

GEONETCast-Americas Training for the Eastern Caribbean States

Day 3 - May 8th

Session 4:

Hands-on NWP Data Processing



Diego Souza
diego.souza@inpe.br

DISSM - Meteorological Satellites and Sensors' Division
CGCT - General Coordination of Earth Sciences
INPE - National Institute for Space Research



Caribbean Institute
for Meteorology
and Hydrology



Presentation Outline

- **Basic Concepts**

- Available Variables

- Basic Plot / Pixel Values / Metadata

- Maps / Shapefile

- Contours / Custom Color Palettes

- Animations

- **More Advanced Concepts**

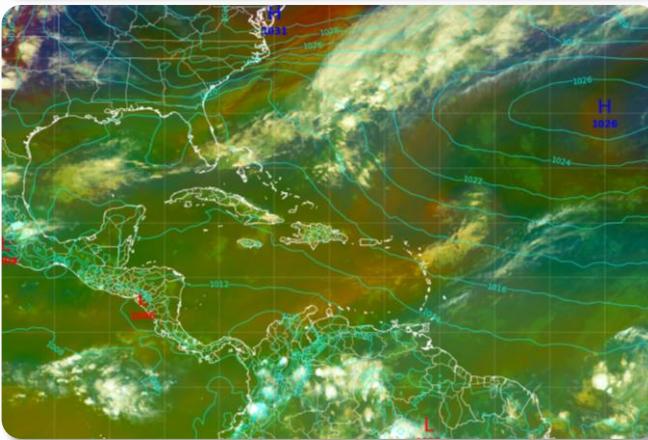
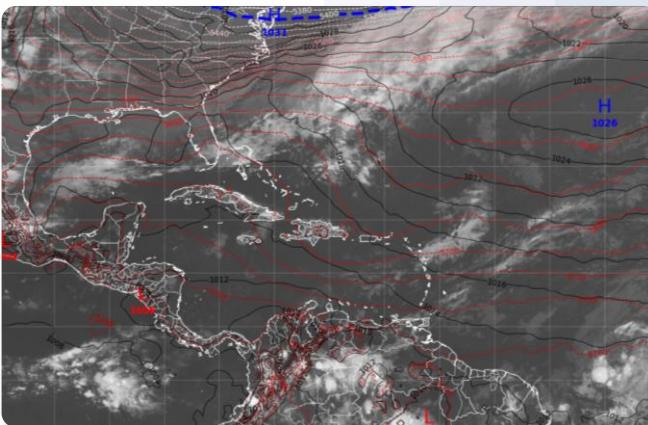
- Downloading Data Using Scripts

- Averages / Minimum / Maximums

- Streamlines / Vectors / Barbs

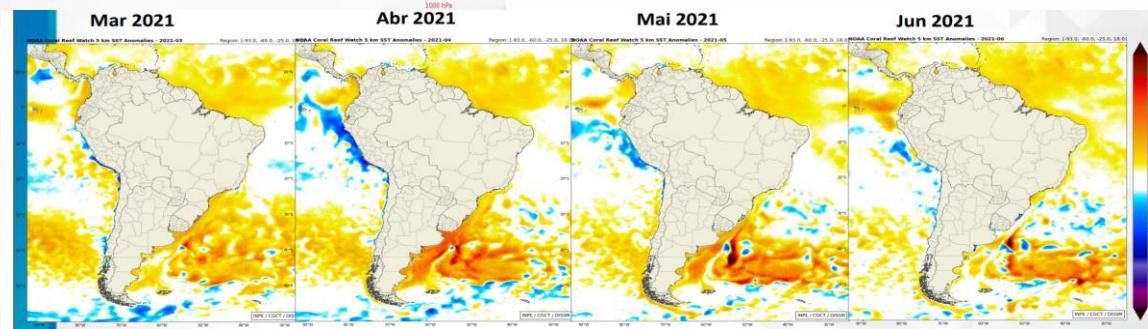
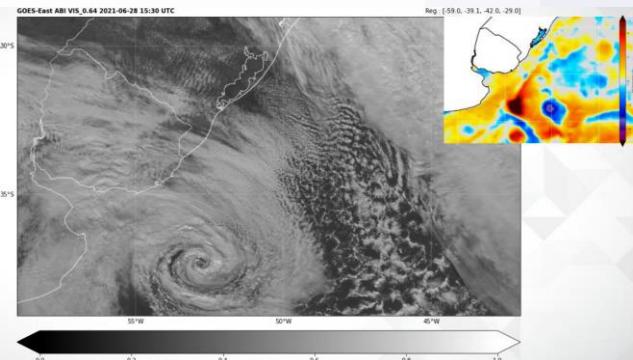
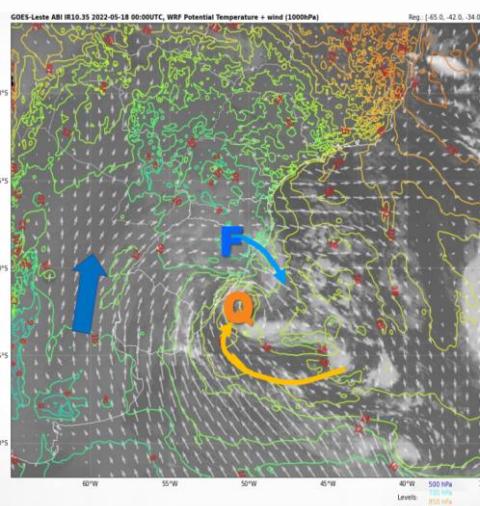
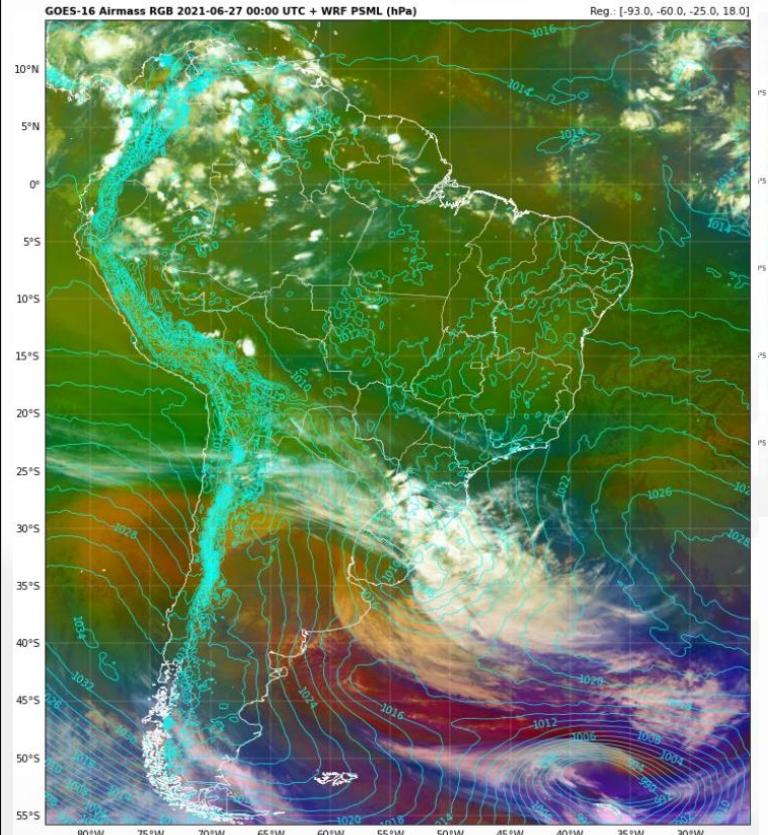
- Plotting a “Tool” (Galvez Davison Index)

- METAR + Satellite + NWP



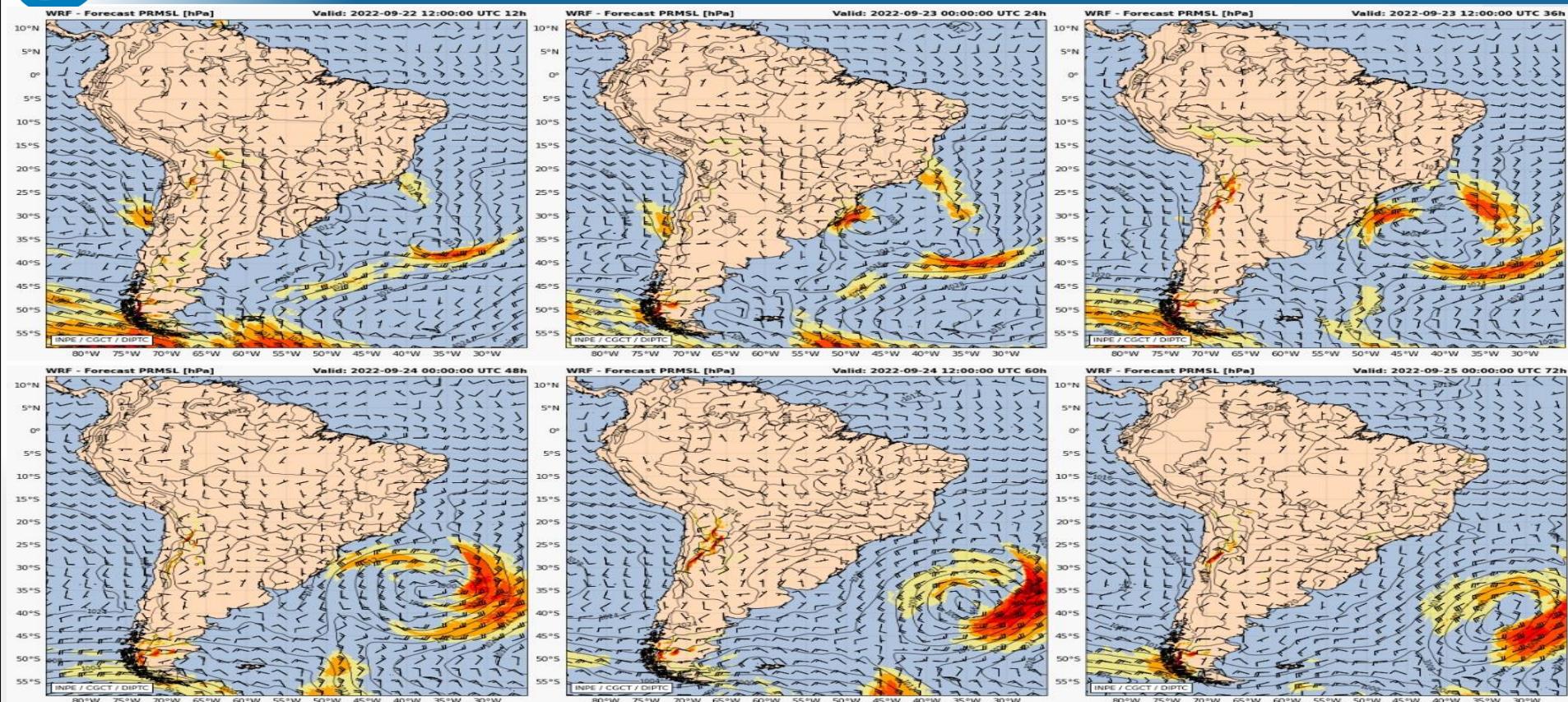


Goal: Create Your Own Plots, With Any Combination



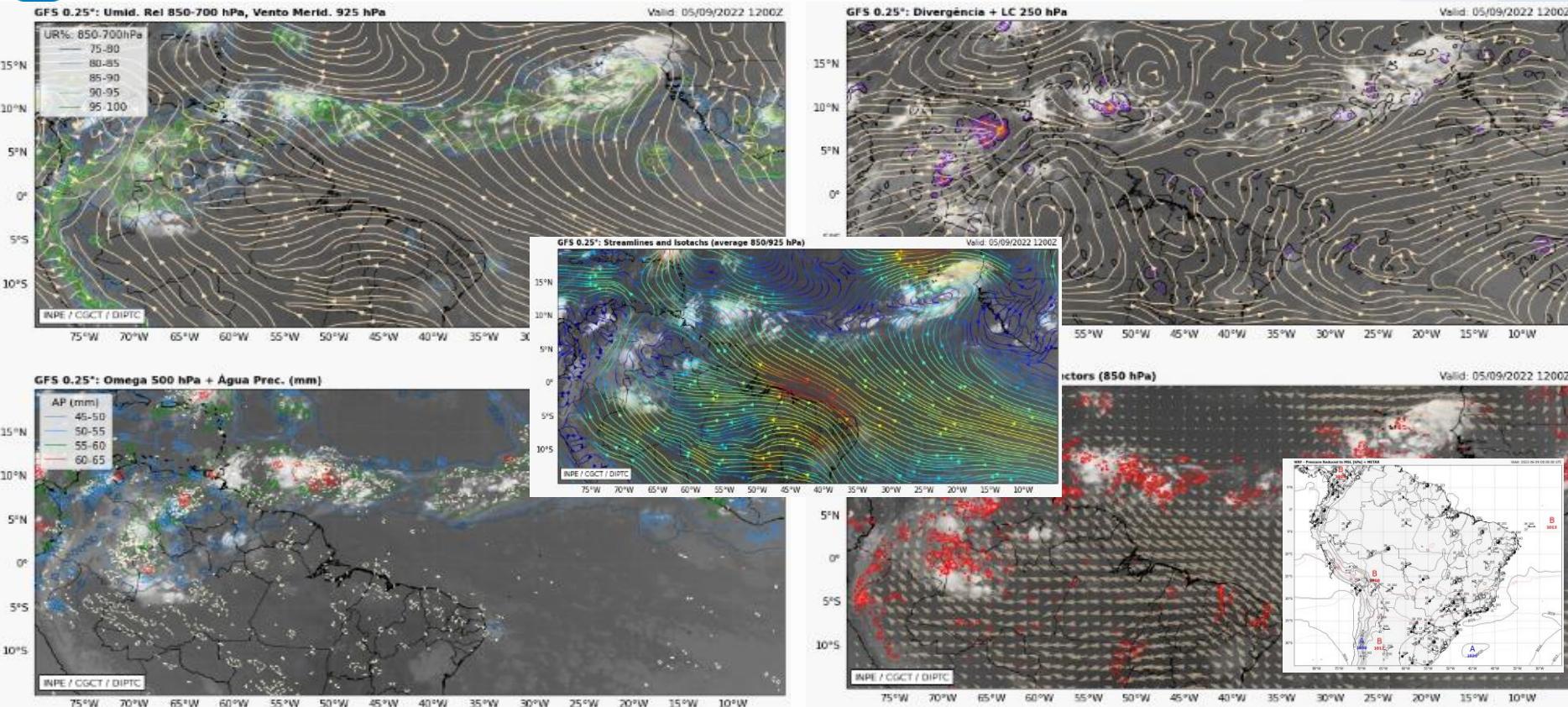
Credits: Regina Ito (INPE / DISSM / CGCT)

Goal: Create Your Own Plots, With Any Combination



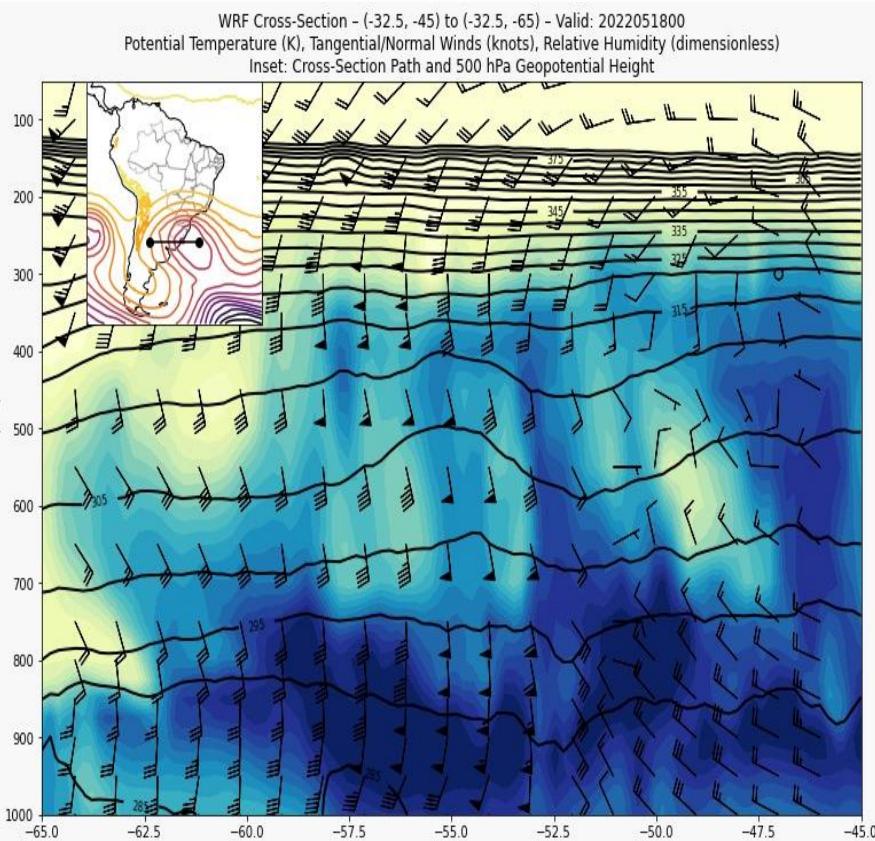
Credits: Marcos Vianna (INPE / DIPTC / CGCT)

Goal: Create Your Own Plots, With Any Combination



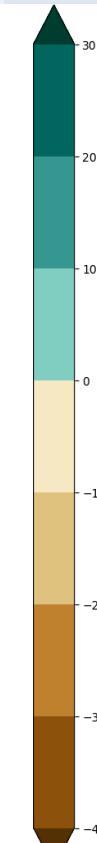
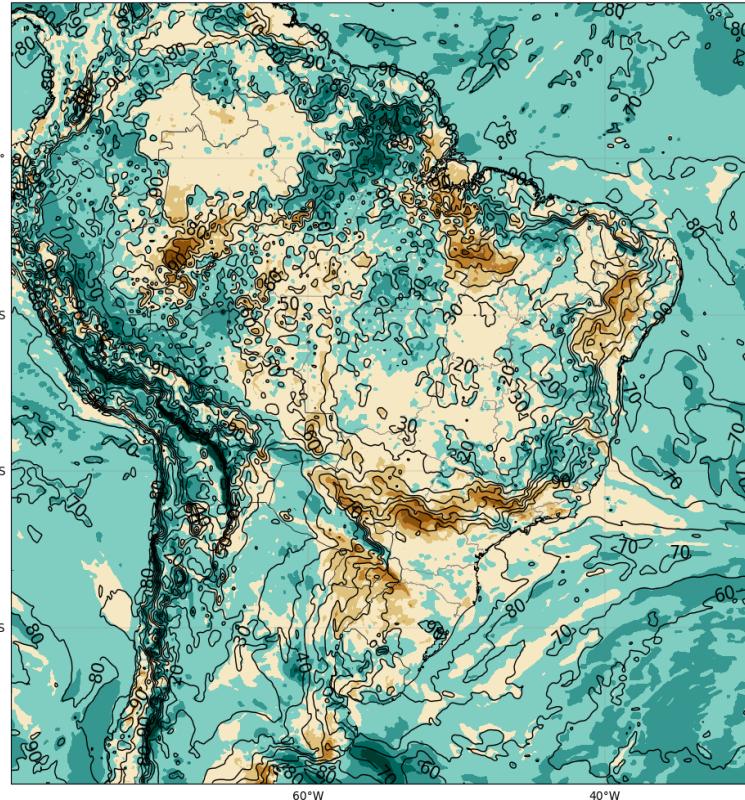
Credits: Marcos Vianna (INPE / DIPTC / CGCT)

Goal: Create Your Own Plots, With Any Combination



Credits: Diogo Arsego (INPE / DIPTC / CGCT)

a) WRF: Umidade Relativa (linha de contorno) e vies (P24 - analise)



PRE-COURSE ACTIVITY

Installing the necessary tools to run scripts locally and making preliminary plots.

ONLINE COURSE - PART I

- **Accessing NWP Data**

Examples of online numerical weather prediction model data repositories and how to download them.

- **Knowing the Data**

Understanding the datasets available in GRIB files using Python scripts.

- **Initial Plots**

Creating basic plots, visualizing pixel values, reading the metadata, adding legends, maps and smoothing- contours.

- **Working With Multiple Files**

Processing several files using the same script (e.g.: calculation of maximums, minimums and averages) and generating animations.

- **Data Access Using Scripts**

In this topic we will learn how to download numerical model data using scripts.

ONLINE COURSE - PART II

- **Learning Different Types of Plots**

In this topic we will see how to read NWP fields by level, create simultaneous plots, with streamlines, wind vectors and barbs.

- **Reading Various Fields**

In this topic we will create the GDI tool, exemplifying how to read several fields and perform more elaborate calculations.

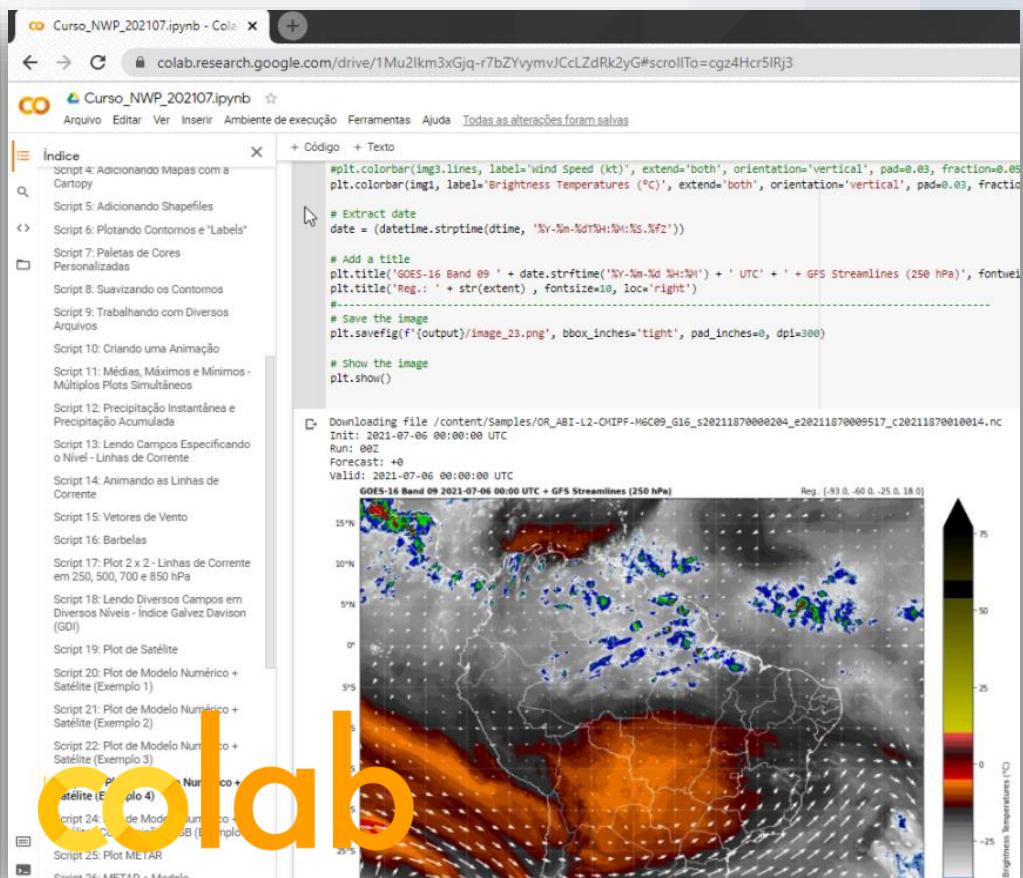
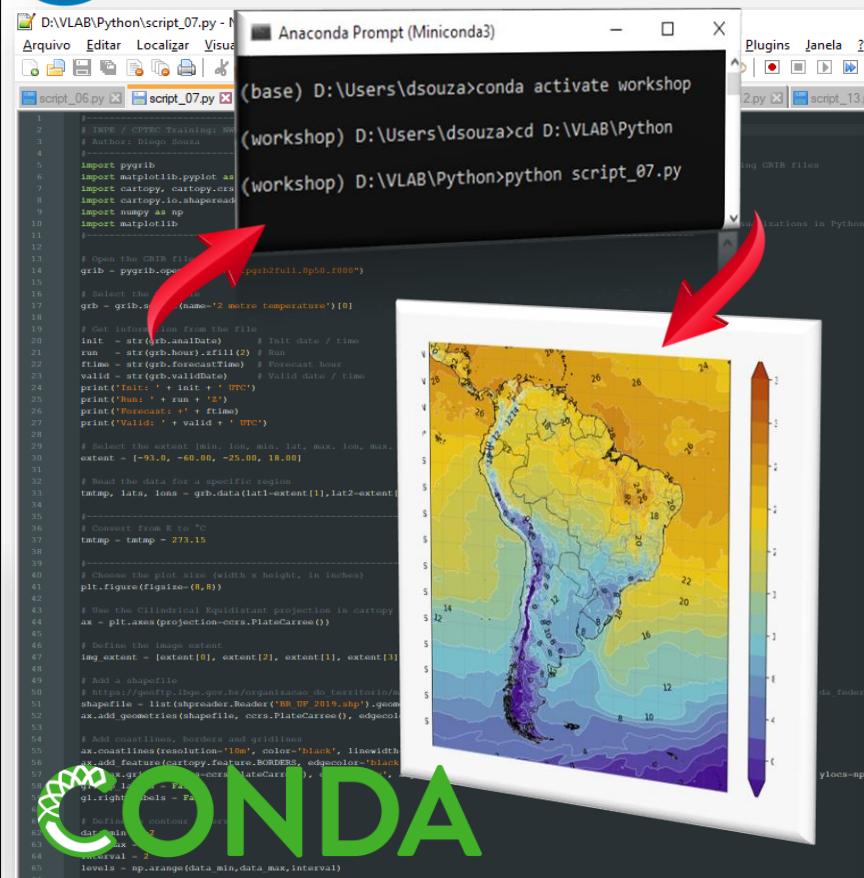
- **METAR + NWP + Satellite**

We will review how to read GOES-16 data, learn how to read METAR data and how to overlay these data with NWP data.

- **Case Studies**

Seeing these concepts in practice through case studies.

Running Scripts Locally or in the Cloud



Running Scripts Locally

password: workshop

<https://geonetcast.wordpress.com/2023/03/20/getting-started-with-python-and-nwp-data/>

Objectives of the Activity

The screenshot shows the GNC-A website with a navigation bar including Home, Blog Content, The GEONETCast System, GEONETCast-Americas, Documentation, Data Processing, User Interaction, GNC-A Forum, Contact, and Author. Below the navigation is a post titled "Protected: Getting Started With Python and NWP Data" dated May 03 - 15 / 2023. The post includes a thumbnail image of a flight deck and a screenshot of a computer monitor displaying a weather map.

Protected: Getting Started With Python and NWP Data



Contact: If you have any questions, please contact:

E-mail: diego.sousa@inpe.br

Learning Objectives: By the end of this activity, participants will

- Learn how to install and use basic tools to start manipulating numerical weather prediction data with Python in your computers.
- Carrying out operations such as:

- Reading GRIB files
- Checking the available fields in a NWP GRIB file
- Create a basic plot visualize pixel values
- Modify color palette, read metadata, add a title and legend to the plot
- Add maps and shapes
- Create contour plots
- Create custom color palettes
- Soften the contours
- Process multiple files
- Creating animations

During the virtual course, we will also learn:

- Data download using scripts.
- Matplotlib plots.
- Read several NWP fields, performing more elaborate calculations.
- Satellite Plots + NWP.
- METAR plots + Satellite + NWP.
- Among other operations!

Estimated duration of this pre-course activity: 2 hours

Notes: This document is only for the use of participants in this course or permission is granted by the author.

Installation of Tools

The screenshot shows the Miniconda Python distribution installation interface. It displays the "INSTALLATION STEPS" section with the first step being "Download and install Miniconda for Python 3.10 at the following link (approximately 60 MB):". Below this is a link to the official Conda repository: <https://docs.conda.io/en/latest/miniconda.html>. The interface shows the progress of the download and the steps required to complete the installation.

Access to Samples (GNC or Cloud)

The screenshot shows a web-based interface for accessing numerical weather prediction (NWP) data samples. It lists "GLOBAL MODELS" with various datasets and their details. The "Data Set" column lists datasets like "GFS0.25", "GFS0.25 0.25", "GFS 0.25 Degree", "GFS 0.25 Degree Hourly", "GFS 0.25 Degree (secondary format)", "GFS 0.25 Degree", and "GFS 0.125 Degree". The "freq" column indicates frequency (e.g., 4 hours), and the "grib filter" column shows options like "grb" and "grb2". Below the table, it says "Choosing the desired data on the NOMADS server".

Local Data Processing

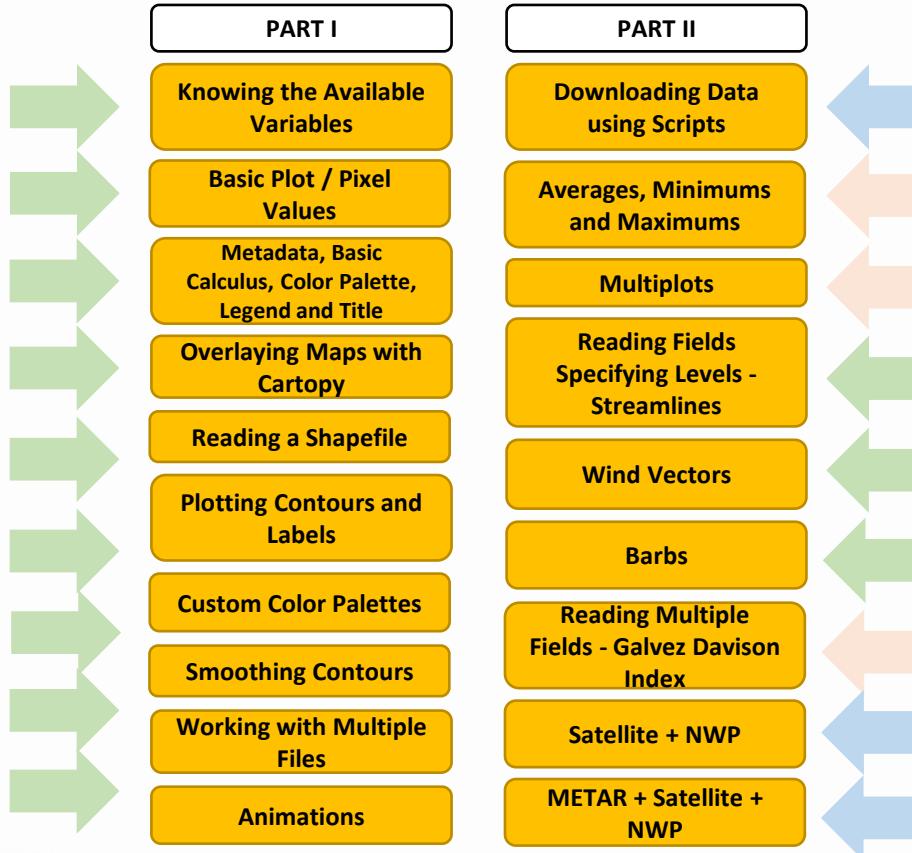
The screenshot shows a terminal window titled "Data Transfer: NCEP GFS Forecasts (0.50 degree grid)". It displays a command-line interface for transferring data from a NOMADS server. The command shown is "grib2 1985-01-01.nc & rm -rf ./tmp/nomads_ncas_gfs/grib2/1985-01-01.nc". The terminal also shows a preview of a GFS temperature forecast map for 1985-01-01 at 00Z.

Part 1: Basic Concepts

Higher priority

Medium priority

Low priority



Accessing NWP Data (GEONETCast-Americas)

ess@j0153-dpsg:ess/data/satellite/fazzt

File Edit View Search Terminal Help

```
[ess@j0153-dpsg ~]$ cd /ess/data/satellite/fazzt/
[ess@j0153-dpsg fazzt]$ ls -l
total 725560
drwxr-xr-x 6 fazzt fazzt 79 Mar  1 07:40 CIMSS
drwxr-xr-x 2 fazzt fazzt 28672 Mar  2 18:04 CIRA
drwxr-xr-x 2 fazzt fazzt 18702336 Mar 29 19:39 EUMETSAT
drwxr-xr-x 18 fazzt fazzt 230 Mar  3 00:13 GOES-R-CMI-Imagery
drwxr-xr-x 2 fazzt fazzt 52404224 Mar 29 19:43 GOES-R-DCS
drwxr-xr-x 2 fazzt fazzt 31903744 Mar 29 19:43 GOES-R-GLM-Products
drwxr-xr-x 26 fazzt fazzt 328 Mar  3 00:42 GOES-R-Level-2-Products
drwxr-xr-x 5 fazzt fazzt 48 Mar  2 00:03 GOES-T-CMI-Imagery
drwxr-xr-x 2 fazzt fazzt 4096 Mar  3 14:04 IMN-CostaRica
drwxr-xr-x 2 fazzt fazzt 114 Mar  3 11:05 Info&Admin
drwxr-xr-x 2 fazzt fazzt 77824 Mar 29 19:03 INPE
drwxr-xr-x 2 fazzt fazzt 2150400 Mar 29 19:33 ISCS-ADMIN
drwxr-xr-x 2 fazzt fazzt 970752 Mar 29 19:39 ISCS-ANLZ-CLIMATE
drwxr-xr-x 2 fazzt fazzt 1507328 Mar 29 19:26 ISCS-BUFR
drwxr-xr-x 2 fazzt fazzt 27742208 Mar 29 19:42 ISCS-FCAST
drwxr-xr-x 2 fazzt fazzt 135557120 Mar  3 16:50 ISCS-GRIB1
drwxr-xr-x 2 fazzt fazzt 62435328 Mar  3 16:46 ISCS-GRIB2
drwxr-xr-x 2 fazzt fazzt 1241088 Mar 29 19:37 ISCS-PIC
drwxr-xr-x 2 fazzt fazzt 4493312 Mar 29 19:39 ISCS-RADAR
drwxr-xr-x 2 fazzt fazzt 1695744 Mar 29 19:26 ISCS-SAT
drwxr-xr-x 2 fazzt fazzt 223494144 Mar 29 19:43 ISCS-SURFACE
drwxr-xr-x 2 fazzt fazzt 59944960 Mar 29 19:39 ISCS-UA
drwxr-xr-x 2 fazzt fazzt 6193152 Mar 29 19:42 ISCS-WARN
drwxr-xr-x 3 fazzt fazzt 22 Mar  3 00:03 IPSS
drwxr-xr-x 2 fazzt fazzt 4096 Mar  3 06:14 MARN-EL Salvador
drwxr-xr-x 4 fazzt fazzt 39 Mar  3 00:03 MSG-0degree
drwxr-xr-x 2 fazzt fazzt 442368 Mar  3 14:16 NOAA-NESDIS
drwxr-xr-x 2 fazzt fazzt 319 Mar 29 19:33 RANET
drwxr-xr-x 2 fazzt fazzt 186 Mar  3 10:02 Test
drwxr-xr-x 2 fazzt fazzt 52 Mar  3 16:46 USEPA
[ess@j0153-dpsg fazzt]$
```

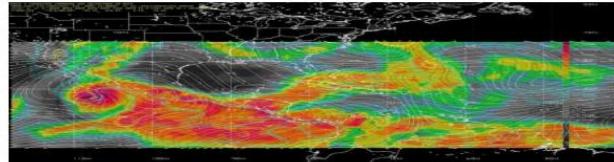
</ess/data/satellite/fazzt/MARN-El Salvador>

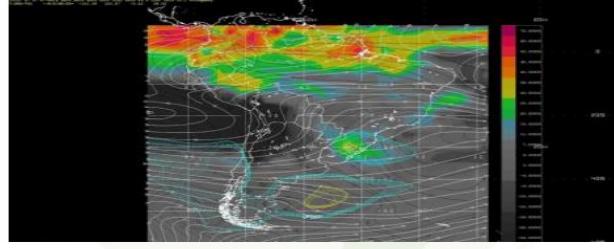
 GEONETCast-Americas <small>ILLUSTRATED PRODUCT CATALOG</small>	
---	---

5 FORECAST PRODUCTS

5.1 GFS (Global Forecast System)

- 0.5° South America and Caribbean Forecast Products





Data Provider: MARN El Salvador
 Folder: MARN-El Salvador
 Format: GRIB2
 Frequency: 2 cycles per day (00h and 12h), 40 files per cycle, 80 files per region (160 files per day). Average Size: ~10 MB (Central America and Caribbean) / 14 MB (South America) – 2 GB per region.
 Spatial resolution: 0.5 degrees
 Naming Convention: gfs_RRR_Op50_CC.f0FFF, Where:
 RRR: Region (ctr: Central America + Caribbean / sam: South America)
 CC: Execution Cycle (00 and 12 UTC) | FFF: Forecast (0 – 120 h, every 3 hours)

 Python Script Available

Note: Please access the list of GFS bands and their meaning at [this link](#).

INPE - National Institute for Space Research - Meteorological Satellites and Sensors's Division

Page 50 of 70

Accessing NWP Data (GEONETCast-Americas)

ess@j0153-dpsg:ess/data/satellite/fazzt

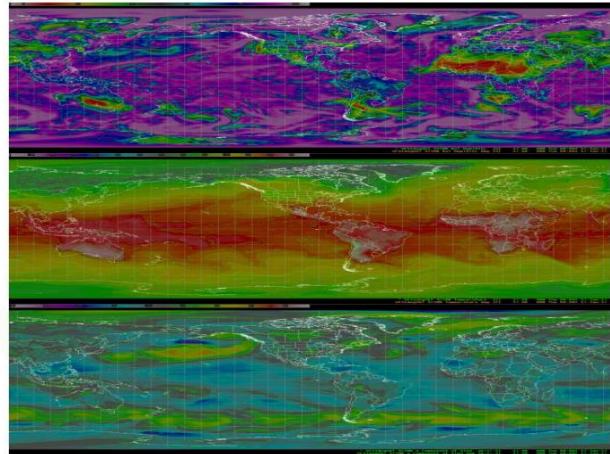
File Edit View Search Terminal Help

```
[ess@j0153-dpsg ~]$ cd /ess/data/satellite/fazzt/
[ess@j0153-dpsg fazzt]$ ls -l
total 725560
drwxr-xr-x  6 fazzt fazzt      79 Mar  1 07:40 CIMSS
drwxr-xr-x  2 fazzt fazzt    28672 Mar  2 18:04 CIRA
drwxr-xr-x  2 fazzt fazzt  18702336 Mar 29 19:39 EUMETSAT
drwxr-xr-x 18 fazzt fazzt      230 Mar  3 00:13 GOES-R-CMI-Imagery
drwxr-xr-x  2 fazzt fazzt  52404224 Mar 29 19:43 GOES-R-DCS
drwxr-xr-x  2 fazzt fazzt  31903744 Mar 29 19:43 GOES-R-GLM-Products
drwxr-xr-x 26 fazzt fazzt     328 Mar  3 00:42 GOES-R-Level-2-Products
drwxr-xr-x  5 fazzt fazzt      48 Mar  2 00:03 GOES-T-CMI-Imagery
drwxr-xr-x  2 fazzt fazzt    4096 Mar  3 14:04 IMN-CostaRica
drwxr-xr-x  2 fazzt fazzt     114 Mar  3 11:05 Info&Admin
drwxr-xr-x  2 fazzt fazzt   77824 Mar 29 19:03 INPE
drwxr-xr-x  2 fazzt fazzt  2150400 Mar 29 19:33 ISCS-ADMIN
drwxr-xr-x  2 fazzt fazzt   970752 Mar 29 19:39 ISCS-ANLZ-CLIMATE
drwxr-xr-x  2 fazzt fazzt  1507328 Mar 29 19:26 ISCS-BUFR
drwxr-xr-x  2 fazzt fazzt  27742208 Mar 29 19:42 ISCS-FCAST
drwxr-xr-x  2 fazzt fazzt 1355557120 Mar  3 16:50 ISCS-GRIB1
drwxr-xr-x  2 fazzt fazzt  62435328 Mar  3 16:46 ISCS-GRIB2
drwxr-xr-x  2 fazzt fazzt 1241088 Mar 29 19:37 ISCS-PIC
drwxr-xr-x  2 fazzt fazzt  4493312 Mar 29 19:39 ISCS-RADAR
drwxr-xr-x  2 fazzt fazzt  1695744 Mar 29 19:26 ISCS-SAT
drwxr-xr-x  2 fazzt fazzt 223494144 Mar 29 19:43 ISCS-SURFACE
drwxr-xr-x  2 fazzt fazzt  59944960 Mar 29 19:39 ISCS-UA
drwxr-xr-x  2 fazzt fazzt  6193152 Mar 29 19:42 ISCS-WARN
drwxr-xr-x  3 fazzt fazzt     22 Mar  3 00:03 JPSS
drwxr-xr-x  2 fazzt fazzt   4096 Mar  3 06:14 MARN-EL Salvador
drwxr-xr-x  4 fazzt fazzt     39 Mar  3 00:03 MSG-0degree
drwxr-xr-x  2 fazzt fazzt  442368 Mar  3 14:16 NOAA-NESDIS
drwxr-xr-x  2 fazzt fazzt    319 Mar 29 19:33 RANET
drwxr-xr-x  2 fazzt fazzt    186 Mar  3 10:02 Test
drwxr-xr-x  2 fazzt fazzt     52 Mar  3 16:46 USEPA
[ess@j0153-dpsg fazzt]$
```

</ess/data/satellite/fazzt/ISCS-GRIB2>



- 1.0° Global Forecast Products



Data Provider: NWS
 Folder: ISCS-GRIB2
 Format: GRIB2
 Frequency: 4 cycles per day (00h, 06h, 12h and 18h), 0 ~ 240 h Forecast
 Average Size, per file: 62 KB
 Spatial Resolution: 1 degree
 Naming Convention: T_PPPPPPKWCDDHHMM_C_KWBC_YYYYMMDDxxxxx_xxxxxxx-xxxx.bin
 Where:
 PPPPP: Product Type, forth letter varies from "A" (+0h) to "Y" (+240h),
 DDHHMM: Day, Hour, Minutes from model run (0000 / 0600 / 1200 / 1800)
 YYYYMMDD: Year, month and day for that given run

Python Script Available

Note: Please access the list of GFS fields and their nomenclature at [this link](#).



Accessing NWP Data (NOMADS)

Data Download in GRIB Format

<https://nomads.ncep.noaa.gov/>

Click on “grib filter” to filter parameters (region, levels, variables)

National Weather Service

NCEP Central Operations

Home News Organization

Local forecast by "City, St"
City, St Go
Search NCEP

Current Hazards
Watch/Warnings
Outlooks
National
Current Conditions
Observations
Satellite Images
Radar Imagery
Lakes & Rivers
Space Weather
Unified Surface Analysis
Northern Hemisphere
Surface Analysis
Product Loops
Environmental Models
Product Info
Current Status
Model Analyses & Guidance
Forecasts
Current
6 to 10 Day
Aviation
Hurricane
Marine
Tropical Marine
Fire Weather
Forecast Maps
Climate
Climate Prediction
Climate Archives
Weather Safety
Storm Ready
NOAA
Central Library
Photo Library
Public Affairs
Employment
Education
Resources
Question of the Month

NOMADS

NOAA Operational Model Archive and Distribution System

Description of NOAA's NOMADS servers hosting NCEP model data

Help Desk: Questions or problems please use the link to submit a service ticket.

Background: Background documents about the NOMADS project.

Service Description: OCWWS Service Description Document

Archive of Model Data: Some data sets are archived by NCEI, check availability with the link.

Terms of Data Usage: Use this link to view the Term of Data Usage / Disclaimer.

Based on user requests NCEP will now be utilizing a subscriber list to make announcements with regards to all of our data servers. [Follow this link](#) to subscribe and review the list details.

If you have issues with any NCEP server please send your email to:
ncep.pmb.dataflow@noaa.gov

Include as much information about the error as you can - your connection information, the syntax of your request, when the problem started, the error reported, can you replicate the problem, etc.

Click on link in the Data Set field for description and availability info.

Click on the column headings for description of each data access method.

Data Set	freq	grib filter	https	gds
Global Models				
GDAS	6 hours	grib filter	https	OpenDAP
GDAS 0.25	6 hours	grib filter	https	OpenDAP
GFS 0.25 Degree	6 hours	grib filter	https	OpenDAP
GFS 0.25 Degree Hourly	6 hours	grib filter	https	OpenDAP
GFS 0.25 Degree (SecondaryParms)	6 hours	grib filter	https	-
GFS 0.50 Degree	6 hours	grib filter	https	OpenDAP
GFS 1.00 Degree	6 hours	grib filter	https	OpenDAP
GFS sftrx	6 hours	grib filter	https	-
GFS Ensemble 0.5 Degree	6 hours	grib filter	https	OpenDAP
GFS Ensemble 0.5 Degree (SecondaryParms)	6 hours	grib filter	https	OpenDAP
GFS Ensemble 0.25 Degree	6 hours	grib filter	https	-
GFS Ensemble Chem 0.5 Degree	6 h			-
GFS Ensemble Chem 0.25 Degree	6 h			-
GFS Ensemble 0.5 Degree Bias-Corrected	6 h			enDAP
GFS Ensemble NDGD resolution Bias-Corrected	6 h			enDAP
NAEFS high resolution Bias-Corrected	6 h			enDAP
NAEFS NDGD resolution Bias-Corrected	6 h			enDAP

- GFS 0.25°
 - GFS 0.50°
 - GFS 1.00°
 - And others
 - Possible to select region
 - Possible to select levels
 - Possible to select variables

Accessing NWP Data (NOMADS)

Click on “grib filter” to filter parameters (region, levels, variables)

GFS

URL:
<https://nomads.ncep.noaa.gov/>

Available period:
 last 10 days

Size:
 ~ Varies by resolution and region (you can choose the desired region and variables)

Example Naming Convention:
 gfs.t00z.pgrb2.0p25.f000

Format:
 GRIB2

Resolution:
 0.25° (every hour or every 3 hours)
 0.50°
 1.00°

Runs:
 00, 06, 12, 18

Timesteps:
 1 h or 3 h / 16 days

Region:
 Global

Data Set	freq	grib filter	https	gds
Global Models				
GDAS	6 hours	grib filter	https	OpenDAP
GDAS 0.25	6 hours	grib filter	https	OpenDAP
GFS 0.25 Degree	6 hours	grib filter	https	OpenDAP
GFS 0.25 Degree Hourly	6 hours	grib filter	https	OpenDAP
GFS 0.25 Degree (Secondary Parms)	6 hours	grib filter	https	-
GFS 0.50 Degree	6 hours	grib filter	https	OpenDAP
GFS 1.00 Degree	6 hours	grib filter	https	OpenDAP
GFS sflux	6 hours	grib filter	https	-
GFS Ensemble 0.5 Degree	6 hours	grib filter	https	OpenDAP
GFS Ensemble 0.5 Degree (Secondary Parms)	6 hours	grib filter	https	OpenDAP
GFS Ensemble 0.25 Degree	6 hours	grib filter	https	-
GFS Ensemble Chem 0.5 Degree	6 hours	grib filter	https	-
GFS Ensemble Chem 0.25 Degree	6 hours	grib filter	https	-
GFS Ensemble 0.5 Degree Bias-Corrected	6 hours	grib filter	https	OpenDAP
GFS Ensemble NDGD resolution Bias-Corrected	6 hours	1-) Desired Data		
NAEFS high resolution Bias-Corrected	6 hours	grib filter	https	-
NAEFS NDGD resolution Bias-Corrected	6 hours	grib filter	https	OpenDAP

Data Transfer: NCEP GFS Forecasts (0.25 degree grid)

g2sub V1.1

g2subset (grib2 subset) allows you to subset (time, field, level, or region) a GRIB2 file and sends you the result

Directory: /gfs.20210519

Subdirectory

18
12
06
00

Select subdirectory from above list.

3-) Run

g2sub 1.1.0 beta-6 and comments: Wesley.Ebisuzaki@noaa.gov, Jun.Wang@noaa.gov

Data Transfer: NCEP GFS Forecasts (0.25 degree grid)

g2sub V1.1

g2subset (grib2 subset) allows you to subset (time, field, level, or region) a GRIB2 file and sends you the result

Subdirectory

gfs.20210520
gfs.20210519
gfs.20210518
gfs.20210517
gfs.20210516
gfs.20210515
gfs.20210514
gfs.20210513
gfs.20210512
gfs.20210511

Select subdirectory from above list.

2-) Date

g2sub 1.1.0 beta-6 and comments: Wesley.Ebisuzaki@noaa.gov, Jun.Wang@noaa.gov

Data Transfer: NCEP GFS Forecasts (0.25 degree grid)

g2sub V1.1

g2subset (grib2 subset) allows you to subset (time, field, level, or region) a GRIB2 file and sends you the result

Directory: /gfs.20210519/00

Subdirectory

atmos

Select subdirectory from above list.

4-) Subdirectory

g2sub 1.1.0 beta-6 and comments: Wesley.Ebisuzaki@noaa.gov, Jun.Wang@noaa.gov



Accessing NWP Data (NOMADS)

GFS

URL:
<https://nomads.ncep.noaa.gov/>

Available period:
 last 10 days

Size:
 ~ Varies by resolution and region (you can choose the desired region and variables)

Example Naming Convention:
 gfs.t00z.pgrb2.0p25.f000

Format:
 GRIB2

Resolution:
 0.25° (every hour or every 3 hours)
 0.50°
 1.00°

Runs:
 00, 06, 12, 18

Timesteps:
 1 h or 3 h / 16 days

Region:
 Global

Data Transfer: NCEP GFS Forecasts (0.25 degree grid)

5-) Configuration

g2subset (grib2 subset) allows you to subset (time, field, level, or region) a GRIB2 file and sends you the result
 Directory: /gfs.20210519_00_atmos
 NEW Select one file only (size in bytes)
 [gfs.t00z.pgrb2.0p25.anl (452192486) ▾]

FILE (Analysis, +0, +3, +6, etc.)

GRIB Filter

For GRIB data you have to option to filter the data.

Extract Levels and Variables

You may select some or all levels and variables. The selections below represent common choices which may or may not be relevant to the files that you have selected. For example choosing RH (relative humidity) would be pointless in file of sea-surface temperatures. In addition, not all possibilities are allowed. For example, suppose you only want the virtual temperature at the tropopause at 01Z. In this case you'd have to transfer the entire file.

For GRIB-2 data only.

Select the levels desired:

all 0.01 mb 0-0.1 m below ground 0.02 mb 0.04 mb 0.07 mb 0.1-0.4 m below ground 0.1 mb 0.2 mb 0.33-1 sigma layer 0.4-1 m below ground 0.44-0.72 sigma layer 0.44-1 sigma layer 0.4 mb 0.72-0.94 sigma layer 0.7 mb 0.995 sigma level 0C isotherm 1000 m above ground 1000 mb 100 m above ground 100 mb 10 m above ground 10 mb 150 mb 15 mb 180-0 mb above ground 1829 m above mean sea level 1 mb 200 mb 20 m above ground 20 mb 250 mb 255-0 mb above ground 2743 m above mean sea level 2 hybrid level 2 m above ground 2 mb 3000-0 m above ground 300 mb 30-0 mb above ground 30 m above ground 30 mb 350 mb 35-0 mb above ground 3 mb 4000 m above ground 400 mb 40 m above ground 40 mb 450 mb 500 mb 50 m above ground 50 mb 550 mb 5 mb 6000-0 m above ground 600 mb 650 mb 700 mb 70 mb 750 mb 7 mb 8000-0 m above ground 800 mb 850 mb 8 mb 900 mb 90-0 mb above ground 925 mb 950 mb 975 mb boundary layer cloud layer cloud ceiling convective cloud bottom level convective cloud layer convective cloud top level entire atmosphere entire atmosphere (extreme) single layer high cloud bottom level high cloud layer high cloud top level highest tropospheric freezing level low cloud bottom level low cloud layer low cloud top level max wind mean sea level middle cloud bottom level middle cloud layer middle cloud top level planetary boundary layer PV=-2e-06 (Km^2 kg/s) surface PV=2e-06 (Km^2 kg/s) surface surface top of atmosphere tropopause

Select the variables desired:

all 4LFTX ABSV ACPCP ALBDO APPC CCAPE CFRZR CICEP CIN CLWMR CNWAT CPOPF CPRAT CRAIN CSNOW CWAT CWORK DLWRF DPT DSWRF DZDT FLDCP FRICV GFLUX GRLE GUST HCDC HGT HINDEX HLCLY HPBL ICART ICEC ICEG ICETK ICETM ICJMR LAND LCDC LFTX LHFL MCDC MSLET O3MR OLR PBLT PRAT PRECIP PREPS PWAT REFC REFD RH RWMR SFCR SHFL SNAR SNOD SOILL SOTYP SPFH SUNSD TCDC TMAX TMN TMP TOZNE TSOIL UFLD UPRW USWRF VEG VFLX VGRD V-GWD VIS VRATE VSTM VVEL VWSH WATR WEASD WILT

LEVELS

VARIABLES

Extract Subregion

File transfer times can be reduced by only transferring a subregion. You can use this section to extract a geographic subsection from a most GRIB files. Use negative numbers for south and west.

make subregion left longitude [0] right longitude [360]
 top latitude [90] bottom latitude [-90]

REGION

View the URL

Show the URL only for web programming

Don't forget to pause before resubmitting requests.

URL VIEW (FOR PROGRAMMING)

Script 1: Knowing the Variables Available in a GRIB File

Let's view our scripts as "blocks" of code

```
1  #-
2  # INPE / CPTEC - Training: NWP Data Processing With Python - Script 1: Knowing the Available Variables
3  # Author: Diego Souza
4  #-
5  # Required modules
6  import pygrib          # Provides a high-level interface to the ECWMF ECCODES C library
7  #-
8
9  # Open the GRIB file
10 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
11
12 # Print all variables in the terminal and save them in a text file
13 f = open("variables.txt", "w") # Create and open the file
14 for variables in grib:
15     # Put the variables in the txt file
16     print(variables, file=f)
17     # Print the variables in the terminal
18     print(variables)
19 f.close() # Close the file
```

LIBRARIES

DATA READING AND
MANIPULATION

Script 1: Knowing the Variables Available in a GRIB File

Let's view our scripts as "blocks" of code

```
1  #-
2  # INPE / CPTEC - Training: NWP Data Processing With Python - Script 1: Knowing the Available Variables
3  # Author: Diego Souza
4  #
5  # Required modules Imports the PyGRIB
6  import pygrib           # Provides a high-level interface to the ECMWF ECCODES C library
7  #
8
9  # Open the GRIB file
10 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")          Open the GRIB file
11
12 # Print all variables in the terminal and save them in a text file
13 f = open("variables.txt", "w") # Create and open the file Create the text file if it doesn't exist
14 for variables in grib:      For each variable available in the GRIB file
15     # Put the variables in the txt file
16     print(variables, file=f) List the variable in the text file
17     # Print the variables in the terminal
18     print(variables)        Show the variable in the terminal
19 f.close() # Close the file Close the text file
```

LIBRARIES

DATA READING AND MANIPULATION



Script 1: Knowing the Variables Available in a GRIB File

TERMINAL

Anaconda Prompt (Miniconda3)

```
1013:Vertical speed shear:s***-1 (instant):regular_ll:potentialVorticity:level 5e-07 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1014:U component of wind:m s***-1 (instant):regular_ll:potentialVorticity:level -2.147483148 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1015:V component of wind:m s***-1 (instant):regular_ll:potentialVorticity:level -2.147483148 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1016:Temperature:K (instant):regular_ll:potentialVorticity:level -2.147483148 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1017:Geopotential Height:gpm (instant):regular_ll:potentialVorticity:level -2.147483148 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1018:Pressure:Pa (instant):regular_ll:potentialVorticity:level -2.147483148 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1019:Vertical speed shear:s***-1 (instant):regular_ll:potentialVorticity:level -2.147483148 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1020:U component of wind:m s***-1 (instant):regular_ll:potentialVorticity:level 1e-06 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1021:V component of wind:m s***-1 (instant):regular_ll:potentialVorticity:level 1e-06 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1022:Temperature:K (instant):regular_ll:potentialVorticity:level 1e-06 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1023:Geopotential Height:gpm (instant):regular_ll:potentialVorticity:level 1e-06 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1024:Pressure:Pa (instant):regular_ll:potentialVorticity:level 1e-06 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1025:Vertical speed shear:s***-1 (instant):regular_ll:potentialVorticity:level 1e-06 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1026:U component of wind:m s***-1 (instant):regular_ll:potentialVorticity:level -2.147482648 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1027:V component of wind:m s***-1 (instant):regular_ll:potentialVorticity:level -2.147482648 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1028:Temperature:K (instant):regular_ll:potentialVorticity:level -2.147482648 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1029:Geopotential Height:gpm (instant):regular_ll:potentialVorticity:level -2.147482648 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1030:Pressure:Pa (instant):regular_ll:potentialVorticity:level -2.147482648 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1031:Vertical speed shear:s***-1 (instant):regular_ll:potentialVorticity:level -2.147482648 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1032:U component of wind:m s***-1 (instant):regular_ll:potentialVorticity:level 1.5e-06 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1033:V component of wind:m s***-1 (instant):regular_ll:potentialVorticity:level 1.5e-06 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1034:Temperature:K (instant):regular_ll:potentialVorticity:level 1.5e-06 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1035:Geopotential Height:gpm (instant):regular_ll:potentialVorticity:level 1.5e-06 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1036:Pressure:Pa (instant):regular_ll:potentialVorticity:level 1.5e-06 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1037:Vertical speed shear:s***-1 (instant):regular_ll:potentialVorticity:level 1.5e-06 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1038:U component of wind:m s***-1 (instant):regular_ll:potentialVorticity:level -2.147482148 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1039:V component of wind:m s***-1 (instant):regular_ll:potentialVorticity:level -2.147482148 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1040:Temperature:K (instant):regular_ll:potentialVorticity:level -2.147482148 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1041:Geopotential Height:gpm (instant):regular_ll:potentialVorticity:level -2.147482148 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1042:Pressure:Pa (instant):regular_ll:potentialVorticity:level -2.147482148 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
1043:Vertical speed shear:s***-1 (instant):regular_ll:potentialVorticity:level -2.147482148 K m2 kg-1 s-1:fcst time 0 hrs:from 202107020000
(workshop) D:\VLAB\Python>
```



Script 1: Knowing the Variables Available in a GRIB File

TEXT FILE

variables - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
570:Soil Temperature:K (instant):regular_ll:depthBelowLandLayer:levels 0.4-1.0 m:fcst time 0 hrs:from 2021070
571:Volumetric soil moisture content:Proportion (instant):regular_ll:depthBelowLandLayer:levels 0.4-1.0 m:fcs
572:Liquid volumetric soil moisture (non-frozen):Proportion (instant):regular_ll:depthBelowLandLayer:levels 0
573:Soil Temperature:K (instant):regular_ll:depthBelowLandLayer:levels 1.0-2.0 m:fcst time 0 hrs:from 2021070
574:Volumetric soil moisture content:Proportion (instant):regular_ll:depthBelowLandLayer:levels 1.0-2.0 m:fcs
575:Liquid volumetric soil moisture (non-frozen):Proportion (instant):regular_ll:depthBelowLandLayer:levels 1
576:Plant canopy surface water:kg m**-2 (instant):regular_ll:surface:level 0:fcst time 0 hrs:from 20210702000
577:Water equivalent of accumulated snow depth (deprecated):kg m**-2 (instant):regular_ll:surface:level 0:fcs
578:Snow depth:m (instant):regular_ll:surface:level 0:fcst time 0 hrs:from 202107020000
579:Ice thickness:m (instant):regular_ll:surface:level 0:fcst time 0 hrs:from 202107020000
580:2 metre temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 0 hrs:from 202107020000
581:2 metre specific humidity:kg kg**-1 (instant):regular_ll:heightAboveGround:level 2 m:fcst time 0 hrs:from
582:2 metre dewpoint temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 0 hrs:from 2021
583:2 metre relative humidity:% (instant):regular_ll:heightAboveGround:level 2 m:fcst time 0 hrs:from 2021070
584:Apparent temperature:K (instant):regular_ll:heightAboveGround:level 2 m:fcst time 0 hrs:from 202107020000
585:10 metre U wind component:m s**-1 (instant):regular_ll:heightAboveGround:level 10 m:fcst time 0 hrs:from
586:10 metre V wind component:m s**-1 (instant):regular_ll:heightAboveGround:level 10 m:fcst time 0 hrs:from
587:Ice growth rate:m s**-1 (instant):regular_ll:heightAboveSea:level 10 m:fcst time 0 hrs:from 202107020000
588:Percent frozen precipitation:% (instant):regular_ll:surface:level 0:fcst time 0 hrs:from 202107020000
589:Precipitation rate:kg m**-2 s**-1 (instant):regular_ll:surface:level 0:fcst time 0 hrs:from 202107020000
590:Categorical snow:(Code table 4.222) (instant):regular_ll:surface:level 0:fcst time 0 hrs:from 202107020000
591:Categorical ice pellets:(Code table 4.222) (instant):regular_ll:surface:level 0:fcst time 0 hrs:from 2021
592:Categorical freezing rain:(Code table 4.222) (instant):regular_ll:surface:level 0:fcst time 0 hrs:from 20
593:Categorical rain:(Code table 4.222) (instant):regular_ll:surface:level 0:fcst time 0 hrs:from 202107020000
```

Some Quick Modifications

- Check variables from another model



Script 2: Basic Plot

With each script, our “blocks of code” will get more complex

```
1  #  
2  # INPE / CPTEC Training: NWP Data Processing With Python - Script 2: Basic Plot  
3  # Author: Diego Souza  
4  #  
5  import pygrib           # Provides a high-level interface to the ECMWF ECCODES C library for reading GRIB files  
6  import matplotlib.pyplot as plt # Plotting library  
7  #  
8  #  
9  # Open the GRIB file  
10 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")  
11  
12 # Select the variable  
13 grb = grib.select(name='2 metre temperature')[0]  
14  
15 # Select the extent [min. lon, min. lat, max. lon, max. lat]  
16 extent = [-93.0, -60.00, -25.00, 18.00]  
17  
18 # Read the data for the selected extent  
19 tmtmp, lats, lons = grb.data(latl=extent[1],lat2=extent[3],lonl=extent[0]+360,lon2=extent[2]+360)  
20  
21  
22 # Choose the plot size (width x height, in inches)  
23 plt.figure(figsize=(8,8))  
24  
25 # Plot the image  
26 plt.imshow(tmtmp, origin='lower', cmap='jet')  
27  
28 # Save the image  
29 plt.savefig('image_2.png')  
30  
31 # Show the image  
32 plt.show()
```

LIBRARIES

DATA READING AND
MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

Script 2: Basic Plot

With each script, our “blocks of code” will get more complex

```

1  #--#
2  # INPE / CPTEC Training: NWP Data Processing With Python - Script 2: Basic Plot
3  # Author: Diego Souza
4  #
5  import pygrib           # Provides a high-level interface to the ECMWF ECCODES C library for reading GRIB files
6  import matplotlib.pyplot as plt # Plotting library
7  #
8
9  # Open the GRIB file
10 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
11
12 # Select the variable
13 grb = grib.select(name='2 metre temperature')[0]  — Variable selection
14
15 # Select the extent [min. lon, min. lat, max. lon, max. lat]
16 extent = [-93.0, -60.00, -25.00, 18.00] — Desired extent
17
18 # Read the data for the selected extent
19 tmtmp, lats, lons = grb.data(latl=extent[1],lat2=extent[3],lonl=extent[0]+360,lon2=extent[2]+360)
20
21
22 # Choose the plot size (width x height, in inches)
23 plt.figure(figsize=(8,8)) — Plot Size
24
25 # Plot the image
26 plt.imshow(tmtmp, origin='lower', cmap='jet') — Color palette
27
28 # Save the image
29 plt.savefig('image_2.png') — Save the image
30
31 # Show the image
32 plt.show() — Show the plot

```

LIBRARIES

DATA READING AND MANIPULATION

Reading the variable,
latitudes and
longitudes

PLOT CONFIGURATION

IMAGE GENERATION

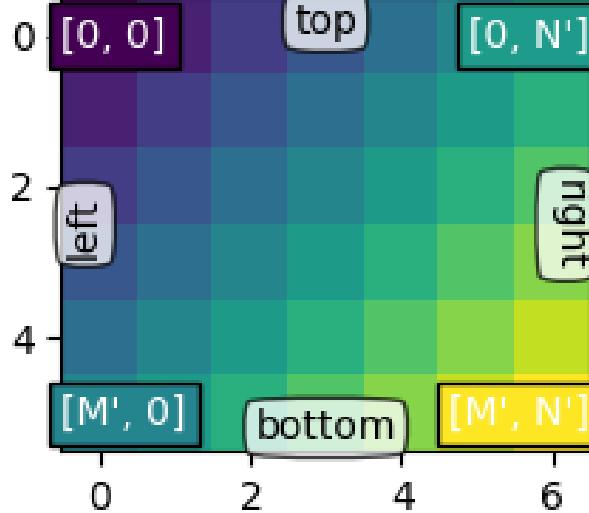
Script 2: Basic Plot

```
# Plot the image  
plt.imshow(tmtmp, origin='lower', cmap='jet')  
#-----
```

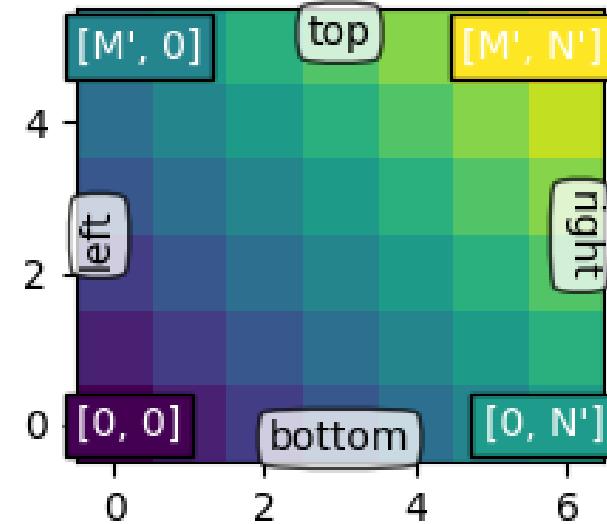
extent=

origin: upper

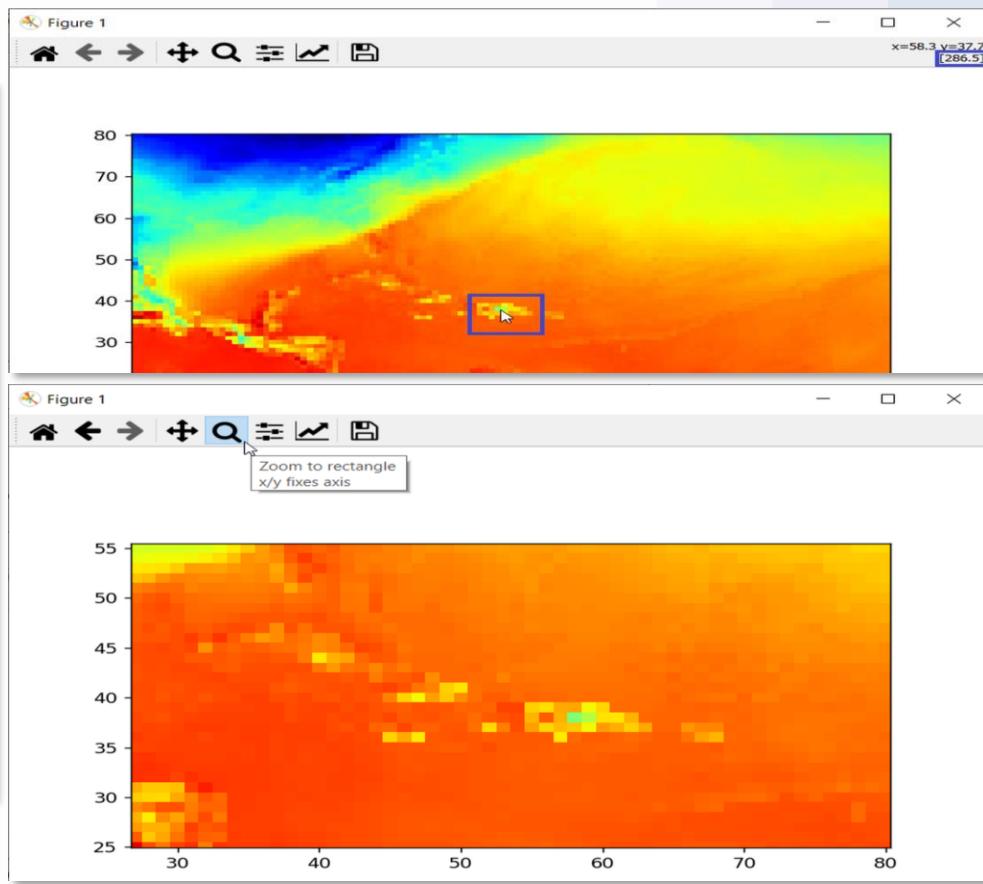
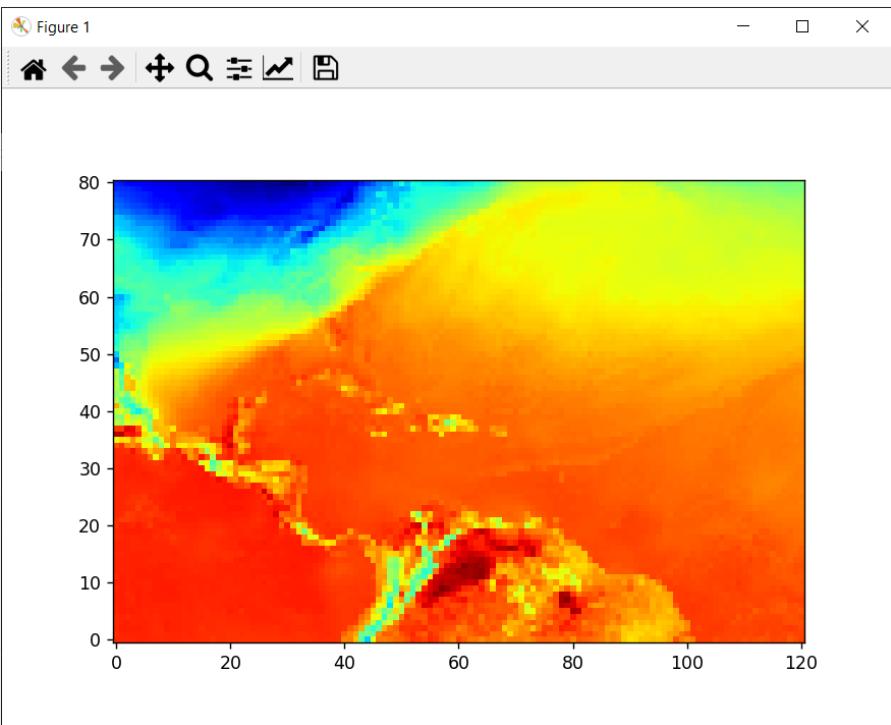
None



origin: lower



Script 2: Basic Plot



Some Quick Modifications

- Change the region
- Change the plot size
- Change the variable



Explaining the new instructions:

```

1  # INPE / CPTEC Training: NWP Data Processing
2  # Author: Diego Souza
3
4  # Import required libraries
5  import pygrib           # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files
6  import matplotlib.pyplot as plt # Plotting library
7
8
9  # Open the GRIB file
10 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
11
12 # Select the variable
13 grb = grib.select(name='2 metre temperature')[0]
14
15 # Get information from the file
16 init = str(grb.analDate)      # Init date / time
17 run = str(grb.hour).zfill(2)   # Run
18 ftime = str(grb.forecastTime)  # Forecast hour
19 valid = str(grb.validDate)    # Valid date / time
20 print('Init: ' + init + ' UTC')
21 print('Run: ' + run + 'Z')
22 print('Forecast: ' + ftime)
23 print('Valid: ' + valid + ' UTC')
24
25 # Select the extent [min. lon, min. lat, max. lon, max. lat]
26 extent = [-93.0, -60.00, -25.00, 18.00]
27
28 # Read the data for a specific region
29 tmtmp, lats, lons = grb.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)
30
31 # Convert from K to °C
32 tmtmp = tmtmp - 273.15
33
34
35 # Choose the plot size (width x height, in inches)
36 plt.figure(figsize=(8,8))
37
38 # Plot the image
39 plt.imshow(tmtmp, origin='lower', cmap='nipy_spectral')
40
41 # Add a colorbar
42 plt.colorbar(label='2 m Temperature (°C)', extend='both', orientation='horizontal', pad=0.05, fraction=0.05)
43
44 # Add a title
45 plt.title('GFS: 2 m Temperature' , fontweight='bold', fontsize=10, loc='left')
46 plt.title('Valid: ' + valid, fontsize=10, loc='right')
47
48
49 # Save the image
50 plt.savefig('image_3.png')
51
52 # Show the image
53 plt.show()

```

LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

Explaining the new instructions:

```

1  # INPE / CPTEC Training: NWP Data Processing
2  # Author: Diego Souza
3
4  # Import required libraries
5  import pygrib           # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files
6  import matplotlib.pyplot as plt # Plotting library
7
8
9  # Open the GRIB file
10 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
11
12 # Select the variable
13 grb = grib.select(name='2 metre temperature')[0]
14
15 # Get information from the file
16 init = str(grb.analDate)      # Init date / time
17 run  = str(grb.hour).zfill(2)  # Run
18 ftime = str(grb.forecastTime) # Forecast hour
19 valid = str(grb.validDate)    # Valid date / time
20 print('Init: ' + init + ' UTC')
21 print('Run: ' + run + 'Z')
22 print('Forecast: ' + ftime)
23 print('Valid: ' + valid + ' UTC')
24
25 # Select the extent [min. lon, min. lat, max. lon, max. lat]
26 extent = [-93.0, -60.00, -25.00, 18.00]
27
28 # Read the data for a specific region
29 tmtmp, lats, lons = grb.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)
30
31 #
32 # Convert from K to °C
33 tmtmp = tmtmp - 273.15
34
35
36 # Choose the plot size (width x height, in inches)
37 plt.figure(figsize=(8,8))
38
39 # Plot the image
40 plt.imshow(tmtmp, origin='lower', cmap='nipy_spectral')
41
42 # Add a colorbar
43 plt.colorbar(label='2 m Temperature (°C)', extend='both', orientation='horizontal', pad=0.05, fraction=0.05)
44
45 # Add a title
46 plt.title('GFS: 2 m Temperature', fontweight='bold', fontsize=10, loc='left')
47 plt.title('Valid: ' + valid, fontsize=10, loc='right')
48
49 # Save the image
50 plt.savefig('image_3.png')
51
52 # Show the image
53 plt.show()

```

LIBRARIES

DATA READING AND MANIPULATION

Reading the Metadata



Conversion from K to °C

Creation of Colorbar

PLOT CONFIGURATION

Size relative to the plot

shrink

size
multiplier

PLOT CONFIGURATION

Distance relative to the plot

Creation of the Title

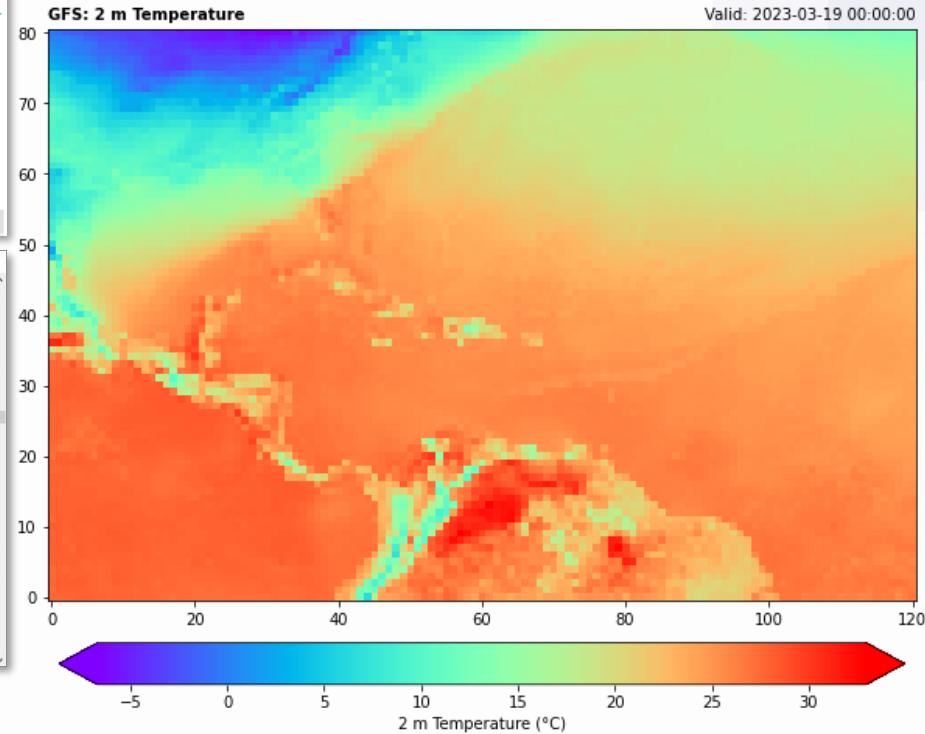
IMAGE GENERATION

Script 3: Metadata, Basic Operation, Color Palette, Legend and Title

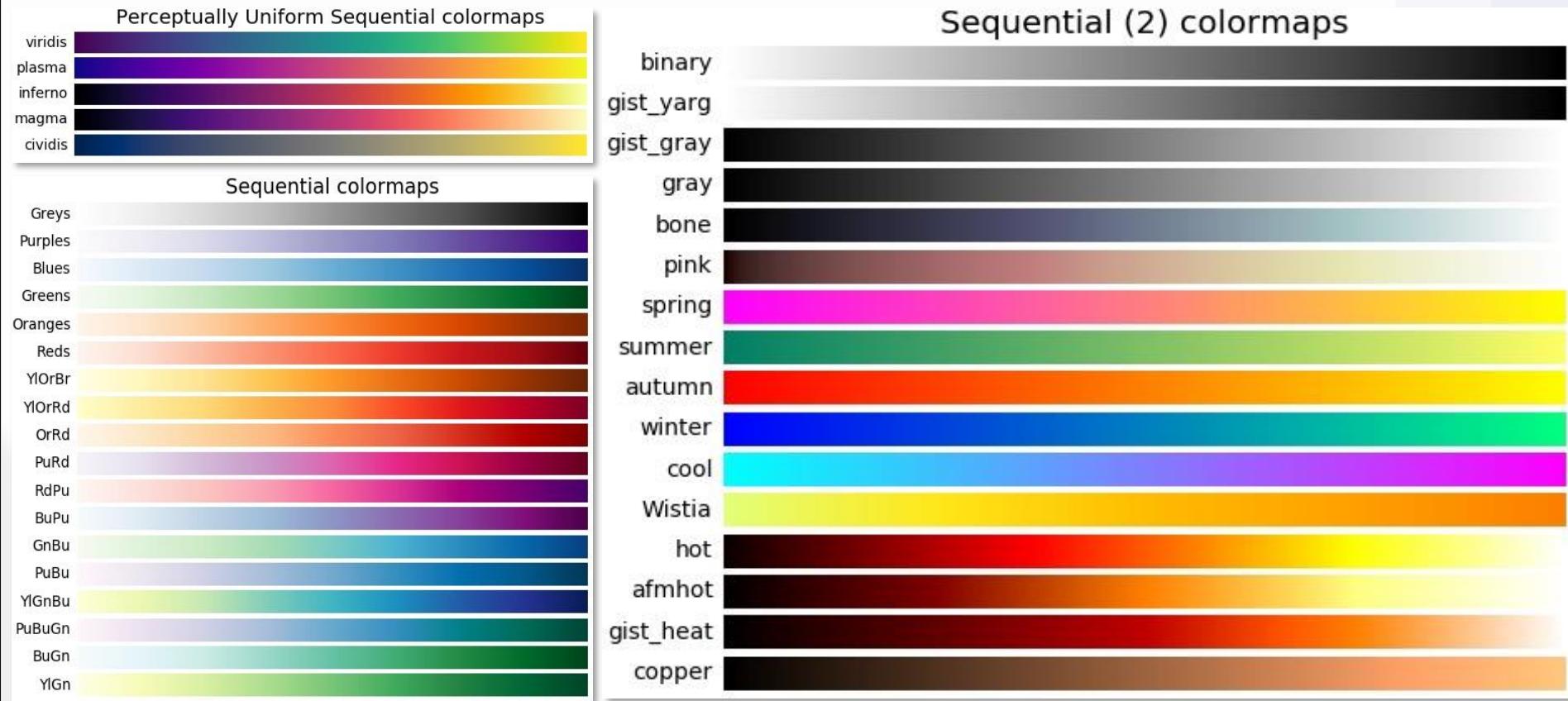
```
1 #---  
2 import pygrib                      # Provides a high-level int  
3 import matplotlib.pyplot as plt      # Plotting library  
4 #---  
5  
6 # Open the GRIB file  
7 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")  
8  
9 # Select the variable  
10 grb = grib.select(name='2 metre temperature')[0]  
11  
12 # Print the available keys  
13 print("GRIB Keys : ", grb.keys())
```

■ Appendix B script (Miniconda2) - python script_02.py

```
(workshop) D:\VLAB\Python>python script_03.py
Init: 2021-07-02 00:00:00 UTC
Run: 00Z
Forecast: +0
Valid: 2021-07-02 00:00:00 UTC
```

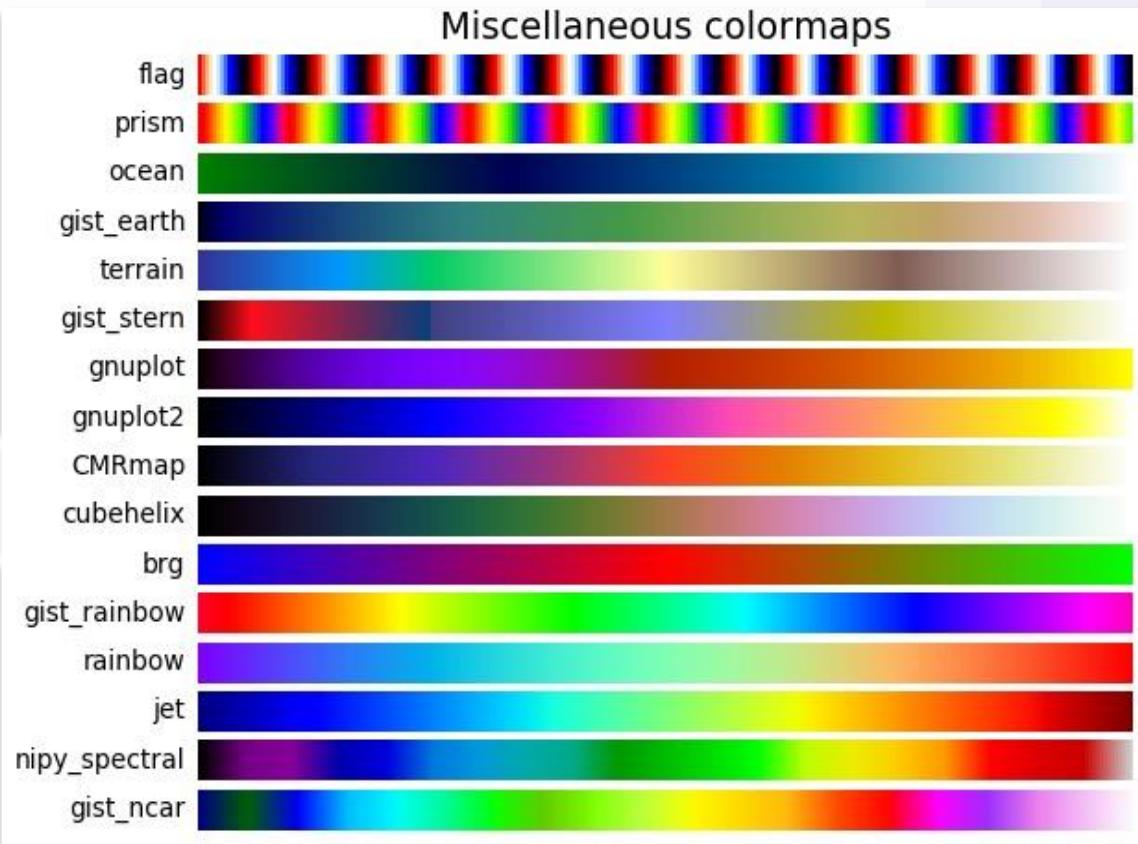
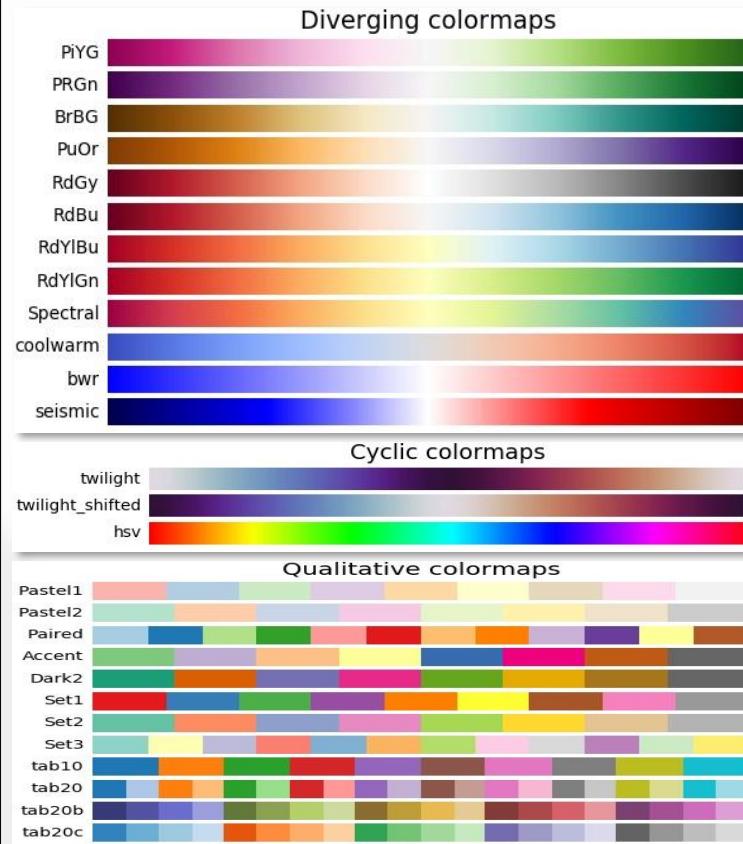


<https://matplotlib.org/stable/tutorials/colors/colormaps.html>



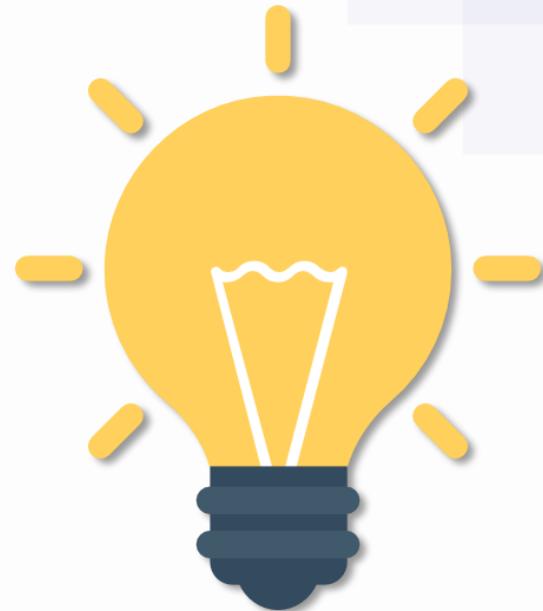
Script 3: Metadata, Basic Operation, Color Palette, Legend and Title

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



Some Quick Modifications

- Change the colormap
- Change the colorbar
- Change the title
- Read other metadata



Script 4: Overlaying Maps With Cartopy

```

1  #!/usr/bin/python
2  # INPE / CPTEC Training: NWP Data Processing With Python - Script 4: Adding a Map with Cartopy
3  # Author: Diego Souza
4
5  import pygrib           # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files
6  import matplotlib.pyplot as plt   # Plotting library
7  import cartopy, cartopy.crs as ccrs # Plot maps
8  import numpy as np    # Scientific computing with Python
9
10
11 # Open the GRIB file
12 grb = pygrib.open("gfs.t00z.pgrb2full.0p50.f600")
13
14 # Select the variable
15 gbd = grb.select(name='2 metre temperature')[0]
16
17 # Get information from the file
18 init = str(grb.analDate) # Init date / time
19 run = str(grb.hour).zfill(2) # Run
20 ftime = str(grb.forecastTime) # Forecast hour
21 valid = str(grb.validDate) # Valid date / time
22 print('Init: ' + init + ' UTC')
23 print('Run: ' + run + 'Z')
24 print('Forecast: +' + ftime)
25 print('Valid: ' + valid + ' UTC')
26
27 # Select the extent [min. lon, min. lat, max. lon, max. lat]
28 extent = [-93.0, -60.00, -25.00, 15.00]
29
30 # Read the data for a specific region
31 ttmp, lats, lons = grb.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)
32
33 #---#
34 # Convert from K to °C
35 ttmp = ttmp - 273.15
36
37
38 # Choose the plot size (width x height, in inches)
39 plt.figure(figsize=(8,8))
40
41 # Use the Cylindrical Equidistant projection in cartopy
42 ax = plt.axes(projection=ccrs.PlateCarree())
43
44 # Define the image extent [min. lon, max. lon, min. lat, max. lat]
45 img_extent = (extent[0], extent[2], extent[1], extent[3])
46
47 # Add coastlines, borders and gridlines
48 ax.coastlines(resolution='10m', color='black', linewidth=0.8)
49 ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
50 gl = ax.gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
51 gl.top_labels = False
52 gl.right_labels = False
53
54 # Plot the image
55 img = ax.imshow(ttmp, origin='lower', extent=img_extent, vmin=-20, vmax=40, cmap='jet')
56
57 # Add a colorbar
58 plt.colorbar(img, label='2 m Temperature (°C)', extend='both', orientation='vertical', pad=0.05, fraction=0.05)
59
60 # Add a title
61 plt.title('GFS: 2 m Temperature', fontweight='bold', fontsize=10, loc='left')
62 plt.title('Init: ' + init + ' UTC', fontweight='normal', fontsize=10, loc='left')
63
64 # Save the image
65 plt.savefig('image_4.png')
66
67 # Show the image
68 plt.show()

```

LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

Script 4: Overlaying Maps With Cartopy

Explaining the new instructions:

```
1  #+
2  # INPE / CPTEC Training: NWP Data Processing With Python - Script 4: Adding a Map with Cartopy
3  # Author: Diego Souza
4
5  import pygrib           # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files
6  import matplotlib.pyplot as plt    # Plotting library
7  import cartopy, cartopy.crs as ccrs  # Plot maps
8  import numpy as np   # Scientific computing with Python
9
10
11 # Open the GRIB file
12 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
13
14 # Select the variable
15 grb = grib.select(name='2 metre temperature')[0]
16
17 # Get information from the file
18 init  = str(grb.analDate)      # Init date / time
19 run   = str(grb.hour).zfill(2) # Run
20 ftime = str(grb.forecastTime)  # Forecast hour
21 valid = str(grb.validDate)    # Valid date / time
22 print('Init: ' + init + ' UTC')
23 print('Run: ' + run + 'Z')
24 print('Forecast: ' + ftime)
25 print('Valid: ' + valid + ' UTC')
26
27 # Select the extent [min. lon, min. lat, max. lon, max. lat]
28 extent = [-93.0, -60.00, -25.00, 18.00]
29
30 # Read the data for a specific region
31 tmtmp, lats, lons = grb.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)
32
33 #-----
34 # Convert from K to °C
35 tmtmp = tmtmp - 273.15
36
37 #
```

LIBRARIES

DATA READING AND
MANIPULATION

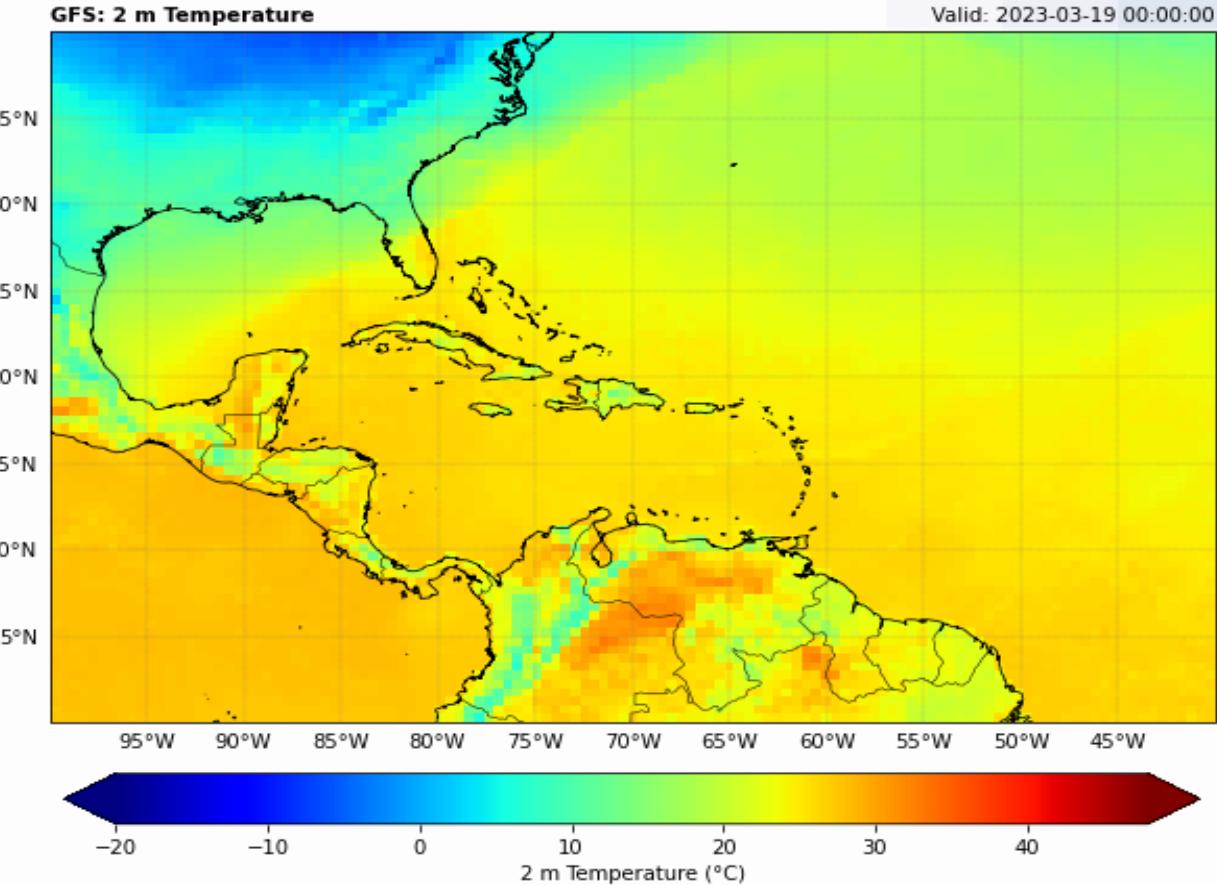
Script 4: Overlaying Maps With Cartopy

```
27 # Select the extent [min. lon, min. lat, max. lon, max. lat]
28 extent = [-93.0, -60.00, -25.00, 18.00]
29
30 # Read the data for a specific region
31 tmtmp, lats, lons = grb.data(lat=extent[1],lat2=extent[3],lonl=extent[0]+360,lon2=extent[2]+360)
32
33 -----
34 # Convert from K to °C
35 tmtmp = tmtmp - 273.15
36
37 -----
38 # Choose the plot size (width x height, in inches)
39 plt.figure(figsize=(8,8))
40
41 # Use the Cylindrical Equidistant projection in cartopy
42 ax = plt.axes(projection=ccrs.PlateCarree()) └── Cartopy projection
43
44 # Define the image extent [min. lon, max. lon, min. lat, max. lat]
45 img_extent = [extent[0], extent[2], extent[1], extent[3]] └── Region
46
47 # Add coastlines, borders and gridlines
48 ax.coastlines(resolution='10m', color='black', linewidth=0.8) └── Coastlines
49 ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5) └── Countries
50 gl = ax.gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
51 gl.top_labels = False └── Latitude and Longitude gridlines
52 gl.right_labels = False └── Disable upper and lower labels
53
54 # Plot the image
55 img = ax.imshow(tmtmp, origin='lower', extent=img_extent, vmin=-20, vmax=48, cmap='jet')
56
57 # Add a colorbar
58 plt.colorbar(img, label='2 m Temperature (°C)', extend='both', orientation='vertical', pad=0.05, fraction=0.05)
59
60 # Add a title
61 plt.title('GFS: 2 m Temperature', fontweight='bold', fontsize=10, loc='left')
62 plt.title('Valid: ' + valid, fontsize=10, loc='right')
63
64 # Save the image
65 plt.savefig('image_4.png')
66
67 # Show the image
68 plt.show()
```

PLOT CONFIGURATION

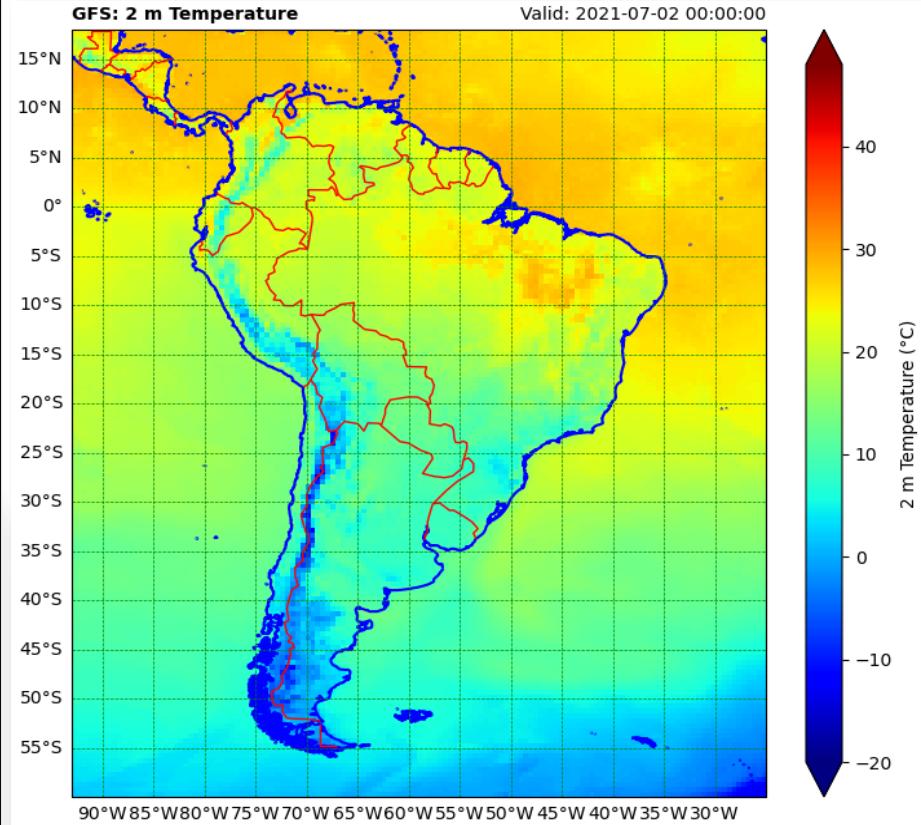
Lon. interval	Lat. interval
█	█

Script 4: Overlaying Maps With Cartopy



Script 4: Overlaying Maps With Cartopy

https://matplotlib.org/stable/gallery/color/named_colors.html



black	bisque	forestgreen	slategrey
dimgray	darkorange	limegreen	lightsteelblue
dimgrey	burlywood	darkgreen	cornflowerblue
gray	antiquewhite	green	royalblue
grey	tan	lime	ghostwhite
darkgray	navajowhite	seagreen	lavender
darkgrey	blanchedalmond	mediumseagreen	midnightblue
silver	papayawhip	springgreen	navy
lightgray	moccasin	mintcream	darkblue
lightgrey	orange	mediumspringgreen	mediumblue
gainsboro	wheat	mediumaquamarine	blue
whitesmoke	oldlace	aquamarine	slateblue
white	floralwhite	turquoise	darkslateblue
snow	darkgoldenrod	lightseagreen	mediumslateblue
rosybrown	goldenrod	mediumturquoise	mediumpurple
lightcoral	cornsilk	azure	rebeccapurple
indianred	gold	lightcyan	indigo
brown	lemonchiffon	paleturquoise	darkorchid
firebrick	khaki	darkslategray	darkviolet
maroon	palegoldenrod	darkslategrey	mediumorchid
darkred	darkkhaki	teal	thistle
red	ivory	darkcyan	plum
mistyrose	beige	aqua	violet
salmon	lightyellow	cyan	purple
tomato	lightgoldenrodyellow	darkturquoise	darkmagenta
darksalmon	olive	cadetblue	fuchsia
coral	yellow	powderblue	magenta
orangered	olivedrab	lightblue	orchid
lightsalmon	yellowgreen	deepskyblue	mediumvioletred
sienna	darkolivegreen	skyblue	deepink
seashell	greenyellow	lightskyblue	hotpink
chocolate	chartreuse	steelblue	lavenderblush
saddlebrown	lawngreen	aliceblue	palevioletred
sandybrown	honeydew	dodgerblue	crimson
peachpuff	darkseagreen	lightslategray	pink
peru	palegreen	lightslategrey	slategray
linen	lightgreen		

- Change the map colors
- Change the line thicknesses
- Show specific ranges

```
tmtmp[tmtmp < 23] = np.nan  
tmtmp[tmtmp > 23] = np.nan  
tmtmp[np.logical_or(tmtmp < 5, tmtmp > 15)] = np.nan
```



Script 5: Reading a Shapefile

```

1  # INPE / CPTEC Training: NWP Data Processing With Python - Script 5: Adding a Shapefile
2  # Author: Diego Souza
3
4
5  import pygrib           # Provides a high-level interface to the ECWMP ECCODES C library for reading GRIB files
6  import matplotlib.pyplot as plt # Plotting library
7  import cartopy.crs as ccrs # Plot shapefiles
8  import cartopy.io.shapereader as shapereader # Import shapefiles
9  import numpy as np # Scientific computing with Python
10
11
12  # Open the GRIB file
13  grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
14
15  # Select the variable
16  grib = grib.select(name='2 metre temperature')[0]
17
18  # Get information from the file
19  init = str(grib.analDate) # Init date / time
20  run = str(grib.hour).zfill(2) # Run
21  ftime = str(grib.forecastTime) # Forecast hour
22  valid = str(grib.validDate) # Valid date / time
23
24  print("Init: " + init + " UTC")
25  print("Run: " + run + "Z")
26  print("Forecast: " + ftime)
27  print("Valid: " + valid + " UTC")
28
29  # Select the extent [min. lon, min. lat, max. lon, max. lat]
30  extent = [-93.0, -60.00, -25.00, 18.00]
31
32  # Read the data for a specific region
33  ttmp, lats, lons = grib.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)
34
35  # Convert from K to °C
36  ttmp = ttmp - 273.15
37
38
39
40  plt.figure(figsize=(8,8))
41
42  # Use the Cylindrical Equidistant projection in cartopy
43  ax = plt.axes(projection=ccrs.PlateCarree())
44
45  # Define the image extent
46  img_extent = [extent[0], extent[2], extent[1], extent[3]]
47
48  # Add a shapefile
49  shapefile = list(shapereader.Reader('ne_10m_admin_1_states_provinces.shp').geometries())
50  ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='red', facecolor='none', linewidth=0.3)
51
52  # Add coastlines, borders and gridlines
53  ax.coastlines(resolution='110m', color='black', linewidth=0.8)
54  ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
55  gl = ax.gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
56  gl.top_labels = False
57  gl.right_labels = False
58
59  # Plot the image
60  img = ax.imshow(ttmp, origin='lower', extent=img_extent, vmin=-20, vmax=40, cmap='jet')
61
62  # Add a colorbar
63  plt.colorbar(img, label='2 m Temperature (°C)', extend='both', orientation='vertical', pad=0.05, fraction=0.05)
64
65  # Add a title
66  plt.title('GFS: 2 m Temperature', fontweight='bold', fontsize=16, loc='left')
67  plt.title('Valid: ' + valid, fontsize=16, loc='right')
68
69
70  plt.savefig('image_5.png')
71
72  # Show the image
73  plt.show()

```

LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

Script 5: Reading a Shapefile

Explaining the new instructions:

```
1 #  
2 # INPE / CPTEC Training: NWP Data Processing With Python - Script 5: Adding a Shapefile  
3 # Author: Diego Souza  
4 #  
5 import pygrib # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files  
6 import matplotlib.pyplot as plt # Plotting library  
7 import cartopy, cartopy.crs as ccrs # Plot maps  
8 import cartopy.io.shapereader as shapereader # Import shapefiles  
9 import numpy as np # Scientific computing with Python  
10 #  
11  
12 # Open the GRIB file  
13 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")  
14  
15 # Select the variable  
16 grb = grib.select(name='2 metre temperature')[0]  
17  
18 # Get information from the file  
19 init = str(grb.analDate) # Init date / time  
20 run = str(grb.hour).zfill(2) # Run  
21 ftime = str(grb.forecastTime) # Forecast hour  
22 valid = str(grb.validDate) # Valid date / time  
23 print('Init: ' + init + ' UTC')  
24 print('Run: ' + run + 'Z')  
25 print('Forecast: +' + ftime)  
26 print('Valid: ' + valid + ' UTC')  
27  
28 # Select the extent [min. lon, min. lat, max. lon, max. lat]  
29 extent = [-93.0, -60.00, -25.00, 18.00]  
30  
31 # Read the data for a specific region  
32 tmtmp, lats, lons = grb.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)  
33  
34 #-----  
35 # Convert from K to °C  
36 tmtmp = tmtmp - 273.15  
37  
38 #-----
```

Imports the Cartopy "shapereader" module

LIBRARIES

DATA READING AND MANIPULATION

Script 5: Reading a Shapefile

```
34 #-----  
35 # Convert from K to °C  
36 tmtmp = tmtmp - 273.15  
37  
38 # Choose the plot size (width x height, in inches)  
39 plt.figure(figsize=(8,8))  
40  
41 # Use the Cylindrical Equidistant projection in cartopy  
42 ax = plt.axes(projection=ccrs.PlateCarree())  
43  
44 # Define the image extent  
45 img_extent = [extent[0], extent[2], extent[1], extent[3]]  
46  
47 # Add a shapefile  
48 # https://geoftp.ibge.gov.br/organizacao\_do\_territorio/malhas\_territoriais/malhas\_municipais/municipio\_2019/Brasil/BR\_unidades\_da\_federacao.zip  
49 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries()) ────────── Read the Shapefile  
50 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='red', facecolor='none', linewidth=0.3) ────────── Configure the Shapefile  
51  
52 # Add coastlines, borders and gridlines  
53 ax.coastlines(resolution='10m', color='black', linewidth=0.8)  
54 ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)  
55 gl = ax.gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)  
56 gl.top_labels = False  
57 gl.right_labels = False  
58  
59 # Plot the image  
60 img = ax.imshow(tmtmp, origin='lower', extent=img_extent, vmin=-20, vmax=48, cmap='jet')  
61  
62 # Add a colorbar  
63 plt.colorbar(img, label='2 m Temperature (°C)', extend='both', orientation='vertical', pad=0.05, fraction=0.05)  
64  
65 # Add a title  
66 plt.title('GFS: 2 m Temperature', fontweight='bold', fontsize=10, loc='left')  
67 plt.title('Valid: ' + valid, fontsize=10, loc='right')  
68  
69 # Save the image  
70
```

PLOT CONFIGURATION

Script 5: Reading a Shapefile

(C:) > VLAB > Python > NWP

Name

gfs.t00z.pgrb2full.0p50.f000

image_2.png

image_3.png

image_4.png

ne_10m_admin_1_states_provinces.dbf

ne_10m_admin_1_states_provinces.shp

ne_10m_admin_1_states_provinces.shx

script_01.py

script_02.py

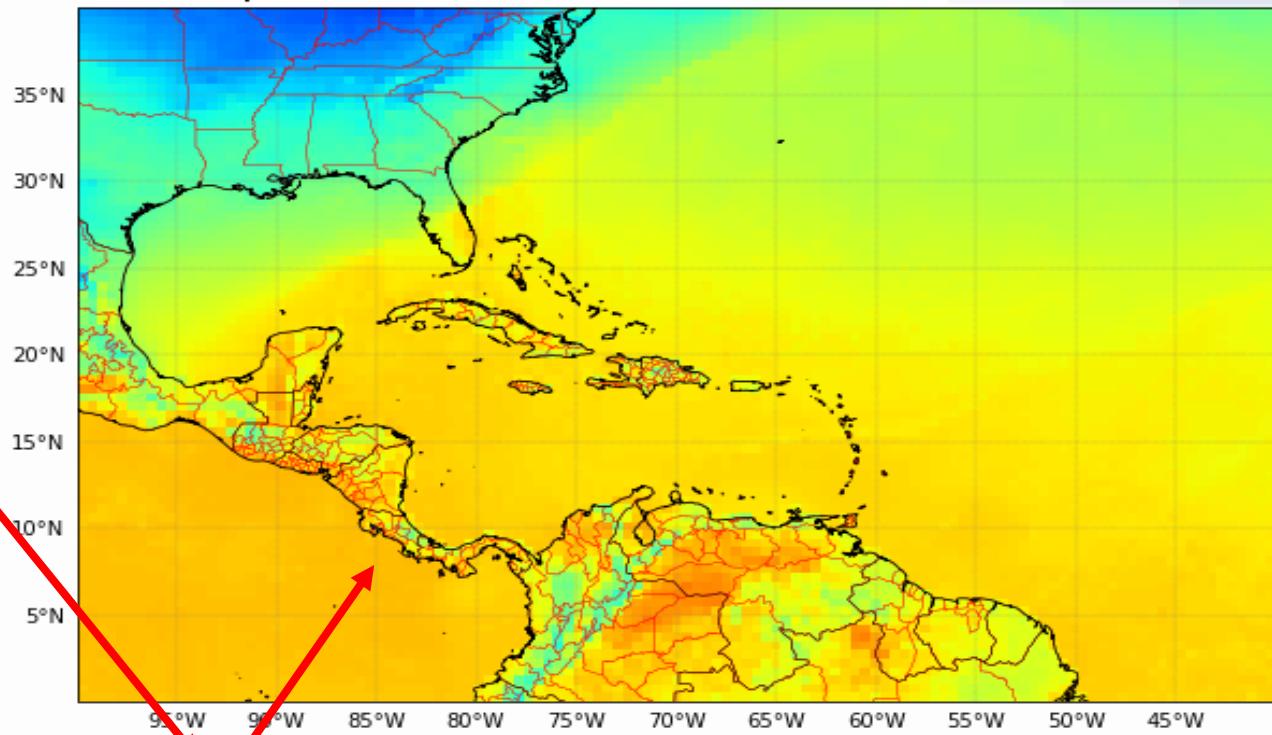
script_03.py

script_04.py

variables.txt

GFS: 2 m Temperature

Valid: 2023-03-19 00:00:00



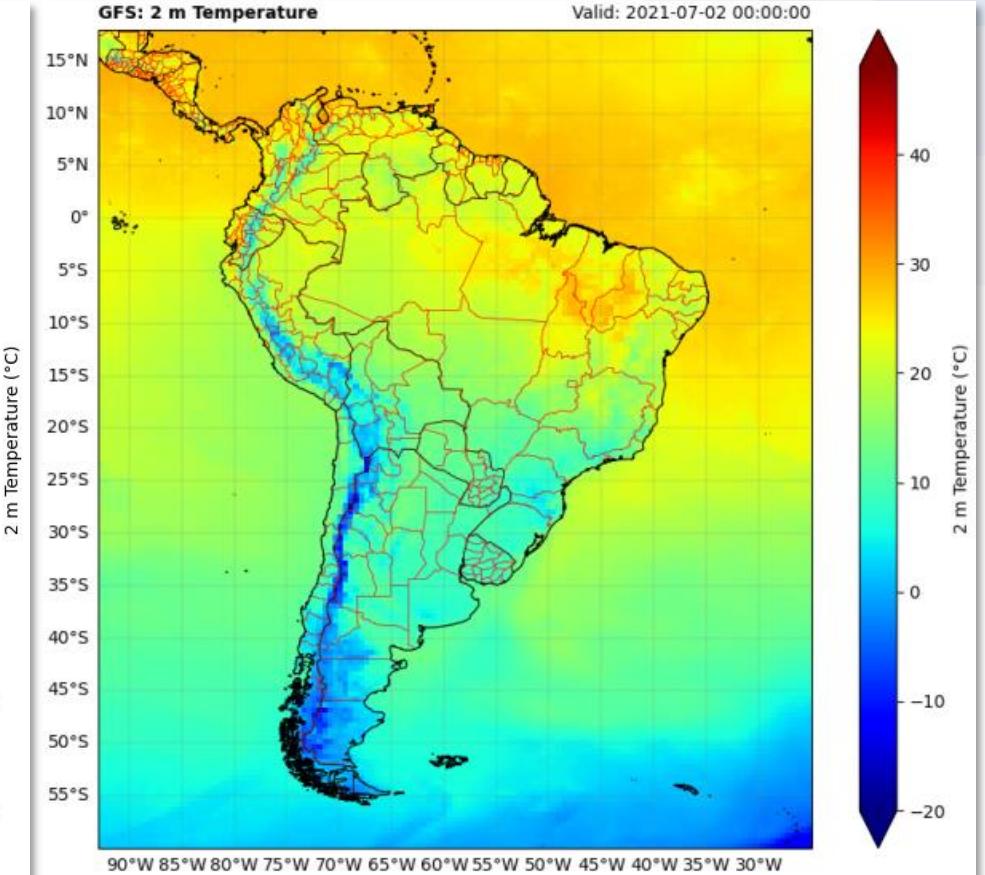
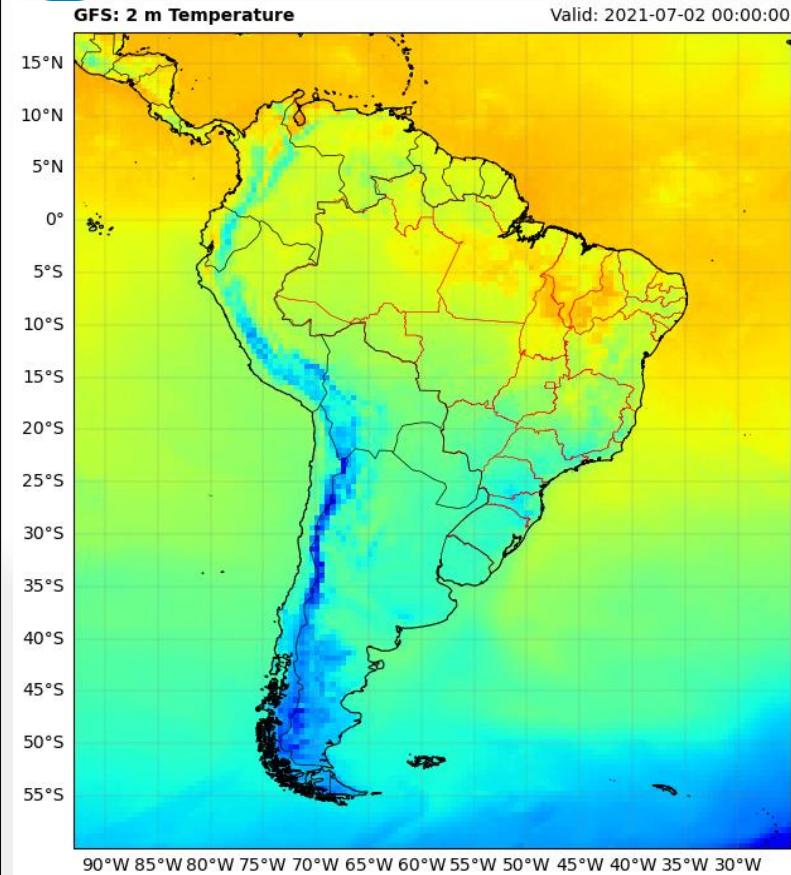
```
48 # Add a shapefile
49 shapefile = list(shpreader.Reader('ne_10m_admin_1_states_provinces.shp')).geometries()
50 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='red', facecolor='none', linewidth=0.3)
```

Some Quick Modifications

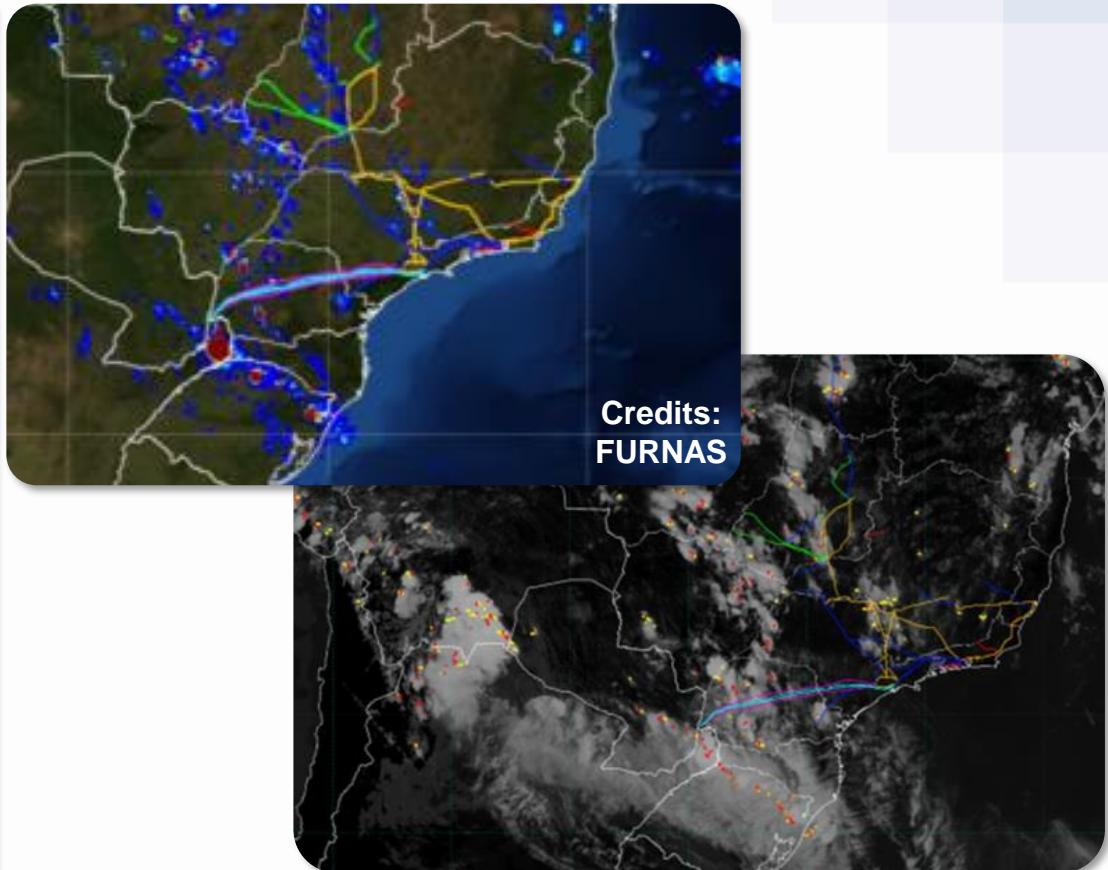
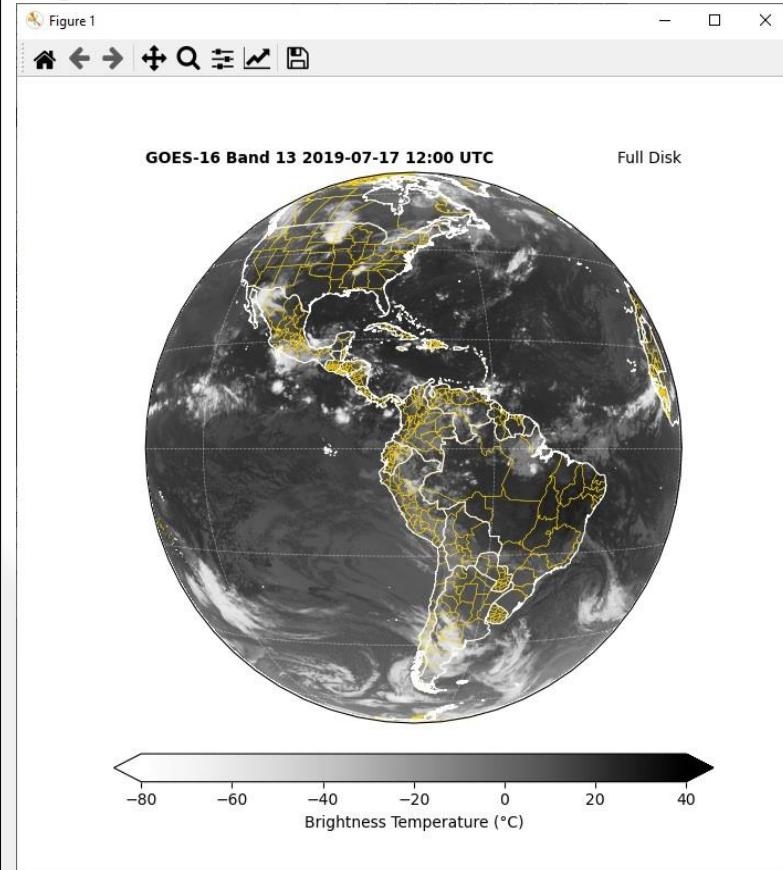
- Change the appearance of the shapefile
- Change the shapefile to be read



Script 5: Reading a Shapefile



Script 5: Reading a Shapefile



Script 6: Plotting Contours and Labels

```

1  INPE / CPMEC Training: NWP Data Processing with Python - Script 6: Plotting Contours and Labels
2  # Anthony Diaz Mora
3
4  import pygrb           # Provides a high level interface to the ROMS ROMSNET C library for reading GRIB files
5  import matplotlib.pyplot as plt   # Plotting library
6  import cartopy, cartopy.crs as ccrs # Plot maps
7  import cartopy.io.shapereader as shapereader # Import shapefiles
8  import numpy as np    # Scientific computing with Python
9
10
11
12  grib = pygrb.open('grib_ec00x.grib2full.0p50.0000')
13
14  # Select the variable
15  grib = grib.select(name='2 metre temperature')[0]
16
17  # Get information from the file
18  init = str(grib.analDate)      # Init date / time
19  run = str(grib.hour).zfill(2)  # Run
20  time = str(grib.forecastTime)  # Forecast hour
21  valid = str(grib.validDate)    # Valid date / time
22  print('Init: ' + init + ' UTC')
23  print('Run: ' + run + ' UTC')
24  print('Forecast: ' + time)
25  print('Valid: ' + valid + ' UTC')
26
27
28  # Select the extent [min, lon, min, lat, max, lon, max, lat]
29  extent = [-93.0, -60.0, -25.00, 18.00]
30
31
32  # Read the data for a specific region
33  temp, lats, lons = grib.select(lat1=extent[0], lat2=extent[1], lon1=extent[0]+360, lon2=extent[2]+360)
34
35
36  # Convert from K to °C
37  temp = temp - 273.15
38
39
40
41  plt.figure(figsize=(8,8))
42
43  # Use the Cylindrical Equal-Area 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://geosft.ibge.gov.br/geotools/do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/neuini/00_ibge_municipios.zip
51  shapefile = list(shapereader.Reader('00_ibge_2019.shp').geometries())
52  ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='red', facecolor='none', linewidth=0.5)
53
54
55  # Add coastlines, borders and gridlines
56  ax.coastlines(resolution='10m', color='black', linewidth=0.8)
57  ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
58  gl = ax.gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyles='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
59  gl.top_labels = False
60  gl.right_labels = False
61
62  # Define de contour interval
63  data_min = -20
64  data_max = 40
65  interval = 2
66
67  levels = np.arange(data_min,data_max,interval)
68
69  # Plot the contours
70  img1 = ax.contourf(lons, lats, temp, cmap='jet', levels=levels, extend='both')
71  img2 = ax.contour(lons, lats, temp, colors='white', linewidths=0.5, levels=levels)
72  ax.clabel(img1, inline=True, inline_spacing=0, fontsize=10, fmt = '%1.0f', colors='black')
73
74  # Add a colorbar
75  plt.colorbar(img1, label='2 m Temperature (°C)', orientation='vertical', pad=0.05, fraction=0.05)
76
77  # Save the image
78  plt.savefig('image_6.png')
79
80  # Show the image
81  plt.show()

```

LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

Script 6: Plotting Contours and Labels

Explaining the new instructions:

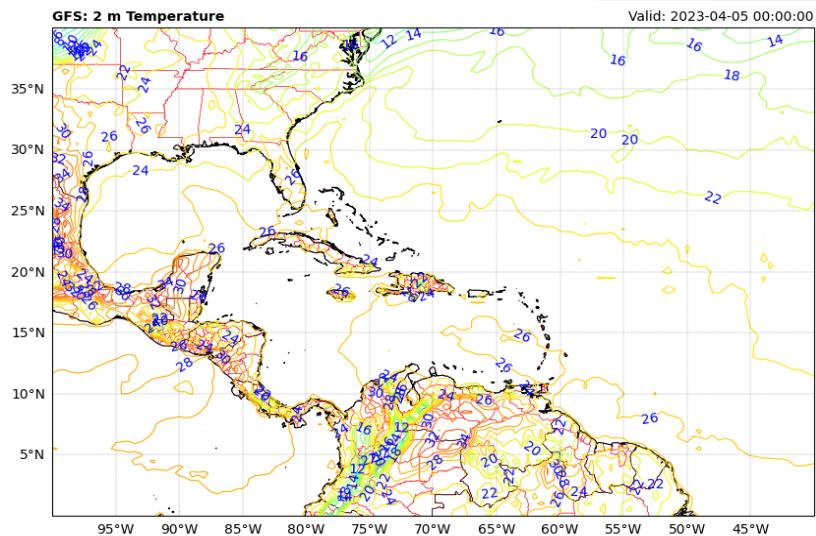
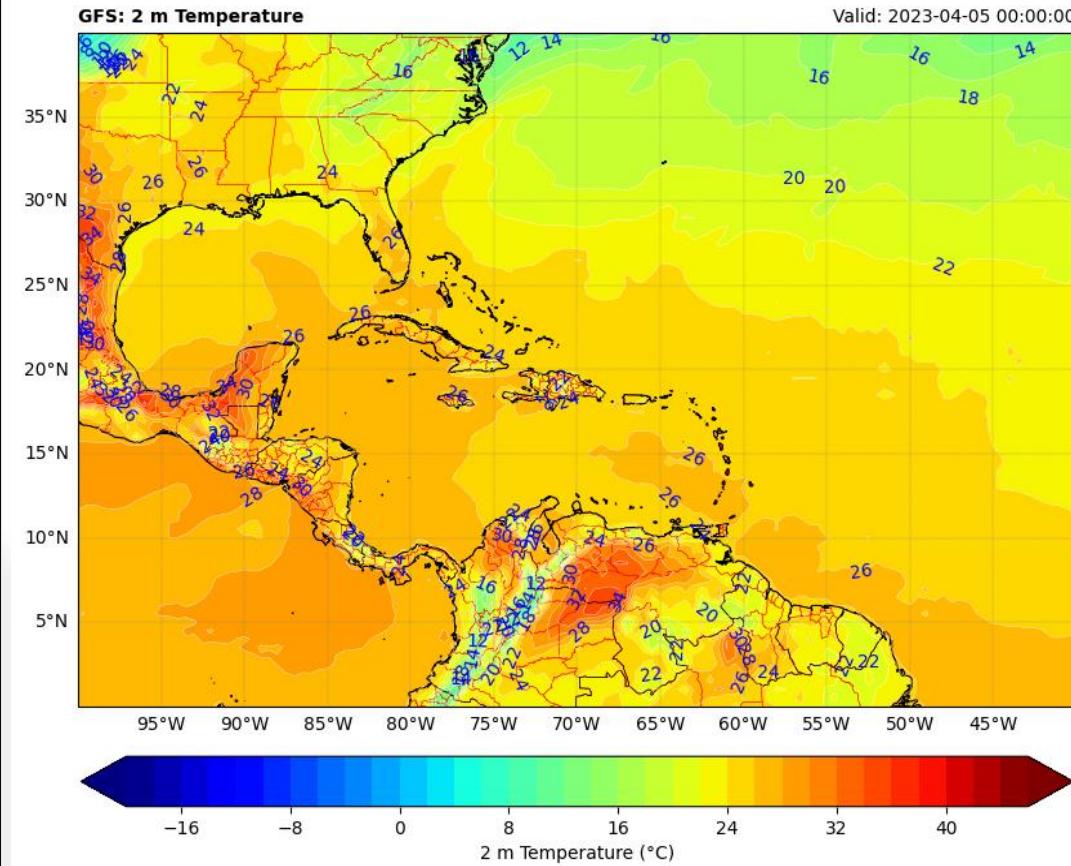
CONFIGURAÇÃO DO PLOT

```

34
35 # Convert from K to °C
36 tmtmp = tmtmp - 273.15
37
38
39 # Choose the plot size (width x height, in inches)
40 plt.figure(figsize=(8,8))
41
42 # Use the Cylindrical Equidistant projection in cartopy
43 ax = plt.axes(projection=ccrs.PlateCarree())
44
45 # Define the image extent
46 img_extent = [extent[0], extent[2], extent[1], extent[3]]
47
48 # Add a shapefile
49 # https://geotftp.ibge.gov.br/organizacao\_do\_territorio/malhas\_territoriais/malhas\_municipais/municipio\_2019/Brasil/BR/br\_unidades\_da\_federacao.zip
50 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
51 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='red', facecolor='none', linewidth=0.3)
52
53 # Add coastlines, borders and gridlines
54 ax.coastlines(resolution='10m', color='black', linewidth=0.8)
55 ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
56 gl = ax.gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
57 gl.top_labels = False
58 gl.right_labels = False
59
60 # Define de contours
61 data_min = -20
62 data_max = 48
63 interval = 2
64 levels = np.arange(data_min,data_max,interval) Creation of the array
65
66 # Plot the contours
67 img1 = ax.contourf(lons, lats, tmtmp, cmap='jet', levels=levels, extend='both') Filled contour
68 img2 = ax.contour(lons, lats, tmtmp, colors='white', linewidths=0.3, levels=levels) Unfilled contour
69 ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10', fmt = '%1.0f', colors= 'black')
70
71 # Add a colorbar
72 plt.colorbar(img1, label='2 m Temperature (°C)', orientation='vertical', pad=0.05, fraction=0.05) Label Configuration
73
74 # Add a title
75 plt.title('GFS: 2 m Temperature' , fontweight='bold', fontsize=10, loc='left')
76 plt.title('Valid: ' + valid, fontsize=10, loc='right')
77
78 # Save the image
79 plt.savefig('image_6.png')

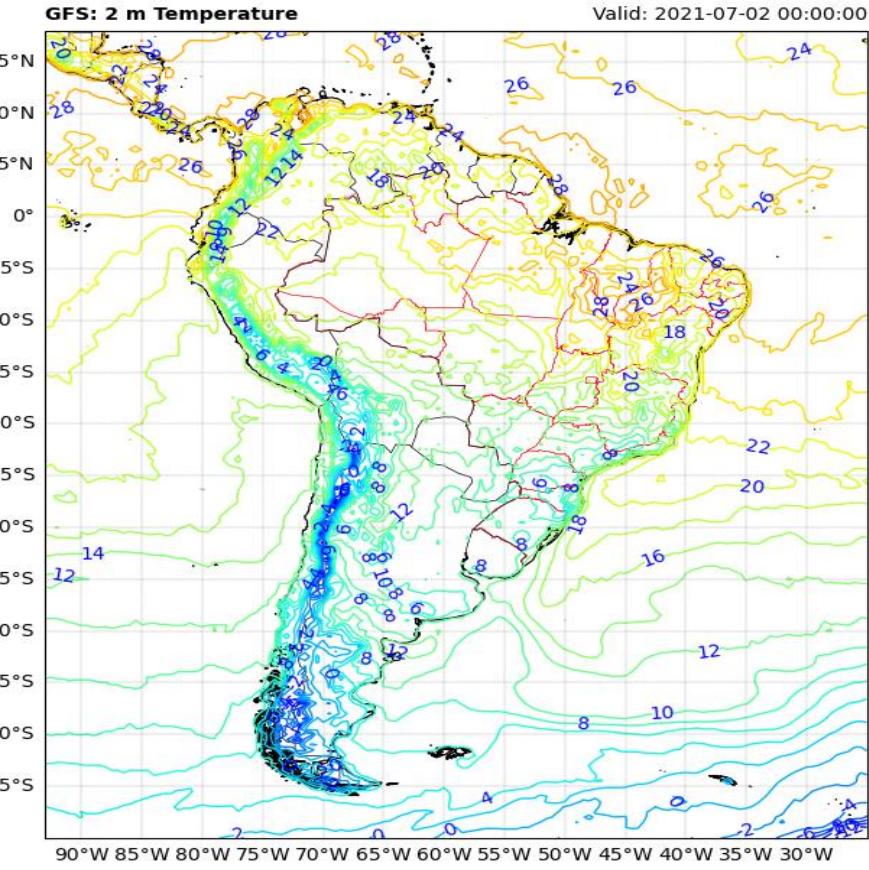
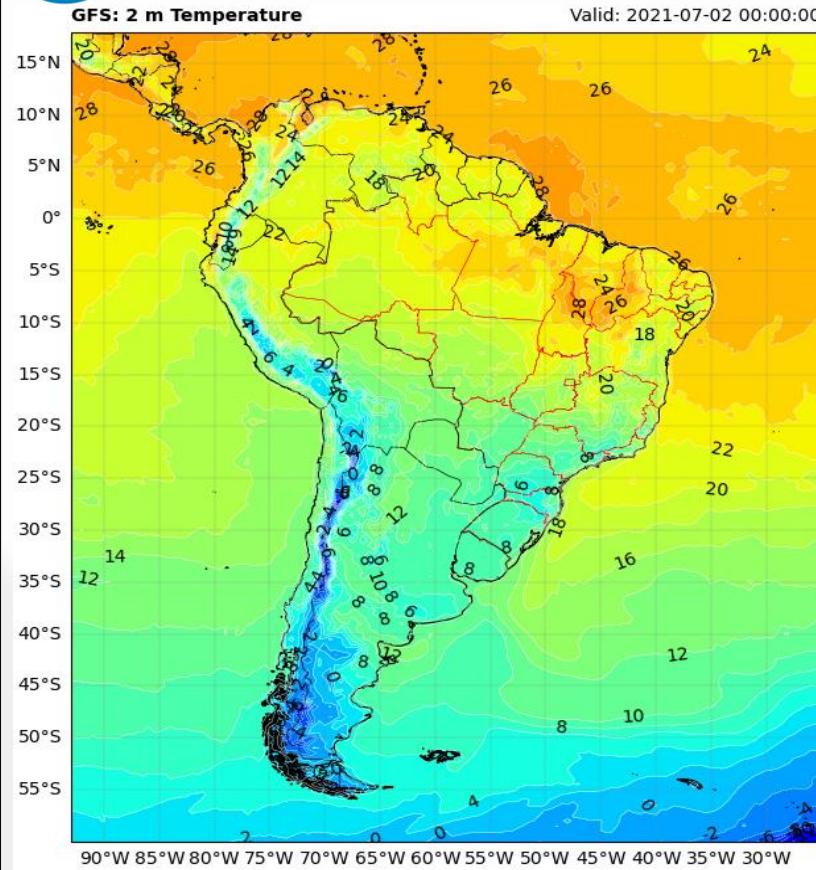
```

Script 6: Plotting Contours and Labels



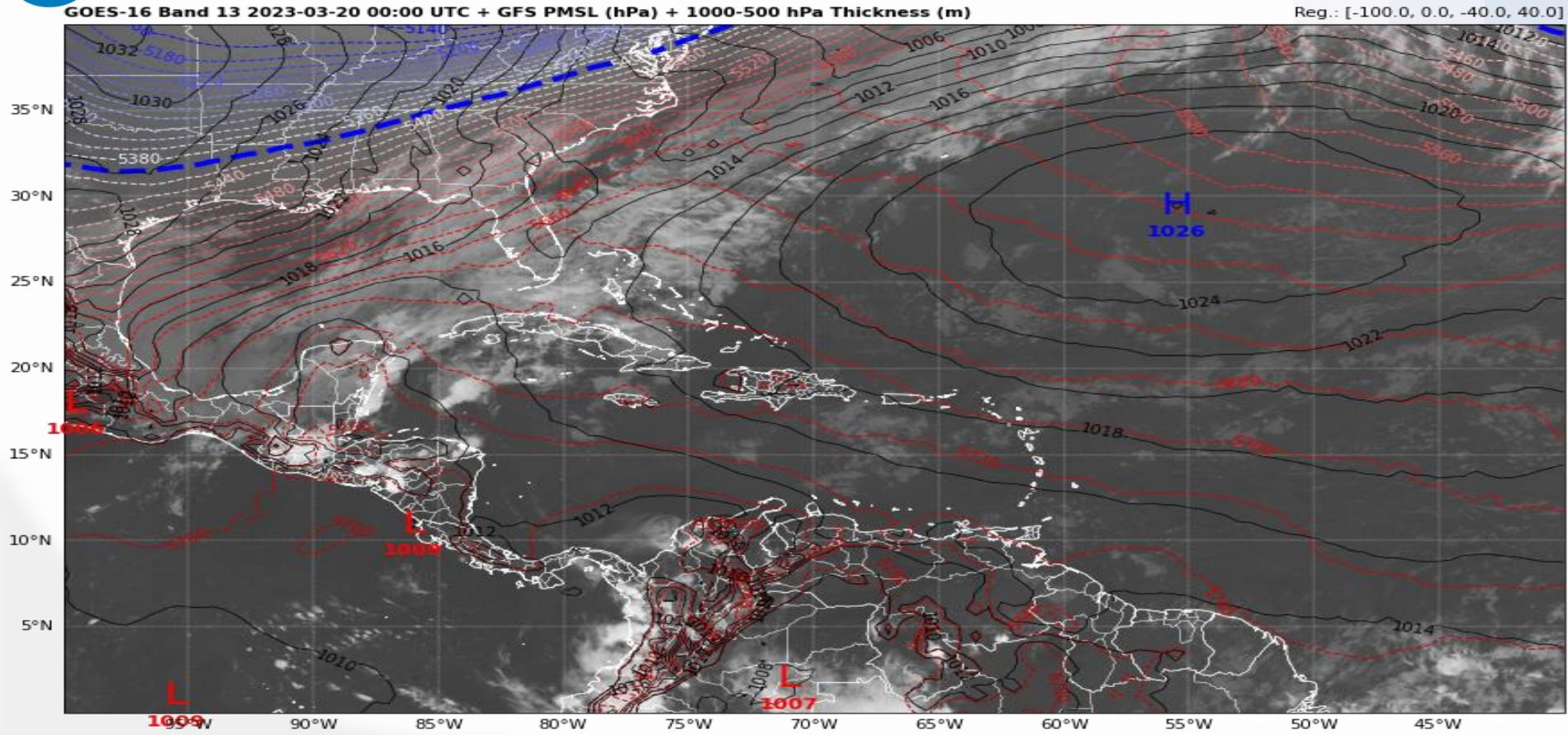
```
img2 = ax.contour(lons, lats, tmtmp, cmap='jet', linewidths=0.8, levels=levels)
```

Script 6: Plotting Contours and Labels





Unfilled Contours Will be Important for Future Plots



Some Quick Modifications

- Change the appearance and type of contours / labels





Script 7: Custom Color Palettes

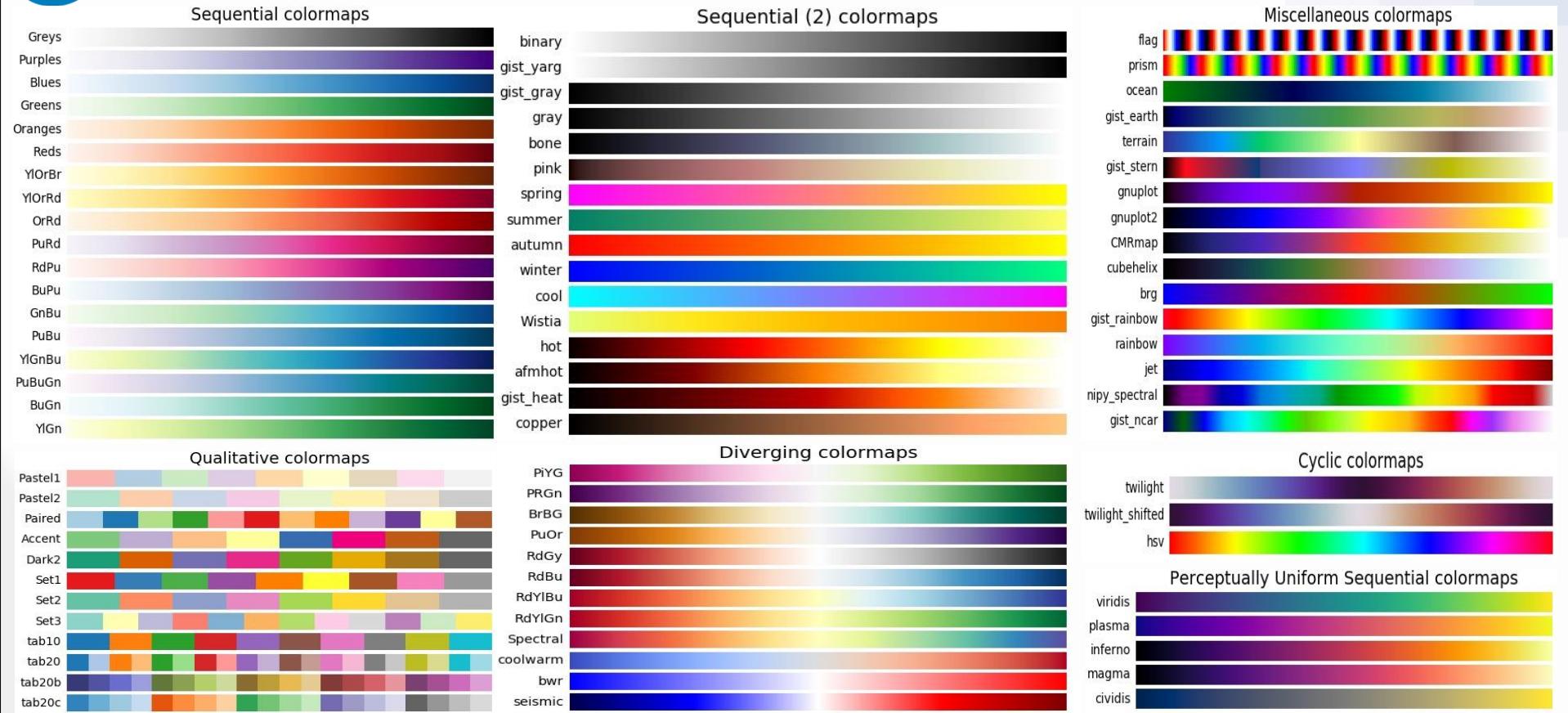
LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

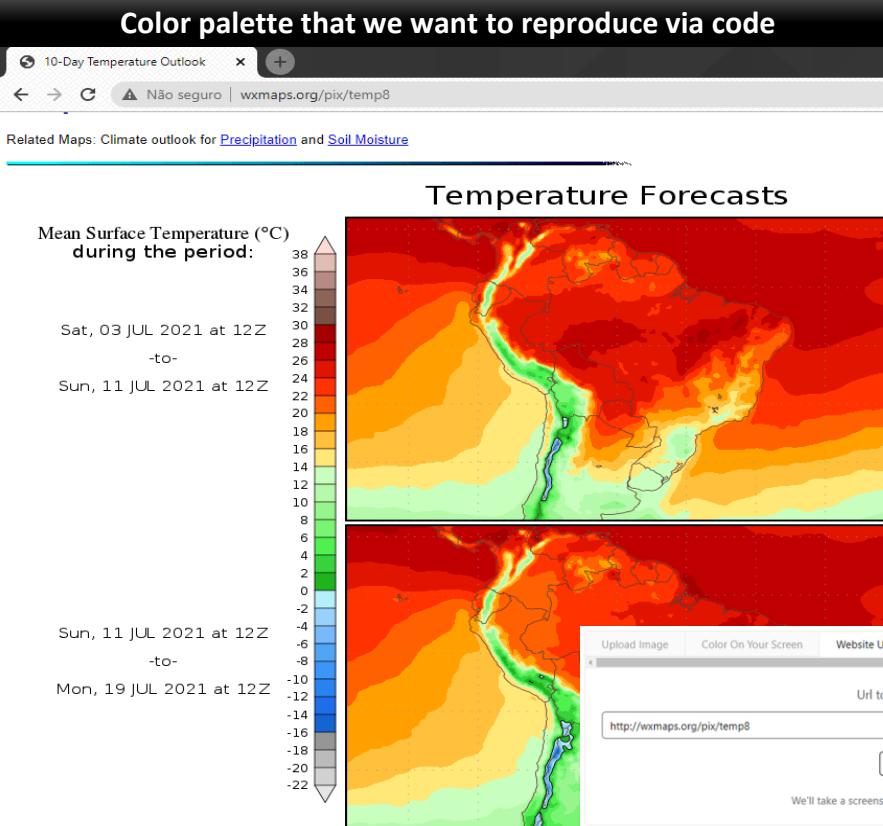
IMAGE GENERATION

Script 7: Custom Color Palettes

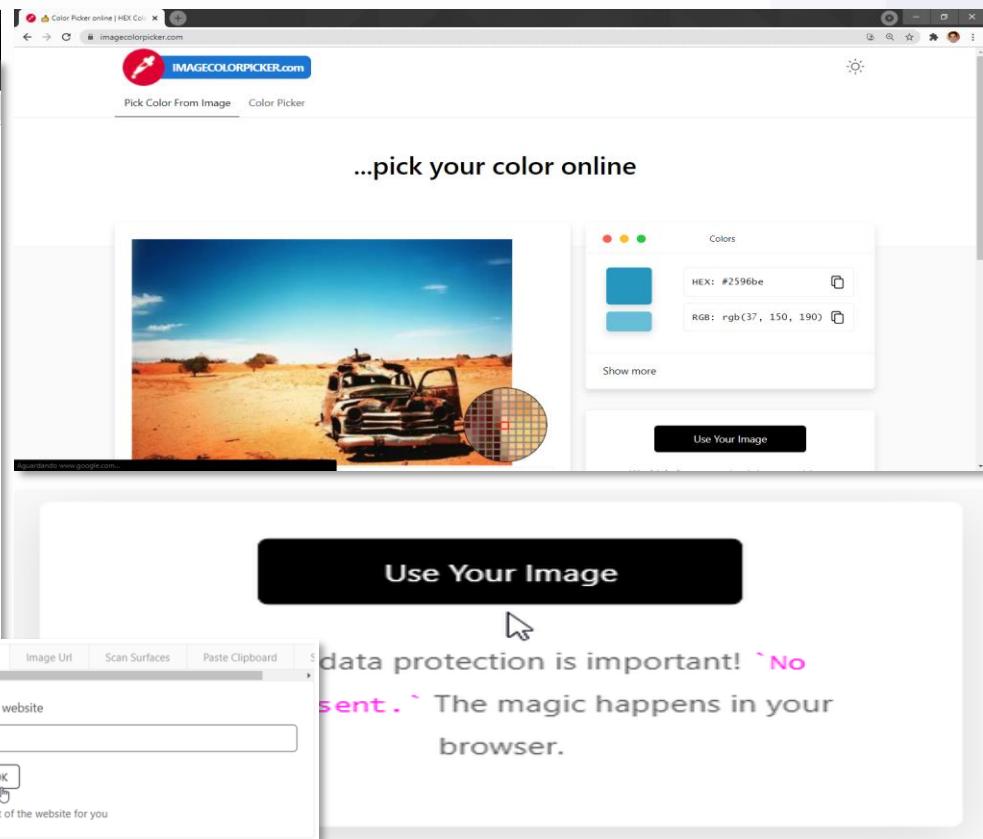


Script 7: Custom Color Palettes

<http://wxmaps.org/pix/temp8>



<https://imagecolorpicker.com/>



...pick your color online

Colors

HEX: #2596be

RGB: rgb(37, 150, 190)

Show more

Use Your Image

Use Your Image

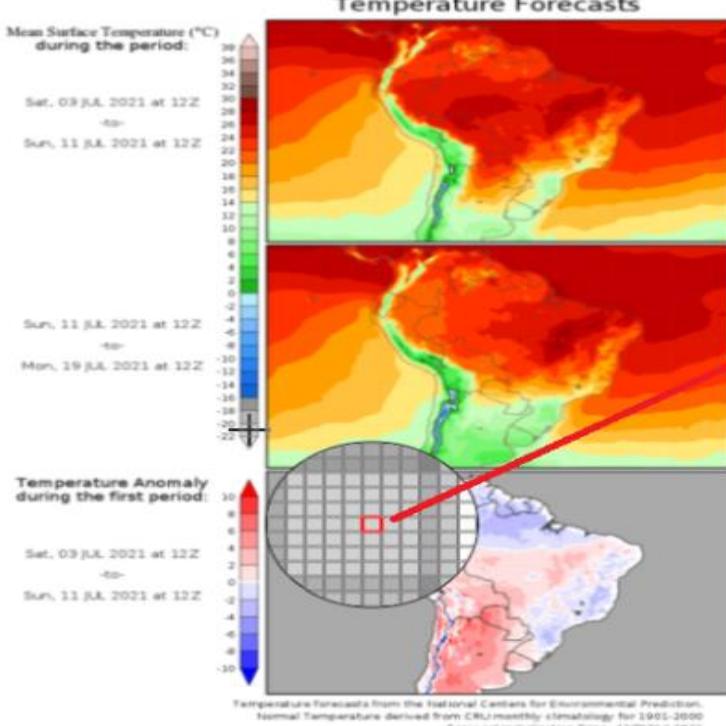
data protection is important! ^ No
sent. ^ The magic happens in your
browser.

Script 7: Custom Color Palettes

<https://imagecolorpicker.com/>

Temperature Outlook for South America

Related Map: Climate outlook for Precipitation and Soil Moisture



Colors

HEX: #d2d2d2

RGB: rgba(210,210,210,

Show more

Use Your Image

We think data protection is important! **No data is sent.** The magic happens in your browser.

Script 7: Custom Color Palettes

Explaining the new instructions:

```
1  #  
2  # INPE / CPTEC Training: NWP Data Processing With Python - Script 7: Custom Color Palettes  
3  # Author: Diego Souza  
4  #  
5  import pygrib           # Provides a high-level interface to the ECMWF ECCODES C library for reading GRIB files  
6  import matplotlib.pyplot as plt    # Plotting library  
7  import cartopy, cartopy.crs as ccrs  # Plot maps  
8  import cartopy.io.shapereader as shapereader # Import shapefiles  
9  import numpy as np      # Scientific computing with Python  
10 import matplotlib # Imports Matplotlib (General)  # General-purpose library for creating static, animated, and interactive visualizations in Python  
11 #  
12  
13 # Open the GRIB file  
14 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")  
15  
16 # Select the variable  
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('Forecast: ' + ftime)  
27 print('Valid: ' + valid + ' UTC')  
28  
29 # Select the extent [min. lon, min. lat, max. lon, max. lat]  
30 extent = [-93.0, -60.00, -25.00, 18.00]  
31  
32 # Read the data for a specific region  
33 tmtmp, lats, lons = grb.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)  
34  
35 #-----  
36 # Convert from K to °C  
37 tmtmp = tmtmp - 273.15  
38  
39 #
```

LIBRARIES

DATA READING AND MANIPULATION

Script 7: Custom Color Palettes

PLOT CONFIGURATION

```

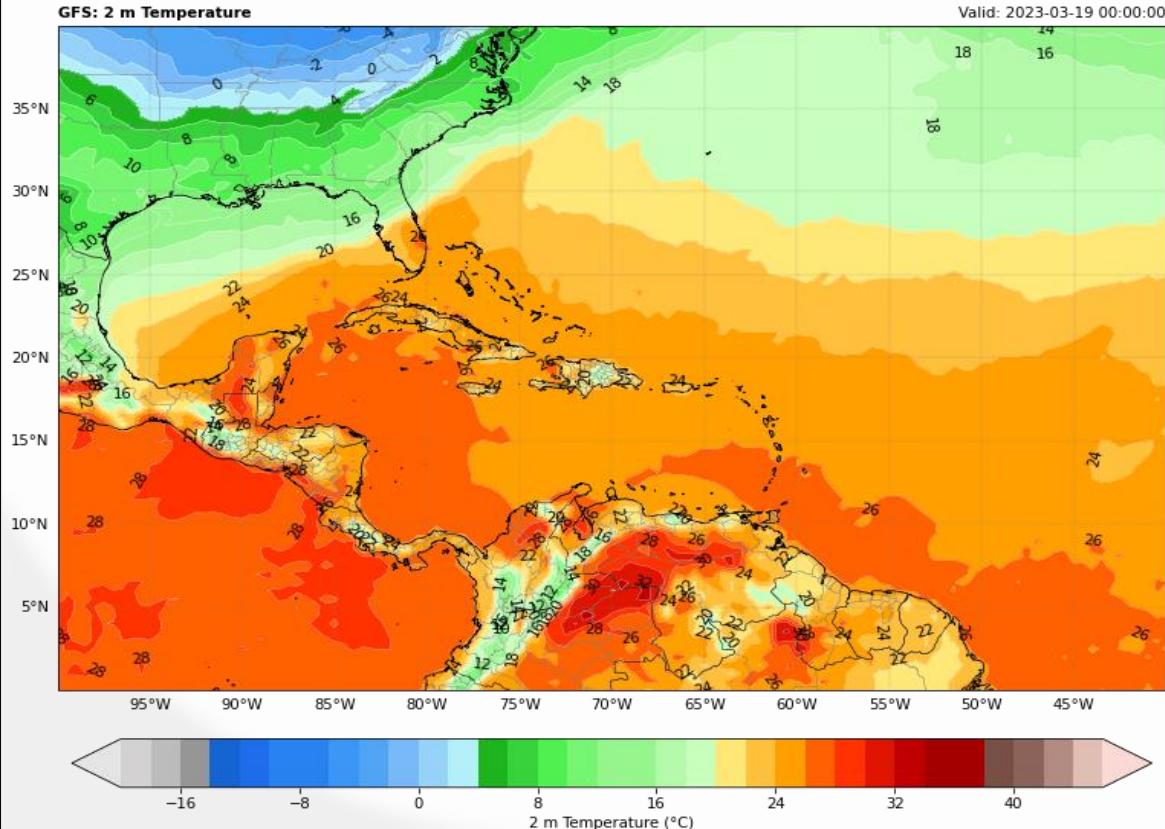
39
40 # Choose the plot size (width x height, in inches)
41 plt.figure(figsize=(8,8))
42
43 # Use the Cylindrical Equidistant projection in cartopy
44 ax = plt.axes(projection=ccrs.PlateCarree())
45
46 # Define the image extent
47 img_extent = [extent[0], extent[2], extent[1], extent[3]]
48
49 # Add a shapefile
50 # https://geoftp.ibge.gov.br/organizacao-do-territorio/malhas-territoriais/malhas-municipais/municipio-2019/Brasil-municipios-da-federacao.shp
51 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
52 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
53
54 # Add coastlines, borders and gridlines
55 ax.coastlines(resolution='10m', color='black', linewidth=0.8)
56 ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
57 gl = ax.gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25)
58 gl.top_labels = False
59 gl.right_labels = False
60
61 # Define de contour interval
62 data_min = -2
63 data_max = 38
64 interval = 2
65 levels = np.arange(data_min,data_max,interval)
66
67 # Create a custom color palette
68 colors = ['#d3d2d2', '#bcbcbc', '#969696', '#l464d2', '#le6eeb', '#2882f0',
69 '#3c96f5', '#50a5f5', '#78b9fa', '#96d2fa', '#b4f0fa', '#leb4le', '#37d23c',
70 '#50f050', '#78f573', '#96f58c', '#b4faaa', '#c0ffbe', '#ffe878', '#ffc03c',
71 '#ffa000', '#ff6000', '#ff3200', '#e11400', '#c00000', '#a50000', '#785046',
72 '#9c6355', '#b48b82', '#elbeb4']
73 cmap = matplotlib.colors.ListedColormap(colors)
74 cmap.set_over('#fadad5') — Create the new
75 cmap.set_under('#e5e5e5') — Values over max
76 — Values under min colormap
77
78 # Plot the contours
79 img1 = ax.contourf(lons, lats, tmpp, transform=ccrs.PlateCarree(), cmap=cmap, levels=levels, extend='both')
80 img2 = ax.contour(lons, lats, tmpp, transform=ccrs.PlateCarree(), colors='white', linewidths=0.5, levels=levels)
81 ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10', fmt = '%1.0f', colors= 'black')
82
83 # Add a colorbar
84 plt.colorbar(img1, label='2 m Temperature (°C)', orientation='vertical', pad=0.05, fraction=0.05)
85
86 # Add a title
87 plt.title('GFS: 2 m Temperature', fontweight='bold', fontsize=10, loc='left')
88 plt.title('Valid: ' + valid, fontsize=10, loc='right')
89

```

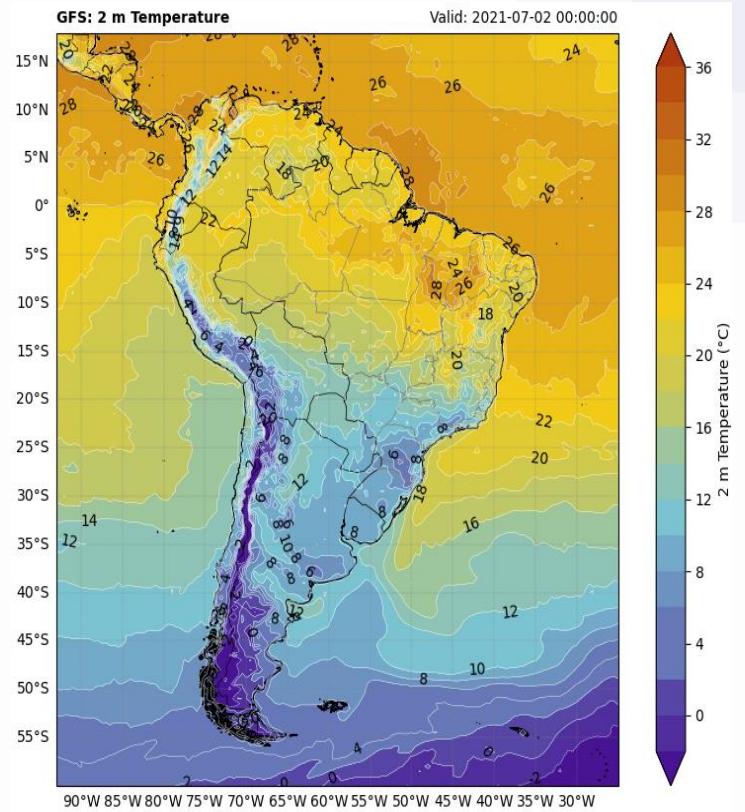
List of Colors (Hex values)

Script 7: Custom Color Palettes

Reference: <http://wxmaps.org/pix/temp8>

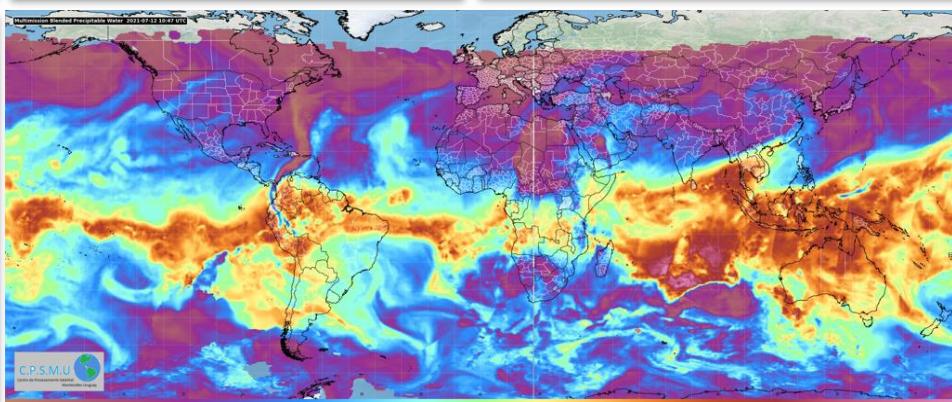
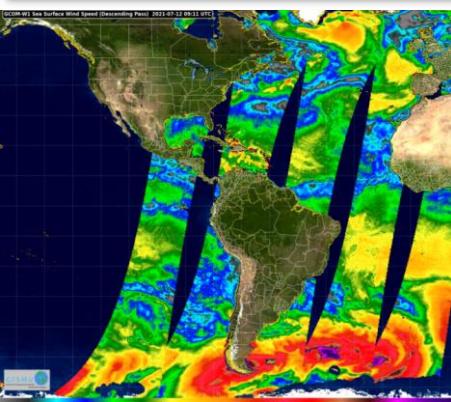
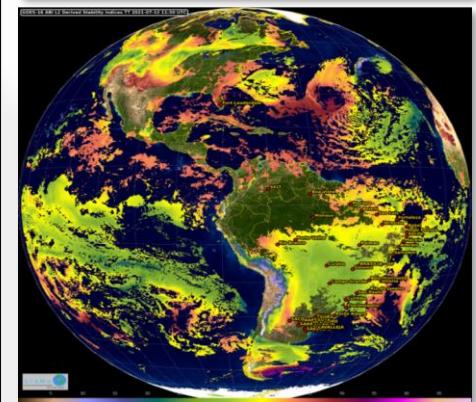
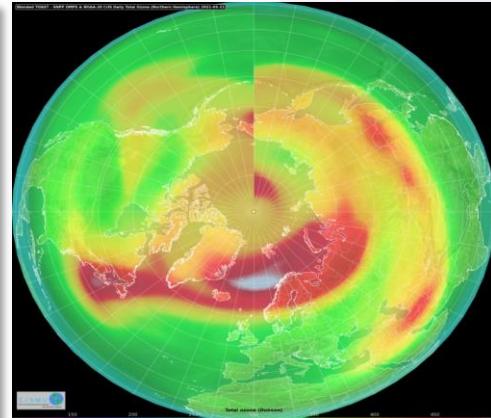
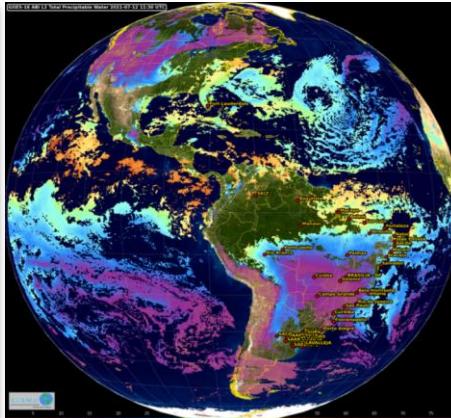
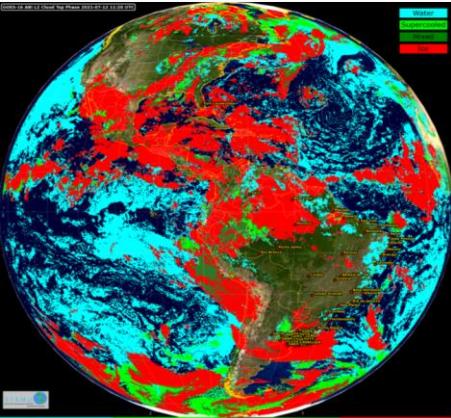
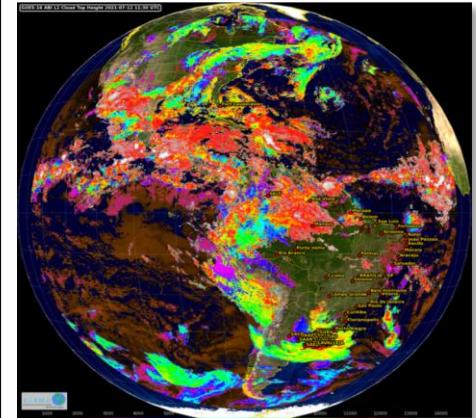


Reference: <https://previsaonumerica.cptec.inpe.br/>



Script 7: Custom Color Palettes

Some examples of satellite data plots with custom color palettes





Script 8: Smoothing Contours

LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

Script 8: Smoothing Contours

What is the size of our array?

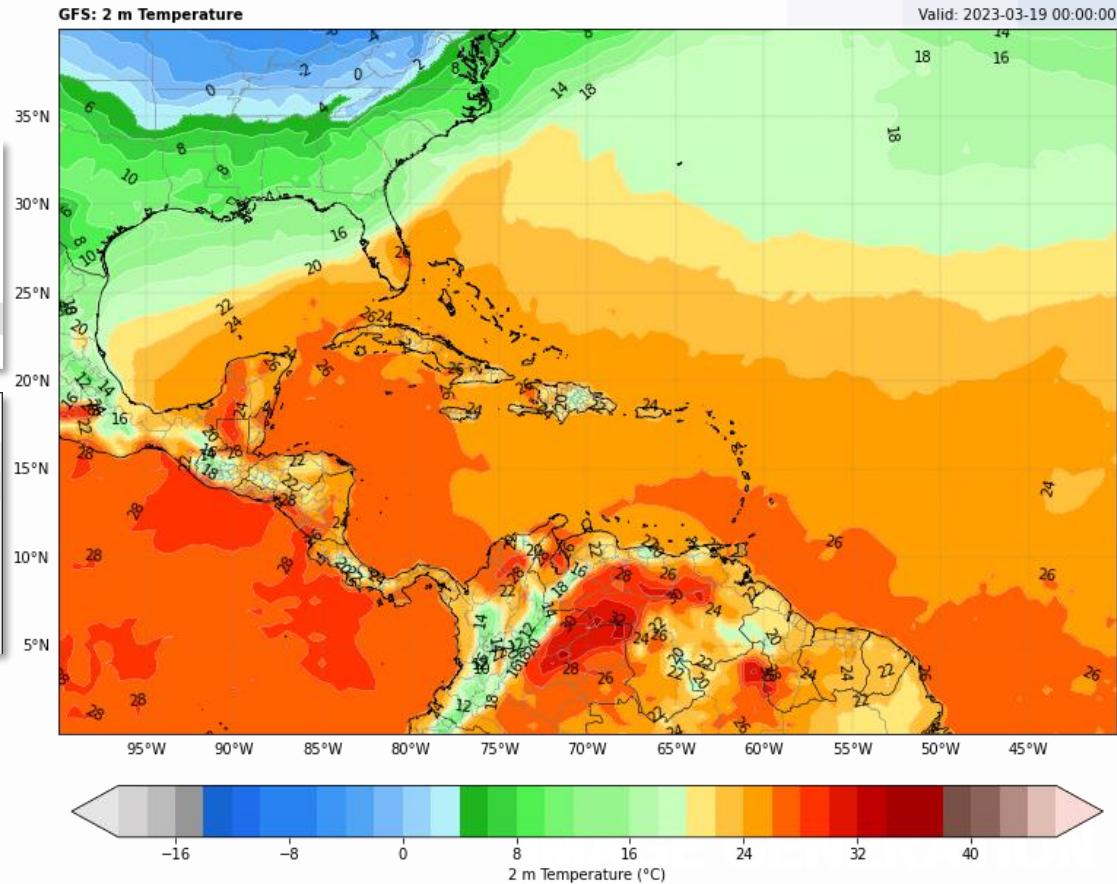
```
1 #-----  
2 # Convert to Celsius  
3 tmtmp = tmtmp - 273.15  
4  
5 # Print the array dimensions  
6 print("Array dimensions : ", tmtmp.shape)  
7 #-----
```

Anaconda Prompt (miniconda3) - python script_08.py

(workshop) c:\VLAB\Python\NWP>python script_08.py
Init: 2023-03-19 00:00:00 UTC
Run: 00Z
Forecast: +0
Valid: 2023-03-19 00:00:00 UTC

Array dimensions:
(81, 121)

Let's plot a smaller region



Script 8: Smoothing Contours

Changing the plot region

```
# Select the extent [min. lon, min. lat, max. lon, max. lat]
#extent = [-100.0, 0.00, -40.00, 40.00]
extent = [-85.0, 10.00, -60.00, 25.00]
```

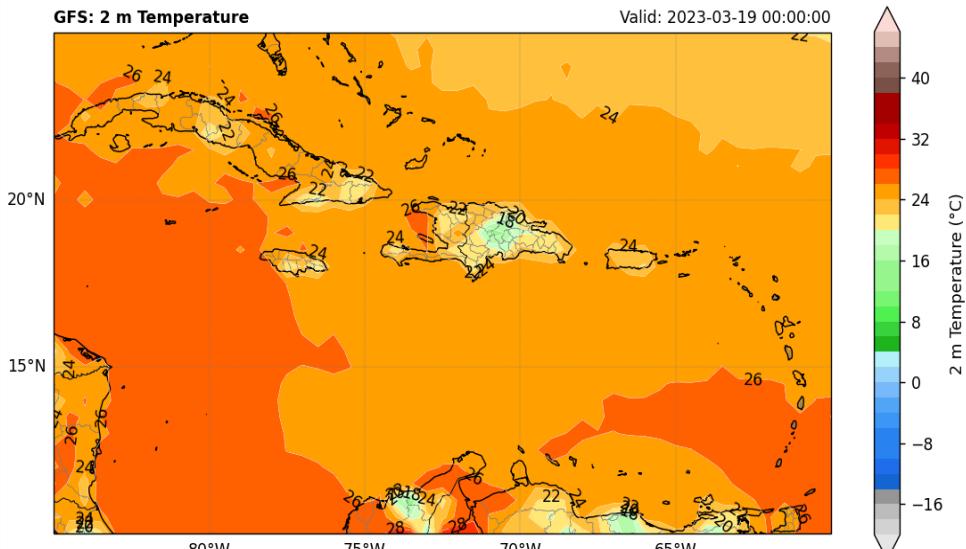
```
1 #-----
2 # Convert to Celsius
3 tmtmp = tmtmp - 273.15
4
5 # Print the array dimensions
6 print("Array dimensions :", tmtmp.shape)
7 #-----
```

```
Anaconda Prompt (miniconda3) - python script_08.py
(workshop) c:\VLAB\Python\NWP>python script_08.py
Init: 2023-03-19 00:00:00 UTC
Run: 00Z
Forecast: +0
Valid: 2023-03-19 00:00:00 UTC

Array dimensions:
(31, 51)
```

Dimensions of the reduced matrix

The low resolution of the data is evident



Script 8: Smoothing Contours

Explaining the new instructions:

```
11 #
12 #-----#
13 # Open the GRIB file
14 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
15
16 # Select the variable
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('Forecast: ' + ftime)
27 print('Valid: ' + valid + ' UTC')
28
29 # Select the extent [min. lon, min. lat, max. lon, max. lat]
30 extent = [-55.0, -15.00, -35.00, 5.00]
31
32 # Read the data for a specific region
33 tmtmp, lats, lons = grb.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)
34
35 #-----
36 # Convert from K to °C
37 tmtmp = tmtmp - 273.15
38
39 print("\nArray dimensions before smoothing:")
40 print(tmtmp.shape)
41
42 # Smooth the contours
43 import scipy.ndimage
44 tmtmp = scipy.ndimage.zoom(tmtmp, 3) # Imports the “ndimage” library
45 lats = scipy.ndimage.zoom(lats, 3) # Apply the “zoom” in the data array
46 lons = scipy.ndimage.zoom(lons, 3) # Apply the “zoom” in the latitude array
47
48 print("Array dimensions after smoothing:")
49 print(tmtmp.shape)
```

DATA READING AND MANIPULATION

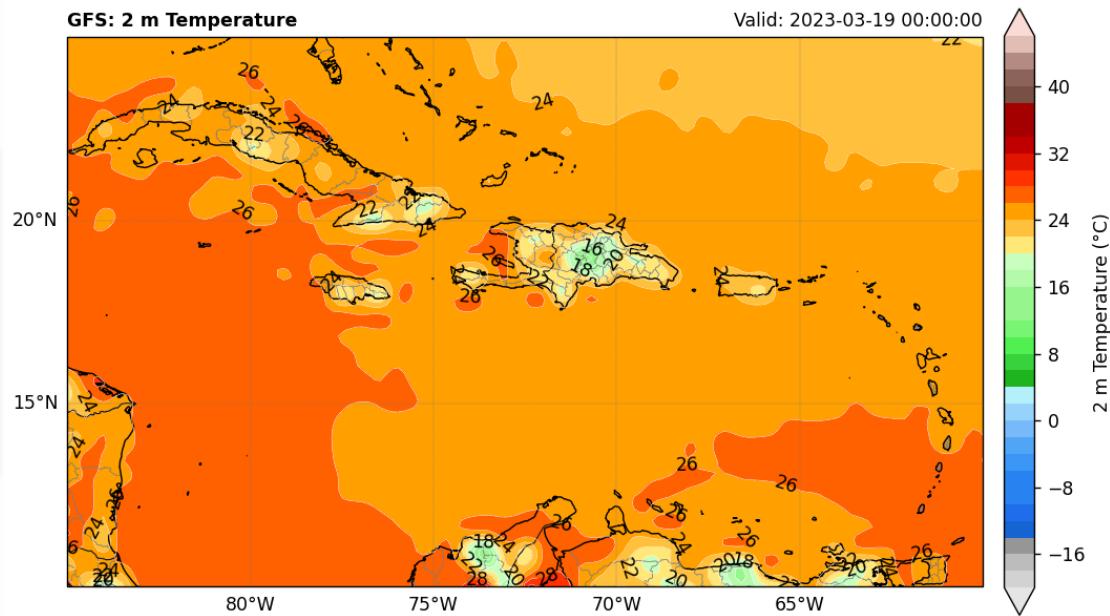
Imports the “ndimage” library
Apply the “zoom” in the data array
Apply the “zoom” in the latitude array
Apply the “zoom” in the longitude array

Script 8: Smoothing Contours

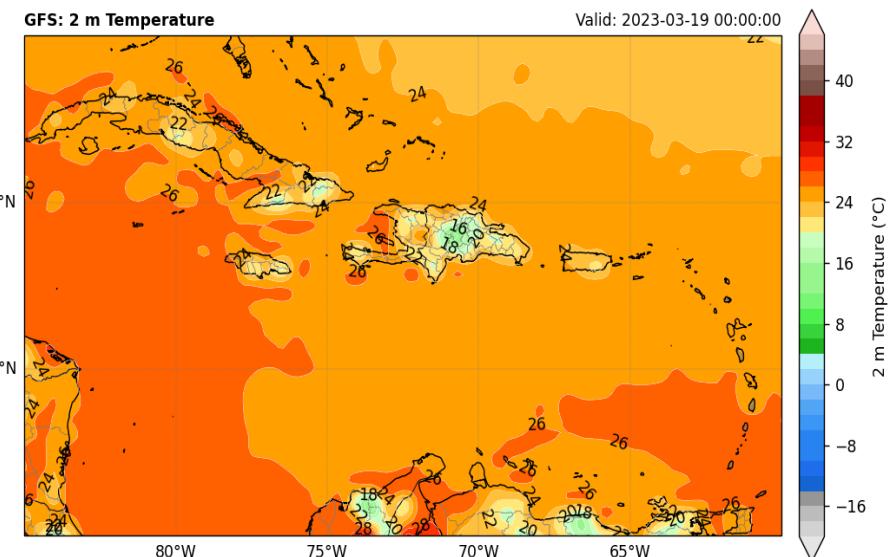
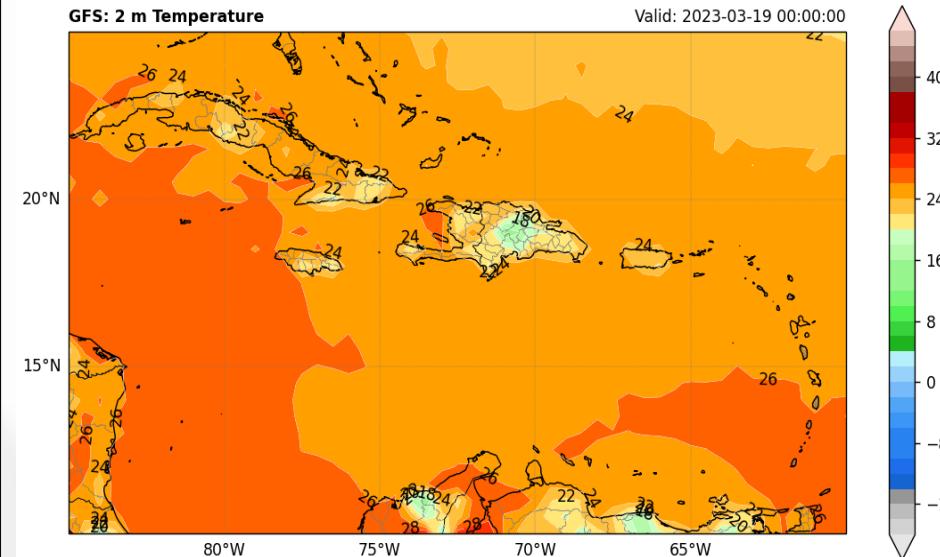
```
Anaconda Prompt (miniconda3) - python script_08.py
(workshop) c:\VLAB\Python\NWP>python script_08.py
Init: 2023-03-19 00:00:00 UTC
Run: 00Z
Forecast: +0
Valid: 2023-03-19 00:00:00 UTC

Array dimensions before smoothing:
(31, 51)
Array dimensions after smoothing:
(93, 153)
```

New dimensions after smoothing



Script 8: Smoothing Contours



Script 9: Working With Multiple Files

So far we have only worked with a single input file, generating a single plot in the output

```
#!/usr/bin /CLOUD/Training/HDF Data Processing With Python / Scripts / Smoothing the Contents

# Author: Diego Stocco

# Import python modules
import pyrtif          # Provides a high-level interface to the ROMS/HDF5 C library for reading GRIB files
import cartopy           # Plot maps
import numpy             # Numerical computations
import xarray            # Scientific computing with Python
import matplotlib       # Comprehensive library for creating static, animated, and interactive visualizations in Python

# Open the GRIB file
grib = pyrtif.open("rome_0000.grib/full.gphd.0000")

# grib.select(name='2 metre temperature') #0

# Get information from the file
init = grib.getInitiate() # init date / time
forecastTime = grib.getForecastTime() # Forecast hour
validTime = init + forecastTime # Valid date / time
print("Init: " + str(init))
print("Forecast: " + str(forecastTime))
print("Valid: " + str(validTime))

# Set extent
extent = [-155.0, -150.0, -35.00, 5.00]

# Read the data for a specific region
lat0, lon0 = grib.getRect(extent[0], extent[1], extent[2]-extent[0], extent[3]-extent[0]+360, extent[2]+360)

# Convert from m to TC
temp = temp * 273.15

print("Latitude dimensions before smoothing:")
print(lat0.shape)

# Smooth the data
import scipy.ndimage
temp = scipy.ndimage.gaussian_filter(temp, 3)
lat0 = scipy.ndimage.gaussian(lat0, 3)
lon0 = scipy.ndimage.gaussian(lon0, 3)

print("Array dimensions after smoothing:")
print(temp.shape)

# Create a figure
plt.figure(figsize=(8,8))

# Show the GFS13000m Regridded projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree())
ax.set_extent(extent[0], extent[1], extent[2], extent[3])

# Add a coastline
ax.add_feature(cartopy.feature.COASTLINES, color='black', linewidth=0.5)
ax.add_feature(cartopy.feature.BORDERS, color='black', linewidth=0.5)
ax.add_feature(cartopy.feature.LAND, color='brown', alpha=0.5, linewidth=0.5)
ax.add_feature(cartopy.feature.OCEAN, color='blue', alpha=0.5, linewidth=0.5)
ax.grid_label = False
ax.grid_label = False

# Add a grid
data_min = -20
data_max = 40
interval = np.arange(data_min,data_max,interval)
levels = np.arange(data_min,data_max,interval)

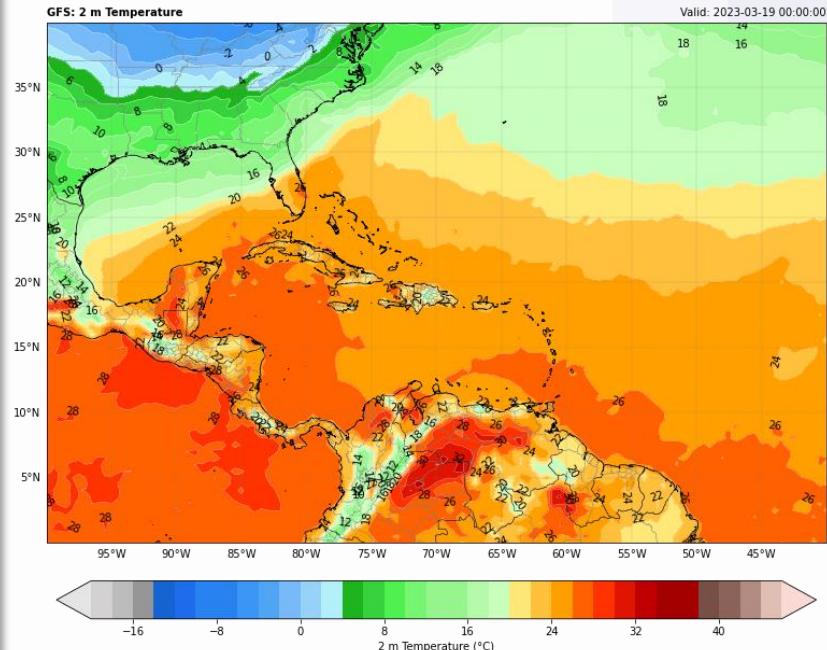
# Color map
colors = ["#f0f0f0", "#e6e6fa", "#d9d9ff", "#cfcfcf", "#bcbcb0", "#a6a6a6", "#999999", "#808080", "#707070", "#606060", "#505050", "#404040", "#303030", "#202020", "#101010", "#000000"]
cmap = matplotlib.colors.ListedColormap(colors)
cmap.set_over('#ffffcc')
cmap.set_under('#cccccc')

# Plot the contours
img = ax.contourf(lat0, temp, levels, transform=ccrs.PlateCarree(), colors=cmap, linewidths=0.5, antialiased=True)
img.set_under(color='white', alpha=0.5, linewidths=0.5, transform=ccrs.PlateCarree())
img.set_over(color='white', alpha=0.5, linewidths=0.5, transform=ccrs.PlateCarree())
img.set_under(alpha=0.5, linewidths=0.5, transform=ccrs.PlateCarree())
img.set_over(alpha=0.5, linewidths=0.5, transform=ccrs.PlateCarree())

# Add a colorbar
pcbar = plt.colorbar(img, label="2 metre temperature", orientation='vertical', pad=0.05, fraction=0.05)
pcbar.set_label("2 metre temperature", rotation=90, verticalalignment='bottom', fontweight='bold', fontstyle='italic')
pcbar.set_label("Valid: " + validTime.strftime("%Y-%m-%d %H:%M"), verticalalignment='top', fontweight='bold', fontstyle='italic')

# Save the image
plt.savefig("Image_B.png")

# Show the image
plt.show()
```



Let's see how to work with multiple input files using a single script

Script 9: Working With Multiple Files

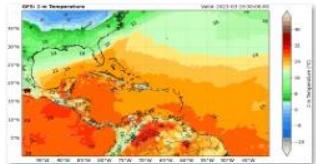
The diagram illustrates a workflow for generating a temperature animation. It starts with a code snippet for **LIBRARIES**, followed by **PREV. CONFIGURATION**. A large red arrow points down to a code snippet for **LOOP DATA READING AND MANIP.**. Another red arrow points down to a code snippet for **PLOT CONF.**. A final red arrow points down to a code snippet for **IMAGE GENERATION**. To the right of the flowchart is a screenshot of a file browser window titled "(C:) > VLAB > Python > NWP > Animation". The browser shows a list of files: "animation.gif", "image_loop_0.png", "image_loop_3.png", "image_loop_6.png", "image_loop_9.png", "image_loop_12.png", "image_loop_15.png", "image_loop_18.png", "image_loop_21.png", and "image_loop_24.png". A search bar at the top right contains the text "Search Animation".

```
LIBRARIES
PREV.
CONFIGURATION
LOOP
DATA READING
AND MANIP.
PLOT CONF.
IMAGE GENERATION
```

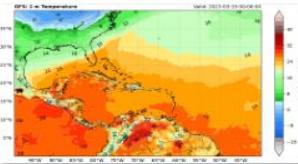
```
(C:) > VLAB > Python > NWP > Animation
animation.gif
image_loop_0.png
image_loop_3.png
image_loop_6.png
image_loop_9.png
image_loop_12.png
image_loop_15.png
image_loop_18.png
image_loop_21.png
image_loop_24.png
Search Animation
```



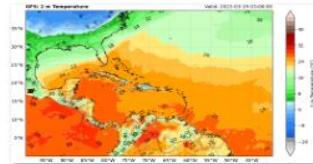
Search Animation



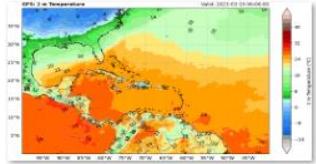
animation.gif



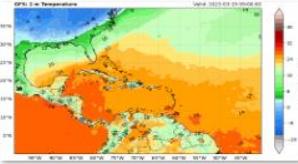
image_loop_0.png



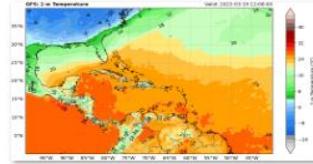
image_loop_3.png



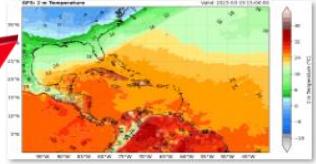
image_loop_6.png



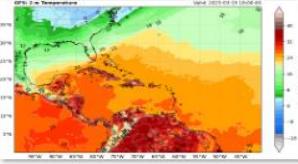
image_loop_9.png



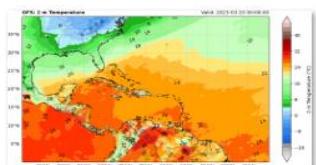
image_loop_12.png



image_loop_15.png



image_loop_18.png



image_loop_24.png

Script 9: Working With Multiple Files

Explaining the new instructions:

```

1  # -----
2  # INPE / CPTEC Training: NWP Data Processing With Python - Script 9: Working With Multiple Files
3  # Author: Diego Souza
4
5  import pygrib                                # Provides a high-level interface to the ECMWF ECCODES C library for reading GRIB files
6  import matplotlib.pyplot as plt                # Plotting library
7  import cartopy, cartopy.crs as ccrs            # Plot maps
8  import cartopy.io.shapereader as shapereader   # Import shapefiles
9  import numpy as np                            # Scientific computing with Python
10 import matplotlib                           # Comprehensive library for creating static, animated, and interactive visualizations in Python
11 import os                                     # Operating system interfaces
12
13
14 # Animation directory
15 dir = "Animation"; os.makedirs(dir, exist_ok=True)  — Create a folder named “Animation”
16
17 # Select the extent [min. lon, min. lat, max. lon, max. lat]
18 extent = [-78.0, -40.0, -30.0, 12.00]
19
20 #-----
21
22 # GRIB file name without the last three characters
23 file = ("gfs.t00z.pgrb2full.0p50.f")  — GRIB File Name without Last 3 Characters
24
25 # Data you want to process in the loop
26 hour_ini = 0      # Init time
27 hour_end = 24     # End time
28 hour_int = 3       # Interval
29
30
31 for hour in range(hour_ini, hour_end + 1, hour_int):  — For each time interval between start time and end time
32
33     # File to process
34     grib = file + str(hour).zfill(3)  — GRIB File Name + Time (three characters)
35
36     # If the file exists
37     if (os.path.exists(grib)):
38
39         # Process the file
40         print("\nProcessing file: ", grib)
41
42         # Read the GRIB file
43         grib = pygrib.open(grib)  — ... process the file
44

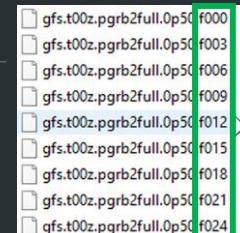
```

LIBRARIES

INITIAL CONFIGURATION

LOOP

The diagram illustrates the execution flow of the script. It begins with importing libraries, followed by creating an 'Animation' directory. The main loop processes GRIB files for each hour from 'hour_ini' to 'hour_end'. Inside the loop, it checks if a file exists before processing it. An arrow points from the 'file' variable in the loop to a list of GRIB files on the right, which are highlighted with a green border.



Script 9: Working With Multiple Files

Explaining the new instructions:

PLOT CONFIGURATION

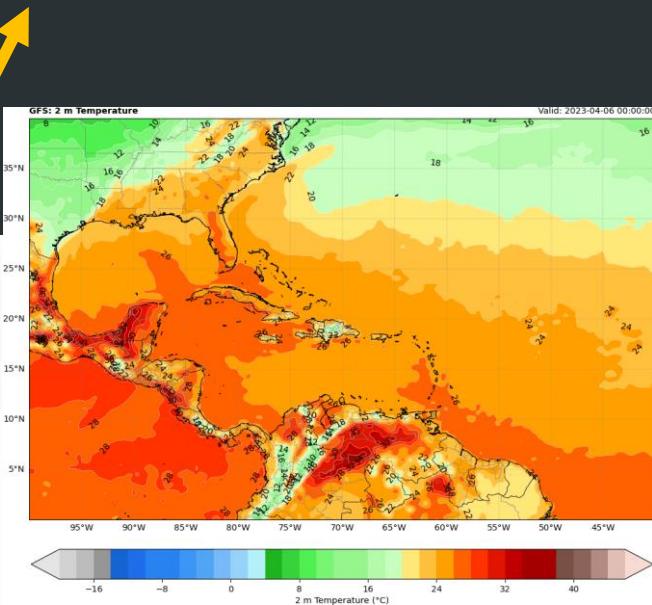
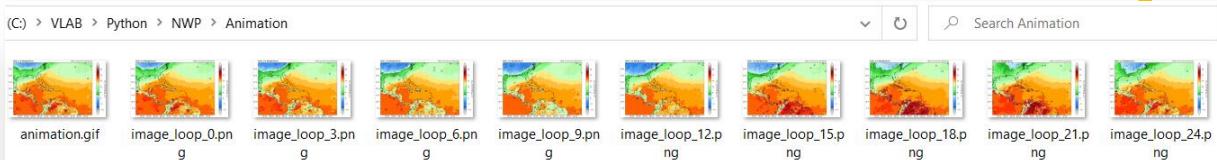
```

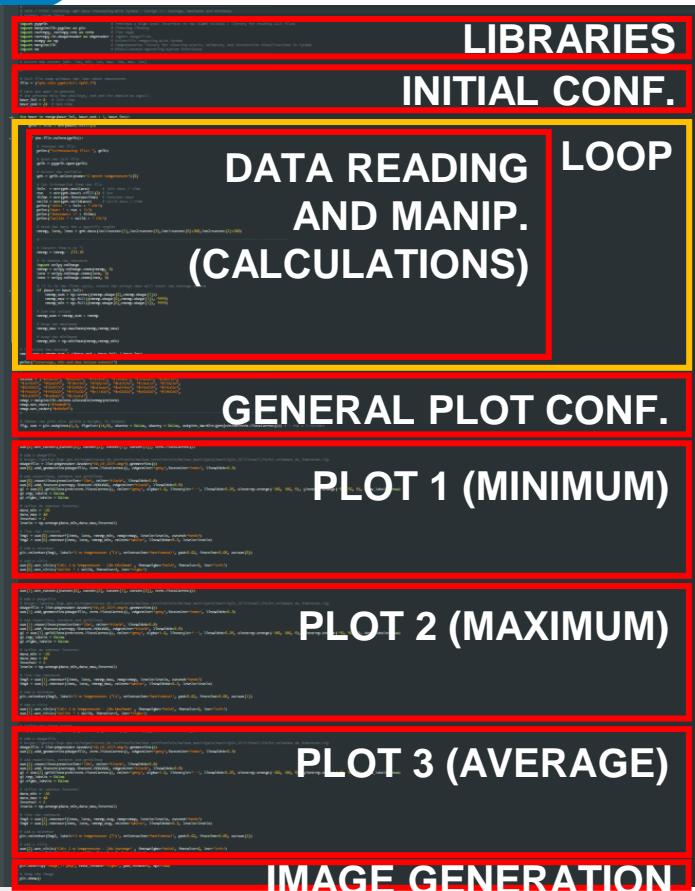
73
74     # Choose the plot size (width x height, in inches)
75     plt.figure(figsize=(8,8))
76
77     # Use the Cylindrical Equidistant projection in cartopy
78     ax = plt.axes(projection=ccrs.PlateCarree())
79
80     # Define the image extent
81     img_extent = [extent[0], extent[2], extent[1], extent[3]]
82
83     # Add a shapefile
84     # https://geoftp.ibge.gov.br/organizacao\_do\_territorio/malhas\_territoriais/malhas\_municipais/municipio\_2019/Brasil/BR/br\_unidades\_da\_federacao.zip
85     shapefile = list(shapereader.Reader('BR_UF_2019.shp').geometries())
86     ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
87
88     # Add coastlines, borders and gridlines
89     ax.coastlines(resolutions='10m', colors='black', linewidth=0.8)
90     ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
91     gl = ax.gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
92     gl.top_labels = False
93     gl.right_labels = False
94
95     # Define de contour interval
96     data_min = -20
97     data_max = 48
98     interval = 2
99     levels = np.arange(data_min,data_max,interval)
100
101    # Create a custom color palette
102    colors = ['#d3d2d2', '#bcbcbc', '#969696', '#1464d2', '#le6eeb', '#2882f0',
103              '#3c36f5', '#50a5f5', '#78b5fa', '#86d2fa', '#b4f0fa', '#leb4le', '#37d23c',
104              '#50f050', '#78f573', '#96f58c', '#b4faaa', '#c8ffbe', '#ffe878', '#ffc03c',
105              '#ffa000', '#ff6000', '#fff3200', '#e11400', '#c00000', '#a50000', '#785046',
106              '#8c6359', '#b48b82', '#1beb4b']
107    cmap = matplotlib.colors.ListedColormap(colors)
108    cmap.set_over('#fadad5')
109    cmap.set_under('#e5e5e5')
110
111    # Plot the contours
112    img1 = ax.contourf(lons, lats, tmmp, transform=ccrs.PlateCarree(), cmap=cmap, levels=levels, extend='both')
113    img2 = ax.contour(lons, lats, tmmp, transform=ccrs.PlateCarree(), colors='white', linewidths=0.3, levels=levels)
114    ax.clabel(img2, inline=1, inline_spacing=0, fontsize=10, fmt = '%1.0f', colors= 'black')
115
116    # Add a colorbar
117    plt.colorbar(img1, label='2 m Temperature (°C)', orientation='vertical', pad=0.05, fraction=0.05)
118
119    # Add a title
120    plt.title('GFS: 2 m Temperature', fontweight='bold', fontsize=10, loc='left')
121    plt.title('Valid: ' + valid, fontsize=10, loc='right')
122
123    # Save the image
124    plt.savefig('Animation//image_loop_' + str(hour) + '.png', bbox_inches='tight', pad_inches=0, dpi=100)

```

Remove white spaces in the image boundaries

Script 10: Creating an Animation





```
#
# INPE / CPTEC Training: NWP Data Processing With Python - Script 11: Average, Maximums and Minimums
# Author: Diego Souza

# Provides a high-level interface for reading GRIB files
# Plotting library
# Plot maps
# Import shapefiles
# Scientific computing with Python
# Comprehensive library for creating static, interactive and animated plots
# Miscellaneous operating system interfaces

# Select the extent [min. lon, min. lat, max. lon, max. lat]
extent = [-78.0, -40.00, -30.00, 12.00]

# GRIB file name without the last three characters
file = ("gfs.t00z.pgrb2full.0p50.f")

# Data you want to process
# (to process only the analysis, end and inc should be equal).
hour_ini = 0 # Init time
hour_end = 24 # End time
hour_int = 3 # Increment

for hour in range(hour_ini, hour_end + 1, hour_int):

    grib = file + str(hour).zfill(3)

    # If the file exists
    if (os.path.exists(grib)):

        # Process the file
        print("\nProcessing file: ", grib)

        # Read the GRIB file
        grib = pygrib.open(grib)
```

Our scripts are getting bigger

```
27 for hour in range(hour_ini, hour_end + 1, hour_int):
28     grib = file + str(hour).zfill(3)
29
30     # If the file exists
31     if (os.path.exists(grib)):
32
33         # Process the file
34         print("\nProcessing file: ", grib)
35
36         # Read the GRIB file
37         grib = pygrib.open(grib)
38
39         # Select the variable
40         grb = grib.select(name='2 metre temperature')[0]
41
42         # Get information from the file
43         init = str(grb.analDate)      # Init date / time
44         run = str(grb.hour).zfill(2)  # Run
45         ftime = str(grb.forecastTime) # Forecast hour
46         valid = str(grb.validDate)   # Valid date / time
47         print('Init: ' + init + ' UTC')
48         print('Run: ' + run + 'Z')
49         print('Forecast: +' + ftime)
50         print('Valid: ' + valid + ' UTC')
51
52
53         # Read the data for a specific region
54         tmtmp, lats, lons = grb.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)
55
56
57         #-----
58
59         # Convert from K to °C
60         tmtmp = tmtmp - 273.15
61
62         # To smooth the contours
63         import scipy.ndimage
64         tmtmp = scipy.ndimage.zoom(tmtmp, 3)
65         lats = scipy.ndimage.zoom(lats, 3)
66         lons = scipy.ndimage.zoom(lons, 3)
```

Explain the new instructions:

LOOP

DATA READING AND MANIPULATION

```
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102
```

DATA READING AND MANIPULATION (CONT.)

If it's the first cycle,
create the arrays with the
appropriate dimensions

In each cycle, sums the current value

In each cycle, it keeps the maximum

In each cycle, keeps the minimum

LOOP (CONT.)

No final do ciclo, calcula a média

GENERAL PLOT CONFIGURATION

1 row x 3 columns plot

Cartopy projection

```
# Create a custom color palette  
colors = ["#d3d2d2", "#bcbcbc", "#969696", "#1464d2", "#1e6eeb", "#2882f0",  
"#3c96f5", "#50a5f5", "#78b9fa", "#96d2fa", "#b4f0fa", "#leb4le", "#37d23c",  
"#50f050", "#78f573", "#96f58c", "#b4faaa", "#c8ffbe", "#ffe878", "#ffc03c",  
"#ffa000", "#ff6000", "#ff3200", "#e11400", "#c00000", "#a50000", "#785046",  
"#8c6359", "#b48b82", "#elbeb4"]  
cmap = matplotlib.colors.ListedColormap(colors)  
cmap.set_over('#fadad5')  
cmap.set_under('#e5e5e5')  
#--  
# Choose the plot size (width x height, in inches)  
fig, axs = plt.subplots(1,3, figsize=(14,5), sharex = False, subplot_kw=dict(projection=ccrs.PlateCarree())) # 1 row x 3 columns
```

Explaining the new instructions:

PLOT 1 (MINIMUM)

```
102 #-----
103
104 # Define the image extent
105 axs[0].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
106
107 # Add a shapefile
108 # https://geoftp.ibge.gov.br/organizacao\_do\_territorio/malhas\_teritoriais/malhas\_municipais/municipio\_2019/Brasil/BR/br\_unidades\_da\_federacao.zip
109 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
110 axs[0].add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
111
112 # Add coastlines, borders and gridlines
113 axs[0].coastlines(resolution='10m', color='black', linewidth=0.8)
114 axs[0].add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
115 gl = axs[0].gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
116 gl.top_labels = False
117 gl.right_labels = False
118
119 # Define de contour interval
120 data_min = -20
121 data_max = 48
122 interval = 2
123 levels = np.arange(data_min,data_max,interval)
124
125 # Plot the contours
126 img1 = axs[0].contourf(lons, lats, ttmp_min, cmap=cmap, levels=levels, extend='both')
127 img2 = axs[0].contour(lons, lats, ttmp_min, colors='white', linewidths=0.3, levels=levels)
128
129 # Add a colorbar
130 plt.colorbar(img1, label='2 m Temperature (°C)', orientation='horizontal', pad=0.02, fraction=0.05, ax=axs[0])
131
132 # Add a title
133 axs[0].set_title('GFS: 2 m Temperature - 24h Minimum', fontweight='bold', fontsize=6, loc='left')
134 axs[0].set_title('Valid: ' + valid, fontsize=6, loc='right')
135
136 #-----
```

IMAGE GENERATION

Explaining the new instructions:

```
136 #--  
137 # Define the image extent  
138 axs[1].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())  
139 # Add a shapefile  
140 # https://deftp.ibge.gov.br/organizacao_dos_territorios/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip  
141 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())  
142 axs[1].add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)  
143 # Add coastlines, borders and gridlines  
144 axs[1].coastlines(resolution='10m', color='black', linewidth=0.8)  
145 axs[1].add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)  
146 gl = axs[1].gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)  
147 gl.top_labels = False  
148 gl.right_labels = False  
149 # Define de contour interval  
150 data_min = -20  
151 data_max = 48  
152 interval = 2  
153 levels = np.arange(data_min,data_max,interval)  
154 # Plot the contours  
155 img3 = axs[1].contourf(lons, lats, tmtmp_max, cmap=cmap, levels=levels, extend='both')  
156 img4 = axs[1].contour(lons, lats, tmtmp_max, colors='white', linewidths=0.3, levels=levels)  
157 # Add a colorbar  
158 plt.colorbar(img3, label='2 m Temperature (°C)', orientation='horizontal', pad=0.02, fraction=0.05, ax=axs[1])  
159 # Add a title  
160 axs[1].set_title('GFS: 2 m Temperature - 24h Maximum', fontweight='bold', fontsize=6, loc='left')  
161 axs[1].set_title('Valid: ' + valid, fontsize=6, loc='right')  
162  
163  
164  
165  
166  
167  
168  
169
```

PLOT 2 (MAXIMUM)

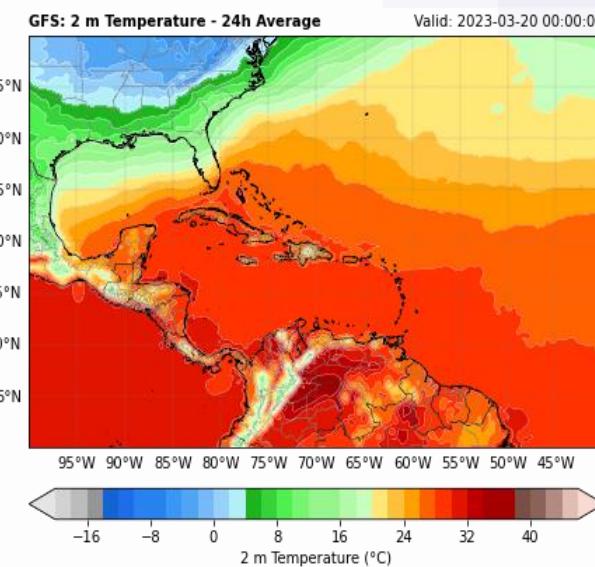
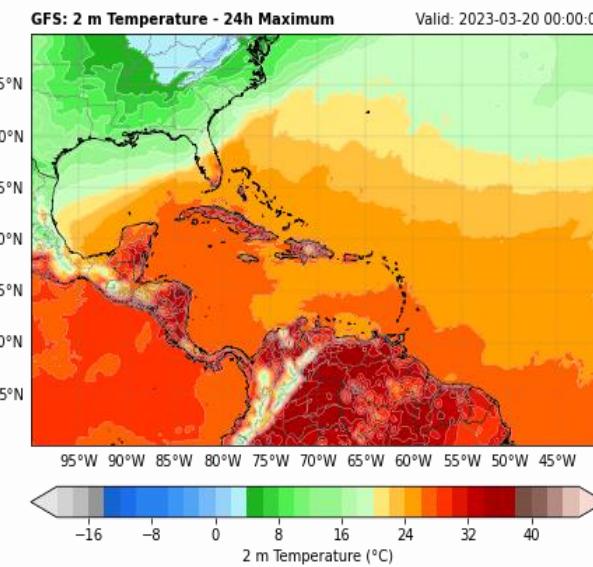
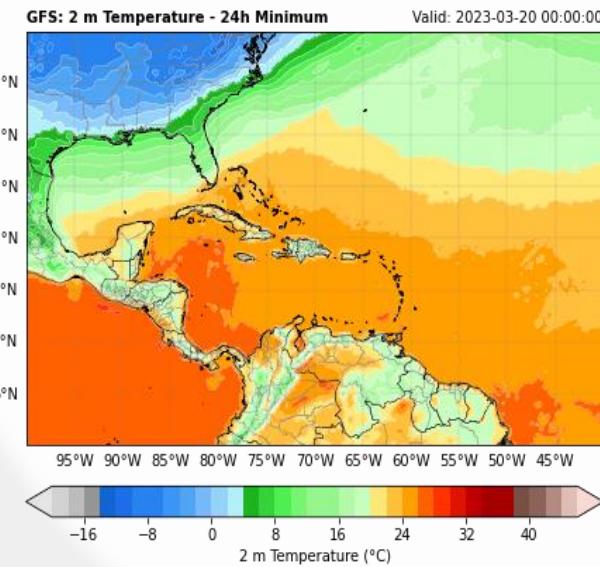
IMAGE GENERATED BY

Explaining the new instructions:

```
169 #-----
170 # Define the image extent
171 axs[2].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
172 # Add a shapefile
173 # https://geoftp.ibge.gov.br/organizacao\_do\_territorio/malhas\_teritoriais/malhas\_municipais/municipio\_2019/Brasil/BR/br\_unidades\_da\_federacao.zip
174 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
175 axs[2].add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
176
177 # Add coastlines, borders and gridlines
178 axs[2].coastlines(resolution='10m', color='black', linewidth=0.8)
179 axs[2].add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
180 gl = axs[2].gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
181 gl.left_labels = False
182 gl.right_labels = False
183
184 # Define de contour interval
185 data_min = -20
186 data_max = 48
187 interval = 2
188 levels = np.arange(data_min,data_max,interval)
189
190 # Plot the contours
191 img3 = axs[2].contourf(lons, lats, tmtmp_avg, cmap=cmap, levels=levels, extend='both')
192 img4 = axs[2].contour(lons, lats, tmtmp_avg, colors='white', linewidths=0.3, levels=levels)
193
194 # Add a colorbar
195 plt.colorbar(img3, label='2 m Temperature (°C)', orientation='horizontal', pad=0.02, fraction=0.05, ax=axs[2])
196
197 # Add a title
198 axs[2].set_title('GFS: 2 m Temperature - 24h Average', fontweight='bold', fontsize=6, loc='left')
199 axs[2].set_title('Valid: ' + valid, fontsize=6, loc='right')
200
201 # Save the image
202 plt.savefig('image_ll.png', bbox_inches='tight', pad_inches=0, dpi=100)
203
204 # Show the image
205 plt.show()
```

PLOT 3 (AVERAGE)

IMAGE GENERATION



Script 12: Precipitation and Accumulated Precipitation

```

LIBRARIES
import pygrib
import matplotlib.pyplot as plt
import cartopy, cartopy.crs as ccrs
import cartopy.io.shapereader as shapereader
import numpy as np
import matplotlib
# Provides a high-level interface to the ECW/GIF/GRIB/GTS files
# Plotting library
# Plot maps
# Import shapefiles
# Scientific computing with Python
# Comprehensive library for creating static, animated, and interactive plots
# Select the extent [min. lon, min. lat, max. lon, max. lat]
extent = [-93.0, -60.0, -25.0, 18.0]
# Open the GRIB file
grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f024")
#-----#
# Read the instant precip
grb = grib.select(name='Precipitation rate', typeOfLevel = 'surface')[0]
# Get information from the file
init = str(grb.analDate) # Init date / time
run = str(grb.hour).zfill(2) # Run
ftime = str(grb.forecastTime) # Forecast hour
valid = str(grb.validDate) # Valid date / time
print('Init: ' + init + ' UTC')
print('Run: ' + run + 'Z')
print('Forecast: ' + ftime)
print('Valid: ' + valid + ' UTC')
#-----#
# Read the data for a specific region
precip, lats, lons = grb.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)
# Convert from kg m**-2 s**-1 to mm/h
precip = precip * 60 * 60
# Smooth the contours
import scipy.ndimage
precip = scipy.ndimage.zoom(precip, 3)
lats = scipy.ndimage.zoom(lats, 3)
lons = scipy.ndimage.zoom(lons, 3)
# Remove values lower than 0.1
precip[precip < 0.1] = np.nan

```

LIBRARIES

DATA READING AND MANIPULATION (DATA 1)

DATA READ. AND MANIP. (DATA 2)

GENERAL PLOT CONF.

PLOT DATA 1

PLOT DATA 2

IMAGE GENERATION

LIBRARIES

DATA READING AND MANIPULATION (PREC.)

“Not a Number”(does not show in the plot)

All pixels < 0.1 mm, convert to NaN

```
51 #
52 # Select the variable
53 totpr = grib.select(name='Total Precipitation', typeOfLevel = 'surface')[1]
54
55 # Read the data for a specific region
56 totpr = totpr.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]
57
58 # Smooth the contours
59 totpr = scipy.ndimage.zoom(totpr, 3)
60
61 # Remove values lower than 1.0
62 totpr[totpr < 1.0] = np.nan
63
64 "Not a Number"
65 All pixels < 0.1 mm, convert to NaN
66 (does not show in the plot)
67
68 # Create a custom color palette
69 colors = ["#b4f0f0", "#96d2fa", "#78b9fa", "#3c95f5", "#1e6deb", "#148cd2",
70 "#0fa00f", "#28be28", "#50f050", "#72f06e", "#b3faaa", "#ffff9aa",
71 "#ffe978", "#fffc13c", "#ffa200", "#ff6200", "#ff3300", "#ff1500",
72 "#c00100", "#a50200", "#870000", "#653b32"]
73 cmap = matplotlib.colors.ListedColormap(colors)
74 cmap.set_over('#000000')
75 cmap.set_under('#ffffff')
76
77 # Choose the plot size (width x height, in inches)
78 fig, axs = plt.subplots(1,2, figsize=(10,5), sharex = False, sharey = False, subplot_kw=dict(projection=ccrs.PlateCarree())) # 1 row x 2 columns
79
80 #
81
82 # Define the image extent
83 axs[0].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
84
85 # Add a background image
86 import cartopy.feature as cfeature
87 land = axs[0].add_feature(cfeature.LAND, facecolor='whitesmoke')
88 ocean = axs[0].add_feature(cfeature.OCEAN, facecolor='white')
```

DATA READING AND MANIPULATION (ACCUMULATED PRECIP.)

GENERAL PLOT CONFIGURATION

1 row x 2 columns plot

Cartopy Projection

Script 12: Precipitation and Accumulated Precipitation

```
81
82
83 # Define the image extent
84 axs[0].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
85
86 # Add a background image
87 import cartopy.feature as cfeature
88 land = axs[0].add_feature(cfeature.LAND, facecolor='whitesmoke')
89 ocean = axs[0].add_feature(cfeature.OCEAN, facecolor='white')
90
91 # Add a shapefile
92 # https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_teritoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
93 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
94 axs[0].add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
95
96 # Add coastlines, borders and gridlines
97 axs[0].coastlines(resolution='10m', color='black', linewidth=0.8)
98 axs[0].add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
99 gl = axs[0].gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
100 gl.top_labels = False
101 gl.right_labels = False
102
103 # Define de contour interval
104 data_min = 0.1
105 data_max = 50
106 interval = 1
107 levels = np.arange(data_min,data_max + interval,interval)
108
109 # Plot the contours
110 img1 = axs[0].contourf(lons, lats, precip, cmap=cmap, levels=levels, extend='max')
111 img2 = axs[0].contour(lons, lats, precip, colors='white', linewidths=0.3, levels=levels)
112
113 # Define the ticks to be shown
114 ticks = [0.1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50] — Values that will appear in the
115
116 # Add a colorbar
117 plt.colorbar(img1, label='Instant Precipitation Rate (mm/h)', orientation='horizontal', pad=0.02, fraction=0.05, ticks=ticks, ax=axs[0])
118
119 # Set title
120 axs[0].set_title('GFS: Instant Precipitation Rate (mm/h)', fontweight='bold', fontsize=10, loc='left')
121 axs[0].set_title('Valid: ' + valid, fontsize=10, loc='right')
122
123
```

PLOT DATA 1 (PRECIP.)

Add Surface and
Ocean with
Specific Colors

Values that will
appear in the
colorbar

Script 12: Precipitation and Accumulated Precipitation

```
123 #  
124 # Define the image extent  
125 axs[1].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())  
126 # Add a background image  
127 import cartopy.feature as cfeature  
128 land = axs[1].add_feature(cfeature.LAND, facecolor='whitesmoke')  
129 ocean = axs[1].add_feature(cfeature.OCEAN, facecolor='white')  
130 # Add a shapefile  
131 # https://ceoftp.ibge.gov.br/organizacao\_digital/territorio/malhas\_territoriais/malhas\_municipais/municipio\_2019/Brasil/BR\_br\_unidades\_da\_federacao.zip  
132 shapefile = list(shapereader.Reader('BR_UF_2019.shp').geometries())  
133 axs[1].add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)  
134 # Add coastlines, borders and gridlines  
135 axs[1].coastlines(resolution='10m', color='black', linewidth=0.8)  
136 axs[1].add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)  
137 gl = axs[1].gridlines(crs=ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)  
138 gl.top_labels = False  
139 gl.right_labels = False  
140 # Define de contour interval  
141 data_min = 0.1  
142 data_max = 100  
143 interval = 5  
144 levels = np.arange(data_min,data_max + interval,interval)  
145 # Plot the contours (high values)  
146 img3 = axs[1].contourf(lons, lats, totpr, cmap=cmap, levels=levels, extend='max')  
147 img4 = axs[1].contour(lons, lats, totpr, colors='white', linewidths=0.3, levels=levels)  
148 # Define the ticks to be shown  
149 ticks = [1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]  
150 # Add a colorbar  
151 plt.colorbar(img3, label='Total Precipitation (mm - 24h)', orientation='horizontal', pad=0.02, fraction=0.05, ticks=ticks, ax=axs[1])  
152 # Set title  
153 axs[1].set_title('GFS: Total Precipitation (mm - 24h)', fontweight='bold', fontsize=10, loc='left')  
154 axs[1].set_title('Valid: ' + valid, fontsize=10, loc='right')  
155 # Save the image  
156 plt.savefig('image_12.png')  
157 # Show the image  
158 plt.show()
```

Add Surface and
Ocean with
Specific Colors

PLOT DATA 2
(ACC. PRECIP.)

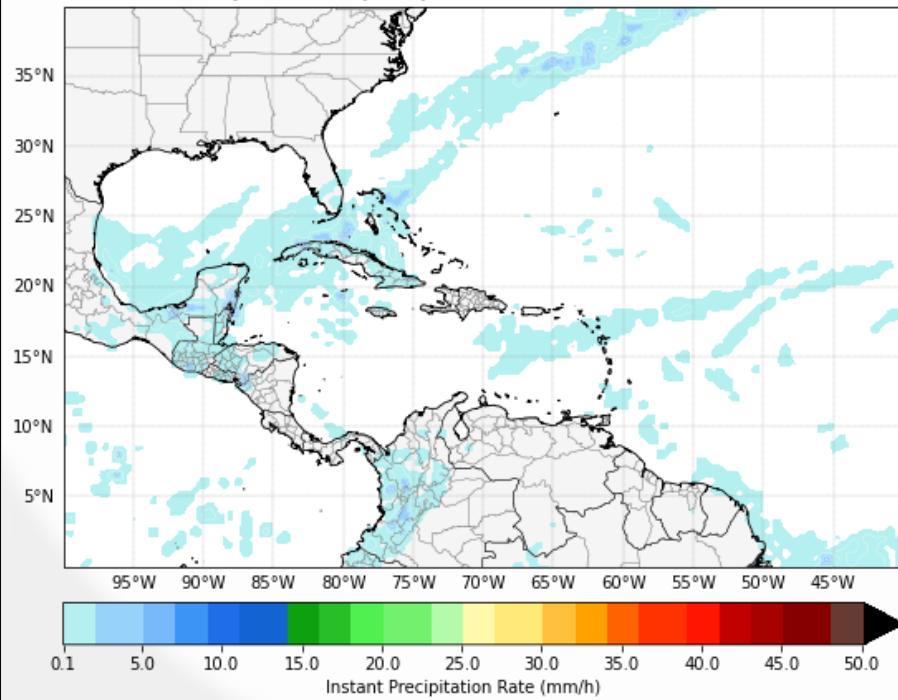
Values that will
appear in the
colorbar

IMAGE GENERATION

Script 12: Precipitation and Accumulated Precipitation

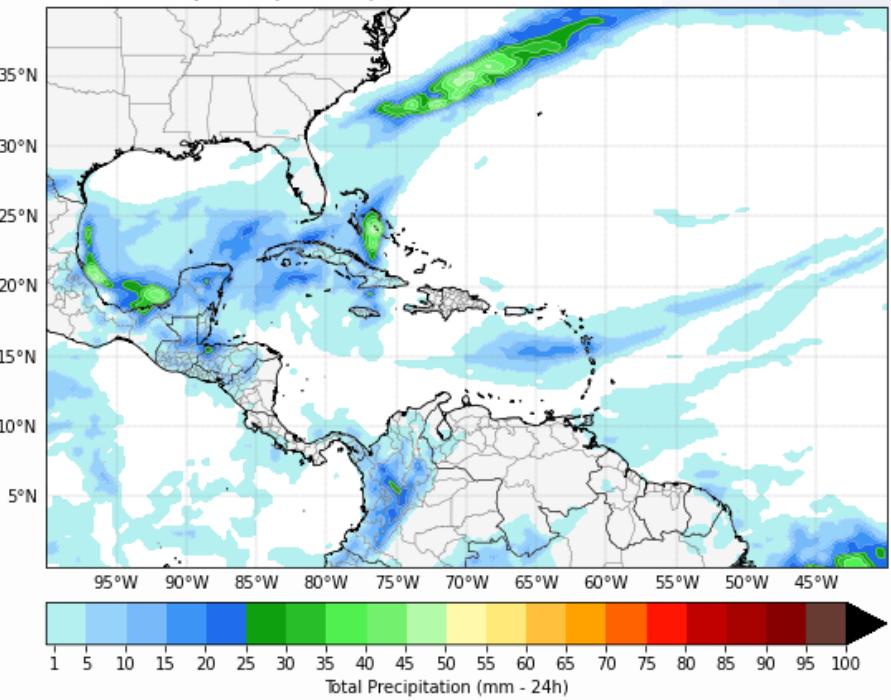
GFS: Instant Precipitation Rate (mm/h)

Valid: 2023-03-20 00:00:00



GFS: Total Precipitation (mm - 24h)

Valid: 2023-03-20 00:00:00





Part 2: More Advanced Concepts

PART I

Knowing the Available Variables

Basic Plot / Pixel Values

Metadata, Basic Calculus, Color Palette, Legend and Title

Overlaying Maps with Cartopy

Reading a Shapefile

Plotting Contours and Labels

Custom Color Palettes

Smoothing Contours

Working with Multiple Files

Animations

PART II

Downloading Data using Scripts

Averages, Minimums and Maximums

Multiplots

Reading Fields Specifying Levels - Streamlines

Wind Vectors

Barbs

Reading Multiple Fields - Galvez Davison Index

Satellite + NWP

METAR + Satellite + NWP



Script: Downloading Data From NOMADS

```

1  # INPE / CPTEC - Training: NWP Data Processing With Python - NWP Download with Python (GFS)
2  # Author: Diego Sozzi
3
4
5  # Required modules
6  from datetime import datetime    # Basic Dates and time types
7  import os                       # Standard library for managing system interfaces
8  import requests                 # HTTP library for Python
9  import time as t                # Time access and conversion
10
11
12  print('-----')
13  print('GFS Download (NOMADS) - Script started.')
14  print('-----')
15
16  # Start the time counter
17  start_time = t.time()
18
19
20
21  # Define local directory
22  dir = "GFSplots"; os.makedirs(dir, exist_ok=True)
23
24  # Desired date (last 10 days only) (format: -YYYYMMDD)
25  date = datetime.today().strftime("%Y%m%d")
26
27
28  # Desired metrics
29  max_lon = -53.00
30  max_lat = +25.00
31  min_lat = +60.00
32  max_lat = +18.00
33
34
35  # Desired resolutions: '25' or '50' or '1'
36  resolution = '50'
37
38  # Desired run: '00' or '06' or '12' or '18'
39  hour_run = '00'
40
41  # Desired forecast hours
42  hour_init = 0 # Init time
43  hour_end = 24 # End time
44  hour_int = 3 # Interval
45
46
47  # URL
48  # https://nomads.noaa.gov/data/
49
50
51  def download_gfs(date, url):
52
53      # Create the URL's based on the resolution
54      if (resolution == '25'):
55          url = "https://nomads.noaa.gov/data/gfs-bin/timer.gph?resolution=25&forecast=0&date=%s" % date
56          file_name = "%s-%s-%s.gph25" % (date[0:4], date[4:6], date[6:8])
57
58      elif (resolution == '50'):
59          url = "https://nomads.noaa.gov/data/gfs-bin/timer.gph?resolution=50&forecast=0&date=%s" % date
60          file_name = "%s-%s-%s.gph50" % (date[0:4], date[4:6], date[6:8])
61
62      elif (resolution == '1'):
63          url = "https://nomads.noaa.gov/data/gfs-bin/timer.gph?resolution=1&forecast=0&date=%s" % date
64          file_name = "%s-%s-%s.gph1" % (date[0:4], date[4:6], date[6:8])
65
66      # Print the file name
67      print("File name: %s" % file_name)
68
69      # Open the file in the specified url
70      myfile = requests.get(url)
71
72      # Download the file
73      open(file_name, 'wb').write(myfile.content)
74
75
76  for hour in range(hour_init, hour_end + 1, hour_int):
77
78      print('Downloading GFS file')
79      print('-----')
80      print('Forecast hour: %s' % hour)
81      print('Resolution: %s' % resolution)
82      print('Run: %s' % date)
83      print('Forecast hour: %s %s' % (date, str(hour).zfill(3)))
84
85      download_gfs(date, hour)
86
87
88  # End the time counter
89  print("\nTotal Processing Time: %s seconds." % round((t.time() - start_time), 2))

```

LIBRARIES

DOWNLOAD CONFIGURATION

DOWNLOAD FUNCTION

DOWNLOAD FUNCTION CALL LOOP

Script: Downloading Data From NOMADS

```
1  # INPE / CPTEC - Training: NWP Data Processing With Python - NWP Download with Python (GFS)
2  # Author: Diego Souza
3  #
4  #
5  # Required modules
6  from datetime import datetime          # Basic Dates and time types
7  import os                            # Miscellaneous operating system interfaces
8  import requests                      # HTTP library for Python
9  import time as t                     # Time access and conversion
10 #
11 #
12 print('-----')
13 print('GFS Download (NOMADS) - Script started.')
14 print('-----')
15 #
16 #
17 # Start the time counter
18 start_time = t.time()
19 #
20 #
21 # Download directory
22 dir = "Samples"; os.makedirs(dir, exist_ok=True)
23 #
24 #
25 # Desired date (last 10 days only!): Format - 'YYYYMMDD'
26 date = datetime.today().strftime('%Y%m%d')
27 #
28 # Desired extent
29 min_lon = '-93.00'
30 max_lon = '-25.00'
31 min_lat = '-60.00'
32 max_lat = '18.00'
33 #
34 # Desired resolution: '25' or '50' or 'l'
35 resolution = '50'
36 #
37 # Desired run: '00' or '06' or '12' or '18'
38 hour_run = '00'
39 #
40 # Desired forecast hours
41 hour_ini = 0 # Init time
42 hour_end = 24 # End time
43 hour_int = 3 # Interval
44 #
45 # -----
```

LIBRARIES

DOWNLOAD CONFIGURATION

Script: Downloading Data From NOMADS

```
47 # Link (select "grib filter" and check "Show the URL only for web programming" to verify the URL's):
48 # https://nomads.ncep.noaa.gov/
49
50 def download_gfs(date, iii):
51
52     # Create the URL's based on the resolution
53     if (resolution == '25'):
54         url = 'https://nomads.ncep.noaa.gov/cgi-bin/filter_gfs_0p'+resolution+'.pl?file=gfs.t'+hour_run+'z.pgrb2.0p'+resolution+'.f'+str(hour)
55         file_name = 'gfs.t'+hour_run+'z.pgrb2.0p'+resolution+'.f'+str(hour).zfill(3)
56     elif(resolution == '50'):
57         url = 'https://nomads.ncep.noaa.gov/cgi-bin/filter_gfs_0p'+resolution+'.pl?file=gfs.t'+hour_run+'z.pgrb2full.0p'+resolution+'.f'+str(hour)
58         file_name = 'gfs.t'+hour_run+'z.pgrb2.0p'+resolution+'.f'+str(hour).zfill(3)
59     elif (resolution == '1'):
60         url = 'https://nomads.ncep.noaa.gov/cgi-bin/filter_gfs_0p'+resolution+'p00.pl?file=gfs.t'+hour_run+'z.pgrb2.'+resolution+'p00.f'+str(hour)
61         file_name = 'gfs.t'+hour_run+'z.pgrb2.'+resolution+'p00.f'+str(hour).zfill(3)
62
63     # Print the file name
64     print("File name: ", file_name)
65     # Sends a GET request to the specified url
66     myfile = requests.get(url)
67
68     # Download the file
69     open(dir + '//' + file_name, 'wb').write(myfile.content)
70
71
72
73 # Download loop
74 for hour in range(hour_ini, hour_end + 1, hour_int):
75     print('-----')
76     print('Downloading GFS File:')
77     print('-----')
78     print('Resolution: ' + resolution)
79     print('Date: ' + date)
80     print('Run: ' + hour_run)
81     print('Forecast Hour: f' + str(hour).zfill(3))
82     # Call the download function
83     download_gfs(date,hour)
84
85 #-----
86
87 # End the time counter
88 print('\nTotal Processing Time:', round((t.time() - start_time),2), 'seconds.')
```

DOWNLOAD FUNCTION

DOWNLOAD FUNCTION CALL LOOP



Script 13: Reading Fields by Level - Streamlines

```
import pygrib           # Provides a high level interface to the ECMWF EC2000C C library for reading GRIB files
import matplotlib.pyplot as plt   # Plotting library
import numpy as np        # Numerical library
import cartopy.crs as ccrs # Import cartopy
import cartopy.io.shapereader as shapereader # Import shapefiles
import shapely.geometry # Import geometry for working with Python
import matplotlib      # Comprehensive library for creating static, animated, and interactive visualizations in Python

# constants
lat0 = 10.0; lon0 = -60.0; lat1 = 20.0; lon1 = 10.0
```

LIBRARIES

DATA READING AND MANIPULATION

```
plt.figure(figsize=(6,8))
# Create a map with a simplified projection in cartopy
ax = plt.axes(projection=PlateCarree())
# Define the map extent
map_extent = [extent[0], extent[2], extent[1], extent[3]]
# Set extent
ax.set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
# Add a grid
ax.grid()
# Plot the boundaries for the territories in Mexico
shapefile = fiona.open('C:\Users\user\Downloads\TerritoriosMexico\TerritoriosMexico\mexico.shp')
ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor="gray", facecolor="None", linewidth=0.5)
# Plot the coastlines
ax.coastlines(resolution='50m', color='black', linewidth=0.8)
# Add features to the map
ax.add_feature(cartopy.feature.OCEAN, edgecolor='black', linewidth=0.5)
```

PLOT CONFIGURATION

```

g1.right_label = False
g1.left_label = False

# Create the contour intervals
data_min = -60
data_max = 60
interval = 5
levels = np.arange(data_min,data_max,interval)

cstors = ["#0000ff", "#00ffff", "#000000", "#008000", "#008080", "#00ffff", "#00ffff", "#00ffff", "#00ffff", "#00ffff", "#00ffff", "#00ffff"]
cmapp = plt.get_cmap('hsv', len(levels)-1)
cmapp.set_over(cstors[-1])
cmapp.set_under(cstors[0])

# Plot the contours
img1 = ax1.contourf(lons, lats, wv, cmapp, levels=levels, extend='both')
img1.set_label("WV")
img1.set_colorbar_label("WV (dBZ)", rotation=90, horizontalalignment="center", verticalalignment="bottom", labelsize=10, fontweight="bold", color="black")

# Plot the streamlines
from matplotlib.patches import Arrow
img2 = Arrow(ax2, wv, lons, lats, u, v, width=1, color="gray", transform=ax2.transPlateCarree())
img2.set_label("Streamlines")
img2.set_colorbar_label("Streamlines", rotation=90, horizontalalignment="center", verticalalignment="bottom", labelsize=10, fontweight="bold", color="black")

# Plot the counterflow
g1.set_counterflow(img3, label="Counterflow", orientation="vertical", pad=0.05, fraction=0.05)
img3.set_label("Counterflow")
img3.set_colorbar_label("Counterflow", rotation=90, horizontalalignment="center", verticalalignment="bottom", labelsize=10, fontweight="bold", color="black")

# Save the image
plt.savefig("image1.png", bbox_inches='tight', pad_inches=0, dpi=100)
plt.show()

```

IMAGE GENERATION

Explaining the new instructions: Fields by Levels - Streamlines

LIBRARIES

DATA READING AND MANIPULATION

Selecting variables specifying the level

INPE / CPTEC Training: Fields by Levels - Streamlines

Author: Diego Souza

```
port pygrib           # Provides a high-level interface to the ECWRF ECCODES C library f
port matplotlib.pyplot as plt    # Plotting library
port cartopy, cartopy.crs as ccrs # Plot maps
port cartopy.io.shapereader as shpreader # Import shapefiles
port numpy as np          # Scientific computing with Python
port matplotlib          # Comprehensive library for creating static, animated, and interact

LIBRARIES
```

Select the extent [min. lon, min. lat, max. lon, max. lat]
extent = [-93.0, -60.00, -25.00, 18.00]

Open the GRIB file
ib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")

DATA READING AND MANIPULATION

Select the variable
comp = grib.select(name='U component of wind')

```
typeOfLevel = 'isobaricInhPa', level = 250]()
```

```
Get information from the file
it = str(ucomp.analDate)           # Init date / time
n = str(ucomp.hour).zfill(2)       # Run
ime = str(ucomp.forecastTime)      # Forecast hour
ldt = str(ucomp.validDate)         # Valid date / time
int('Init: ' + init + ' UTC')
int('Run: ' + run + 'Z')
int('Forecast: ' + ftime)
int('Valid: ' + valid + ' UTC')
```

Selecting variables specifying the level

```
Read the data for a specific region
comp, lats, lons = uncomp.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)

Select the variable
comp = grib.select(name='V component of wind', typeOfLevel = 'isobaricInhPa', level = 250)[0]

Read the data for a specific region
comp = vcomp.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]
```

```
Calculate the wind speed  
= np.sqrt(ucomp**2 + vcomp**2) — Calculating wind speed
```

```
Choose the plot size (width x height, in inches)  
t.figure(figsize=(8,8))
```

Calculating wind speed

Script 13: Reading Fields by Level - Streamlines

variables - Bloco de Notas

Remembering the “variables.txt” file

Arquivo Editar Formatar Exibir Ajuda

```
264:Ice water mixing ratio:kg kg**-1 (instant):regular_ll:isobaricInhPa:level 20000 Pa:fcst time 0 hrs:from 202107020000
265:Rain mixing ratio:kg kg**-1 (instant):regular_ll:isobaricInhPa:level 20000 Pa:fcst time 0 hrs:from 202107020000
266:Snow mixing ratio:kg kg**-1 (instant):regular_ll:isobaricInhPa:level 20000 Pa:fcst time 0 hrs:from 202107020000
267:Graupel (snow pellets):kg kg**-1 (instant):regular_ll:isobaricInhPa:level 20000 Pa:fcst time 0 hrs:from 202107020000
268:Ozone mixing ratio:kg kg**-1 (instant):regular_ll:isobaricInhPa:level 20000 Pa:fcst time 0 hrs:from 202107020000
269:Geopotential Height:gpm (instant):regular_ll:isobaricInhPa:level 25000 Pa:fcst time 0 hrs:from 202107020000
270:Temperature:K (instant):regular_ll:isobaricInhPa:level 25000 Pa:fcst time 0 hrs:from 202107020000
271:Relative humidity:% (instant):regular_ll:isobaricInhPa:level 25000 Pa:fcst time 0 hrs:from 202107020000
272:Total Cloud Cover:% (instant):regular_ll:isobaricInhPa:level 25000 Pa:fcst time 0 hrs:from 202107020000
273:Specific humidity:kg kg**-1 (instant):regular_ll:isobaricInhPa:level 25000 Pa:fcst time 0 hrs:from 202107020000
274:Vertical velocity:Pa s**-1 (instant):regular_ll:isobaricInhPa:level 25000 Pa:fcst time 0 hrs:from 202107020000
275:Geometric vertical velocity:m s**-1 (instant):regular_ll:isobaricInhPa:level 25000 Pa:fcst time 0 hrs:from 202107020000
276:U component of wind:m s**-1 (instant):regular_ll:isobaricInhPa:level 25000 Pa:fcst time 0 hrs:from 202107020000
277:V component of wind:m s**-1 (instant):regular_ll:isobaricInhPa:level 25000 Pa:fcst time 0 hrs:from 202107020000
278:Absolute vorticity:s**-1 (instant):regular_ll:isobaricInhPa:level 25000 Pa:fcst time 0 hrs:from 202107020000
279:Cloud mixing ratio:kg kg**-1 (instant):regular_ll:isobaricInhPa:level 25000 Pa:fcst time 0 hrs:from 202107020000
280:Ice water mixing ratio:kg kg**-1 (instant):regular_ll:isobaricInhPa:level 25000 Pa:fcst time 0 hrs:from 202107020000
2 21      # Select the variable
2 22      ucomp = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 250)[0]
2 23
284:Ozone mixing ratio:kg kg**-1 (instant):regular_ll:isobaricInhPa:level 25000 Pa:fcst time 0 hrs:from 202107020000
285:Geopotential Height:gpm (instant):regular_ll:isobaricInhPa:level 30000 Pa:fcst time 0 hrs:from 202107020000
286:Temperature:K (instant):regular_ll:isobaricInhPa:level 30000 Pa:fcst time 0 hrs:from 202107020000
287:Relative humidity:% (instant):regular_ll:isobaricInhPa:level 30000 Pa:fcst time 0 hrs:from 202107020000
```

Ln 276, Col 17 100% Windows (CRLF) UTF-8

Script 13: Reading Fields by Level - Streamlines

```

50
51 # Choose the plot size (width x height, in inches)
52 plt.figure(figsize=(10,6))
53
54 # Use the Cylindrical Equidistant projection in cartopy
55 ax = plt.axes(projection=ccrs.PlateCarree(central_longitude=extent[0] - extent[2]))
56
57 # Define the image extent
58 img_extent = [extent[0], extent[2], extent[1], extent[3]]
59 ax.set_extent([extent[0], extent[2], extent[1], extent[3]], crs=ccrs.PlateCarree())
60
61 # Define de contour interval
62 data_min = 0
63 data_max = 60
64 interval = 5
65 levels = np.arange(data_min,data_max,interval)
66
67 # Create a custom color palette
68 colors = ["#e7f2f4", "#cceace", "#b6e2e8", "#abdcff", "#a4cd685", "#9cd04e",
69     "#abcf2a", "#fc9d2b", "#e8d5c", "#ffd100", "#ffba00", "#ffa200"]
70 cmap = matplotlib.colors.ListedColormap(colors)
71 cmap.set_over('#fff8c00')
72 cmap.set_under('#ffffafa')
73
74 # Plot the contours
75 img1 = ax.contourf(lons, lats, ws, cmap=cmap, levels=levels, extend='both', transform=ccrs.PlateCarree())
76 img2 = ax.contour(lons, lats, ws, colors='white', linewidths=0.3, levels=levels, transform=ccrs.PlateCarree())
77 ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10', fmt = '%1.0f', colors= 'black')
78
79 # Plot the streamlines
80 img3 = ax.streamplot(lons, lats, ucomp, vcomp, density=[4, 4], linewidth=1, color='gray', transform=ccrs.PlateCarree())
81
82 # Add a shapefile
83 shapefile = list(shpreader.Reader('ne_10m_admin_1_states_provinces.shp').geometries())
84 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
85
86 # Add coastlines, borders and gridlines
87 ax.coastlines(resolution='10m', color='black', linewidth=0.8)
88 ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
89 gl = ax.gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
90 gl.top_labels = False
91 gl.right_labels = False
92
93 # Add a colorbar
94 plt.colorbar(img1, label='Isotachs (kt)', orientation='horizontal', pad=0.05, fraction=0.05)
95
96 # Add a title
97 plt.title('GFS: Streamlines and Isotachs (250 hPa)', fontweight='bold', fontsize=10, loc='left')
98 plt.title('Valid: ' + valid, fontsize=10, loc='right')
99
100 # Save the image
101 plt.savefig('image_13.png', bbox_inches='tight', pad_inches=0, dpi=100)
102
103 # Show the image
104 plt.show()

```

PLOT CONFIGURATION

Lats and Lons Density

Plot the windspeed

U and V components

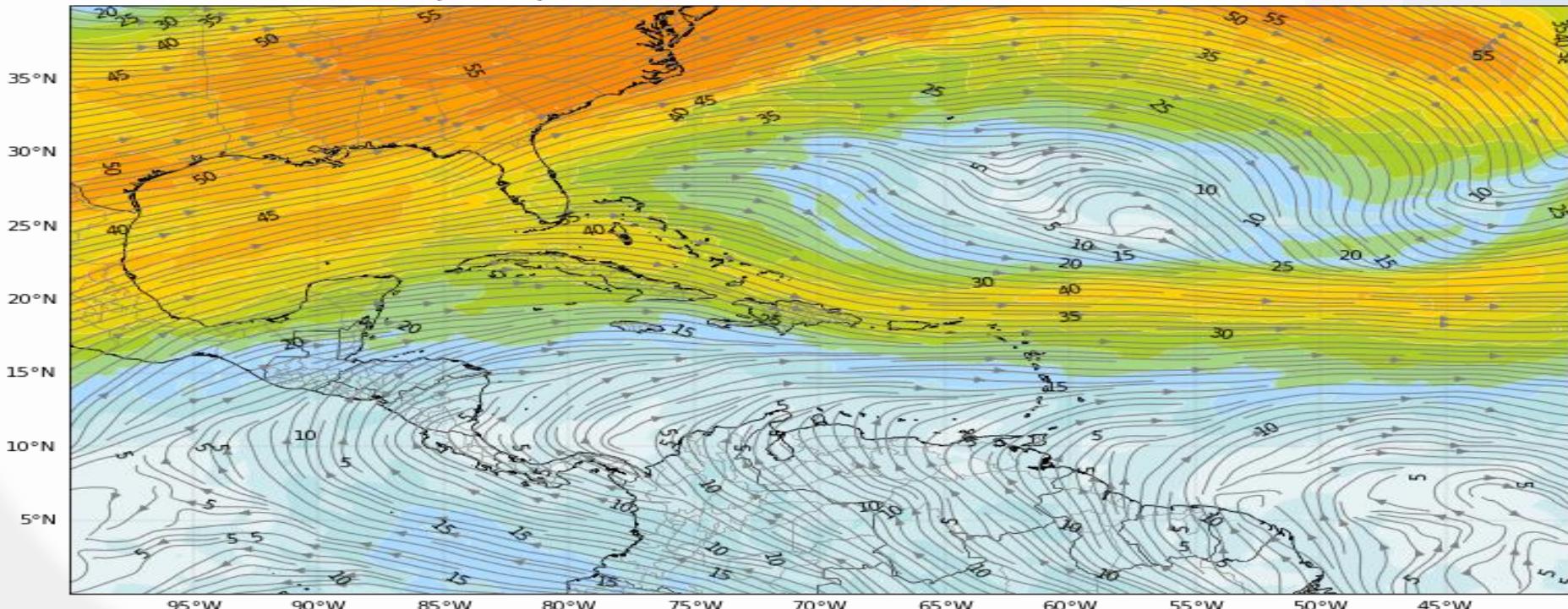
Plot the streamlines

IMAGE GENERATION

Script 13: Reading Fields by Level - Streamlines

GFS: Streamlines and Isotachs (250 hPa)

Valid: 2023-03-19 00:00:00



```
78  
79 # Plot the streamlines  
80 img3 = ax.streamplot(lons, lats, ucomp, vcomp, density=[4, 4], linewidth=1, color='gray', transform=ccrs.PlateCarree())  
81
```

Some Quick Modifications

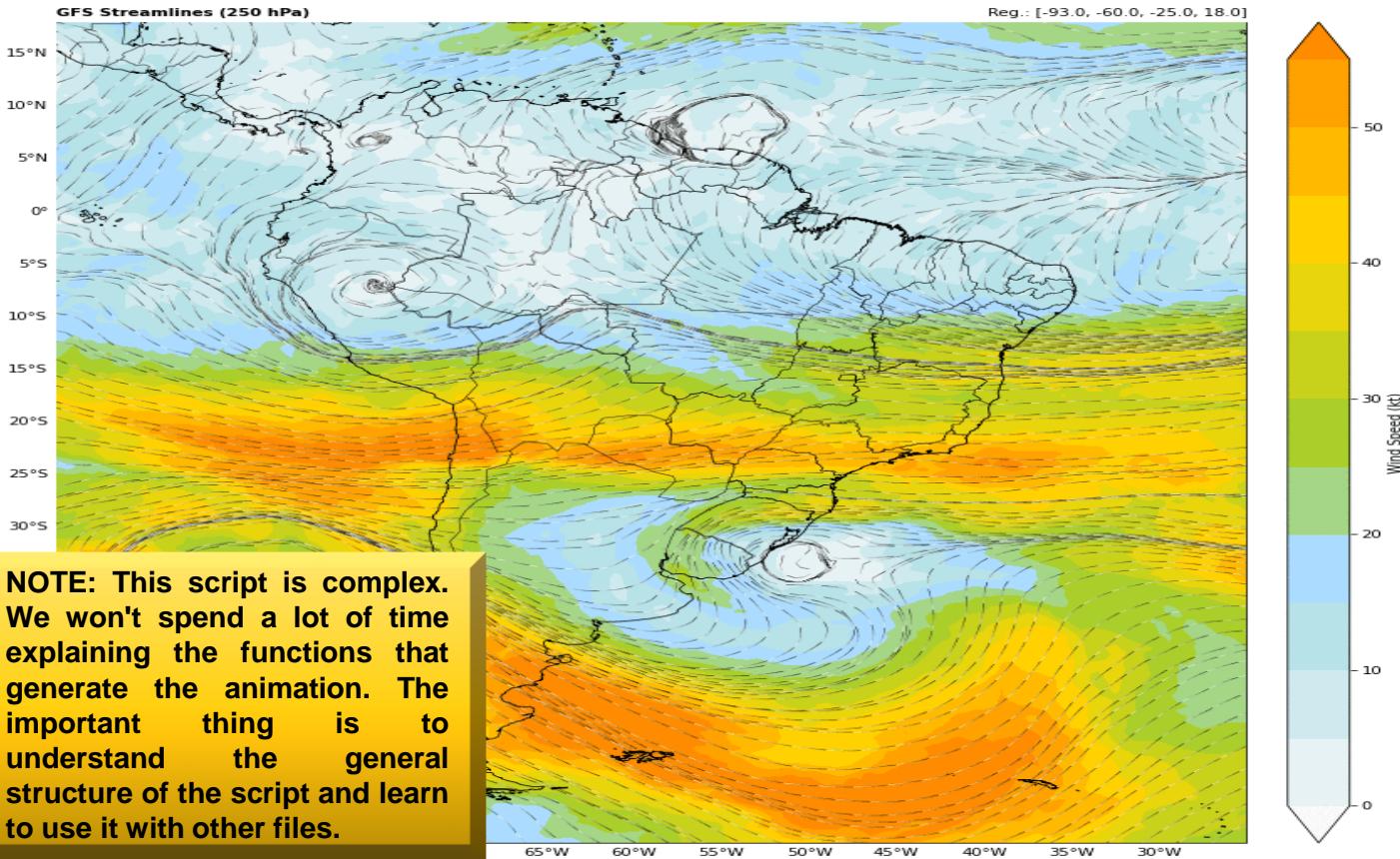
- Change the appearance of the streamlines (density, colors and colormaps)



Script 14: Animating Streamlines

A LOT OF
CODE

NOTE: This script is complex. We won't spend a lot of time explaining the functions that generate the animation. The important thing is to understand the general structure of the script and learn to use it with other files.



Script 14: Animating Streamlines

LIBRARIES

FUNCTIONS

```
180
181
182 # Select the extent [min. lon, min. lat, max. lon, max. lat]
183 extent = [-93.0, -60.0, -25.0, 18.0]
184
185 # Open the GRIB file
186 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
187
188 #
189
190 # Select the variable
191 ucomp = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 200)[0]
192
193 # Get information from the file
194 init = str(ucomp.analDate)           # Init date / time
195 run = str(ucomp.hour).zfill(2)       # Run
196 ftime = str(ucomp.forecastTime)      # Forecast hour
197 valid = str(ucomp.validDate)         # Valid date / time
198 print('Init: ' + init + ' UTC')
199 print('Run: ' + run + 'Z')
200 print('Forecast: ' + ftime)
201 print('Valid: ' + valid + ' UTC')
202
203 # Read the data for a specific region
204 ucomp, lats, lons = ucomp.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)
205
206 #
207
208 # Select the variable
209 vcomp = grib.select(name='V component of wind', typeOfLevel = 'isobaricInhPa', level=200)[0][0]
210
211 # Read the data for a specific region
212 vcomp = vcomp.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]
213
214 #
215
216 # Calculate the wind speed
217 ws = np.sqrt(ucomp**2 + vcomp**2)
218
219 # Correct the longitudes from 0 - 360
220 lons_corr = lons - 360
221
222 # Calculate the components
223 Y, X = lats, lons_corr
224 U, V = ucomp, vcomp
225
226 A
227
228 # Choose the plot size (width x height, in inches)
229 fig = plt.figure(figsize=(8,8))
```

DATA READING AND MANIPULATION

Explaining the new instructions:



Variables needed for functions

Script 14: Animating Streamlines

```
225
226
227
228 # Choose the plot size (width x height, in inches)
229 fig = plt.figure(figsize=(8,8))
230
231 # Use the Cylindrical Equidistant projection in cartopy
232 ax = fig.add_subplot(1, 1, 1,projection=ccrs.PlateCarree())
233 ax.set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
234
235 # Define the image extent
236 img_extent = [extent[0], extent[2], extent[1], extent[3]]
237
238 # Define de contour interval
239 data_min = 0
240 data_max = 60
241 interval = 5
242 levels = np.arange(data_min,data_max,interval)
243
244 # Create a custom color palette
245 colors = ["#e72f4", "#ceeaee", "#b6e2e8", "#abdcff", "#a4d685", "#9cd04e", "#abcf2a", "#c9d21b", "#e8d50c", "#ffd100", "#ffba00", "#ffa200"]
246 cmap = matplotlib.colors.ListedColormap(colors)
247 cmap.set_over('#ff8c00')
248 cmap.set_under('#ffffafa')
249
250 # Plot the contours
251 img1 = ax.contourf(lons, lats, ws, cmap=cmap, levels=levels, extend='both', alpha=1.0, zorder = 10)
252
253 # Add a shapefile
254 # https://geoftp.ibge.gov.br/organizacao\_do\_territorio/malhas\_territoriais/malhas\_municipais/municipio\_2019/Brasil/BR/br\_unidades\_da\_federacao.zip
255 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
256 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='black', facecolor='none', linewidth=0.3, zorder=14)
257
258 # Add coastlines, borders and gridlines
259 ax.coastlines(resolution='10m', color='black', linewidth=0.8, zorder=15)
260 ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5, zorder=16)
261 gl = ax.gridlines(ccrs.PlateCarree(), color='white', alpha=1.0, linestyle='--', linewidth=0.5, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
262 gl.top_labels = False
263 gl.right_labels = False
264
265 # Add a colorbar
266 plt.colorbar(img1, label='Wind Speed (kt)', orientation='vertical', pad=0.03, fraction=0.05)
267
268 # Add a title
269 plt.title('GFS: Wind Speed (kt) & Direction (250 hPa) - Animation', fontweight='bold', fontsize=8, loc='left')
270 plt.title('Reg.: ' + str(extent) , fontsize=8, loc='right')
271
272
273
```

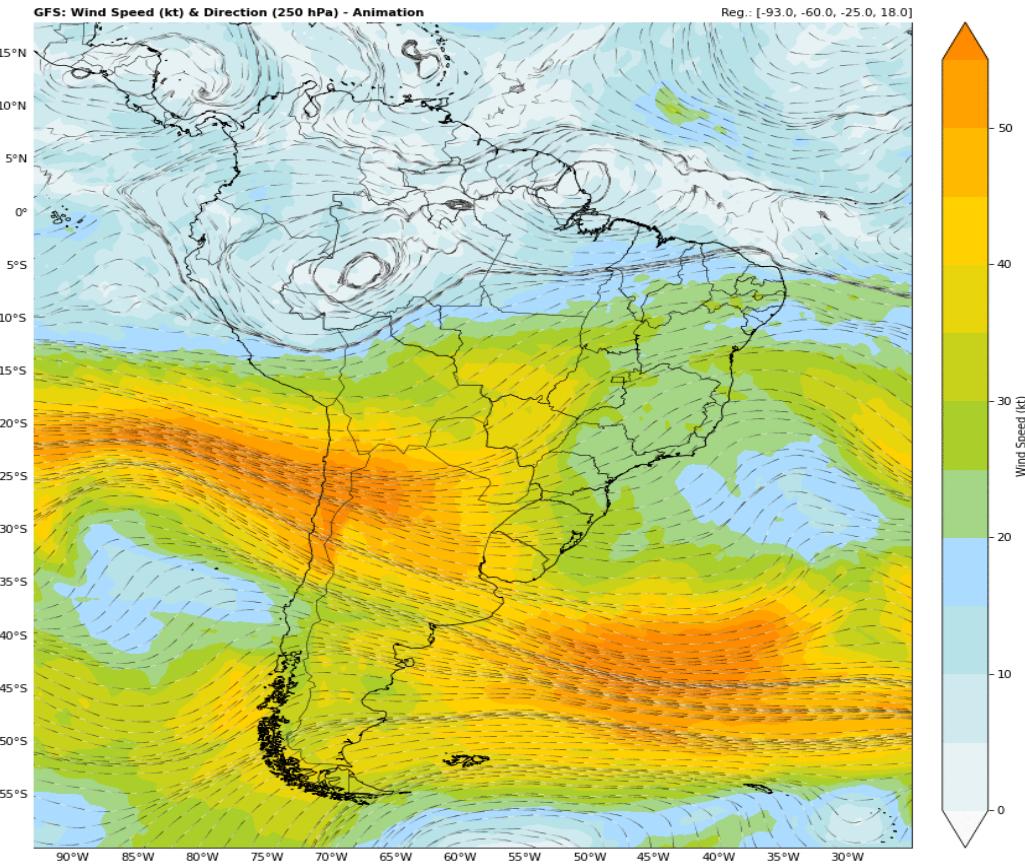
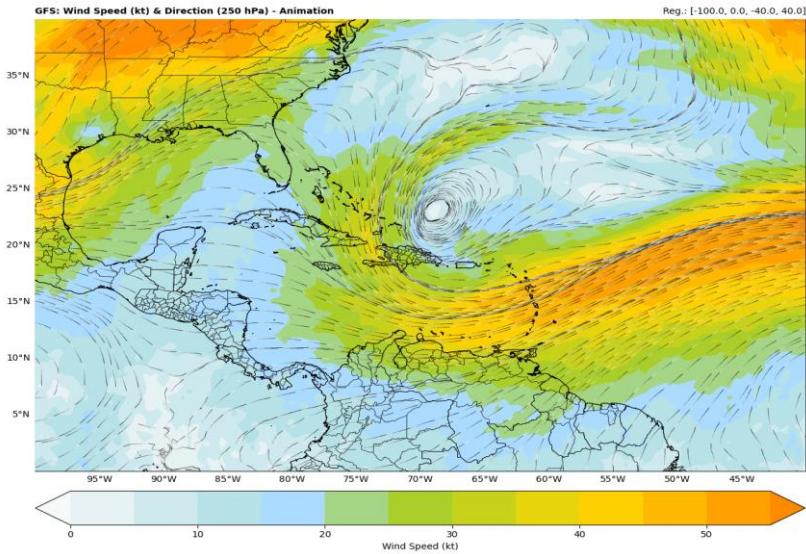
PLOT CONFIGURATION

Script 14: Animating Streamlines

```
274 # Create the streamlines animation
275 lengths = []
276 colors = []
277 lines = []
278 s = Streamlines(X, Y, U, V)
279
280 for streamline in s.streamlines:
281     x, y = streamline
282
283     # Points
284     points = np.array([x, y]).T.reshape(-1, 1, 2)
285
286     # Segments
287     segments = np.concatenate([points[:-1], points[1:]], axis=1)
288     n = len(segments)
289
290     # Lengths
291     D = np.sqrt(((points[1:] - points[:-1])**2).sum(axis=-1))
292     L = D.cumsum().reshape(n,1) + np.random.uniform(0,1)
293
294     # Colors
295     C = np.zeros((n,3))
296     C[:] = ((L*1.5) % 1)
297
298     # Lines
299     line = LineCollection(segments, linewidth=0.5, zorder=13)
300
301     # Append lengths, colors and lines
302     lengths.append(L)
303     colors.append(C)
304     lines.append(line)
305     ax.add_collection(line)
306
307 # Plot extent
308 ax.set_xlim(extent[0],extent[2]), ax.set_xticks([])
309 ax.set_ylim(extent[1],extent[3]), ax.set_yticks([])
310 plt.tight_layout()
311
312 #-----
313
314 # Number of frames
315 n = 30
316
317 # Create the animation
318 animation = FuncAnimation(fig, update, frames=n, interval=20)
319
320 # Update the progress bar
321 #pbar = tqdm.tqdm(total=n)
322
323 # Save animation as GIF
324 print("\nGenerating the GIF...")
325 animation.save('streamlines_animation.gif', writer='pillow', fps=30)
326
327 # Close the progress bar
328 #pbar.close()
```

CREATING THE ANIMATION, CALLNG THE FUNCTIONS

Script 14: Animating Streamlines



Some Quick Modifications

- Change the level to be animated





Script 15: Wind Vectors

► GEONETCast-Americas Training for the Eastern Caribbean States

► GEONETCast-Americas Training for the Eastern Caribbean States | Day 3 - Session 4: NWP Data Processing

LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

LIBRARIES

DATA READING AND MANIPULATION

Selecting variables specifying the level

```
# INPE / CPTEC Training: NWP Data Processing With Python - Script 15: Wind Vectors
# Author: Diego Souza
#
import pygrib                                # Provides a high-level interface to the ECWMF ECCODES C library
import matplotlib.pyplot as plt                 # Plotting library
import cartopy, cartopy.crs as ccrs            # Plot maps
import cartopy.io.shapereader as shapereader   # Import shapefiles
import numpy as np                            # Scientific computing with Python
import matplotlib                           # Comprehensive library for creating static, animated, and interactive plots
#
#
# Select the extent [min. lon, min. lat, max. lon, max. lat]
extent = [-93.0, -60.00, -25.00, 18.00]

# Open the GRIB file
grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
#
#
# Select the variable
ucomp = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 925)[0]
# Get information from the file
init = str(ucomp.analDate)          # Init date / time
run = str(ucomp.hour).zfill(2)      # Run
ftime = str(ucomp.forecastTime)     # Forecast hour
valid = str(ucomp.validDate)        # Valid date / time
print('Init: ' + init + ' UTC')
print('Run: ' + run + ' Z')
print('Forecast: ' + ftime)
print('Valid: ' + valid + ' UTC')

# Read the data for a specific region
ucomp, lats, lons = ucomp.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)

#
#
# Select the variable
vcomp = grib.select(name='V component of wind', typeOfLevel = 'isobaricInhPa', level = 925)[0]
# Read the data for a specific region
vcomp = vcomp.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]

#
#
# Calculate the wind speed
ws = np.sqrt(ucomp**2 + vcomp**2)

#
# Choose the plot size (width x height, in inches)
plt.figure(figsize=(8,8))
```

LIBRARIES

DATA READING AND MANIPULATION

Selecting variables specifying the level

Script 15: Wind Vectors

PLOT CONFIGURATION

```

50
51 # Choose the plot size (width x height, in inches)
52 plt.figure(figsize=(8,8))
53
54 # Use the Cylindrical Equidistant projection in cartopy
55 ax = plt.axes(projection=ccrs.PlateCarree())
56
57 # Define the image extent
58 img_extent = [extent[0], extent[2], extent[1], extent[3]]
59 ax.set_extent(extent[0], extent[2], extent[1], extent[3]), ccrs.PlateCarree())
60
61 # Add a shapefile
62 # https://geoftp.ibge.gov.br/organizacao_dos_territorios/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR_unidades_da_federacao.zip
63 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
64 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
65
66 # Add coastlines, borders and gridlines
67 ax.coastlines(resolution='10m', color='black', linewidth=0.8)
68 ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
69 gl = ax.gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
70 gl.top_labels = False
71 gl.right_labels = False
72
73 # Define de contour interval
74 data_min = 0
75 data_max = 60
76 interval = 5
77 levels = np.arange(data_min,data_max,interval)
78
79 # Create a custom color palette
80 colors = ['#e7f2f4', '#cceeae', '#b6e2e8', '#abdcff', '#a4d685', '#9cd04e', '#abcf2a', '#c9d21b', '#e8d50c', '#ffd100', '#ffba00', '#ffa200']
81 cmap = matplotlib.colors.ListedColormap(colors)
82 cmap.set_over('#fffc00')
83 cmap.set_under('#ffafaf')
84
85 # Plot the contours
86 img1 = ax.contourf(lons, lats, ws, cmap=cmap, levels=levels, extend='both')
87 img2 = ax.contour(lons, lats, ws, colors='white', linewidths=0.3, levels=levels)
88 ax.clabel(img2, inline=1, inline_spacing=10, fmt='%1.1f', colors='black')
89
90 # Plot the quiver
91 img3 = ax.quiver(lons[:,::4,::4], lats[:,::4,::4], ws[:,::4,::4], vcomp[:,::4,::4])
92 qk = ax.quiverkey(img3, 0.50, 0.89, 20, '20 kt', labelpos='E', coordinates='figure')
93
94 # Add a colorbar
95 plt.colorbar(img1, label='Wind Speed (kt)', orientation='vertical', pad=0.05, fraction=0.05)
96
97 # Add a title
98 plt.title('GFS: Wind Speed (kt) & Direction (925 hPa)', fontweight='bold', fontsize=10, loc='left')
99 plt.title('Valid at 12:00 UTC on 10/01/2023', fontweight='normal', fontsize=8, loc='right')
100
101 # Save the image
102 plt.savefig('image_15.png', bbox_inches='tight', pad_inches=0, dpi=100)
103
104 # Show the image
105 plt.show()

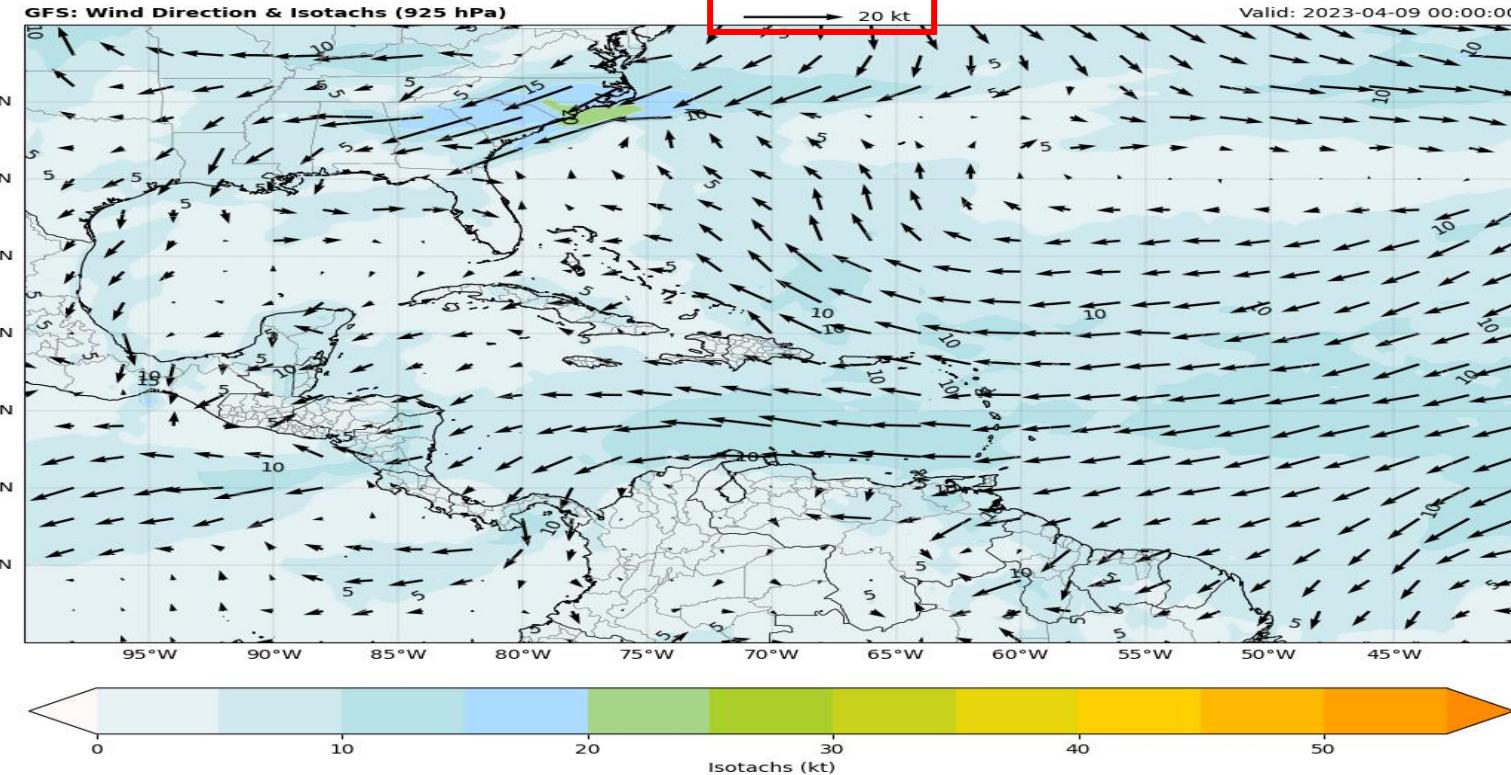
```

Latitudes and Longitudes U and V components Plot the wind vectors and legends

IMAGE GENERATION

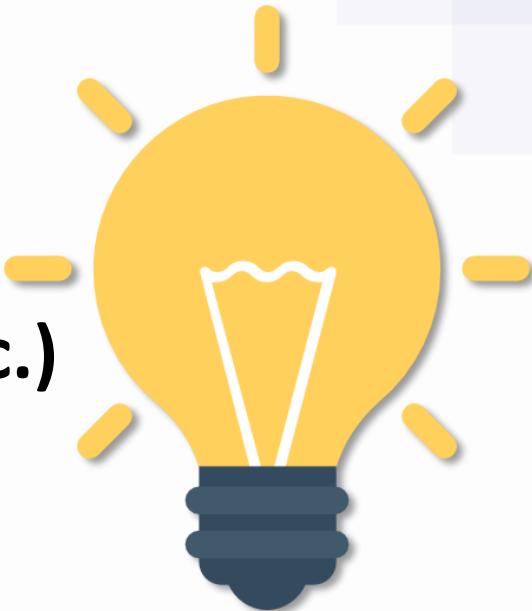
Script 15: Wind Vectors

```
90 # Plot the quiver
91 img3 = ax.quiver(lons[::4,::4], lats[::4,::4], ucomp[::4,::4], vcomp[::4,::4])
92 qk = ax.quiverkey(img3, 0.50, 0.89, 20, '20 kt', labelpos='E', coordinates='figure')
```



Some Quick Modifications

- Change the appearance of the wind vector (densities, colors, colormap, etc.)





Script 16: Bars

LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

LIBRARIES

DATA READING AND MANIPULATION

Selecting variables specifying the level

```
# INPE / CPTEC Training: NWP Data Processing With Python - Script 16: Barbs
# Author: Diego Souza

import pygrib                                # Provides a high-level interface to the ECMWF ECCODES C library
import matplotlib.pyplot as plt                # Plotting library
import cartopy, cartopy.crs as ccrs            # Plot maps
import cartopy.io.shapereader as shapereader   # Import shapefiles
import numpy as np                            # Scientific computing with Python
import matplotlib                           # Comprehensive library for creating static, animated, and interactive plots

#
# Select the extent [min. lon, min. lat, max. lon, max. lat]
extent = [-93.0, -60.0, -25.00, 18.00]

# Open the GRIB file
grib = pygrib.open("gfs.t00z.pgrb2full.0p50.5000")

#
# Select the variable
ucomp = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 850) [0]

# Get information from the file
init = str(ucomp.initDate)                      # Init date / time
run = str(ucomp.hour).zfill(2)                   # Run
ftime = str(ucomp.forecastTime)                 # Forecast hour
valid = str(ucomp.validDate)                    # Valid date / time
print('Init: ' + init + ' UTC')
print('Run: ' + run + 'Z')
print('Forecast: ' + ftime)
print('Valid: ' + valid + ' UTC')

# Read the data for a specific region
ucomp, lats, lons = ucomp.data(lat1=extent[1], lat2=extent[3], lon1=extent[0]+360, lon2=extent[2]+360)

#
# Select the variable
vcomp = grib.select(name='V component of wind', typeOfLevel = 'isobaricInhPa', level = 850) [0]

# Read the data for a specific region
vcomp = vcomp.data(lat1=extent[1], lat2=extent[3], lon1=extent[0]+360, lon2=extent[2]+360) [0]

#
# Select the variable
prmls = grib.select(name='Pressure reduced to MSL') [0]

# Read the data for a specific region
prmls = prmls.data(lat1=extent[1], lat2=extent[3], lon1=extent[0]+360, lon2=extent[2]+360) [0]

#
# Calculate the wind speed
ws = np.sqrt(ucomp**2 + vcomp**2)

# Convert to hPa
prmls = prmls / 100

#
# Choose the plot size (width x height, in inches)
plt.figure(figsize=(8,8))


```

LIBRARIES

DATA READING AND MANIPULATION

Selecting variables specifying the level

Single field, no need to specify level

Single field, no need to specify level

Script 16: Barbs

```

62 # Choose the plot size (width x height, in inches)
63 plt.figure(figsize=(8,8))
64
65 # Use the Cylindrical Equidistant projection in cartopy
66 ax = plt.axes(projection=ccrs.PlateCarree())
67
68 # Define the image extent
69 img_extent = [extent[0], extent[2], extent[1], extent[3]]
70 ax.set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
71
72 # Add a shapefile
73 # https://geoftp.ibge.gov.br/organizacao do territorio/malhas territoriais/malhas_municipais/municipio_2019/Brasil/BR_unidades_da_federacao.zip
74 shapefile = list(shapereader.Reader('BR_UF_2019.shp').geometries())
75 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
76
77 # Add coastlines, borders and gridlines
78 ax.coastlines(resolution='10m', color='black', linewidth=0.8)
79 ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
80 gl = ax.gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
81 gl.top_labels = False
82 gl.right_labels = False
83
84 # Define de contour interval
85 data_min = 990
86 data_max = 1050
87 interval = 2
88 levels = np.arange(data_min,data_max,interval)
89
90 # Create a custom color palette
91 colors = ["#31d000", "#63b000", "#942800", "#c53500", "#fd6123", "#fb824e", "#faa679", "#f8c1a1", "#f1e1b7", "#fafafaee", "#fcfcf5", "#fefefc", "#e8eef5", "#d9e2ef", "#c9d7e5", "#bacce2"]
92 cmap = matplotlib.colors.ListedColormap(colors)
93 cmap.set_over('#000000')
94 cmap.set_under('#000000')
95
96 # Plot the image
97 img1 = ax.contourf(lons, lats, prmls, cmap=cmap, levels=levels, extend='both')
98 img2 = ax.contour(lons, lats, prmls, colors='black', linewidths=0.1, levels=levels)
99 ax.clabel(img2, inline=1, inline_spacing=0, fontsize=10, fmt = '%1.0f', colors= 'black')
100
101 # Create a flag to determine which barbs are flipped
102 flip_flag = np.zeros((uncomp.shape[0],uncomp.shape[1]))
103
104 # All flags below the equator will
105 flip_flag[lats < 0] = 1
106
107 # Plot the barbs
108 img3 = ax.barbs(lons[::4,::4], lats[::4,::4], uncomp[::4,::4], vcomp[::4,::4], length=0.5, capstyle='block', dict(emptybarb=0.0, spacing=0.2, height=0.5), linewidth=0.8, pivot='middle', barbcolor='gray').
109 flip_barb = flip_flag[::4,::4]
110
111 # Add a colorbar
112 plt.colorbar(img1, label='PSML (hPa)', orientation='vertical', pad=0.05, fraction=0.05)
113
114 # Add a title
115 plt.title('GFS: PSML + Winds (850 hPa)', fontweight='bold', fontsize=10, loc='left')
116 plt.title('Valid: ' + valid, fontsize=10, loc='right')
117
118 # Save the image
119 plt.savefig('image_16.png', bbox_inches='tight', pad_inches=0, dpi=100)
120
121 # Show the image
122 plt.show()

```

PLOT CONFIGURATION

Plot the PSML

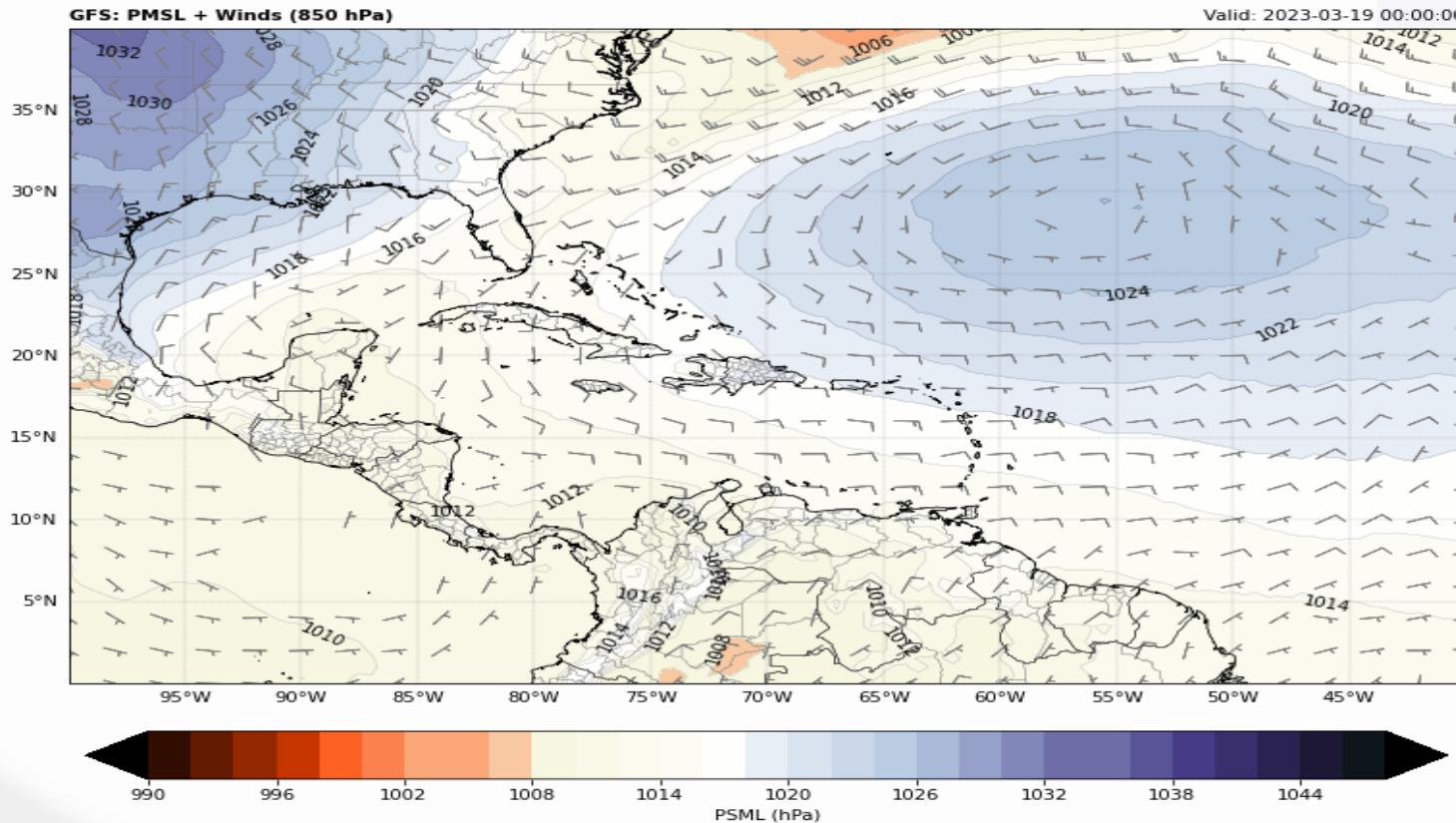
Creates the array that will indicate which barbs should be inverted

Plot the barbs

IMAGE GENERATION

Pixels below 0° set to 1 flip_barb parameter

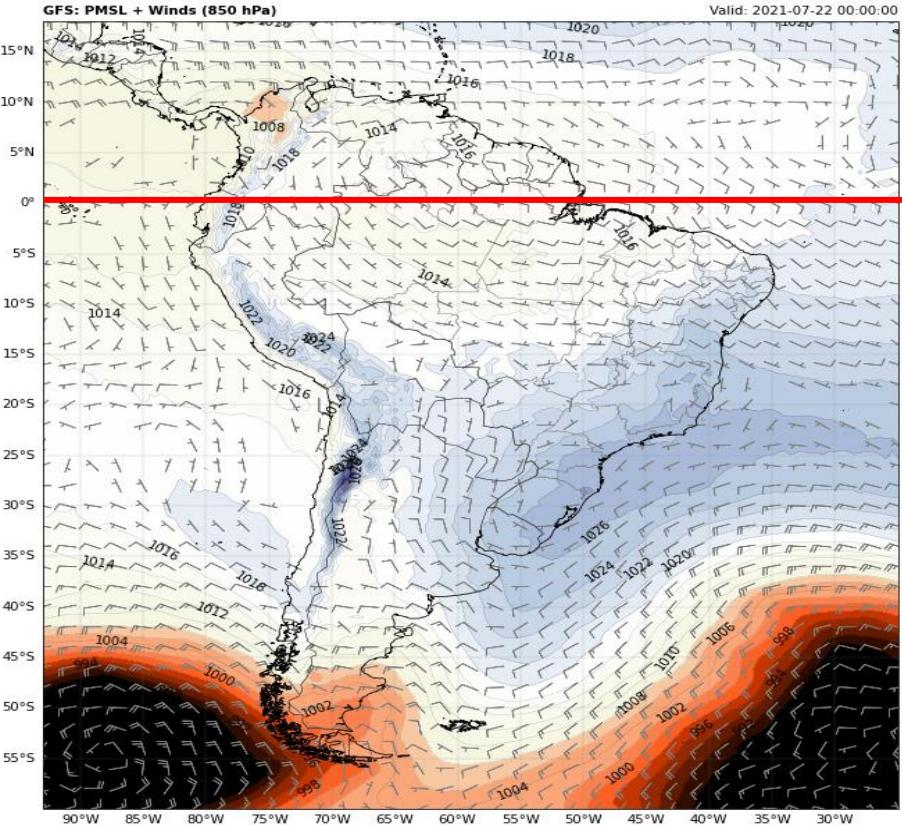
Script 16: Barbs



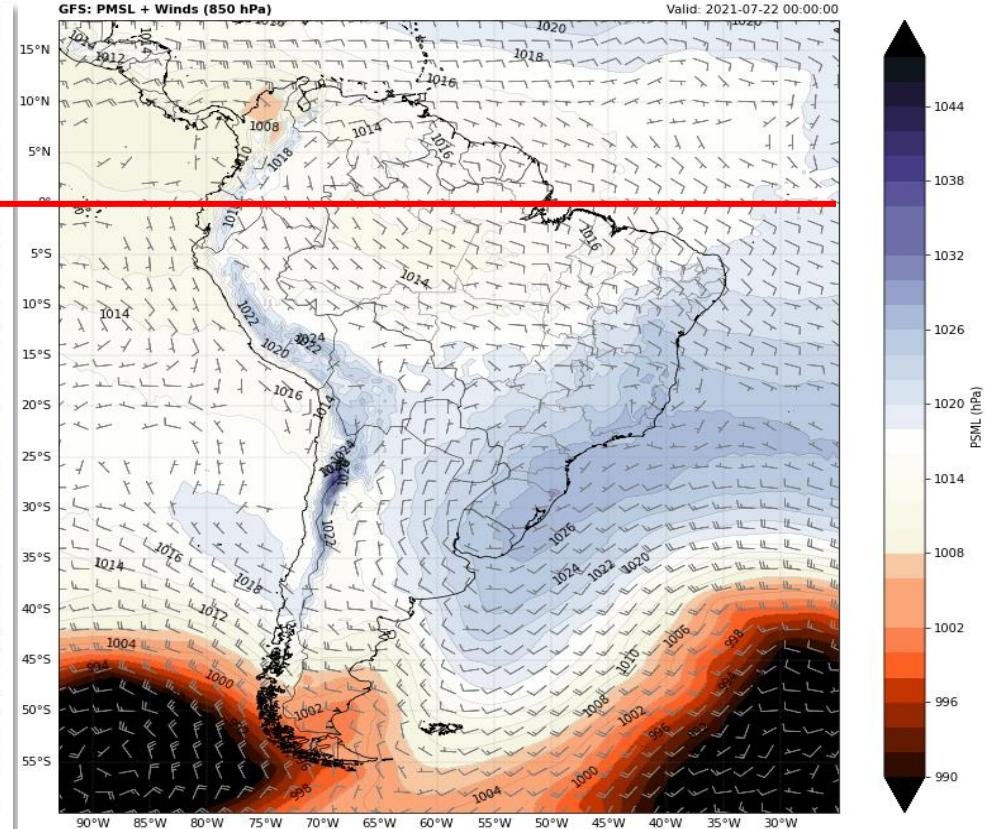
```
107 # Plot the barbs
108 img3 = ax.barbs(lons[::4,::4], lats[::4,::4], ucomp[::4,::4], vcomp[::4,::4], length = 5.0, sizes = dict(emptybarb=0.0, spacing=0.2, height=0.5), linewidth=0.8, pivot='middle', barbcolor='gray',
109 flip_barb = flip_flag[::4,::4])
```

Script 16: Barbs

With the flip_barb parameter:



Without the flip_barb parameter:



Some Quick Modifications

- Change the appearance of the barbs
(density, colors, colormap, size, etc.)



LIBRARIES

DATA READING AND MANIPULATION

GENERAL PLOT CONFIG.

PLOT 1

PLOT 2

PLOT 3

PLOT 4

IMAGE GENERATION

```
1 # INPE / CPTEC Training: NWP Data Processing With Python - Script 17: 2 x 2 Plot - Streamlines (250, 500, 700 and 850 hPa)
2 # Author: Diego Souza
3
4 import pygrib                                # Provides a high-level interface to the ECWMF ECCODES C library
5 import matplotlib.pyplot as plt                 # Plotting library
6 import cartopy, cartopy.crs as ccrs            # Plot maps
7 import cartopy.io.shapereader as shapereader   # Import shapefiles
8 import numpy as np                            # Scientific computing with Python
9 import matplotlib                           # Comprehensive library for creating static, animated, and interactive
10
11
12 # Select the extent [min. lon, min. lat, max. lon, max. lat]
13 extent = [-93.0, -60.0, -25.0, 18.00]
14
15 # Open the GRIB file
16 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
17
18 -----
19
20 # Select the variable
21 ucomp_250 = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 250)[0]
22
23
24 # Get information from the file
25 init = str(ucomp_250.analDate)      # Init date / time
26 run = str(ucomp_250.hour).zfill(2)  # Run
27 ftime = str(ucomp_250.forecastTime) # Forecast hour
28 valid = str(ucomp_250.validDate)    # Valid date / time
29 print('Init: ' + init + ' UTC')
30 print('Run: ' + run + 'Z')
31 print('Forecast: ' + ftime)
32 print('Valid: ' + valid + ' UTC')
33
34
35 # Read the data for a specific region
36 ucomp_250, lats, lons = ucomp_250.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)
37
38 -----
39
40 # Select the variable
41 vcomp_250 = grib.select(name='V component of wind', typeOfLevel = 'isobaricInhPa', level = 250)[0]
42
43 # Read the data for a specific region
44 vcomp_250 = vcomp_250.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]
45
46
47 # Select the variable
48 ucomp_500 = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 500)[0]
49
50 # Read the data for a specific region
51 ucomp_500 = ucomp_500.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]
52
53
```

DATA READING AND MANIPULATION (PART I)

LIBRARIES

Script 17: 2 x 2 Plot - Streamlines at 250, 500, 700 and 850 hPa

LIBRARIES

DATA READING AND MANIPULATION

GENERAL PLOT CONFIG.

PLOT 1

PLOT 2

PLOT 3

PLOT 4

IMAGE GENERATION

```
45
46
47 # Select the variable
48 ucomp_500 = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 500)[0]
49
50 # Read the data for a specific region
51 ucomp_500 = ucomp_500.data(lat1=extent[1],lat2=extent[3],lonl=extent[0]+360,lon2=extent[2]+360)[0]
52
53 -----
54
55 # Select the variable
56 vcomp_500 = grib.select(name='V component of wind', typeOfLevel = 'isobaricInhPa', level = 500)[0]
57
58 # Read the data for a specific region
59 vcomp_500 = vcomp_500.data(lat1=extent[1],lat2=extent[3],lonl=extent[0]+360,lon2=extent[2]+360)[0]
60
61 -----
62
63 # Select the variable
64 ucomp_700 = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 700)[0]
65
66 # Read the data for a specific region
67 ucomp_700 = ucomp_700.data(lat1=extent[1],lat2=extent[3],lonl=extent[0]+360,lon2=extent[2]+360)[0]
68
69 -----
70
71 # Select the variable
72 vcomp_700 = grib.select(name='V component of wind', typeOfLevel = 'isobaricInhPa', level = 700)[0]
73
74 # Read the data for a specific region
75 vcomp_700 = vcomp_700.data(lat1=extent[1],lat2=extent[3],lonl=extent[0]+360,lon2=extent[2]+360)[0]
76
77 -----
78
79 # Select the variable
80 ucomp_850 = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 850)[0]
81
82 # Read the data for a specific region
83 ucomp_850 = ucomp_850.data(lat1=extent[1],lat2=extent[3],lonl=extent[0]+360,lon2=extent[2]+360)[0]
84
85 -----
86
87 # Select the variable
88 vcomp_850 = grib.select(name='V component of wind', typeOfLevel = 'isobaricInhPa', level = 850)[0]
89
90 # Read the data for a specific region
91 vcomp_850 = vcomp_850.data(lat1=extent[1],lat2=extent[3],lonl=extent[0]+360,lon2=extent[2]+360)[0]
92
93
94
95 # Calculate the wind speed
96 ws_250 = np.sqrt(ucomp_250**2 + vcomp_250**2)
97 ws_500 = np.sqrt(ucomp_500**2 + vcomp_500**2)
98 ws_700 = np.sqrt(ucomp_700**2 + vcomp_700**2)
99 ws_850 = np.sqrt(ucomp_850**2 + vcomp_850**2)
```

Wind speed
calculation

DATA READING AND
MANIPULATION (CONT.)

GENERAL PLOT CONFIGURATION

2 row x 2 column plot

Spacing

```

101
102
103 # Define de contour interval
104 data_min = 0
105 data_max = 60
106 interval = 5
107 levels = np.arange(data_min,data_max,interval)
108
109 # Create a custom color palette
110 colors = ["#c7f2f4", "#ceaeae", "#b6e2e8", "#abdcff", "#a4d685", "#9cd04e", "#abcf2a", "#c9d21b", "#e8d50c", "#ffd100", "#feba00", "#ffa200"]
111 cmap = matplotlib.colors.ListedColormap(colors)
112 cmap.set_over('#ff8c00')
113 cmap.set_under('#ffffafa')
114
115
116 # Choose the plot size (width x height, in inches)
117 fig, axs = plt.subplots(2,2, figsize=(16,16), sharex = False, sharey = False, gridspec_kw={'left':0, 'bottom':0, 'right':1, 'top':1, 'hspace':0.05, 'wspace':0.05}, subplot_kw=dict(
118 projection=ccrs.PlateCarree())) # 2 row x 2 columns
119
120
121
122 # Define the image extent
123 axs[0,0].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
124
125 # Add a shapefile
126 # https://geoftnp.ibge.gov.br/organizacao do territorio/malhas territoriais/malhas municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
127 shpreader = list(shpreader.Reader('BR_UF_2019.shp').geometries())
128 axs[0,0].add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
129
130 # Plot the coastlines, borders and gridlines
131 axs[0,0].coastlines(resolution='10m', color='black', linewidth=0.8)
132 axs[0,0].add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
133 gl = axs[0,0].gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
134 gl.top_labels = False
135 gl.right_labels = False
136
137 # Plot the contours
138 img1 = axs[0,0].contourf(lons, lats, ws_250, cmap=cmap, levels=levels, extend='both')
139 img2 = axs[0,0].contour(lons, lats, ws_250, colors='white', linewidths=0.3, levels=levels)
140 axs[0,0].clabel(img2, inline=1, inline_spacing=0, fontsize='10', fmt = '%1.0f', colors= 'black')
141
142 # Plot the streamlines
143 from matplotlib.axes import Axes
144 img3 = Axes.streamplot(axs[0,0], lons, lats, ucomp_250, vcomp_250, density=[4, 4], linewidth=1, color='gray', transform=ccrs.PlateCarree())
145
146 # Add a colorbar
147 plt.colorbar(img1, label='Isotachs (kt)', orientation='vertical', pad=0.02, fraction=0.05, ax=axs[0,0])
148
149 # Title
150 axs[0,0].set_title('GFS: Streamlines and Isotachs (250 hPa)', fontweight='bold', fontsize=10, loc='left')
151 axs[0,0].set_title('Valid: ' + valid, fontsize=10, loc='right')

```

PLOT 1 (250 hPa)

```

122 # Define the image extent
123 axs[0,0].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
124
125 # Add a shapefile
126 # https://geoftnp.ibge.gov.br/organizacao do territorio/malhas territoriais/malhas municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
127 shpreader = list(shpreader.Reader('BR_UF_2019.shp').geometries())
128 axs[0,0].add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
129
130 # Plot the coastlines, borders and gridlines
131 axs[0,0].coastlines(resolution='10m', color='black', linewidth=0.8)
132 axs[0,0].add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
133 gl = axs[0,0].gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
134 gl.top_labels = False
135 gl.right_labels = False
136
137 # Plot the contours
138 img1 = axs[0,0].contourf(lons, lats, ws_250, cmap=cmap, levels=levels, extend='both')
139 img2 = axs[0,0].contour(lons, lats, ws_250, colors='white', linewidths=0.3, levels=levels)
140 axs[0,0].clabel(img2, inline=1, inline_spacing=0, fontsize='10', fmt = '%1.0f', colors= 'black')
141
142 # Plot the streamlines
143 from matplotlib.axes import Axes
144 img3 = Axes.streamplot(axs[0,0], lons, lats, ucomp_250, vcomp_250, density=[4, 4], linewidth=1, color='gray', transform=ccrs.PlateCarree())
145
146 # Add a colorbar
147 plt.colorbar(img1, label='Isotachs (kt)', orientation='vertical', pad=0.02, fraction=0.05, ax=axs[0,0])
148
149 # Title
150 axs[0,0].set_title('GFS: Streamlines and Isotachs (250 hPa)', fontweight='bold', fontsize=10, loc='left')
151 axs[0,0].set_title('Valid: ' + valid, fontsize=10, loc='right')

```

Script 17: 2 x 2 Plot - Streamlines at 250, 500, 700 and 850 hPa

```
153  +
154
155 # Define the image extent
156 axs[0,1].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
157
158 # Add a shapefile
159 # https://geoftp.ibge.gov.br/organizacao_dos_territorios/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR_unidades_da_federacao.zip
160 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
161 axs[0,1].add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
162
163 # Add coastlines, borders and gridlines
164 axs[0,1].coastlines(resolution='10m', color='black', linewidth=0.8)
165 axs[0,1].add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
166 gl = axs[0,1].gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
167 gl.top_labels = False
168 gl.right_labels = False
169
170 # Plot the contours
171 img4 = axs[0,1].contourf(lons, lats, ws_500, cmap=cmap, levels=levels, extend='both')
172 img5 = axs[0,1].contour(lons, lats, ws_500, colors='white', linewidths=0.3, levels=levels)
173 axs[0,1].clabel(img5, inline=1, inline_spacing=0, fontsize='10', fmt = '%1.0f', colors= 'black')
174
175 # Plot the streamlines
176 from matplotlib.axes import Axes
177 img6 = Axes.streamplot(axs[0,1], lons, lats, uncomp_500, vcomp_500, density=[4, 4], linewidth=1, color='gray', transform=ccrs.PlateCarree())
178
179 # Add a colorbar
180 plt.colorbar(img4, label='Isotachs (kt)', orientation='vertical', pad=0.02, fraction=0.05, ax=axs[0,1])
181
182 # Add a title
183 axs[0,1].set_title('GFS: Streamlines and Isotachs (500 hPa)', fontweight='bold', fontsize=10, loc='left')
184 axs[0,1].set_title('Valid: ' + valid, fontsize=10, loc='right')
185
186 #-
```

PLOT 2 (500 hPa)

Script 17: 2 x 2 Plot - Streamlines at 250, 500, 700 and 850 hPa

```
186  #-
187
188  # Define the image extent
189  axs[1,0].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
190
191  # Add a shapefile
192  # https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
193  shapefile = list(shapereader.Reader('BR_UF_2019.shp').geometries())
194  axs[1,0].add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
195
196  # Add coastlines, borders and gridlines
197  axs[1,0].coastlines(resolution='10m', color='black', linewidth=0.8)
198  axs[1,0].add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
199  gl = axs[1,0].gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
200  gl.top_labels = False
201  gl.right_labels = False
202
203  # Plot the contours
204  img7 = axs[1,0].contour(lons, lats, ws_700, cmap=cmap, levels=levels, extend='both')
205  img8 = axs[1,0].contour(lons, lats, ws_700, colors='white', linewidths=0.3, levels=levels)
206  axs[1,0].clabel(img8, inline=1, inline_spacing=0, fontsize='10', fmt = '%1.0f', colors= 'black')
207
208  # Plot the streamlines
209  from matplotlib.axes import Axes
210  img9 = Axes.streamplot(axs[1,0], lons, lats, ucomp_700, vcomp_700, density=[4, 4], linewidth=1, color='gray', transform=ccrs.PlateCarree())
211
212  # Add a colorbar
213  plt.colorbar(img7, label='Isotachs (kt)', orientation='vertical', pad=0.02, fraction=0.05, ax=axs[1,0])
214
215  # Add a title
216  axs[1,0].set_title('GFS: Streamlines and Isotachs (700 hPa)', fontweight='bold', fontsize=10, loc='left')
217  axs[1,0].set_title('Valid: ' + valid, fontsize=10, loc='right')
218
219  #-
```

PLOT 3 (700 hPa)

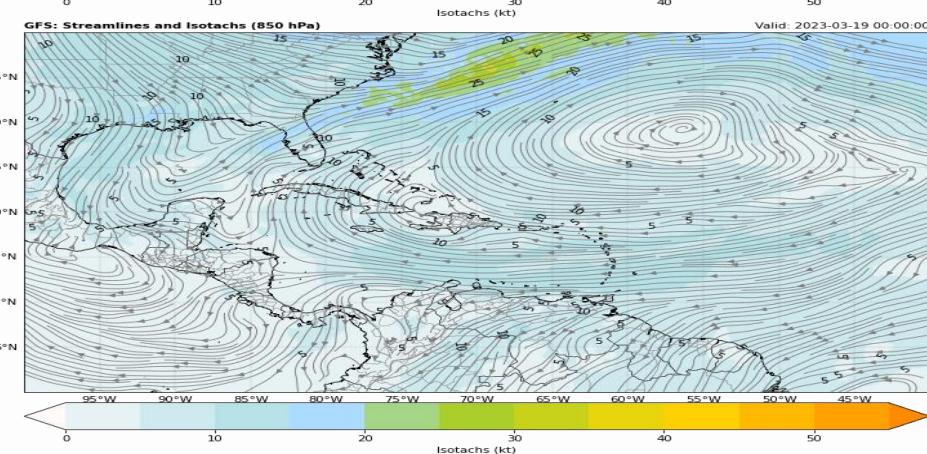
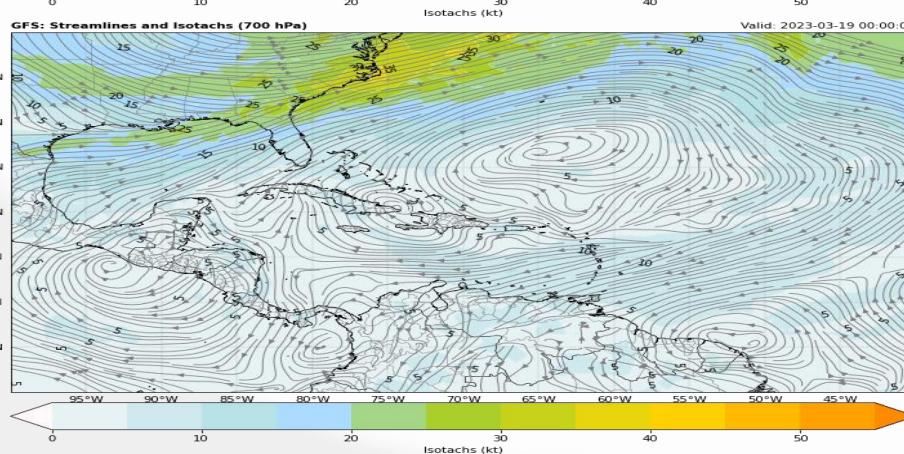
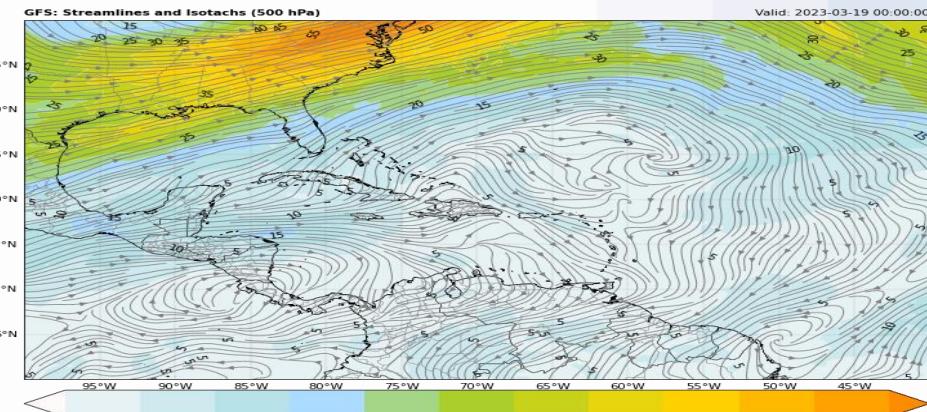
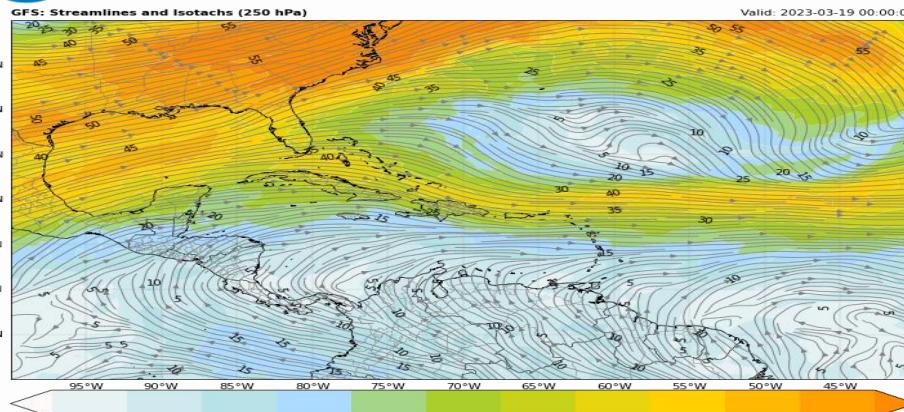
Script 17: 2 x 2 Plot - Streamlines at 250, 500, 700 and 850 hPa

```
218
219
220
221     # Define the image extent
222     axs[1,1].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
223
224     # Add a shapefile
225     # https://geoftp.ibge.gov.br/organizacao\_do\_territorio/malhas\_territoriais/malhas\_municipais/municipio\_2019/Brasil/BR\_unidades\_da\_federacao.zip
226     shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
227     axs[1,1].add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
228
229     # Add coastline, borders and gridlines
230     axs[1,1].coastlines(resolution='10m', color='black', linewidth=0.8)
231     axs[1,1].add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
232     gl = axs[1,1].gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
233     gl.top_labels = False
234     gl.right_labels = False
235
236     # Plot the contours
237     img10 = axs[1,1].contourf(lons, lats, ws_850, cmap=cmap, levels=levels, extend='both')
238     img11 = axs[1,1].contour(lons, lats, ws_850, colors='white', linewidths=0.3, levels=levels)
239     axs[1,1].clabel(img11, inline=1, inline_spacing=0, fontsize='10', fmt = '%1.0f', colors= 'black')
240
241     # Plot the streamlines
242     from matplotlib.axes import Axes
243     img12 = Axes.streamplot(axs[1,1], lons, lats, ucomp_850, vcomp_850, density=[4, 4], linewidth=1, color='gray', transform=ccrs.PlateCarree())
244
245     # Add a colorbar
246     plt.colorbar(img10, label='Isotachs (kt)', orientation='vertical', pad=0.02, fraction=0.05, ax=axs[1,1])
247
248     # Add a title
249     axs[1,1].set_title('GFS: Streamlines and Isotachs (850 hPa)', fontweight='bold', fontsize=10, loc='left')
250     axs[1,1].set_title('Valid: ' + valid, fontsize=10, loc='right')
251
252
253
254     # Save the image
255     plt.savefig('image_17.png', bbox_inches='tight', pad_inches=0, dpi=100)
256
257     # Show the image
258     plt.show()
```

PLOT 4 (850 hPa)

IMAGE GENERATION

Script 17: 2 x 2 Plot - Streamlines at 250, 500, 700 and 850 hPa



Script 18: Galvez Davison Index (GDI)

Source: <https://www.wpc.ncep.noaa.gov/international/gdi/>

(1) Layer-averaged thetas and mixing ratios

$$\theta_A = \theta_{950} = T_{950}(1000/950)^{2/7} \quad (1.1)$$

$$\theta_B = 0.5(\theta_{850} + \theta_{700}) = 0.5[T_{850}(1000/850)^{2/7} + T_{700}(1000/700)^{2/7}] \quad (1.2)$$

$$\theta_C = \theta_{500} = T_{500}(1000/950)^{2/7} \quad (1.3)$$

$$r_A = r_{950} \quad (1.4)$$

$$r_B = 0.5(r_{850} + r_{700}) \quad (1.5)$$

$$r_C = r_{500} \quad (1.6)$$

(2) Theta-E (EPT) Proxies

- Used formula from Davies-Jones 2009
- Replaced T_{LCL} by T_{850} to simplify (small effect)

$$EPTP_A = \theta_A e^{\left(\frac{-L_o T_A}{c_{pd} T_{850}} \right)} \quad (1.7)$$

$$EPTP_B = \theta_B e^{\left(\frac{-L_o T_B}{c_{pd} T_{850}} \right)} + \alpha \quad (1.8)$$

$$EPTP_C = \theta_C e^{\left(\frac{-L_o T_C}{c_{pd} T_{850}} \right)} + \alpha \quad (1.9)$$

where $\alpha = -10$ [K] is an empirical adjustment constant, $L_o = 2.69 \times 10^6$ [J kg⁻¹] is a latent heat constant adjusted for this formula and $c_{pd} = 1005.7$ [J kg⁻¹ K⁻¹] is the specific heat of dry air at constant pressure.

(3) EPT Core index (ECI)

$$ME = EPTP_C - \beta \quad (1.10)$$

$$LE = EPTP_A - \beta \quad (1.11)$$

where $\beta = 303$ [K] is an empirical constant that enhances EPTP variability. The *ECI* is then calculated via (1.12):

$$ECI = \begin{cases} \gamma \times LE \times ME & , LE > 0 \\ 0 & , LE \leq 0 \end{cases} \quad (1.12)$$

where $\gamma = 6.5 \times 10^{-2}$ [K⁻¹] is an empirical scaling constant. The reasoning behind (1.12) is that if

(4) Mid Warming Index (MWI)

$$MWI = \begin{cases} \mu \times (T_{500} - \tau) & , T_{500} - \tau > 0 \\ 0 & , T_{500} - \tau \leq 0 \end{cases} \quad (1.13)$$

where $\mu = -7$ [K⁻¹] is an empirical scaling constant and T_{500} is the 500 hPa temperature in [K].

(5) Inversion Index (II)

$$S = \sigma \times (T_{950} - T_{700}) \quad (1.14)$$

where $\sigma = 1.5$ [K⁻¹] is an empirical scaling constant. A Drying factor D is then calculated based on the difference of EPTP between layers A and B via (1.15). In the tropics, dry air associated with subsidence inversions has a strong signature in the EPTP field resulting in a sharp decrease of EPTP with height.

$$D = \sigma \times (EPTP_B - EPTP_A) \quad (1.15)$$

Both S and D are combined via (1.16) to calculate the II :

$$II = \begin{cases} 0 & , D + S > 0 \\ D + S & , D + S \leq 0 \end{cases} \quad (1.16)$$

(6) GDI

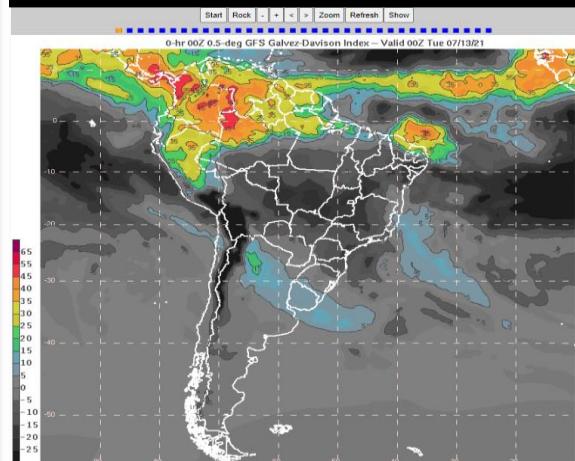
$$GDI = ECI + MWI + II \quad (1.17)$$

*Orography correction just to aid visualization for forecaster. Co is added to the GDI

$$Co = P_3 - \frac{P_2}{P_{SFC} - P_1}$$

where $P_1 = 500$ [hPa], $P_2 = 9000$ [hPa] and $P_3 = 18$

Let's try to create this tool



GDI > +45	Potential for scattered to widespread heavy rain producing thunderstorms.
+35 to +45	Potential for scattered thunderstorms some capable of producing heavy rainfall.
+25 to +35	Potential for scattered thunderstorms or scattered shallow convection with isolated thunderstorms.
+15 to +25	Potential for a few isolated thunderstorms, but mostly shallow convection.
+05 to +15	Potential for shallow convection. A very isolated and brief thunderstorm is possible.
-20 to +05	Potential for isolated to scattered shallow convection. Strong subsidence inversion likely.
-20 > GDI	Strong subsidence inversion. Any convection should be very shallow, isolated, and produce trace accumulations.

Script 18: Galvez Davison Index (GDI)

► WHICH VARIABLES DO WE NEED TO CREATE GDI?

(1) Layer-averaged thetas and mixing ratios

$$\theta_A = \theta_{950} = T_{950}(1000/950)^{2/7} \quad (1.1)$$

$$\theta_B = 0.5(\theta_{850} + \theta_{700}) = 0.5[T_{850}(1000/850)^{2/7} + T_{700}(1000/700)^{2/7}] \quad (1.2)$$

$$\theta_C = \theta_{500} = T_{500}(1000/950)^{2/7} \quad (1.3)$$

$$r_A = r_{950} \quad (1.4)$$

$$r_B = 0.5(r_{850} + r_{700}) \quad (1.5)$$

$$r_C = r_{500} \quad (1.6)$$

Temperatures (950, 850, 700 and 500 hPa)
(950, 850, 700 and 500 hPa) Specific Humidities or Mixing Ratio

(2) Theta-E (EPT) Proxys

- Used formula from Davies-Jones 2009
- Replaced T_{LCL} by T_{850} to simplify (small effect)

$$EPTP_A = \theta_A e^{\left(\frac{L_o}{c_{pd}T_{850}}\right)} \quad (1.7)$$

$$EPTP_B = \theta_B e^{\left(\frac{L_o}{c_{pd}T_{850}}\right)} + \alpha \quad (1.8)$$

$$EPTP_C = \theta_C e^{\left(\frac{L_o}{c_{pd}T_{850}}\right)} + \alpha \quad (1.9)$$

where $\alpha = -10$ [K] is an empirical adjustment constant, $L_o = 2.69 \times 10^6$ [J kg⁻¹] is a latent heat constant adjusted for this formula and $c_{pd} = 1005.7$ [J kg⁻¹ K⁻¹] is the specific heat of dry air at constant pressure.

(3) EPT Core index (ECI)

$$ME = EPTP_C - \beta \quad (1.10)$$

$$LE = EPTP_A - \beta \quad (1.11)$$

where $\beta = 303$ [K] is an empirical constant that enhances EPTP variability. The ECI is then calculated via (1.12):

$$ECI = \begin{cases} \gamma \times LE \times ME & , LE > 0 \\ 0 & , LE \leq 0 \end{cases} \quad (1.12)$$

where $\gamma = 6.5 \times 10^{-2}$ [K⁻¹] is an empirical scaling constant. The reasoning behind (1.12) is that if

*Orography correction just to aid visualization for forecaster. Co is added to the GDI

$$C_0 = P_3 - \frac{P_2}{P_{SFC} - P_1}$$

where $P_1 = 500$ [hPa], $P_2 = 9000$ [hPa] and $P_3 = 18$

Surface Pressure

(4) Mid Warming Index (MWI)

$$MWI = \begin{cases} \mu \times \left(\frac{T_{500} - \tau}{0}\right) & , T_{500} - \tau > 0 \\ 0 & , T_{500} - \tau \leq 0 \end{cases} \quad (1.13)$$

where $\mu = -7$ [K⁻¹] is an empirical scaling constant and T_{500} is the 500 hPa temperature in [K].

(5) Inversion Index (II)

$$S = \sigma \times (T_{500} - \tau_{500}) \quad (1.14)$$

where $\sigma = 1.5$ [K⁻¹] is an empirical scaling constant. A drying factor D is then calculated based on the difference between the 500 hPa temperature and τ via (1.15). In the tropics, dry air associated with mid-tropospheric inversions has a strong signature in the EPTP field resulting in a sharp decrease of EPTP with height:

$$D = \sigma \times (EPTP_B - EPTP_C) \quad (1.15)$$

Both S and D are combined via (1.16) to calculate the II:

$$II = \begin{cases} 0 & , D + S \geq 0 \\ 1 & , D + S < 0 \end{cases} \quad (1.16)$$

Reference: https://www.wpc.ncep.noaa.gov/international/gdi/GDI_Manuscript_V20161021.pdf

Script 18: Galvez Davison Index (GDI)

LIBRARIES

READING THE NECESSARY DATA

CALCULATIONS ACCORDING TO PROCEDURE

PLOT CONFIGURATION

IMAGE GENERATION

```
1  #-----  
2  # INPE / CPTEC Training: NWP Data Processing - Galvez Davison Index (GDI)  
3  # Author: Diego Souza / HUGE THANKS TO JUAN AMIDES FIGUEROA (MARN EL SALVADOR) AND JOSE GALVEZ  
4  #-----  
5  import pygrib                                # Provides a high-level interface to the ECWMF ECCODES C library  
6  import matplotlib.pyplot as plt                 # Plotting library  
7  import cartopy, cartopy.crs as ccrs            # Plot maps  
8  import cartopy.io.shapereader as shapereader    # Import shapefiles  
9  import numpy as np                            # Scientific computing with Python  
10 import matplotlib                           # Comprehensive library for creating static, animated, and interactive visualizations  
11 import math                                  # Mathematical Functions  
12 #-----  
13  
14 # Select the extent [min. lon, min. lat, max. lon, max. lat]  
15 extent = [-93.0, -60.00, -25.00, 18.00]  
16  
17 # Open the GRIB file  
18 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")  
19  
20 #-----  
21  
22 # Read the surface pressure  
23 sfcps = grib.select(name='Surface pressure')[0]  
24  
25 # Get information from the file  
26 init = str(sfcps.analDate)                      # Init date / time  
27 run = str(sfcps.hour).zfill(2)                   # Run  
28 ftime = str(sfcps.forecastTime)                  # Forecast hour  
29 valid = str(sfcps.validDate)                    # Valid date / time  
30 print('Init: ' + init + ' UTC')  
31 print('Run: ' + run + 'z')  
32 print('Forecast: +' + ftime)  
33 print('Valid: ' + valid + ' UTC')  
34  
35 # Read the data for a specific region  
36 sfcps, lats, lons = sfcps.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)  
37  
38 # Convert the surface pressure  
39 sfcps = sfcps / 100  
40  
41
```

Explaining the new instructions: Galvez Davison Index (GDI)

LIBRARIES

READING THE NECESSARY DATA (PART I)

Convert Pressure to Hectopascal

Script 18: Galvez Davison Index (GDI)

LIBRARIES

READING THE NECESSARY DATA

CALCULATIONS ACCORDING TO PROCEDURE

PLOT CONFIGURATION

IMAGE GENERATION

```
41 # Read the temperature in 950 hPa
42 temp950 = grib.select(name='Temperature', typeOfLevel = 'isobaricInhPa', level = 950) [0]
43
44 # Read the data for a specific region
45 temp950 = temp950.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360) [0]
46
47 # Read the specific humidity in 950 hPa
48 spfh950 = grib.select(name='Specific humidity', typeOfLevel = 'isobaricInhPa', level = 950) [0]
49
50 # Read the data for a specific region
51 R950 = spfh950.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360) [0]
52
53 # Read the temperature in 850 hPa
54 temp850 = grib.select(name='Temperature', typeOfLevel = 'isobaricInhPa', level = 850) [0]
55
56 # Read the data for a specific region
57 temp850 = temp850.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360) [0]
58
59 # Read the specific humidity in 850 hPa
60 spfh850 = grib.select(name='Specific humidity', typeOfLevel = 'isobaricInhPa', level = 850) [0]
61
62 # Read the data for a specific region
63 R850 = spfh850.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360) [0]
64
65 # Read the temperature in 700 hPa
66 temp700 = grib.select(name='Temperature', typeOfLevel = 'isobaricInhPa', level = 700) [0]
67
68 # Read the data for a specific region
69 temp700 = temp700.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360) [0]
70
71 # Read the specific humidity in 700 hPa
72 spfh700 = grib.select(name='Specific humidity', typeOfLevel = 'isobaricInhPa', level = 700) [0]
73
74 # Read the data for a specific region
75 R700 = spfh700.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360) [0]
76
77 # Read the temperature in 500 hPa
78 temp500 = grib.select(name='Temperature', typeOfLevel = 'isobaricInhPa', level = 500) [0]
79
80 # Read the data for a specific region
81 temp500 = temp500.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360) [0]
82
83 # Read the specific humidity in 500 hPa
84 spfh500 = grib.select(name='Specific humidity', typeOfLevel = 'isobaricInhPa', level = 500) [0]
85
86 # Read the data for a specific region
87 R500 = spfh500.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360) [0]
88
89
```

READING THE NECESSARY DATA (CONT.)

Script 18: Galvez Davison Index (GDI)

(1) Layer-averaged thetas and mixing ratios

$$\theta_A = \theta_{950} = T_{950}(1000/950)^{2/7} \quad (1.1)$$

$$\theta_B = 0.5(\theta_{850} + \theta_{700}) = 0.5[T_{850}(1000/850)^{2/7} + T_{700}(1000/700)^{2/7}] \quad (1.2)$$

$$\theta_C = \theta_{500} = T_{500}(1000/950)^{2/7} \quad (1.3)$$

$$r_A = r_{950} \quad (1.4)$$

$$r_B = 0.5(r_{850} + r_{700}) \quad (1.5)$$

$$r_C = r_{500} \quad (1.6)$$

```

107 # Layer-averaged thetas and mixing ratios
108 THETAA = T950
109 A1 = T850 + T700
110 THETAB = 0.5 * A1
111 THETAC = T500
112 RA = R950
113 C2 = R850 + R700
114 RB = 0.5 * C2
115 RC = R500

```

```

92 # Calculate the Theta 950
93 T950 = ((temp950) * (pow((1000/950), (2/7))))
94
95 # Calculate the Theta 850
96 T850 = ((temp850) * (pow((1000/850), (2/7))))
97
98 # Calculate the Theta 700
99 T700 = ((temp850) * (pow((1000/700), (2/7))))
100
101 # Calculate the Theta 500
102 T500 = ((temp500) * (pow((1000/500), (2/7))))

```

Script 18: Galvez Davison Index (GDI)

(2) Theta-E (EPT) Proxys

- Used formula from Davies-Jones 2009
- Replaced T_{LCL} by T_{850} to simplify (small effect)

$$EPTP_A = \theta_A e^{\left(\frac{L_0 r_A}{c_{pd} T_{850}}\right)} \quad (1.7)$$

$$EPTP_B = \theta_B e^{\left(\frac{L_0 r_B}{c_{pd} T_{850}}\right)} + \alpha \quad (1.8)$$

$$EPTP_C = \theta_C e^{\left(\frac{L_0 r_C}{c_{pd} T_{850}}\right)} + \alpha \quad (1.9)$$

where $\alpha = -10$ [K] is an empirical adjustment constant, $L_0 = 2.69 \times 10^6$ [J kg⁻¹] is a latent heat constant adjusted for this formula and $c_{pd} = 1005.7$ [J kg⁻¹ K⁻¹] is the specific heat of dry air at constant pressure.

```

120 # Calculate the Theta-E (EPT) Proxys
121 L0 = 2690000
122 alpha = -10
123 cpd = 1005.7
124 p1 = L0 * RA
125 p2 = L0 * RB
126 p3 = L0 * RC
127 p4 = cpd * T850
128 x1 = p1 / p4
129 x2 = p2 / p4
130 x3 = p3 / p4
131 y1 = np.exp(x1)
132 y2 = np.exp(x2)
133 y3 = np.exp(x3)
134 EPTPA = THETAA * y1
135 EPTPB = THETAB * y2 + alpha
136 EPTPC = THETAC * y3 + alpha
    
```

**Numpy:
Exponential**

(2) Theta-E (EPT) Proxys

- Used formula from Davies-Jones 2009
- Replaced T_{LCL} by T_{850} to simplify (small effect)

$$EPTP_A = \theta_A e^{\left(\frac{L_0 r_A}{c_{pd} T_{850}}\right)} \quad (1.7)$$

$$EPTP_B = \theta_B e^{\left(\frac{L_0 r_B}{c_{pd} T_{850}}\right)} + \alpha \quad (1.8)$$

$$EPTP_C = \theta_C e^{\left(\frac{L_0 r_C}{c_{pd} T_{850}}\right)} + \alpha \quad (1.9)$$

where $\alpha = -10$ [K] is an empirical adjustment constant, $L_0 = 2.69 \times 10^6$ [J kg⁻¹] is a latent heat constant adjusted for this formula and $c_{pd} = 1005.7$ [J kg⁻¹ K⁻¹] is the specific heat of dry air at constant pressure.

Script 18: Galvez Davison Index (GDI)

```

LIBRARIES
    import numpy as np
    from netCDF4 import Dataset
    import os
    import math
    import sys
    import warnings
    warnings.filterwarnings('ignore')

    # Importing the required libraries for data processing and visualization

READING THE
NECESSARY DATA
    # Reading the necessary data from NetCDF files
    # This section includes reading temperature profiles, dew point profiles, and pressure profiles for layers A and B.

CALCULATIONS
ACCORDING TO
PROCEDURE
    # Calculations according to the procedure
    # This section includes calculating the EPT Core Index (ECI), Mid Warming Index (MWI), Inversion Index (II), and Orography correction factor (CO).

PLOT CONFIGURATION
    # Plot configuration
    # This section includes setting up plot parameters like resolution and projection.

IMAGE GENERATION
    # Image generation
    # This section includes generating the final GDI map.

```

```

141 # Calculate the EPT Core Index (ECI)
142 beta = 303
143 ME = EPTPC - beta
144 LE = EPTPA - beta
145 gamma= 0.065
146
147 # Decide the value of the ECI
148 ECI = np.where(LE <= 0, 0, gamma * LE * ME)
149
150 Numpy: Where
151 #
152 # Calculate the Mid Warming Index (MWI)
153 tau = 263.15
154 mu = -7
155 calc1 = temp500 - tau
156
157 # Decide the value of the MWI
158 MWI = np.where(calc1 <= 0, 0, calc1 * mu)
159
160 #
161 #
162 #
163
164 # Calculate the Inversion Index (II)
165 sigma = 1.5
166 op1 = temp950 - temp700
167 S = sigma * op1
168 op2 = EPTPB - EPTPA
169 D = sigma * op2
170 calc2 = D + S
171
172 # Decide the value of the II
173 II = np.where(calc2 <= 0, D + S, 0)
174
175 #
176 #
177
178 # Orography correction just to aid visualization
179 pp1 = 500
180 pp2 = 9000
181 pp3 = 18
182 divisor = sfcps - ppl
183 division = pp2 / divisor
184 CO = pp3 - division
185

```

(3) EPT Core index (ECI)

$$ME = EPTPC - \beta \quad (1.10)$$

$$LE = EPTPA - \beta \quad (1.11)$$

where $\beta = 303$ [K] is an empirical constant that enhances EPTP variability. The ECI is then calculated via (1.12):

$$ECI = \begin{cases} \gamma \times LE \times ME & , LE > 0 \\ 0 & , LE \leq 0 \end{cases} \quad (1.12)$$

where $\gamma = 6.5 \times 10^{-2}$ [K⁻¹] is an empirical scaling constant. The reasoning behind (1.12) is that if

(4) Mid Warming Index (MWI)

$$MWI = \begin{cases} \mu \times (T_{500} - \tau) & , T_{500} - \tau > 0 \\ 0 & , T_{500} - \tau \leq 0 \end{cases} \quad (1.13)$$

where $\mu = -7$ [K⁻¹] is an empirical scaling constant and T_{500} is the 500 hPa temperature in [K].

(5) Inversion Index (II)

$$S = \sigma \times (T_{950} - T_{700}) \quad (1.14)$$

where $\sigma = 1.5$ [K⁻¹] is an empirical scaling constant. A Drying factor D is then calculated based on the difference of EPT between layers A and B via (1.15). In the tropics, dry air associated with subsidence inversions has a strong signature in the EPTP field resulting in a sharp decrease of EPTP with height.

$$D = \sigma \times (EPTPB - EPTPA) \quad (1.15)$$

Both S and D are combined via (1.16) to calculate the II:

$$II = \begin{cases} 0 & , D + S > 0 \\ D + S & , D + S \leq 0 \end{cases} \quad (1.16)$$

*Orography correction just to aid visualization for forecaster. CO is added to the GDI

$$CO = P_3 - \frac{P_2}{P_{SFC} - P_1}$$

where $P_1 = 500$ [hPa], $P_2 = 9000$ [hPa] and $P_3 = 18$

(6) GDI

$$GDI = ECI + MWI + II$$

(1.17)

```
189 # Galvez Davison Index - Indices with and without correction
190 GDI = ECI + MWI + II
191 GDIC = ECI + MWI + II + CO ←
192
193 # Smooth the contours
194 import scipy.ndimage
195 GDIC = scipy.ndimage.zoom(GDIC, 3)
196 lats = scipy.ndimage.zoom(lats, 3)
197 lons = scipy.ndimage.zoom(lons, 3)
```

*Orography correction just to aid visualization for forecaster. CO is added to the GDI

$$CO = P_3 - \frac{P_2}{P_{SFC} - P_1}$$

where $P_1 = 500$ [hPa], $P_2 = 9000$ [hPa] and $P_3 = 18$

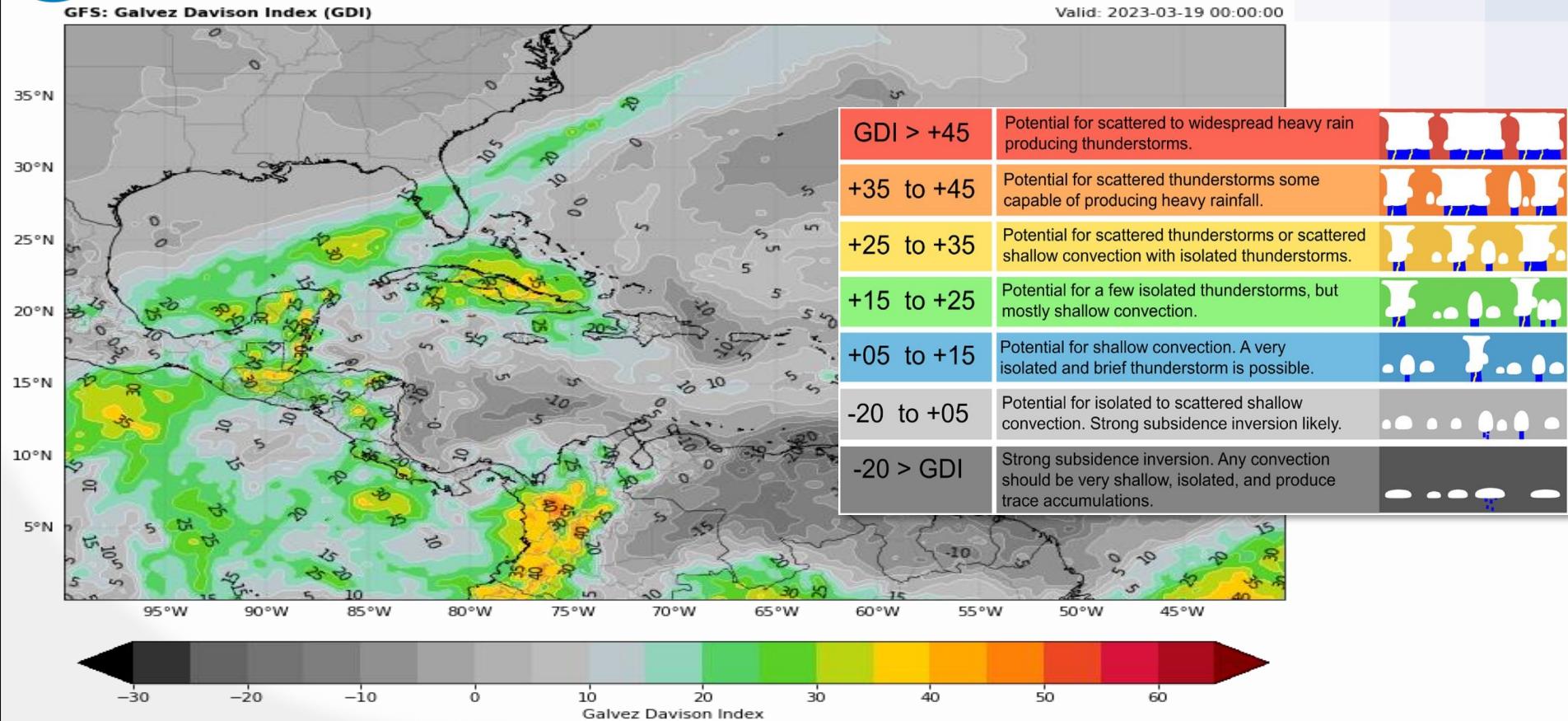
Script 18: Galvez Davison Index (GDI)

```
199  #-
200
201 # Choose the plot size (width x height, in inches)
202 plt.figure(figsize=(8,8))
203
204 # Use the Cylindrical Equidistant projection in cartopy
205 ax = plt.axes(projection=ccrs.PlateCarree())
206
207 # Define the image extent
208 img_extent = [extent[0], extent[2], extent[1], extent[3]]
209
210 # Add a shapefile
211 # https://geotp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR_unidades_da_federacao.zip
212 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
213 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
214
215 # Add coastlines, borders and gridlines
216 ax.coastlines(resolution='10m', color='black', linewidth=0.8)
217 ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
218 gl = ax.gridlines(ccrs.PlateCarree(), color='gray', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
219 gl.top_labels = False
220 gl.right_labels = False
221
222 # Define de contour interval
223 data_min = -30
224 data_max = 70
225 interval = 5
226 levels = np.arange(data_min,data_max,interval)
227
228 # Create the color scale
229 colors = ["#323232", "#646464", "#737373", "#7e7e7e", "#909090", "#a3a3a3", "#b1b1b1", "#bcdbcdb", "#bbc7cb", "#b2d2dd", "#90d5bb", "#55d065", "#5acf28", "#bad411", "#ffcc00", "#ffa900", "#fc8106", "#eb4722", "#d8133a", "#ac0ald"]
230 cmap = matplotlib.colors.ListedColormap(colors)
231 cmap.set_over('#800000')
232 cmap.set_under('#000000')
233
234 # Plot the contours
235 img1 = ax.contourf(lons, lats, GDIC, cmap=cmap, levels=levels, extend='both')
236 img2 = ax.contour(lons, lats, GDIC, colors='white', linewidths=0.3, levels=levels)
237 ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10', fmt = '%1.0f', colors= 'black')
238
239 # Add a colorbar
240 plt.colorbar(img1, label='Galvez Davison Index', orientation='vertical', pad=0.05, fraction=0.05)
241
242 # Add a title
243 plt.title('GFS: Galvez Davison Index (GDI)', fontweight='bold', fontsize=10, loc='left')
244 plt.title('Valid: ' + valid, fontsize=10, loc='right')
245
246 # Save the image
247 plt.savefig('image_18.png', bbox_inches='tight', pad_inches=0, dpi=100)
248
249 # Show the image
250 plt.show()
```

PLOT CONFIGURATION

IMAGE GENERATION

Script 18: Galvez Davison Index (GDI)



Part 2: More Advanced Concepts

PART I	PART II
Knowing the Available Variables	Downloading Data using Scripts
Basic Plot / Pixel Values	Averages, Minimums and Maximums
Metadata, Basic Calculus, Color Palette, Legend and Title	Multiplots
Overlaying Maps with Cartopy	Reading Fields Specifying Levels - Streamlines
Reading a Shapefile	Wind Vectors
Plotting Contours and Labels	Barbs
Custom Color Palettes	Reading Multiple Fields - Galvez Davison Index
Smoothing Contours	Satellite + NWP
Working with Multiple Files	METAR + Satellite + NWP
Animations	

Script 19: Satellite Imagery Plot

```
1  #----  
2  # INPE / CPTEC - Training: Python and GOES-R Imagery: Script 19 - Satellite Plot  
3  # Author: Diego Souza  
4  #  
5  # Required modules  
6  from netCDF4 import Dataset  
7  from osgeo import gdal  
8  import matplotlib.pyplot as plt  
9  import cartopy, cartopy.crs as ccrs  
10 import cartopy.io.shapereader as shapereader  
11 import os  
12 import numpy as np  
13 from matplotlib import cm  
14 from datetime import timedelta, date, datetime  
15 from utilities import download_CMI  
16 from utilities import reproject  
17 from utilities import loadCPT  
18 gdal.PushErrorHandler('CPLQuietErrorHandler')  
19 #  
20 # Input and output directories  
21 input = "Samples"; os.makedirs(input, exist_ok=True)  
22 output = "Output"; os.makedirs(output, exist_ok=True) # Function to download data from AWS  
# Function for reprojection  
# Function to load a colormap in the CPT format  
# Avoid showing warnings  
23 # Select the extent [min. lon, min. lat, max. lon, max. lat]  
24 extent = [-93.0, -60.00, -25.00, 18.00]  
25  
26 # Datetime to process (today in this example, to match the GFS date)  
27 #date = datetime.today().strftime('%Y%m%d')  
28 #yyyymmddhhmn = date + '0000'  
29 yyyymmddhhmn = '202107020000' # CHANGE THIS DATE TO THE SAME DATE OF YOUR NWP DATA # Desired time  
#----- and data  
30 # Channel  
31 #  
32 # Download the ABI file  
33 file_ir = download_CMI(yyyymmddhhmn, 13, input) # Call the download function
```

LIBRARIES AND FUNCTIONS

Note: In COLAB, the date and time are the date of the script execution

Script 19: Satellite Imagery Plot

```
34 # Download the ABI file
35 file_ir = download_CMI(yyyymmddhhmn, 10, Input)
36
37 #
38 # Variable
39 var = 'CMI' Variable to be read
40
41 # Open the file
42 img = gdal.Open(f'NETCDF:{input}/{file_ir}.nc:' + var) Data reading (specify drive [NetCDF, GeoTIFF, GRIB, etc])
43
44 # Read the header metadata
45 metadata = img.GetMetadata()
46 scale = float(metadata.get(var + '#scale_factor'))
47 offset = float(metadata.get(var + '#add_offset'))
48 undef = float(metadata.get(var + '#_FillValue'))
49 dttime = metadata.get('NC_GLOBAL#time_coverage_start') Metadata (scale, offset, NaN, start of scan, etc)
50
51 # Load the data
52 ds_cmi = img.ReadAsArray(0, 0, img.RasterXSize, img.RasterYSize).astype(float)
53
54 # Apply the scale, offset and convert to celsius
55 ds_cmi = (ds_cmi * scale + offset) - 273.15 Read only the data and convert it to an array
56 Applies "scale", "offset" and converts to °C
57 # Reproject the file
58 filename_ret = f'{output}/IR_{yyyymmddhhmn}.nc' Name of the NetCDF file to be generated
59 reproject(filename_ret, img, ds_cmi, extent, undef) Call the reprojection function
60
61 # Open the reprojected GOES-R image
62 file = Dataset(filename_ret) Open the reprojected file
63
64 # Get the pixel values
65 data = file.variables['Band1'][:] Read the reprojected data
66
67
68 # Choose the plot size (width x height, in inches)
69 plt.figure(figsize=(8,8))
```

REPROJECTION AND CROPPING

Script 19: Satellite Imagery Plot

```

64 # Get the pixel values
65 data = file.variables['Band1'][:, :]
66
67
68 # Choose the plot size (width x height, in inches)
69 plt.figure(figsize=(8,8))
70
71 # Use the Geostationary projection in cartopy
72 ax = plt.axes(projection=ccrs.PlateCarree())
73
74 # Define the image extent
75 img_extent = [extent[0], extent[2], extent[1], extent[3]]
76 ax.set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
77
78 # Define the color scale based on the channel
79 colormap = "gray_r" # White to black for IR channels — Inverted gray scale
80
81 # Plot the image
82 img1 = ax.imshow(data, origin='upper', vmin=-80, vmax=60, extent=img_extent, cmap=colormap, alpha=1.0) — Satellite plot
83
84 # Add a shapefile
85 # https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
86 shapefile = list(shapereader.Reader('BR_UF_2019.shp').geometries())
87 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='white', facecolor='none', linewidth=0.3)
88
89 # Add coastlines, borders and gridlines
90 ax.coastlines(resolution='10m', color='white', linewidth=0.8)
91 ax.add_feature(cartopy.feature.BORDERS, edgecolor='white', linewidth=0.5)
92 gl = ax.gridlines(ccrs.PlateCarree(), color='white', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5),
93 draw_labels=True)
94 gl.top_labels = False
95 gl.right_labels = False
96
97 # Add a colorbar
98 plt.colorbar(img1, label='Brightness Temperatures (°C)', extend='both', orientation='vertical', pad=0.03, fraction=0.05)
99
100 # Extract date
101 date = (datetime.strptime(dtime, '%Y-%m-%d %H:%M:%S.%fZ'))
102
103 # Add a title
104 plt.title('GOES-16 Band 13 ' + date.strftime('%Y-%m-%d %H:%M') + ' UTC', fontweight='bold', fontsize=10, loc='left')
105 plt.title('Reg.: ' + str(extent), fontsize=10, loc='right')
106
107 # Save the image
108 plt.savefig(f'{output}/image_19.png', bbox_inches='tight', pad_inches=0, dpi=300)
109
110 # Show the image
111 plt.show()

```

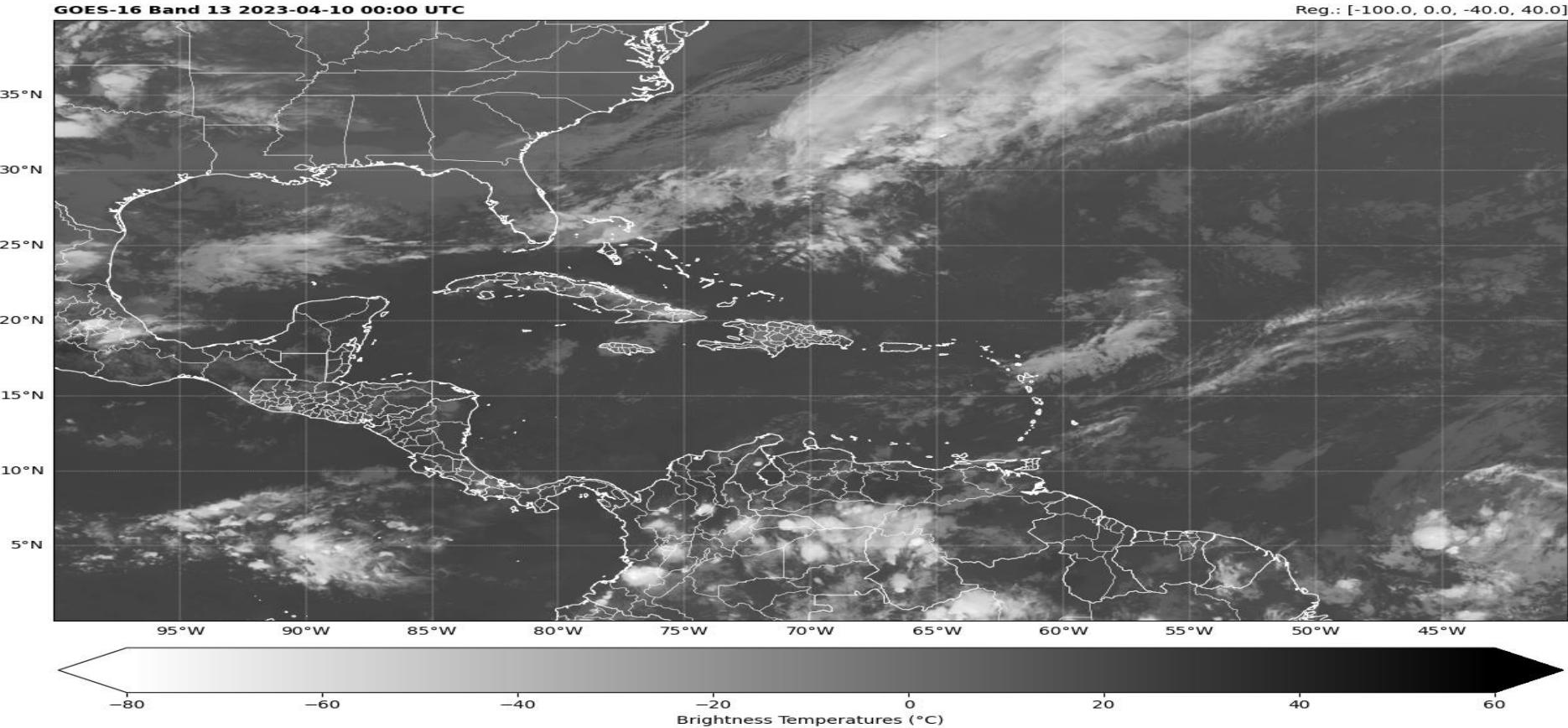
Explaining the new instructions:

PLOT CONFIGURATION

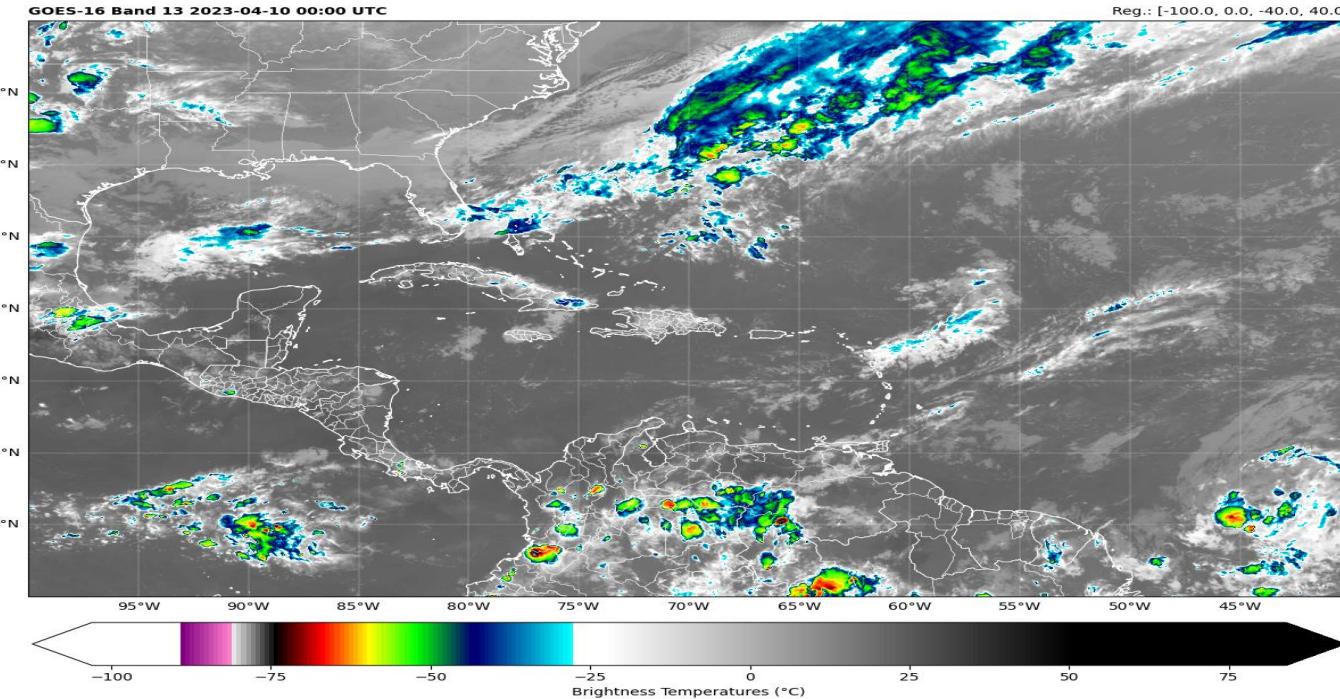
GERAÇÃO DA IMAGEM

f > expressions/variables enclosed in braces in string

Script 19: Satellite Imagery Plot



Script 19: Satellite Imagery Plot



```
# Converts a CPT file to be used in Python
cpt = loadCPT('IR4AVHRR6.cpt')
cmap = cm.colors.LinearSegmentedColormap('cpt', cpt)

# Plot the image
img1 = ax.imshow(data, origin='upper', vmin=-103.0, vmax=84, extent=img_extent, cmap=cmap, alpha=1.0)
```

Some Quick Modifications

- Read other GOES-16 bands



Script 20: NWP + Satellite (Example 1)

```

LIBRARIES AND FUNCTIONS
SATELLITE DATA DOWNLOAD
SATELLITE DATA REPROJECTION AND CROPPING
NWP DATA READING AND MANIPULATION
PLOT CONFIGURATION
IMAGE GENERATION

```

Explaining the new instructions:

```

1  #-----#
2  # INPE / CPTEC - Training: Python and GEONETCast Example 1 NWP + Satellite (Example 1)
3  # Author: Diego Souza
4
5  # Required modules
6  from netCDF4 import Dataset
7  from osgeo import gdal
8  import matplotlib.pyplot as plt
9  import cartopy, cartopy.crs as ccrs
10 import cartopy.io.shapereader as shapereader
11 import os
12 import numpy as np
13 from matplotlib import cm
14 from datetime import timedelta, date, datetime
15 from utilities import download_CMI
16 from utilities import reproject
17 from utilities import loadCPT
18 import pygrib
19 # Note: Import the PyGRIB library
20 gdal.PushErrorHandler('CPLQuietErrorHandler')
21 #-----#
22 # Input and output directories
23 input = "Samples"; os.makedirs(input, exist_ok=True)
24 output = "Output"; os.makedirs(output, exist_ok=True)
25
26 # Select the extent [min. lon, min. lat, max. lon, max. lat]
27 extent = [-93.0, -60.0, -25.0, 18.0]
28
29 # Datetime to process (today in this example, to match the GFS date)
30 #date = datetime.today().strftime('%Y%m%d')
31 #yyyymmddhhmm = date + '0000'
32 yyyymmddhhmm = '202107020000' # CHANGE THIS DATE TO THE SAME DATE OF YOUR NWP DATA
33
34
35 # Download the ABI file
36 file_ir = download_CMI(yyyymmddhhmm, 13, input)
37
38 #-----#

```

LIBRARIES AND FUNCTIONS

SATELLITE DATA DOWNLOAD

Script 20: NWP + Satellite (Example 1)

```
36 file_ir = download_CMI(yyyymmddhhmm, 13, input)
37
38 # Variable
39 var = 'CMI'
40
41 # Open the file
42 img = gdal.Open(f'NETCDF:{input}/{file_ir}.nc' + var)
43
44 # Read the header metadata
45 metadata = img.GetMetadata()
46 scale = float(metadata.get(var + '_scale_factor'))
47 offset = float(metadata.get(var + '_add_offset'))
48 undef = float(metadata.get(var + '_FillValue'))
49 dttime = metadata.get('NC_GLOBAL$time_coverage_start')
50
51 # Load the data
52 ds_cmi = img.ReadAsArray(0, 0, img.RasterXSize, img.RasterYSize).astype(float)
53
54 # Apply the scale, offset and convert to celsius
55 ds_cmi = (ds_cmi * scale + offset) - 273.15
56
57 # Reproject the file
58 filename_ret = f'{output}/IR_{yyyymmddhhmm}.nc'
59 reproject(filename_ret, img, ds_cmi, extent, undef)
60
61 # Open the reprojected GOES-R image
62 file = Dataset(filename_ret)
63
64 # Get the pixel values
65 data = file.variables['Band1'][:]
66
67
68 # Open the GRIB file
69 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
70
71 # Select the variable
72 grb = grib.select(name='Geopotential Height', typeOfLevel = 'isobaricInhPa', level = 500)[0]
73
74 # Get information from the file
75 init = str(grb.analDate) # Init date / time
76 run = str(grb.hour).zfill(2) # Run
77 ftime = str(grb.forecastTime) # Forecast hour
78 valid = str(grb.validDate) # Valid date / time
79 print('Init: ' + init + ' UTC')
80 print('Run: ' + run + 'Z')
81 print('Forecast: ' + ftime)
82 print('Valid: ' + valid + ' UTC')
83
84 # Read the data for a specific region
85 gh500, lats, lons = grb.data(lat1=extent[1], lat2=extent[3], lon1=extent[0]+360, lon2=extent[2]+360)
86
87 # Convert to gpdm
88 gh500 = gh500 / 10
89
90 # Smooth the contours
91 import scipy.ndimage
92 gh500 = scipy.ndimage.zoom(gh500, 3)
93 lats = scipy.ndimage.zoom(lats, 3)
94 lons = scipy.ndimage.zoom(lons, 3)
95
96
```

SATELLITE DATA REPROJECTION AND CROPPING

NWP DATA READING AND MANIPULATION

Script 20: NWP + Satellite (Example 1)

Explaining the new instructions:

```

97  #-
98  # Choose the plot size (width x height, in inches)
99  plt.figure(figsize=(8,8))
100
101 # Use the Geostationary projection in cartopy
102 ax = plt.axes(projection=ccrs.PlateCarree())
103
104 # Define the image extent
105 img_extent = [extent[0], extent[2], extent[1], extent[3]]
106
107 # Define the color scale based on the channel
108 colormap = "gray_r" # White to black for IR channels
109
110 # Plot the image
111 img1 = ax.imshow(data, origin='upper', vmin=-80, vmax=60, extent=img_extent, cmap=colormap, alpha=1.0)
112
113 # Define de contour interval
114 data_min = 400
115 data_max = 600
116 interval = 2
117 levels = np.arange(data_min,data_max,interval)
118
119 # Plot the contours
120 img2 = ax.contour(lons, lats, gh500, cmap='jet', linewidths=1.5, levels=levels)
121 ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10', fmt = '%1.0f', colors= 'blue') # For the labels to have the same colors as the cmap, just omit the "colors" variable
122
123 # Add a shapefile
124 # https://geoftp.ibge.gov.br/organizacao\_territorial/malhas\_territoriais/malhas\_municipais/municipio\_2019/Brasil/BR\_br\_unidades\_da\_federacao.zip
125 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
126 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='black', facecolor='none', linewidth=0.3)
127
128 # Add coastlines, borders and gridlines
129 ax.coastlines(resolution='10m', color='black', linewidth=0.8)
130 ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
131 gl = ax.gridlines(ccrs.PlateCarree(), color='white', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
132 gl.top_labels = False
133 gl.right_labels = False
134
135 # Extract date
136 date = (datetime.strptime(dtime, '%Y-%m-%dT%H:%M:%S.%fZ'))
137
138 # Add a title
139 plt.title('GOES-16 Band 13 ' + date.strftime('%Y-%m-%d %H:%M') + ' UTC' + ' + GFS Geopotential Height - 500 hPa (gpdm)', fontweight='bold', fontsize=6, loc='left')
140 plt.title('Reg.: ' + str(extent), fontsize=6, loc='right')
141
142
143 # Save the image
144 plt.savefig(f'{output}/image_20.png', bbox_inches='tight', pad_inches=0, dpi=300)
145
146 # Show the image
147 plt.show()

```

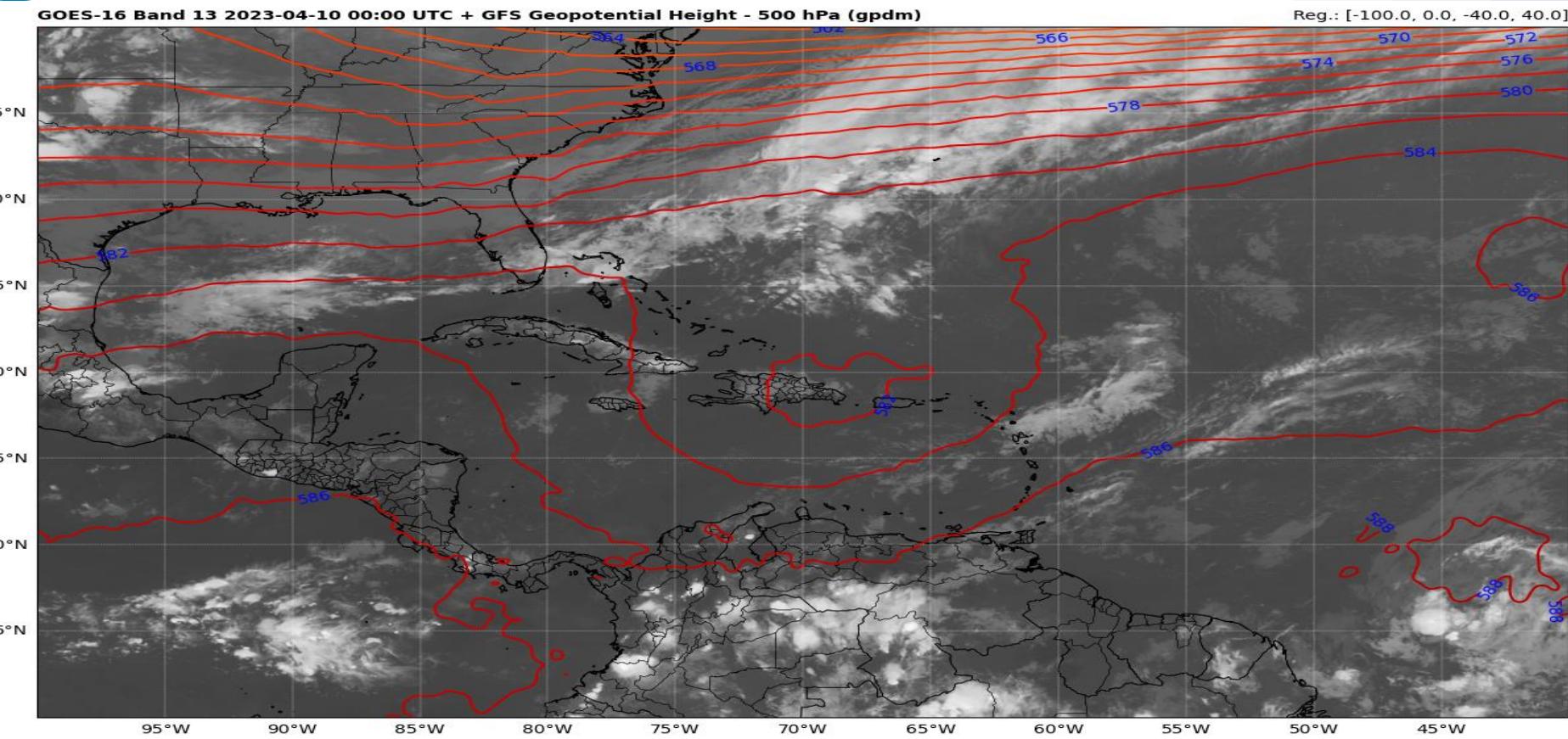
PLOT CONFIGURATION

Satellite plot

Plot the geopotential
height

IMAGE GENERATION

Script 20: NWP + Satellite (Example 1)



Script 21: NWP + Satellite (Example 2)

```

# INPE / CPTEC - Training: Python and Satellite Processing - Example 2 - NWP Plot (Example 2)

# LIBRARIES AND FUNCTIONS
# SATELLITE DATA DOWNLOAD
# SATELLITE DATA REPROJECTION AND CROPPING
# NWP DATA READING AND MANIPULATION
# PLOT CONFIGURATION
# IMAGE GENERATION

```

Explaining the new instructions:

```

1  #-----
2  # INPE / CPTEC - Training: Python and Satellite Processing - Example 2 - NWP Plot (Example 2)
3  # Author: Diego Souza
4
5
6  # Required modules
7  from netCDF4 import Dataset
8  from osgeo import gdal
9  import matplotlib.pyplot as plt
10 import cartopy, cartopy.crs as ccrs
11 import cartopy.io.shapereader as shapereader
12 import os
13 import numpy as np
14 from matplotlib import cm
15 from datetime import timedelta, date, datetime
16 from utilities import download_CMI
17 from utilities import reproject
18 from utilities import loadCPT
19 import pygrib
20 gdal.PushErrorHandler('CPLQuietErrorHandler')
21
22 # Input and output directories
23 input = "Samples"; os.makedirs(input, exist_ok=True)
24 output = "Output"; os.makedirs(output, exist_ok=True)
25
26 # Select the extent [min. lon, min. lat, max. lon, max. lat]
27 extent = [-93.0, -60.0, -25.0, 18.0]
28
29 # Datetime to process (today in this example, to match the GFS date)
30 #date = datetime.today().strftime('%Y%m%d')
31 #yyyymmddhhmm = date + '0000'
32 yyyymmddhhmm = '202107020000' # CHANGE THIS DATE TO THE SAME DATE OF YOUR NWP DATA
33
34
35 # Download the ABI file
36 file_ir = download_CMI(yyyymmddhhmm, 13, input)
37
38

```

LIBRARIES AND FUNCTIONS

SATELLITE DATA DOWNLOAD

Script 21: NWP + Satellite (Example 2)

```
36 file_ir = download_CMI(yyyymmddhhmm, 13, input)
37
38 # Variable
39 var = 'CMI'
40
41 # Open the file
42 img = gdal.Open(f'NETCDF:(input)/(file_ir).nc:' + var)
43
44 # Read the header metadata
45 metadata = img.GetMetadata()
46 scale = float(metadata.get(var + '_scale_factor'))
47 offset = float(metadata.get(var + '_add_offset'))
48 undef = float(metadata.get(var + '_FillValue'))
49 dttime = metadata.get('NC_GLOBAL@time_coverage_start')
50
51 # Load the data
52 ds_cmi = img.ReadAsArray(0, 0, img.RasterXSize, img.RasterYSize).astype(float)
53
54 # Apply the scale, offset and convert to celsius
55 ds_cmi = (ds_cmi * scale + offset) - 273.15
56
57 # Reproject the file
58 filename_ret = f'{output}/IR_{yyyymmddhhmm}.nc'
59 reproject(filename_ret, img, ds_cmi, extent, undef)
60
61 # Open the reprojected GOES-R image
62 file = Dataset(filename_ret)
63
64 # Get the pixel values
65 data = file.variables['Band1'][:]
66
67
68 # Open the GRIB file
69 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
70
71 # Select the variable
72 grib = grib.select(name='Temperature', typeOfLevel = 'isobaricInhPa', level = 850)[0]
73
74 # Get information from the file
75 init = str(grib.analDate) # Init date / time
76 run = str(grib.hour).zfill(2) # Run
77 ftime = str(grib.forecastTime) # Forecast hour
78 valid = str(grib.validDate) # Valid date / time
79 print('Init: ' + init + ' UTC')
80 print('Run: ' + run + 'Z')
81 print('Forecast: ' + ftime)
82 print('Valid: ' + valid + ' UTC')
83
84
85 # Read the data for a specific region
86 tp850, lats, lons = grib.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)
87
88 # Convert from K to °C
89 tp850 = tp850 - 273.15
90
91 # Smooth the contours
92 import scipy.ndimage
93 tp850 = scipy.ndimage.zoom(tp850, 3)
94 lats = scipy.ndimage.zoom(lats, 3)
95 lons = scipy.ndimage.zoom(lons, 3)
96
97
```

SATELLITE DATA REPROJECTION AND CROPPING

NWP DATA READING AND MANIPULATION

Script 21: NWP + Satellite (Example 2)

Explaining the new instructions:

PLOT CONFIGURATION

```

99
100 # Choose the plot size (width x height, in inches)
101 plt.figure(figsize=(10,6))
102
103 # Use the Geostationary projection in cartopy
104 ax = plt.axes(projection=ccrs.PlateCarree())
105
106 # Define the image extent
107 img_extent = [extent[0], extent[2], extent[1], extent[3]]
108
109 # Define the color scale based on the channel
110 colormap = "gray_r" # White to black for IR channels
111
112 # Plot the image
113 img1 = ax.imshow(data, origin='upper', vmin=-80, vmax=60, extent=img_extent, cmap=colormap, alpha=1.0)
114
115 # Define de contour interval
116 data_min = -20
117 data_max = 48
118 interval = 2
119 levels = np.arange(data_min,data_max,interval)
120
121 # Plot the contours
122 img3 = ax.contour(lons, lats, tp850, cmap='jet', linewidths=1.0, levels=levels)
123 ax.clabel(img3, inline=1, inline_spacing=0, fontsize=10, fmt = '%1.0f', colors= 'black') # For the labels to have the same colors as the cmap, just omit the "colors" variable
124
125 # Get the index of elements with value "0"
126 zero_value = int(np.where(levels == 0)[0])
127 img3.collections[zero_value].set_linewidth(4)
128 img3.collections[zero_value].set_edgecolor('black')
129
130 # Add a shapefile
131 shapefile = list(shpreader.Reader('ne_10m_admin_1_states_provinces.shp').geometries())
132 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='white', facecolor='none', linewidth=0.3)
133
134 # Add coastlines, borders and gridlines
135 ax.coastlines(resolution='10m', color='white', linewidth=0.8)
136 ax.add_feature(cartopy.feature.BORDERS, edgecolor='white', linewidth=0.5)
137 gl = ax.gridlines(ccrs.PlateCarree(), color='white', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
138 gl.top_labels = False
139 gl.right_labels = False
140
141 # Extract date
142 date = (datetime.strptime(dtime, '%Y-%m-%dT%H:%M:%S.%fZ'))
143
144 # Add a title
145 plt.title('GOES-16 Band 13 ' + date.strftime('%Y-%m-%d %H:%M') + ' UTC' + ' ' + GFS_Temperature - 850 hPa (°C)', fontweight='bold', fontsize=6, loc='left')
146 plt.title('Reg.: ' + str(extent), fontsize=6, loc='right')
147
148 plt.savefig(f'{output}/image_21.png', bbox_inches='tight', pad_inches=0, dpi=300)
149
150 # Show the image
151 plt.show()

```

Satellite plot

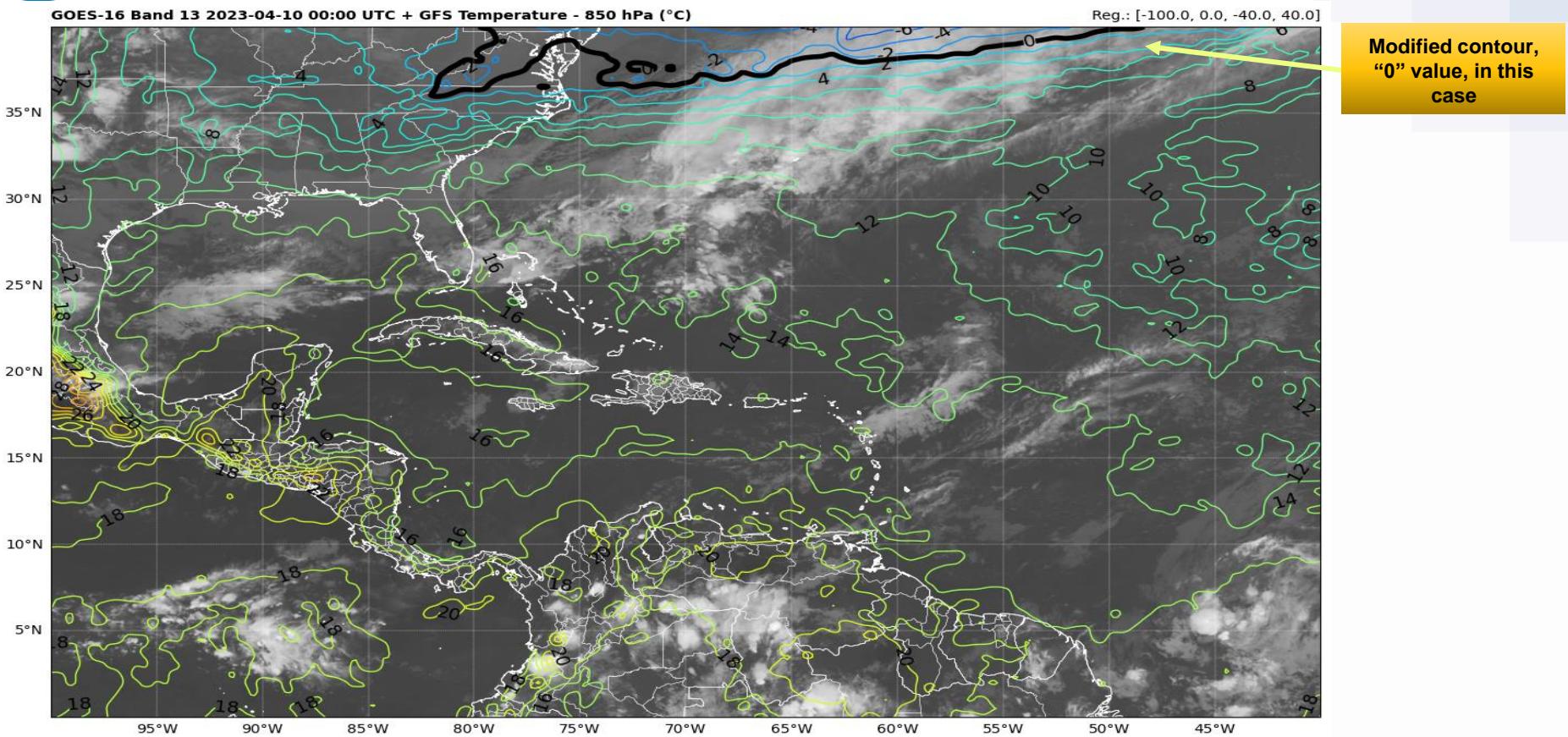
Temperature plot

Numpy: Where

Modify an specific value of the contour ("0", in this case)

IMAGE GENERATION

Script 21: NWP + Satellite (Example 2)



Script 22: NWP + Satellite (Example 3)

LIBRARIES AND FUNCTIONS

FUNTION TO PLOT HIGH / LOW PRESSURE LABELS

SATELLITE DATA DOWNLOAD

SATELLITE DATA REPROJECTION AND CROPPING

NWP DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

Explaining the new instructions:

LIBRARIES AND FUNCTIONS

FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS

```

1  # INPE / CPTEC - Training: Python and GOES-R Imagery
2  # Author: Diego Souza
3
4
5  # Required modules
6  from netCDF4 import Dataset
7  from osgeo import gdal
8  import matplotlib.pyplot as plt
9  import cartopy, cartopy.crs as ccrs
10 import cartopy.io.shapereader as shapereader
11 import os
12 import numpy as np
13 from matplotlib import cm
14 from datetime import timedelta, date, datetime
15 from utilities import download_CMI
16 from utilities import reproject
17 from utilities import loadCPT
18 import pygrib
19 gdal.PushErrorHandler('CPLQuietErrorHandler')
20
21
22 def plot_maxmin_points(lon, lat, data, extrema, nsize, symbol, color='k',
23                         plotValue=True, transform=None):
24     """
25     This function will find and plot relative maximum and minimum for a 2D array
26     can be used to plot an H for maximum values (e.g., High pressure) and an L for minimum
27     values (e.g., low pressure). It is best to used filtered data to obtain a synoptic scale
28     max/min value. The symbol text can be set to a string value and optionally the color
29     symbol and any plotted value can be set with the parameter color
30     lon = plotting longitude values (2D)
31     lat = plotting latitude values (2D)
32     data = 2D data that you wish to plot the max/min symbol placement
33     extrema = Either a value of max for Maximum Values or min for Minimum Values
34     nsize = Size of the grid box to filter the max and min values to plot a reasonable number
35     symbol = String to be placed at location of max/min value
36     color = String matplotlib colormapname to plot the symbol (and numeric value, if plotted)
37     plot_value = Boolean (True/False) of whether to plot the numeric value of max/min point
38     The max/min symbol will be plotted on the current axes within the bounding frame
39     (e.g., clip_on=True)
40
41     from scipy.ndimage filters import maximum_filter, minimum_filter
42
43     if (extrema == 'max'):
44         data_ext = maximum_filter(data, nsize, mode='nearest')
45     elif (extrema == 'min'):
46         data_ext = minimum_filter(data, nsize, mode='nearest')
47     else:
48         raise ValueError('Value for hilo must be either max or min')
49
50     mxy, mxm = np.where(data_ext == data)  —— Gets coordinates of the max and min points
51
52     for i in range(len(mxy)):
53         txt1 = ax.annotate(symbol, xy=(lon[mxy[i], mxm[i]], lat[mxy[i], mxm[i]]), xycoords=ccrs.PlateCarree().as_mpl_transform(ax), color=color, size=20,
54                           clip_on=True, annotation_clip=True, horizontalalignment='center', verticalalignment='center',
55                           transform=ccrs.PlateCarree())
56
57         txt2 = ax.annotate('\n' + str(int(data[mxy[i], mxm[i]])), xy=(lon[mxy[i], mxm[i]], lat[mxy[i], mxm[i]]), xycoords=ccrs.PlateCarree().as_mpl_transform(ax),
58                           color=color, size=10, clip_on=True, annotation_clip=True, fontweight='bold', horizontalalignment='center', verticalalignment='top',
59                           transform=ccrs.PlateCarree())

```

Scipy's maximum and minimum filters

Creates annotations at the coordinates

Script 22: NWP + Satellite (Example 3)

```
61
62
63 # Input and output directories
64 input = "Samples"; os.makedirs(input, exist_ok=True)
65 output = "Output"; os.makedirs(output, exist_ok=True)
66
67 # Select the extent [min. lon, min. lat, max. lon, max. lat]
68 extent = [-93.0, -60.00, -25.00, 18.00]
69
70 # Datetime to process (today in this example, to match the GFS date)
71 #date = datetime.today().strftime('%Y%m%d')
72 #yyyymmddhhmm = date + '0000'
73 yyyymmddhhmm = '202107020000' # CHANGE THIS DATE TO THE SAME DATE OF YOUR NWP DATA
74
75 -----
76
77 # Download the ABI file
78 file_ir = download_CMI(yyyymmddhhmm, 13, input)
79
80
81 # Variable
82 var = 'CMI'
83
84 # Open the file
85 img = gdal.Open(f'NETCDF:{input}/{file_ir}.nc:' + var)
86
87 # Read the header metadata
88 metadata = img.GetMetadata()
89 scale = float(metadata.get(var + '#scale_factor'))
90 offset = float(metadata.get(var + '#add_offset'))
91 undef = float(metadata.get(var + '#_FillValue'))
92 dtim = metadata.get('NC_GLOBAL#time_coverage_start')
93
94 # Load the data
95 ds_cmi = img.ReadAsArray(0, 0, img.RasterXSize, img.RasterYSize).astype(float)
96
97 # Apply the scale, offset and convert to celsius
98 ds_cmi = (ds_cmi * scale + offset) - 273.15
99
100 # Reproject the file
101 filename_ret = f'{output}/IR_{yyyymmddhhmm}.nc'
102 reproject(filename_ret, img, ds_cmi, extent, undef)
103
104 # Open the reprojected GOES-R image
105 file = Dataset(filename_ret)
106
107 # Get the pixel values
108 data = file.variables['Band1'][:]
109
110
111 # Open the GRIB file
112 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
113
114 # Select the variable
115 prmsl = grib.select(name='Pressure reduced to MSL')[0]
```

SATELLITE DATA DOWNLOAD

SATELLITE DATA REPROJECTION AND CROPPING

Script 22: NWP + Satellite (Example 3)

Explaining the new instructions:

```
109
110
111
112 # Open the GRIB file
113 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
114
115 # Select the variable
116 prmls = grib.select(name='Pressure reduced to MSL')[0]
117
118 # Get information from the file
119 init = str(prmls.analDate) # Init date / time
120 run = str(prmls.hour).zfill(2) # Run
121 ftime = str(prmls.forecastTime) # Forecast hour
122 valid = str(prmls.validDate) # Valid date / time
123 print('Init: ' + init + ' UTC')
124 print('Run: ' + run + 'Z')
125 print('Forecast: ' + ftime)
126 print('Valid: ' + valid + ' UTC')
127
128 # Read the data for a specific region
129 prmls, lats, lons = prmls.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)
130
131 # Convert to hPa
132 prmls = prmls / 100
133
134 -----
135
136 # Select the variable
137 hght_1000 = grib.select(name='Geopotential Height', typeOfLevel = 'isobaricInhPa', level = 1000)[0]
138
139 # Read the data for a specific region
140 hght_1000 = hght_1000.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]
141
142 # Select the variable
143 hght_500 = grib.select(name='Geopotential Height', typeOfLevel = 'isobaricInhPa', level = 500)[0]
144
145 # Read the data for a specific region
146 hght_500 = hght_500.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]
147
148 # Calculate and smooth 1000-500 hPa thickness
149 thickness_1000_500 = hght_500 - hght_1000 ━━ Thickness (1000-500 hPa)
150
151
152 # Choose the plot size (width x height, in inches)
153 plt.figure(figsize=(8,8))
154
```

NWP DATA READING AND MANIPULATION

Thickness (1000-500 hPa)

Script 22: NWP + Satellite (Example 3)

Explaining the new instructions:

PLOT CONFIGURATION

```

152 # Choose the plot size (width x height, in inches)
153 plt.figure(figsize=(10,6))
154
155 # Use the Geostationary projection in cartopy
156 ax = plt.axes(projection=ccrs.PlateCarree())
157 ax.set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
158
159 # Define the image extent
160 img_extent = [extent[0], extent[2], extent[1], extent[3]]
161
162 # Define the color scale based on the channel
163 colormap = "gray_r" # White to black for IR channels
164
165 # Plot the image
166 img1 = ax.imshow(data, origin='upper', vmin=-80, vmax=60, extent=img_extent, cmap=colormap, alpha=1.0)
167
168 # Define de contour interval
169 data_min = 4900
170 data_max = 5900
171 interval = 20
172 levels = np.arange(data_min,data_max,interval)
173
174 # Plot the contours
175 img2 = ax.contour(lons, lats, thickness_1000_500, cmap='seismic', linestyles='dashed', linewidths=1.0, levels=levels)
176 ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10', fmt = '%1.0f') #, colors= 'black') # For the labels to have the same colors as the cmap, just omit the "colors" variable
177
178 # Get the index of elements with value "5400"
179 mid_value = int(np.where(levels == 5400)[0])
180 img2.collections[mid_value].set_linewidth(4)
181 img2.collections[mid_value].set_edgecolor('blue')
182
183 # Define de contour interval
184 data_min = 500
185 data_max = 1050
186 interval = 2
187 levels = np.arange(data_min,data_max,interval)
188
189 # Plot the contours
190 img3 = ax.contour(lons, lats, prmls, colors='black', linewidths=0.7, levels=levels)
191 ax.clabel(img3, inline=1, inline_spacing=0, fontsize='10', fmt = '%1.0f', colors= 'black')
192
193 # Add a shapefile
194 shapefile = list(shpreader.Reader('ne_10m_admin_1_states_provinces.shp').geometries())
195 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='white', facecolor='none', linewidth=0.3)
196
197 # Add coastlines, borders and gridlines
198 ax.coastlines(resolution='10m', color='white', linewidth=0.8)
199 ax.add_feature(cartopy.feature.BORDERS, edgecolor='white', linewidth=0.5)
200 gl = ax.gridlines(ccrs.PlateCarree(), color='white', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
201 gl.top_labels = False
202 gl.right_labels = False
203
204 # Use definition https://matplotlib.org/3.1.1/gallery/statistics/contour\_max\_min.html
205 plot_maxmin_points(lons, lats, prmls, 'max', 50, symbol='H', color='b', transform=ccrs.PlateCarree())
206 plot_maxmin_points(lons, lats, prmls, 'min', 25, symbol='L', color='r', transform=ccrs.PlateCarree())
207
208 # Extract date
209 date = (datetime.strptime(dtime, '%Y-%m-%dT%H:%M:%S.%fZ'))
210
211 # Add a title
212 plt.title('GOES-16 Band 13 ' + date.strftime('%Y-%m-%d %H:%M') + ' UTC' + ' ' + GFS_PMSL ('hPa') + 1000-500 hPa Thickness (m)', fontweight='bold', fontsize=6, loc='left')
213 plt.title('Reg.: ' + str(extent), fontsize=5, loc='right')
214

```

Satellite plot

Thickness plot

PSML Plot

Arrays

Max. or Min?

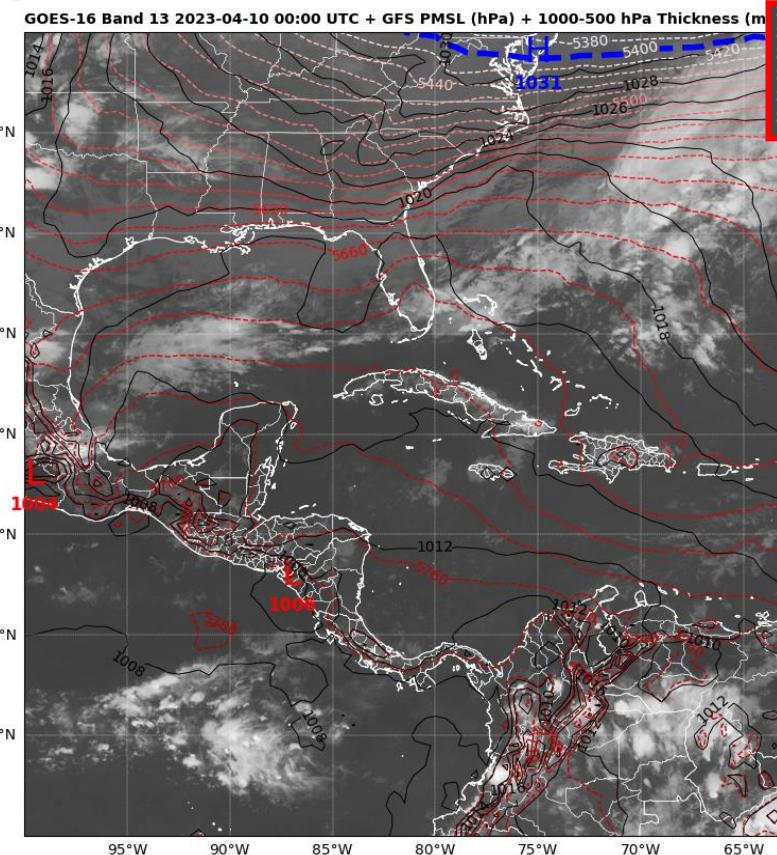
Color

Projection

Symbol

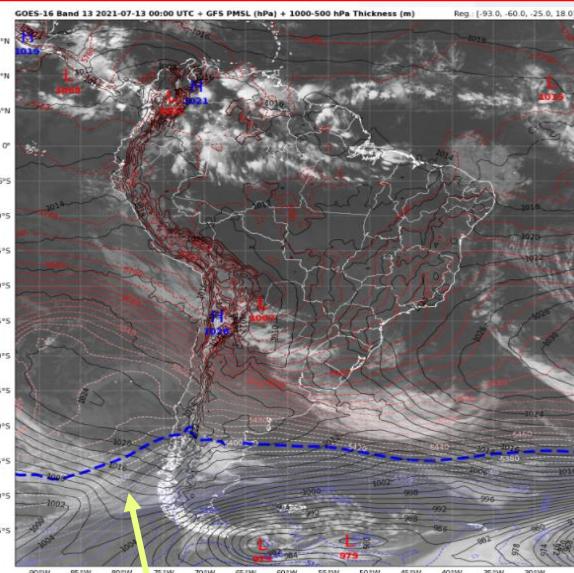
Call the function to plot the high and low pressure labels

Script 22: NWP + Satellite (Example 3)



```
215 # Save the image  
216 plt.savefig(f'{output}/image_22.png', bbox_inches='tight', pad_inches=0, dpi=300)  
217 # Show the image  
218 plt.show()
```

IMAGE GENERATION



Modified contour,
“5400”, in this
case

Some Quick Modifications

- Change the symbol to be plotted



Script 23: NWP + Satellite (Example 4)

LIBRARIES AND FUNCTIONS

SATELLITE DATA DOWNLOAD

SATELLITE DATA REPROJECTION AND CROPPING

NWP DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

```
1  -----
2  # INPE / CPTEC - Training: Python and GOES-R Imagery: Script 23 - Satellite + NWP Plot (Example 4)
3  # Author: Diego Souza
4
5  # Required modules
6  from netCDF4 import Dataset
7  from osgeo import gdal
8  import matplotlib.pyplot as plt
9  import cartopy, cartopy.crs as ccrs
10 import cartopy.io.shapereader as shapereader
11 import os
12 import numpy as np
13 from matplotlib import cm
14 from datetime import timedelta, date, datetime
15 from utilities import download_CMI
16 from utilities import reproject
17 from utilities import loadCPT
18 import pygrib
19 gdal.PushErrorHandler('CPLQuietErrorHandler')
20
21 # Input and output directories
22 input = "Samples"; os.makedirs(input, exist_ok=True)
23 output = "Output"; os.makedirs(output, exist_ok=True)
24
25 # Select the extent [min. lon, min. lat, max. lon, max. lat]
26 extent = [-93.0, -60.0, -25.00, 18.00]
27
28 # Datetime to process (today in this example, to match the GFS date)
29 #date = datetime.today().strftime('%Y%m%d')
30 #yyyymmddhhmm = date + '0000'
31 yyyymmddhhmm = '202107020000' # CHANGE THIS DATE TO THE SAME DATE OF YOUR NWP DATA
32
33
34
35 # Download the ABI file
36 file_ir = download_CMI(yyyymmddhhmm, 9, input)
37
38 -----
```

LIBRARIES AND FUNCTIONS

SATELLITE DATA DOWNLOAD

Script 23: NWP + Satellite (Example 4)

```
34
35 # Download the ABI file
36 file_ir = download_CMI(yyyymmddhhmn, 9, input)
37
38 # Variable
39 var = 'CMI'
40
41 # Open the file
42 img = gdal.Open(f'NETCDF:{input}/{file_ir}.nc:' + var)
43
44 # Read the header metadata
45 metadata = img.GetMetadata()
46 scale = float(metadata.get(var + '#scale_factor'))
47 offset = float(metadata.get(var + '#add_offset'))
48 undef = float(metadata.get(var + '#_FillValue'))
49 dtime = metadata.get('NC_GLOBAL#time_coverage_start')
50
51 # Load the data
52 ds_cmi = img.ReadAsArray(0, 0, img.RasterXSize, img.RasterYSize).astype(float)
53
54 # Apply the scale, offset and convert to celsius
55 ds_cmi = (ds_cmi * scale + offset) - 273.15
56
57 # Reproject the file
58 filename_ret = f'{output}/IR_{yyyymmddhhmn}.nc'
59 reproject(filename_ret, img, ds_cmi, extent, undef)
60
61 # Open the reprojected GOES-R image
62 file = Dataset(filename_ret)
63
64 # Get the pixel values
65 data = file.variables['Band1'][:]
66 #-----
67
68 # Open the GRIB file
69 grib = pvarib.open("afs.t00z.porb2full.0p50.f000")
```

SATELLITE DATA REPROJECTION AND CROPPING

Script 23: NWP + Satellite (Example 4)

```
65 # Get the pixel values
66 data = file.variables['Band1'][:]
67 #
68 #
69 # Open the GRIB file
70 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
71 #
72 # Select the variable
73 ucomp = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 250)[0]
74 #
75 # Get information from the file
76 init = str(ucomp.analDate)      # Init date / time
77 run = str(ucomp.hour).zfill(2)   # Run
78 ftime = str(ucomp.forecastTime)  # Forecast hour
79 valid = str(ucomp.validDate)    # Valid date / time
80 print('Init: ' + init + ' UTC')
81 print('Run: ' + run + 'Z')
82 print('Forecast: ' + ftime)
83 print('Valid: ' + valid + ' UTC')
84 #
85 # Read the data for a specific region
86 ucomp, lats, lons = ucomp.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)
87 #
88 #
89 # Select the variable
90 vcomp = grib.select(name='V component of wind', typeOfLevel = 'isobaricInhPa', level = 250)[0]
91 #
92 # Read the data for a specific region
93 vcomp = vcomp.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]
94 #
95 #
96 #
97 # Calculate the wind speed
98 ws = np.sqrt(ucomp**2 + vcomp**2)
99 #
100 "
```

NWP DATA READING AND MANIPULATION

Script 23: NWP + Satellite (Example 4)

```
101
102
103 # Choose the plot size (width x height, in inches)
104 plt.figure(figsize=(8,8))
105
106 # Use the Geostationary projection in cartopy
107 ax = plt.axes(projection=ccrs.PlateCarree())
108
109 # Define the image extent
110 img_extent = [extent[0], extent[2], extent[1], extent[3]]
111 ax.set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
112
113 # Converts a CPT file to be used in Python
114 cpt = loadCPT('SVCANWV_TEMP.cpt')
115 cmap = cm.colors.LinearSegmentedColormap('cpt', cpt)
116
117 # Plot the image
118 img1 = ax.imshow(data, origin='upper', vmin=-112.15, vmax=77.00, extent=img_extent, cmap=cmap, alpha=1.0)
119
120 # Define de contour interval
121 data_min = -20
122 data_max = 48
123 interval = 4
124 levels = np.arange(data_min,data_max,interval)
125
126 # Plot the wind vectors
127 img3 = ax.quiver(lons[:,::4,::4], lats[:,::4,::4], ucomp[:,::4,::4], vcomp[:,::4,::4], color='white')
128
129 # Add a shapefile
130 # https://geoftp.ibge.gov.br/organizacao-do-territorio/malhas-territoriais/malhas-municipais/municipio-2019/Brasil/BR/br_unidades-da-federacao.zip
131 shapefile = list(shapereader.Reader('BR_UF_2019.shp').geometries())
132 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='white', facecolor='none', linewidth=0.3)
133
134 # Add coastlines, borders and gridlines
135 ax.coastlines(resolution='10m', color='white', linewidth=0.8)
136 ax.add_feature(cartopy.feature.BORDERS, edgecolor='white', linewidth=0.5)
137 gl = ax.gridlines(ccrs.PlateCarree(), color='white', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
138 gl.top_labels = False
139 gl.right_labels = False
140
141 # Add a colorbar
142 plt.colorbar(img3, label='Wind Speed (kt)', extend='both', orientation='vertical', pad=0.03, fraction=0.05)
143 plt.colorbar(img1, label='Brightness Temperatures (°C)', extend='both', orientation='vertical', pad=0.03, fraction=0.05)
144
145 # Extract date
146 date = (datetime.strptime(dttime, '%Y-%m-%dT%H:%M:%S.%fZ'))
147
148 # Add a title
149 plt.title('GOES-16 Band 09 ' + date.strftime('%Y-%m-%d %H:%M') + ' UTC' + ' GFS Streamlines (250 hPa)', fontweight='bold', fontsize=6, loc='left')
150 plt.title('Reg.: ' + str(extent), fontsize=6, loc='right')
151
152 # Save the image
153 plt.savefig('output/image_23.png', bbox_inches='tight', pad_inches=0, dpi=300)
154
155 # Show the image
156 plt.show()
```

Explaining the new instructions:

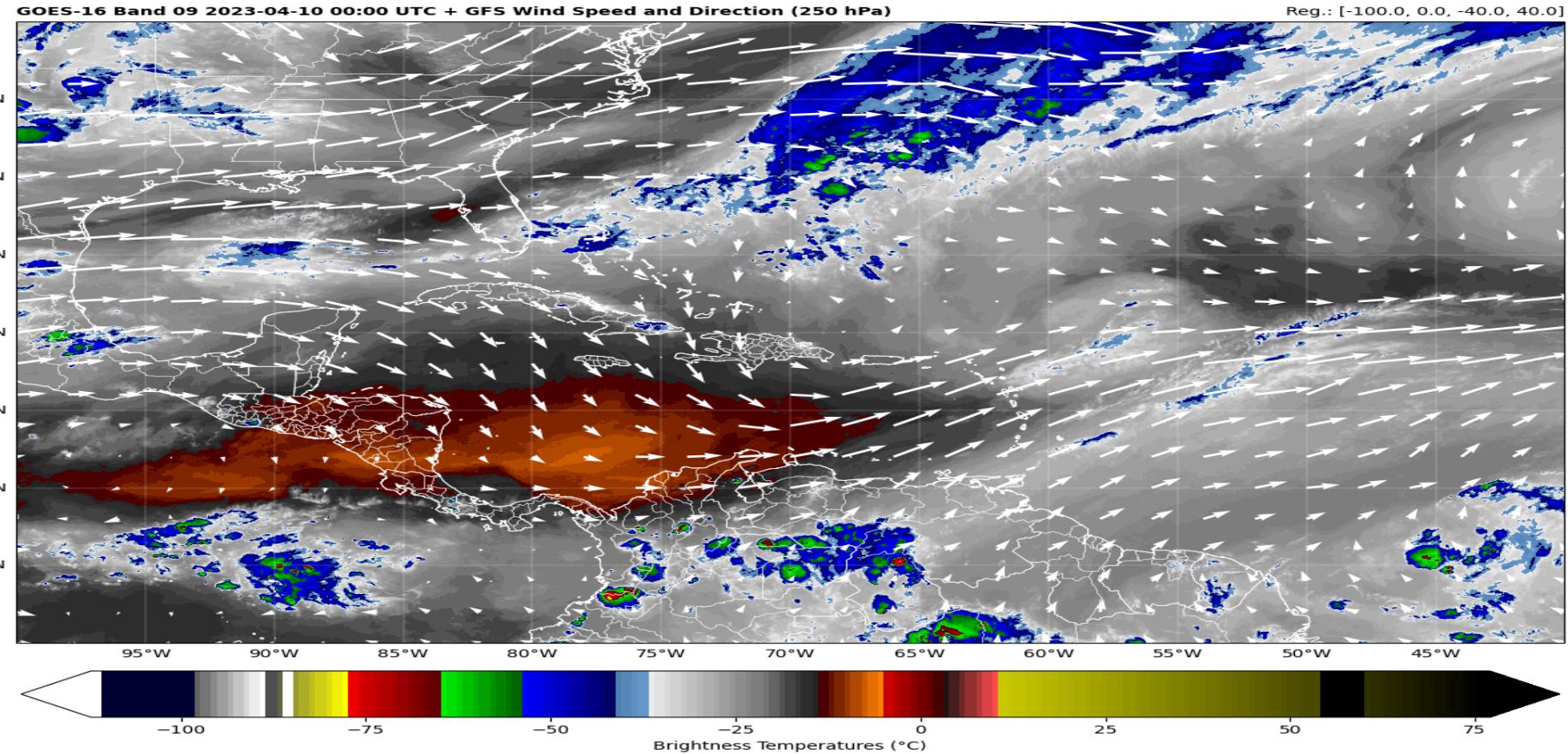
PLOT CONFIGURATION

Satellite plot

Wind vector plot

IMAGE GENERATION

Script 23: NWP + Satellite (Example 4)



Script 24: NWP + Satellite (Example 5)

Air Mass RGB Quick Guide

Why is the Air Mass RGB imagery Important?

The Air Mass RGB is used to diagnose the environment surrounding synoptic systems by enhancing temperature and moisture characteristics of air masses. Cyclogenesis can be inferred by the identification of warm, dry, ozone-rich descending stratospheric air associated with jet streams and potential vorticity (PV) anomalies. The RGB can be used to validate the location of PV anomalies in model data. Additionally, this RGB can distinguish between polar and tropical air masses, especially along upper-level frontal boundaries and identify high-, mid-, and low-level clouds.

Air Mass RGB Recipe

Color	Band / Band Diff. (μm)	Min – Max Gamma	Physically Relates to...	*Small input to pixel indicates...	*Large input to pixel indicates...
Red	6.2 – 7.3	-26.2 to 0.6 C 1	Vertical water vapor difference	Moist upper levels	Dry upper levels
Green	9.6 – 10.3	-43.2 to 6.7 C 1	Tropopause height based on ozone	Low trop and high ozone	High trop and low ozone
Blue	6.2 (Inverted)	-29.25 to -64.65 C 1	Water vapor ~200-500 mb	Dry upper levels	Moist upper levels

Impact on Operations

Primary Application: Inferring cyclogenesis: It is easy to see jet streams and stratospheric air intrusions with high PV, and the cyclonic activity created by these dynamics. Can also track cyclogenesis as shortwaves approach and low- to mid- level clouds form, evolve, and rotate.

Identifying air masses: Polar and tropical air masses are readily seen in the RGB imagery.

Secondary Applications: Upper level moisture boundaries. Distinguishing between warm air masses with high and relatively low moisture, high clouds and mid-level clouds. Inferring turbulence by identifying stratospheric intrusion.

Limitations

Limb effects: The use of longer wavelength channels results in more atmospheric absorption at large viewing angles. As a result of the greater absorption, cooler brightness temperatures are measured. This limb cooling causes false blue and violet colors along the entire limb. Tropical air can appear blue rather than green at the limb.

Upper troposphere only: Conditions in the mid- to upper troposphere can be detected but surface conditions cannot be directly observed.

Intense Day-time heating: red/orange coloring is observed over dry desert regions during the summer; these dry upper levels don't indicate anomalous PV.

Contributor: Dr. Emily Berndt NASA SPoRT <https://weather.msfc.nasa.gov/sport/>

SPoRT

Air Mass RGB Recipe

Color	Band / Band Diff. (μm)	Min – Max Gamma
Red	6.2 – 7.3	-26.2 to 0.6 C 1
Green	9.6 – 10.3	-43.2 to 6.7 C 1
Blue	6.2 (inverted)	-29.25 to -64.65 C 1



Script 24: NWP + Satellite (Example 5)

LIBRARIES AND FUNCTIONS

FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS

DOWNLOAD 4 GOES-16 BANDS

REP. AND CROPPING CHANNEL 8

REP. AND CROPPING CHANNEL 10

REP. AND CROPPING CHANNEL 12

REP. AND CROPPING
CHANNEL 13

RGB RECIPE

NWP DATA READING AND PROCESSING

PLOT CONFIGURATION

IMAGE GENERATION

LIBRARIES AND FUNCTIONS

FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS

Script 24: NWP + Satellite (Example 5)

LIBRARIES AND FUNCTIONS

FUNCTION TO PLOT HIGH
AND LOW PRESSURE
LABELS

DOWNLOAD 4 GOES-16
BANDS

REP. AND CROPPING
CHANNEL 8

REP. AND CROPPING
CHANNEL 10

REP. AND CROPPING
CHANNEL 12

REP. AND CROPPING
CHANNEL 13

RGB RECIPE

NWP DATA READING AND
PROCESSING

PLOT CONFIGURATION

IMAGE GENERATION

```
61 #
62 # Select the extent [min. lon, min. lat, max. lon, max. lat]
63 extent = [-93.0, -60.00, -25.00, 18.00]
64
65
66 # Input and output directories
67 input = "Samples"; os.makedirs(input, exist_ok=True)
68 output = "Output"; os.makedirs(output, exist_ok=True)
69
70 # Datetime to process (today in this example, to match the GFS date)
71 #date = datetime.today().strftime('%Y%m%d')
72 #yyyymmddhhmn = date + '0000'
73 yyyymmddhhmn = '202107020000' # CHANGE THIS DATE TO THE SAME DATE OF YOUR NWP DATA
74
75 -----
76 # Download the ABI file
77 file_ir_8 = download_CMI(yyyymmddhhmn, 8, input)
78
79 # Download the ABI file
80 file_ir_10 = download_CMI(yyyymmddhhmn, 10, input)
81
82 # Download the ABI file
83 file_ir_12 = download_CMI(yyyymmddhhmn, 12, input)
84
85 # Download the ABI file
86 file_ir_13 = download_CMI(yyyymmddhhmn, 13, input)
87 #
```

DOWNLOAD 4 GOES-16 BANDS (AIRMASS)

Script 24: NWP + Satellite (Example 5)

LIBRARIES AND FUNCTIONS

FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS

DOWNLOAD 4 GOES-16 BANDS

REP. AND CROPPING CHANNEL 8

REP. AND CROPPING CHANNEL 10

REP. AND CROPPING CHANNEL 12

REP. AND CROPPING CHANNEL 13

RGB RECIPE

NWP DATA READING AND PROCESSING

PLOT CONFIGURATION

IMAGE GENERATION

```
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126

    # Variable
    var = 'CMI'

    # Open the file
    img = gdal.Open(f'NETCDF:{input}/{file_ir_8}.nc:' + var)

    # Read the header metadata
    metadata = img.GetMetadata()
    scale = float(metadata.get(var + '#scale_factor'))
    offset = float(metadata.get(var + '#add_offset'))
    undef = float(metadata.get(var + '#_FillValue'))
    dtim = metadata.get('NC_GLOBAL#time_coverage_start')

    # Load the data
    ds_cmi = img.ReadAsArray(0, 0, img.RasterXSize, img.RasterYSize).astype(float)

    # Apply the scale, offset and convert to celsius
    ds_cmi = (ds_cmi * scale + offset) - 273.15

    # Reproject the file
    filename_ret = f'{output}/IR_{yyyymmddhhmm}.nc'
    reproject(filename_ret, img, ds_cmi, extent, undef)

    # Open the reprojected GOES-R image
    file = Dataset(filename_ret)

    # Get the pixel values
    data_08 = file.variables['Band1'][:]

    # Open the file
    img = gdal.Open(f'NETCDF:{input}/{file_ir_10}.nc:' + var)

    # Read the header metadata
    metadata = img.GetMetadata()
    scale = float(metadata.get(var + '#scale_factor'))
    offset = float(metadata.get(var + '#add_offset'))
    undef = float(metadata.get(var + '#_FillValue'))
    dtim = metadata.get('NC_GLOBAL#time_coverage_start')
```

Explaining the new instructions:

REPROJECTION AND CROPPING (4 BANDS)

Script 24: NWP + Satellite (Example 5)

LIBRARIES AND FUNCTIONS

```
FUNCTION TO PLOT HIGH  
AND LOW PRESSURE  
LABELS
```

```
DOWNLOAD 4 GOES-16  
BANDS
```

```
REP. AND CROPPING  
CHANNEL 8
```

```
REP. AND CROPPING  
CHANNEL 10
```

```
REP. AND CROPPING  
CHANNEL 12
```

```
REP. AND CROPPING  
CHANNEL 13
```

```
RGB RECIPE
```

```
NWP DATA READING AND  
PROCESSING
```

```
PLOT CONFIGURATION
```

```
IMAGE GENERATION
```

```
# Get the pixel values  
data_13 = file.variables['Band1'][:]  
#--  
# RGB Components  
R = data_08 - data_10  
G = data_12 - data_13  
B = data_08  
  
# Minimuns and Maximuns  
Rmin = -26.2  
Rmax = 0.6  
  
Gmin = -43.2  
Gmax = 6.7  
  
Bmin = -29.25  
Bmax = -64.65  
  
R[R<Rmin] = Rmin  
R[R>Rmax] = Rmax  
  
G[G<Gmin] = Gmin  
G[G>Gmax] = Gmax  
  
B[B<Bmax] = Bmax  
B[B>Bmin] = Bmin  
  
# Choose the gamma  
gamma = 1  
  
# Normalize the data  
R = ((R - Rmin) / (Rmax - Rmin)) ** (1/gamma)  
G = ((G - Gmin) / (Gmax - Gmin)) ** (1/gamma)  
B = ((B - Bmin) / (Bmax - Bmin)) ** (1/gamma)  
  
# Create the RGB  
RGB = np.stack([R, G, B], axis=2)  
#--
```

Explain the new instructions:

— Calculations from recipe

Minimuns and Maximuns

Gamma

Normalization

Create the stack

RGB RECIPE (AIRMASS)

Air Mass RGB Recipe

Color	Band / Band Diff. (μm)	Min – Max Gamma
Red	6.2 – 7.3	-26.2 to 0.6 C 1
Green	9.6 – 10.3	-43.2 to 6.7 C 1
Blue	6.2 (inverted)	-29.25 to -64.65 C 1

Script 24: NWP + Satellite (Example 5)

LIBRARIES AND FUNCTIONS

FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS

DOWNLOAD 4 GOES-16 BANDS

REP. AND CROPPING CHANNEL 8

REP. AND CROPPING CHANNEL 10

REP. AND CROPPING CHANNEL 12

REP. AND CROPPING CHANNEL 13

RGB RECIPE

NWP DATA READING AND PROCESSING

PLOT CONFIGURATION

IMAGE GENERATION

```
222 # Normalize the data
223 R = ((R - Rmin) / (Rmax - Rmin)) ** (1/gamma)
224 G = ((G - Gmin) / (Gmax - Gmin)) ** (1/gamma)
225 B = ((B - Bmin) / (Bmax - Bmin)) ** (1/gamma)
226
227 # Create the RGB
228 RGB = np.stack([R, G, B], axis=2)
229 #-----
230 #-----
231
232 # Open the GRIB file
233 grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
234
235 # Select the variable
236 prmls = grib.select(name='Pressure reduced to MSL')[0]
237
238 # Get information from the file
239 init = str(prmls.analDate) # Init date / time
240 run = str(prmls.hour).zfill(2) # Run
241 ftime = str(prmls.forecastTime) # Forecast hour
242 valid = str(prmls.validDate) # Valid date / time
243 print('Init: ' + init + ' UTC')
244 print('Run: ' + run + 'Z')
245 print('Forecast: ' + ftime)
246 print('Valid: ' + valid + ' UTC')
247
248 # Read the data for a specific region
249 prmls, lats, lons = prmls.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)
250
251 # Convert to hPa
252 prmls = prmls / 100
253 #-----
254 # Choose the plot size (width x height, in inches)
255 plt.figure(figsize=(8,8))
256
257 # Use the Geostationary projection in cartopy
258 ax = plt.axes(projection=ccrs.PlateCarree())
259
260
```

NWP DATA READING AND MANIPULATION

Script 24: NWP + Satellite (Example 5)

LIBRARIES AND FUNCTIONS

FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS

DOWNLOAD 4 GOES-16 BANDS

REP. AND CROPPING CHANNEL 8

REP. AND CROPPING CHANNEL 10

REP. AND CROPPING CHANNEL 12

REP. AND CROPPING CHANNEL 13

RGB RECIPE

NWP DATA READING AND PROCESSING

PLOT CONFIGURATION

IMAGE GENERATION

```

254
255     # Choose the plot size (width x height, in inches)
256     plt.figure(figsize=(8,8))
257
258     # Use the Geostationary projection in cartopy
259     ax = plt.axes(projection=ccrs.PlateCarree())
260
261     # Define the image extent
262     img_extent = [extent[0], extent[2], extent[1], extent[3]]
263
264
265     # Plot the image
266     img1 = ax.imshow(RGB, origin='upper', extent=img_extent, alpha=1.0)
267
268     # Define de contour interval
269     data_min = 990
270     data_max = 1050
271     interval = 2
272     levels = np.arange(data_min,data_max,interval)
273
274     # Plot the contours
275     img2 = ax.contour(lons, lats, prmls, colors='cyan', linewidths=0.7, levels=levels)
276     ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10', fmt = '%1.0f', colors= 'cyan')
277
278     # Add a shapefile
279     # https://geoftp.ibge.gov.br/organizacao do territorio/malhas territoriais/malhas municipais/municipio_2019/Brasil/BR_unidades_da_federacao.zip
280     shapefile = list(shapereader.Reader('BR_UF_2019.shp').geometries())
281     ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='white', facecolor='none', linewidth=0.3)
282
283     # Add coastlines, borders and gridlines
284     ax.coastlines(resolution='10m', color='white', linewidth=0.8)
285     ax.add_feature(cartopy.feature.BORDERS, edgecolor='white', linewidth=0.5)
286     gl = ax.gridlines(ccrs.PlateCarree(), color='white', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
287     gl.top_labels = False
288     gl.right_labels = False
289
290     # Use definition to plot H/L symbols
291     plot_maxmin_points(lons, lats, prmls, 'max', 50, symbol='H', color='b', transform=ccrs.PlateCarree())
292     plot_maxmin_points(lons, lats, prmls, 'min', 25, symbol='L', color='r', transform=ccrs.PlateCarree())
293
294     # Extract date
295     date = (datetime.strptime(dtime, '%Y-%m-%d %H:%M') + timedelta(hours=4))
296
297     # Add a title
298     plt.title('GOES-16 Airmass RGB ' + date.strftime('%Y-%m-%d %H:%M') + ' UTC' + ' ' + GFS_PSLM ('hPa)', fontweight='bold', fontsize=6, loc='left')
299     plt.title('Reg.: ' + str(extent), fontsize=6, loc='right')
300
301     # Save the image
302     plt.savefig(f'{output}/image_24.png', bbox_inches='tight', pad_inches=0, dpi=300)
303
304     # Show the image
305     plt.show()

```

Explaining the new instructions:

NWP PLOT CONFIGURATION

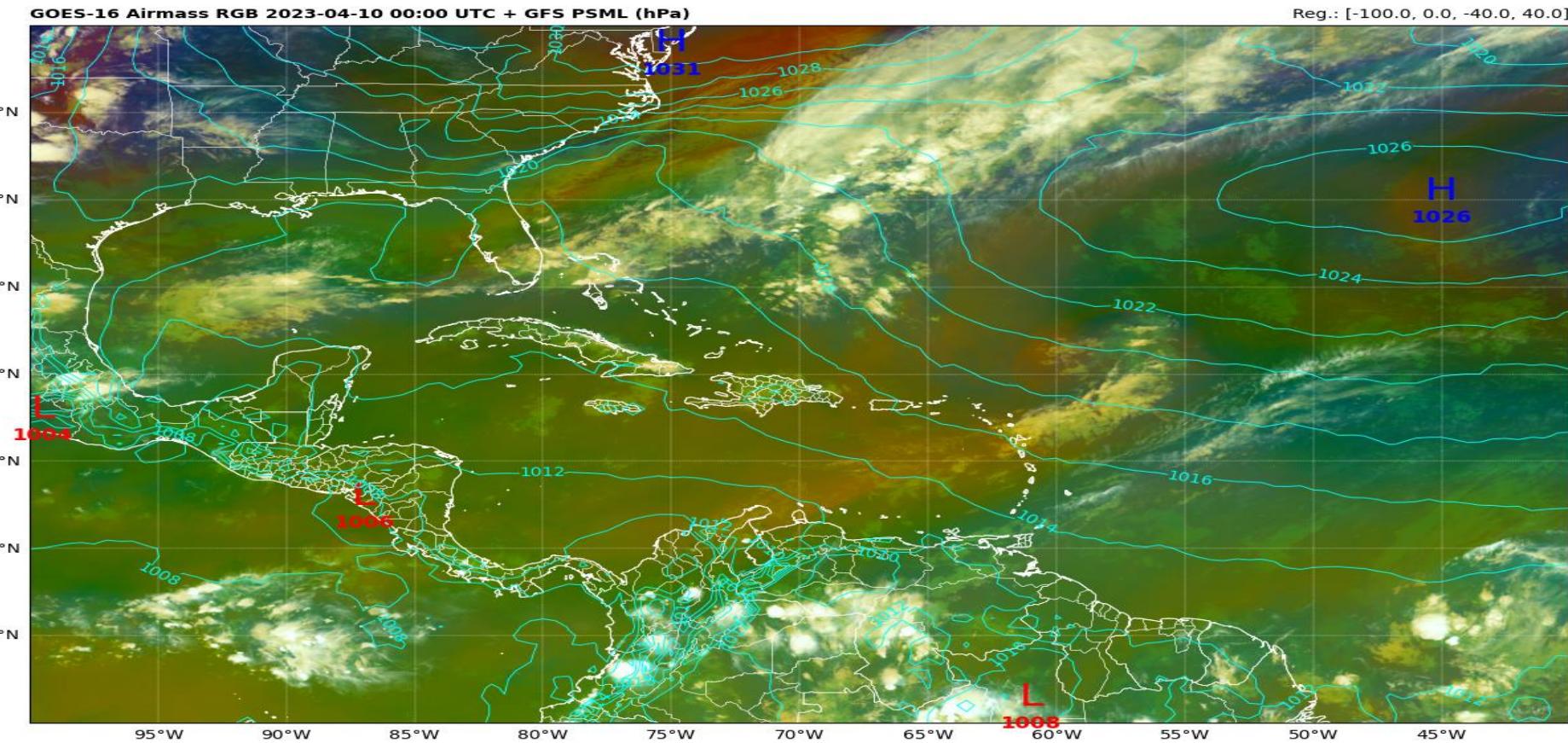
RGB Plot

PSML Plot

Calling the function to plot high and low pressure labels

IMAGE GENERATION

Script 24: NWP + Satellite (Example 5)



Script 25: METAR Plot

```

# INPE / CPTEC - Training: Python and METPy Examples: Script 25 - METAR Plot

# Libraries
import requests
import cartopy
import cartopy.io.shapereader as shapereader
import cartopy.feature as cfeature
import os
import numpy as np
import metpy
from datetime import timedelta, date, datetime
from metpy.calc import reduce_point_density
from metpy.io import metar
from metpy.plots import current_weather, sky_cover, StationPlot

# Set the extent [min. lon, min. lat, max. lon, max. lat]
extent = [-45.0, -10.0, -60.0, -25.0]

# Download the METAR file
url = 'https://thredds-test.unidata.ucar.edu/thredds/fileServer/noaa/0/text/metar'
file_name = 'metar_20210702_0000.txt'

# Send a GET request to the specified url
myfile = requests.get(url + '/' + file_name)

# Download the file
open(dir + '/' + file_name, 'wb').write(myfile.content)

# METAR Data
data = metar.parse_metar_file(dir + '/' + file_name)

# Drop rows with missing winds
data = data.dropna(how='any', subset=['wind_direction', 'wind_speed'])

# Discard missing data
data = data.dropna(how='any', subset=['wind_direction', 'wind_speed'])

# Plotting library
# Plot maps
# Import shapefiles
# Drawing and filtering operations
# Miscellaneous operating system interfaces
# Basic Dates and time types
# Provide tools for unit-aware, meteorological calculations
# Contains functionality for making meteorological plots

```

LIBRARIES

METAR DATA DOWNLOAD

PLOT CONFIGURATION

METAR PLOT

IMAGE GENERATION

```

# INPE / CPTEC - Training: Python and METPy Examples: Script 25 - METAR Plot
# Author: Diego Souza

# Adapted from: https://unidata.github.io/MetPy/latest/examples/plots/Station\_Plot.html

# Plotting library
# Plot maps
# Import shapefiles
# Drawing and filtering operations
# Miscellaneous operating system interfaces
# Basic Dates and time types
# Provide tools for unit-aware, meteorological calculations
# Contains functionality for making meteorological plots

# requests' library
# (the same used to download GFS data)
# Funções da Biblioteca MetPy

# Select the extent [min. lon, min. lat, max. lon, max. lat]
extent = [-93.0, -60.0, -25.0, 18.0]

# Input and output directories
dir = "Samples"; os.makedirs(dir, exist_ok=True)
output = "Output"; os.makedirs(output, exist_ok=True)

# Download the METAR File
#date = datetime.today().strftime('%Y%m%d')
date = '20210702' # CHANGE THIS DATE TO THE SAME DATE OF YOUR NWP DATA
url = 'https://thredds-test.unidata.ucar.edu/thredds/fileServer/noaa/0/text/metar'
file_name = 'metar_' + date + '_0000.txt'

# Sends a GET request to the specified url
myfile = requests.get(url + '/' + file_name)

# Download the file
open(dir + '/' + file_name, 'wb').write(myfile.content)

# METAR Data
# https://unidata.github.io/MetPy/latest/examples/plots/Station_Plot.html
data = metar.parse_metar_file(dir + '/' + file_name)

# Drop rows with missing winds
data = data.dropna(how='any', subset=['wind_direction', 'wind_speed'])

# Discard missing data

```

LIBRARIES

Desired date

Desired time

Data download

Read the METAR data

Note: In COLAB, the date and time are the date of the script execution

Discard missing data

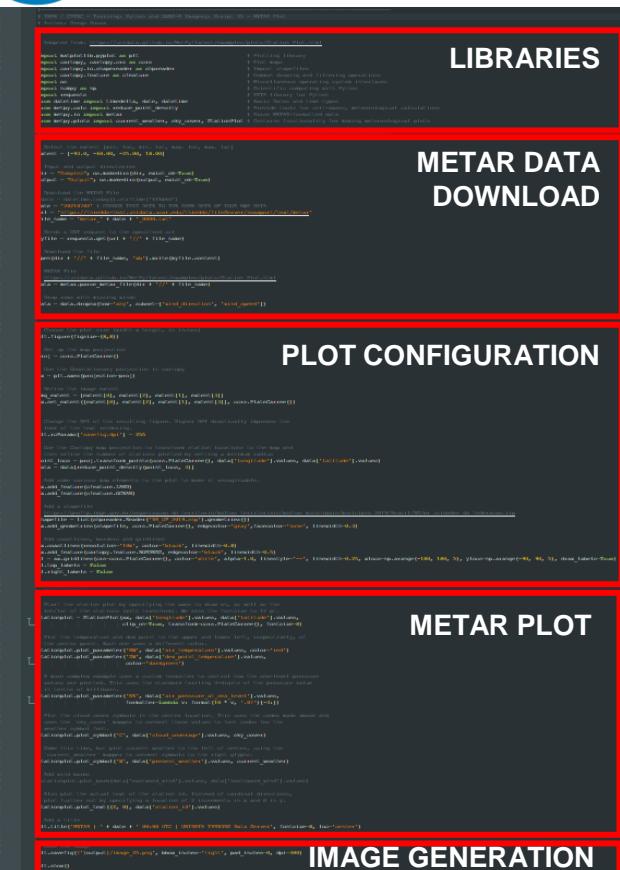


Script 25: METAR Plot

```
43 data = metar.parse_metar_file(dir + '//' + file_name)
44
45 # Drop rows with missing winds
46 data = data.dropna(how='any', subset=['wind_direction', 'wind_speed'])
47
48
49 # Choose the plot size (width x height, in inches)
50 plt.figure(figsize=(8,8))
51
52 # Set up the map projection
53 proj = ccrs.PlateCarree()
54
55 # Use the Geostationary projection in cartopy
56 ax = plt.axes(projection=proj)
57
58 # Define the image extent
59 img_extent = [extent[0], extent[2], extent[1], extent[3]]
60 ax.set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
61
62
63 # Change the DPI of the resulting figure. Higher DPI drastically improves the
64 # look of the text rendering.
65 plt.rcParams['savefig.dpi'] = 255
66
67
68 # Use the Cartopy map projection to transform station locations to the map and
69 # then refine the number of stations plotted by setting a minimum radius
70 point_locs = proj.transform_points(ccrs.PlateCarree(), data['longitude'].values, data['latitude'].values)
71 data = data[reduce_point_density(point_locs, 3)]
72
73 # Add some various map elements to the plot to make it recognizable.
74 ax.add_feature(cfeature.LAND)
75 ax.add_feature(cfeature.OCEAN)
76
77 # Add a shapefile
78 # https://geotpbge.gov.br/organizacao-do-territorio/malhas-territoriais/malhas-municipais/municipio\_2019/Brasil/BR\_BR\_UN
79 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
80 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
81
82 # Add coastlines, borders and gridlines
83 ax.coastlines(resolution='10m', color='black', linewidth=0.8)
84 ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
85 gl = ax.gridlines(ccrs.PlateCarree(), color='white', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 30), ylocs=np.arange(-90, 90, 30))
86 gl.top_labels = False
87 gl.right_labels = False
88
89
90 # Station Plot
91
92 # Start the station plot by specifying the axes to draw on, as well as the
93 # lon/lat of the stations (with transform). We also the fontsize to 12 pt.
94 stationplot = StationPlot(ax, data['longitude'].values, data['latitude'].values,
```



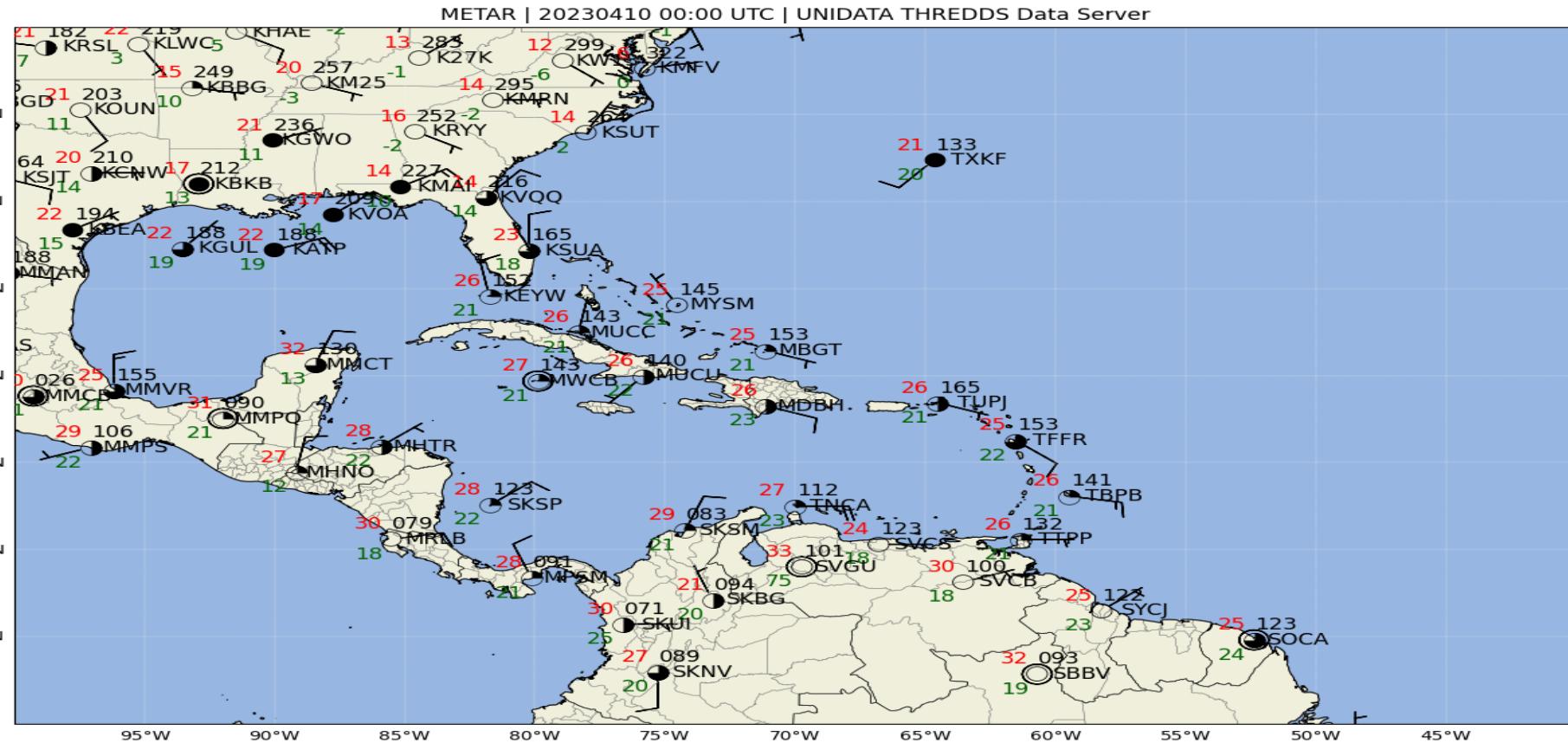
Script 25: METAR Plot



```
gl.right_labels = False
#
# Station Plot
# Start the station plot by specifying the axes to draw on, as well as the
# location of the stations (with transform). We also the fontsize (8pt).
stationplot = StationPlot(ax, data['longitude'].values, data['latitude'].values,
                           clip_on=True, transform=ccrs.PlateCarree(), fontsize=8)
#
Indicates which axis to plot on and
the METAR lon/lat variables
#
Plot the temperature and Dew
# Plot the temperature and dew point to the upper and lower left, respectively, of
# the center Point on the left uses a different color.
stationplot.plot_parameter('NW', data['air_temperature'].values, color='red')
stationplot.plot_parameter('SW', data['dew_point_temperature'].values,
                           color='darkgreen')
#
Plot the pressure at sea level on the right
# A more custom formatter to control how the sea-level pressure
# values are plotted. This uses the standard trailing 3-digits of the pressure value
# in tenths of millibars.
stationplot.plot_parameter('NE', data['air_pressure_at_sea_level'].values,
                           formatter=lambda v: format(10 * v, '.0f')[-3:])
#
Plot the cloud cover
# Plot the cloud coverage at the center location. This uses the codes made above and
# uses the `sky_cover` mapper to convert these values to font codes for the
# weather symbol font.
stationplot.plot_symbol('C', data['cloud_coverage'].values, sky_cover)
#
Plot the current weather
# Same this time, but plot current weather to the left of center, using the
# `current_weather` mapper to convert symbols to the font code.
stationplot.plot_symbol('W', data['current_weather_symbol'].values, current_weather)
#
Plot the wind barbs
# Add wind barbs
stationplot.plot_barb(data['eastward_wind'].values, data['northward_wind'].values)
#
# Also plot the actual text of the station id. Instead of carrying instructions,
# plot further out by specifying a location of 2 increments in x and 0 in y.
stationplot.plot_text((2, 0), data['station_id'].values)
#
Station ID
#
Plot the title
# Add a title
plt.title('METAR | ' + date + ' 00:00 UTC | UNIDATA THREDDS Data Server', fontsize=8, loc='center')
#
# Save the image
plt.savefig(f'{output}/image_25.png', bbox_inches='tight', pad_inches=0.1)
plt.show()
```

IMAGE GENERATION

Script 25: METAR Plot



Script 25: METAR Plot

<https://unidata.github.io/MetPy/latest/api/generated/metpy.plots.StationPlot.html>

Current Weather Symbols

0	1	2	3	4	5	6	7	8	9	10	11
≡≡	<	⊕)•((•)	ꝝ	ꝝ	(ꝝ)	=	≡≡		
12	13	14	15	16	17	18	19	20	21	22	23
~]	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ
24	25	26	27	28	29	30	31	32	33	34	35
+	+	+	+	(≡)	≡≡	≡≡	≡≡	≡≡	≡≡	≡≡	≡≡
36	37	38	39	40	41	42	43	44	45	46	47
ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ
48	49	50	51	52	53	54	55	56	57	58	59
•	••	••	••	••	••	••	••	•*	**		
60	61	62	63	64	65	66	67	68	69	70	71
*	**	**	**	**	**	**	**	**	**	**	**
72	73	74	75	76	77	78	79	80	81	82	83
ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ
84	85	86	87	88	89	90	91	92	93	94	95
ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ
96	97	98	99								

Current Weather Auto Reported Symbols

0	1	2	3	4	5	6	7	8	9	10	11
ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ
12	13	14	15	16	17	18	19	20	21	22	23
ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ
24	25	26	27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44	45	46	47
ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ
48	49	50	51	52	53	54	55	56	57	58	59
○	••	••	••	••	••	••	••	•*	**		
60	61	62	63	64	65	66	67	68	69	70	71
**	**	**	**	**	**	**	**	**	**	**	**
72	73	74	75	76	77	78	79	80	81	82	83
ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ
84	85	86	87	88	89	90	91	92	93	94	95
ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ
96	97	98	99								

Low Cloud Symbols

0	1	2	3	4	5	6	7	8	9
ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ

Mid Cloud Symbols

0	1	2	3	4	5	6	7	8	9
ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ

High Cloud Symbols

0	1	2	3	4	5	6	7	8	9
ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ

Sky Cover Symbols

0	1	2	3	4	5	6	7	8	9	10	11
ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ

Pressure Tendency Symbols

0	1	2	3	4	5	6	7	8	9
ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ	ꝝ

Some Quick Modifications

- Change the appearance of the METAR information (colors, etc.)



Script 26: METAR + NWP

```
# INPE / CPTEC - Training: Python and GOES-R Imagery: Script 26 - METAR + NWP Plot
# Author: Diego Souza
#
# Adapted from: https://unidata.github.io/MetPy/latest/examples/plots/Station_Plot.html

import matplotlib.pyplot as plt
import cartopy, cartopy.crs as ccrs
import cartopy.io.shapereader as shapereader
import cartopy.feature as cfeature
import os
import numpy as np
import requests
from datetime import timedelta, date, datetime
from metpy.calc import reduce_point_density
from metpy.io import metar
from metpy.plots import current_weather, sky_cover, StationPlot
import pygrib

# Plotting library
# Plot maps
# Import shapefiles
# Common drawing and filtering operations
# Miscellaneous operating system interfaces
# Scientific computing with Python
# Basic Dates and time types
# Provide tools for unit-aware, meteorological calculations
# Parse METAR-formatted data
# Contains functionality for making meteorological plots
# Provides a high-level interface to the ECMWF ECCODES C library for reading GRIB files

def plot_maxmin_points(lon, lat, data, extrema, nsizes, symbol, color='k', plotValue=True, transform=None):
    """
    This function will find and plot relative maximum and minimum values in a 2D array. The function can be used to plot an H for maximum values (e.g., High pressure) or an L for minimum values (e.g., low pressure). It is best to use filered data to obtain a synoptic scale max/min value. The symbol text can be set to a string value and optionally the color of the symbol and any plotted value can be set with the parameter color.
    lon = plotting longitude values (2D)
    lat = plotting latitude values (2D)
    data = 2D data that you wish to plot the max/min symbol placement
    extrema = Either a value of max for Maximum Values or min for Minimum Values
    nsizes = Size of the grid box to filter the max and min values to plot a reasonable symbol
    symbol = String to be placed at location of max/min value
    color = String matplotlib colorname to plot the symbol (and numeric value, if plotted)
    plot_value = Boolean (True/False) of whether to plot the numeric value of max/min point. The max/min symbol will be plotted on the current axes within the bounding frame (e.g., clip_on=True)
    """
    from scipy.ndimage.filters import maximum_filter, minimum_filter

    if (extrema == 'max'):
        data_ext = maximum_filter(data, nsizes, mode='nearest')
    elif (extrema == 'min'):
        data_ext = minimum_filter(data, nsizes, mode='nearest')
    else:
        raise ValueError('Value for hilo must be either max or min')

    mxy, mxm = np.where(data_ext == data)

    for i in range(len(mxy)):
        txt1 = ax.annotate(symbol, xy=(lon[mxy[i], mxm[i]], lat[mxy[i], mxm[i]]), xycoords=ccrs.PlateCarree().as_mpl_transform(ax), color=color, size=20, clip_on=True, annotation_clip=True, horizontalalignment='center', verticalalignment='center', transform=ccrs.PlateCarree())

        txt2 = ax.annotate('\n' + str(int(data[mxy[i], mxm[i]])), xy=(lon[mxy[i], mxm[i]], lat[mxy[i], mxm[i]]), xycoords=ccrs.PlateCarree().as_mpl_transform(ax), color=color, size=10, clip_on=True, annotation_clip=True, fontweight='bold', horizontalalignment='center', verticalalignment='top', transform=ccrs.PlateCarree())
```

LIBRARIES

FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS

NWP DATA READING AND MANIPULATION

METAR DATA DOWNLOAD

PLOT CONFIGURATION

METAR PLOT

IMAGE GENEATION

LIBRARIES

FUNCTION TO PLOT THE HIGH AND LOW PRESSURE LABELS

```
# INPE / CPTEC - Training: Python and GOES-R Imagery: Script 26 - METAR + NWP Plot
# Author: Diego Souza
#
# Adapted from: https://unidata.github.io/MetPy/latest/examples/plots/Station_Plot.html

import matplotlib.pyplot as plt
import cartopy, cartopy.crs as ccrs
import cartopy.io.shapereader as shapereader
import cartopy.feature as cfeature
import os
import numpy as np
import requests
from datetime import timedelta, date, datetime
from metpy.calc import reduce_point_density
from metpy.io import metar
from metpy.plots import current_weather, sky_cover, StationPlot
import pygrib

# Plotting library
# Plot maps
# Import shapefiles
# Common drawing and filtering operations
# Miscellaneous operating system interfaces
# Scientific computing with Python
# Basic Dates and time types
# Provide tools for unit-aware, meteorological calculations
# Parse METAR-formatted data
# Contains functionality for making meteorological plots
# Provides a high-level interface to the ECMWF ECCODES C library for reading GRIB files
```

Script 26: METAR + NWP

```
import metpy
from metpy import load_dataset
from metpy import units
from metpy.plots import HighLowPressureLabels
```

LIBRARIES

```
def plot_labels(grib):
    # Load the GRIB file
    grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
    # Select the variable
    prmls = grib.select(name='Pressure reduced to MSL')[0]
    # Get information from the file
    init = str(prmls.analDate) # Init date / time
    run = str(prmls.hour).zfill(2) # Run
    ftime = str(prmls.forecastTime) # Forecast hour
    valid = str(prmls.validDate) # Valid date / time
    print('Init: ' + init + ' UTC')
    print('Run: ' + run + 'Z')
    print('Forecast: ' + ftime)
    print('Valid: ' + valid + ' UTC')

    # Read the data for a specific region
    prmls, lats, lons = prmls.data(lat1=extent[1],lat2=extent[3],lonl=extent[0]+360,lon2=extent[2]+360)

    # Convert to hPa
    prmls = prmls / 100
```

FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS

```
# Set the extent [min. lon, min. lat, max. lon, max. lat]
extent = [-93.0, -60.00, -25.00, 18.00]

# Open the GRIB file
grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")

# Select the variable
prmls = grib.select(name='Pressure reduced to MSL')[0]
```

NWP DATA READING AND MANIPULATION

```
def download_metar():
    # Input and output directories
    dir = "Samples"; os.makedirs(dir, exist_ok=True)
    output = "Output"; os.makedirs(output, exist_ok=True)

    # Download the METAR File
    date = datetime.today().strftime('%Y%m%d')
    date = '20210702' # CHANGE THIS DATE TO THE SAME DATE OF YOUR NWP DATA
    url = "https://thredds-test.unidata.ucar.edu/thredds/fileServer/noaa/0/text/metar"
    file_name = 'metar_' + date + '_0000.txt'

    # Sends a GET request to the specified url
    myfile = requests.get(url + '//' + file_name)

    # Download the file
    open(dir + '//' + file_name, 'wb').write(myfile.content)

    # METAR File
    data = metar.parse_metar_file(dir + '//' + file_name)

    # Drop rows with missing winds
    data = data.dropna(how='any', subset=['wind_direction', 'wind_speed'])
```

METAR DATA DOWNLOAD

```
def plot_labels(grib):
    # Load the GRIB file
    grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
    # Select the variable
    prmls = grib.select(name='Pressure reduced to MSL')[0]
    # Get information from the file
    init = str(prmls.analDate) # Init date / time
    run = str(prmls.hour).zfill(2) # Run
    ftime = str(prmls.forecastTime) # Forecast hour
    valid = str(prmls.validDate) # Valid date / time
    print('Init: ' + init + ' UTC')
    print('Run: ' + run + 'Z')
    print('Forecast: ' + ftime)
    print('Valid: ' + valid + ' UTC')

    # Read the data for a specific region
    prmls, lats, lons = prmls.data(lat1=extent[1],lat2=extent[3],lonl=extent[0]+360,lon2=extent[2]+360)

    # Convert to hPa
    prmls = prmls / 100
```

PLOT CONFIGURATION

```
def plot_labels(grib):
    # Load the GRIB file
    grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
    # Select the variable
    prmls = grib.select(name='Pressure reduced to MSL')[0]
    # Get information from the file
    init = str(prmls.analDate) # Init date / time
    run = str(prmls.hour).zfill(2) # Run
    ftime = str(prmls.forecastTime) # Forecast hour
    valid = str(prmls.validDate) # Valid date / time
    print('Init: ' + init + ' UTC')
    print('Run: ' + run + 'Z')
    print('Forecast: ' + ftime)
    print('Valid: ' + valid + ' UTC')

    # Read the data for a specific region
    prmls, lats, lons = prmls.data(lat1=extent[1],lat2=extent[3],lonl=extent[0]+360,lon2=extent[2]+360)

    # Convert to hPa
    prmls = prmls / 100
```

METAR PLOT

```
def plot_labels(grib):
    # Load the GRIB file
    grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")
    # Select the variable
    prmls = grib.select(name='Pressure reduced to MSL')[0]
    # Get information from the file
    init = str(prmls.analDate) # Init date / time
    run = str(prmls.hour).zfill(2) # Run
    ftime = str(prmls.forecastTime) # Forecast hour
    valid = str(prmls.validDate) # Valid date / time
    print('Init: ' + init + ' UTC')
    print('Run: ' + run + 'Z')
    print('Forecast: ' + ftime)
    print('Valid: ' + valid + ' UTC')

    # Read the data for a specific region
    prmls, lats, lons = prmls.data(lat1=extent[1],lat2=extent[3],lonl=extent[0]+360,lon2=extent[2]+360)

    # Convert to hPa
    prmls = prmls / 100
```

IMAGE GENERATION

DATA READING AND MANIPULATION

METAR DATA DOWNLOAD

Script 26: METAR + NWP

LIBRARIES

FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS

NWP DATA READING AND MANIPULATION

METAR DATA DOWNLOAD

PLOT CONFIGURATION

METAR PLOT

IMAGE GENERATION

```
114
115
116 # Choose the plot size (width x height, in inches)
117 plt.figure(figsize=(8,8))
118
119 # Set up the map projection
120 proj = ccrs.PlateCarree()
121
122 # Use the Geostationary projection in cartopy
123 ax = plt.axes(projection=proj)
124
125 # Define the image extent
126 img_extent = [extent[0], extent[2], extent[1], extent[3]]
127 ax.set_extent(img_extent, ccrs.PlateCarree())
128
129 # Change the DPI of the resulting figure. Higher DPI drastically improves the
130 # look of the text rendering.
131 plt.rcParams['savefig.dpi'] = 256
132
133 # Use the Cartopy map projection to transform station locations to the map and
134 # then refine the number of stations plotted by setting a minimum radius
135 point_locs = proj.transform_points(ccrs.PlateCarree(), data['longitude'].values, data['latitude'].values)
136 data = data[reduce_point_density(point_locs, 3)]
137
138 # Add some various map elements to the plot to make it recognizable.
139 ax.add_feature(cfeature.LAND)
140 ax.add_feature(cfeature.OCEAN)
141
142 # Define de contour interval
143 data_min = 500
144 data_max = 1050
145 interval = 2
146 levels = np.arange(data_min,data_max,interval)
147
148 # Plot the contours
149 img1 = ax.contour(lons, lats, prmls, colors='gray', linewidths=0.7, levels=levels)
150 ax.clabel(img1, inline=1, inline_spacing=10, fmt = '%1.0f', colors= 'gray')
151
152 # Use definition to plot H/L symbols
153 plot_maxmin_points(lons, lats, prmls, 'max', 50, symbol='H', color='b', transform=ccrs.PlateCarree())
154 plot_maxmin_points(lons, lats, prmls, 'min', 25, symbol='L', color='r', transform=ccrs.PlateCarree())
155
156 # Add a shapefile
157 # https://geoftp.ibge.gov.br/organizacao-do-territorio/malhas-territoriais/malhas-municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.shp
158 shapefile = list(shapereader.Reader('BR_UF_2019.shp').geometries())
159 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
160
161 # Add coastlines, borders and gridlines
162 ax.coastlines(resolution='10m', color='black', linewidth=0.8)
163 ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
164 gl = ax.gridlines(ccrs.PlateCarree(), color='white', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-50, 90, 5), draw_labels=True)
165 gl.top_labels = False
166 gl.right_labels = False
167
168 #
```

PLOT CONFIGURATION (SAT + NWP)

Script 26: METAR + NWP

```

167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210

```

LIBRARIES

FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS

NWP DATA READING AND MANIPULATION

METAR DATA DOWNLOAD

PLOT CONFIGURATION

METAR PLOT

IMAGE GENERATION

```

# Station Plot

# Start the station plot by specifying the axes to draw on, as well as the
# lon/lat of the stations (with transform). We also set the font size to 12 pt.
stationplot = StationPlot(ax, data['longitude'].values, data['latitude'].values,
                           transform=ccrs.PlateCarree(), fontsize=8)

# Plot the temperature and dew point to the upper and lower left, respectively, of
# the center point. Each one uses a different color.
stationplot.plot_parameter('NW', data['air_temperature'].values, color='red')
stationplot.plot_parameter('SW', data['dew_point_temperature'].values,
                           color='darkgreen')

# A more complex example uses a custom formatter to control how the sea-level pressure
# values are plotted. This uses the standard trailing 3-digits of the pressure value
# in tenths of millibars.
stationplot.plot_parameter('NE', data['air_pressure_at_sea_level'].values,
                           formatter=lambda v: format(10 * v, '.0f')[-3:])

# Plot the cloud cover symbols in the center location. This uses the codes made above and
# uses the `sky_cover` mapper to convert these values to font codes for the
# weather symbol font.
stationplot.plot_symbol('C', data['cloud_coverage'].values, sky_cover)

# Same this time, but plot current weather to the left of center, using the
# `current_weather` mapper to convert symbols to the right glyphs.
stationplot.plot_symbol('W', data['current_wx1_symbol'].values, current_weather)

# Add wind barbs
stationplot.plot_barb(data['eastward_wind'].values, data['northward_wind'].values)

# Also plot the actual text of the station id. Instead of cardinal directions,
# plot further out by specifying a location of 2 increments in x and 0 in y.
stationplot.plot_text((2, 0), data['station_id'].values)

# Add a title
plt.title('METAR + GFS PSML (hPa) | ' + date + ' 00:00 UTC', fontsize=8, loc='center')
#-----

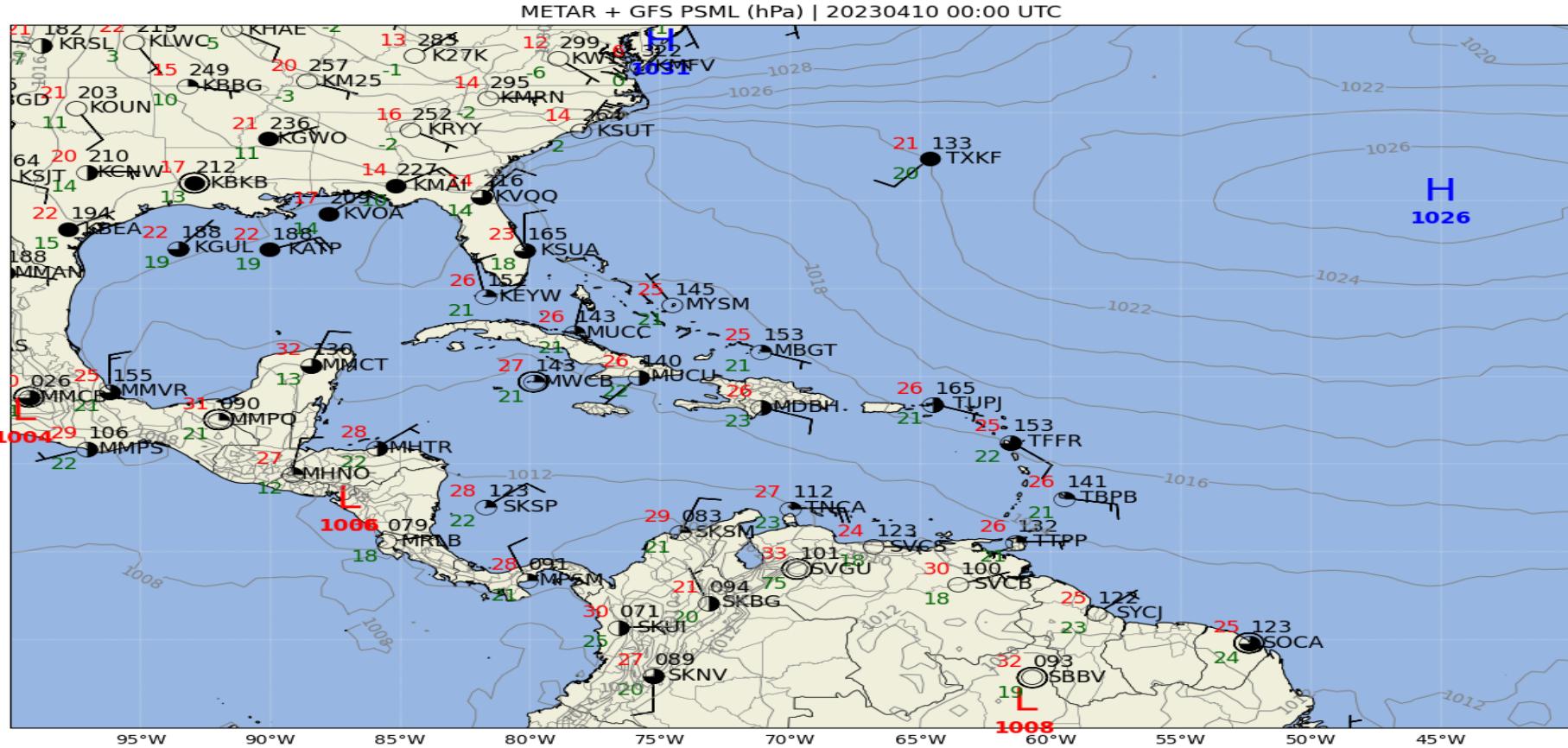
# Save the image
plt.savefig(f'{output}/image_26.png', bbox_inches='tight', pad_inches=0, dpi=300)
plt.show()

```

METAR PLOT

IMAGE GENERATION

Script 26: METAR + NWP



Script 27: METAR + NWP + Satellite

```

LIBRARIES
FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS
SATELLITE DATA DOWNLOAD
SATELLITE DATA REPROJECTION AND CROPPING
NWP DATA READING AND MANIPULATION
METAR DATA DOWNLOAD
PLOT CONFIGURATION
METAR PLOT
IMAGE GENERATION

```

```

# Select the extent [min. lon, min. lat, max. lon, max. lat]
extent = [-93.0, -60.00, -25.00, 18.00]

# Input and output directories
input = "Samples"; os.makedirs(input, exist_ok=True)
output = "Output"; os.makedirs(output, exist_ok=True)

# Datetime to process (today in this example, to match the GFS date)
#date = datetime.today().strftime('%Y%m%d')
#yyyymmddhhmm = date + '0000'
yyyymmddhhmm = '202107020000' # CHANGE THIS DATE TO THE SAME DATE OF YOUR NWP DATA

#--#
# Download the ABI file
file_ir = download_CMI(yyyymmddhhmm, 13, input)

# Variable
var = 'CMI'

# Open the file
img = gdal.Open(f'NETCDF:{input}/{file_ir}.nc' + var)

# Read the header metadata
metadata = img.GetMetadata()
scale = float(metadata.get(var + '#scale factor'))
offset = float(metadata.get(var + '#add offset'))
undef = float(metadata.get(var + '#_FillValue'))
dtime = metadata.get('NC_GLOBAL#time_coverage_start')

# Load the data
ds_cmi = img.ReadAsArray(0, 0, img.RasterXSize, img.RasterYSize).astype(float)

# Apply the scale, offset and convert to celsius
ds_cmi = (ds_cmi * scale + offset) - 273.15

# Reproject the file
filename_ret = f'{output}/IR_{yyyymmddhhmm}.nc'
reproject(filename_ret, img, ds_cmi, extent, undef)

# Open the reprojected GOES-R image
file = Dataset(filename_ret)

# Get the pixel values
data = file.variables['Band1'][:]

#--#

```

SATELLITE DATA DOWNLOAD

SATELLITE DATA REPROJECTION AND CROPPING

Script 27: METAR + NWP + Satellite

```

LIBRARIES
FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS
SATELLITE DATA DOWNLOAD
SATELLITE DATA REPROJECTION AND CROPPING
NWP DATA READING AND MANIPULATION
METAR DATA DOWNLOAD
PLOT CONFIGURATION
METAR PLOT
IMAGE GENERATION

```

NWP DATA READING AND MANIPULATION

```

116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164

# Open the GRIB file
grib = pygrib.open("gfs.t00z.pgrb2full.0p50.f000")

# Select the variable
prmls = grib.select(name='Pressure reduced to MSL')[0]

# Get information from the file
init = str(prmls.analDate) # Init date / time
run = str(prmls.hour).zfill(2) # Run
ftime = str(prmls.forecastTime) # Forecast hour
valid = str(prmls.validDate) # Valid date / time
print('Init: ' + init + ' UTC')
print('Run: ' + run + 'Z')
print('Forecast: ' + ftime)
print('Valid: ' + valid + ' UTC')

# Read the data for a specific region
prmls, lats, lons = prmls.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)

# Convert to hPa
prmls = prmls / 100

# Input directory
dir = "Samples"; os.makedirs(dir, exist_ok=True)

# Download the METAR File
#date = datetime.today().strftime('%Y%m%d')
date = '20210702' # CHANGE THIS DATE TO THE SAME DATE OF YOUR NWP DATA
url = 'https://thredds-test.unidata.ucar.edu/thredds/fileServer/noaaport/text/metar'
file_name = 'metar_' + date + '_0000.txt'

# Sends a GET request to the specified url
myfile = requests.get(url + '/' + file_name)

# Download the file
open(dir + '/' + file_name, 'wb').write(myfile.content)

# METAR File
# https://unidata.github.io/MetPy/latest/examples/plots/Station_Plot.html
data_metar = metar.parse_metar_file(dir + '/' + file_name)

# Drop rows with missing winds
data_metar = data_metar.dropna(how='any', subset=['wind_direction', 'wind_speed'])

#

```

METAR DOWNLOAD

Script 27: METAR + NWP + Satellite

LIBRARIES

FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS

SATELLITE DATA DOWNLOAD

SATELLITE DATA REPROJECTION AND CROPPING

NWP DATA READING AND MANIPULATION

METAR DATA DOWNLOAD

PLOT CONFIGURATION

METAR PLOT

IMAGE GENERATION

PLOT CONFIGURATION

```
165 # Choose the plot size (width x height, in inches)
166 plt.figure(figsize=(8,8))
167
168 # Set up the map projection
169 proj = ccrs.PlateCarree()
170
171 # Use the Geostationary projection in cartopy
172 ax = plt.axes(projection=proj)
173
174 # Define the image extent
175 img_extent = [extent[0], extent[2], extent[1], extent[3]]
176 ax.set_extent(extent[0], extent[2], extent[1], extent[3], ccrs.PlateCarree())
177
178 # Change the DPI of the resulting figure. Higher DPI drastically improves the
179 # look of the text rendering.
180 plt.rcParams['savefig.dpi'] = 255
181
182 # Use the Cartopy map projection to transform station locations to the map and
183 # then refine the number of stations plotted by setting a minimum radius
184 point_locs = proj.transform_points(ccrs.PlateCarree(), data_metar['longitude'].values, data_metar['latitude'].values)
185 data_metar = data_metar[reduce_point_density(point_locs, 3)]
186
187 # Define the color scale based on the channel
188 colormap = "gray_r" # White to black for IR channels
189
190 # Plot the image
191 img1 = ax.imshow(data, origin='upper', vmin=-80, vmax=60, extent=img_extent, cmap=colormap, alpha=1.0)
192
193 # Define de contour interval
194 data_min = 500
195 data_max = 1050
196 interval = 2
197 levels = np.arange(data_min,data_max,interval)
198
199 # Plot the contours
200 img2 = ax.contour(lons, lats, prmls, colors="white", linewidths=0.7, levels=levels)
201 ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10', fmt = '%1.0f', colors= 'gray')
202
203 # Use definition to plot H/L symbols
204 plot_maxmin_points(lons, lats, prmls, 'max', 50, symbol='H', color='b', transform=ccrs.PlateCarree())
205 plot_maxmin_points(lons, lats, prmls, 'min', 25, symbol='L', color='r', transform=ccrs.PlateCarree())
206
207 # Add a shapefile
208 # https://seftp.ibge.gov.br/organizacao_territorial/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR.shp.zip
209 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
210 ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray', facecolor='none', linewidth=0.3)
211
212 # Add coastlines, borders and gridlines
213 ax.coastlines(resolution='10m', color='black', linewidth=0.8)
214 ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
215 gl = ax.gridlines(ccrs.PlateCarree(), color='white', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5))
216 gl.top_labels = False
217 gl.right_labels = False
```

Script 27: METAR + NWP + Satellite

```

LIBRARIES
FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS
SATELLITE DATA DOWNLOAD
SATELLITE DATA REPROJECTION AND CROPPING
NWP DATA READING AND MANIPULATION
METAR DATA DOWNLOAD
PLOT CONFIGURATION
METAR PLOT
IMAGE GENERATION

```

```

# Station Plot
# Start the station plot by specifying the axes to draw on, as well as the
# lon/lat of the stations (with transform). We also the fontsize=12 font.
stationplot = StationPlot(ax, data_metar['longitude'].values, data_metar['latitude'].values,
                           clip_on=True, transform=ccrs.PlateCarree(), fontsize=8)

# Plot the temperature and dew point to the upper and lower left, respectively, of
# the center point. Each one uses a different color.
stationplot.plot_parameter('NW', data_metar['air_temperature'].values, color='red')
stationplot.plot_parameter('SW', data_metar['dew_point_temperature'].values,
                           color='darkgreen')

# A more complex example uses a custom formatter to control how the sea-level pressure
# values are plotted. This uses the standard trailing 3-digits of the pressure value
# in tenths of millibars.
stationplot.plot_parameter('NE', data_metar['air_pressure_at_sea_level'].values,
                           formatter=lambda v: format(10 * v, '.0f')[-3:])

# Plot the cloud cover symbols in the center location. This uses the codes made above and
# uses the `sky_cover` mapper to convert these values to font codes for the
# weather symbol font.
stationplot.plot_symbol('C', data_metar['cloud_coverage'].values, sky_cover)

# Same this time, but plot current weather to the left of center, using the
# `current_weather` mapper to convert symbols to the right glyphs.
stationplot.plot_symbol('W', data_metar['current_wx1_symbol'].values, current_weather)

# Add wind barbs
stationplot.plot_barb(data_metar['eastward_wind'].values, data_metar['northward_wind'].values)

# Also plot the actual text of the station id. Instead of cardinal directions,
# plot further out by specifying a location of 2 increments in x and 0 in y.
stationplot.plot_text((2, 0), data_metar['station_id'].values)
.

# Extract the date from the satellite data
date = (datetime.strptime(dtime, '%Y-%m-%dT%z') - timedelta(hours=4)).strftime('%Y-%m-%d %H:%M')

# Add a title
plt.title('GOES-16 Band 13 ' + date.strftime('%Y-%m-%d %H:%M') + ' UTC' + ' ' + METAR + GFS_PSM)
plt.title('Reg.: ' + str(extent), fontsize=6, loc='right')

# Save the image
plt.savefig(f'{output}/image_27.png', bbox_inches='tight', pad_inches=0, dpi=300)

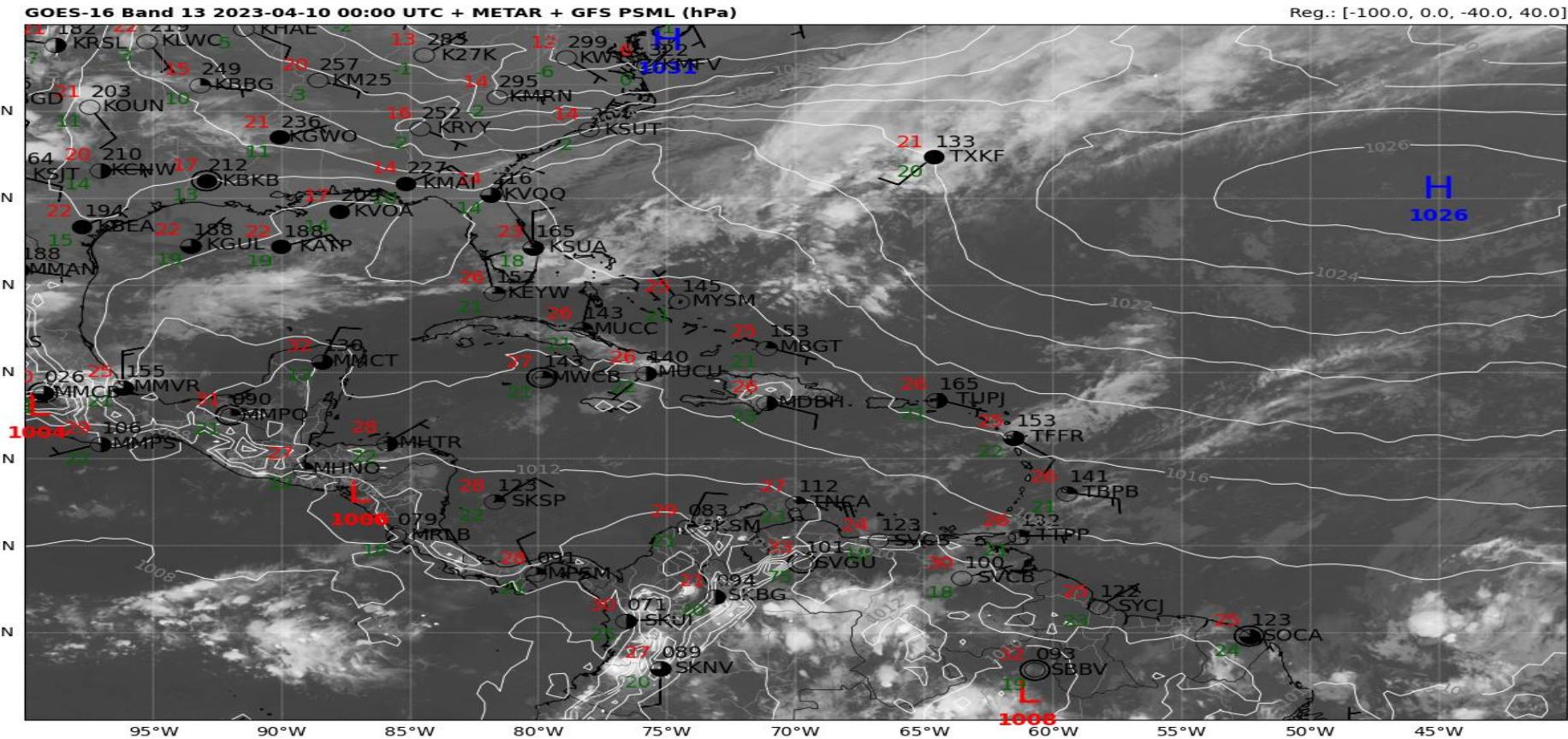
plt.show()

```

METAR PLOT

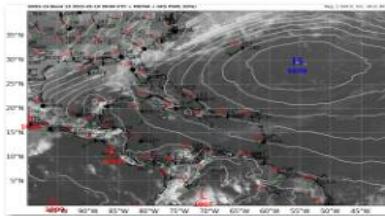
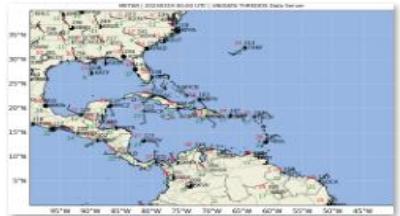
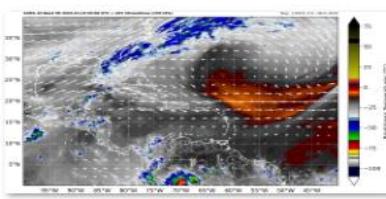
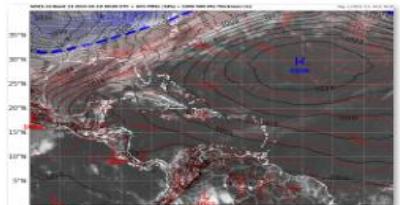
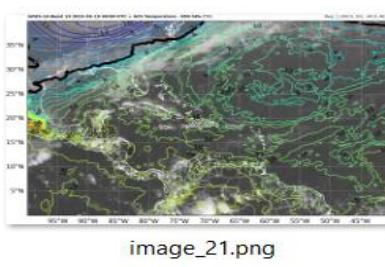
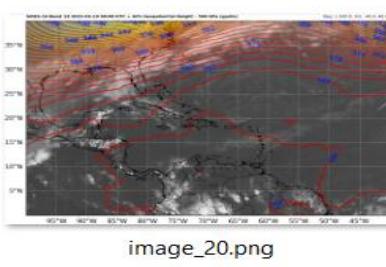
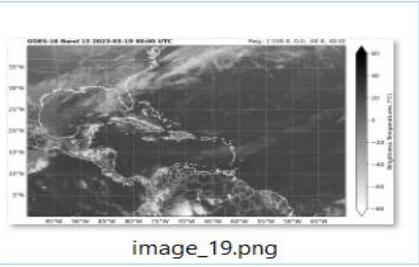
IMAGE GENERATION

Script 27: METAR + NWP + Satellite



“Output” Directory With Imagery

(C:) > VLAB > Python > NWP > Output



Arquivos

Search Output

Output

- IR_202007301200.nc
- image_19.png
- image_20.png
- image_21.png
- image_22.png
- image_23.png
- image_24.png
- image_25.png
- image_26.png
- image_27.png
- image_cs_1.png

Samples

- sample_data
- BR_UF_2019.cpg
- BR_UF_2019.dbf
- BR_UF_2019.prj
- BR_UF_2019.shp
- BR_UF_2019.shx
- IR4AVHRR6 cpt
- SVGAWVX_TEMP cpt
- animation.gif
- br_unidades_da_federacao.zip

- image_11.png
- image_12.png
- image_13.png
- image_15.png
- image_16.png
- image_17.png
- image_18.png

GEONETCast-Americas Training for the Eastern Caribbean States

Day 3 - May 8th

Session 4:

Hands-on

NWP Data Processing

THANK YOU! QUESTIONS?



Diego Souza
diego.souza@inpe.br

DISSM - Meteorological Satellites and Sensors' Division
CGCT - General Coordination of Earth Sciences
INPE - National Institute for Space Research

This is made possible by the generous support of the American people through the United States Agency for International Development (USAID)

