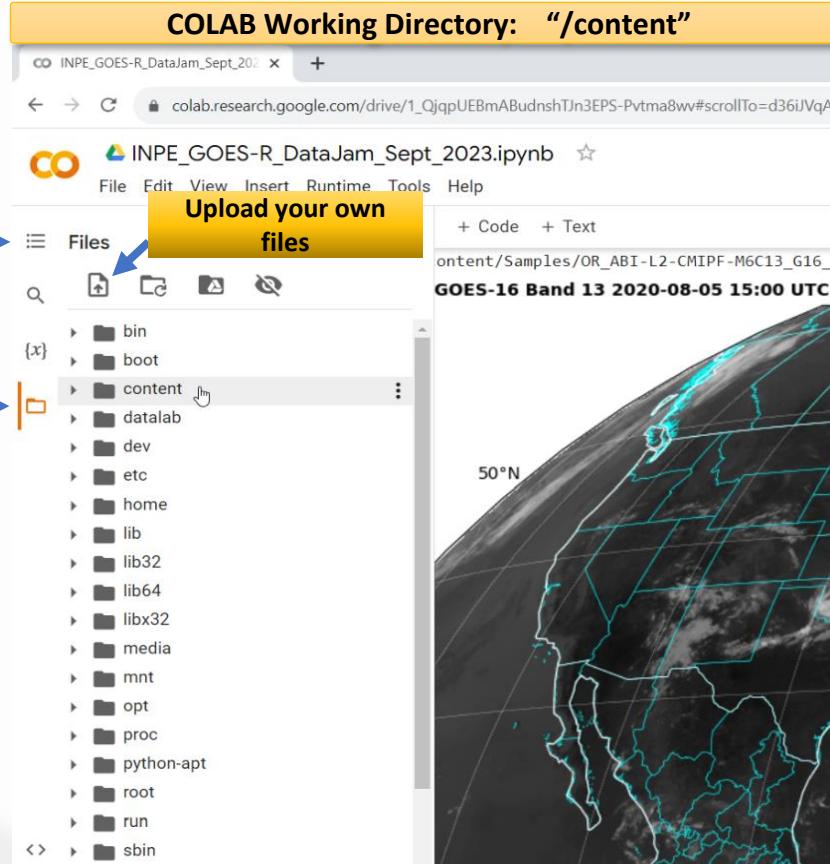
Ctrl+F10
Ctrl+M I
Ctrl+M .

Resets Python (e.g.: variables), maintaining the other aspects of the machine (e.g.: libraries)

Resets the machine to the initial state, including libraries

Viewing Table of Contents and Directories

COLAB Working Directory: "/content"



View Table of Contents → 

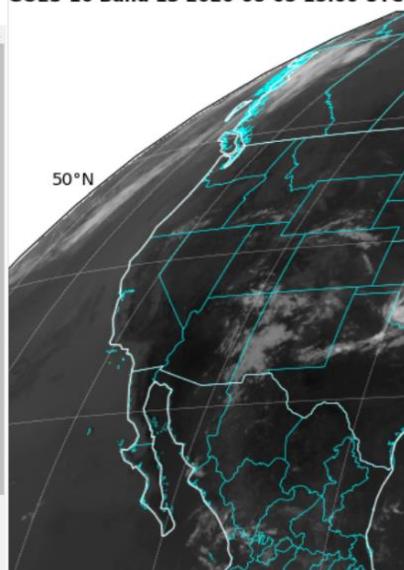
View Directories → 

Upload your own files

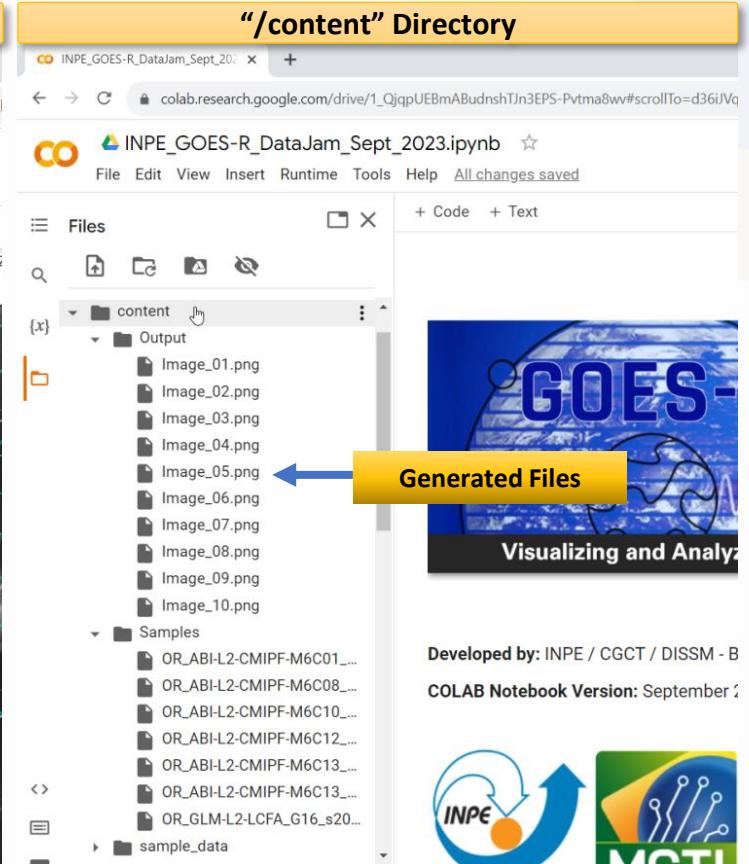
Files

- content
- dataLab
- dev
- etc
- home
- lib
- lib32
- lib64
- libx32
- media
- mnt
- opt
- proc
- python-apt
- root
- run
- sbin

content/Samples/OR_ABI-L2-CMIPF-M6C13_G16_s2
GOES-16 Band 13 2020-08-05 15:00 UTC



"/content" Directory



Generated Files → 

content

- Output
 - Image_01.png
 - Image_02.png
 - Image_03.png
 - Image_04.png
 - Image_05.png
 - Image_06.png
 - Image_07.png
 - Image_08.png
 - Image_09.png
 - Image_10.png
- Samples
 - OR_ABI-L2-CMIPF-M6C01_...
 - OR_ABI-L2-CMIPF-M6C08_...
 - OR_ABI-L2-CMIPF-M6C10_...
 - OR_ABI-L2-CMIPF-M6C12_...
 - OR_ABI-L2-CMIPF-M6C13_...
 - OR_ABI-L2-CMIPF-M6C13_...
 - OR_GLM-L2-LCFA_G16_s20...
- sample_data

Developed by: INPE / CGCT / DISSM - B
COLAB Notebook Version: September 2023



Installing Libraries and Downloading Samples

Let's explore the short version of our Colab Notebooks

<https://colab.research.google.com/drive/1mEVuUir-9J5pGM37z231dAV7gxDpt2DD?usp=sharing>

```
# 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
!mkdir -p samples # where we will store the GOES-R samples
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/
```

Install the NetCDF4 library

Install the Cartopy library

Create a folder called “samples”
(to store our samples)

Download GOES-R
samples from the Amazon
Web Services (AWS)

GOES-R Channels and Wavelengths

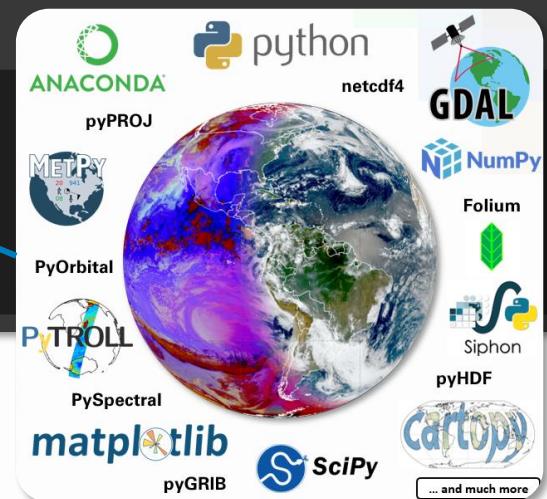
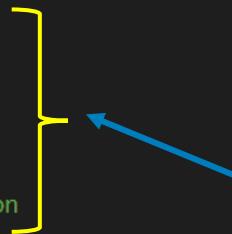
For your reference, this is a table with all ABI bands, central wavelenghts, resolutions and spectral ranges:

ABI Band	Central Wavelength (μm)	Wavelength Range (μm)	Resolution (km)	Spectral range	Descriptive Name
1	0.47	0.45 - 0.49	1	Visible	Blue
2	0.64	0.68 - 0.68	0.5	Visible	Red
3	0.86	0.847 - 0.882	1	Near Infrared	Vegetation
4	1.37	1.366 - 1.380	2	Near Infrared	Cirrus
5	1.6	1.59 - 1.63	1	Near Infrared	Snow/Ice
6	2.2	2.22 - 2.27	2	Near Infrared	Cloud Particle Size
7	3.9	3.80 - 3.99	2	Shortwave Infrared	Shortwave window
8	6.2	5.79 - 6.59	2	Midwave Infrared	Upper-level water vapor
9	6.9	6.72 - 7.14	2	Midwave Infrared	Midlevel water vapor
10	7.3	7.24 - 7.43	2	Midwave Infrared	Lower / midlevel watervapor
11	8.4	8.23 - 8.66	2	Longwave Infrared	Cloud-top phase
12	9.6	9.42 - 9.80	2	Longwave Infrared	Ozone
13	10.3	10.18 - 10.48	2	Longwave Infrared	Clean longwave window
14	11.2	10.82 - 11.60	2	Longwave Infrared	Longwave window
15	12.3	11.83 - 12.75	2	Longwave Infrared	Dirty longwave window
16	13.3	12.99-13.56	2	Longwave Infrared	CO2

Importing the Necessary Python Libraries

In this step we will import all the libraries we will need for this demonstration: "netCDF4" to open files and read the pixel values, "matplotlib" to generate images, "datetime" to read the acquisition time and date, "cartopy" to add maps and "numpy" to make some calculations when needed.

```
[ ] # 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
```



Opening a G16 File and Checking the Datasets

GOES-R File Naming Convention

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

First, we will create an object called "file" (you may call it anyway you want) and open a GOES-16 ABI image with the NetCDF4 "Dataset" command:

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

We have several datasets inside a NetCDF file (with very useful information!). With the "variables.keys()" command, we can list all the available datasets.

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

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

```
dict_keys(['CMI', 'DQE', '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'])
```

Verifying the Characteristics of the CMI Dataset

Let's check the characteristics of the "CMI" dataset (or "Cloud and Moisture Imagery", as NOAA calls it). According to the product documentation, this is the dataset where we may retrieve the "**reflectance factor**" (for visible bands) and the "**brightness temperatures**" for IR bands:

<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
```

This particular band
(channel 13 - 10.3 um) has
5424 rows x 5424 columns
and the unit is Kelvin

Please note that for this particular channel (Band 13 - 10.3 μm - 2 km - Full Disk) we have **5424 rows x 5424 columns**. For a 1 km Full Disk band we would have twice as much and for a 0.5 km Full Disk band we would have 4 times as much (plotting a 0.5 km band requires good computational power). Please also note that the unit for this particular band is Kelvin.

Considering that, let's read the CMI dataset and create a preliminary plot.

Reading and Plotting the CMI Dataset

First, we will read the CMI dataset pixel values, convert the pixels to °C and store all values from this 2D array (5424 x 5424) variable (again, you may call it anyway you want). We will use this variable to plot an image, but remember, you have all the information you need to do whatever you want with this data.

My variable name 2D array, so you can use it anyway you want (perform any calculation, make graphs, etc).

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

CMI Dataset **The whole 2D array** **Convert from K to °C**

Now, we will set the image size (in inches) and plot the "data" variable with a grayscale colormap, called "Greys" (white for temperature values and black for higher BT values) using the "imshow" command from matplotlib.

We will pass some parameters to the "imshow" command:

- "data": the variable we want to plot.
- "vmin" and "vmax": the minimum and maximum values of the colormap. You may adjust this values.
- "cmap": the colormap to be used. Please access the following link for a list of default colormaps from matplotlib:
<https://matplotlib.org/stable/tutorials/colors/colormaps.html>

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

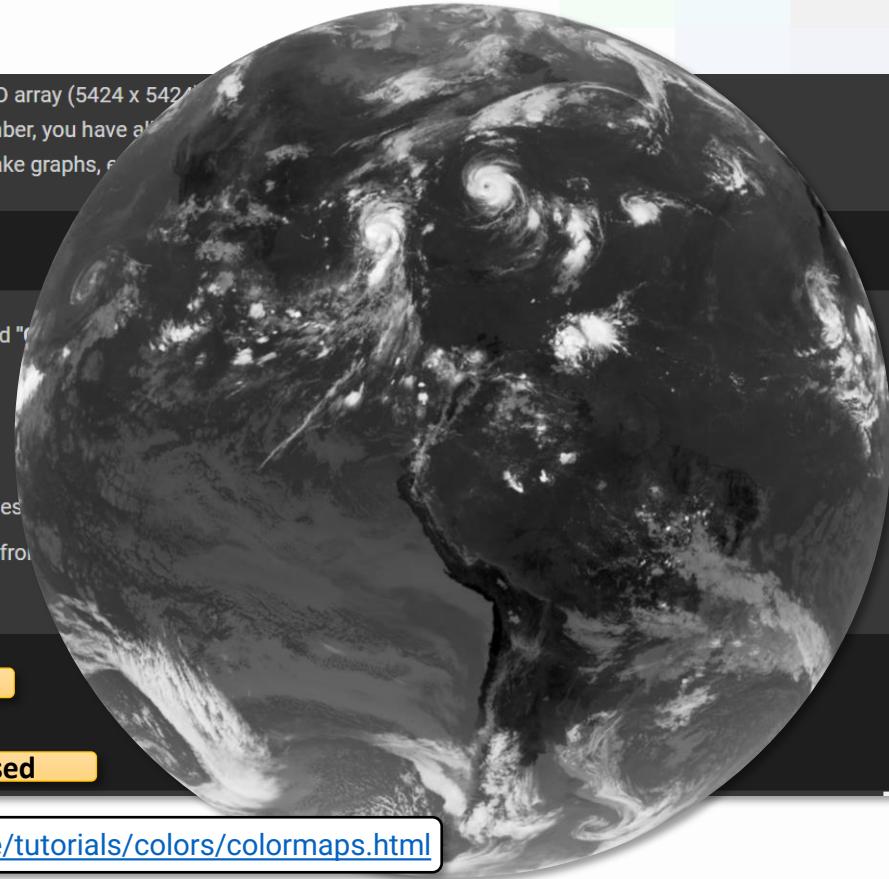
Plot size (width x height)

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

Colormap to be used

Minimum value of the colormap **Maximum value of the colormap**

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



Reading the Acquisition Time (Start of the Scan)

1 - Reading the time / date from file

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

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

2 - Informing the “datetime” library how the time / date is formatted

```
# 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 timedate 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 - How do I want to format and show it

```
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>

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 <code>+HHMM[ss].[ffffff]</code> (empty string if the object is naive).	(empty), +0000, -0400, +1030, +063415, -030712.345216	(6)
%Z	Time zone name (empty string if the object is naive).	(empty), UTC, GMT	(6)



Adding Titles and a Colorbar

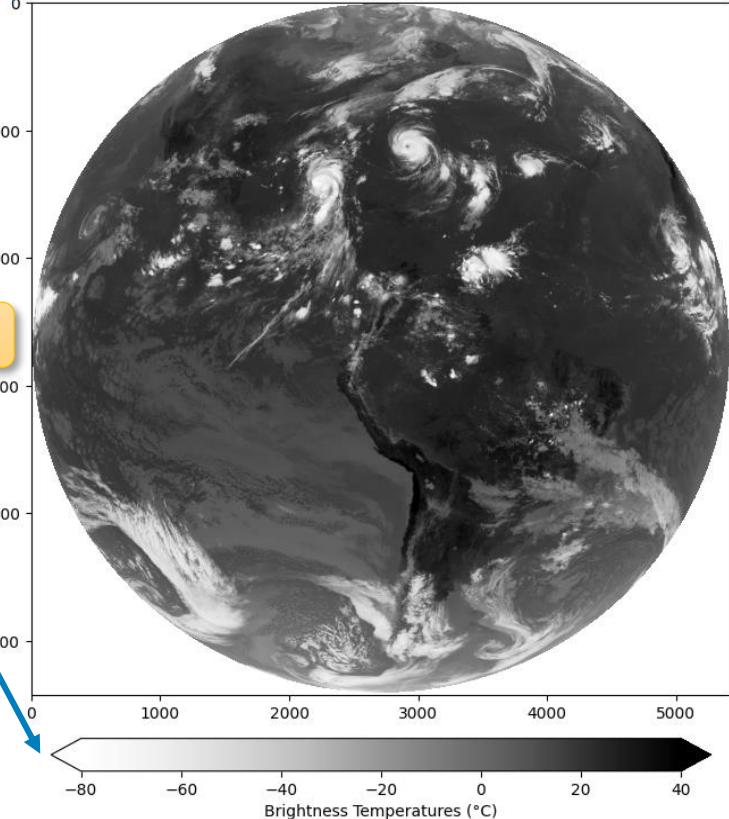
How do I want to format the time / date

```
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



Now that we already know how to read and format the acquisition time and date, let's create a plot adding the datetime string as a title. We will also add a colorbar with the "colorbar" command:

```
▶ # 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')
```

Add title on the left

Add another title on the right

Title of the colorbar

Extend the colorbar on which side?

Orientation of the colorbar

Relative distance (%) to the main plot

Size multiplier

Adding Maps With Cartopy

```
# read the satellite longitude from the metadata  
longitude = file.variables['goes_imager_projection'].longitude_of_projection_origin
```

Read the satellite central longitude from the metadata

```
# read the satellite altitude from the metadata  
height = file.variables['goes_imager_projection'].perspective_point_height
```

Read the satellite height from the metadata

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

Tell Cartopy the projection of our data - “geostationary”
in this case, passing the central longitude and satellite
height as parameters

```
# 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)
```

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



Add country borders with Cartopy

Add coastlines with Cartopy

Add gridlines with Cartopy

Resulting plot

... now let's see how to plot this map
with our GOES-R ABI image

Adding Maps With Cartopy

We also need to inform Cartopy the “extent” of our data

We also need to inform the "extent" of our data. According to the product documentation (<https://www.goes-r.gov/products/docs/PUG-L2+-vol5.pdf>), the extent can be calculated by multiplying half of the full disk by the pixel size in radians by the satellite height. Again, all this information is provided in the metadata. Calculating this by reading the metadata is very useful: as we will see, we can use the same scripts to plot the Full Disk, CONUS and Mesoscale sectors:

Paragraph 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 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)
print(img_extent)
```

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

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

According to the GOES-R documentation, the extent is:

Number of pixels (half of full disk) x pixel size (radians) x satellite height

This first multiplication is already available in the metadata, called as 'x' and 'y' datasets

This is what Cartopy needs

Adding Maps With Cartopy

All the instructions needed to create the image on the right

```
# 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)

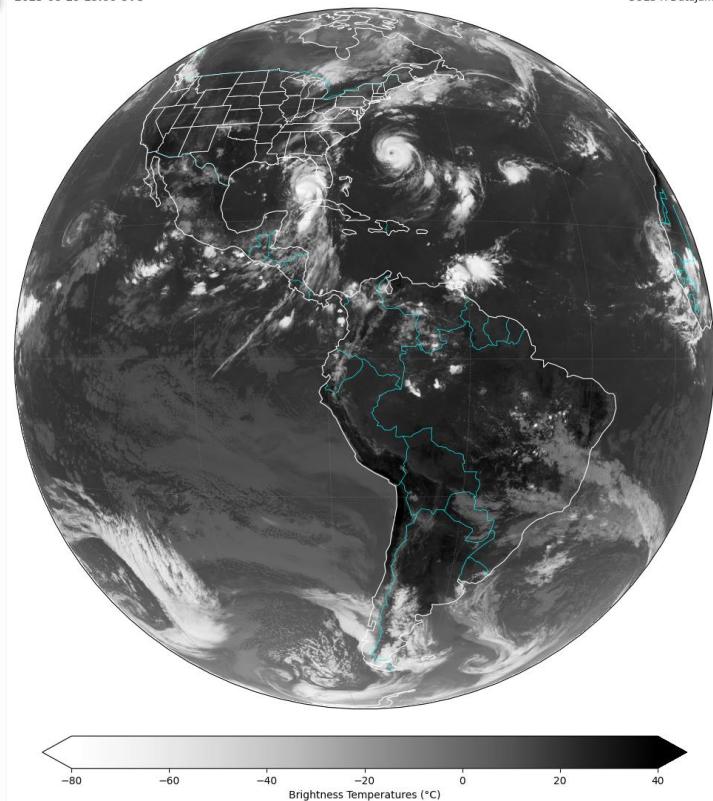
# add a title (one on the left and one on the right)
plt.title(f'GOES-16 Band 13 (10.3 μm)\n[{date}]', fontweight='bold', fontsize=10, loc='left')
plt.title('GOES-R DataJam', fontsize=10, loc='right')
```

DATA READING AND MANIPULATION

PLOT CONFIGURATION

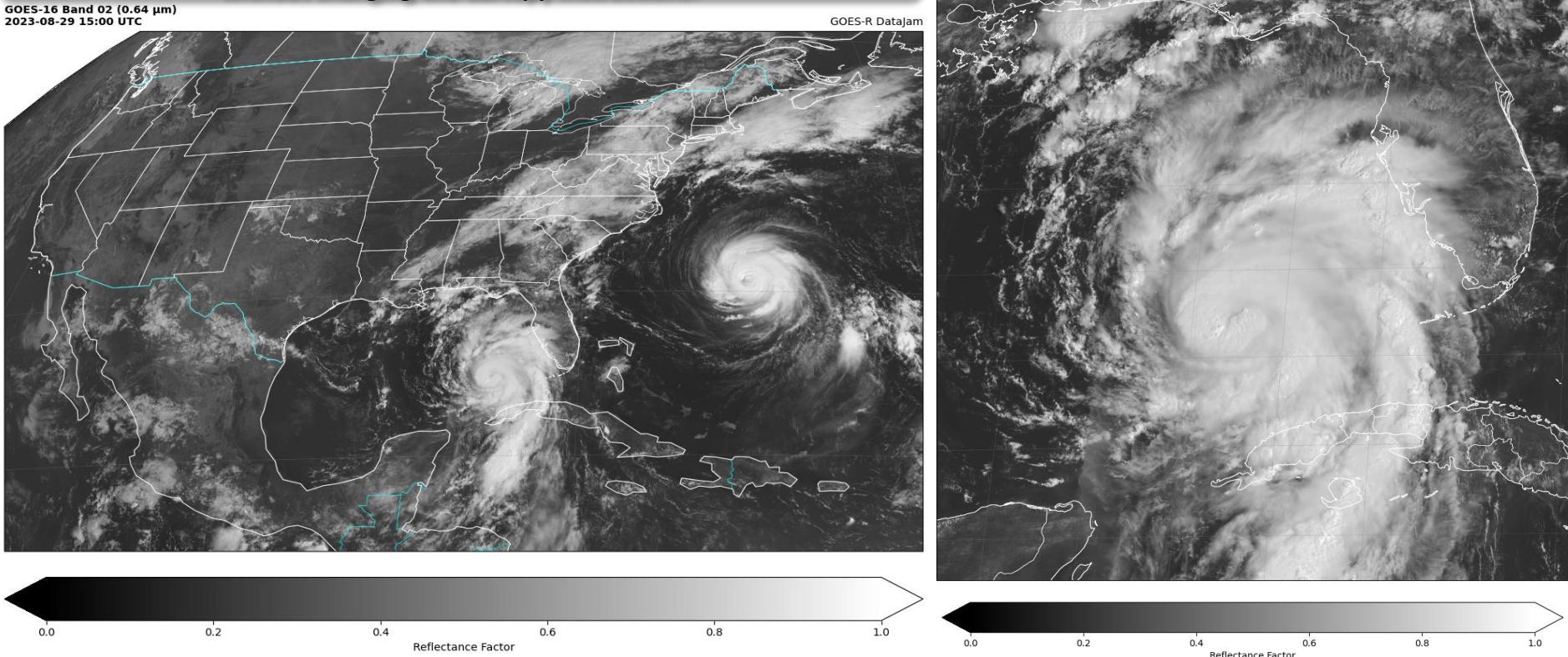
GOES-16 Band 13 (10.3 μm)
2023-08-29 15:00 UTC

GOES-R DataJam



Plotting CONUS and MESOSCALES

We are reading parameters from the metadata, so we can plot other sectors without changing the Cartopy instructions!



For Your Future Reference

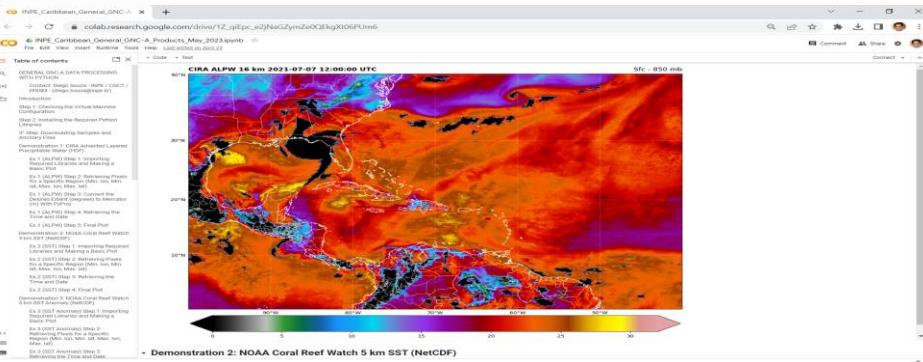
GOES-R DATA AND PRODUCTS - LONGER VERSION WITH ABI, GLM, RGBs AND MUCH MORE!

https://colab.research.google.com/drive/1_QjqpUEBmABudnshTJn3EPS-Pvtma8wv?usp=sharing



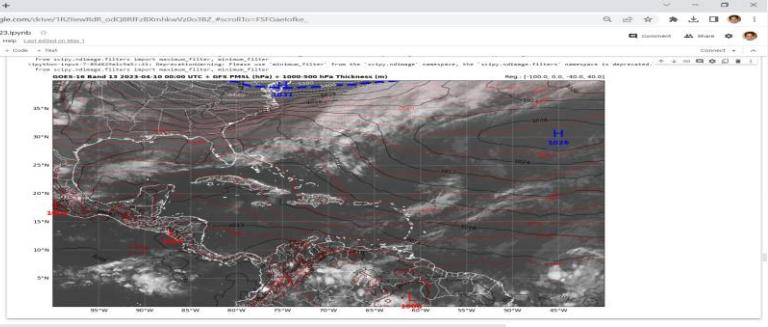
MISCELLANEOUS PRODUCTS (TPW, VEGETATION, HOTPOSTS, ETC)

https://colab.research.google.com/drive/1Z_qiEpc_e2jNaGZymZe0QEkgXt06PUm6?usp=sharing



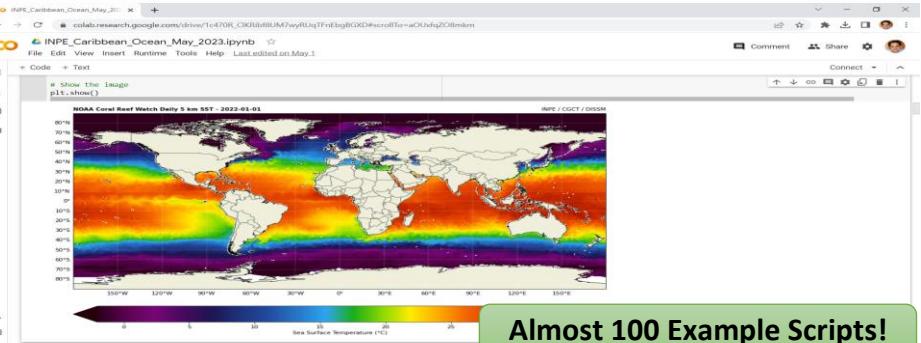
NWP (VARIOUS PLOT TYPES, OVERLAY WITH SATELLITES, METAR, ETC)

<https://colab.research.google.com/drive/17yFOKezLEUhIziUUUXNlbYEwndp-gAmp?usp=sharing>



OCEANOGRAPHIC PRODUCTS

https://colab.research.google.com/drive/1c470R_CIKRibf8UM7wyRUqTFnEbgBGXD?usp=sharing



Almost 100 Example Scripts!

For Your Future Reference

<https://goesrdatajam.sched.com/event/1RI91/visualizing-and-analyzing-goes-r-data-using-google-colab>



Procedure to process GOES-R data locally:

<https://geonetcast.wordpress.com/2023/03/19/getting-started-with-python-and-satellite-imagery/>

 Main Slides From Recording [PDF](#)

 Extra Introduction To Python [PDF](#)

 Extra GOES R Data Processing With Python [PDF](#)

 Extra NWP Data Processing With Python [PDF](#)

 Extra Introduction To Visual Studio Code [PDF](#)

Slides used for this presentation

Extra resources:

- Longer introduction to Python
- Slide deck explaining each instruction from the long version of the notebook.
- Slide deck explaining how to process NWP data
- Slide deck explaining how to use Visual Studio Code, a very nice IDE (Integrated Development Environment) to process data locally.

Monday September 18, 2023 12:15pm - 12:30pm EDT

Virtual

 Trainer Material, Coding Examples, ABI, GLM, Google Colab, Google

Some Useful Extra Resources

Practical introduction to Python from scratch: There are many materials available for free

E-books distributed free of charge by the authors (CC licenses): <https://www.freetechbooks.com/python-f6.html>

<https://www.freetechbooks.com/a-practical-introduction-to-python-programming-t1376.html>

A Practical Introduction to Python Programming
An introductory programming text for students with no prior programming experience.

Tag(s): [Introduction to Computer Programming](#) [Python](#)

Publication date: 01 Dec 2012 **Type:** Book
ISBN-10: n/a **Publisher:** n/a
ISBN-13: n/a **License:** [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported](#)
Paperback: 263 pages **Paperback:** 495 pages
Views: 13,797 **Post time:** 02 Jul 2021 01:00:00

Summary/Excerpts of (and not a substitute for) the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported:

You are free to:

Share — copy and redistribute the material in any medium or format
Adapt — remix, transform, and build upon the material

The licensor cannot revoke these freedoms as long as you follow the license terms.

Click [here](#) to read the full license.

<https://www.freetechbooks.com/dive-into-python-3-t1008.html>

Dive Into Python 3
Dive Into Python 3 covers Python 3 and its differences from Python 2. Compared to Dive Into Python, it's about 20% revised and 80% new material.

Tag(s): [Python](#)

Publication date: 31 Dec 2011 **Type:** N/A
ISBN-10: 1430224150 **Publisher:** [APress](#)
ISBN-13: 9781430224150 **License:** [Creative Commons Attribution-ShareAlike 3.0 Unported](#)
Paperback: 495 pages **Post time:** 23 May 2016 12:00:00
Views: 9,804

Summary/Excerpts of (and not a substitute for) the Creative Commons Attribution-ShareAlike 3.0 Unported:

You are free to:

Share — copy and redistribute the material in any medium or format
Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Click [here](#) to read the full license.

<https://www.freetechbooks.com/a-whirlwind-tour-of-python-t1350.html>

A Whirlwind Tour of Python
This free e-book is a fast-paced intro to Python aimed at researchers and engineers.

Tag(s): [Data Science](#) [Python](#)

Publication date: 01 Aug 2016 **Type:** Book
ISBN-10: n/a **Publisher:** [O'Reilly Media, Inc.](#)
ISBN-13: 9781491964644 **License:** [Creative Commons CC0 "No Rights Reserved"](#)
Paperback: 98 pages **Post time:** 29 Jul 2020 06:00:00
Views: 12,697

Summary/Excerpts of (and not a substitute for) the Creative Commons CC0 "No Rights Reserved":

CC0 enables scientists, educators, artists and other creators and owners of copyright- or database-protected content to waive those interests in their works and thereby place them as completely as possible in the public domain, so that others may freely build upon, enhance and reuse the works for any purposes without restriction under copyright or database law.

In contrast to CC's licenses that allow copyright holders to choose from a range of permissions while retaining their copyright, CC0 empowers yet another choice altogether - the choice to opt out of copyright and database protection, and the exclusive rights automatically granted to creators - the "no rights reserved" alternative to our licenses.

<https://www.freetechbooks.com/a-practical-introduction-to-python-programming-t1376.html>

<http://getpython3.com/diveintopython3/>

<https://jakevdp.github.io/WhirlwindTourOfPython/>

...and much more

GOES-R

DataJam

THANK YOU!

Visualizing and Analyzing GOES-R Data Using Google Colab



Diego Souza

diego.souza@inpe.br

Engineer - INPE (Brazil)
National Institute for Space Research

Marcial Garbanzo

marcial.garbanzo@ucr.ac.cr

Professor - UCR
University of Costa Rica

