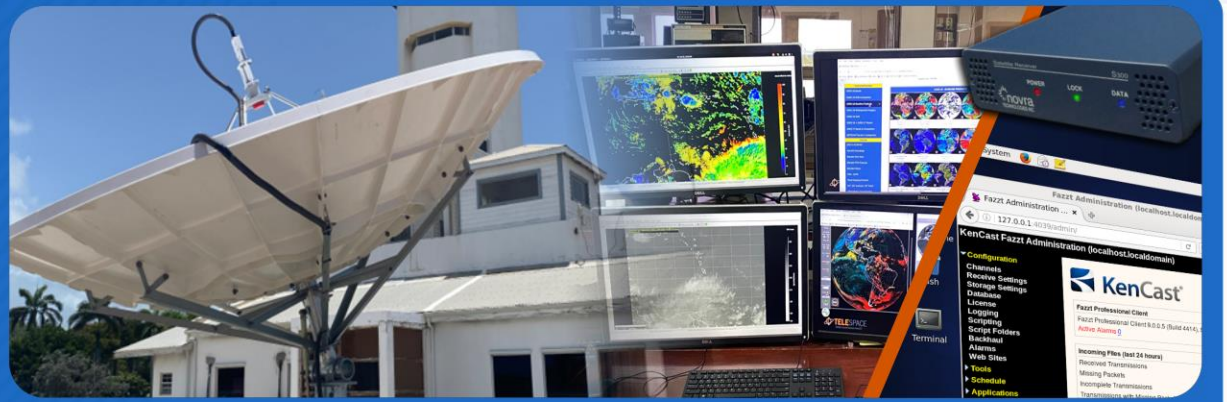
GEONETCast-Americas Training for the Eastern Caribbean States

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

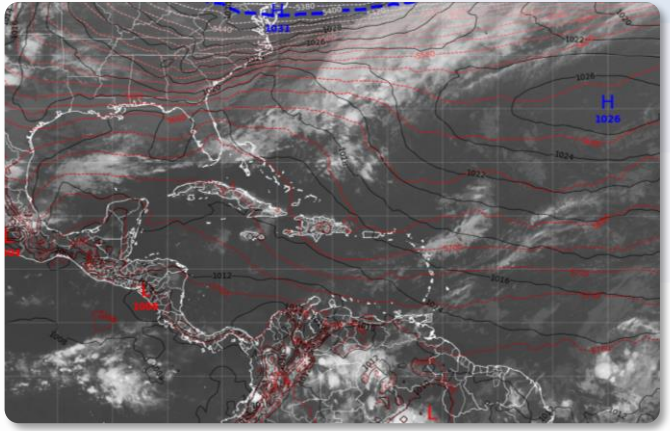
SDCR PROJECT
Strengthening Disaster and Climate Resilience in the Eastern and Southern Caribbean

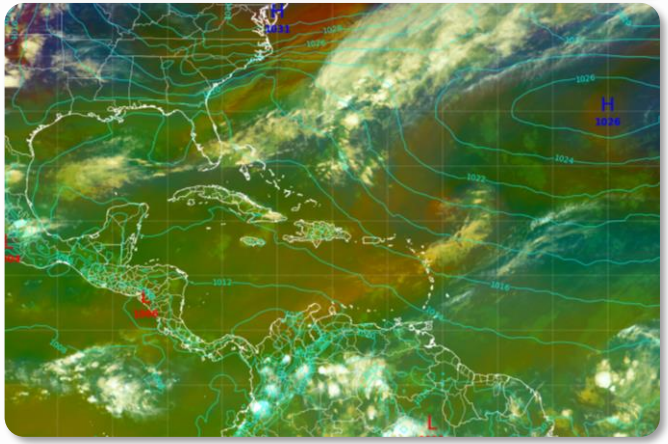
USAID FROM THE AMERICAN PEOPLE

Caribbean Institute for Meteorology and Hydrology

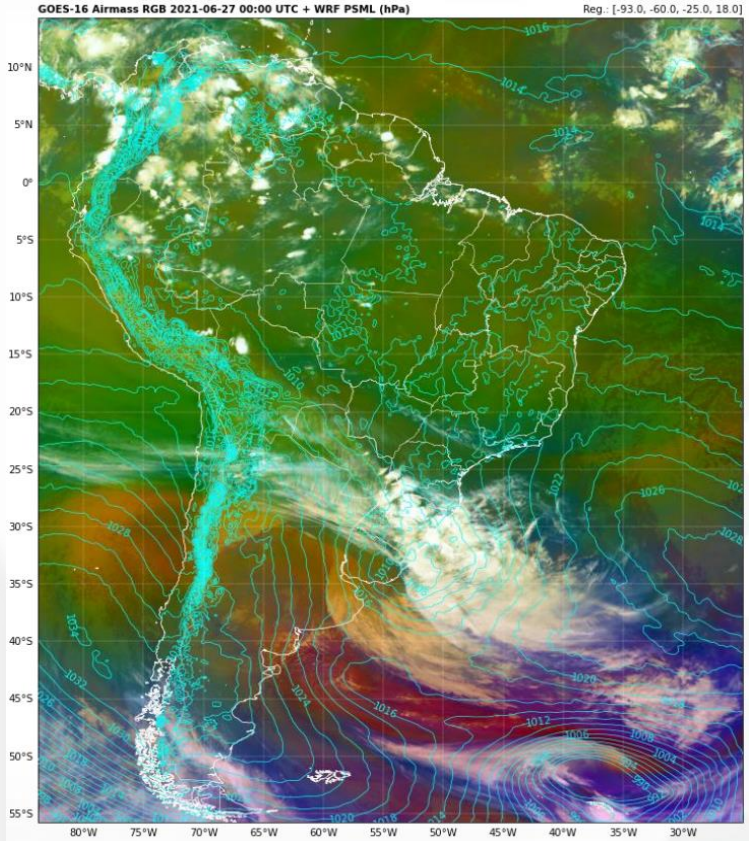
Presentation Outline

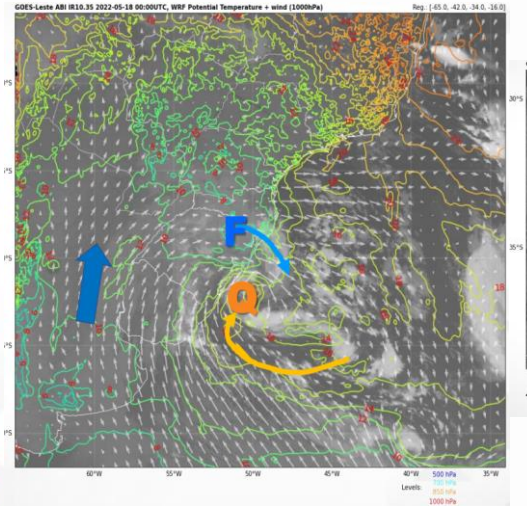
- **Basic Concepts**
  - Available Variables
  - Basic Plot / Pixel Values / Metadata
  - Maps / Shapefile
  - Contours / Custom Color Palettes
  - Animations
- **More Advanced Concepts**
  - Downloading Data Using Scripts
  - Averages / Minimum / Maximums
  - Streamlines / Vectors / Barbs
  - Plotting a “Tool” (Galvez Davison Index)
  - METAR + Satellite + NWP

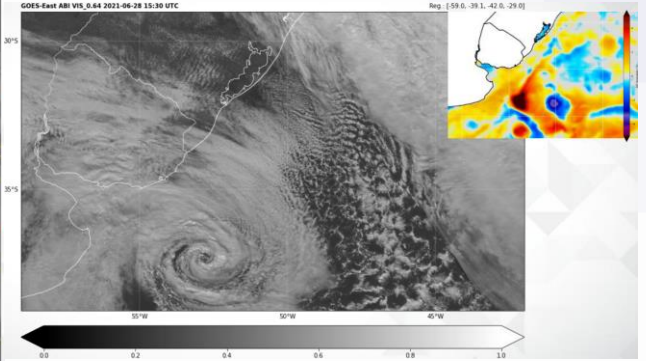


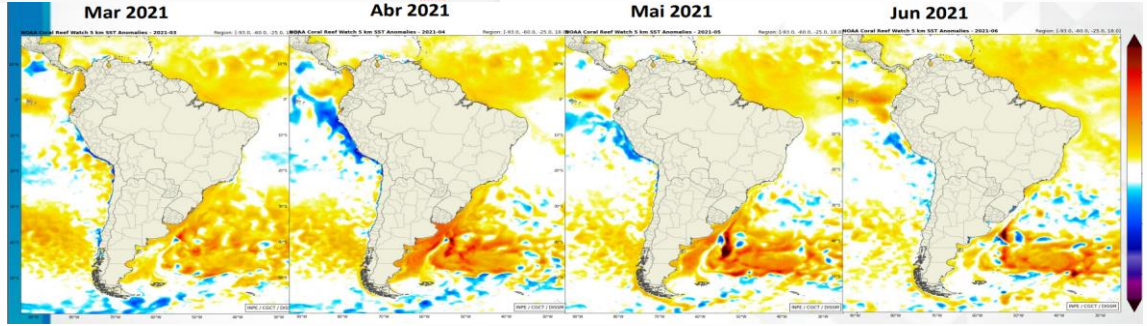


Goal: Create Your Own Plots, With Any Combination



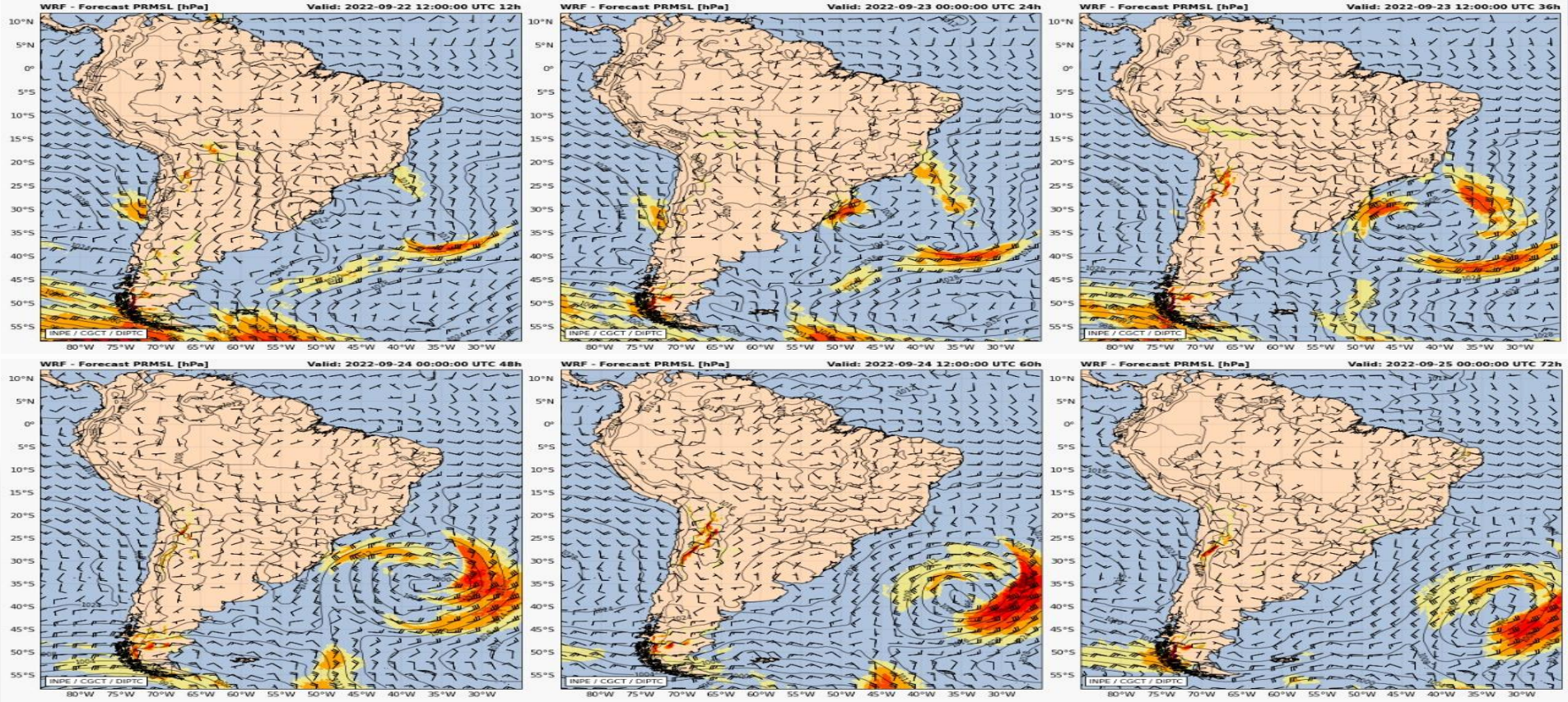






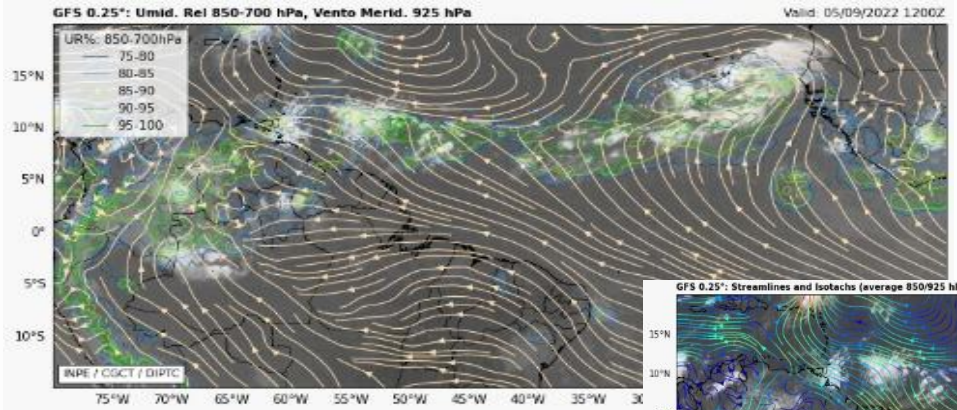
Credits: Regina Ito (INPE / DISSM / CGCT)

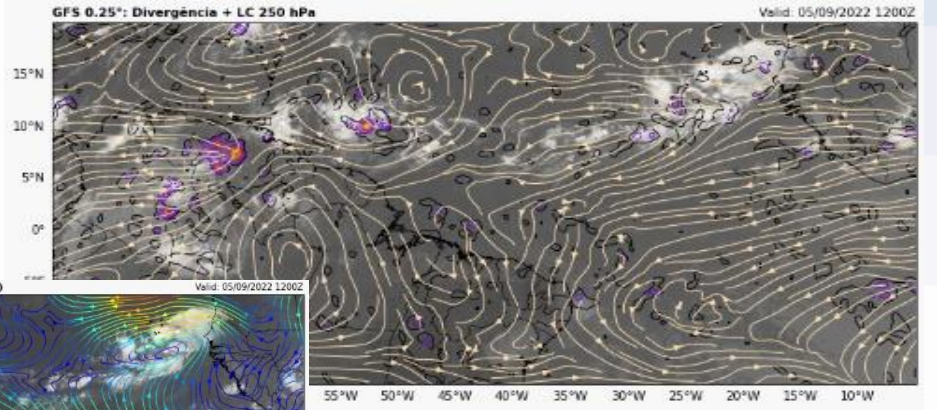
Goal: Create Your Own Plots, With Any Combination

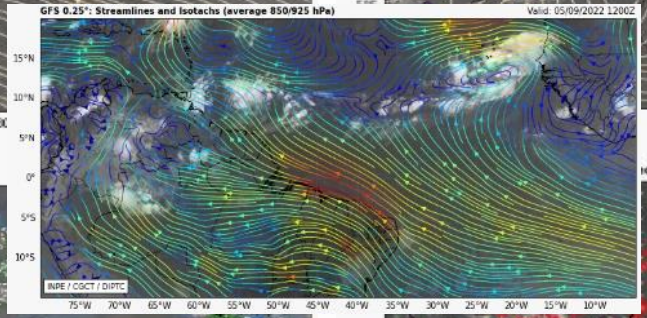


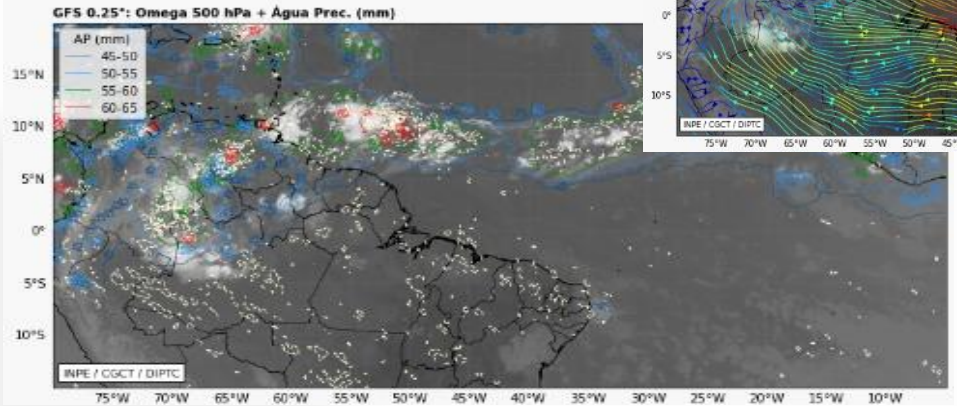
Credits: Marcos Vianna (INPE / DIPTC / CGCT)

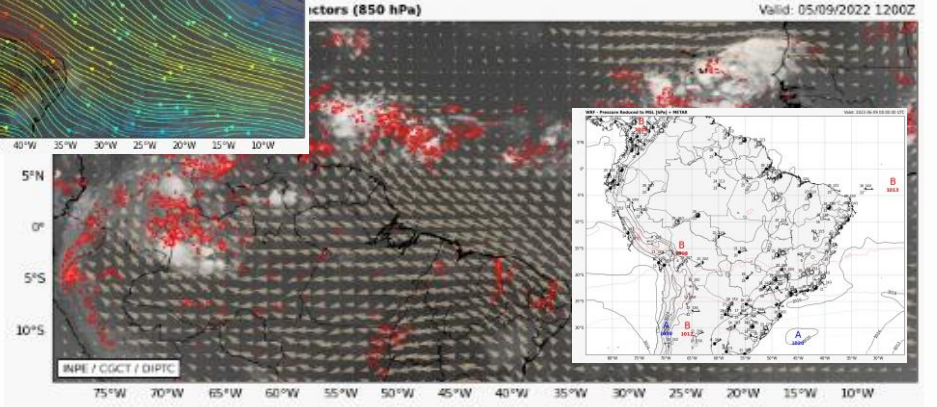
Goal: Create Your Own Plots, With Any Combination





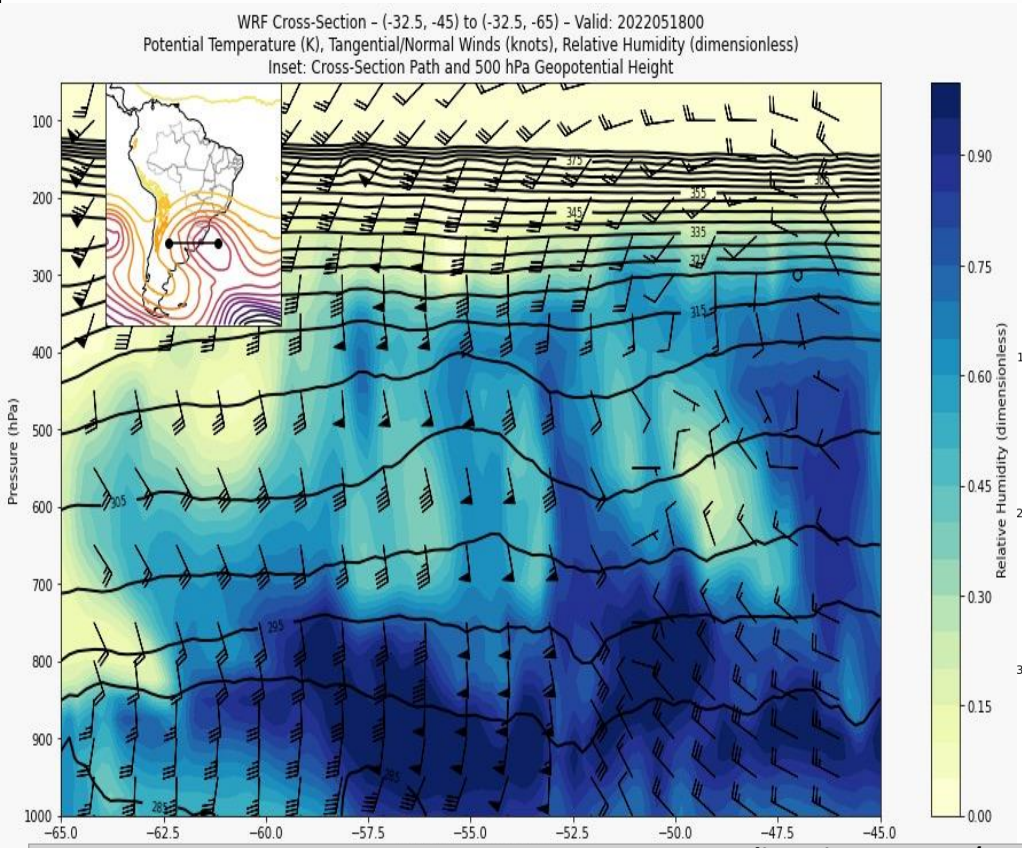


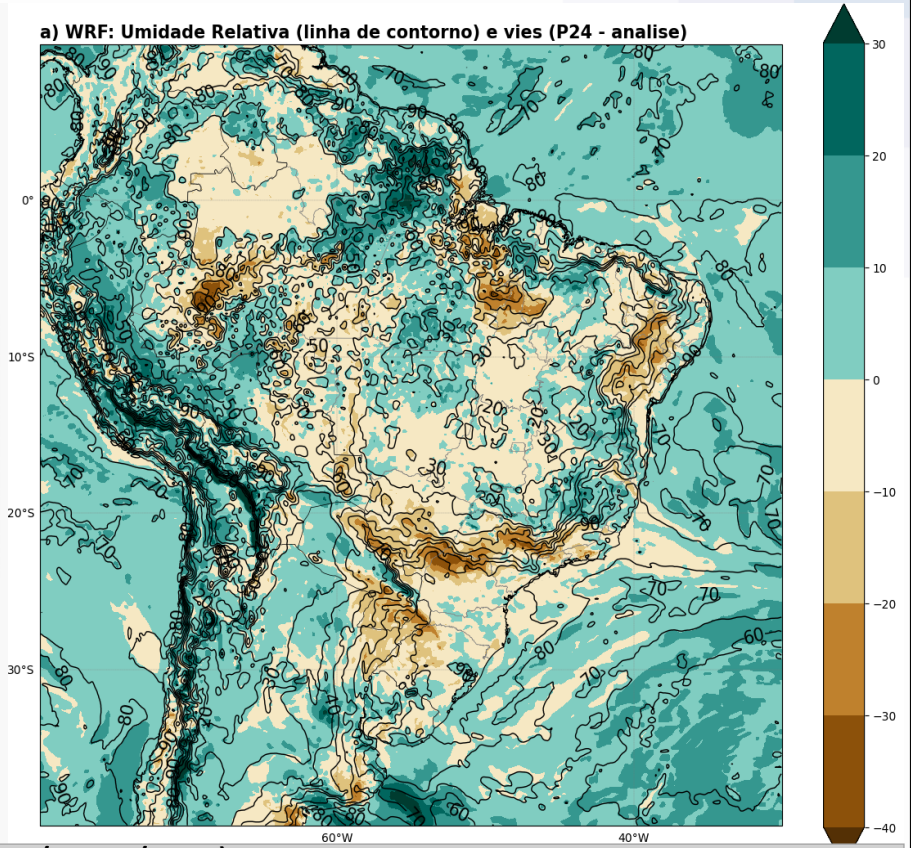




Credits: Marcos Vianna (INPE / DIPTC / CGCT)

Goal: Create Your Own Plots, With Any Combination





Credits: Diogo Arsego (INPE / DIPTC / CGCT)

Training Topics

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.

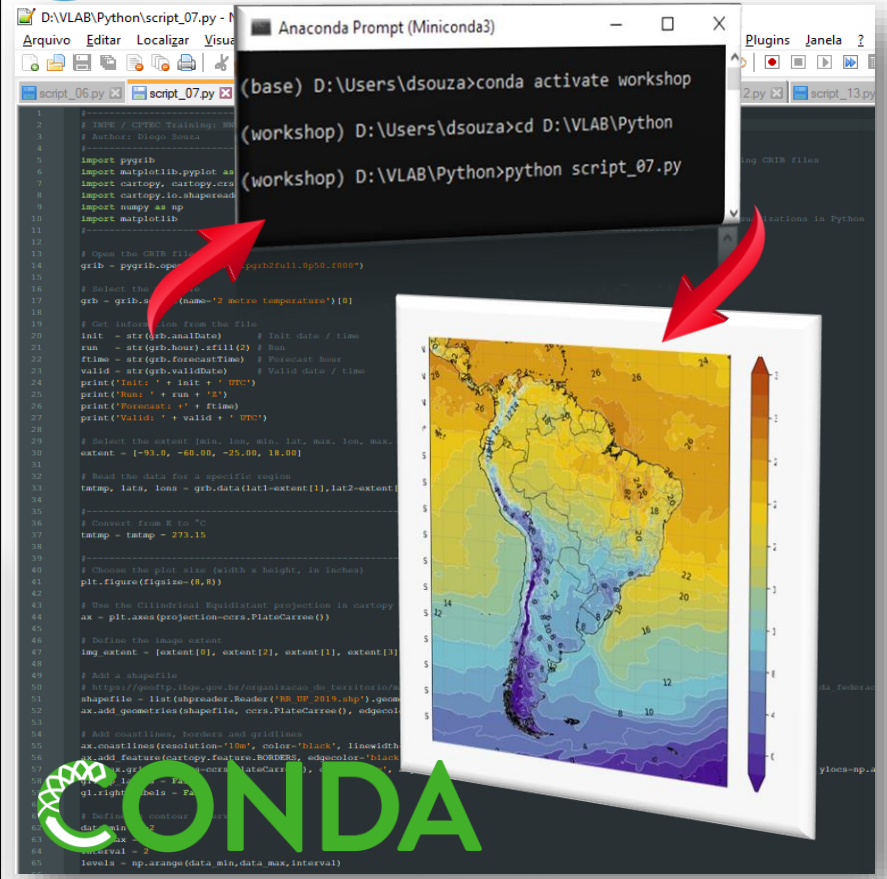
- **METAR + NWP + Satellite**

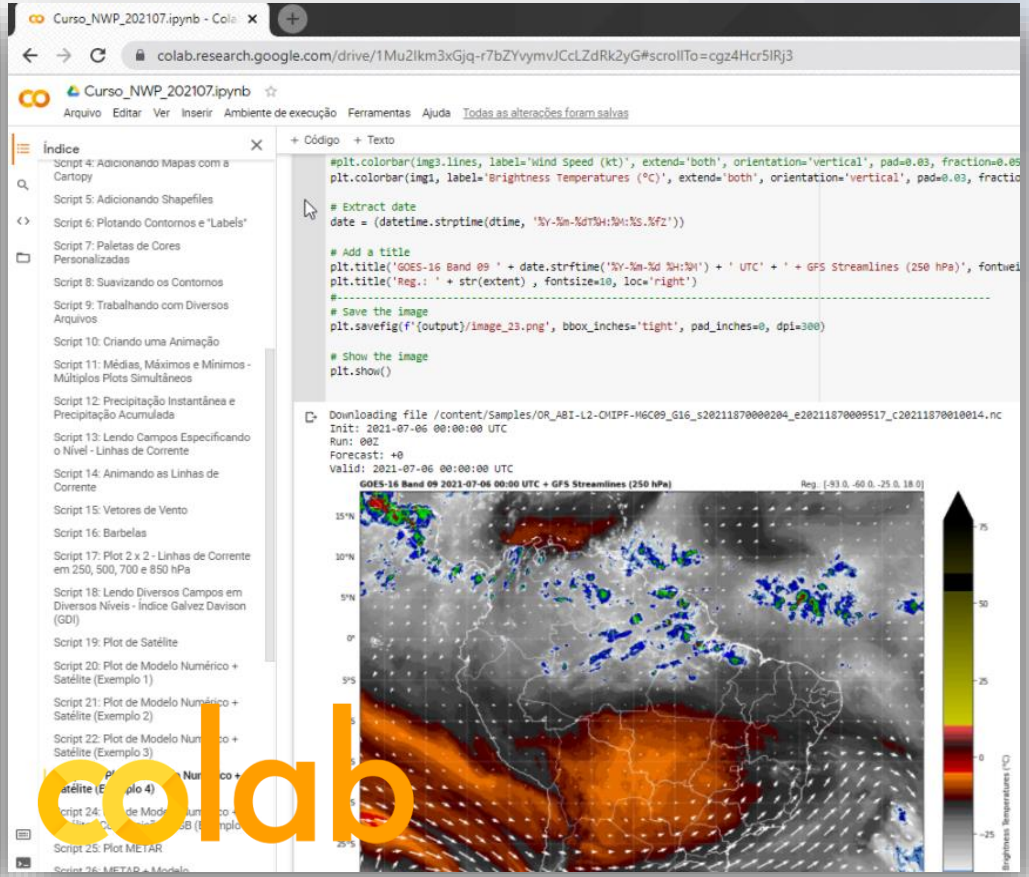
We will review how to read GOES-16 data, learn how to read METAR data and how to overlay these data with NWP data.

- **Case Studies**

Seeing these concepts in practice through case studies.

Running Scripts Locally or in the Cloud



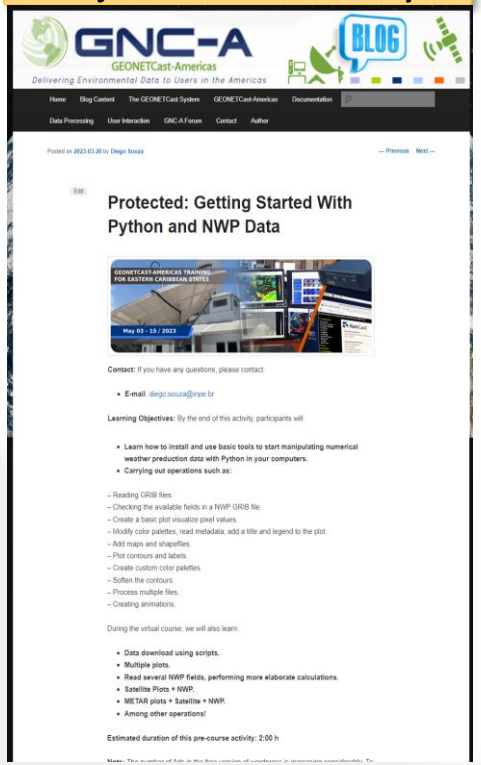


Running Scripts Locally

password: workshop

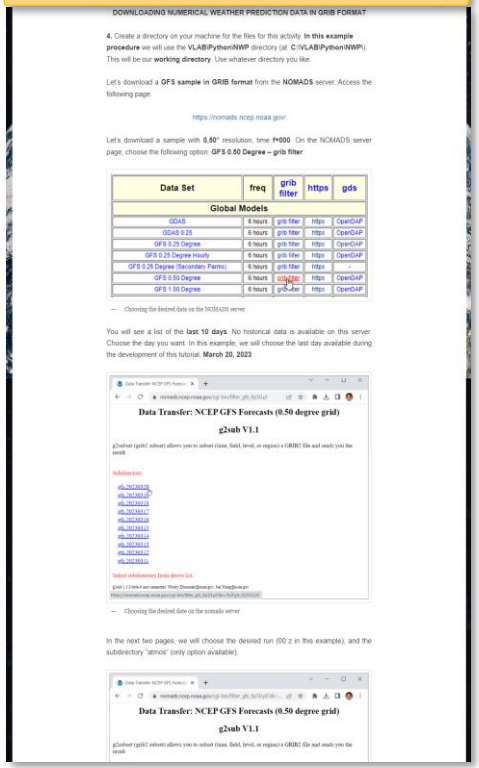
https://geonetcast.wordpress.com/2023/03/20/getting-started-with-python-and-nwp-data/

Objectives of the Activity

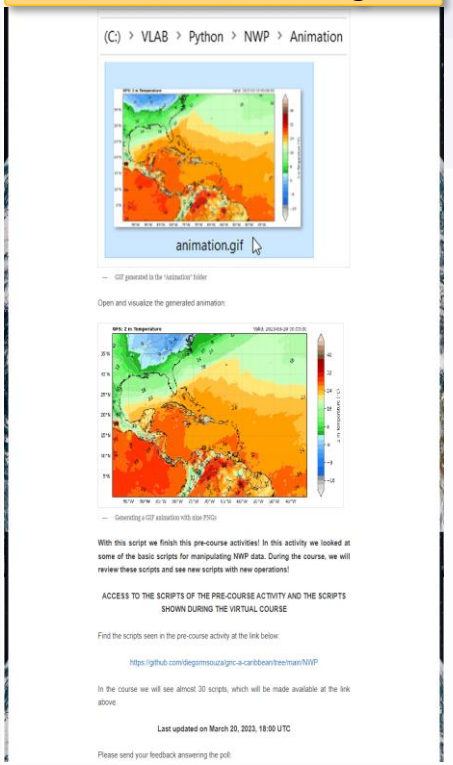


Installation of Tools

Access to Samples (GNC or Cloud)



Local Data Processing

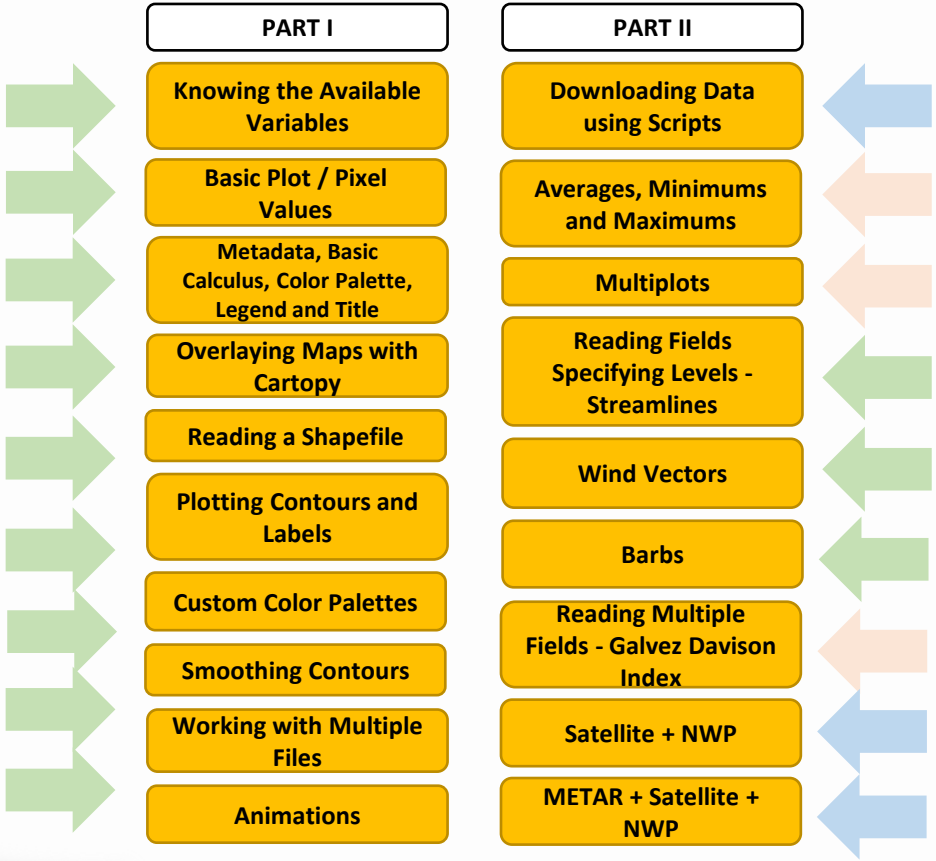


Part 1: Basic Concepts

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 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/MARN-El Salvador

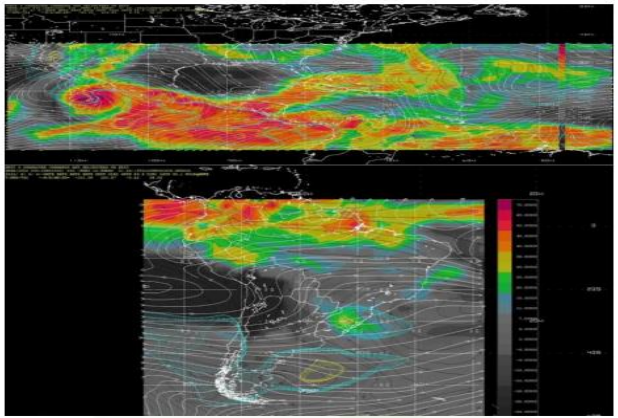
GEONETCast-Americas

ILLUSTRATED PRODUCT CATALOG

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, per file: 11 MB (Central America and Caribbean) / 14 MB (South America) – 2 GB per
Spatial Resolution: 0.5 degree
Naming Convention: gfs_RRR_0p50_CC.f0FFF, Where:
RRR: Region (**crb:** 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 Reseach - Meteorological Satellites and Sensors's Division Page 50 of 76

Accessing NWP Data (GEONETCast-Americas)

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

File Edit View Search Terminal Help

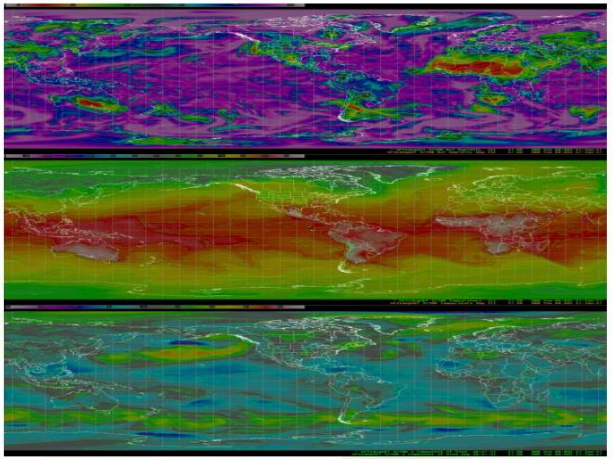
```
[ess@j0153-dpsg ~]$ cd /ess/data/satellite/fazzt/
[ess@j0153-dpsg fazzt]$ ls -l
total 725560
drwxr-xr-x   6 fazzt fazzt         79 Mar  1 07:40 CIMSS
drwxr-xr-x   2 fazzt fazzt      28672 Mar  2 18:04 CIRA
drwxr-xr-x   2 fazzt fazzt   18702336 Mar 29 19:39 EUMETSAT
drwxr-xr-x  18 fazzt fazzt        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 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

GEONETCast-Americas

ILLUSTRATED PRODUCT CATALOG

- **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_PPPPPPKWBCDDHHMN_C_KWBC_YYYYMMDDxxxxxx_xxxxxxxx-xxxx.bin
Where:
PPPPPP: Product Type, forth letter varies from "A" (+0h) to "Y" (+240h),
DDHHMN: 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.

INPE - National Institute for Space Reseach - Meteorological Satellites and Sensors's Division Page 52 of 77

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
Go

Current Hazards
Watches/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 (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 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

Data Transfer: NCEP GFS

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

Directory: /gfs.20210519/12/atmos

NEW Select one file only (size in bytes)

gfs.t12z.pgrb2.0p25.anl (44893140)

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 | OC isotherm | 1000 m above ground | 1000 mb | 100 m above ground | 100 mb | 10 m above ground | 10 m above mean sea level | 10 mb | 1-2 m below ground | 150 mb | 15 mb | 180-0 mb above ground | 1829 m above mean sea level | 1 hybrid 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 | 3658 m above mean sea level | 3 mb | 4000 m above ground | 400 mb | 40 m above ground | 430 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 | 800 mb | 80 m above ground | 850 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 (considered as a 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 | APCP | CAPE | CFRZR | CICEP | CIN | CLWMR | CNWAT | CPOFP | CPRAT | CRAIN | CSNOW | CWAT | CWORK | DLWRF | DPT | DSWRF | DZDT | FLDCP | FRICV | GFLUX | GRLE | GUST | HCDC | HGT | HINDEX | HLCY | HPBL | ICAHT | ICEC | ICEG | ICETK | ICETMP | ICMR | LAND | LCDC | LFTX | LHTFL | MCDC | MSLET | O3MR | PEVPR | PLPL | POT | PRATE | PRES | PRMSL | PWAT | REFC | REFD | RH | RWMR | SFCR | SHTFL | SNMR | SNOD | SOILL | SOILW | SOTYP | SPFH | SUNSD | TCDC | TMAX | TMIN | TMP | TOZNE | TSOIL | UFLX | UGRD | U-GWD | ULWRF | USTM | USWRF | VEG | VFLX | VGRD | V-GWD | VIS | VRATE | VSTM | VVEL | VWSH | WATR | WEASD | WILT

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

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

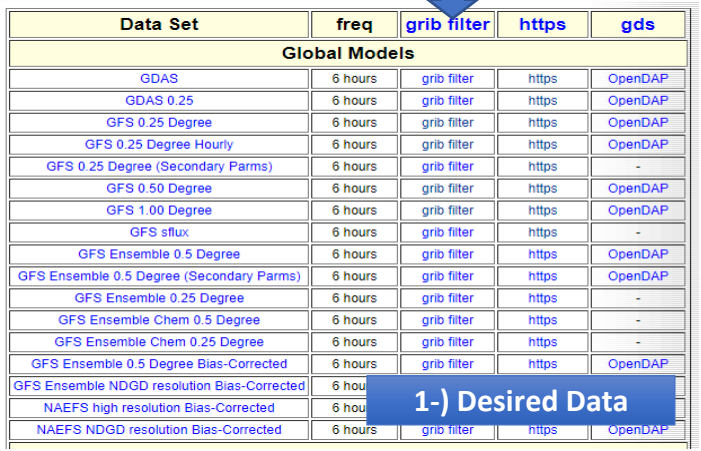
Resolution:
0.25° (every hour or every 3 hours)
0.50°
1.00°

Runs:
00, 06, 12, 18

Timesteps:
1 h or 3 h / 16 days

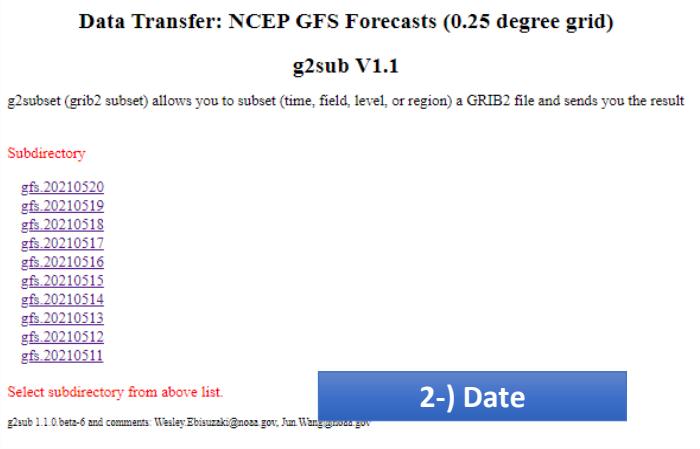
Region:
Global

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



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 hou			
NAEFS high resolution Bias-Corrected	6 hou			
NAEFS NDGD resolution Bias-Corrected	6 hours	grib filter	https	OpenDAP

1-) Desired Data



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

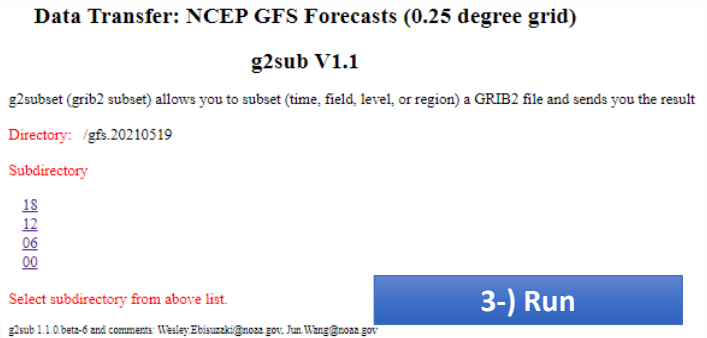
Subdirectory

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.

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

2-) Date



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

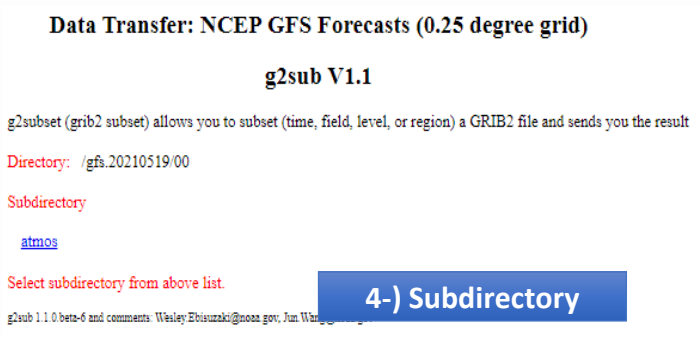
Subdirectory

18
12
06
00

Select subdirectory from above list.

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

3-) Run



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.

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

4-) Subdirectory

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

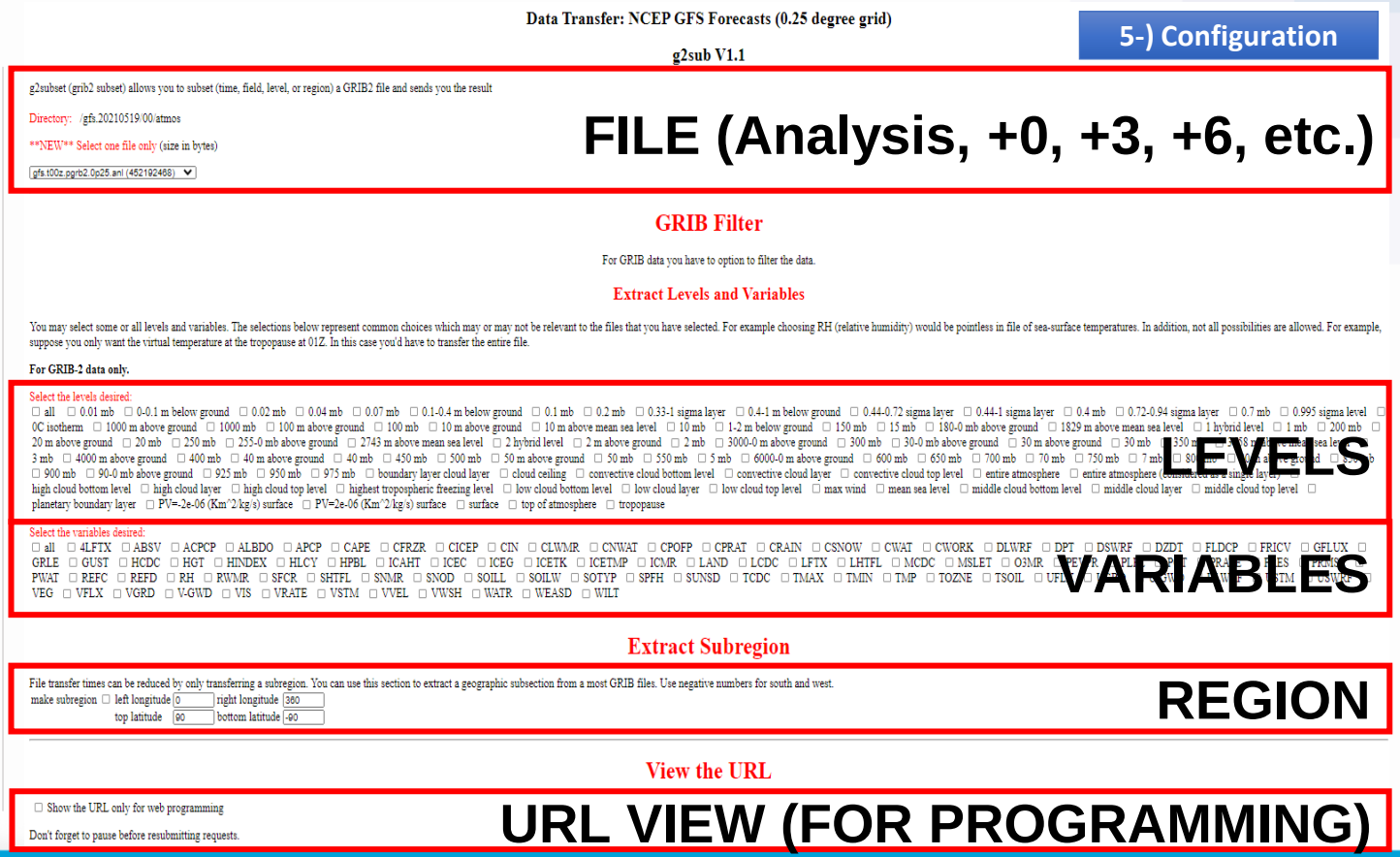
Format:
GRIB2

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



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
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
```

Imports the PyGRIB

LIBRARIES

Open the GRIB file

DATA READING AND MANIPULATION

Create the text file if it doesn't exist

For each variable available in the GRIB file

List the variable in the text file

Show the variable in the terminal

Close the text file

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 20210702000
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 20210702000
```

Some Quick Modifications

- **Check variables from another model**

Script 2: Basic Plot

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

```
#-------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing With Python - Script 2: Basic Plot
# Author: Diego Souza
#-------------------------------------------------------------------------------
import pygrib                    # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files
import matplotlib.pyplot as plt  # Plotting library
#-------------------------------------------------------------------------------

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

# Select the variable
grb = grib.select(name='2 metre temperature')[0]

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

# Read the data for the selected extent
tmtmp, lats, lons = grb.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)

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

# Plot the image
plt.imshow(tmtmp, origin='lower', cmap='jet')
#-------------------------------------------------------------------------------
# Save the image
plt.savefig('image_2.png')

# Show the image
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

```
#-------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing With Python - Script 2: Basic Plot
# Author: Diego Souza
#
import pygrib                          # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files
import matplotlib.pyplot as plt # Plotting library
#-------------------------------------------------------------------------------

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

# Select the variable
grb = grib.select(name='2 metre temperature')[0]

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

# Read the data for the selected extent
tmtmp, lats, lons = grb.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)

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

# Plot the image
plt.imshow(tmtmp, origin='lower', cmap='jet')
#-------------------------------------------------------------------------------
# Save the image
plt.savefig('image_2.png')

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

Import the Matplotlib

LIBRARIES

DATA READING AND MANIPULATION

Variable selection

Desired extent

Reading the variable, latitudes and longitudes

PLOT CONFIGURATION

Plot Size

Color palette

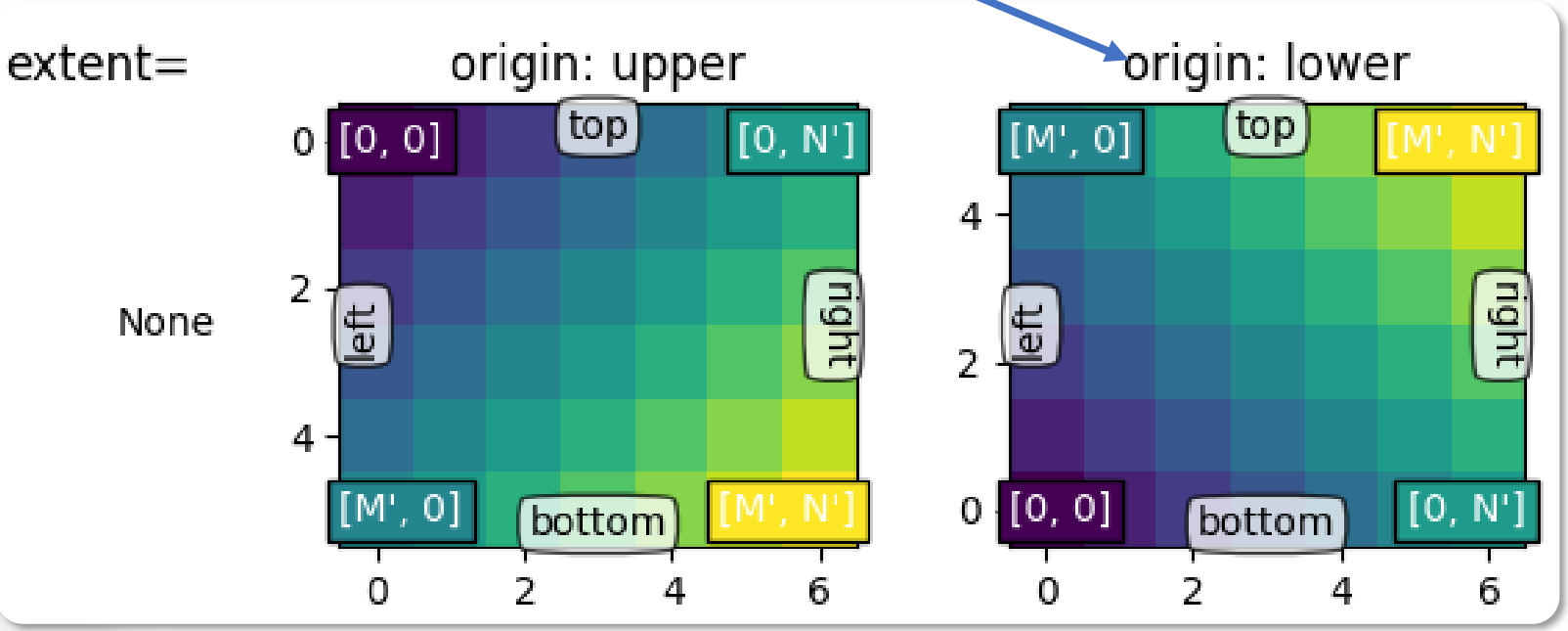
IMAGE GENERATION

Save the image

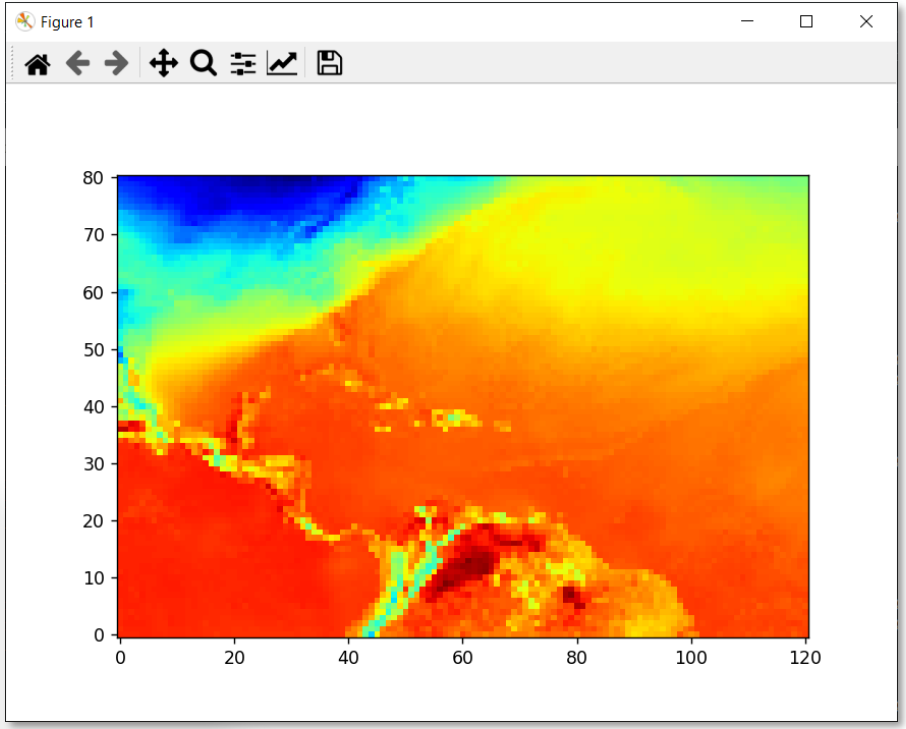
Show the plot

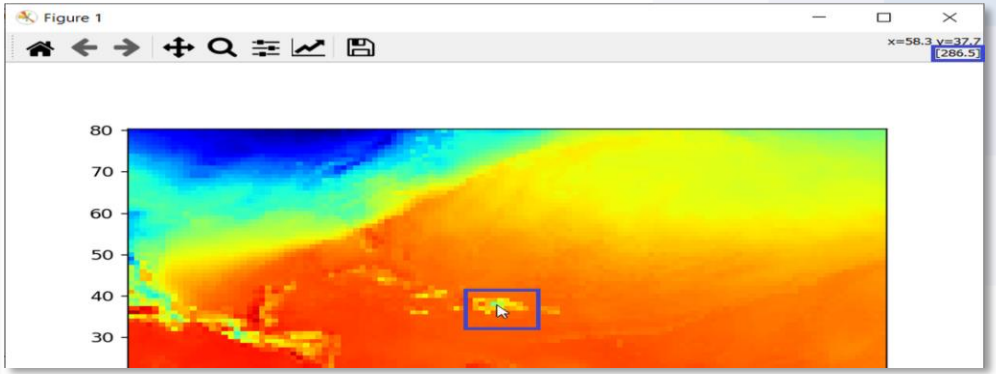
Script 2: Basic Plot

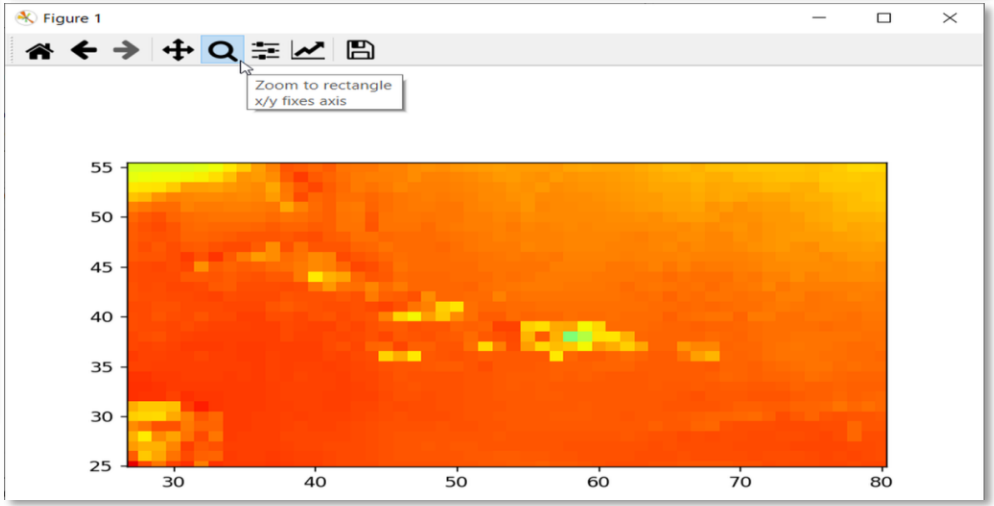
```
# Plot the image
plt.imshow(tmtmp, origin='lower', cmap='jet')
#-----------------------------------------------
```



Script 2: Basic Plot







Some Quick Modifications

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

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

Explaining the new instructions:

LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

```
#---------------------------------------------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing: Metadata / Basic Operation / Color Palettes / Colorbar / Title / Date
# Author: Diego Souza
#---------------------------------------------------------------------------------------------------------------------
import pygrib                           # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files
import matplotlib.pyplot as plt         # Plotting library
#---------------------------------------------------------------------------------------------------------------------

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

# Select the variable
grb = grib.select(name='2 metre temperature')[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')

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

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

#---------------------------------------------------------------------------------------------------------------------
# Convert from K to °C
tmtmp = tmtmp - 273.15

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

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

# Add a colorbar
plt.colorbar(label='2 m Temperature (°C)', extend='both', orientation='horizontal', pad=0.05, fraction=0.05)

# Add a title
plt.title('GFS: 2 m Temperature' , fontweight='bold', fontsize=10, loc='left')
plt.title('Valid: ' + valid, fontsize=10, loc='right')

#---------------------------------------------------------------------------------------------------------------------
# Save the image
plt.savefig('image_3.png')

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

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

Explaining the new instructions:

```python
#-----------------------------------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processi ... ration / Color Palettes / Colorbar / Title / Date
# Author: Diego Souza
#-----------------------------------------------------------------------------------------------------------
import pygrib                        # Provides a high-level interface to the ECWMF ECCODES C library for re...
import matplotlib.pyplot as plt      # Plotting library
#-----------------------------------------------------------------------------------------------------------

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

# Select the variable
grb = grib.select(name='2 metre temperature')[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')

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

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

#-----------------------------------------------------------------------------------------------------------
# Convert from K to °C
tmtmp = tmtmp - 273.15

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

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

# Add a colorbar
plt.colorbar(label='2 m Temperature (°C)', extend='both', orientation='horizontal', pad=0.05, fraction=0.05)

# Add a title
plt.title('GFS: 2 m Temperature' , fontweight='bold', fontsize=10, loc='left')
plt.title('Valid: ' + valid, fontsize=10, loc='right')
#
# Save the image
plt.savefig('image_3.png')

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

LIBRARIES

DATA READING AND MANIPULATION

Reading the Metadata

Conversion from K to °C

PLOT CONFIGURATION

Creation of Colorbar

Size relative to the plot

shrink

Distance relative to the plot

size multiplier

Creation of the Title

IMAGE GENERATION

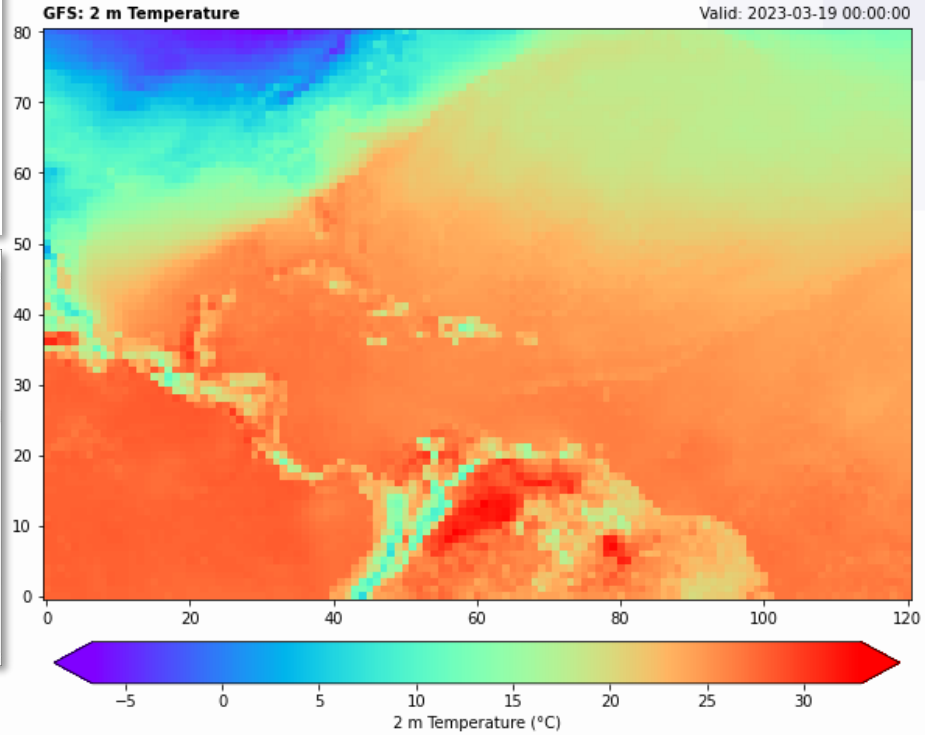
Script 3: Metadata, Basic Operation, Color Palette, Legend and Title

```python
#---------------------------------------------------------------
import pygrib                           # Provides a high-level int
import matplotlib.pyplot as plt         # Plotting library
#---------------------------------------------------------------

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

# Select the variable
grb = grib.select(name='2 metre temperature')[0]

# Print the available keys
print("GRIB Keys :", grb.keys())
```

Anaconda Prompt (Miniconda3) - python script_02.py

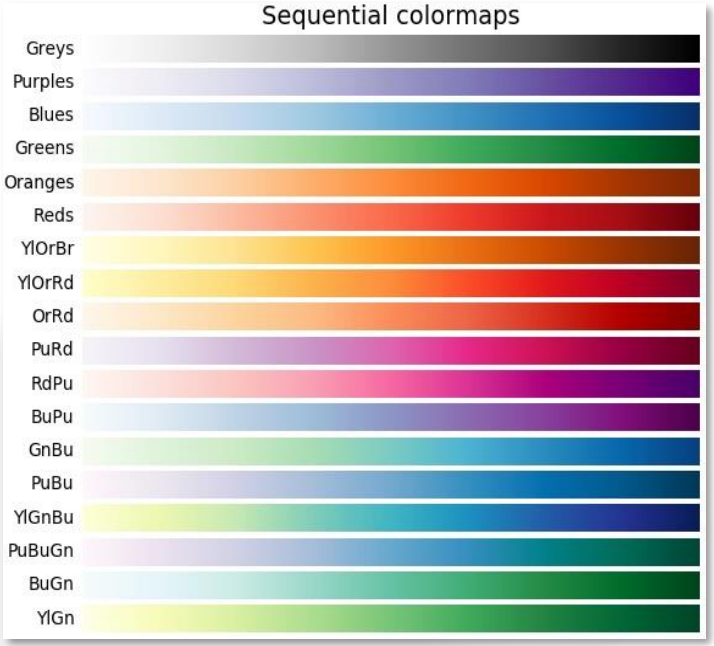
```
(workshop) D:\VLAB\Python>python script_02.py
GRIB Keys : ['globalDomain', 'GRIBEditionNumber', 'tablesVersionLatestOfficial', 'tablesVersionLatest', 'grib2divider', 'is_efas', 'angleSubdivisions', 'm
issingValue', 'ieeeFloats', 'isHindcast', 'section0Length', 'identifier', 'discipline', 'editionNumber', 'totalLength', 'sectionNumber', 'section1Length',
 'numberOfSection', 'centre', 'centreDescription', 'subCentre', 'tablesVersion', 'masterDir', 'localTablesVersion', 'significanceOfReferenceTime', 'year',
 'month', 'day', 'hour', 'minute', 'second', 'dataDate', 'julianDay', 'dataTime', 'productionStatusOfProcessedData', 'typeOfProcessedData', 'md5Section1',
 'selectStepTemplateInterval', 'selectStepTemplateInstant', 'stepType', 'is_chemical', 'is_chemical_distfn', 'is_chemical_srcsink', 'is_aerosol', 'is_aero
sol_optical', 'setCalendarId', 'deleteCalendarId', 'is_uerra', 'sectionNumber', 'grib2LocalSectionPresent', 'deleteLocalDefinition', 'sectionNumber', 'gri
dDescriptionSectionPresent', 'section3Length', 'numberOfSection', 'sourceOfGridDefinition', 'numberOfDataPoints', 'numberOfOctectsForNumberOfPoints', 'int
erpretationOfNumberOfPoints', 'PLPresent', 'gridDefinitionTemplateNumber', 'gridDefinitionDescription', 'shapeOfTheEarth', 'scaleFactorOfRadiusOfSpherical
Earth', 'scaledValueOfRadiusOfSphericalEarth', 'scaleFactorOfEarthMajorAxis', 'scaledValueOfEarthMajorAxis', 'scaleFactorOfEarthMinorAxis', 'scaledValueOf
EarthMinorAxis', 'radius', 'Ni', 'Nj', 'basicAngleOfTheInitialProductionDomain', 'mBasicAngle', 'angleMultiplier', 'mAngleMultiplier', 'subdivisionsOfBasi
cAngle', 'angleDivisor', 'latitudeOfFirstGridPoint', 'longitudeOfFirstGridPoint', 'resolutionAndComponentFlags', 'resolutionAndComponentFlags1', 'resoluti
onAndComponentFlags2', 'iDirectionIncrementGiven', 'jDirectionIncrementGiven', 'uvRelativeToGrid', 'resolutionAndComponentFlags6', 'resolutionAndComponent
Flags7', 'resolutionAndComponentFlags8', 'ijDirectionIncrementGiven', 'latitudeOfLastGridPoint', 'longitudeOfLastGridPoint', 'iDirectionIncrement', 'jDire
ctionIncrement', 'scanningMode', 'iScansNegatively', 'jScansPositively', 'jPointsAreConsecutive', 'alternativeRowScanning', 'iScansPositively', 'scanningM
ode5', 'scanningMode6', 'scanningMode7', 'scanningMode8', 'g2grid', 'latitudeOfFirstGridPointInDegrees', 'longitudeOfFirstGridPointInDegrees', 'latitudeOf
LastGridPointInDegrees', 'longitudeOfLastGridPointInDegrees', 'iDirectionIncrementInDegrees', 'jDirectionIncrementInDegrees', 'latLonValues', 'latitudes',
 'longitudes', 'distinctLatitudes', 'distinctLongitudes', 'gridType', 'md5Section3', 'sectionNumber', 'section4Length', 'numberOfSection', 'NV', 'neitherP
resent', 'productDefinitionTemplateNumber', 'genVertHeightCoords', 'parameterCategory', 'parameterNumber', 'parameterUnits', 'parameterName', 'typeOfGener
atingProcess', 'backgroundProcess', 'generatingProcessIdentifier', 'hoursAfterDataCutoff', 'minutesAfterDataCutoff', 'indicatorOfUnitOfTimeRange', 'stepUn
its', 'forecastTime', 'startStep', 'endStep', 'stepRange', 'stepTypeInternal', 'validityDate', 'validityTime', 'typeOfFirstFixedSurface', 'unitsOfFirstFix
edSurface', 'nameOfFirstFixedSurface', 'scaleFactorOfFirstFixedSurface', 'scaledValueOfFirstFixedSurface', 'typeOfSecondFixedSurface', 'unitsOfSecondFixed
Surface', 'nameOfSecondFixedSurface', 'scaleFactorOfSecondFixedSurface', 'scaledValueOfSecondFixedSurface', 'pressureUnits', 'typeOfLevel', 'level', 'bott
omLevel', 'topLevel', 'tempPressureUnits', 'paramIdECMF', 'paramId', 'shortNameECMF', 'shortName', 'unitsECMF', 'units', 'nameECMF', 'name', 'cfNameECMF',
 'cfName', 'cfVarNameECMF', 'cfVarName', 'modelName', 'ifsParam', 'PVPresent', 'deletePV', 'md5Section4', 'lengthOfHeaders', 'md5Headers', 'sectionNumber'
, 'section5Length', 'numberOfSection', 'numberOfValues', 'dataRepresentationTemplateNumber', 'packingType', 'referenceValue', 'referenceValueError', 'bina
ryScaleFactor', 'decimalScaleFactor', 'optimizeScaleFactor', 'bitsPerValue', 'typeOfOriginalFieldValues', 'md5Section5', 'sectionNumber', 'section6Length'
, 'numberOfSection', 'bitMapIndicator', 'bitmapPresent', 'md5Section6', 'sectionNumber', 'section7Length', 'numberOfSection', 'codedValues', 'values', 'pa
ckingError', 'unpackedError', 'maximum', 'minimum', 'average', 'numberOfMissing', 'standardDeviation', 'skewness', 'kurtosis', 'isConstant', 'changeDecima
lPrecision', 'decimalPrecision', 'setBitsPerValue', 'getNumberOfValues', 'scaleValuesBy', 'offsetValuesBy', 'productType', 'md5Section7', 'section8Length'
, 'analDate', 'validDate']
```

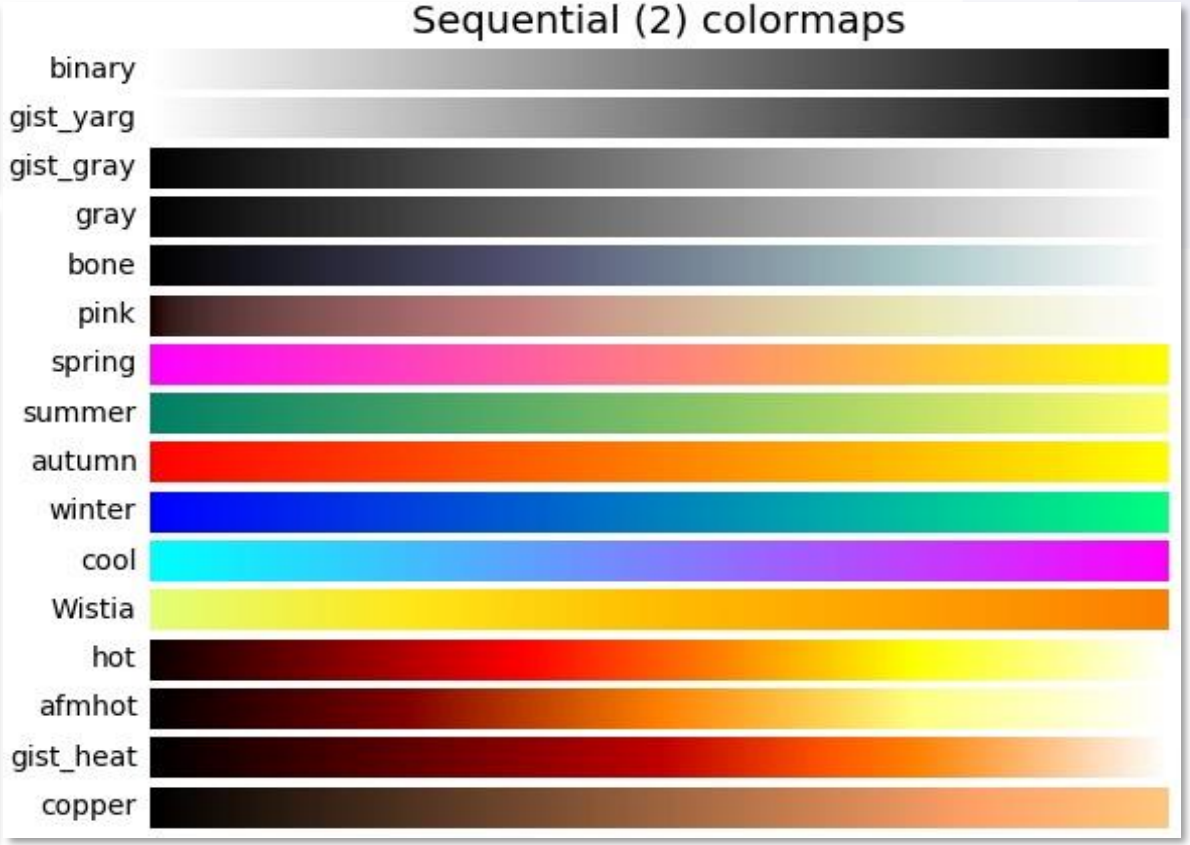
```
(workshop) D:\VLAB\Python>python script_03.py
Init: 2021-07-02 00:00:00 UTC
Run: 00Z
Forecast: +0
Valid: 2021-07-02 00:00:00 UTC
```



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

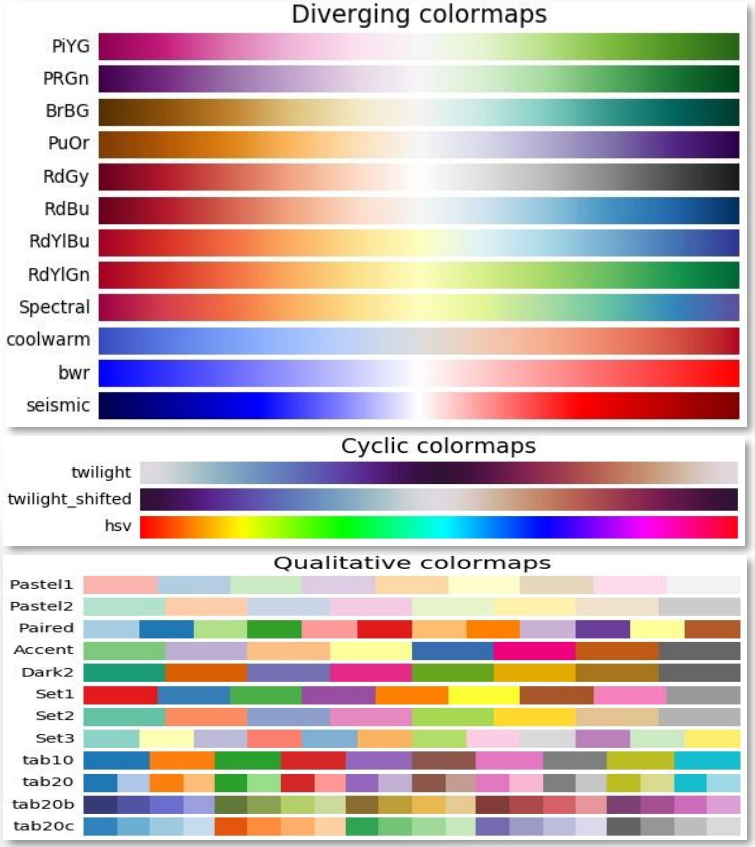
https://matplotlib.org/stable/tutorials/colors/colormaps.html

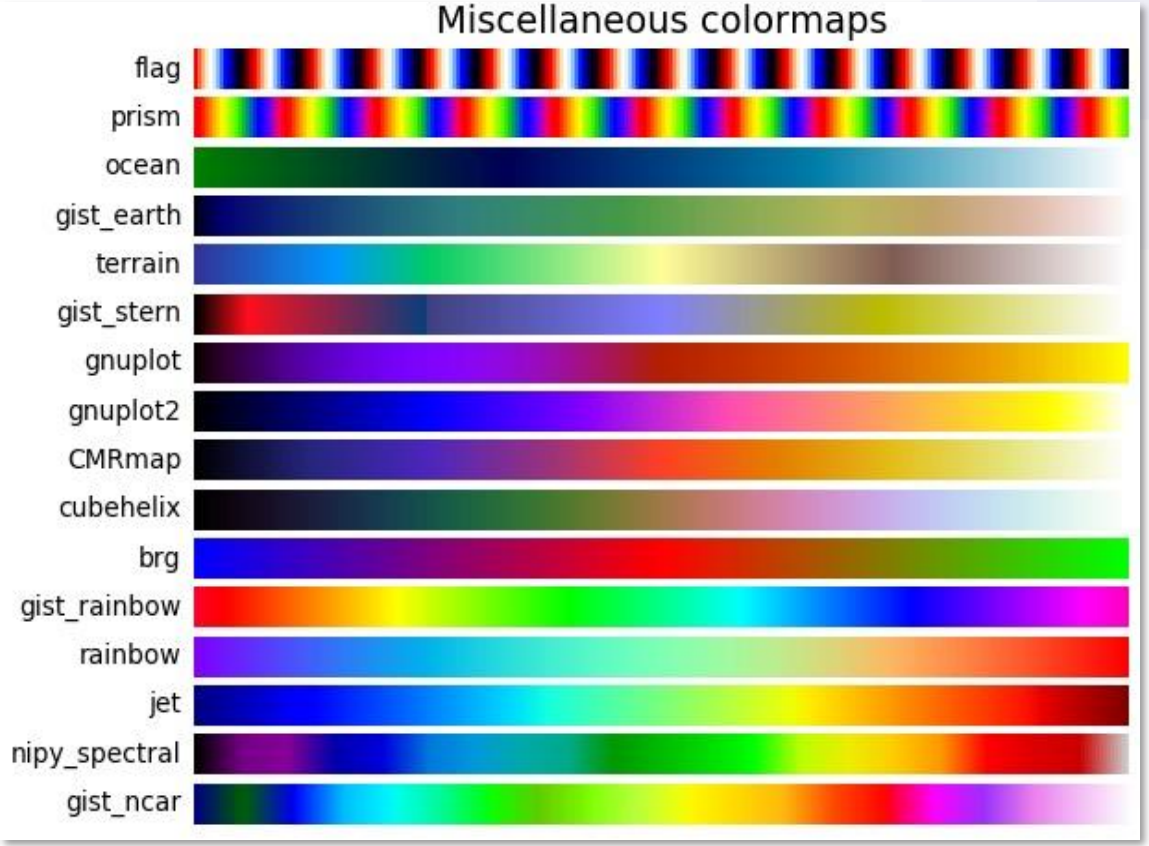




Script 3: Metadata, Basic Operation, Color Palette, Legend and Title

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

```python
#-----------------------------------------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing With Python - Script 4: Adding a Map with Cartopy
# Author: Diego Souza
#-----------------------------------------------------------------------------------------------------------------
import pygrib                       # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files
import matplotlib.pyplot as plt     # Plotting library
import cartopy, cartopy.crs as ccrs # Plot maps
import numpy as np                  # Scientific computing with Python
#-----------------------------------------------------------------------------------------------------------------

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

# Select the variable
grb = grib.select(name='2 metre temperature')[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')

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

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

#-----------------------------------------------------------------------------------------------------------------
# Convert from K to °C
tmtmp = tmtmp - 273.15

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

# Use the Cilindrical Equidistant projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree())

# Define the image extent [min. lon, max. lon, min. lat, max. lat]
img_extent = [extent[0], extent[2], extent[1], extent[3]]

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.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)
gl.top_labels = False
gl.right_labels = False

# Plot the image
img = ax.imshow(tmtmp, origin='lower', extent=img_extent, vmin=-20, vmax=48, cmap='jet')

# Add a colorbar
plt.colorbar(img, label='2 m Temperature (°C)', extend='both', orientation='vertical', pad=0.05, fraction=0.05)

# Add a title
plt.title('GFS: 2 m Temperature' , fontweight='bold', fontsize=10, loc='left')
plt.title('Valid: ' + valid, fontsize=10, loc='right')

# Save the image
plt.savefig('image_4.png')

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

LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

Script 4: Overlaying Maps With Cartopy

Explaining the new instructions:

```
#-----------------------------------------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing With Python - Script 4: Adding a Map with Cartopy
# Author: Diego Souza
#
import pygrib                           # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files
import matplotlib.pyplot as plt         # Plotting library
import cartopy, cartopy.crs as ccrs     # Plot maps
import numpy as np                      # Scientific computing with Python
#-----------------------------------------------------------------------------------------------------------------

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

# Select the variable
grb = grib.select(name='2 metre temperature')[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')

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

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

#-----------------------------------------------------------------------------------------------------------------
# Convert from K to °C
tmtmp = tmtmp - 273.15

#-----------------------------------------------------------------------------------------------------------------
```

Import Cartopy

Import Numpy

LIBRARIES

DATA READING AND MANIPULATION

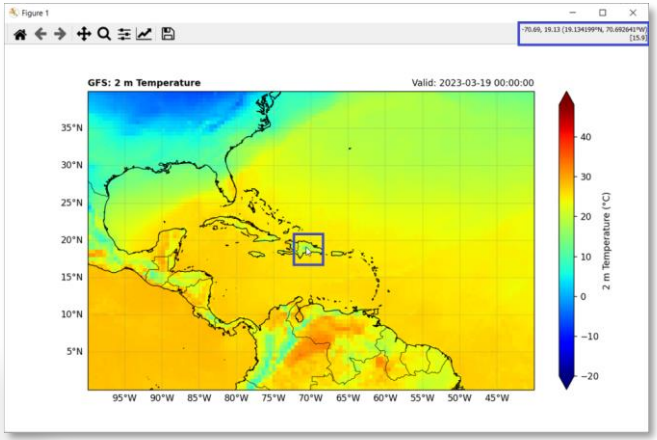
Script 4: Overlaying Maps With Cartopy

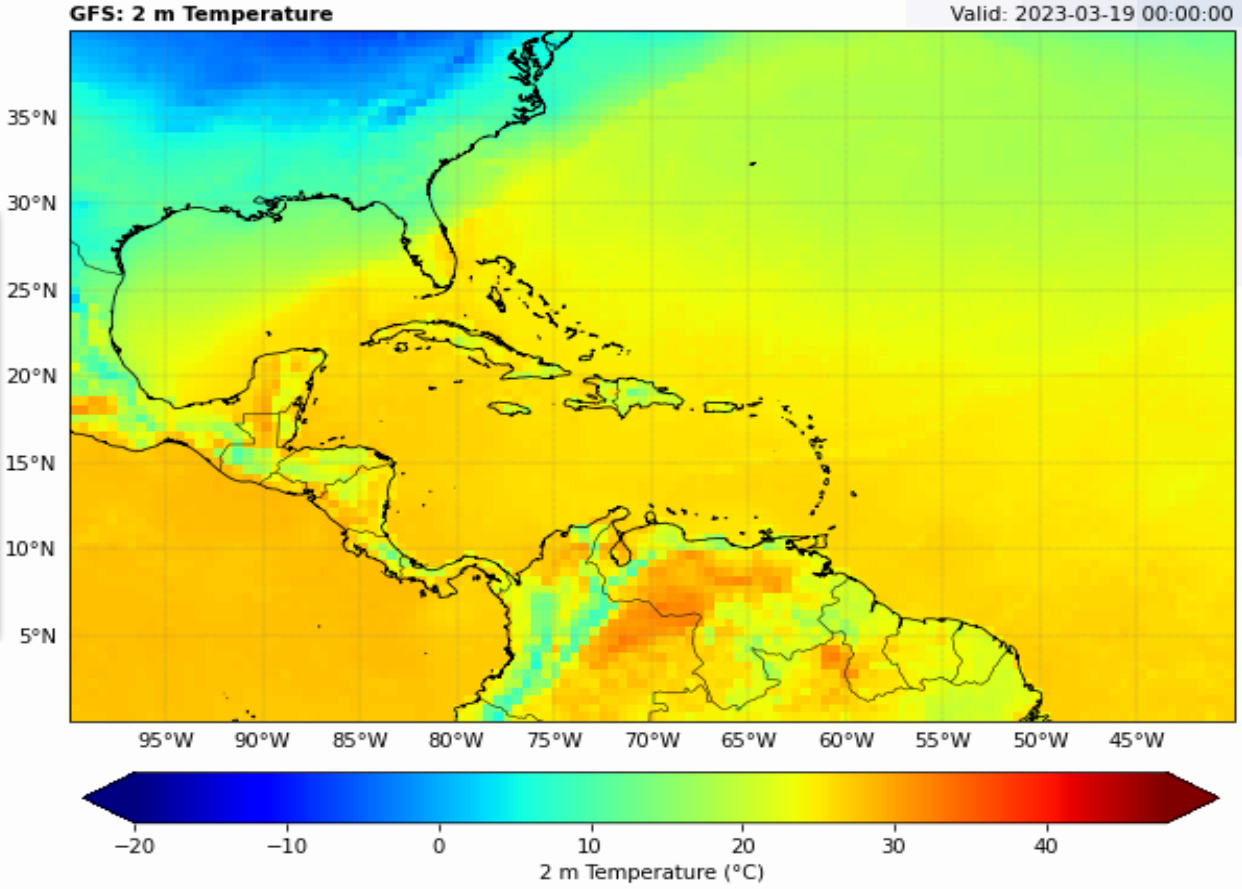
```
# Select the extent [min. lon, min. lat, max. lon, max. lat]
extent = [-93.0, -60.00, -25.00, 18.00]

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

#-----------------------------------------------------------------------------------------------------------------
# Convert from K to °C
tmtmp = tmtmp - 273.15

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

# Use the Cilindrical Equidistant projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree())

# Define the image extent [min. lon, max. lon, min. lat, max. lat]
img_extent = [extent[0], extent[2], extent[1], extent[3]]

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.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)
gl.top_labels = False
gl.right_labels = False

# Plot the image
img = ax.imshow(tmtmp, origin='lower', extent=img_extent, vmin=-20, vmax=48, cmap='jet')

# Add a colorbar
plt.colorbar(img, label='2 m Temperature (°C)', extend='both', orientation='vertical', pad=0.05, fraction=0.05)

# Add a title
plt.title('GFS: 2 m Temperature' , fontweight='bold', fontsize=10, loc='left')
plt.title('Valid: ' + valid, fontsize=10, loc='right')
#-----------------------------------------------------------------------------------------------------------------
# Save the image
plt.savefig('image_4.png')

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

PLOT CONFIGURATION

- Cartopy projection
- Region
- Coastlines
- Countries
- Lon. interval
- Lat. interval
- Latitude and Longitude gridlines
- Disable upper and lower labels

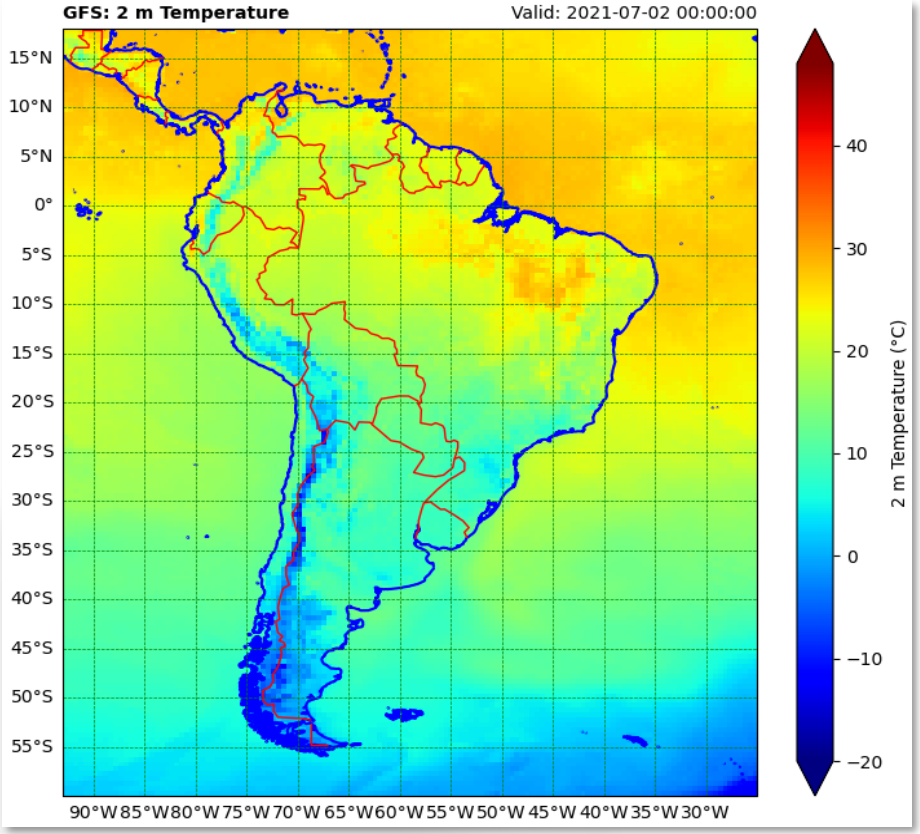
Script 4: Overlaying Maps With Cartopy





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	blueviolet
brown	lemonchiffon	paleturquoise	indigo
firebrick	khaki	darkslategray	darkorchid
maroon	palegoldenrod	darkslategrey	darkviolet
darkred	darkkhaki	teal	mediumorchid
red	ivory	darkcyan	thistle
mistyrose	beige	aqua	plum
salmon	lightyellow	cyan	violet
tomato	lightgoldenrodyellow	darkturquoise	purple
darksalmon	olive	cadetblue	darkmagenta
coral	yellow	powderblue	fuchsia
orangered	olivedrab	lightblue	magenta
lightsalmon	yellowgreen	deepskyblue	orchid
sienna	darkolivegreen	skyblue	mediumvioletred
seashell	greenyellow	lightskyblue	deeppink
chocolate	chartreuse	steelblue	hotpink
saddlebrown	lawngreen	aliceblue	lavenderblush
sandybrown	honeydew	dodgerblue	palevioletred
peachpuff	darkseagreen	lightslategray	crimson
peru	palegreen	lightslategrey	pink
linen	lightgreen	slategray	lightpink

Some Quick Modifications

- 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

```python
#-----------------------------------------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing With Python - Script 5: Adding a Shapefile
# Authors: Diego Souza
#-----------------------------------------------------------------------------------------------------------------
import pygrib                       # Provides a high-level interface to the ECMWF ECCODES C library for reading GRIB files
import matplotlib.pyplot as plt     # Plotting library
import cartopy, cartopy.crs as ccrs # Plot maps
import cartopy.io.shapereader as shpreader  # Import shapefiles
import numpy as np                  # Scientific computing with Python

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

# Select the variable
grb = grib.select(name='2 metre temperature')[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')

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

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

#-----------------------------------------------------------------------------------------------------------------
# Convert from K to °C
tmtmp = tmtmp - 273.15

#-----------------------------------------------------------------------------------------------------------------
plt.figure(figsize=(8,8))

# Use the Cilindrical Equidistant projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree())

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]

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

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.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)
gl.top_labels = False
gl.right_labels = False

# Plot the image
img = ax.imshow(tmtmp, origin='lower', extent=img_extent, vmin=-20, vmax=48, cmap='jet')

# Add a colorbar
plt.colorbar(img, label='2 m Temperature (°C)', extend='both', orientation='vertical', pad=0.05, fraction=0.05)

# Add a title
plt.title('GFS: 2 m Temperature' , fontweight='bold', fontsize=10, loc='left')
plt.title('Valid: ' + valid, fontsize=10, loc='right')

# Save the image
plt.savefig('image_5.png')

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

LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

Script 5: Reading a Shapefile

Explaining the new instructions:

Imports the Cartopy "shpreader" module

LIBRARIES

DATA READING AND MANIPULATION

```
#---------------------------------------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing with Python - Script 5: Adding a Shapefile
# Author: Diego Souza
#---------------------------------------------------------------------------------------------------------------
import pygrib                                  # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files
import matplotlib.pyplot as plt                # Plotting library
import cartopy, cartopy.crs as ccrs            # Plot maps
import cartopy.io.shapereader as shpreader     # Import shapefiles
import numpy as np                             # Scientific computing with Python
#---------------------------------------------------------------------------------------------------------------

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

# Select the variable
grb = grib.select(name='2 metre temperature')[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')

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

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

#---------------------------------------------------------------------------------------------------------------
# Convert from K to °C
tmtmp = tmtmp - 273.15

#---------------------------------------------------------------------------------------------------------------
```

Script 5: Reading a Shapefile

```
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 Cilindrical 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://geoftp.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(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)
57  gl.top_labels = False
58  gl.right_labels = False
59
60  # Plot the image
61  img = ax.imshow(tmtmp, origin='lower', extent=img_extent, vmin=-20, vmax=48, cmap='jet')
62
63  # Add a colorbar
64  plt.colorbar(img, label='2 m Temperature (°C)', extend='both', orientation='vertical', pad=0.05, fraction=0.05)
65
66  # Add a title
67  plt.title('GFS: 2 m Temperature' , fontweight='bold', fontsize=10, loc='left')
68  plt.title('Valid: ' + valid, fontsize=10, loc='right')
69  #-----------------------------------------------------------------------------------------------------------------------
70  # Save the image
```

PLOT CONFIGURATION

Read the Shapefile

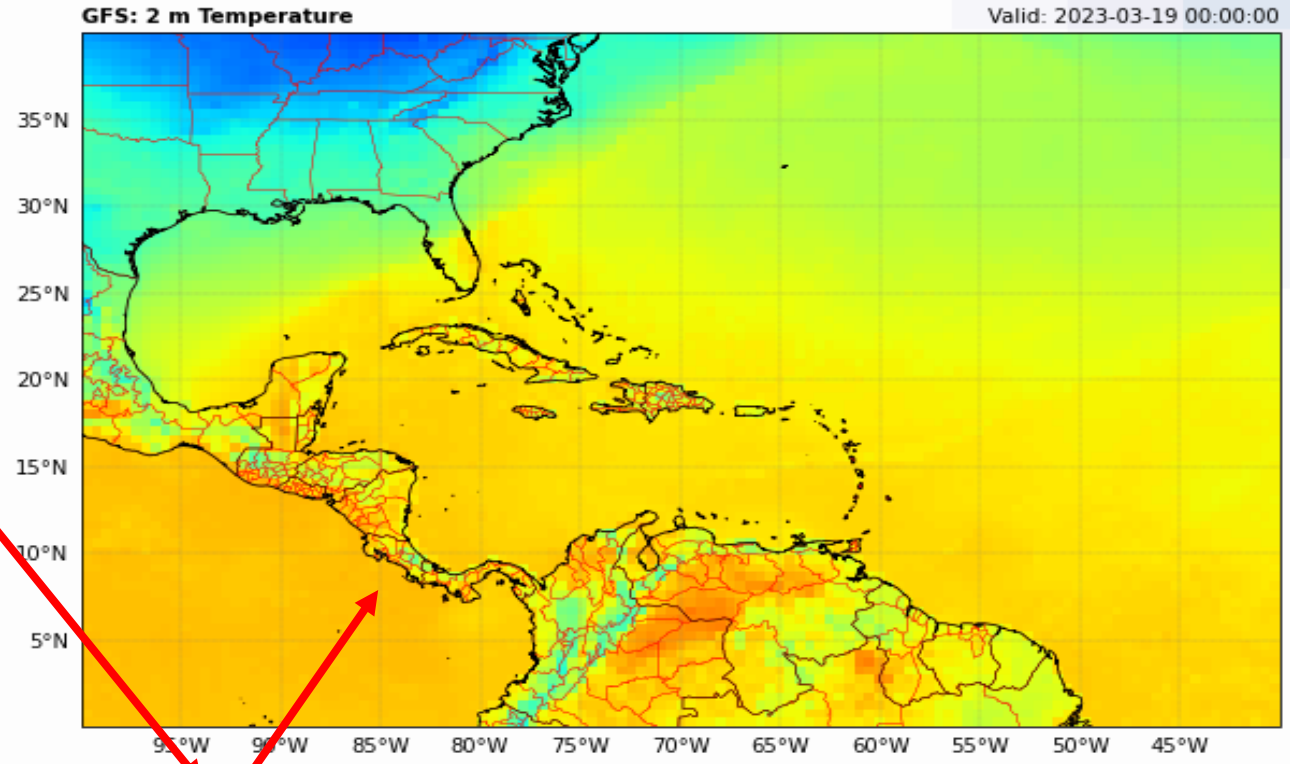
Configure the Shapefile

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

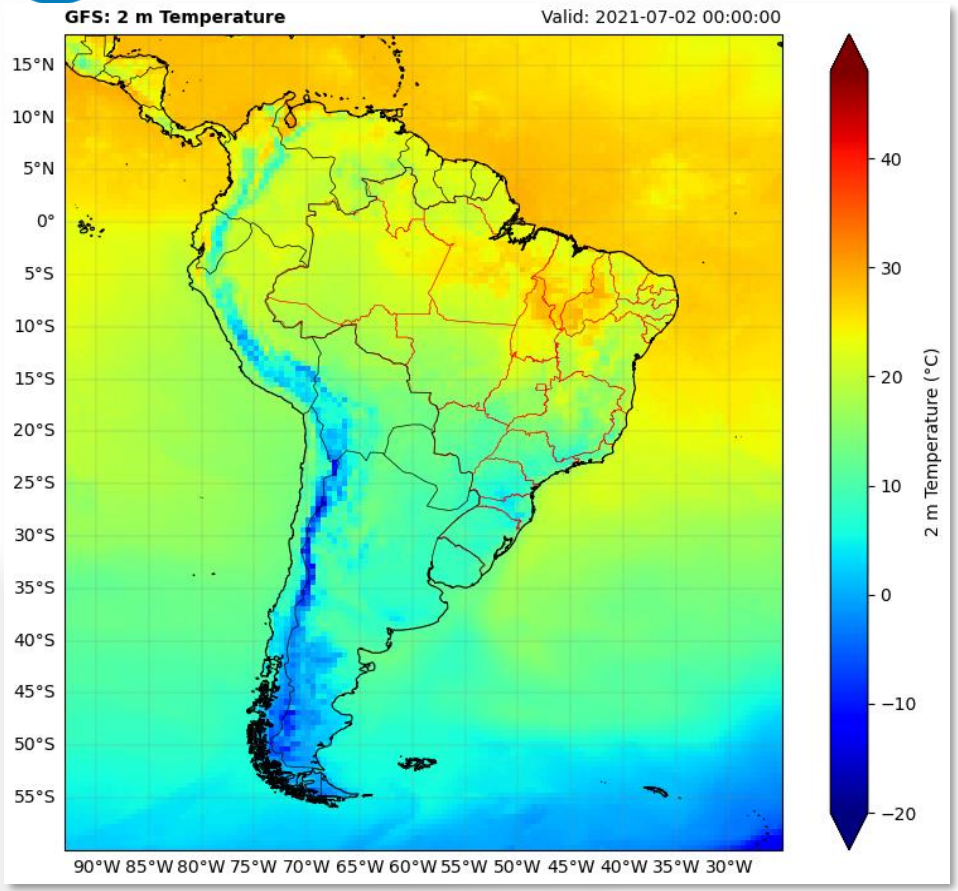


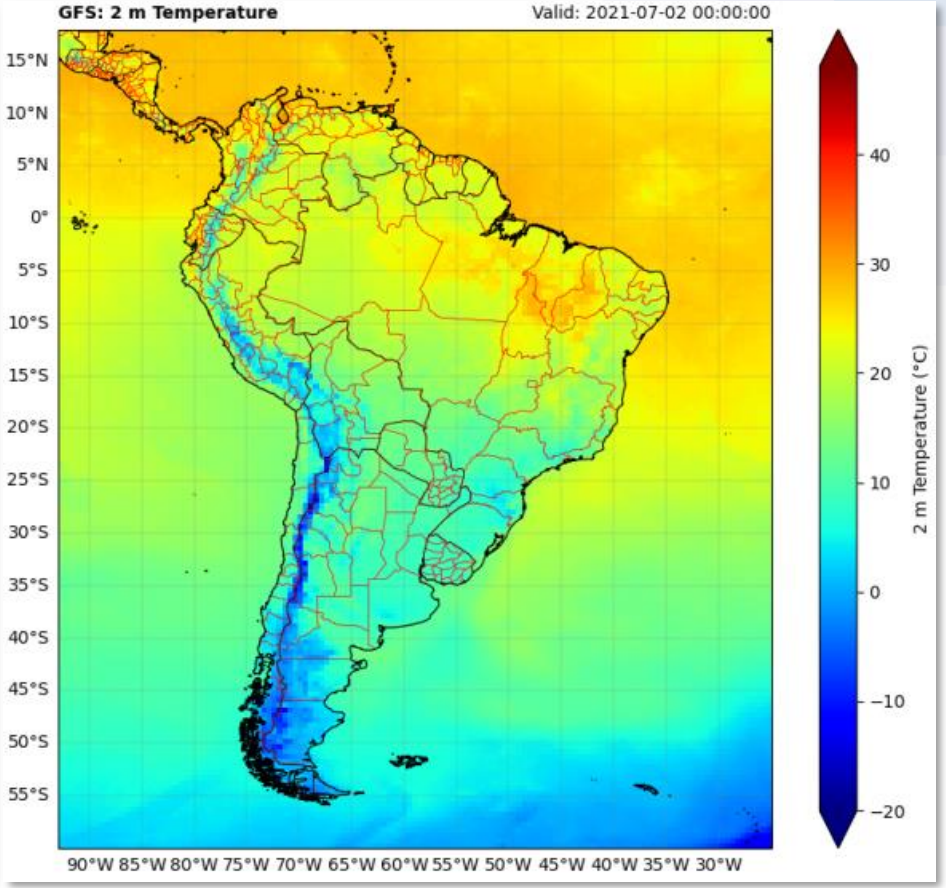

```
48  # Add a shapefile
49  shapefile = list(shpreader.Reader('ne_10m_admin_1_states_provinces.shp').geometries())
50  ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='red',facecolor='none', linewidth=0.3)
```

Some Quick Modifications

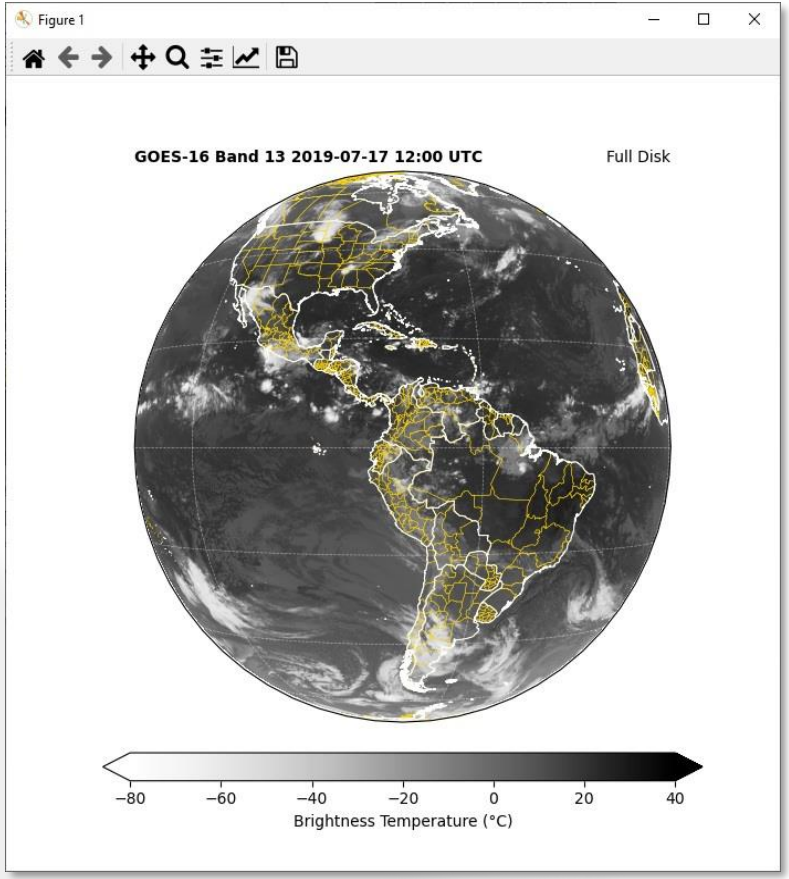
- Change the appearance of the shapefile
- Change the shapefile to be read

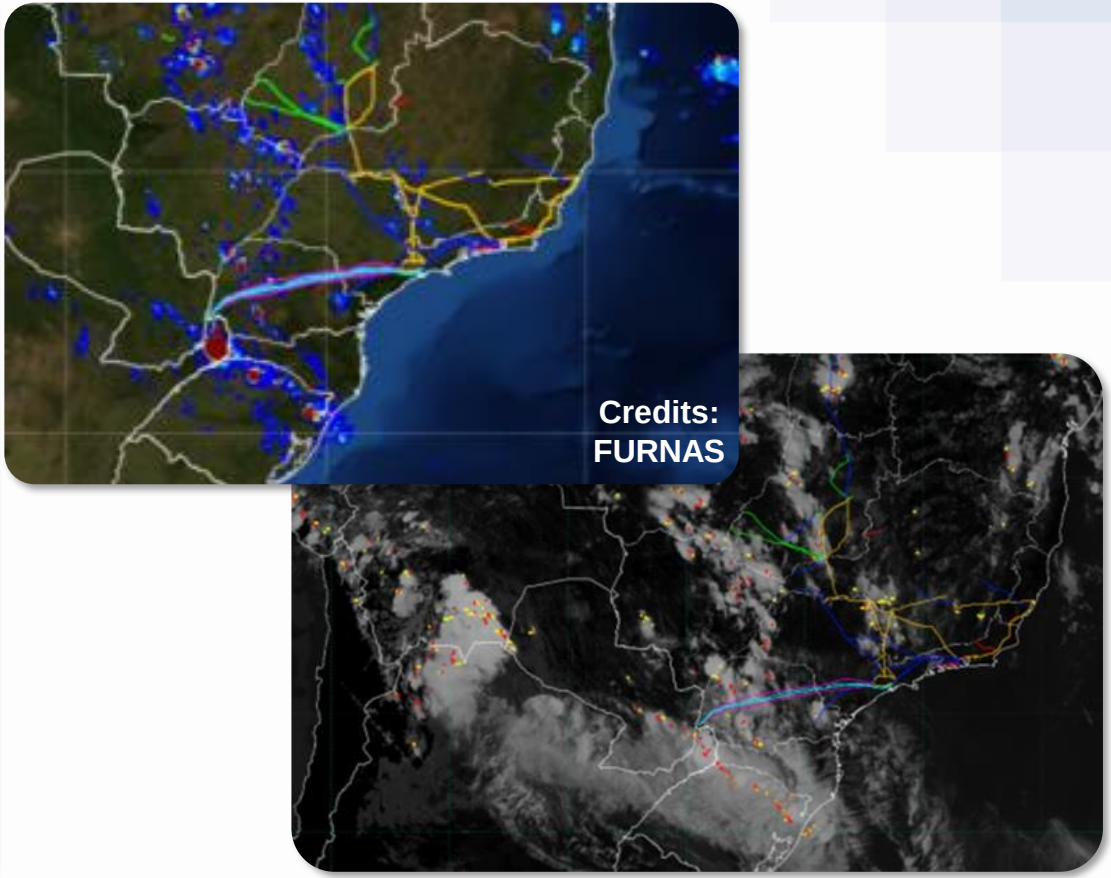
Script 5: Reading a Shapefile





Script 5: Reading a Shapefile





Script 6: Plotting Contours and Labels

```
#
# INPE / CPTEC Training: NWP Data Processing With Python - Script 6: Plotting Contours and Labels
# Authors: Diego Souza
#
#-------------------------------------------------------------------------------------------------
import pygrib                  # Provides a high-level interface to the ECMWF ECCODES C library for reading GRIB files
import matplotlib.pyplot as plt # Plotting library
import cartopy, cartopy.crs as ccrs # Plot maps
import cartopy.io.shapereader as shpreader # Import shapefiles
import numpy as np             # Scientific computing with Python
#-------------------------------------------------------------------------------------------------
```

LIBRARIES

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

# Select the variable
grb = grib.select(name='2 metre temperature')[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')

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

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

#-------------------------------------------------------------------------------------------------
# Convert from K to °C
tmtmp = tmtmp - 273.15
```

DATA READING AND MANIPULATION

```
plt.figure(figsize=(8,8))

# Use the Cylindrical Equidistant projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree())

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='red',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.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)
gl.top_labels = False
gl.right_labels = False

# Define de contour interval
data_min = -20
data_max = 48
interval = 2
levels = np.arange(data_min,data_max,interval)

# Plot the contours
img1 = ax.contourf(lons, lats, tmtmp, cmap='jet', levels=levels, extend='both')
img2 = ax.contour(lons, lats, tmtmp, colors='white', linewidths=0.3, levels=levels)
ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10',fmt = '%1.0f', colors= 'black')

# Add a colorbar
plt.colorbar(img1, label='2 m Temperature (°C)', orientation='vertical', pad=0.05, fraction=0.05)

# Add a title
plt.title('GFS: 2 m Temperature' , fontweight='bold', fontsize=10, loc='left')
```

PLOT CONFIGURATION

```
# Save the image
plt.savefig('image_6.png')

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

IMAGE GENERATION

Script 6: Plotting Contours and Labels

Explaining the new instructions:

CONFIGURAÇÃO DO PLOT

```
#----------------------------------------------------------------------------------------
# Convert from K to °C
tmtmp = tmtmp - 273.15

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

# Use the Cilindrical Equidistant projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree())

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='red',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.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)
gl.top_labels = False
gl.right_labels = False

# Define de contours values
data_min = -20
data_max = 48
interval = 2
levels = np.arange(data_min,data_max,interval)

# Plot the contours
img1 = ax.contourf(lons, lats, tmtmp, cmap='jet', levels=levels, extend='both')
img2 = ax.contour(lons, lats, tmtmp, colors='white', linewidths=0.3, levels=levels)
ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10',fmt = '%1.0f', colors= 'black')

# Add a colorbar
plt.colorbar(img1, label='2 m Temperature (°C)', orientation='vertical', pad=0.05, fraction=0.05)

# Add a title
plt.title('GFS: 2 m Temperature' , fontweight='bold', fontsize=10, loc='left')
plt.title('Valid: ' + valid, fontsize=10, loc='right')
#----------------------------------------------------------------------------------------
# Save the image
plt.savefig('image_6.png')
```

Min. value

Max. value

Interval

Creation of the array

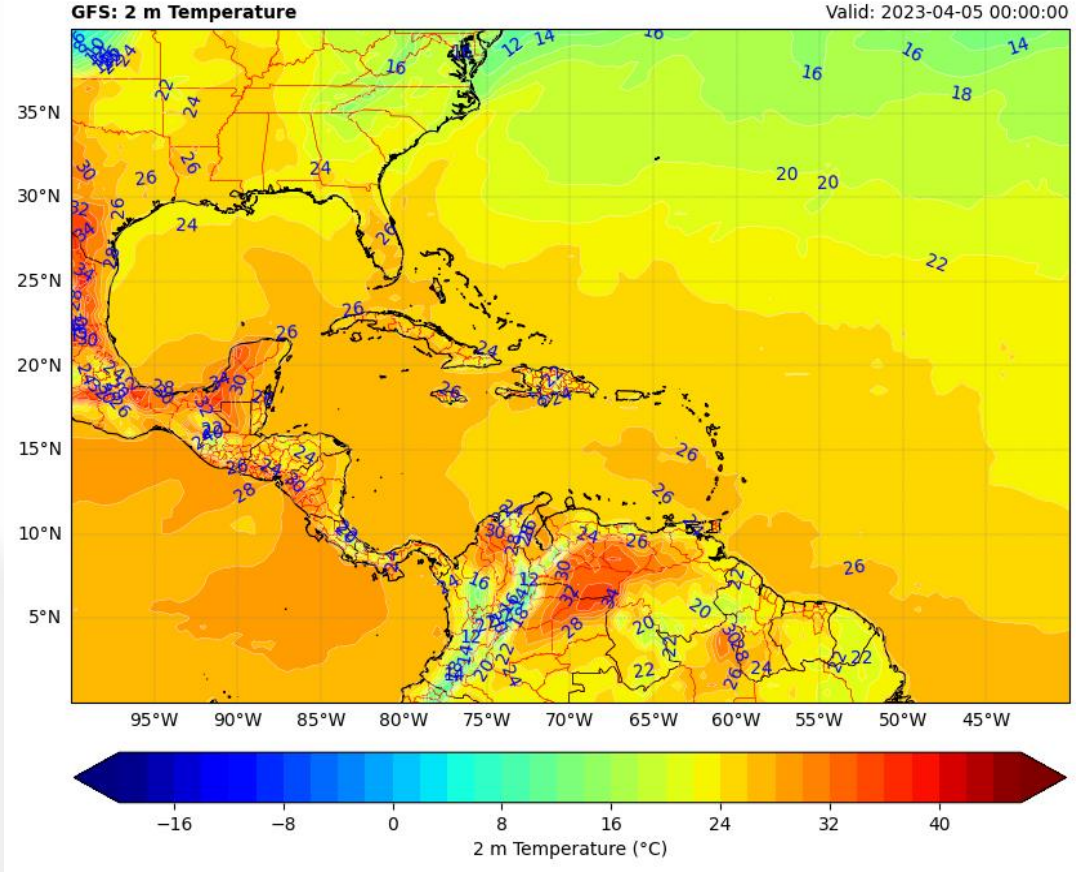
Filled contour

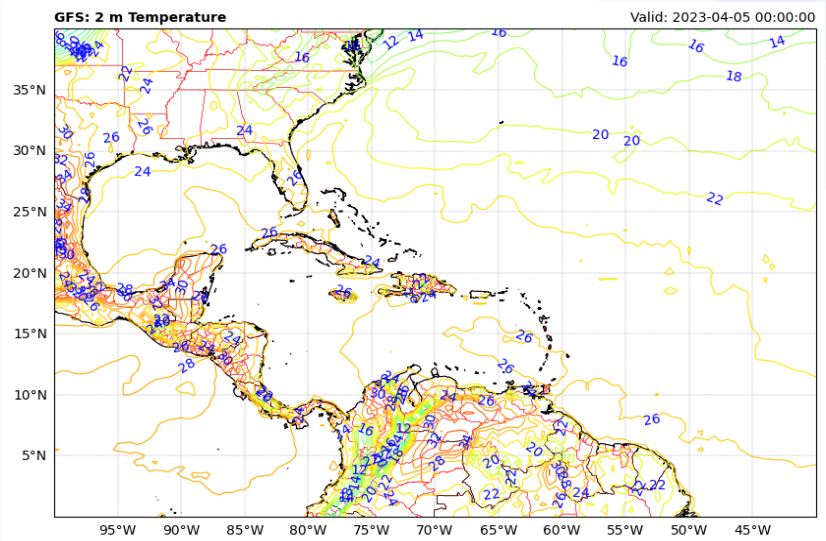
Unfilled contour

Label

Configuration

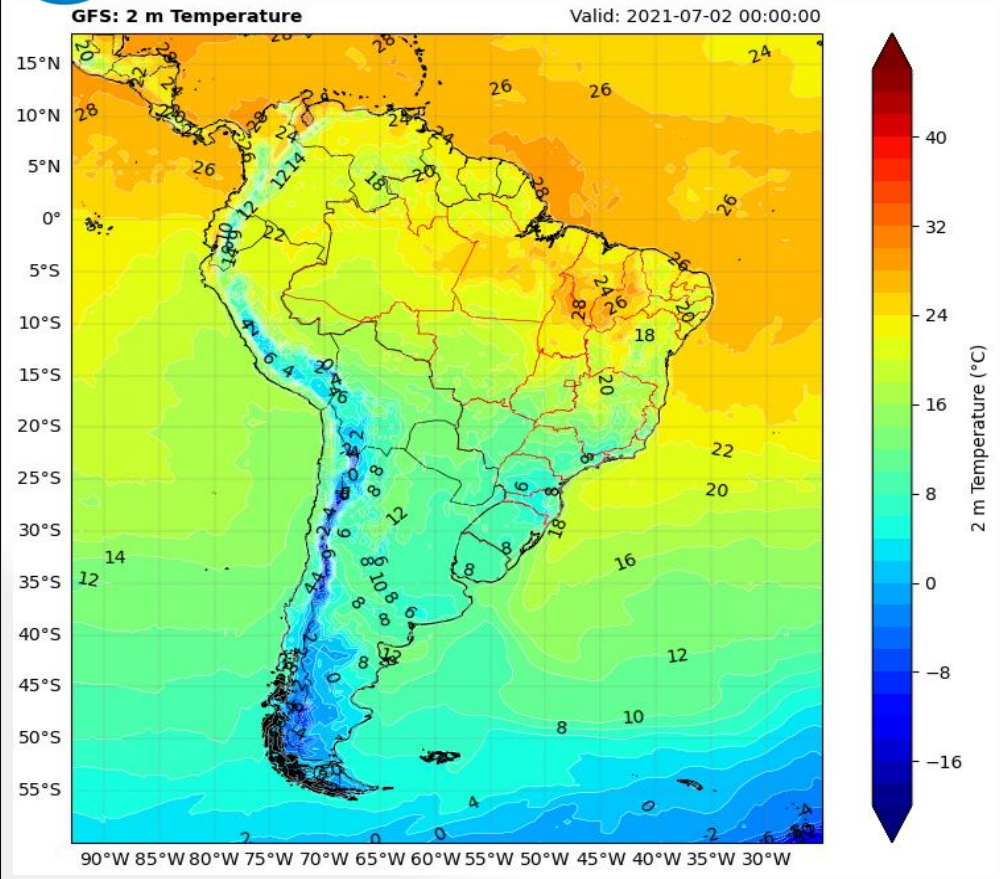
Script 6: Plotting Contours and Labels

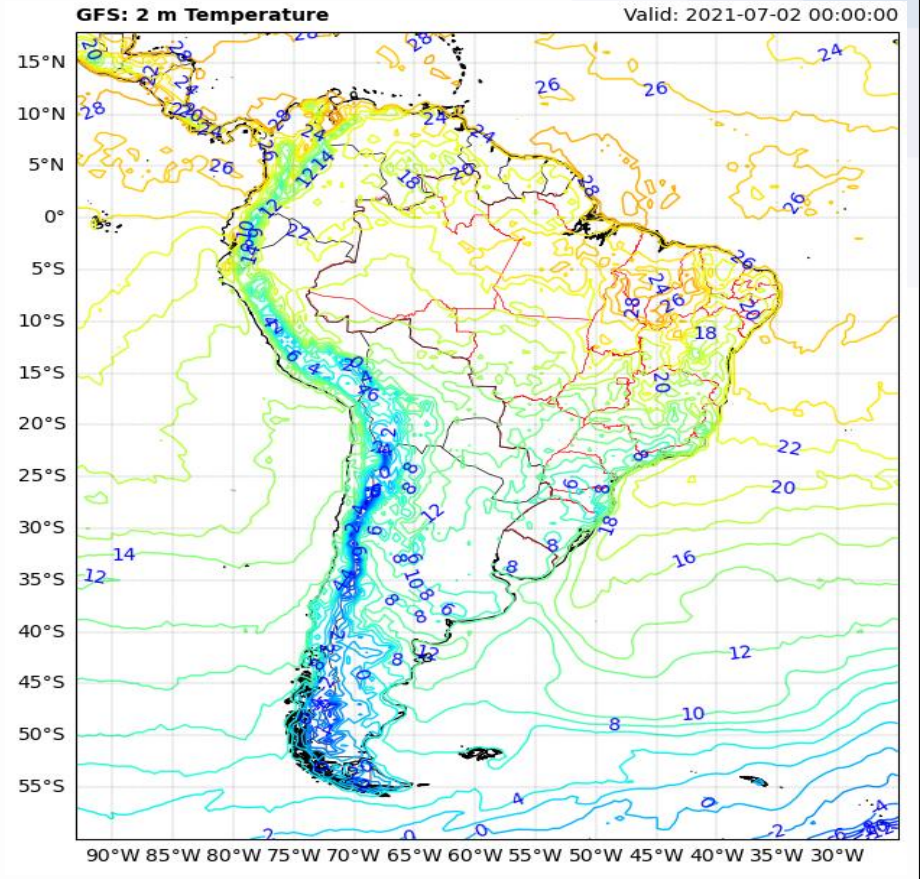




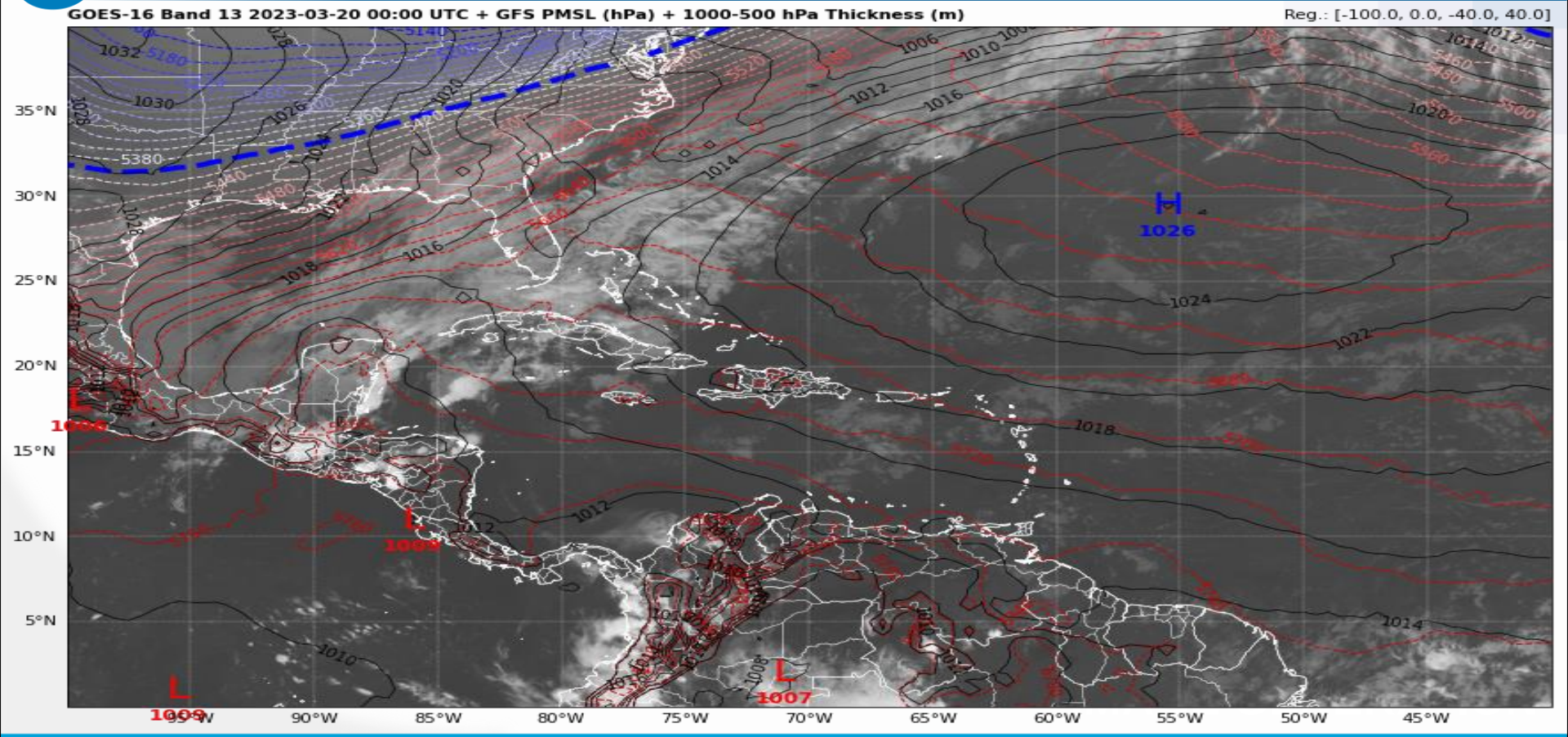

```
img2 = ax.contour(lons, lats, tmtmp, cmap='jet', linewidths=0.8, levels=levels)
```

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

```python
#-----------------------------------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing With Python - Script 7: Custom Color Palettes
# Author: Diego Souza
#-----------------------------------------------------------------------------------------------------------
import pygrib                                # Provides a high-level interface to the ECMWF ECCODES C library for reading GRIB files
import matplotlib.pyplot as plt              # Plotting library
import cartopy, cartopy.crs as ccrs          # Plot maps
import cartopy.io.shapereader as shpreader   # Import shapefiles
import numpy as np                           # Scientific computing with Python
import matplotlib                            # Comprehensive library for creating static, animated, and interactive visualizations in Python

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

# Select the variable
grb = grib.select(name='2 metre temperature')[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')

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

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

#-----------------------------------------------------------------------------------------------------------
# Convert from K to °C
tmtmp = tmtmp - 273.15

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

# Use the Cilindrical Equidistant projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree())

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.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)
gl.top_labels = False
gl.right_labels = False

# Define de contour interval
data_min = -2
data_max = 38
interval = 2
levels = np.arange(data_min,data_max,interval)

# Create a custom color palette
colors = ["#d3d2d2", "#bcbcbc", "#969696", "#1464d2", "#1e6eeb", "#2882f0",
"#3c9655", "#50a555", "#78b95a", "#96d25a", "#b4f05a", "#1eb41e", "#37d23c",
"#50f050", "#78f573", "#96f58c", "#b4faaa", "#c8ffbe", "#ffe878", "#ffc03c",
"#ffa000", "#ff6000", "#ff3200", "#e11400", "#c00000", "#a50000", "#785046",
"#8c6359", "#b48b82", "#e1beb4"]
cmap = matplotlib.colors.ListedColormap(colors)
cmap.set_over('#fadad5')
cmap.set_under('#e5e5e5')

# Plot the contours
img1 = ax.contourf(lons, lats, tmtmp, transform=ccrs.PlateCarree(), cmap=cmap, levels=levels, extend='both')
img2 = ax.contour(lons, lats, tmtmp, transform=ccrs.PlateCarree(), colors='white', linewidths=0.3, levels=levels)
ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10',fmt = '%1.0f', colors= 'black')

# Add a colorbar
plt.colorbar(img1, label='2 m Temperature (°C)', orientation='vertical', pad=0.05, fraction=0.05)

# Add a title

#-----------------------------------------------------------------------------------------------------------
# Save the image
plt.savefig('image_7.png')

# Show the image
```

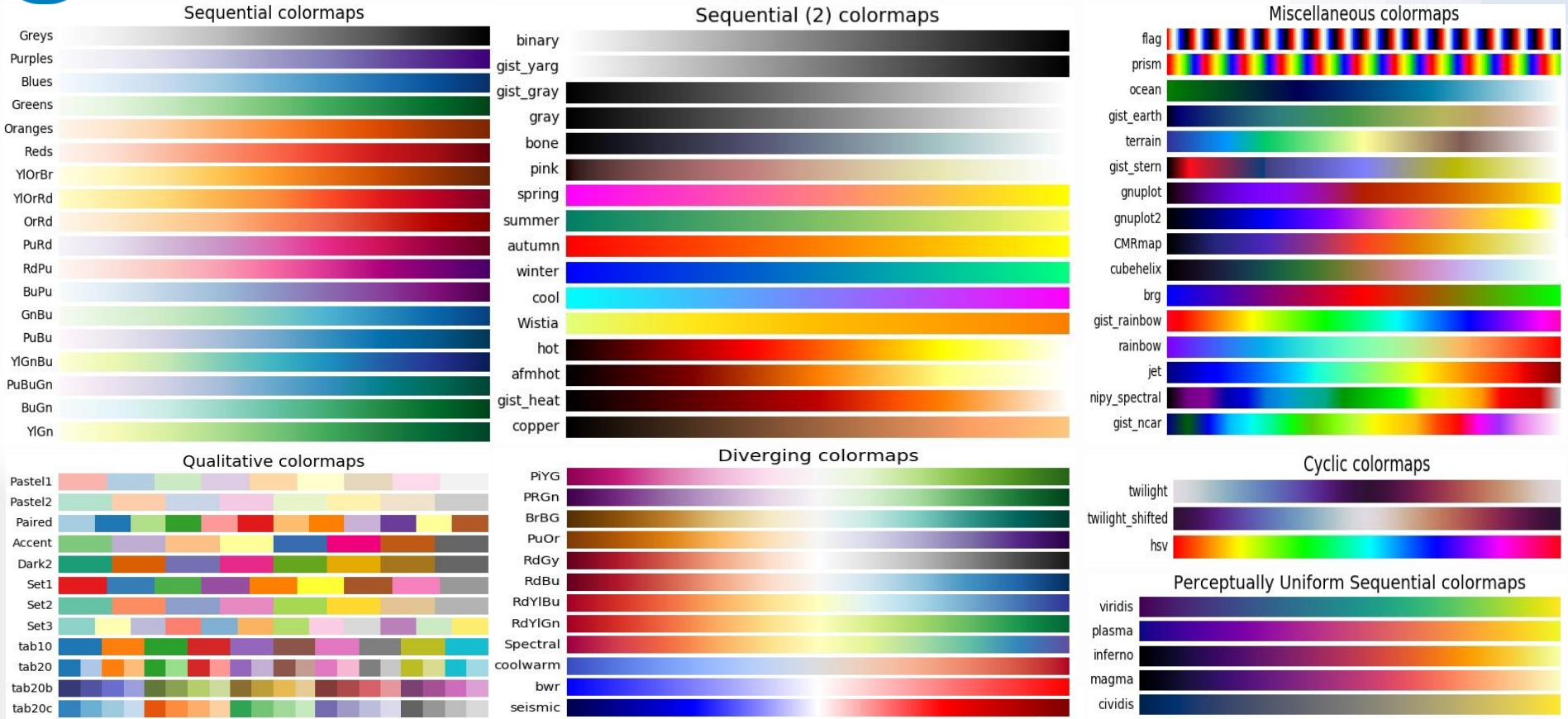
LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

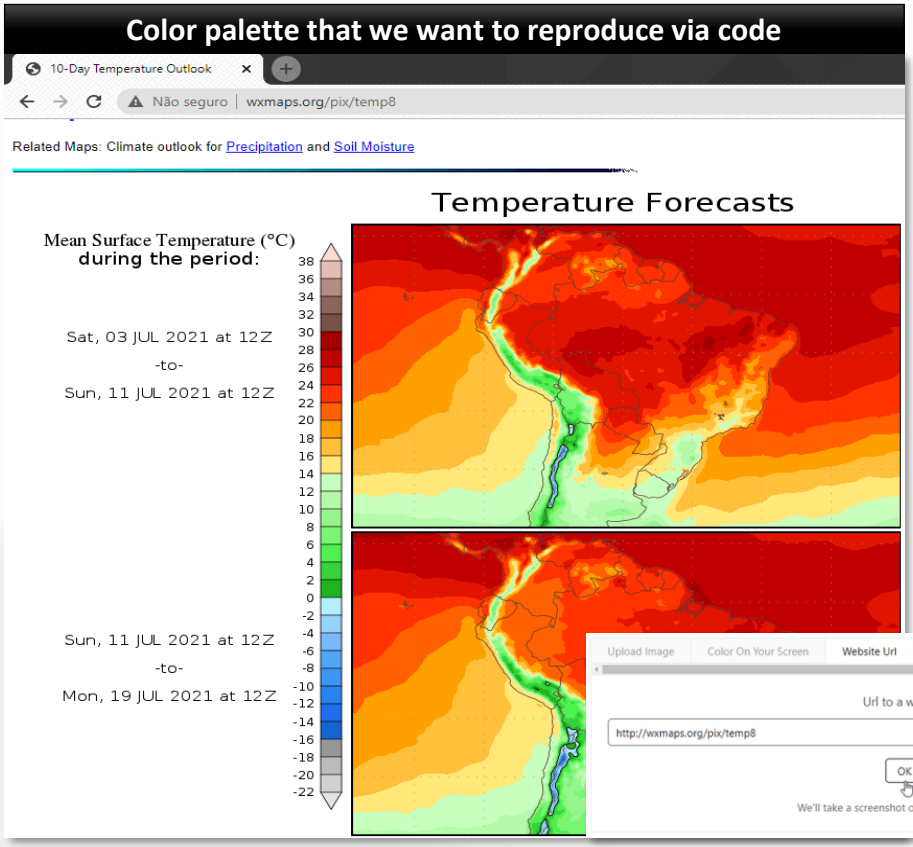
Script 7: Custom Color Palettes

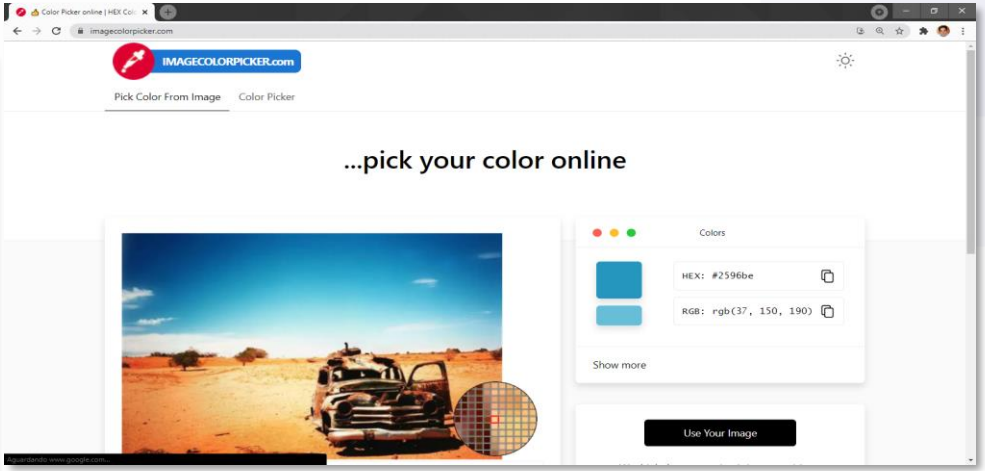


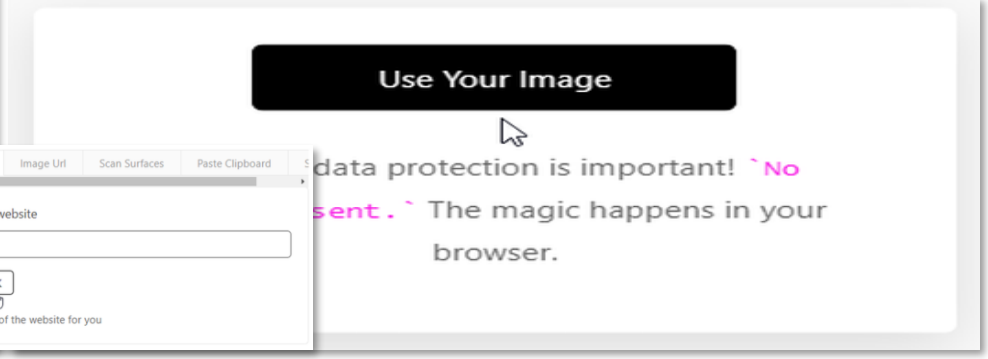
Script 7: Custom Color Palettes

http://wxmaps.org/pix/temp8

https://imagecolorpicker.com/

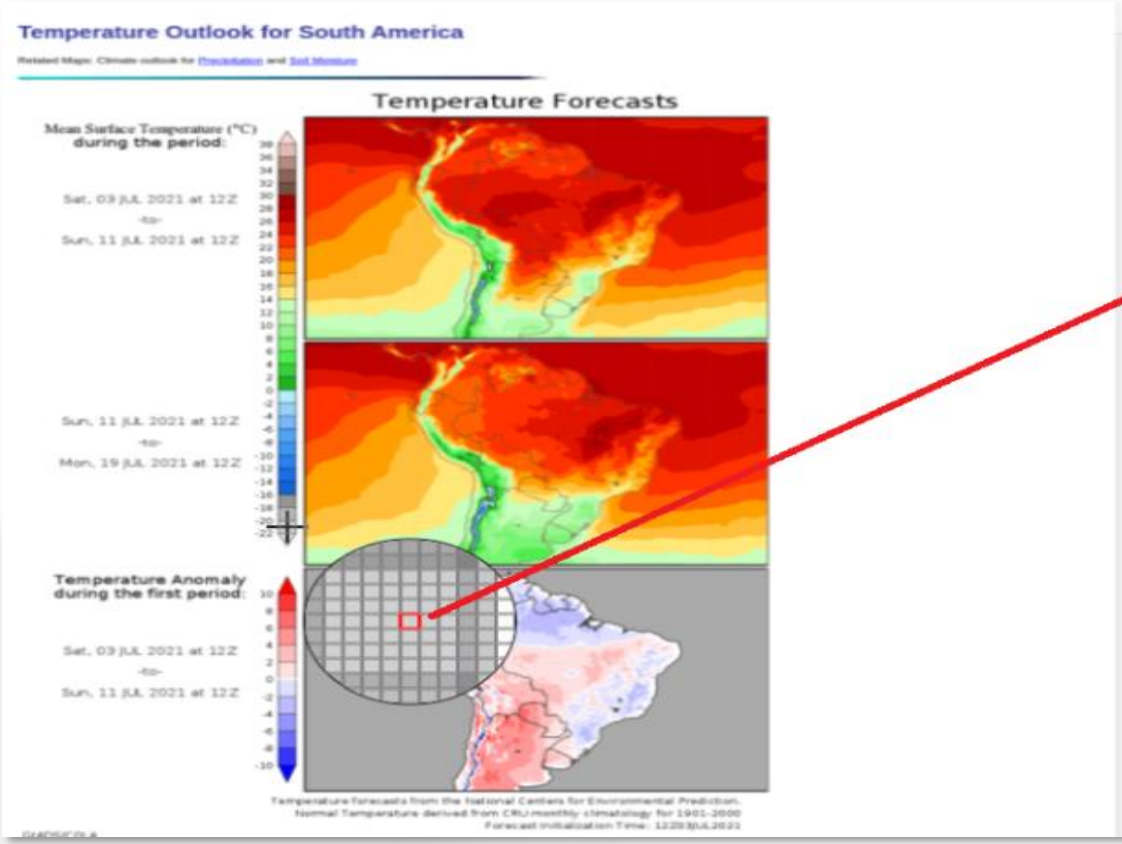


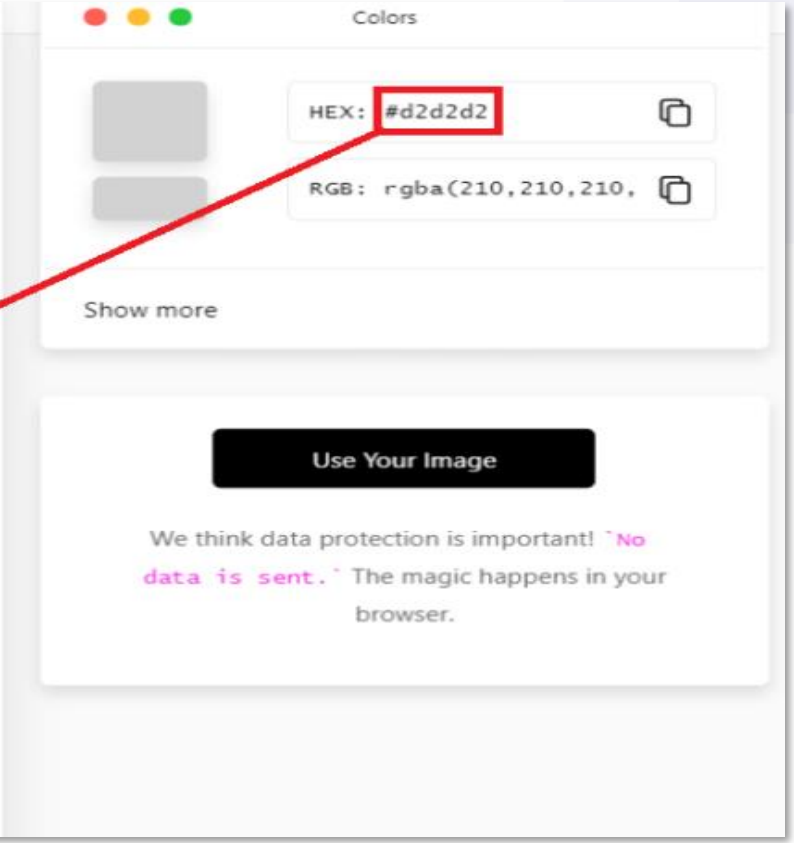




Script 7: Custom Color Palettes

https://imagecolorpicker.com/





Script 7: Custom Color Palettes

Explaining the new instructions:

LIBRARIES

Imports Matplotlib (General)

DATA READING AND MANIPULATION

```
#----------------------------------------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing With Python - Script 7: Custom Color Palettes
# Author: Diego Souza
#----------------------------------------------------------------------------------------------------------------
import pygrib                                # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files
import matplotlib.pyplot as plt              # Plotting library
import cartopy, cartopy.crs as ccrs          # Plot maps
import cartopy.io.shapereader as shpreader   # Import shapefiles
import numpy as np                           # Scientific computing with Python
import matplotlib                            # Comprehensive library for creating static, animated, and interactive visualizations in Python
#----------------------------------------------------------------------------------------------------------------

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

# Select the variable
grb = grib.select(name='2 metre temperature')[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')

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

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

#----------------------------------------------------------------------------------------------------------------
# Convert from K to °C
tmtmp = tmtmp - 273.15

#----------------------------------------------------------------------------------------------------------------
```

Script 7: Custom Color Palettes

PLOT CONFIGURATION

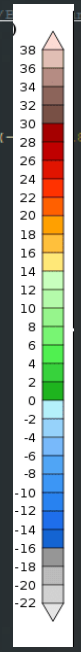
```
# Choose the plot size (width x height, in inches)
plt.figure(figsize=(8,8))

# Use the Cilindrical Equidistant projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree())

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/unidades_da_federacao.zip
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.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)
gl.top_labels = False
gl.right_labels = False

# Define de contour interval
data_min = -2
data_max = 38
interval = 2
levels = np.arange(data_min,data_max,interval)

# Create a custom color palette
colors = ["#d3d2d2", "#bcbcbc", "#969696", "#1464d2", "#1e6eeb", "#2882f0",
"#3c96f5", "#50a5f5", "#78b9fa", "#96d2fa", "#b4f0fa", "#1eb41e", "#37d23c",
"#50f050", "#78f573", "#96f58c", "#b4faaa", "#c8ffbe", "#ffe878", "#ffc03c",
"#ffa000", "#ff6000", "#ff3200", "#e11400", "#c00000", "#a50000", "#785046",
"#8c6359", "#b48b82", "#e1beb4"]
cmap = matplotlib.colors.ListedColormap(colors)
cmap.set_over('#fadad5')
cmap.set_under('#e5e5e5')

# Plot the contours
img1 = ax.contourf(lons, lats, tmtmp, transform=ccrs.PlateCarree(), cmap=cmap, levels=levels, extend='both')
img2 = ax.contour(lons, lats, tmtmp, transform=ccrs.PlateCarree(), colors='white', linewidths=0.5, levels=levels)
ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10',fmt = '%1.0f', colors= 'black')

# Add a colorbar
plt.colorbar(img1, label='2 m Temperature (°C)', orientation='vertical', pad=0.05, fraction=0.05)

# Add a title
plt.title('GFS: 2 m Temperature' , fontweight='bold', fontsize=10, loc='left')
plt.title('Valid: ' + valid, fontsize=10, loc='right')
```

List of Colors (Hex values)

Create the new colormap

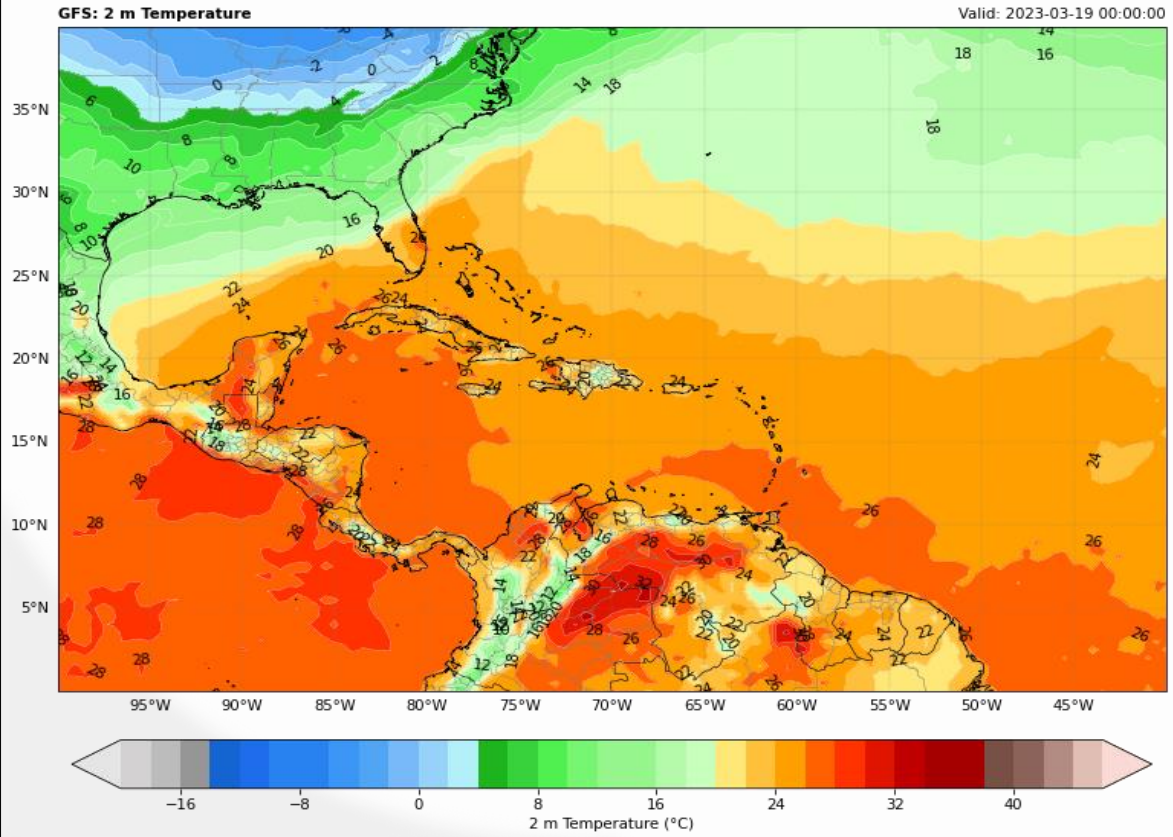
Values over max

Values under min

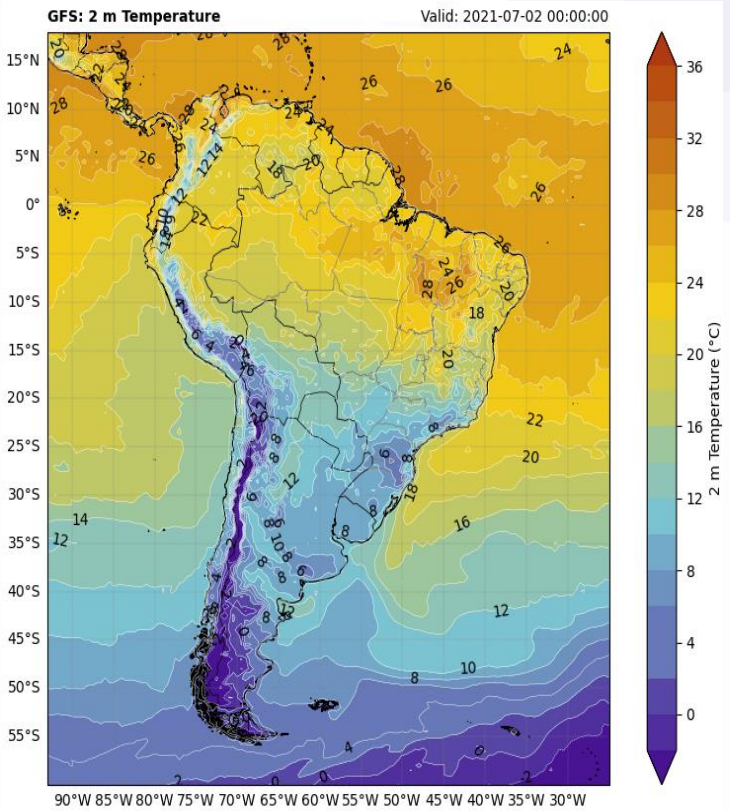


Script 7: Custom Color Palettes

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

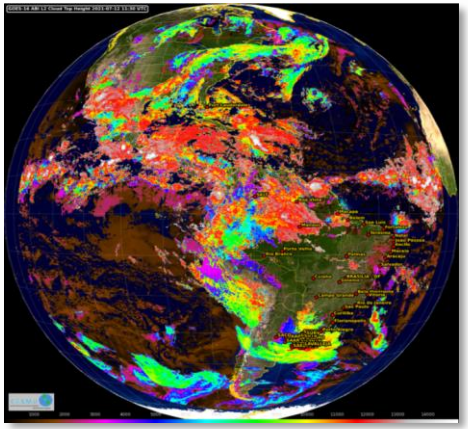


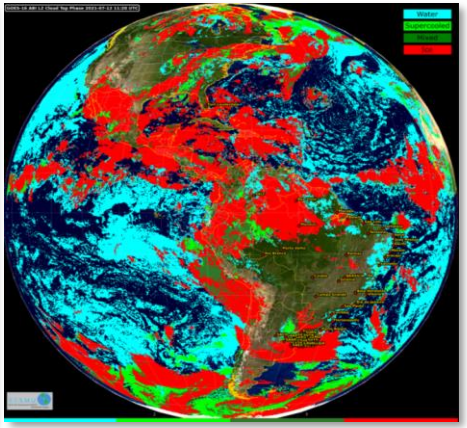
Reference: https://previsaonumerica.cptec.inpe.br/

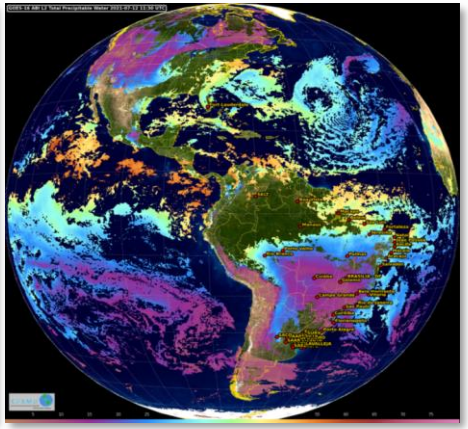


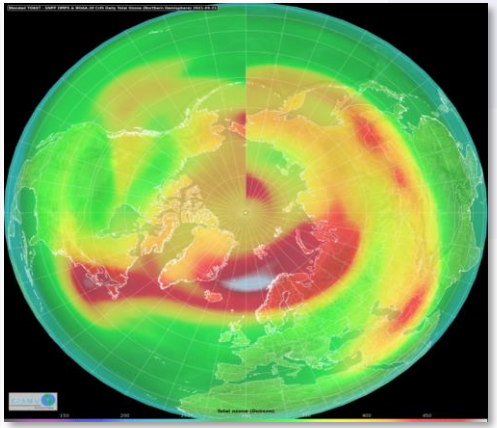
Script 7: Custom Color Palettes

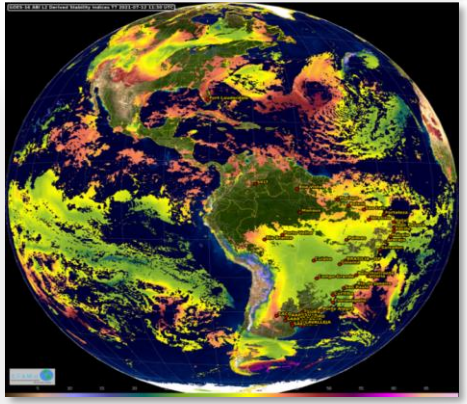
Some examples of satellite data plots with custom color palettes

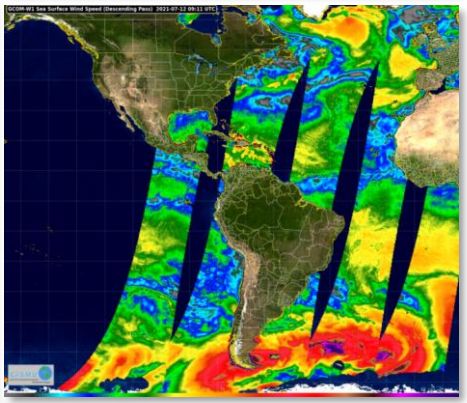


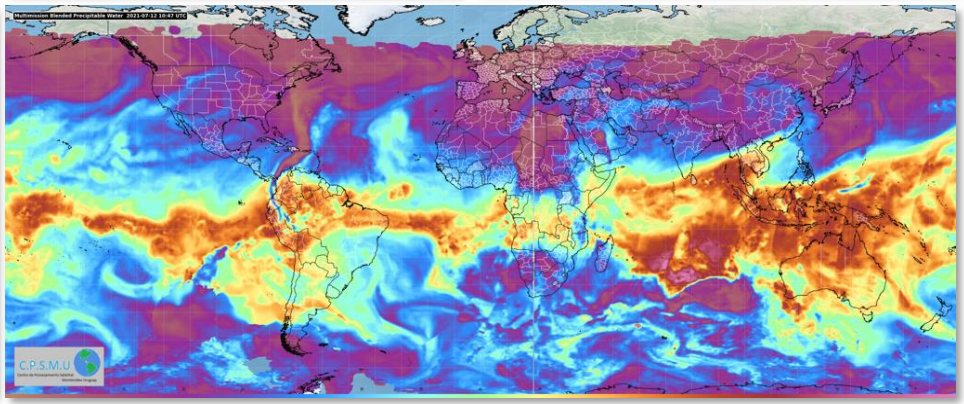












Script 8: Smoothing Contours

```python
#-----------------------------------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing With Python - Script 8: Smoothing the Contours
# Author: Diego Souza
#-----------------------------------------------------------------------------------------------------------
import pygrib                                    # Provides a high-level interface to the ECMWF ECCODES C library for reading GRIB files
import matplotlib.pyplot as plt                  # Plotting library
import cartopy, cartopy.crs as ccrs              # Plot maps
import cartopy.io.shapereader as shpreader       # Import shapefiles
import numpy as np                               # Scientific computing with Python
import matplotlib                                # Comprehensive library for creating static, animated, and interactive visualizations in Python

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

# Select the variable
grb = grib.select(name='2 metre temperature')[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')

# Select the extent [min. lon, min. lat, max. lon, max. lat]
extent = [-55.0, -15.00, -35.00, 5.00]

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

#-----------------------------------------------------------------------------------------------------------
# Convert from K to °C
tmtmp = tmtmp - 273.15

print("\nArray dimensions before smoothing:")
print(tmtmp.shape)

# Smooth the contours
import scipy.ndimage
tmtmp = scipy.ndimage.zoom(tmtmp, 3)
lats = scipy.ndimage.zoom(lats, 3)
lons = scipy.ndimage.zoom(lons, 3)

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

# Use the Cilindrical Equidistant projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree())

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]

# Add a shapefile
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.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)
gl.top_labels = False
gl.right_labels = False

# Define de contour interval
data_min = -20
data_max = 48
interval = 2
levels = np.arange(data_min,data_max,interval)

# Create a custom color palette
colors = ["#d3d2d2", "#bcbcbc", "#969696", "#1464d2", "#1e6eeb", "#2882f0",
"#3c96f5", "#50a5f5", "#78b9fa", "#96d2fa", "#b4f0fa", "#1eb41e", "#37d23c",
"#50f050", "#78f573", "#96f58c", "#b4faaa", "#c8ffbe", "#ffe878", "#ffc03c",
"#ffa000", "#ff6000", "#ff3200", "#e11400", "#c00000", "#a50000", "#785046",
"#8c6359", "#b48b82", "#e1beb4"]
cmap = matplotlib.colors.ListedColormap(colors)
cmap.set_over('#fadad5')
cmap.set_under('#e5e5e5')

# Plot the contours
img1 = ax.contourf(lons, lats, tmtmp, transform=ccrs.PlateCarree(), cmap=cmap, levels=levels, extend='both')
img2 = ax.contour(lons, lats, tmtmp, transform=ccrs.PlateCarree(), colors='white', linewidths=0.3, levels=levels)
ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10',fmt = '%1.0f', colors= 'black')

# Add a colorbar
plt.colorbar(img1, label='2 m Temperature (°C)', orientation='vertical', pad=0.05, fraction=0.05)

# Add a title

#-----------------------------------------------------------------------------------------------------------
# Save the image
plt.savefig('image_8.png')

# Show the image
```

LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

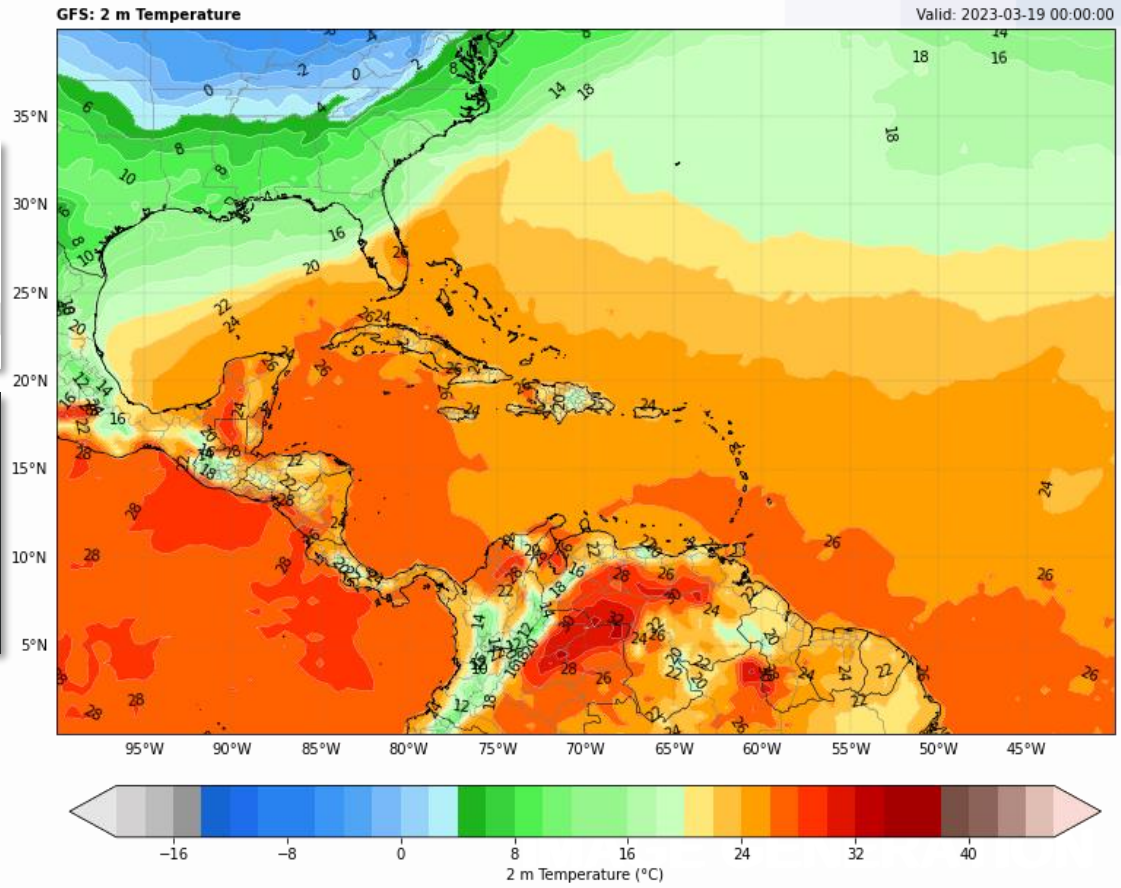
Script 8: Smoothing Contours

What is the size of our array?

```
#----------------------------------------
# Convert to Celsius
tmtmp = tmtmp - 273.15

# Print the array dimensions
print("Array dimensions :", tmtmp.shape)
#----------------------------------------
```

```
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

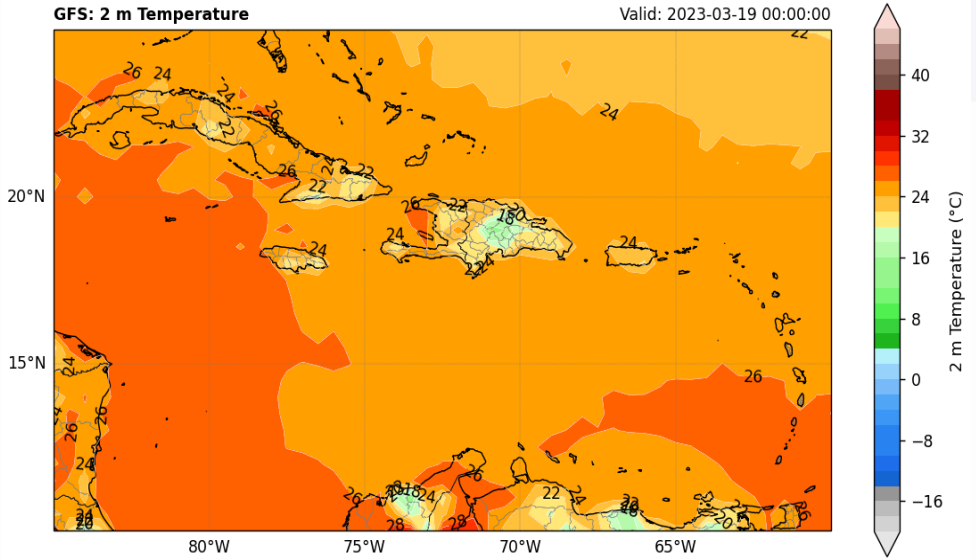


IMAGE GENERATION

Script 8: Smoothing Contours

Explaining the new instructions:

DATA READING AND MANIPULATION

```
#-------------------------------------------------------------------------------------------------------------------

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

# Select the variable
grb = grib.select(name='2 metre temperature')[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')

# Select the extent [min. lon, min. lat, max. lon, max. lat]
extent = [-55.0, -15.00, -35.00, 5.00]

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

#-------------------------------------------------------------------------------------------------------------------
# Convert from K to °C
tmtmp = tmtmp - 273.15

print("\nArray dimensions before smoothing:")
print(tmtmp.shape)

# Smooth the contours
import scipy.ndimage
tmtmp = scipy.ndimage.zoom(tmtmp, 3)
lats = scipy.ndimage.zoom(lats, 3)
lons = scipy.ndimage.zoom(lons, 3)

print("Array dimensions after smoothing:")
print(tmtmp.shape)
#-------------------------------------------------------------------------------------------------------------------
```

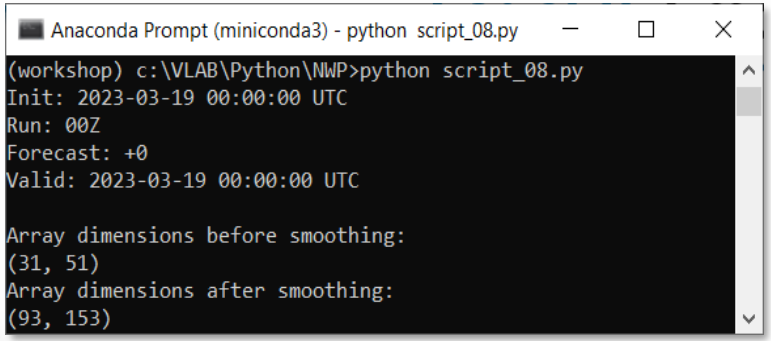
Imports the “ndimage” library

Apply the “zoom” in the data array

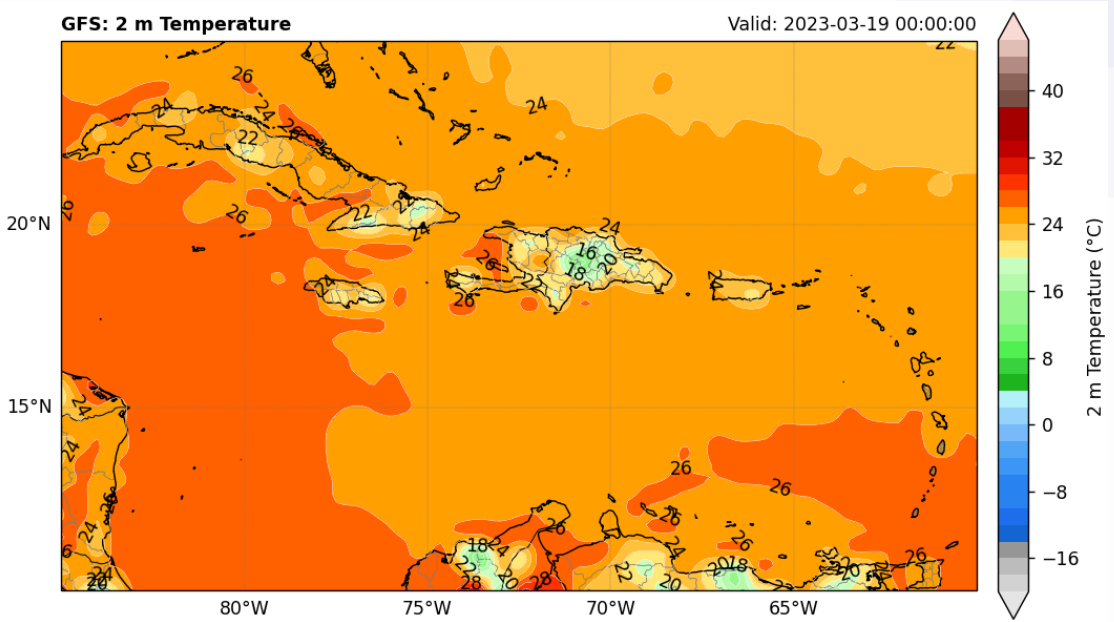
Apply the “zoom” in the latitude array

Apply the “zoom” in the longitude array

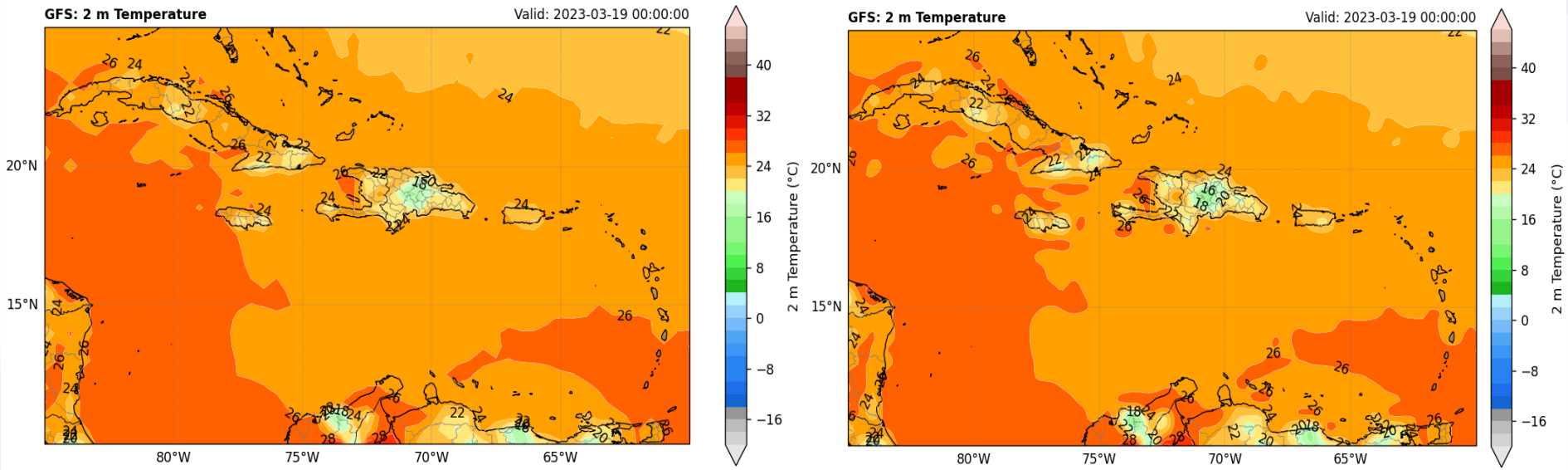
Script 8: Smoothing Contours



New dimensions after smoothing

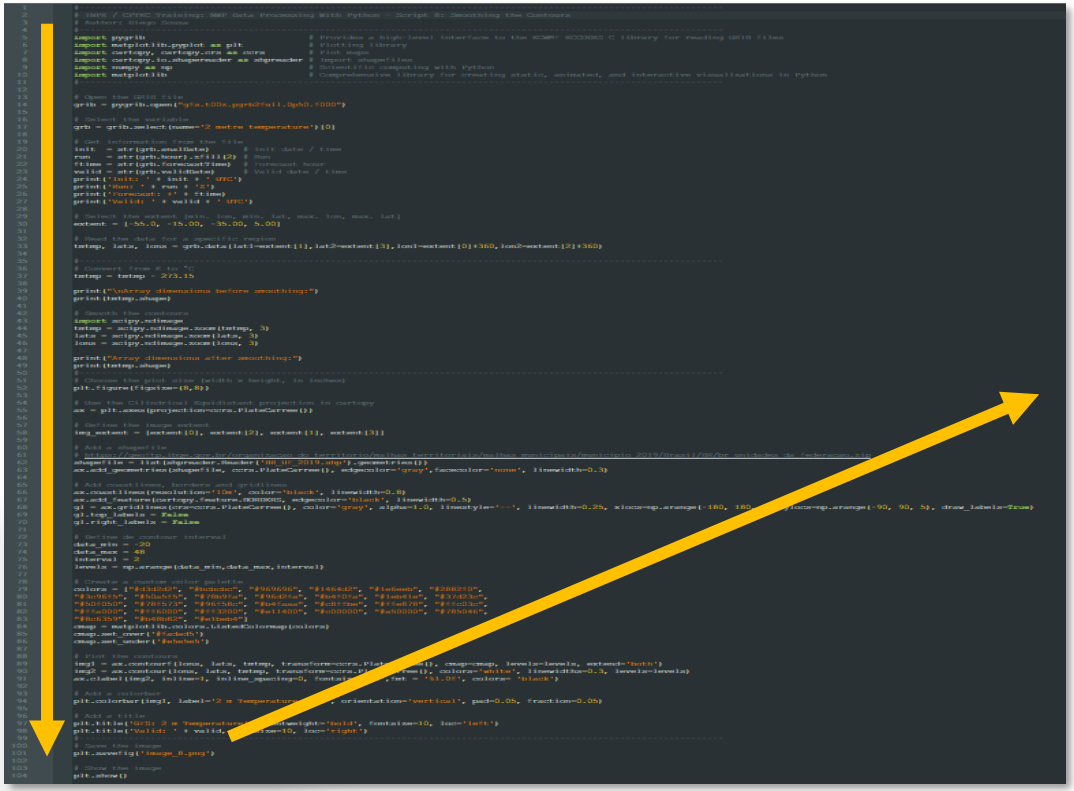


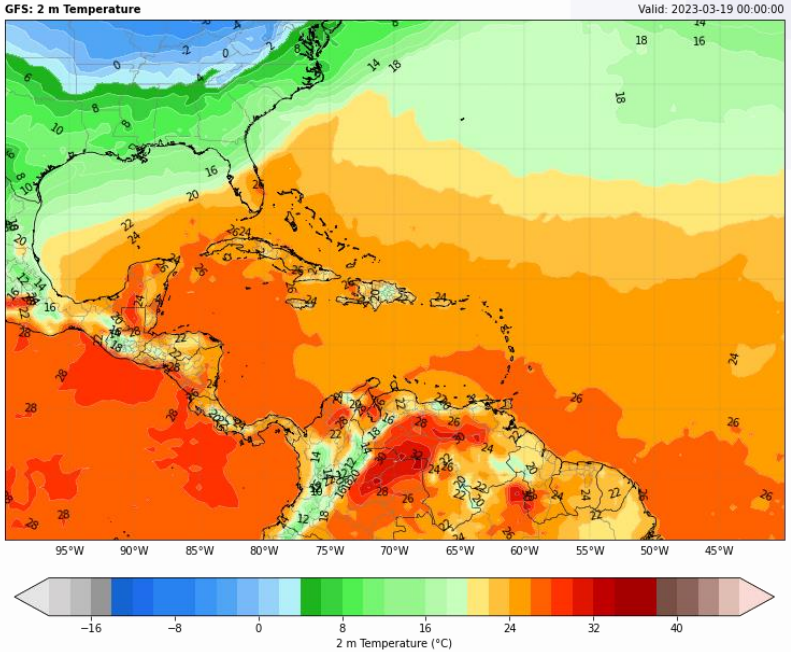
Script 8: Smoothing Contours



Script 9: Working With Multiple Files

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





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

Script 9: Working With Multiple Files

LIBRARIES

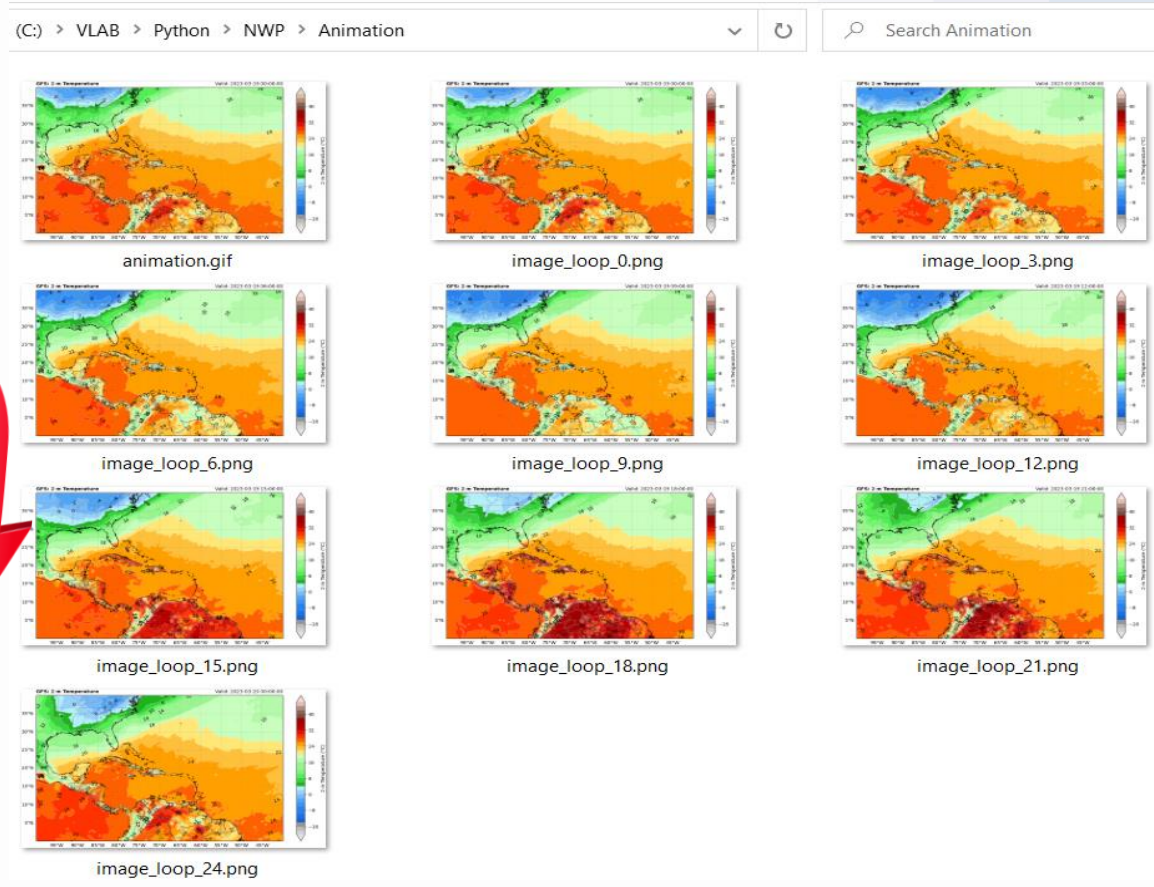
PREV. CONFIGURATION

LOOP

DATA READING AND MANIP.

PLOT CONF.

IMAGE GENERATION



Script 9: Working With Multiple Files

Explaining the new instructions:

```
#----------------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing With Python - Script 9: Working With Multiple Files
# Author: Diego Souza
#----------------------------------------------------------------------------------------
import pygrib                           # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files
import matplotlib.pyplot as plt         # Plotting library
import cartopy, cartopy.crs as ccrs     # Plot maps
import cartopy.io.shapereader as shpreader # Import shapefiles
import numpy as np                      # Scientific computing with Python
import matplotlib                       # Comprehensive library for creating static, animated, and interactive visualizations in Python
import os                               # Miscellaneous operating system interfaces
#----------------------------------------------------------------------------------------

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

# 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 in the loop
hour_ini = 0    # Init time
hour_end = 24   # End time
hour_int = 3    # Interval

#----------------------------------------------------------------------------------------

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

    # File to process
    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)
```

LIBRARIES

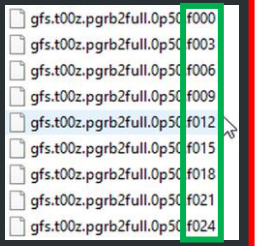
Imports the O.S. library (operating system interface)

INITIAL CONFIGURATION

Create a folder named “Animation”

GRIB File Name without Last 3 Characters

Start, End and Interval Time



LOOP

For each time interval between start time and end time

GRIB File Name + Time (three characters)

If file exists...

… process the file

Script 9: Working With Multiple Files

Explaining the new instructions:

PLOT CONFIGURATION

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

# Use the Cilindrical Equidistant projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree())

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.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)
gl.top_labels = False
gl.right_labels = False

# Define de contour interval
data_min = -20
data_max = 48
interval = 2
levels = np.arange(data_min,data_max,interval)

# Create a custom color palette
colors = ["#d3d2d2", "#bcbcbc", "#969696", "#1464d2", "#1e6eeb", "#2882f0",
"#3c96f5", "#50a5f5", "#78b9fa", "#96d2fa", "#b4f0fa", "#leb41e", "#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')

# Plot the contours
img1 = ax.contourf(lons, lats, tmtmp, transform=ccrs.PlateCarree(), cmap=cmap, levels=levels, extend='both')
img2 = ax.contour(lons, lats, tmtmp, transform=ccrs.PlateCarree(), colors='white', linewidths=0.3, levels=levels)
ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10',fmt = '%1.0f', colors= 'black')

# Add a colorbar
plt.colorbar(img1, label='2 m Temperature (°C)', orientation='vertical', pad=0.05, fraction=0.05)

# Add a title
plt.title('GFS: 2 m Temperature' , fontweight='bold', fontsize=10, loc='left')
plt.title('Valid: ' + valid, fontsize=10, loc='right')
#-------------------------------------------------------------------------------


# Save the image
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

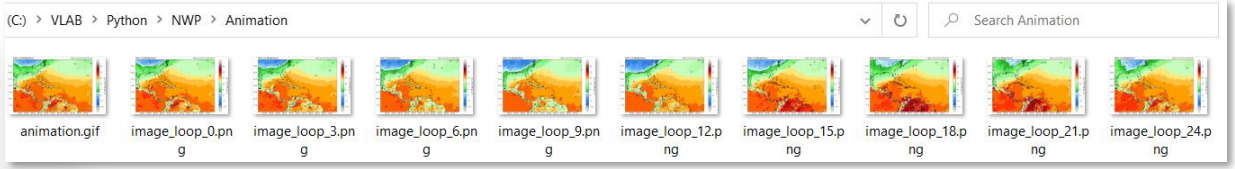
```python
#---------------------------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing With Python - Script 10: Creating an Animation
# Author: Diego Souza
#---------------------------------------------------------------------------------------------------
import imageio.v2 as imageio          # Python interface to read and write a wide range of image data

# Images we want to include in the GIF
files = ['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']

# Create the GIF
images = []
for file in files:
    images.append(imageio.imread('Animation//' + file))

# Save the GIF
imageio.mimsave('Animation//animation.gif', images, fps=1)
```

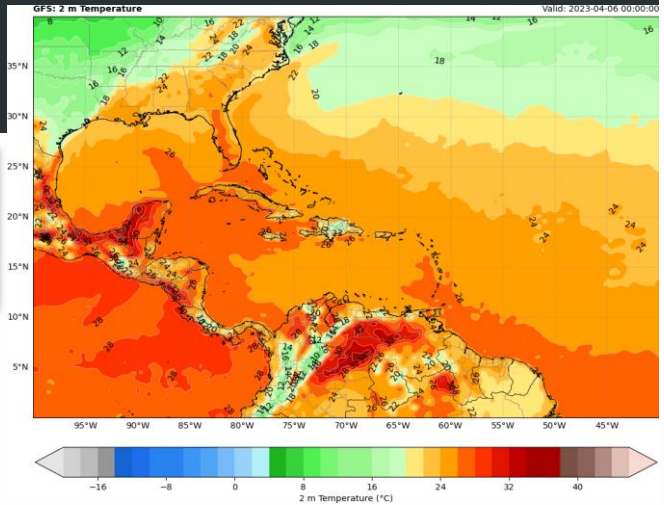
Import the imageio library

List of images to be included in the animation

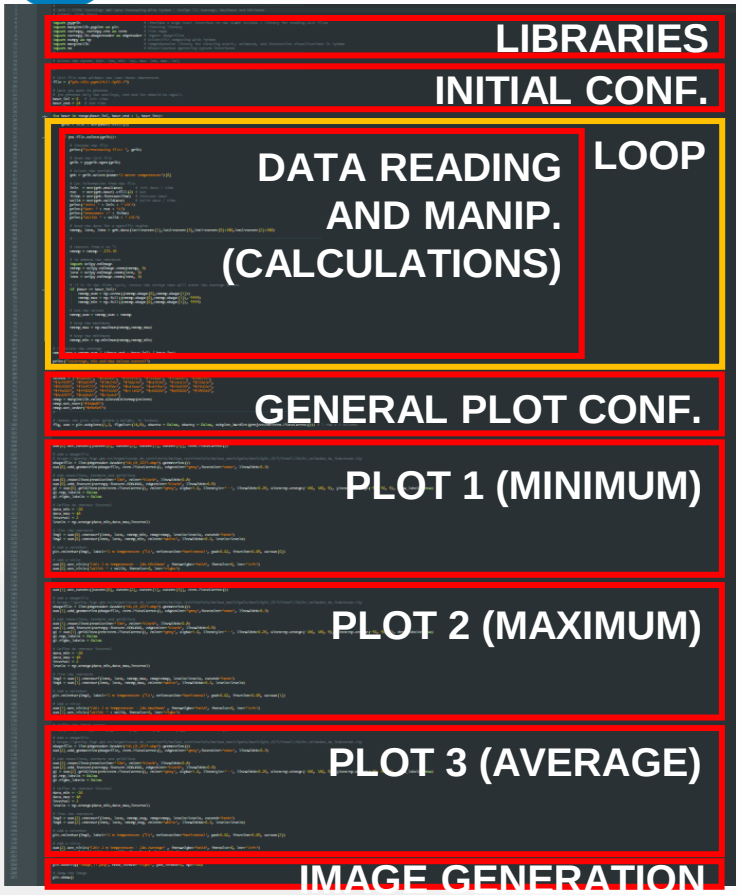
For PNG file in list above, add to GIF

Generates the GIF, with 1 frame per second





Script 11: Averages, Maximums and Minimums - Simultaneous Multiple Plots




```python
#-----------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing With Python - Script 11: Average, Maximuns
# Author: Diego Souza
#-----------------------------------------------------------------------------
import pygrib                                # Provides a high-level interface to read and write GRIB files
import matplotlib.pyplot as plt              # Plotting library
import cartopy, cartopy.crs as ccrs          # Plot maps
import cartopy.io.shapereader as shpreader   # Import shapefiles
import numpy as np                           # Scientific computing with Python
import matplotlib                            # Comprehensive library for creating static,
import os                                    # 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 analisys, 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)
```

LIBRARIES

INITIAL CONF.

LOOP

Our scripts are getting bigger

Script 11: Averages, Maximums and Minimums - Simultaneous Multiple Plots

Explaining the new instructions:

LOOP

DATA READING AND MANIPULATION

```
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)

        # Select the variable
        grb = grib.select(name='2 metre temperature')[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
        tmtmp, lats, lons = grb.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)

        #-------------------------------------------------------------------------------------------------------------

        # Convert from K to °C
        tmtmp = tmtmp - 273.15

        # To smooth the contours
        import scipy.ndimage
        tmtmp = scipy.ndimage.zoom(tmtmp, 3)
        lats = scipy.ndimage.zoom(lats, 3)
        lons = scipy.ndimage.zoom(lons, 3)
```

Script 11: Averages, Maximums and Minimums - Simultaneous Multiple Plots

```
58        # Convert from K to °C
59        tmtmp = tmtmp - 273.15
60
61        # To smooth the contour
62        import scipy.ndimage
63        tmtmp = scipy.ndimage.zoom(tmtmp, 3)
64        lats = scipy.ndimage.zoom(lats, 3)
65        lons = scipy.ndimage.zoom(lons, 3)
66
67        # If it is the first cycle, create the arrays that will store the average values
68        if (hour == hour_ini):
69            tmtmp_sum = np.zeros((tmtmp.shape[0],tmtmp.shape[1]))
70            tmtmp_max = np.full((tmtmp.shape[0],tmtmp.shape[1]),-9999)
71            tmtmp_min = np.full((tmtmp.shape[0],tmtmp.shape[1]), 9999)
72
73        # Sum the values
74        tmtmp_sum = tmtmp_sum + tmtmp
75
76        # Keep the maximuns
77        tmtmp_max = np.maximum(tmtmp,tmtmp_max)
78
79        # Keep the minimuns
80        tmtmp_min = np.minimum(tmtmp,tmtmp_min)
81
82    # Calculate the average
83    tmtmp_avg = tmtmp_sum / ((hour_end - hour_ini) / hour_int)
84
85    print("\nAverage, Min and Max values stored!")
86
87    #---------------------------------------------------------------------------------------------------------------
88    # Create a custom color palette
89    colors = ["#d3d2d2", "#bcbcbc", "#969696", "#1464d2", "#1e6eeb", "#2882f0",
90    "#3c96f5", "#50a5f5", "#78b9fa", "#96d2fa", "#b4f0fa", "#1eb41e", "#37d23c",
91    "#50f050", "#78f573", "#96f58c", "#b4faaa", "#c8ffbe", "#ffe878", "#ffc03c",
92    "#ffa000", "#ff6000", "#ff3200", "#e11400", "#c00000", "#a50000", "#785046",
93    "#8c6359", "#b48b82", "#e1beb4"]
94    cmap = matplotlib.colors.ListedColormap(colors)
95    cmap.set_over('#fadad5')
96    cmap.set_under('#e5e5e5')
97    #---------------------------------------------------------------------------------------------------------------
98
99    # Choose the plot size (width x height, in inches)
100   fig, axs = plt.subplots(1,3, figsize=(14,5), sharex = False, sharey = False, subplot_kw=dict(projection=ccrs.PlateCarree())) # 1 row x 3 columns
101
102   #---------------------------------------------------------------------------------------------------------------
```

DATA READING AND MANIPULATION (CONT.)

LOOP (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

No final do ciclo, calcula a média

GENERAL PLOT CONFIGURATION

1 row x 3 columns plot

Cartopy projection

Script 11: Averages, Maximums and Minimums - Simultaneous Multiple Plots

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_territoriais/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(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)
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, tmtmp_min, cmap=cmap, levels=levels, extend='both')
127 img2 = axs[0].contour(lons, lats, tmtmp_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

Script 11: Averages, Maximums and Minimums - Simultaneous Multiple Plots

Explaining the new instructions:

PLOT 2 (MAXIMUM)

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

IMAGE GENERATION

Script 11: Averages, Maximums and Minimums - Simultaneous Multiple Plots

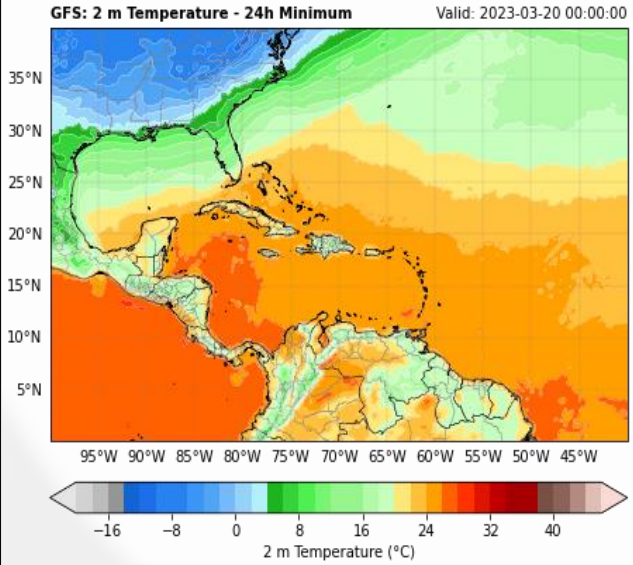
Explaining the new instructions:

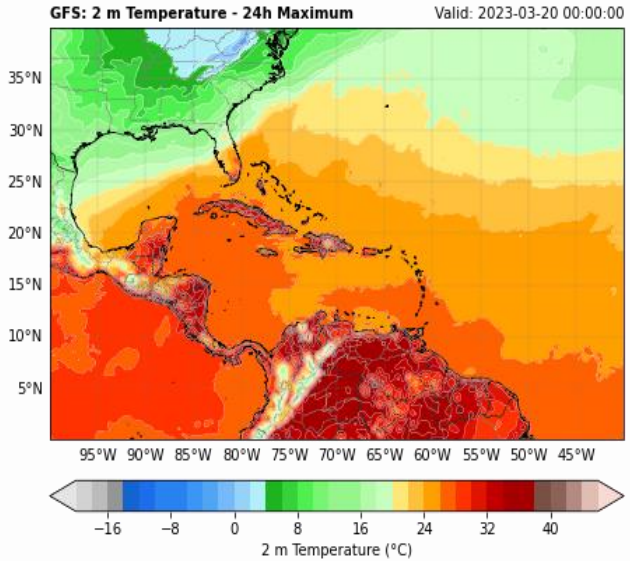
```
169 #-----------------------------------------------------------------------------------------------------------------
170
171 # Define the image extent
172 axs[2].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
173
174 # Add a shapefile
175 # https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
176 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
177 axs[2].add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray',facecolor='none', linewidth=0.3)
178
179 # Add coastlines, borders and gridlines
180 axs[2].coastlines(resolution='10m', color='black', linewidth=0.8)
181 axs[2].add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
182 gl = axs[2].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)
183 gl.top_labels = False
184 gl.right_labels = False
185
186 # Define de contour interval
187 data_min = -20
188 data_max = 48
189 interval = 2
190 levels = np.arange(data_min,data_max,interval)
191
192 # Plot the contours
193 img3 = axs[2].contourf(lons, lats, tmtmp_avg, cmap=cmap, levels=levels, extend='both')
194 img4 = axs[2].contour(lons, lats, tmtmp_avg, colors='white', linewidths=0.3, levels=levels)
195
196 # Add a colorbar
197 plt.colorbar(img3, label='2 m Temperature (°C)', orientation='horizontal', pad=0.02, fraction=0.05, ax=axs[2])
198
199 # Add a title
200 axs[2].set_title('GFS: 2 m Temperature - 24h Average' , fontweight='bold', fontsize=6, loc='left')
201 axs[2].set_title('Valid: ' + valid, fontsize=6, loc='right')
202
203 # Save the image
204 plt.savefig('image_11.png', bbox_inches='tight', pad_inches=0, dpi=100)
205
206 # Show the image
207 plt.show()
```

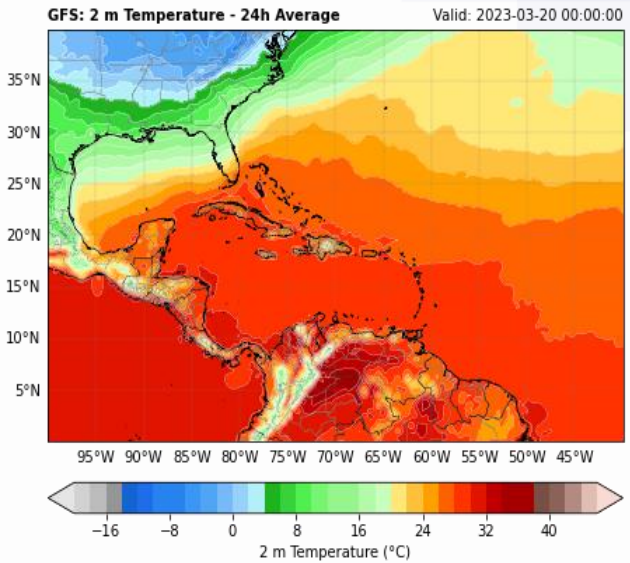
PLOT 3 (AVERAGE)

IMAGE GENERATION

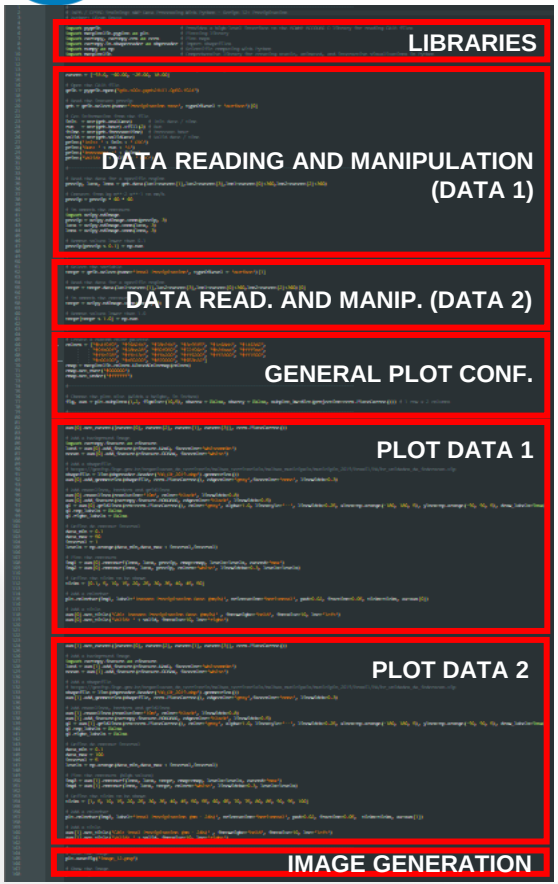
Script 11: Averages, Maximums and Minimums - Simultaneous Multiple Plots







Script 12: Precipitation and Accumulated Precipitation




```
#----------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing With Python - Script 12: Precipitation
# Author: Diego Souza
#----------------------------------------------------------------------------
import pygrib                                  # Provides a high-level interface to the ECMWF ECCODES
import matplotlib.pyplot as plt                # Plotting library
import cartopy, cartopy.crs as ccrs            # Plot maps
import cartopy.io.shapereader as shpreader     # Import shapefiles
import numpy as np                             # Scientific computing with Python
import matplotlib                              # Comprehensive library for creating static, animated,
#----------------------------------------------------------------------------

# 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.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 (PREC.)

"Not a Number"(does not show in the plot)

All pixels < 0.1 mm, convert to NaN

Script 12: Precipitation and Accumulated Precipitation

```
#------------------------------------------------------------------------------

# Select the variable
totpr = grib.select(name='Total Precipitation', typeOfLevel = 'surface')[1]

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

# Smooth the contours
totpr = scipy.ndimage.zoom(totpr, 3)

# Remove values lower than 1.0
totpr[totpr < 1.0] = np.nan

#------------------------------------------------------------------------------

# Create a custom color palette
colors = ["#b4f0f0", "#96d2fa", "#78b9fa", "#3c95f5", "#1e6deb", "#1450d2",
          "#0fa00f", "#28be28", "#50f050", "#72f06e", "#b3faaa", "#fff9aa",
          "#ffe978", "#ffc13c", "#ffa200", "#ff6200", "#ff3300", "#ff1500",
          "#c00100", "#a50200", "#870000", "#653b32"]
cmap = matplotlib.colors.ListedColormap(colors)
cmap.set_over('#000000')
cmap.set_under('#ffffff')

#------------------------------------------------------------------------------

# Choose the plot size (width x height, in inches)
fig, axs = plt.subplots(1,2, figsize=(10,5), sharex = False, sharey = False, subplot_kw=dict(projection=ccrs.PlateCarree())) # 1 row x 2 columns

#------------------------------------------------------------------------------

# Define the image extent
axs[0].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())

# Add a background image
import cartopy.feature as cfeature
land = axs[0].add_feature(cfeature.LAND, facecolor='whitesmoke')
ocean = axs[0].add_feature(cfeature.OCEAN, facecolor='white')
```

DATA READING AND MANIPULATION (ACCUMULATED PRECIP.)

All pixels < 0.1 mm, convert to NaN

"Not a Number" (does not show in the plot)

GENERAL PLOT CONFIGURATION

1 row x 2 columns plot

Cartopy Projection

Script 12: Precipitation and Accumulated Precipitation

```
#
# Define the image extent
axs[0].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())

# Add a background image
import cartopy.feature as cfeature
land = axs[0].add_feature(cfeature.LAND, facecolor='whitesmoke')
ocean = axs[0].add_feature(cfeature.OCEAN, facecolor='white')

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
axs[0].add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
axs[0].coastlines(resolution='10m', color='black', linewidth=0.8)
axs[0].add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = axs[0].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)
gl.top_labels = False
gl.right_labels = False

# Define de contour interval
data_min = 0.1
data_max = 50
interval = 1
levels = np.arange(data_min,data_max + interval,interval)

# Plot the contours
img1 = axs[0].contourf(lons, lats, precip, cmap=cmap, levels=levels, extend='max')
img2 = axs[0].contour(lons, lats, precip, colors='white', linewidths=0.3, levels=levels)

# Define the ticks to be shown
ticks = [0.1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

# Add a colorbar
plt.colorbar(img1, label='Instant Precipitation Rate (mm/h)', orientation='horizontal', pad=0.02, fraction=0.05, ticks=ticks, ax=axs[0])

# Add a title
axs[0].set_title('GFS: Instant Precipitation Rate (mm/h)' , fontweight='bold', fontsize=10, loc='left')
axs[0].set_title('Valid: ' + valid, fontsize=10, loc='right')

#--------------------------------------------------------------------------------------------------
```

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
125 # Define the image extent
126 axs[1].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())
127
128 # Add a background image
129 import cartopy.feature as cfeature
130 land = axs[1].add_feature(cfeature.LAND, facecolor='whitesmoke')
131 ocean = axs[1].add_feature(cfeature.OCEAN, facecolor='white')
132
133 # Add a shapefile
134 # https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
135 shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
136 axs[1].add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray',facecolor='none', linewidth=0.3)
137
138 # Add coastlines, borders and gridlines
139 axs[1].coastlines(resolution='10m', color='black', linewidth=0.8)
140 axs[1].add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
141 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)
142 gl.top_labels = False
143 gl.right_labels = False
144
145 # Define de contour interval
146 data_min = 0.1
147 data_max = 100
148 interval = 5
149 levels = np.arange(data_min,data_max + interval,interval)
150
151 # Plot the contours (high values)
152 img3 = axs[1].contourf(lons, lats, totpr, cmap=cmap, levels=levels, extend='max')
153 img4 = axs[1].contour(lons, lats, totpr, colors='white', linewidths=0.3, levels=levels)
154
155 # Define the ticks to be shown
156 ticks = [1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]
157
158 # Add a colorbar
159 plt.colorbar(img3, label='Total Precipitation (mm - 24h)', orientation='horizontal', pad=0.02, fraction=0.05,  ticks=ticks, ax=axs[1])
160
161 # Add a title
162 axs[1].set_title('GFS: Total Precipitation (mm - 24h)' , fontweight='bold', fontsize=10, loc='left')
163 axs[1].set_title('Valid: ' + valid, fontsize=10, loc='right')
164
165 #
166 # Save the image
167 plt.savefig('image_12.png')
168
169 # Show the image
170 plt.show()
```

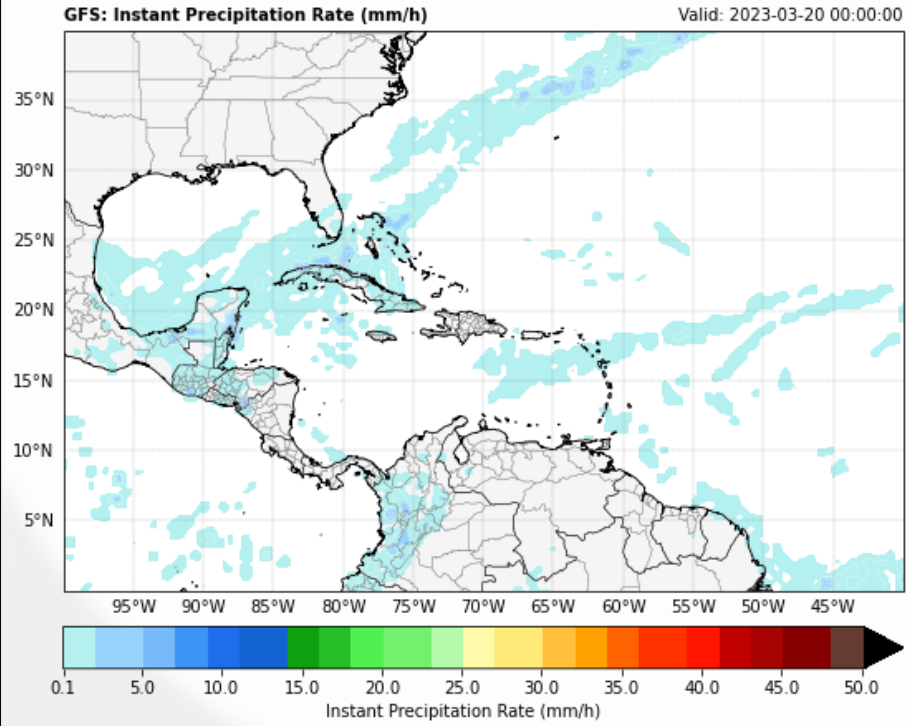
PLOT DATA 2 (ACC. PRECIP.)

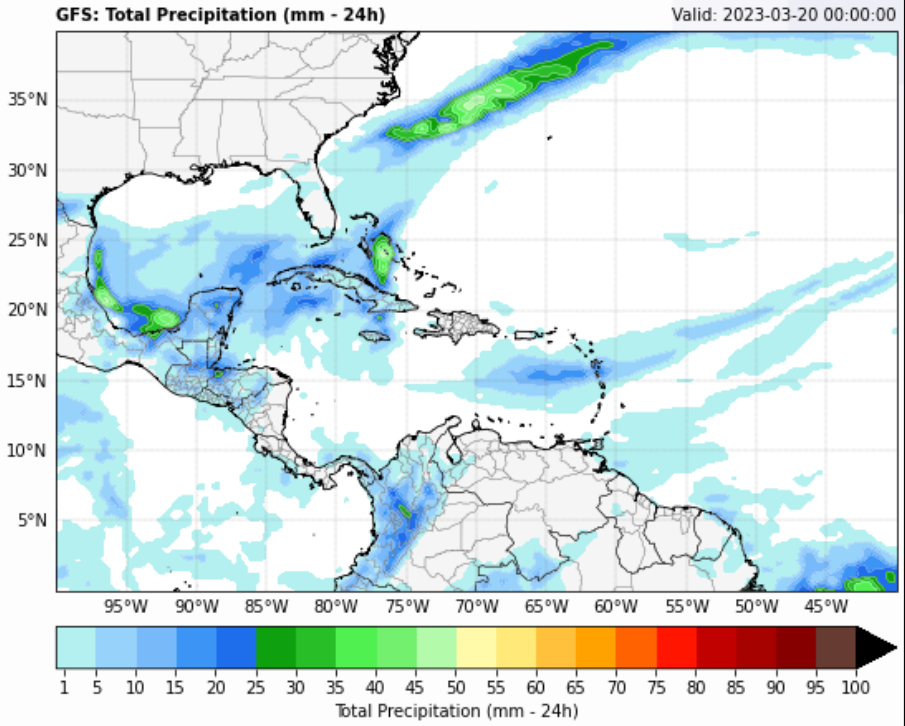
Add Surface and Ocean with Specific Colors

Values that will appear in the colorbar

IMAGE GENERATION

Script 12: Precipitation and Accumulated Precipitation





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: Downloading Data From NOMADS

```python
#-----------------------------------------------------------------------------------------------------------
# INPE / CPTEC - Training: NWP Data Processing With Python - NWP Download with Python (GFS)
# Author: Diego Souza
#-----------------------------------------------------------------------------------------------------------

# Required modules
from datetime import datetime          # Basic Dates and time types
import os                              # Miscellaneous operating system interfaces
import requests                        # HTTP library for Python
import time as t                       # Time access and conversion
#-----------------------------------------------------------------------------------------------------------

print('----------------------------------------')
print('GFS Download (NOMADS) - Script started.')
print('----------------------------------------')

# Start the time counter
start_time = t.time()

#-----------------------------------------------------------------------------------------------------------

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

# Desired date (last 10 days only!): Format - 'YYYYMMDD'
date = datetime.today().strftime('%Y%m%d')

# Desired extent
min_lon = '-93.00'
max_lon = '-25.00'
min_lat = '-60.00'
max_lat = '18.00'

# Desired resolution: '25' or '50' or '1'
resolution = '50'

# Desired run: '00' or '06' or '12' or '18'
hour_run = '00'

# Desired forecast hours
hour_ini = 0  # Init time
hour_end = 24 # End time
hour_int = 3  # Interval

#-----------------------------------------------------------------------------------------------------------

# https://nomads.ncep.noaa.gov/

def download_gfs(date, iii):

    # Create the URL's based on the resolution
    if (resolution == '25'):
        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).zfill(3)+'&all_lev=on&all_var=on&subregion=&leftlon='+min_lon+'&rightlon='+max_lon+'&toplat='+max_lat+'&bottomlat='+min_lat+'&dir=%2Fgfs.'+date+'%2F00%2Fatmos'
        file_name = 'gfs.t'+hour_run+'z.pgrb2.0p'+resolution+'.f'+str(hour).zfill(3)
    elif (resolution == '50'):
        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).zfill(3)+'&all_lev=on&all_var=on&subregion=&leftlon='+min_lon+'&rightlon='+max_lon+'&toplat='+max_lat+'&bottomlat='+min_lat+'&dir=%2Fgfs.'+date+'%2F00%2Fatmos'
        file_name = 'gfs.t'+hour_run+'z.pgrb2.0p'+resolution+'.f'+str(hour).zfill(3)
    elif (resolution == '1'):
        url = 'https://nomads.ncep.noaa.gov/cgi-bin/filter_gfs_'+resolution+'p00.pl?file=gfs.t'+hour_run+'z.pgrb2.'+resolution+'p00.f'+str(hour).zfill(3)+'&all_lev=on&all_var=on&subregion=&leftlon='+min_lon+'&rightlon='+max_lon+'&toplat='+max_lat+'&bottomlat='+min_lat+'&dir=%2Fgfs.'+date+'%2F00%2Fatmos'
        file_name = 'gfs.t'+hour_run+'z.pgrb2.'+resolution+'p00.f'+str(hour).zfill(3)

    # Print the file name
    print("File name: ", file_name)
    # Sends a GET request to the specified url
    myfile = requests.get(url)

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

#-----------------------------------------------------------------------------------------------------------

for hour in range(hour_ini, hour_end + 1, hour_int):
    print('\n---------------------')
    print('Downloading GFS File:')
    print('---------------------')
    print('Resolution: ' + resolution)
    print('Date: ' + date)
    print('Run: ' + hour_run)
    print('Forecast Hour: f' + str(hour).zfill(3))
    # Call the download function
    download_gfs(date,hour)

#-----------------------------------------------------------------------------------------------------------

# End the time counter
print('\nTotal Processing Time:', round((t.time() - start_time),2), 'seconds.')
```

LIBRARIES

DOWNLOAD CONFIGURATION

DOWNLOAD FUNCTION

DOWNLOAD FUNCTION CALL LOOP

Script: Downloading Data From NOMADS

```
#-------------------------------------------------------------------------------------------------------------
# INPE / CPTEC - Training: NWP Data Processing With Python - NWP Download with Python (GFS)
# Author: Diego Souza
#-------------------------------------------------------------------------------------------------------------

# Required modules
from datetime import datetime          # Basic Dates and time types
import os                              # Miscellaneous operating system interfaces
import requests                        # HTTP library for Python
import time as t                       # Time access and conversion

print('-----------------------------------------')
print('GFS Download (NOMADS) - Script started.')
print('-----------------------------------------')

# Start the time counter
start_time = t.time()

#-------------------------------------------------------------------------------------------------------------

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

# Desired date (last 10 days only!): Format - 'YYYYMMDD'
date = datetime.today().strftime('%Y%m%d')

# Desired extent
min_lon = '-93.00'
max_lon = '-25.00'
min_lat = '-60.00'
max_lat = '18.00'

# Desired resolution: '25' or '50' or '1'
resolution = '50'

# Desired run: '00' or '06' or '12' or '18'
hour_run = '00'

# Desired forecast hours
hour_ini = 0  # Init time
hour_end = 24 # End time
hour_int = 3  # Interval

#-------------------------------------------------------------------------------------------------------------
```

LIBRARIES

DOWNLOAD CONFIGURATION

Script: Downloading Data From NOMADS

```
# Link (select "grib filter" and check "Show the URL only for web programming" to verify the URL's):
# https://nomads.ncep.noaa.gov/

def download_gfs(date, iii):

    # Create the URL's based on the resolution
    if (resolution == '25'):
        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).
        file_name = 'gfs.t'+hour_run+'z.pgrb2.0p'+resolution+'.f'+str(hour).zfill(3)
    elif (resolution == '50'):
        url = 'https://nomads.ncep.noaa.gov/cgi-bin/filter_gfs_0p'+resolution+'.pl?file=gfs.t'+hour_run+'z.pgrb2full.0p'+resolution+'.f'+str(ho
        file_name = 'gfs.t'+hour_run+'z.pgrb2.0p'+resolution+'.f'+str(hour).zfill(3)
    elif (resolution == '1'):
        url = 'https://nomads.ncep.noaa.gov/cgi-bin/filter_gfs_'+resolution+'p00.pl?file=gfs.t'+hour_run+'z.pgrb2.'+resolution+'p00.f'+str(hou
        file_name = 'gfs.t'+hour_run+'z.pgrb2.'+resolution+'p00.f'+str(hour).zfill(3)

    # Print the file name
    print("File name: ", file_name)
    # Sends a GET request to the specified url
    myfile = requests.get(url)

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

#

# Download loop
for hour in range(hour_ini, hour_end + 1, hour_int):
    print('\n----------------------')
    print('Downloading GFS File:')
    print('----------------------')
    print('Resolution: ' + resolution)
    print('Date: ' + date)
    print('Run: ' + hour_run)
    print('Forecast Hour: f' + str(hour).zfill(3))
    # Call the download function
    download_gfs(date,hour)

#-----------------------------------------------------------------------------------------------------------------------

# End the time counter
print('\nTotal Processing Time:', round((t.time() - start_time),2), 'seconds.')
```

DOWNLOAD FUNCTION

DOWNLOAD FUNCTION CALL LOOP

Script 13: Reading Fields by Level - Streamlines

LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

Explaining the new instructions:

```
#----------------------------------------------------------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing With Python - Script 13: Choosing Fields by Levels - Streamlines
# Author: Diego Souza
#
import pygrib                           # Provides a high-level interface to the ECMWF ECCODES C library for reading GRIB files
import matplotlib.pyplot as plt         # Plotting library
import cartopy, cartopy.crs as ccrs     # Plot maps
import cartopy.io.shapereader as shpreader # Import shapefiles
import numpy as np                      # Scientific computing with Python
import matplotlib                       # Comprehensive library for creating static, animated, and interac
#----------------------------------------------------------------------------------------------------------------------------------

# 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 = 250)[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 = 250)[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))
```

Selecting variables specifying the level

Calculating wind speed

Script 13: Reading Fields by Level - Streamlines

variables - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

Remembering the "variables.txt" file

```
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
```

```
21    # Select the variable
22    ucomp = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 250)[0]
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

```
# Choose the plot size (width x height, in inches)
plt.figure(figsize=(10,6))

# Use the Cilindrical Equidistant projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree(central_longitude=extent[0] - extent[2]))

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]
ax.set_extent([extent[0], extent[2], extent[1], extent[3]], crs=ccrs.PlateCarree())

# Define de contour interval
data_min = 0
data_max = 60
interval = 5
levels = np.arange(data_min,data_max,interval)

# Create a custom color palette
colors = ["#e7f2f4", "#ceeaee", "#b6e2e8", "#abdcff", "#a4d685", "#9cd04e",
          "#abcf2a", "#c9d21b", "#e8d50c", "#ffd100", "#ffba00", "#ffa200"]
cmap = matplotlib.colors.ListedColormap(colors)
cmap.set_over('#ff8c00')
cmap.set_under('#fffafa')

# Plot the contours
img1 = ax.contourf(lons, lats, ws, cmap=cmap, levels=levels, extend='both', transform=ccrs.PlateCarree())
img2 = ax.contour(lons, lats, ws, colors='white', linewidths=0.3, levels=levels, transform=ccrs.PlateCarree())
ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10',fmt = '%1.0f', colors= 'black')

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

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

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.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)
gl.top_labels = False
gl.right_labels = False

# Add a colorbar
plt.colorbar(img1, label='Isotachs (kt)', orientation='horizontal', pad=0.05, fraction=0.05)

# Add a title
plt.title('GFS: Streamlines and Isotachs (250 hPa)' , fontweight='bold', fontsize=10, loc='left')
plt.title('Valid: ' + valid, fontsize=10, loc='right')

# Save the image
plt.savefig('image_13.png', bbox_inches='tight', pad_inches=0, dpi=100)

# Show the image
```

PLOT CONFIGURATION

Plot the windspeed

Lats and Lons

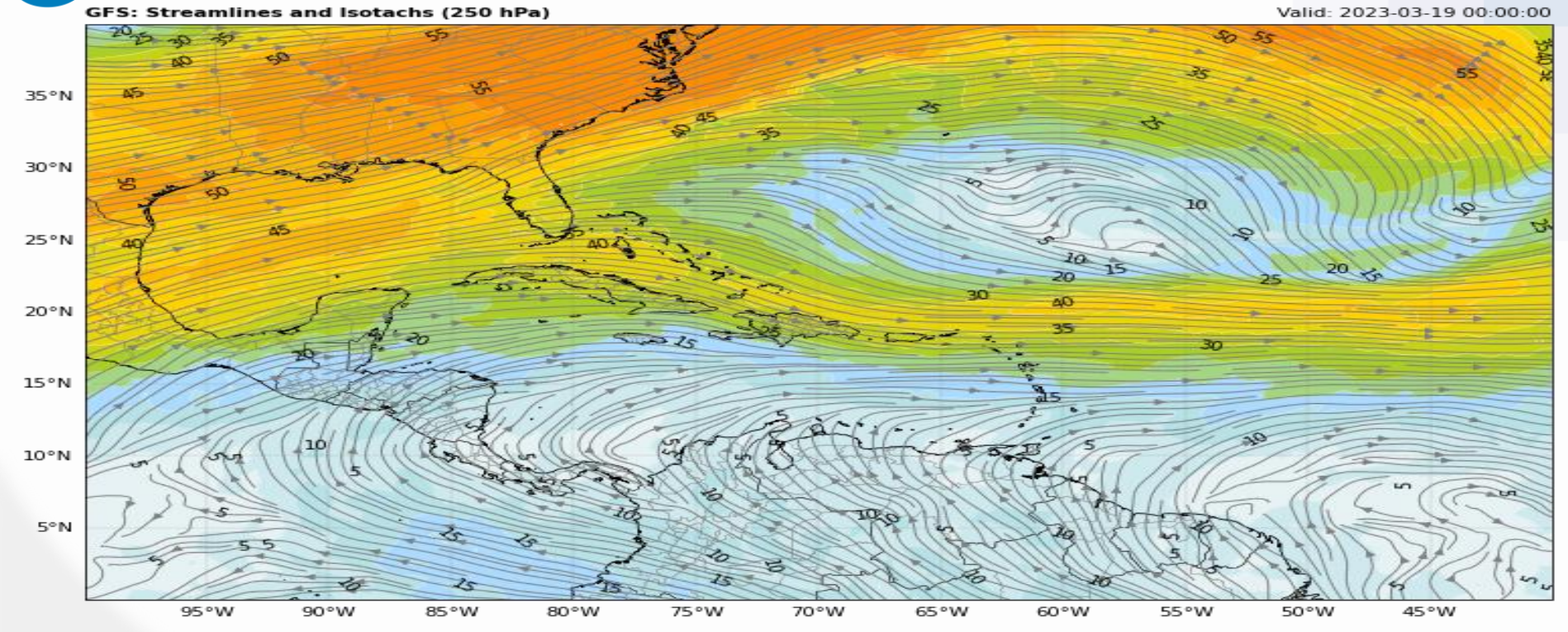
Density

Plot the streamlines

U and V components

IMAGE GENERATION

Script 13: Reading Fields by Level - Streamlines

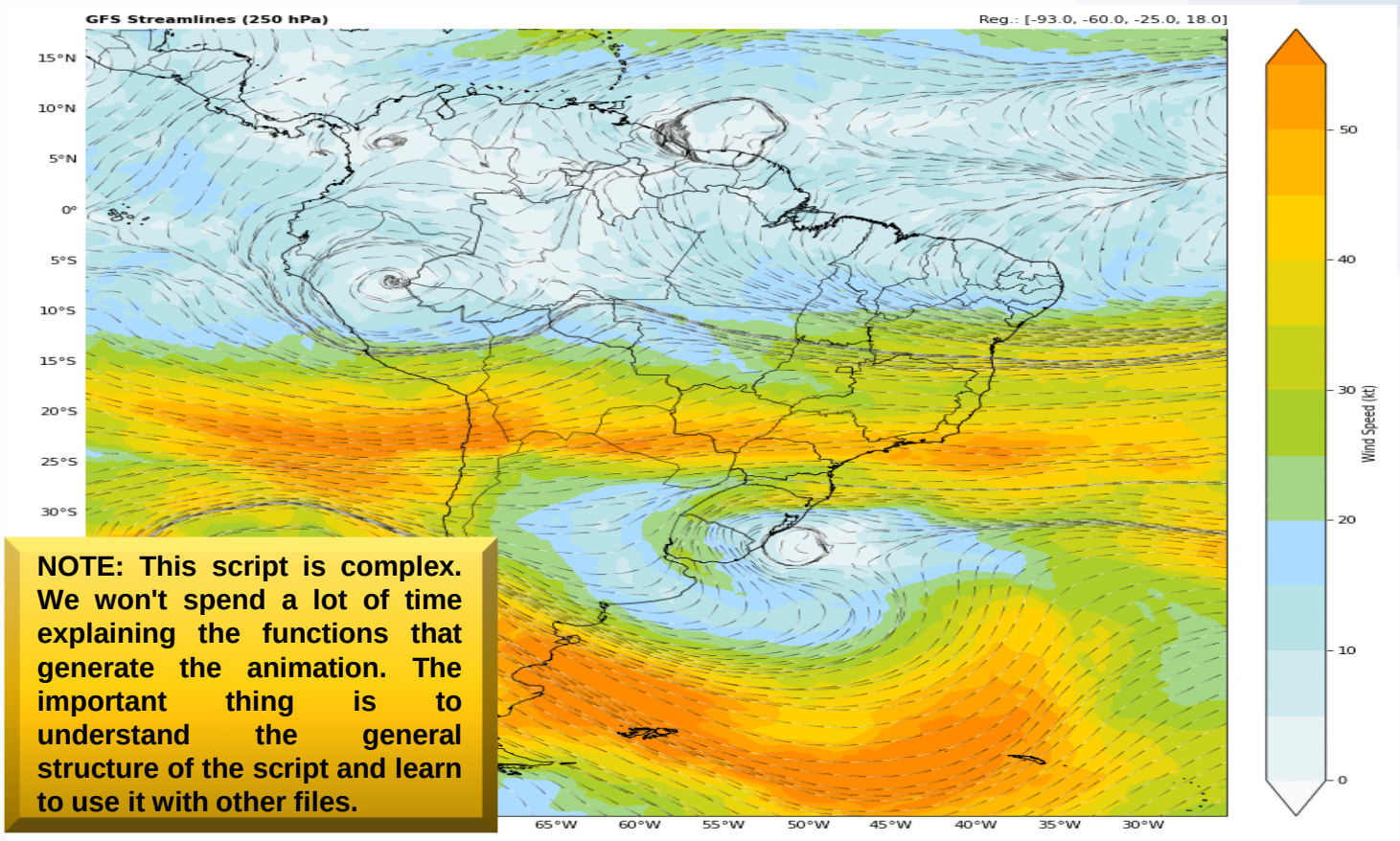



```
78
79    # Plot the streamlines
80    img3 = ax.streamplot(lons, lats, ucomp, vcomp, density=[4, 4], linewidth=1, color='gray', transform=ccrs.PlateCarree())
81
```

Some Quick Modifications

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

Script 14: Animating Streamlines



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

DATA READING AND MANIPULATION

Explaining the new instructions:

```
180
181
182  # Select the extent [min. lon, min. lat, max. lon, max. lat]
183  extent = [-93.0, -60.00, -25.00, 18.00]
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(latl=extent[1],lat2=extent[3],lonl=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', lev 200)[0] )[0]
210
211  # Read the data for a specific region
212  vcomp = vcomp.data(latl=extent[1],lat2=extent[3],lonl=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
227
228  # Choose the plot size (width x height, in inches)
229  fig = plt.figure(figsize=(8,8))
```

Variables needed for functions

Script 14: Animating Streamlines

PLOT CONFIGURATION

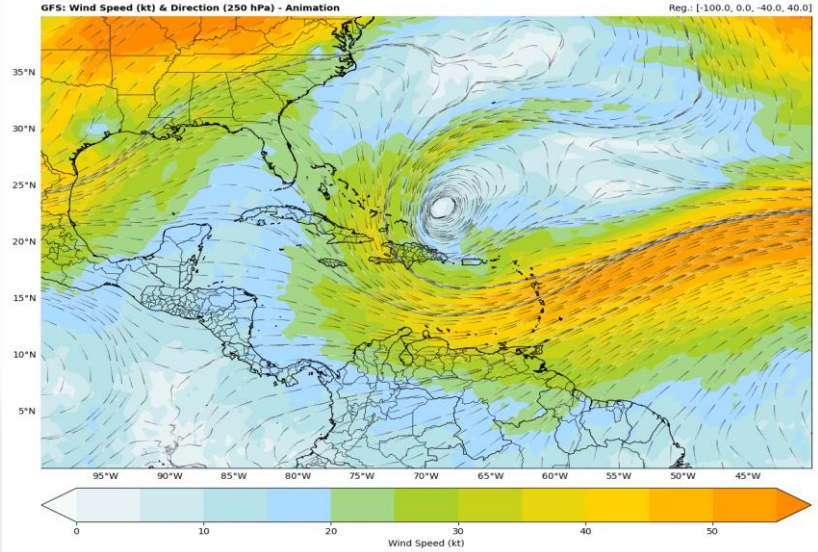
```
#

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

# Use the Cilindrical Equidistant projection in cartopy
ax = fig.add_subplot(1, 1, 1,projection=ccrs.PlateCarree())
ax.set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]

# Define de contour interval
data_min = 0
data_max = 60
interval = 5
levels = np.arange(data_min,data_max,interval)

# Create a custom color palette
colors = ["#e7f2f4", "#ceeaee", "#b6e2e8", "#abdcff", "#a4d685", "#9cd04e", "#abcf2a", "#c9d21b", "#e8d50c", "#ffd100", "#ffba00", "#ffa200"]
cmap = matplotlib.colors.ListedColormap(colors)
cmap.set_over('#ff8c00')
cmap.set_under('#fffafa')

# Plot the contours
img1 = ax.contourf(lons, lats, ws, cmap=cmap, levels=levels, extend='both', alpha=1.0, zorder = 10)

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='black',facecolor='none', linewidth=0.3, zorder=14)

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8, zorder=15)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5, zorder=16)
gl = ax.gridlines(crs=ccrs.PlateCarree(), color='white', alpha=1.0, linestyle='--', linewidth=0.50, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(-90, 90, 5), draw_labels=True)
gl.top_labels = False
gl.right_labels = False

# Add a colorbar
plt.colorbar(img1, label='Wind Speed (kt)', orientation='vertical', pad=0.03, fraction=0.05)

# Add a title
plt.title('GFS: Wind Speed (kt) & Direction (250 hPa) - Animation', fontweight='bold', fontsize=8, loc='left')
plt.title('Reg.: ' + str(extent) , fontsize=8, loc='right')

#
```

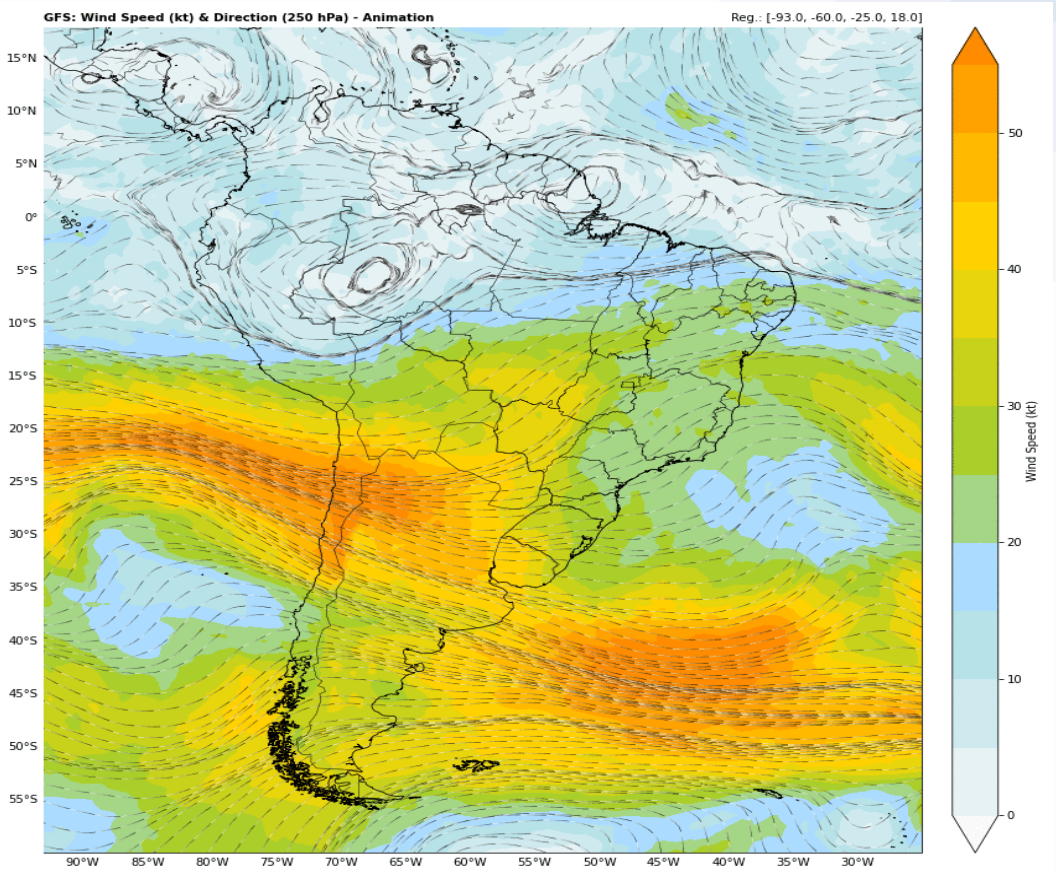
Script 14: Animating Streamlines

```
# Create the stramlines animation
lengths = []
colors = []
lines = []
s = Streamlines(X, Y, U, V)

for streamline in s.streamlines:

    x, y = streamline

    # Points
    points = np.array([x, y]).T.reshape(-1, 1, 2)

    # Segments
    segments = np.concatenate([points[:-1], points[1:]], axis=1)
    n = len(segments)

    # Lenghts
    D = np.sqrt(((points[1:] - points[:-1])**2).sum(axis=-1))
    L = D.cumsum().reshape(n,1) + np.random.uniform(0,1)

    # Colors
    C = np.zeros((n,3))
    C[:] = ((L*1.5) % 1)

    # Lines
    line = LineCollection(segments, linewidth=0.5, zorder=13)

    # Append lenghts, colors and lines
    lengths.append(L)
    colors.append(C)
    lines.append(line)
    ax.add_collection(line)

# Plot extent
ax.set_xlim(extent[0],extent[2]), ax.set_xticks([])
ax.set_ylim(extent[1],extent[3]), ax.set_yticks([])
plt.tight_layout()

#-------------------------------------------------------------------------------------------------------------

# Number of frames
n = 30

# Create the animation
animation = FuncAnimation(fig, update, frames=n, interval=20)

# Update the progress bar
#pbar = tqdm.tqdm(total=n)

# Save animation as GIF
print("\nGenerating the GIF...")
animation.save('streamlines_animation.gif', writer='pillow', fps=30)

# Close the progress bar
#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

LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

```python
#----------------------------------------------------------------------------------------------------
# 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 shpreader # Import shapefiles
import numpy as np                  # Scientific computing with Python
import matplotlib                   # Comprehensive library for creating static, animated, and intera
#----------------------------------------------------------------------------------------------------

# 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))
```

Selecting variables specifying the level

Script 15: Wind Vectors

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

# Use the Cilindrical Equidistant projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree())

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]
ax.set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.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)
gl.top_labels = False
gl.right_labels = False

# Define de contour interval
data_min = 0
data_max = 60
interval = 5
levels = np.arange(data_min,data_max,interval)

# Create a custom color palette
colors = ["#e7f2f4", "#ceeaee", "#b6e2e8", "#abdcff", "#a4d685", "#9cd04e", "#abcf2a", "#c9d21b", "#e8d50c", "#ffd100", "#ffba00", "#ffa200"]
cmap = matplotlib.colors.ListedColormap(colors)
cmap.set_over('#ff8c00')
cmap.set_under('#fffafa')

# Plot the contours
img1 = ax.contourf(lons, lats, ws, cmap=cmap, levels=levels, extend='both')
img2 = ax.contour(lons, lats, ws, colors='white', linewidths=0.3, levels=levels)
ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10',fmt = '%1.0f', colors= 'black')

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

# Add a colorbar
plt.colorbar(img1, label='Wind Speed (kt)', orientation='vertical', pad=0.05, fraction=0.05)

# Add a title
plt.title('GFS: Wind Speed (kt) & Direction (925 hPa)' , fontweight='bold', fontsize=10, loc='left')
plt.title('Valid: ' + valid, fontsize=10, loc='right')

# Save the image
plt.savefig('image_15.png', bbox_inches='tight', pad_inches=0, dpi=100)

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

PLOT CONFIGURATION

IMAGE GENERATION

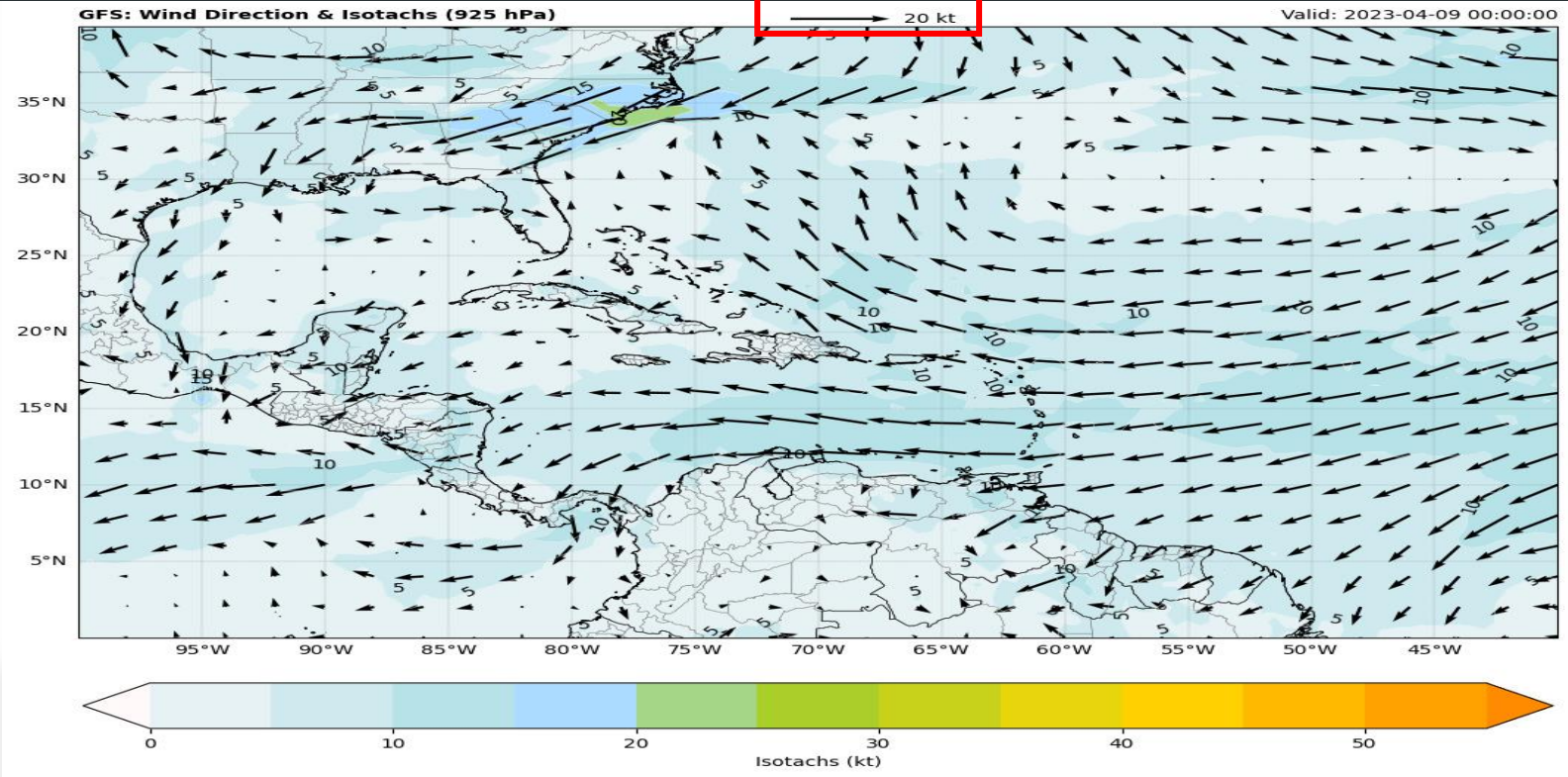
Latitudes and Longitudes

U and V components

Plot the wind vectors and legends

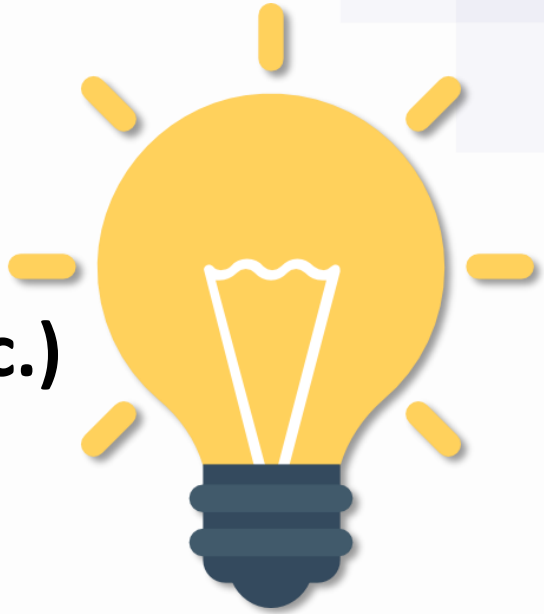
Script 15: Wind Vectors

```
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: Barbs

LIBRARIES

DATA READING AND MANIPULATION

PLOT CONFIGURATION

IMAGE GENERATION

```
#-----------------------------------------------------------------------------
# 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 f
import matplotlib.pyplot as plt        # Plotting library
import cartopy, cartopy.crs as ccrs    # Plot maps
import cartopy.io.shapereader as shpreader # Import shapefiles
import numpy as np                     # Scientific computing with Python
import matplotlib                      # Comprehensive library for creating static, animated, and interac

#-----------------------------------------------------------------------------
# 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 = 850)[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 = 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))
```

Selecting variables specifying the level

Single field, no need to specify level

Script 16: Barbs

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

# Use the Cilindrical Equidistant projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree())

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]
ax.set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.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)
gl.top_labels = False
gl.right_labels = False

# Define de contour interval
data_min = 990
data_max = 1050
interval = 2
levels = np.arange(data_min,data_max,interval)

# Create a custom color palette
colors = ["#310d00", "#631b00", "#9428000", "#c53500", "#fd6123", "#fb824e", "#faa679", "#f8c[illegible]", "#[illegible]", "#fafaee", "#fcfcf5", "#fefefc", "#e8eef5", "#d9e2ef", "#c9d7e9", "#bacce2
cmap = matplotlib.colors.ListedColormap(colors)
cmap.set_over('#000000')
cmap.set_under('#000000')

# Plot the image
img1 = ax.contourf(lons, lats, prmls, cmap=cmap, levels=levels, extend='both')
img2 = ax.contour(lons, lats, prmls, colors='black', linewidths=0.1, levels=levels)
ax.clabel(img2, inline=1, inline_spacing=0, fontsize=10,fmt = '%1.0f', colors= 'black')

# Create a flag to determine which barbs are flipped
flip_flag = np.zeros((ucomp.shape[0],ucomp.shape[1]))

# All flags below the equator will
flip_flag[lats < 0] = 1

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

# Add a colorbar
plt.colorbar(img1, label='PSML (hPa)', orientation='horizontal', pad=0.05, fraction=0.05)

# Add a title
plt.title('GFS: PMSL + Winds (850 hPa)' , fontweight='bold', fontsize=10, loc='left')
plt.title('Valid: ' + valid, fontsize=10, loc='right')
#-------------------------------------------------------------------------------
# Save the image
plt.savefig('image_16.png', bbox_inches='tight', pad_inches=0, dpi=100)

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

PLOT CONFIGURATION

IMAGE GENERATION

Plot the PSML

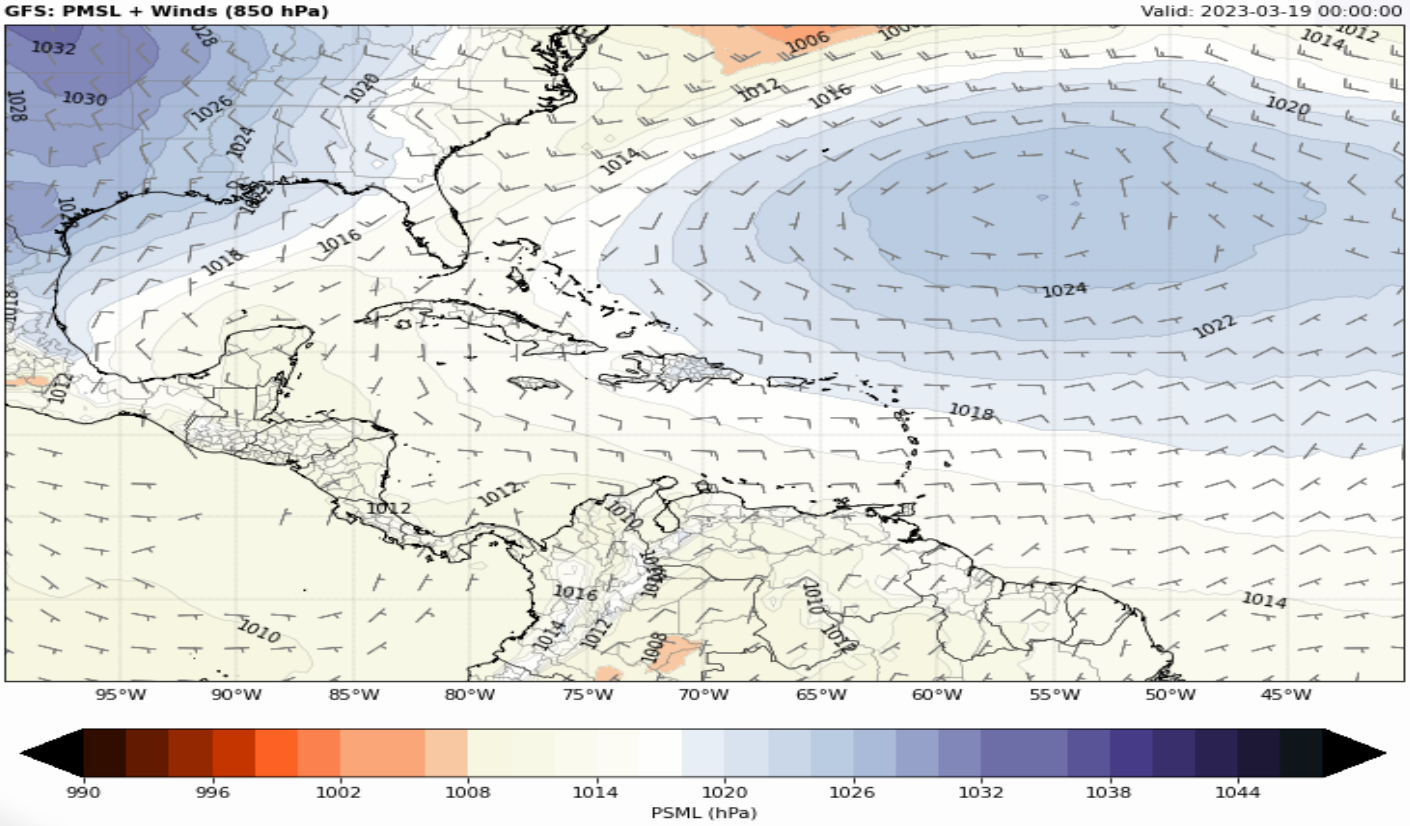
Creates the array that will indicate which barbs should be inverted

Plot the barbs

Pixels below 0° set to 1

flip_barb parameter

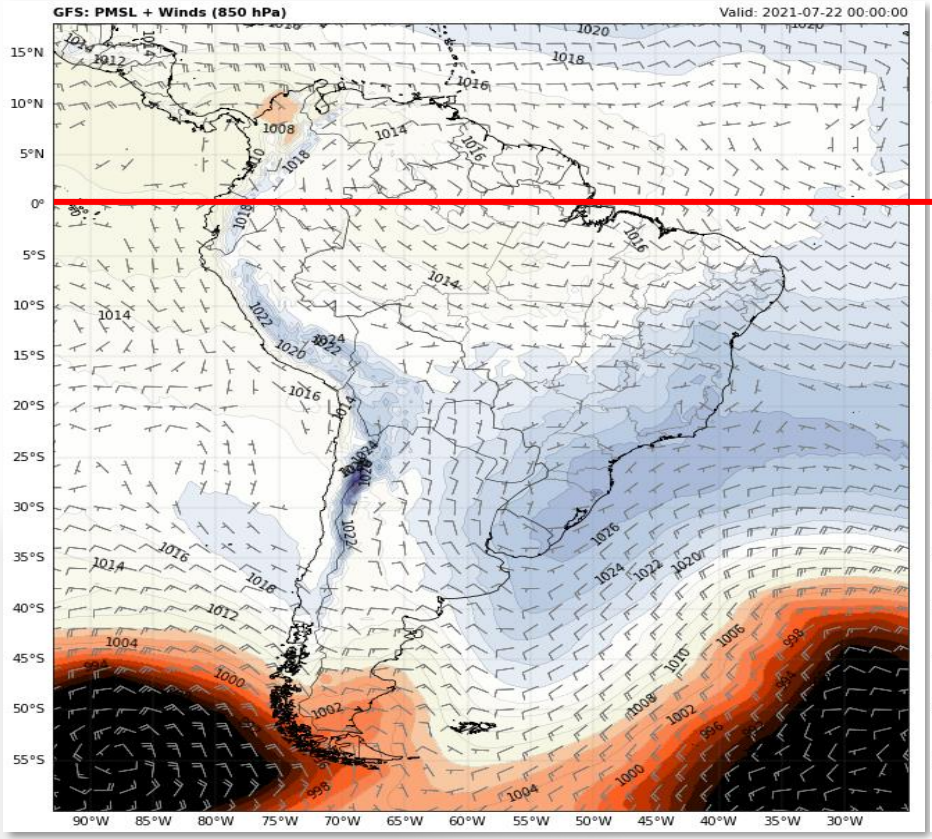
Script 16: Barbs



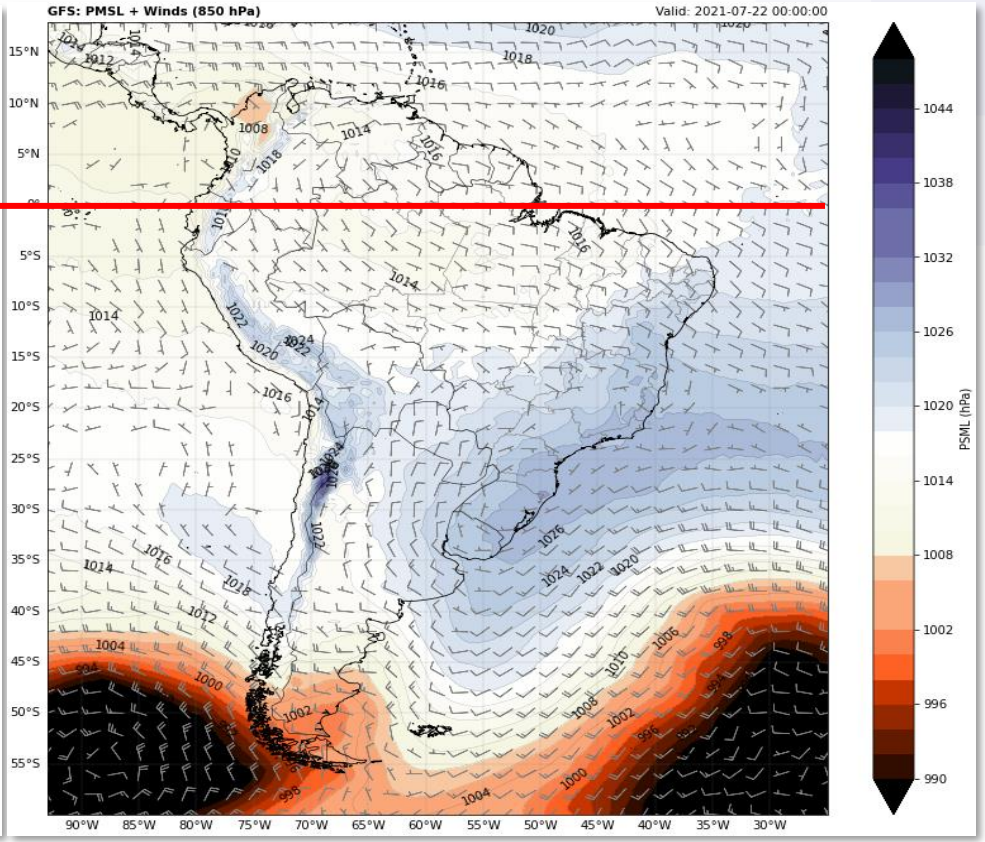

```
107  # Plot the barbs
108  img3 = ax.barbs(lons[::4,::4], lats[::4,::4], ucomp[::4,::4], vcomp[::4,::4], length = 5.0, sizes = dict(emptybarb=0.0, spacing=0.2, height=0.5), linewidth=0.8, pivot='middle', barbcolor='gray',
109  flip_barb = flip_flag[::4,::4])
```

Script 16: Barbs

With the flip_barb parameter:



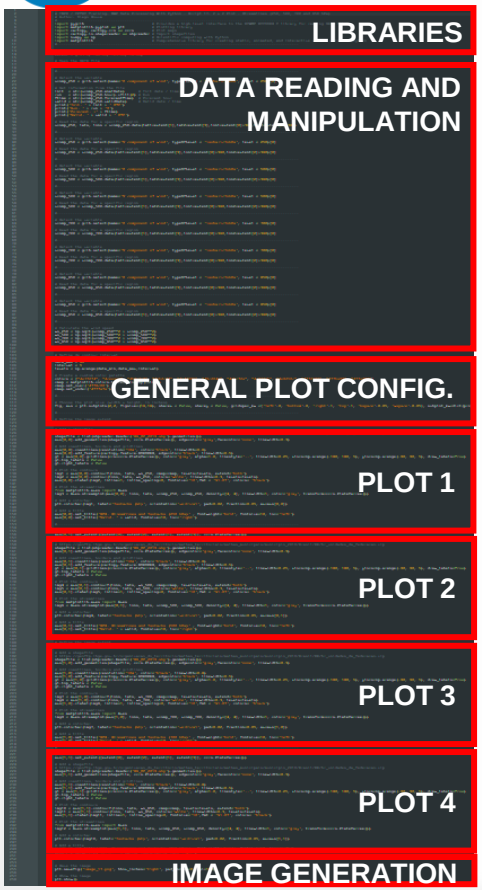
Without the flip_barb parameter:



Some Quick Modifications

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

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



LIBRARIES

```python
#---------------------------------------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP Data Processing With Python - Script 17: 2 x 2 Plot - Streamlines (250, 500, 70
# 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 shpreader # Import shapefiles
import numpy as np                           # Scientific computing with Python
import matplotlib                            # Comprehensive library for creating static, animated, and inter
#
```

DATA READING AND MANIPULATION (PART I)

```python
# 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_250 = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 250)[0]

# Get information from the file
init   = str(ucomp_250.analDate)       # Init date / time
run    = str(ucomp_250.hour).zfill(2) # Run
ftime  = str(ucomp_250.forecastTime)   # Forecast hour
valid  = str(ucomp_250.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_250, lats, lons = ucomp_250.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)

#---------------------------------------------------------------------------------------------------------------

# Select the variable
vcomp_250 = grib.select(name='V component of wind', typeOfLevel = 'isobaricInhPa', level = 250)[0]

# Read the data for a specific region
vcomp_250 = vcomp_250.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]

#---------------------------------------------------------------------------------------------------------------

# Select the variable
ucomp_500 = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 500)[0]

# Read the data for a specific region
ucomp_500 = ucomp_500.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]

#---------------------------------------------------------------------------------------------------------------
```

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

```
#-------------------------------------------------------------------------------

# Select the variable
ucomp_500 = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 500)[0]

# Read the data for a specific region
ucomp_500 = ucomp_500.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]

#-------------------------------------------------------------------------------

# Select the variable
vcomp_500 = grib.select(name='V component of wind', typeOfLevel = 'isobaricInhPa', level = 500)[0]

# Read the data for a specific region
vcomp_500 = vcomp_500.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]

#-------------------------------------------------------------------------------

# Select the variable
ucomp_700 = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 700)[0]

# Read the data for a specific region
ucomp_700 = ucomp_700.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]

#-------------------------------------------------------------------------------

# Select the variable
vcomp_700 = grib.select(name='V component of wind', typeOfLevel = 'isobaricInhPa', level = 700)[0]

# Read the data for a specific region
vcomp_700 = vcomp_700.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]

#-------------------------------------------------------------------------------

# Select the variable
ucomp_850 = grib.select(name='U component of wind', typeOfLevel = 'isobaricInhPa', level = 850)[0]

# Read the data for a specific region
ucomp_850 = ucomp_850.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]

#-------------------------------------------------------------------------------

# Select the variable
vcomp_850 = grib.select(name='V component of wind', typeOfLevel = 'isobaricInhPa', level = 850)[0]

# Read the data for a specific region
vcomp_850 = vcomp_850.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]

#-------------------------------------------------------------------------------

# Calculate the wind speed
ws_250 = np.sqrt(ucomp_250**2 + vcomp_250**2)
ws_500 = np.sqrt(ucomp_500**2 + vcomp_500**2)
ws_700 = np.sqrt(ucomp_700**2 + vcomp_700**2)
ws_850 = np.sqrt(ucomp_850**2 + vcomp_850**2)
```

Wind speed calculation

DATA READING AND MANIPULATION (CONT.)

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

```
#-------------------------------------------------------------------------------------------------------------------

# Define de contour interval
data_min = 0
data_max = 60
interval = 5
levels = np.arange(data_min,data_max,interval)

# Create a custom color palette
colors = ["#e7f2f4", "#ceeaee", "#b6e2e8", "#abdcff", "#a4d685", "#9cd04e", "#abcf2a", "#c9d21b", "#e8d50c", "#ffd100", "#ffba00", "#ffa200"]
cmap = matplotlib.colors.ListedColormap(colors)
cmap.set_over('#ff8c00')
cmap.set_under('#fffafa')

#-------------------------------------------------------------------------------------------------------------------

# Choose the plot size (width x height, in inches)
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(
projection=ccrs.PlateCarree())) # 2 row x 2 columns

#-------------------------------------------------------------------------------------------------------------------

# Define the image extent
axs[0,0].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
axs[0,0].add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
axs[0,0].coastlines(resolution='10m', color='black', linewidth=0.8)
axs[0,0].add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = axs[0,0].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)
gl.top_labels = False
gl.right_labels = False

# Plot the contours
img1 = axs[0,0].contourf(lons, lats, ws_250, cmap=cmap, levels=levels, extend='both')
img2 = axs[0,0].contour(lons, lats, ws_250, colors='white', linewidths=0.3, levels=levels)
axs[0,0].clabel(img2, inline=1, inline_spacing=0, fontsize='10',fmt = '%1.0f', colors= 'black')

# Plot the streamlines
from matplotlib.axes import Axes
img3 = Axes.streamplot(axs[0,0], lons, lats, ucomp_250, vcomp_250, density=[4, 4], linewidth=1, color='gray', transform=ccrs.PlateCarree())

# Add a colorbar
plt.colorbar(img1, label='Isotachs (kt)', orientation='vertical', pad=0.02, fraction=0.05, ax=axs[0,0])

# Add title
axs[0,0].set_title('GFS: Streamlines and Isotachs (250 hPa)' , fontweight='bold', fontsize=10, loc='left')
axs[0,0].set_title('Valid: ' + valid, fontsize=10, loc='right')
```

GENERAL PLOT CONFIGURATION

2 row x 2 column plot

Spacing

PLOT 1 (250 hPa)

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

PLOT 2 (500 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_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/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(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)
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, ucomp_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 #---------------------------------------------------------------------------------------------------------
```

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(shpreader.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(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)
200 gl.top_labels = False
201 gl.right_labels = False
202
203 # Plot the countours
204 img7 = axs[1,0].contourf(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)

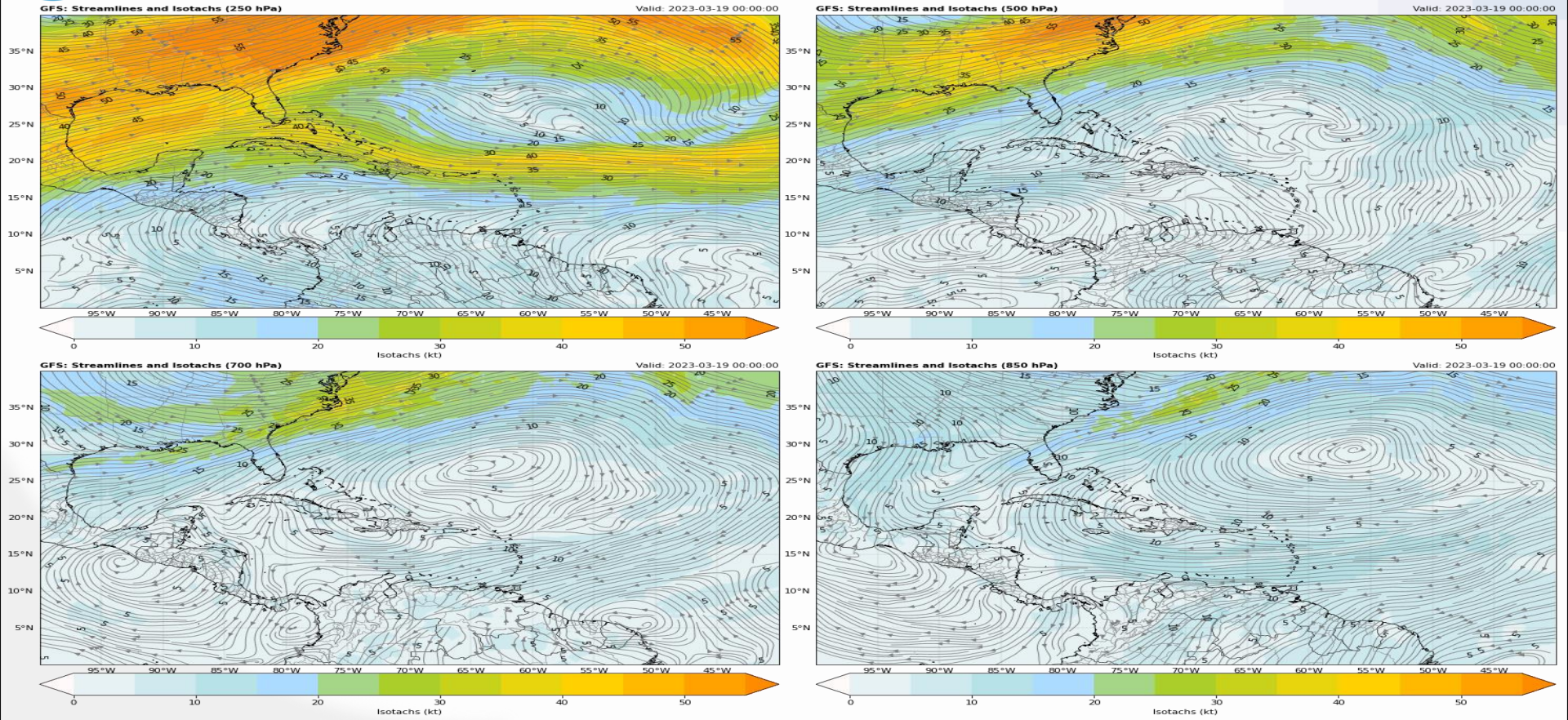
Script 17: 2 x 2 Plot - Streamlines at 250, 500, 700 and 850 hPa

PLOT 4 (850 hPa)

```
#

# Define the image extent
axs[1,1].set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
axs[1,1].add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
axs[1,1].coastlines(resolution='10m', color='black', linewidth=0.8)
axs[1,1].add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = axs[1,1].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)
gl.top_labels = False
gl.right_labels = False

# Plot the contours
img10 = axs[1,1].contourf(lons, lats, ws_850, cmap=cmap, levels=levels, extend='both')
img11 = axs[1,1].contour(lons, lats, ws_850, colors='white', linewidths=0.3, levels=levels)
axs[1,1].clabel(img11, inline=1, inline_spacing=0, fontsize='10',fmt = '%1.0f', colors= 'black')

# Plot the streamlines
from matplotlib.axes import Axes
img12 = Axes.streamplot(axs[1,1], lons, lats, ucomp_850, vcomp_850, density=[4, 4], linewidth=1, color='gray', transform=ccrs.PlateCarree())

# Add a colorbar
plt.colorbar(img10, label='Isotachs (kt)', orientation='vertical', pad=0.02, fraction=0.05, ax=axs[1,1])

# Add a title
axs[1,1].set_title('GFS: Streamlines and Isotachs (850 hPa)' , fontweight='bold', fontsize=10, loc='left')
axs[1,1].set_title('Valid: ' + valid, fontsize=10, loc='right')

#---------------------------------------------------------------------------------------

# Save the image
plt.savefig('image_17.png', bbox_inches='tight', pad_inches=0, dpi=100)

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

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) Proxys

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

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

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

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

where $\propto = -10$ [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

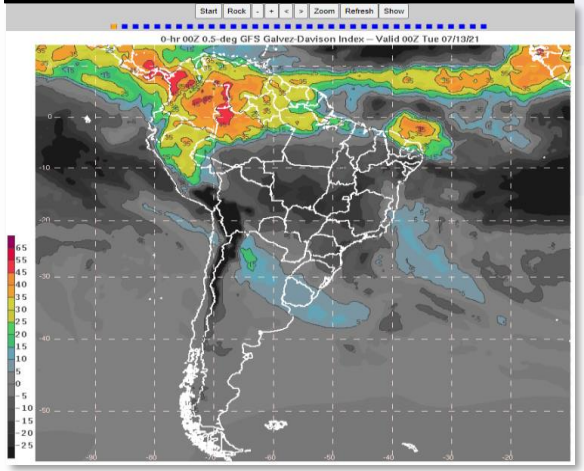
$$GDI = ECI + MWI + II \quad (1.17)$$

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

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

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

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

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

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

Surface Pressure

(2) Theta-E (EPT) Proxys

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

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

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

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

where $\propto = -10$ [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)

$$S = \sigma \times (T_{500} - T_{700}) \quad (1.14)$$

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.

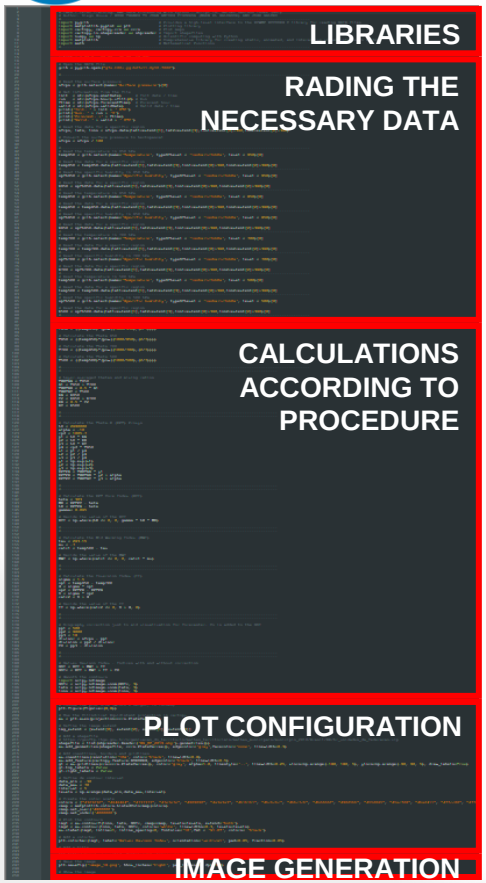
$$D = \sigma \times (EPTP_B - EPTP_A) \quad (1.15)$$

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

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

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

Script 18: Galvez Davison Index (GDI)



Explaining the new instructions:

```
#---------------------------------------------------------------------------------------------------------------------
# INPE / CPTEC Training: NWP D                                                    Galvez Davison Index (GDI)
# Author: Diego Souza / HUGE THANKS TO JUAN AMIDES FIGUEROA (MARN EL SALVADOR) AND JOSE GALVEZ
#
import pygrib                           # Provides a high-level interface to the ECWMF ECCODES C library f
import matplotlib.pyplot as plt         # Plotting library
import cartopy, cartopy.crs as ccrs     # Plot maps
import cartopy.io.shapereader as shpreader # Import shapefiles
import numpy as np                      # Scientific computing with Python
import matplotlib                       # Comprehensive library for creating static, animated, and interac
import math                             # Methematical Functions
#---------------------------------------------------------------------------------------------------------------------

# 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")

#---------------------------------------------------------------------------------------------------------------------

# Read the surface pressure
sfcps = grib.select(name='Surface pressure')[0]

# Get information from the file
init  = str(sfcps.analDate)       # Init date / time
run   = str(sfcps.hour).zfill(2) # Run
ftime = str(sfcps.forecastTime)   # Forecast hour
valid = str(sfcps.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
sfcps, lats, lons = sfcps.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)

# Convert the surface pr
sfcps = sfcps / 100
#---------------------------------------------------------------------------------------------------------------------
```

LIBRARIES

READING THE NECESSARY DATA (PART I)

Convert Pressure to Hectopascal

Script 18: Galvez Davison Index (GDI)

LIBRARIES

RADING THE NECESSARY DATA

CALCULATIONS ACCORDING TO PROCEDURE

PLOT CONFIGURATION

IMAGE GENERATION

READING THE NECESSARY DATA (CONT.)

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

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)

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

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

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

where $\propto = -10$ [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.

```
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)

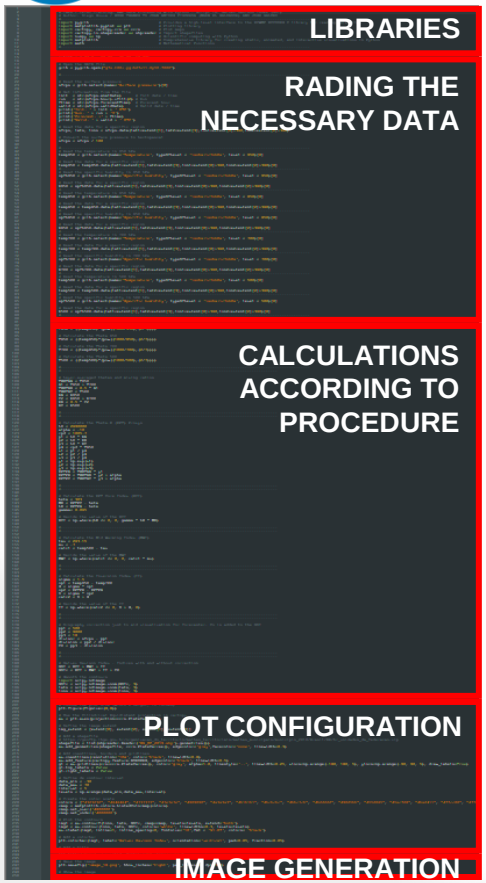
$$EPTP_A = \theta_A e^{\left(\frac{L_o r_A}{c_{pd} T_{850}}\right)} \quad (1.7)$$

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

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

where $\propto = -10$ [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.

Script 18: Galvez Davison Index (GDI)




```python
# Calculate the EPT Core Index (ECI)
beta = 303
ME = EPTPC - beta
LE = EPTPA - beta
gamma= 0.065

# Decide the value of the ECI
ECI = np.where(LE <= 0, 0, gamma * LE * ME)

#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
# Calculate the Mid Warming Index (MWI)
tau = 263.15
mu = -7
calc1 = temp500 - tau

# Decide the value of the MWI
MWI = np.where(calc1 <= 0, 0, calc1 * mu)

#-----------------------------------------------------------------------
#-----------------------------------------------------------------------

# Calculate the Inversion Index (II)
sigma = 1.5
op1 = temp950 - temp700
S = sigma * op1
op2 = EPTPB - EPTPA
D = sigma * op2
calc2 = D + S

# Decide the value of the II
II = np.where(calc2 <= 0, D + S, 0)

#-----------------------------------------------------------------------
#-----------------------------------------------------------------------

# Orography correction just to aid visualizati
pp1 = 500
pp2 = 9000
pp3 = 18
divisor = sfcps - pp1
division = pp2 / divisor
C0 = pp3 - division
```

Numpy: Where

(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:

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

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

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

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

Script 18: Galvez Davison Index (GDI)

(6) GDI

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

```
189    # Galvez Davison Index - Indices with and without correction
190    GDI = ECI + MWI + II
191    GDIc = ECI + MWI + II + C0
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

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

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

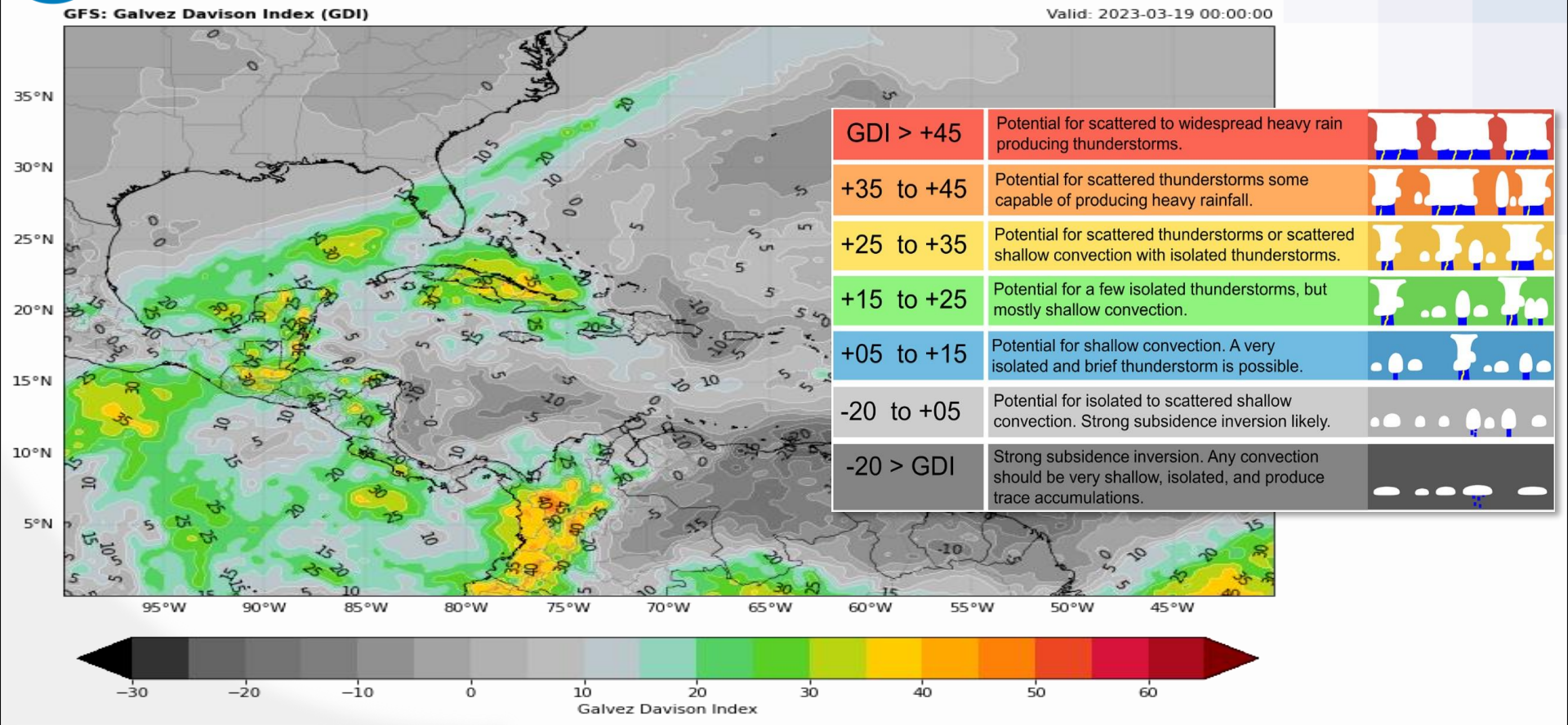
Script 18: Galvez Davison Index (GDI)

```
199 #-----------------------------------------------------------------------------------------------------------
200
201 # Choose the plot size (width x height, in inches)
202 plt.figure(figsize=(8,8))
203
204 # Use the Cilindrical 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://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/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(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)
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", "#bcbcbc", "#bbc7cb", "#b2d2dd", "#90d5bb", "#55d065", "#5acf28", "#bad411", "#ffcc00", "#ffa900",
    "#fc8106", "#eb4722", "#d8133a", "#ac0a1d"]
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)



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.

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

LIBRARIES AND FUNCTIONS

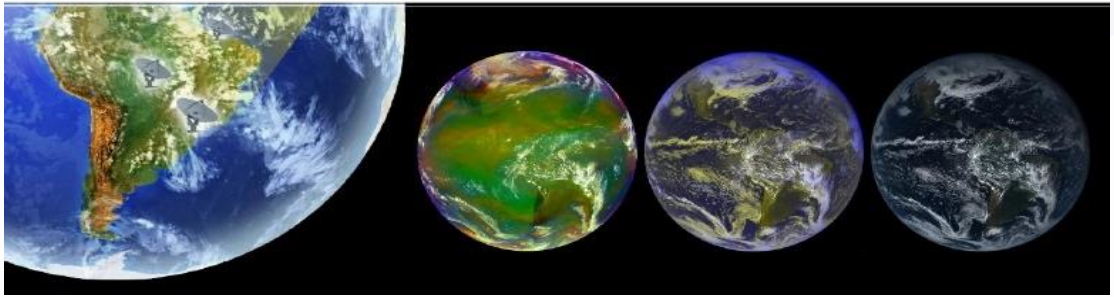
SATELLITE DATA DOWNLOAD

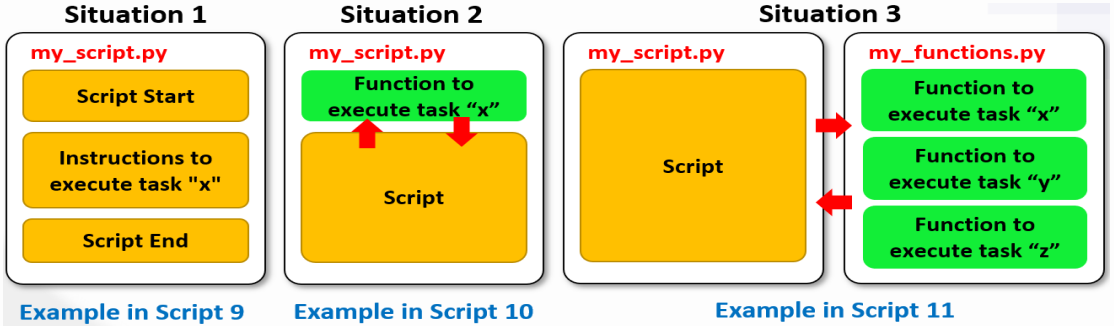
REPROJECTION AND CROPPING

PLOT CONFIGURATION

IMAGE GENERATION

Geostationary Satellite Data Processing With Python: GOES-16





Example in Script 9 | Example in Script 10 | Example in Script 11

Script 19: Satellite Imagery Plot

```
#---------------------------------------------------------------------------------------------------------------
# INPE / CPTEC - Training: Python and GOES-R Imagery: Script 19 - Satellite Plot
# Author: Diego Souza
#---------------------------------------------------------------------------------------------------------------

# Required modules
from netCDF4 import Dataset                          # Read / Write NetCDF4 files
from osgeo import gdal                               # Python bindings for GDAL
import matplotlib.pyplot as plt                      # Plotting library
import cartopy, cartopy.crs as ccrs                  # Plot maps
import cartopy.io.shapereader as shpreader           # Import shapefiles
import os                                            # Miscellaneous operating system interfaces
import numpy as np                                   # Scientific computing with Python
from matplotlib import cm                            # Colormap handling utilities
from datetime import timedelta, date, datetime       # Basic Dates and time types
from utilities import download_CMI                   # Function to download
from utilities import reproject                      # Function for reproject
from utilities import loadCPT                        # Function to load a colormap in the CPT format
gdal.PushErrorHandler('CPLQuietErrorHandler')        # Ignore GDAL warnings
#---------------------------------------------------------------------------------------------------------------
# Input and output directories
input = "Samples"; os.makedirs(input, exist_ok=True)
output = "Output"; os.makedirs(output, exist_ok=True)

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

# Datetime to process (today in this example, to match the GFS date)
#date = datetime.today().strftime('%Y%m%d')
#yyyymmddhhmn = date + '0000'
yyyymmddhhmn = '202107020000' # CHANGE THIS DATE TO THE SAME DATE OF YOUR NWP DATA
#---------------------------------------------------------------------------------------------------------------
# Download the ABI file
file_ir = download_CMI(yyyymmddhhmn, 13, input)
#
```

LIBRARIES AND FUNCTIONS

Function to download data from AWS

Function for reprojection

Function to load a colormap in the CPT format

Avoid showing warnings

Input and output directories

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

Desired time and data

Channel

Call the download function

Script 19: Satellite Imagery Plot

Explaining the new instructions:

```
34  # Download the ABI file
35  file_ir = download_CMI(yyyymmddhhmn, 13, input)
36
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  # Choose the plot size (width x height, in inches)
69  plt.figure(figsize=(8,8))
```

REPROJECTION AND CROPPING

- Variable to be read
- Data reading (specify drive [NetCDF, GeoTIFF, GRIB, etc])
- Metadata (scale, offset, NaN, start of scan, etc)
- Read only the data and convert it to an array
- Applies “scale”, “offset” and converts to °C
- Name of the NetCDF file to be generated
- Call the reprojection function
- Open the reprojected file
- Read the reprojected data

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
80
81  # Plot the image
82  img1 = ax.imshow(data, origin='upper', vmin=-80, vmax=60, extent=img_extent, cmap=colormap, alpha=1.0)
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(shpreader.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(crs=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)
93  gl.top_labels = False
94  gl.right_labels = False
95
96  # Add a colorbar
97  plt.colorbar(img1, label='Brightness Temperatures (°C)', extend='both', orientation='vertical', pad=0.03, fraction=0.05)
98
99  # Extract date
100 date = (datetime.strptime(dtime, '%Y-%m-%dT%H:%M:%S.%fZ'))
101
102 # Add a title
103 plt.title('GOES-16 Band 13 ' + date.strftime('%Y-%m-%d %H:%M') + ' UTC', fontweight='bold', fontsize=10, loc='left')
104 plt.title('Reg.: ' + str(extent) , fontsize=10, loc='right')
105 #---------------------------------------------------------------------------------------------------
106 # Save the image
107 plt.savefig(f'{output}/image_19.png', bbox_inches='tight', pad_inches=0, dpi=300)
108
109 # Show the image
110 plt.show()
```

Explaining the new instructions:

PLOT CONFIGURATION

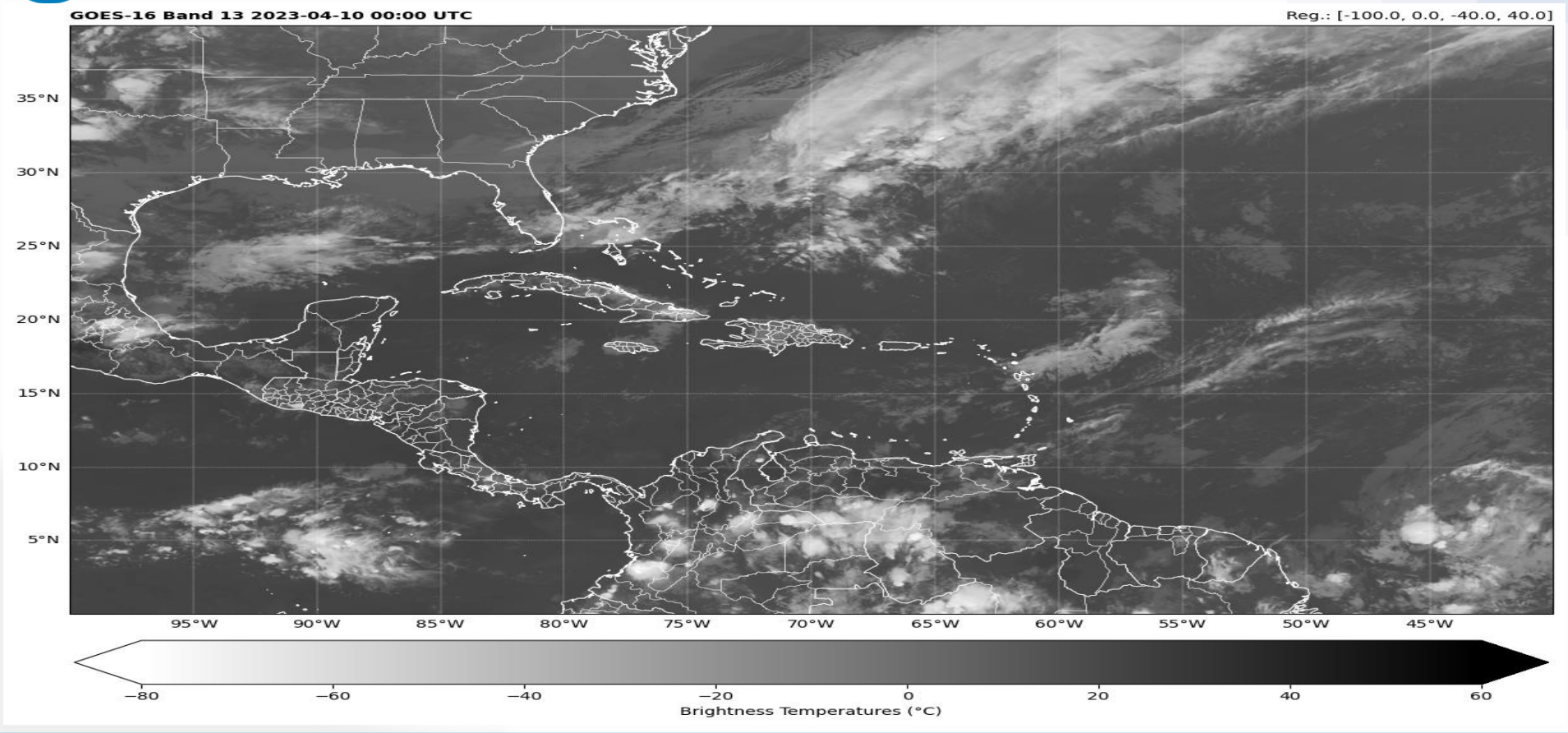
Inverted gray scale

Satellite plot

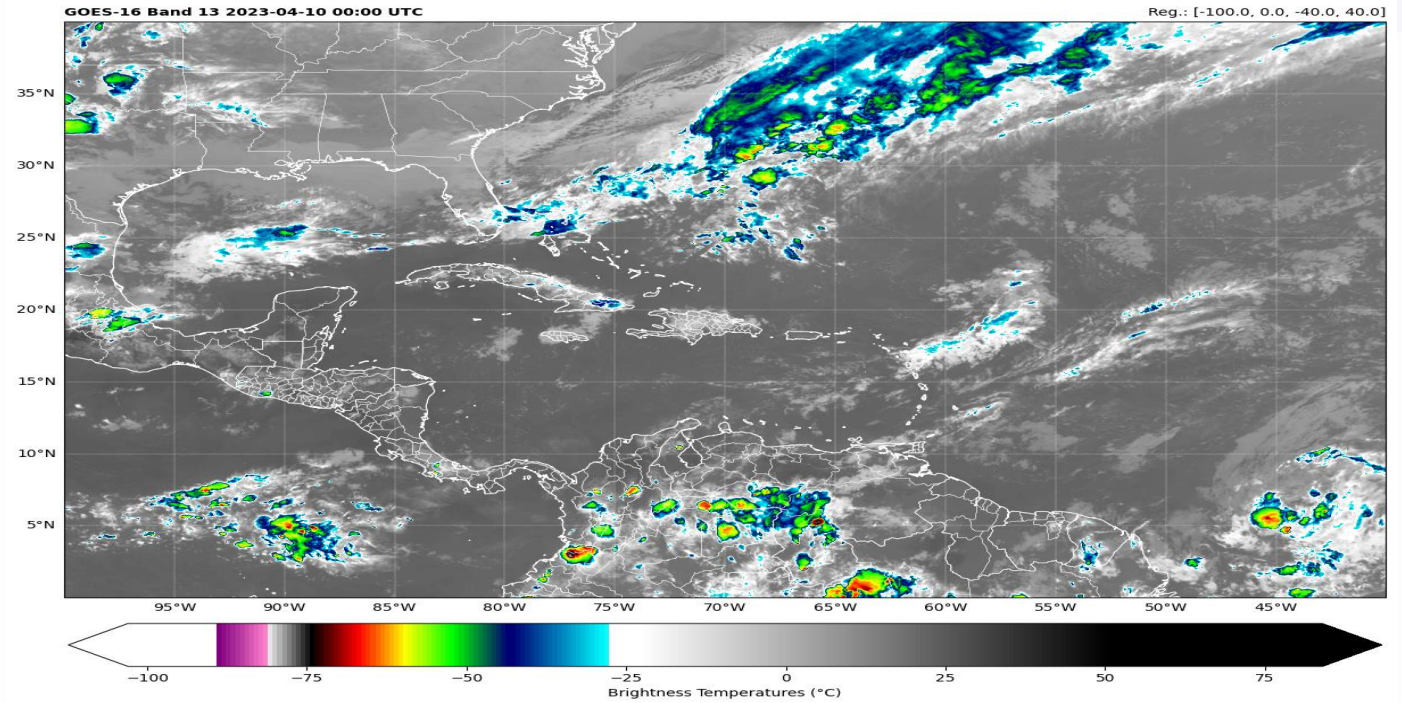
GERAÇÃO DA IMAGEM

f > expressions/variables enclosed in braces in string

Script 19: Satellite Imagery Plot



Script 19: Satellite Imagery Plot

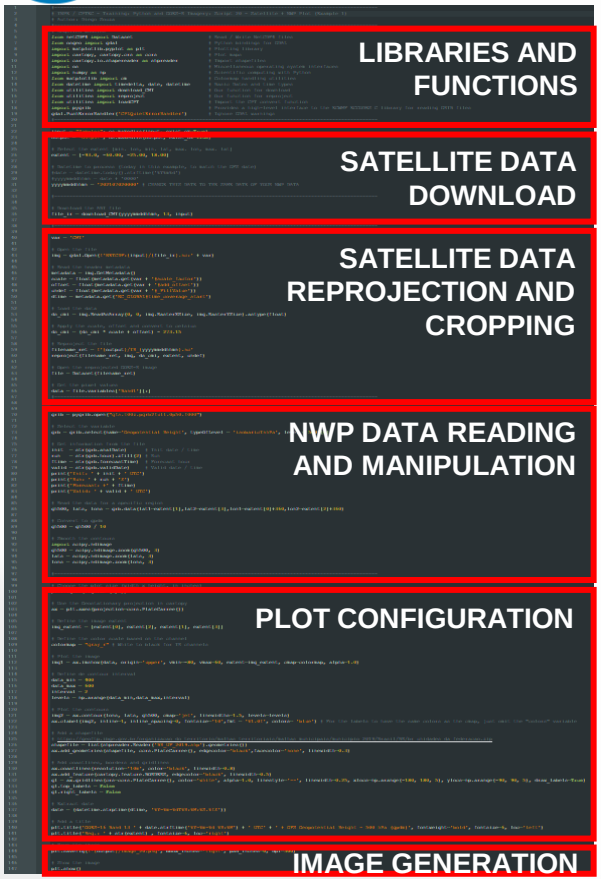



```
# Converts a CPT file to be used in Python
cpt = loadCPT('IR4AVHRR6.cpt')
cmap = cm.colors.LinearSegmentedColormap('cpt', cpt)

# 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)



Explaining the new instructions:

```
#---------------------------------------------------------------------------------------
# INPE / CPTEC - Training: Python and GOES-R Imagery Script 20 - Satellite + NWP Plot (Example 1)
# Author: Diego Souza
#
# Required modules
from netCDF4 import Dataset                  # Read / Write NetCDF4 files
from osgeo import gdal                       # Python binding for GDAL
import matplotlib.pyplot as plt              # Plotting library
import cartopy, cartopy.crs as ccrs          # Plot maps
import cartopy.io.shapereader as shpreader   # Import shapefiles
import os                                    # Miscellaneous operating system interfaces
import numpy as np                           # Scientific computing with Python
from matplotlib import cm                    # Colormap handling utilities
from datetime import timedelta, date, datetime # Basic Dates and time types
from utilities import download_CMI           # Our function for download
from utilities import reproject              # Our function for reproject
from utilities import loadCPT                # Import the CPT convert function
import pygrib                                # Provides a high-level interface to the ECWMF ECCODES C
gdal.PushErrorHandler('CPLQuietErrorHandler') # Ignore GDAL warnings
#---------------------------------------------------------------------------------------
# Input and output directories
input = "Samples"; os.makedirs(input, exist_ok=True)
output = "Output"; os.makedirs(output, exist_ok=True)

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

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

#---------------------------------------------------------------------------------------

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

#---------------------------------------------------------------------------------------
```

Note: Import the PyGRIB library

LIBRARIES AND FUNCTIONS

SATELLITE DATA DOWNLOAD

Script 20: NWP + Satellite (Example 1)

```
file_ir = download_CMI(yyyymmddhhmn, 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_{yyyymmddhhmn}.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'][:]
#-----------------------------------------------------------------------------

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

# Select the variable
grb = grib.select(name='Geopotential Height', typeOfLevel = 'isobaricInhPa', level = 500)[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
gh500, lats, lons = grb.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)

# Convert to gpdm
gh500 = gh500 / 10

# Smooth the contours
import scipy.ndimage
gh500 = scipy.ndimage.zoom(gh500, 3)
lats = scipy.ndimage.zoom(lats, 3)
lons = scipy.ndimage.zoom(lons, 3)

#-----------------------------------------------------------------------------
```

SATELLITE DATA REPROJECTION AND CROPPING

NWP DATA READING AND MANIPULATION

Script 20: NWP + Satellite (Example 1)

Explaining the new instructions:

PLOT CONFIGURATION

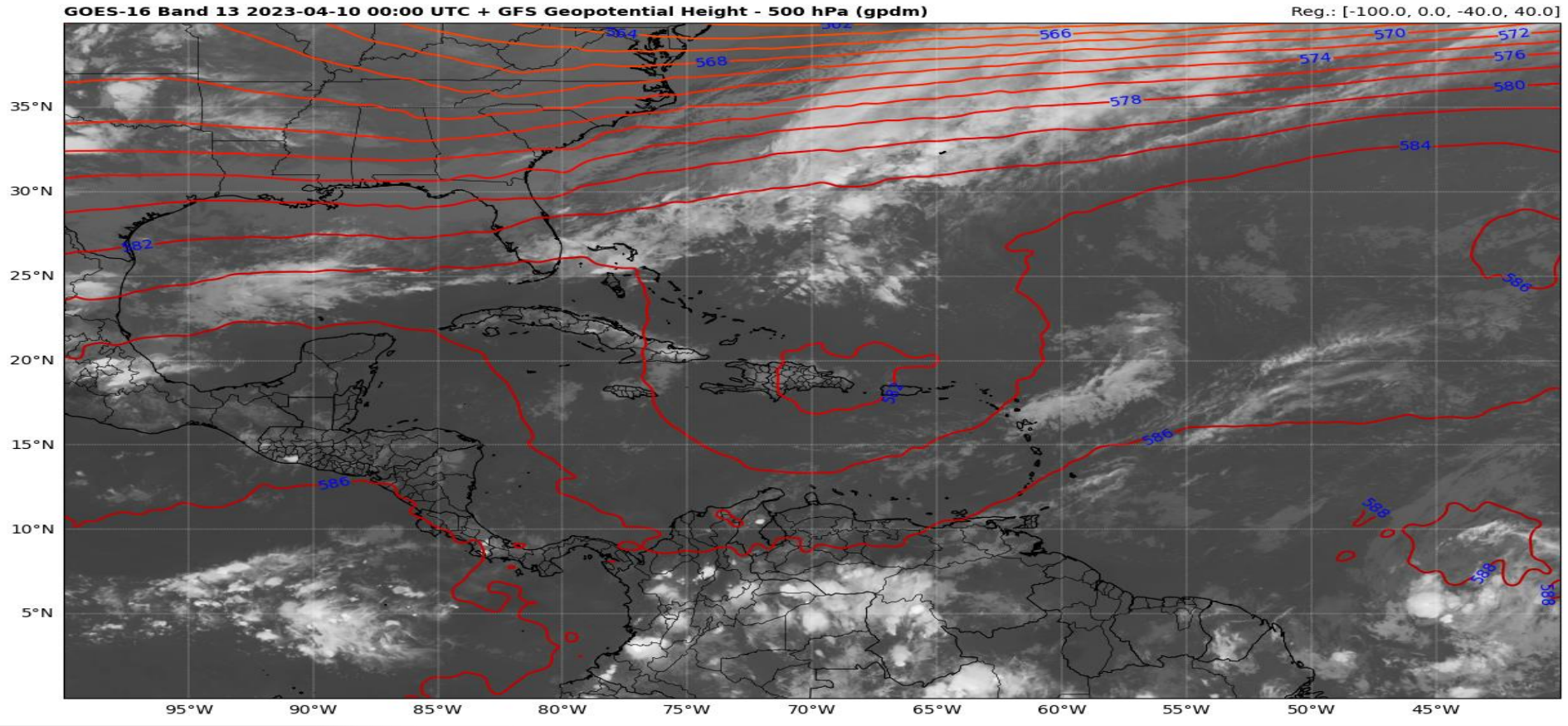
```
#-------------------------------------------------------------------------------------------------------

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

# Use the Geostationary projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree())

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]

# Define the color scale based on the channel
colormap = "gray_r" # White to black for IR channels

# Plot the image
img1 = ax.imshow(data, origin='upper', vmin=-80, vmax=60, extent=img_extent, cmap=colormap, alpha=1.0)

# Define de contour interval
data_min = 400
data_max = 600
interval = 2
levels = np.arange(data_min,data_max,interval)

# Plot the contours
img2 = ax.contour(lons, lats, gh500, cmap='jet', linewidths=1.5, levels=levels)
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

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='black',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.gridlines(crs=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)
gl.top_labels = False
gl.right_labels = False

# Extract date
date = (datetime.strptime(dtime, '%Y-%m-%dT%H:%M:%S.%fZ'))

# Add a title
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')
plt.title('Reg.: ' + str(extent) , fontsize=6, loc='right')
#-------------------------------------------------------------------------------------------------------

# Save the image
plt.savefig(f'{output}/image_20.png', bbox_inches='tight', pad_inches=0, dpi=300)

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

Satellite plot

Plot the geopotential height

IMAGE GENERATION

Script 20: NWP + Satellite (Example 1)



Script 21: NWP + Satellite (Example 2)

Explaining the new instructions:

```
#---------------------------------------------------------------------------------------
# INPE / CPTEC - Training: Python and GOES-R Imagery: Script 21 - Satellite + NWP Plot (Example 2)
# Author: Diego Souza
#
# Required modules
from netCDF4 import Dataset                    # Read / Write NetCDF4 files
from osgeo import gdal                         # Python bindings for GDAL
import matplotlib.pyplot as plt                # Plotting library
import cartopy, cartopy.crs as ccrs            # Plot maps
import cartopy.io.shapereader as shpreader     # Import shapefiles
import os                                      # Miscellaneous operating system interfaces
import numpy as np                             # Scientific computing with Python
from matplotlib import cm                      # Colormap handling utilities
from datetime import timedelta, date, datetime # Basic Dates and time types
from utilities import download_CMI             # Our function for download
from utilities import reproject                # Our function for reproject
from utilities import loadCPT                  # Import the CPT convert function
import pygrib                                  # Provides a high-level interface to the ECWMF ECCODES C
gdal.PushErrorHandler('CPLQuietErrorHandler')  # Ignore GDAL warnings
#
# Input and output directories
input = "Samples"; os.makedirs(input, exist_ok=True)
output = "Output"; os.makedirs(output, exist_ok=True)

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

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

#---------------------------------------------------------------------------------------


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

#
```

LIBRARIES AND FUNCTIONS

SATELLITE DATA DOWNLOAD

Script 21: NWP + Satellite (Example 2)

```
file_ir = download_CMI(yyyymmddhhmn, 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_{yyyymmddhhmn}.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'][:]
#------------------------------------------------------------------------------

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

# Select the variable
grb = grib.select(name='Temperature', typeOfLevel = 'isobaricInhPa', level = 850)[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
tp850, lats, lons = grb.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)

# Convert from K to °C
tp850 = tp850 - 273.15

# Smooth the contours
import scipy.ndimage
tp850 = scipy.ndimage.zoom(tp850, 3)
lats = scipy.ndimage.zoom(lats, 3)
lons = scipy.ndimage.zoom(lons, 3)
#------------------------------------------------------------------------------
```

SATELLITE DATA REPROJECTION AND CROPPING

NWP DATA READING AND MANIPULATION

Script 21: NWP + Satellite (Example 2)

Explaining the new instructions:

PLOT CONFIGURATION

```
# Choose the plot size (width x height, in inches)
plt.figure(figsize=(10,6))

# Use the Geostationary projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree())

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]

# Define the color scale based on the channel
colormap = "gray_r" # White to black for IR channels

# Plot the image
img1 = ax.imshow(data, origin='upper', vmin=-80, vmax=60, extent=img_extent, cmap=colormap, alpha=1.0)

# Define de contour interval
data_min = -20
data_max = 48
interval = 2
levels = np.arange(data_min,data_max,interval)

# Plot the contours
img3 = ax.contour(lons, lats, tp850, cmap='jet', linewidths=1.0, levels=levels)
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

# Get the index of elements with value "0"
zero_value = int(np.where(levels == 0)[0])
img3.collections[zero_value].set_linewidth(4)
img3.collections[zero_value].set_edgecolor('black')

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

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='white', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='white', linewidth=0.5)
gl = ax.gridlines(crs=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)
gl.top_labels = False
gl.right_labels = False

# Extract date
date = (datetime.strptime(dtime, '%Y-%m-%dT%H:%M:%S.%fZ'))

# Add a title
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')
plt.title('Reg.: ' + str(extent) , fontsize=6, loc='right')
#---------------------------------------------------------------------------------------------------
# Save the image
plt.savefig(f'{output}/image_21.png', bbox_inches='tight', pad_inches=0, dpi=300)

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

Satellite plot

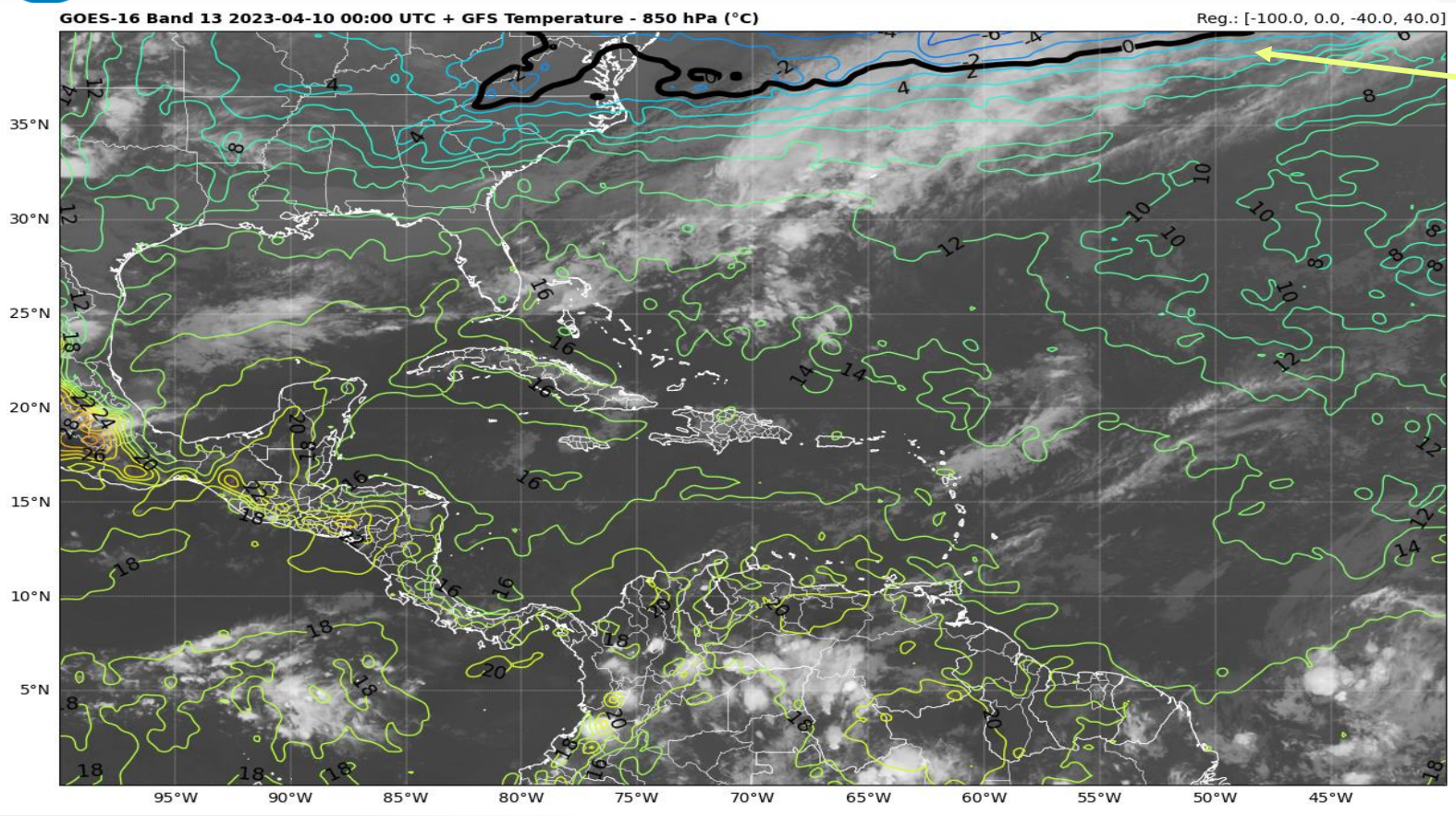
Temperature plot

Numpy: Where

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

IMAGE GENERATION

Script 21: NWP + Satellite (Example 2)



Modified contour, "0" value, in this case

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:

```
#-----------------------------------------------------------------------------------------------------------
# INPE / CPTEC - Training: Python and GOES-R Imagery
# Author: Diego Souza
#-----------------------------------------------------------------------------------------------------------

# Required modules
from netCDF4 import Dataset                       # Read / Write NetCDF4 files
from osgeo import gdal                            # Python bindings for GDAL
import matplotlib.pyplot as plt                   # Plotting library
import cartopy, cartopy.crs as ccrs               # Plot maps
import cartopy.io.shapereader as shpreader        # Import shapefiles
import os                                         # Miscellaneous operating system interfaces
import numpy as np                                # Scientific computing with Python
from matplotlib import cm                         # Colormap handling utilities
from datetime import timedelta, date, datetime    # Basic Dates and time types
from utilities import download_CMI                # Our function for download
from utilities import reproject                   # Our function for reproject
from utilities import loadCPT                     # Import the CPT convert function
import pygrib                                     # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files
gdal.PushErrorHandler('CPLQuietErrorHandler')     # Ignore GDAL warnings

def plot_maxmin_points(lon, lat, data, extrema, nsize, symbol, color='k',
                       plotValue=True, transform=None):
    """
    This function will find and plot relative maximum and minimum for a 2D grid. The function
    can be used to plot an H for maximum values (e.g., High pressure) and an L for minimum
    values (e.g., low pressue). It is best to used filetered 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
    nsize = Size of the grid box to filter the max and min values to plot a reasonable number
    symbol = String to be placed at location of max/min value
    color = String matplotlib colorname to plot the symbol (and numerica 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, nsize, mode='nearest')
    elif (extrema == 'min'):
        data_ext = minimum_filter(data, nsize, mode='nearest')
    else:
        raise ValueError('Value for hilo must be either max or min')

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

    for i in range(len(mxy)):
        txt1 = ax.annotate(symbol, xy=(lon[mxy[i], mxx[i]], lat[mxy[i], mxx[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], mxx[i]])), xy=(lon[mxy[i], mxx[i]], lat[mxy[i], mxx[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 AND FUNCTIONS

FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS

Scipy's maximum and minimum filters

Gets coordinates of the max and min points

Create annotations at the coordinates

Script 22: NWP + Satellite (Example 3)

SATELLITE DATA DOWNLOAD

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

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

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

#-----------------------------------------------------------------------------------------------------------------

# Download the ABI file
file_ir = download_CMI(yyyymmddhhmn, 13, input)
```

SATELLITE DATA REPROJECTION AND CROPPING

```
# 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_{yyyymmddhhmn}.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'][:]
#-----------------------------------------------------------------------------------------------------------------

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

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

Script 22: NWP + Satellite (Example 3)

Explaining the new instructions:

NWP DATA READING AND MANIPULATION

```
#-------------------------------------------------------------------------------------------------------------
# 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

#-------------------------------------------------------------------------------------------------------------

# Select the variable
hght_1000 = grib.select(name='Geopotential Height', typeOfLevel = 'isobaricInhPa', level = 1000)[0]

# Read the data for a specific region
hght_1000 = hght_1000.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]

# Select the variable
hght_500 = grib.select(name='Geopotential Height', typeOfLevel = 'isobaricInhPa', level = 500)[0]

# Read the data for a specific region
hght_500 = hght_500.data(lat1=extent[1],lat2=extent[3],lon1=extent[0]+360,lon2=extent[2]+360)[0]

# Calculate and smooth 1000-500 hPa thickness
thickness_1000_500 = hght_500 - hght_1000
#

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

Thickness (1000-500 hPa)

Script 22: NWP + Satellite (Example 3)

Explaining the new instructions: PLOT CONFIGURATION

```python
# Choose the plot size (width x height, in inches)
plt.figure(figsize=(10,6))

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

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]

# Define the color scale based on the channel
colormap = "gray_r" # White to black for IR channels

# Plot the image
img1 = ax.imshow(data, origin='upper', vmin=-80, vmax=60, extent=img_extent, cmap=colormap, alpha=1.0)

# Define de contour interval
data_min = 4900
data_max = 5900
interval = 20
levels = np.arange(data_min,data_max,interval)

# Plot the contours
img2 = ax.contour(lons, lats, thickness_1000_500, cmap='seismic', linestyles='dashed', linewidths=1.0, levels=levels)
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

# Get the index of elements with value "5400"
mid_value = int(np.where(levels == 5400)[0])
img2.collections[mid_value].set_linewidth(4)
img2.collections[mid_value].set_edgecolor('blue')

# Define de contour interval
data_min = 500
data_max = 1050
interval = 2
levels = np.arange(data_min,data_max,interval)

# Plot the contours
img3 = ax.contour(lons, lats, prmls, colors='black', linewidths=0.7, levels=levels)
ax.clabel(img3, inline=1, inline_spacing=0, fontsize='10',fmt = '%1.0f', colors= 'black')

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

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='white', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='white', linewidth=0.5)
gl = ax.gridlines(crs=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)
gl.top_labels = False
gl.right_labels = False

# Use definition ... ls
plot_maxmin_points(lons, lats, prmls, 'max', 50, symbol='H', color='b',  transform=ccrs.PlateCarree())
plot_maxmin_points(lons, lats, prmls, 'min', 25, symbol='L', color='r', transform=ccrs.PlateCarree())

# Extract date
date = (datetime.strptime(dtime, '%Y-%m-%dT%H:%M:%S.%fZ'))

# Add a title
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')
plt.title('Reg.: ' + str(extent) , fontsize=5, loc='right')
#-----------------------------------------------------------------------
```

Satellite plot

Thickness plot

Modify an specific contour value (“5400”, in this case)

PSML Plot

Arrays

Max. or Min?

Color

Projection

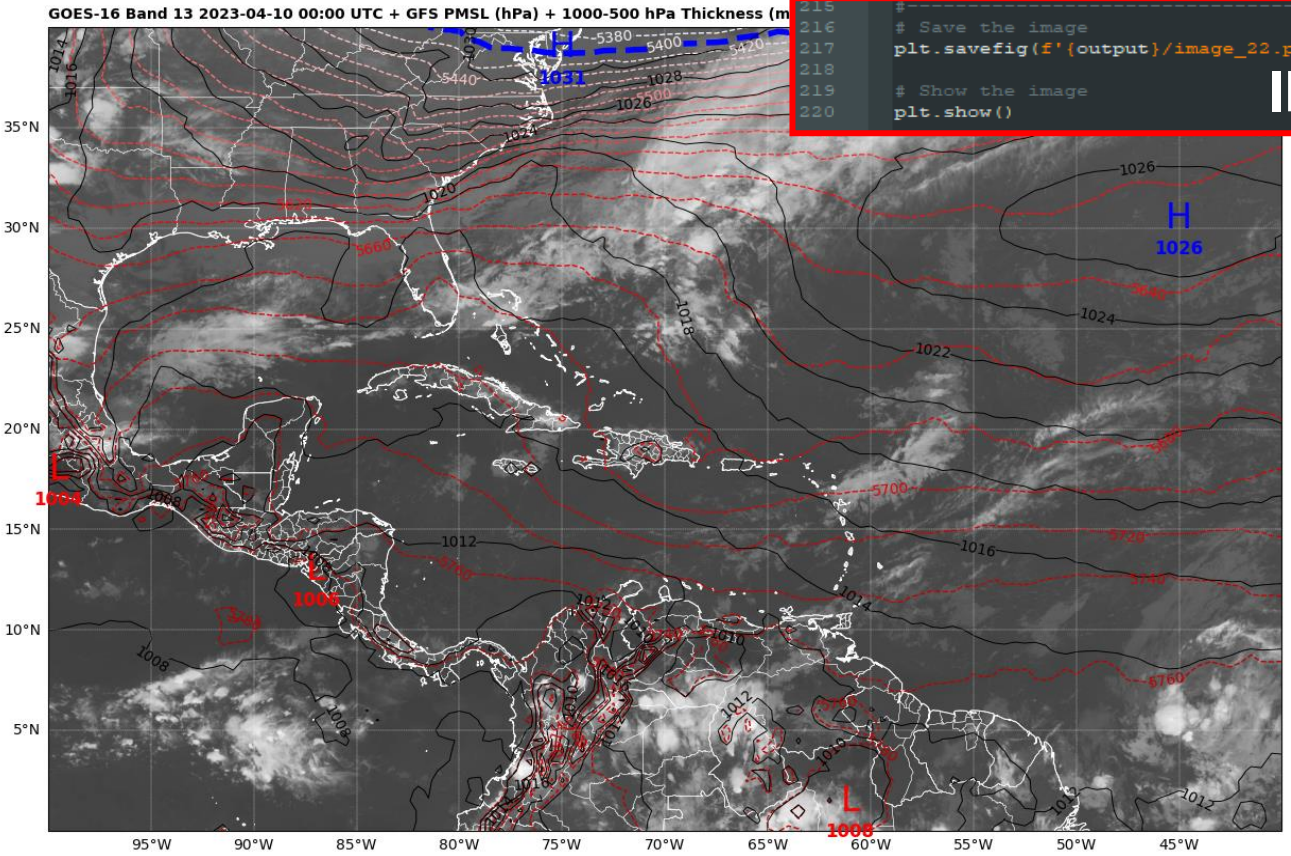
Symbol

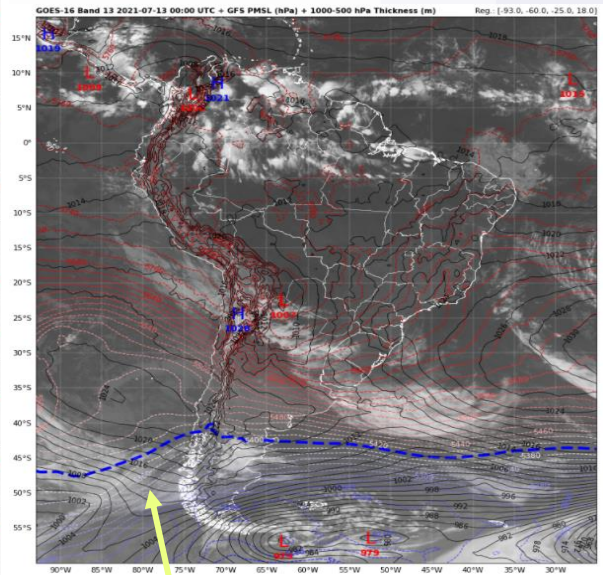
Call the function to plot the high and low pressure labels

Script 22: NWP + Satellite (Example 3)

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

IMAGE GENERATION



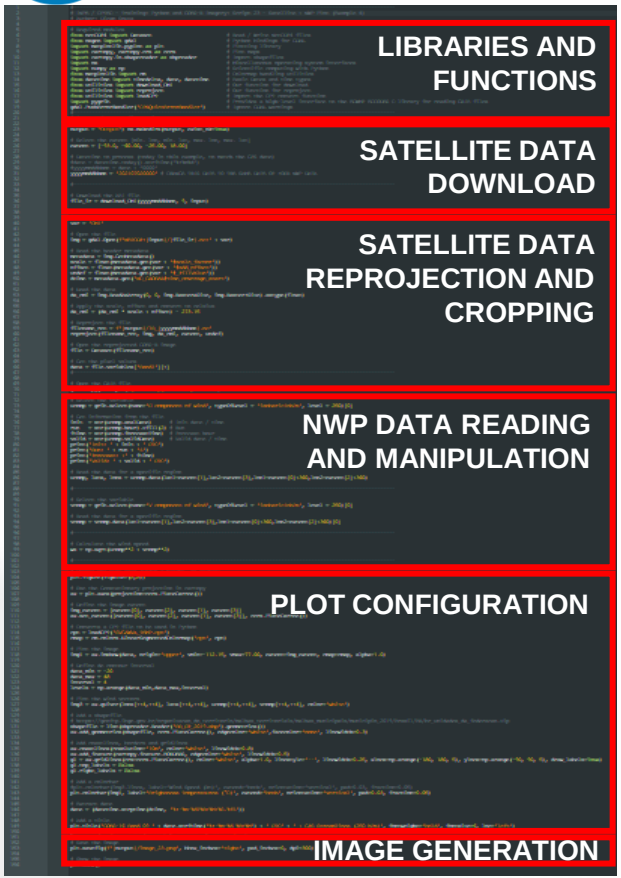


Modified contour, "5400", in this case

Some Quick Modifications

- Change the symbol to be plotted

Script 23: NWP + Satellite (Example 4)




```python
#---------------------------------------------------------------------------------------------------------------------------
# INPE / CPTEC - Training: Python and GOES-R Imagery: Script 23 - Satellite + NWP Plot (Example 4)
# Author: Diego Souza
#
# Required modules
from netCDF4 import Dataset                          # Read / Writ
from osgeo import gdal                               # Python bind
import matplotlib.pyplot as plt                      # Plotting library
import cartopy, cartopy.crs as ccrs                  # Plot maps
import cartopy.io.shapereader as shpreader           # Import shapefiles
import os                                            # Miscellaneous operating system interfaces
import numpy as np                                   # Scientific computing with Python
from matplotlib import cm                            # Colormap handling utilities
from datetime import timedelta, date, datetime       # Basic Dates and time types
from utilities import download_CMI                   # Our function for download
from utilities import reproject                      # Our function for reproject
from utilities import loadCPT                        # Import the CPT convert function
import pygrib                                        # Provides a high-level interface to the ECWMF ECCODES C
gdal.PushErrorHandler('CPLQuietErrorHandler')        # Ignore GDAL warnings
#---------------------------------------------------------------------------------------------------------------------------
# Input and output directories
input = "Samples"; os.makedirs(input, exist_ok=True)
output = "Output"; os.makedirs(output, exist_ok=True)

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

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

#---------------------------------------------------------------------------------------------------------------------------

# Download the ABI file
file_ir = download_CMI(yyyymmddhhmn, 9, input)

#---------------------------------------------------------------------------------------------------------------------------
```

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  #----------------------------------------------------------------------------
39  # Variable
40  var = 'CMI'
41
42  # Open the file
43  img = gdal.Open(f'NETCDF:{input}/{file_ir}.nc:' + var)
44
45  # Read the header metadata
46  metadata = img.GetMetadata()
47  scale = float(metadata.get(var + '#scale_factor'))
48  offset = float(metadata.get(var + '#add_offset'))
49  undef = float(metadata.get(var + '#_FillValue'))
50  dtime = metadata.get('NC_GLOBAL#time_coverage_start')
51
52  # Load the data
53  ds_cmi = img.ReadAsArray(0, 0, img.RasterXSize, img.RasterYSize).astype(float)
54
55  # Apply the scale, offset and convert to celsius
56  ds_cmi = (ds_cmi * scale + offset) - 273.15
57
58  # Reproject the file
59  filename_ret = f'{output}/IR_{yyyymmddhhmn}.nc'
60  reproject(filename_ret, img, ds_cmi, extent, undef)
61
62  # Open the reprojected GOES-R image
63  file = Dataset(filename_ret)
64
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")
```

SATELLITE DATA REPROJECTION AND CROPPING

Script 23: NWP + Satellite (Example 4)

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

# 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 = 250)[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 = 250)[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)

#
```

NWP DATA READING AND MANIPULATION

Script 23: NWP + Satellite (Example 4)

Explaining the new instructions:

PLOT CONFIGURATION

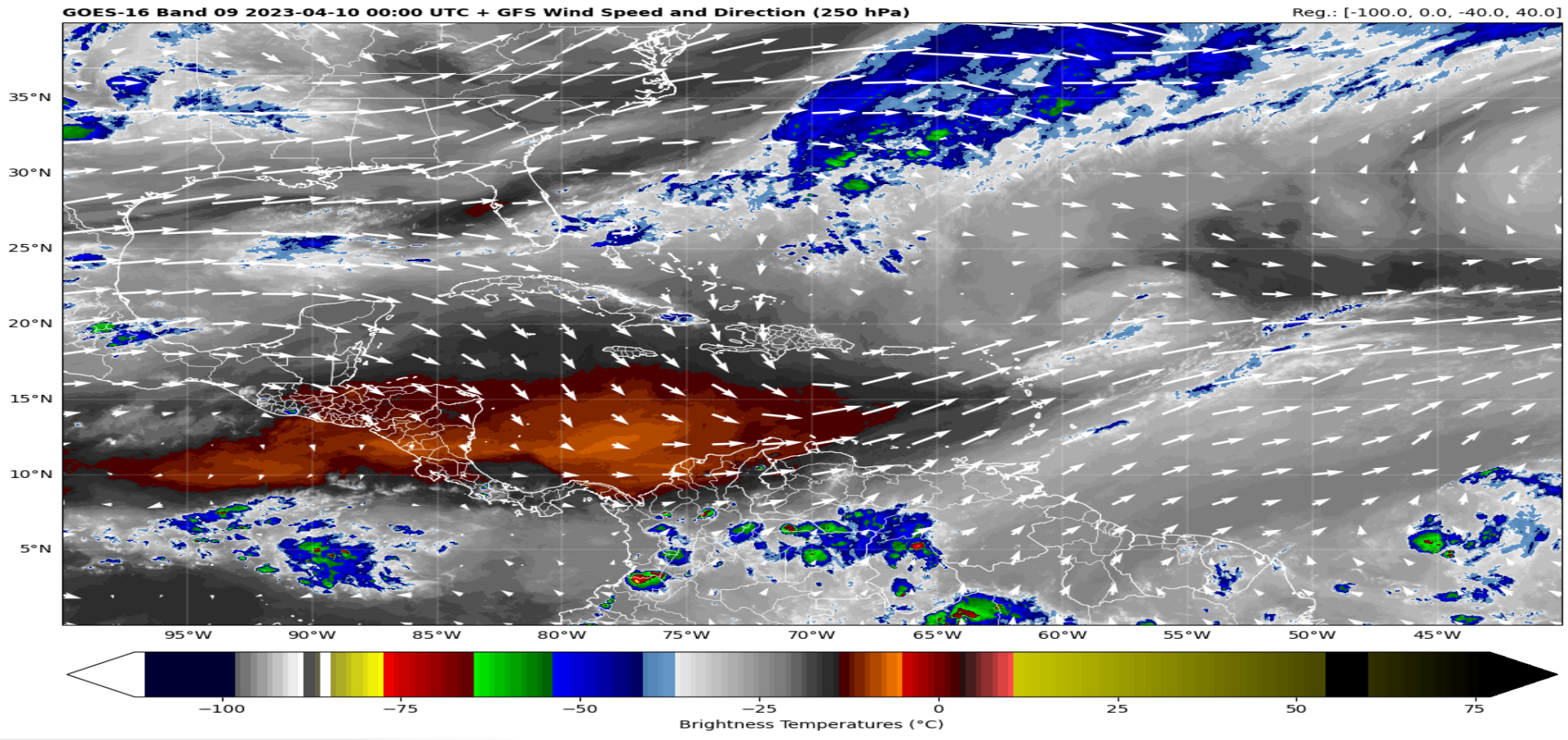
```
#---------------------------------------------------------------------------------------------------

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

# Use the Geostationary projection in cartopy
ax = plt.axes(projection=ccrs.PlateCarree())

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]
ax.set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())

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

# Plot the image
img1 = ax.imshow(data, origin='upper', vmin=-112.15, vmax=77.00, extent=img_extent, cmap=cmap, alpha=1.0)

# Define de contour interval
data_min = -20
data_max = 48
interval = 4
levels = np.arange(data_min,data_max,interval)

# Plot the wind vectors
img3 = ax.quiver(lons[::4,::4], lats[::4,::4], ucomp[::4,::4], vcomp[::4,::4], color='white')

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='white',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='white', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='white', linewidth=0.5)
gl = ax.gridlines(crs=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)
gl.top_labels = False
gl.right_labels = False

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

# Extract date
date = (datetime.strptime(dtime, '%Y-%m-%dT%H:%M:%S.%fZ'))

# Add a title
plt.title('GOES-16 Band 09 ' + date.strftime('%Y-%m-%d %H:%M') + ' UTC' + ' + GFS Streamlines (250 hPa)', fontweight='bold', fontsize=6, loc='left')
plt.title('Reg.: ' + str(extent) , fontsize=6, loc='right')

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

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

Satellite plot

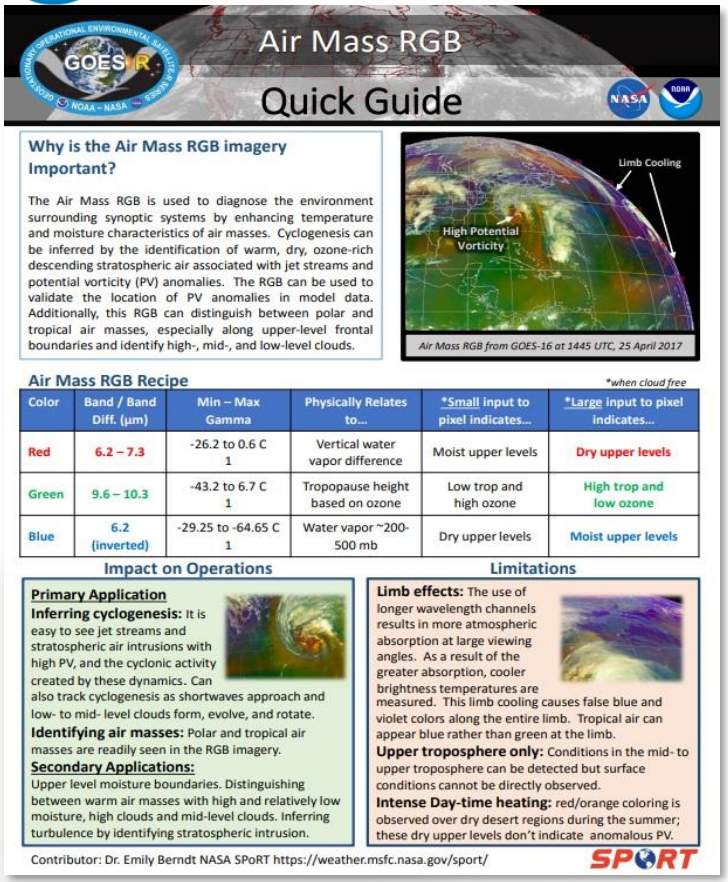
Wind vector plot

IMAGE GENERATION

Script 23: NWP + Satellite (Example 4)



Script 24: NWP + Satellite (Example 5)



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 from GOES-16 at 1445 UTC, 25 April 2017

Air Mass RGB Recipe

*when cloud free

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

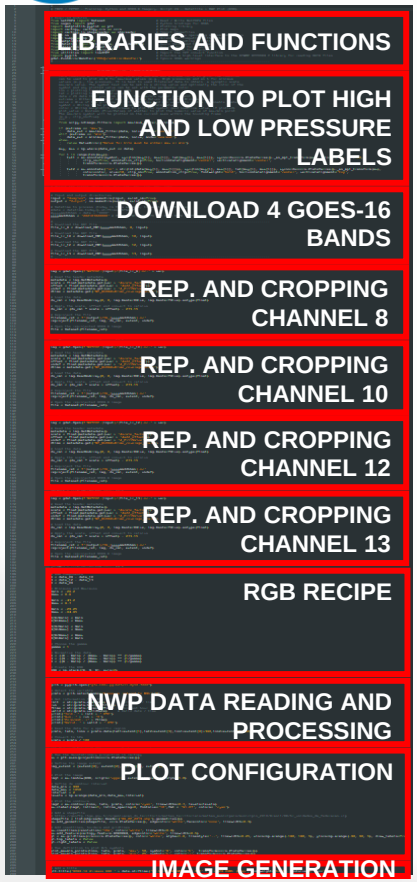
LIBRARIES AND FUNCTIONS

```
#----------------------------------------------------------------------------------------------------
# INPE / CPTEC - Training: Python and GOES-R Imagery: Script 24 - Satellite + NWP Plot (RGB)
# Author: Diego Souza
#----------------------------------------------------------------------------------------------------

# Required modules
from netCDF4 import Dataset                          # Read / Write NetCDF4 files
from osgeo import gdal                               # Python bindings for GDAL
import matplotlib.pyplot as plt                      # Plotting library
import cartopy, cartopy.crs as ccrs                  # Plot maps
import cartopy.io.shapereader as shpreader           # Import shapefiles
import os                                            # Miscellaneous operating system interfaces
import numpy as np                                   # Scientific computing with Python
from matplotlib import cm                            # Colormap handling utilities
from datetime import timedelta, date, datetime       # Basic Dates and time types
from utilities import download_CMI                   # Our function for download
from utilities import reproject                      # Our function for reproject
from utilities import loadCPT                        # Import the CPT convert function
import pygrib                                        # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files
gdal.PushErrorHandler('CPLQuietErrorHandler')        # Ignore GDAL warnings
```

FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS

```
def plot_maxmin_points(lon, lat, data, extrema, nsize, symbol, color='k',
                       plotValue=True, transform=None):
    """
    This function will find and plot relative maximum and minimum for a 2D grid. The function
    can be used to plot an H for maximum values (e.g., High pressure) and an L for minimum
    values (e.g., low pressue). It is best to used filetered 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
    nsize = Size of the grid box to filter the max and min values to plot a reasonable number
    symbol = String to be placed at location of max/min value
    color = String matplotlib colorname to plot the symbol (and numerica 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, nsize, mode='nearest')
    elif (extrema == 'min'):
        data_ext = minimum_filter(data, nsize, mode='nearest')
    else:
        raise ValueError('Value for hilo must be either max or min')

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

    for i in range(len(mxy)):
        txt1 = ax.annotate(symbol, xy=(lon[mxy[i], mxx[i]], lat[mxy[i], mxx[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], mxx[i]])), xy=(lon[mxy[i], mxx[i]], lat[mxy[i], mxx[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())

#----------------------------------------------------------------------------------------------------
```

Script 24: NWP + Satellite (Example 5)



DOWNLOAD 4 GOES-16 BANDS (AIRMASS)

```
61  #----------------------------------------------------------------------------
62
63  # Select the extent [min. lon, min. lat, max. lon, max. lat]
64  extent = [-93.0, -60.00, -25.00, 18.00]
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  #----------------------------------------------------------------------------
```

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

Explaining the new instructions:

REPROJECTION AND CROPPING (4 BANDS)

```
87  #---------------------------------------------------------------------------
88  # Variable
89  var = 'CMI'
90
91  # Open the file
92  img = gdal.Open(f'NETCDF:{input}/{file_ir_8}.nc:' + var)
93
94  # Read the header metadata
95  metadata = img.GetMetadata()
96  scale = float(metadata.get(var + '#scale_factor'))
97  offset = float(metadata.get(var + '#add_offset'))
98  undef = float(metadata.get(var + '#_FillValue'))
99  dtime = metadata.get('NC_GLOBAL#time_coverage_start')
100
101 # Load the data
102 ds_cmi = img.ReadAsArray(0, 0, img.RasterXSize, img.RasterYSize).astype(float)
103
104 # Apply the scale, offset and convert to celsius
105 ds_cmi = (ds_cmi * scale + offset) - 273.15
106
107 # Reproject the file
108 filename_ret = f'{output}/IR_{yyyymmddhhmn}.nc'
109 reproject(filename_ret, img, ds_cmi, extent, undef)
110
111 # Open the reprojected GOES-R image
112 file = Dataset(filename_ret)
113
114 # Get the pixel values
115 data_08 = file.variables['Band1'][:]
116
117 # Open the file
118 img = gdal.Open(f'NETCDF:{input}/{file_ir_10}.nc:' + var)
119
120 # Read the header metadata
121 metadata = img.GetMetadata()
122 scale = float(metadata.get(var + '#scale_factor'))
123 offset = float(metadata.get(var + '#add_offset'))
124 undef = float(metadata.get(var + '#_FillValue'))
125 dtime = metadata.get('NC_GLOBAL#time_coverage_start')
126
```

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

```
192 # Get the pixel values
193 data_13 = file.variables['Band1'][:]
194 #---------------------------------------------------------------------------
195 #---------------------------------------------------------------------------
196 # RGB Components
197 R = data_08 - data_10
198 G = data_12 - data_13
199 B = data_08
200
201 # Minimuns and Maximuns
202 Rmin = -26.2
203 Rmax = 0.6
204
205 Gmin = -43.2
206 Gmax = 6.7
207
208 Bmin = -29.25
209 Bmax = -64.65
210
211 R[R<Rmin] = Rmin
212 R[R>Rmax] = Rmax
213
214 G[G<Gmin] = Gmin
215 G[G>Gmax] = Gmax
216
217 B[B<Bmax] = Bmax
218 B[B>Bmin] = Bmin
219
220 # Choose the gamma
221 gamma = 1
222
223 # Normalize the data
224 R = ((R - Rmin) / (Rmax - Rmin)) ** (1/gamma)
225 G = ((G - Gmin) / (Gmax - Gmin)) ** (1/gamma)
226 B = ((B - Bmin) / (Bmax - Bmin)) ** (1/gamma)
227
228 # Create the RGB
229 RGB = np.stack([R, G, B], axis=2)
230 #---------------------------------------------------------------------------
231 #---------------------------------------------------------------------------
```

RGB RECIPE (AIRMASS)

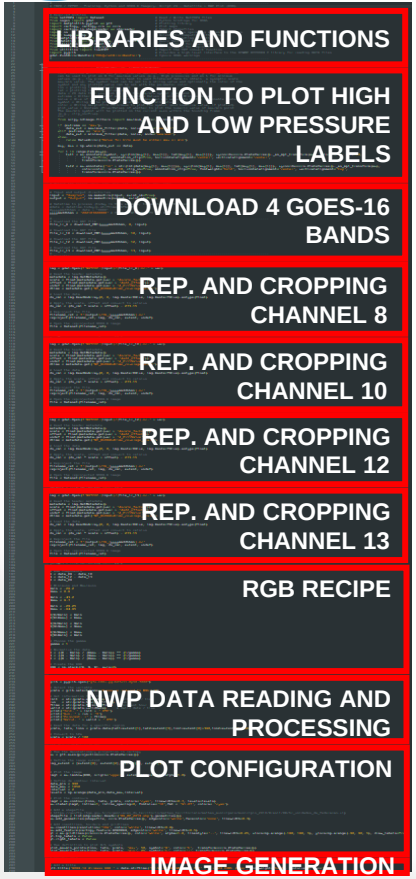
Explaining the new instructions:

- Calculations from recipe
- Minimums and Maximums
- Gamma
- Normalization
- Create the stack

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)




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

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

Explaining the new instructions:

NWP PLOT CONFIGURATION

RGB Plot

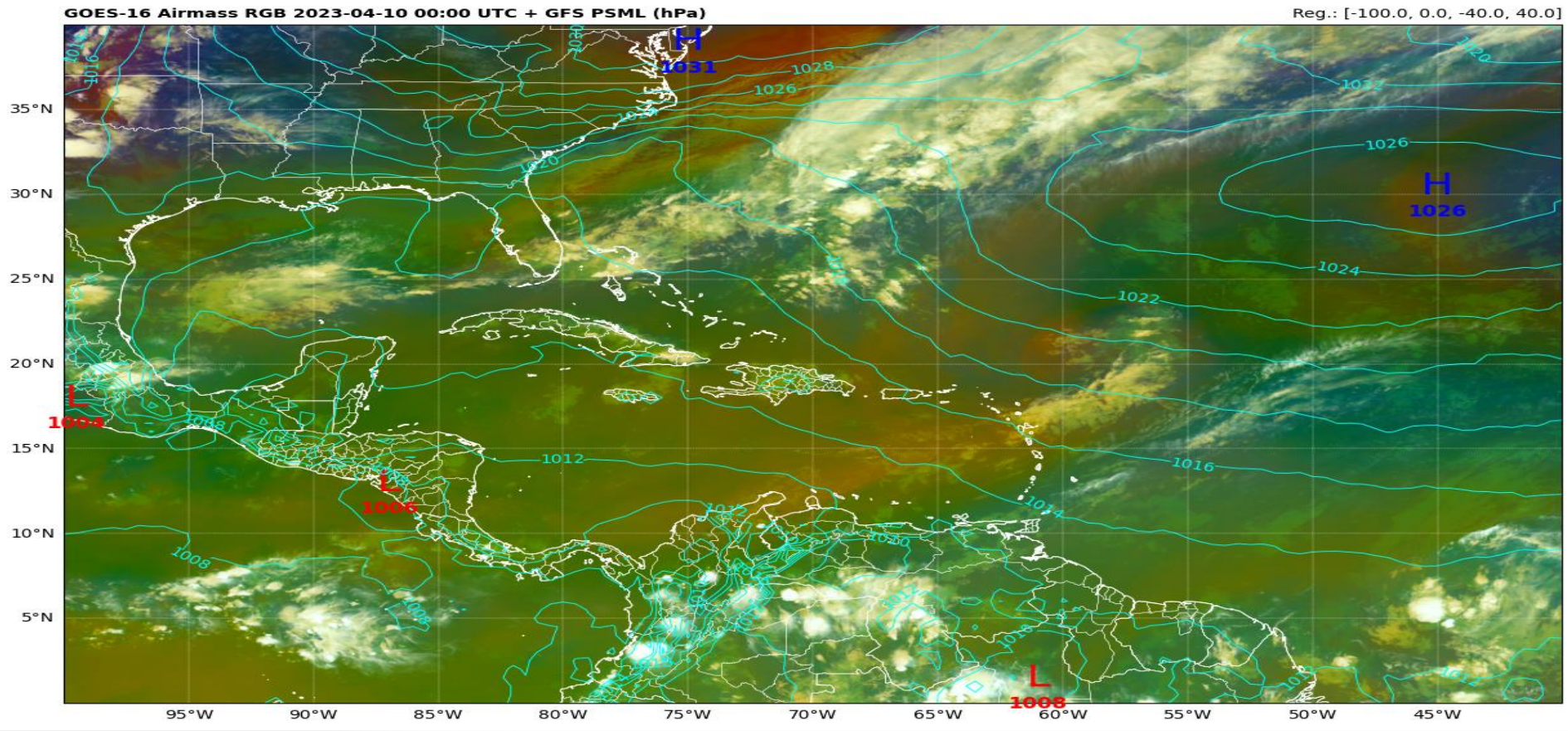
PSML Plot

Calling the function to plot high and low pressure labels

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/br_unidades_da_federacao.zip
280 shapefile = list(shpreader.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(crs=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-%dT%H:%M:%S.%fZ'))
296
297 # Add a title
298 plt.title('GOES-16 Airmass RGB ' + date.strftime('%Y-%m-%d %H:%M') + ' UTC' + ' + GFS PSML (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()
```

Script 24: NWP + Satellite (Example 5)



Script 25: METAR Plot

LIBRARIES

METAR DATA DOWNLOAD

PLOT CONFIGURATION

METAR PLOT

IMAGE GENERATION

Explaining the new instructions:

```python
#-----------------------------------------------------------------------------------------------------------
# INPE / CPTEC - Training: Python a
# Author: Diego Souza
#

# Adapted from: https://unidata.github.io/MetPy/latest/examples/plots/Station_Plot.html

import matplotlib.pyplot as plt                       # Plotting library
import cartopy, cartopy.crs as ccrs                   # Plot maps
import cartopy.io.shapereader as shpreader            # Import shapefiles
import cartopy.feature as cfeature                    #       on drawing and filtering operations
import os                                             # Miscellaneous operating system interfaces
import numpy as np                                    #                           with Python
import requests                                       #                           on
from datetime import timedelta, date, datetime        # Basic Dates and time types
from metpy.calc import reduce_point_density           # Provide tools for unit-aware, meteorological calculations
from metpy.io import metar                            #
from metpy.plots import current_weather, sky_cover, StationPlot    # Contains functionality for making meteorological plots
#-----------------------------------------------------------------------------------------------------------

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

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

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

#-----------------------------------------------------------------------------------------------------------
```

LIBRARIES

'requests' library (the same used to download GFS data)

Funções da Biblioteca MetPy

DOWNLOAD THE METAR DATA

Desired date

Desired time

Data download

Read the METAR data

Discard missing data

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

Script 25: METAR Plot

LIBRARIES

METAR DATA DOWNLOAD

PLOT CONFIGURATION

METAR PLOT

IMAGE GENERATION

```
data = metar.parse_metar_file(dir + '///' + file_name)


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


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

# Set up the map projection
proj = ccrs.PlateCarree()

# Use the Geostationary projection in cartopy
ax = plt.axes(projection=proj)

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]
ax.set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())


# Change the DPI of the resulting figure. Higher DPI drastically improves the
# look of the text rendering.
plt.rcParams['savefig.dpi'] = 255

# Use the Cartopy map projection to transform station locations to the map and
# then refine the number of stations plotted by setting a minimum radius
point_locs = proj.transform_points(ccrs.PlateCarree(), data['longitude'].values, data['latitude'].values)
data = data[reduce_point_density(point_locs, 3)]

# Add some various map elements to the plot to make it recognizable.
ax.add_feature(cfeature.LAND)
ax.add_feature(cfeature.OCEAN)

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_un
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.gridlines(crs=ccrs.PlateCarree(), color='white', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 1
gl.top_labels = False
gl.right_labels = False


#
# 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 to 12 pt.
stationplot = StationPlot(ax, data['longitude'].values, data['latitude'].values,
```

PLOT CONFIGURATION

Script 25: METAR Plot

Explaining the new instructions:

METAR PLOT

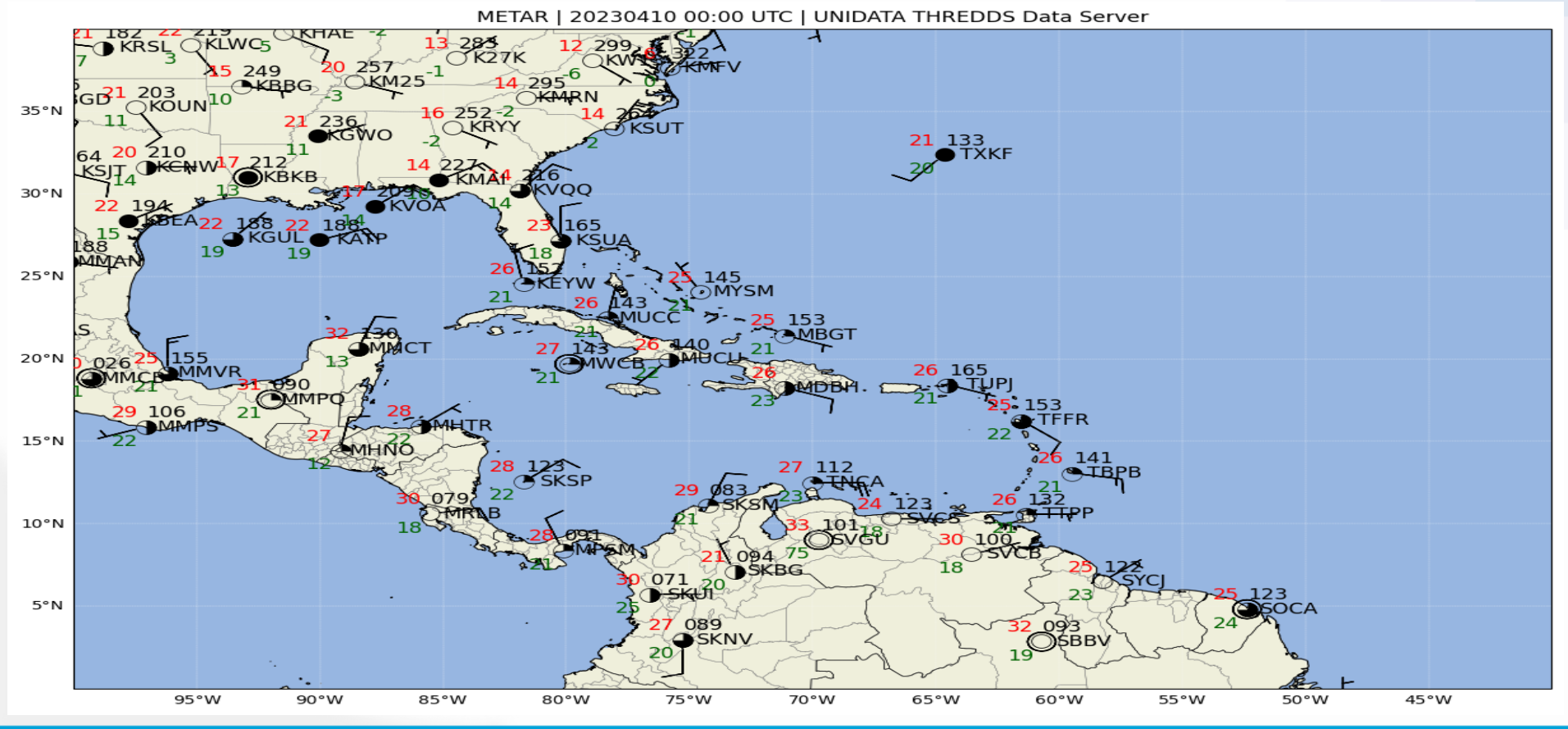
```python
gl.right_labels = False

#-----------------------------------------------------------------------------------------------------------
# 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 to 12 pt.
stationplot = StationPlot(ax, data['longitude'].values, data['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['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 | ' + date + ' 00:00 UTC | UNIDATA THREDDS Data Server', fontsize=8, loc='center')
#-----------------------------------------------------------------------------------------------------------
```

- Indicates which axis to plot on and the METAR lon/lat variables
- Plot the temperature and Dew Point on the left
- Plot the pressure at sea level on the right
- Plot the cloud cover
- Plot the current weather
- Plot the wind barbs
- Station ID
- Plot the title

IMAGE GENERATION

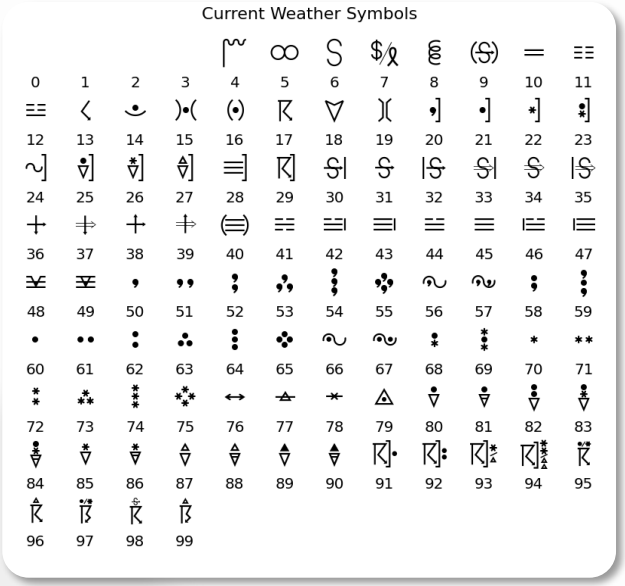
```python
# Save the image
plt.savefig(f'{output}/image_25.png', bbox_inches='tight', pad_inches=0, dpi=300)

plt.show()
```

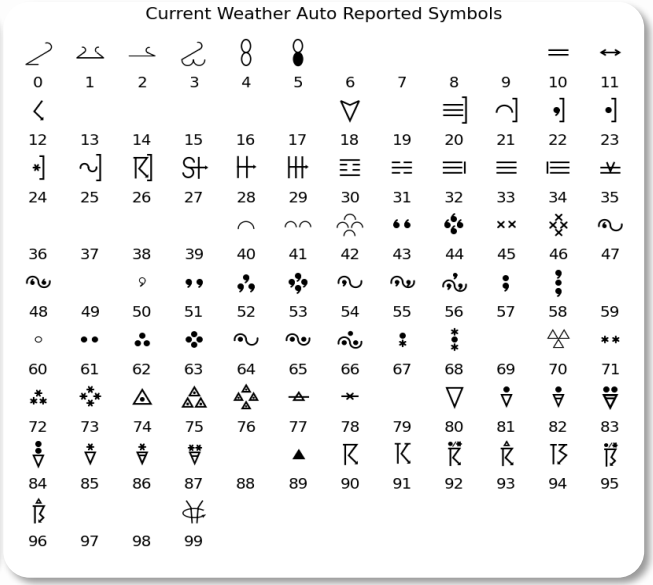
Script 25: METAR Plot

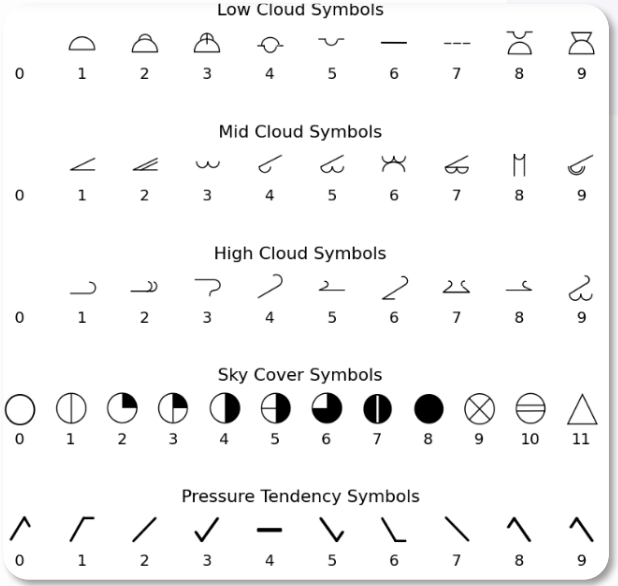


Script 25: METAR Plot

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



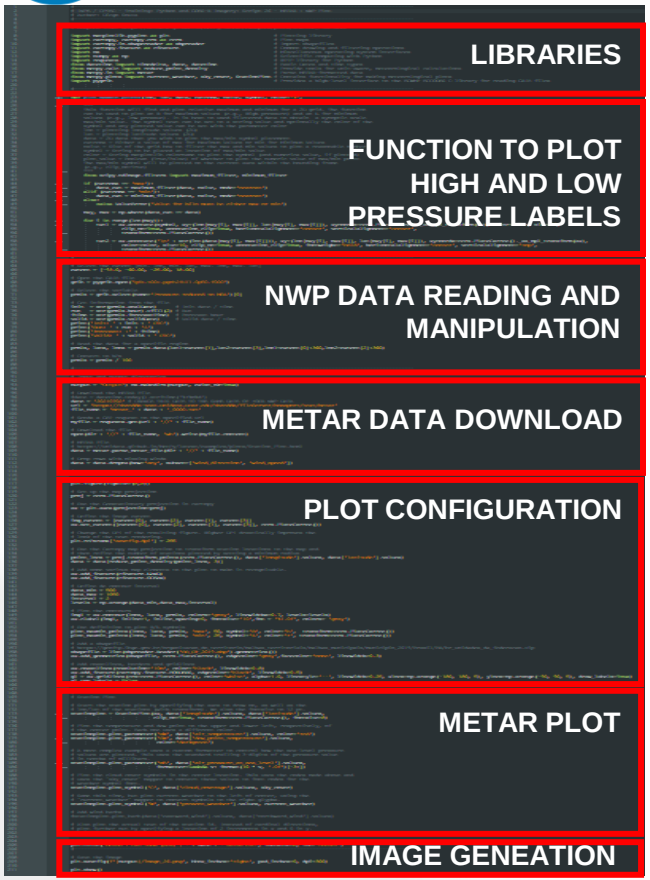




Some Quick Modifications

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

Script 26: METAR + NWP

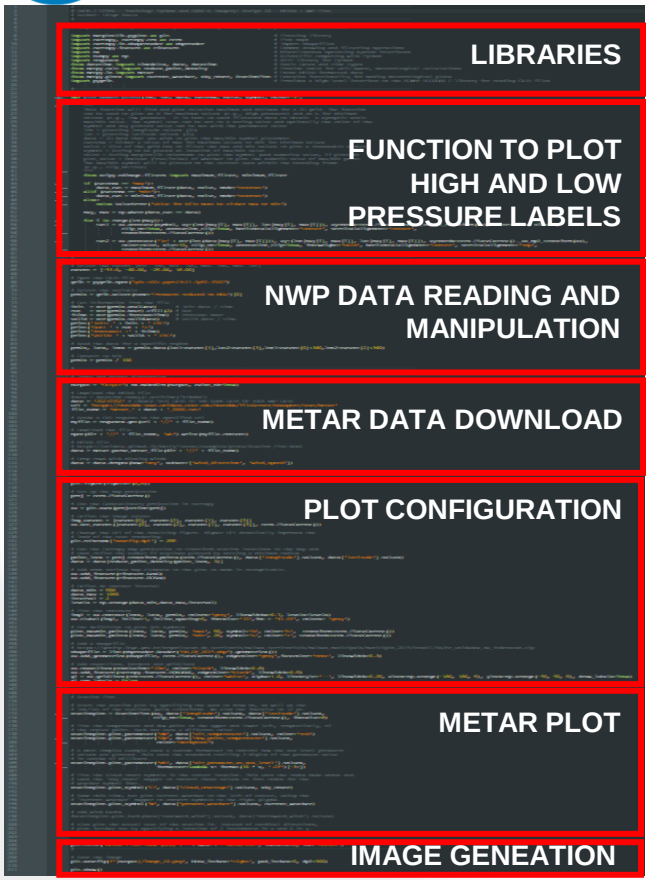


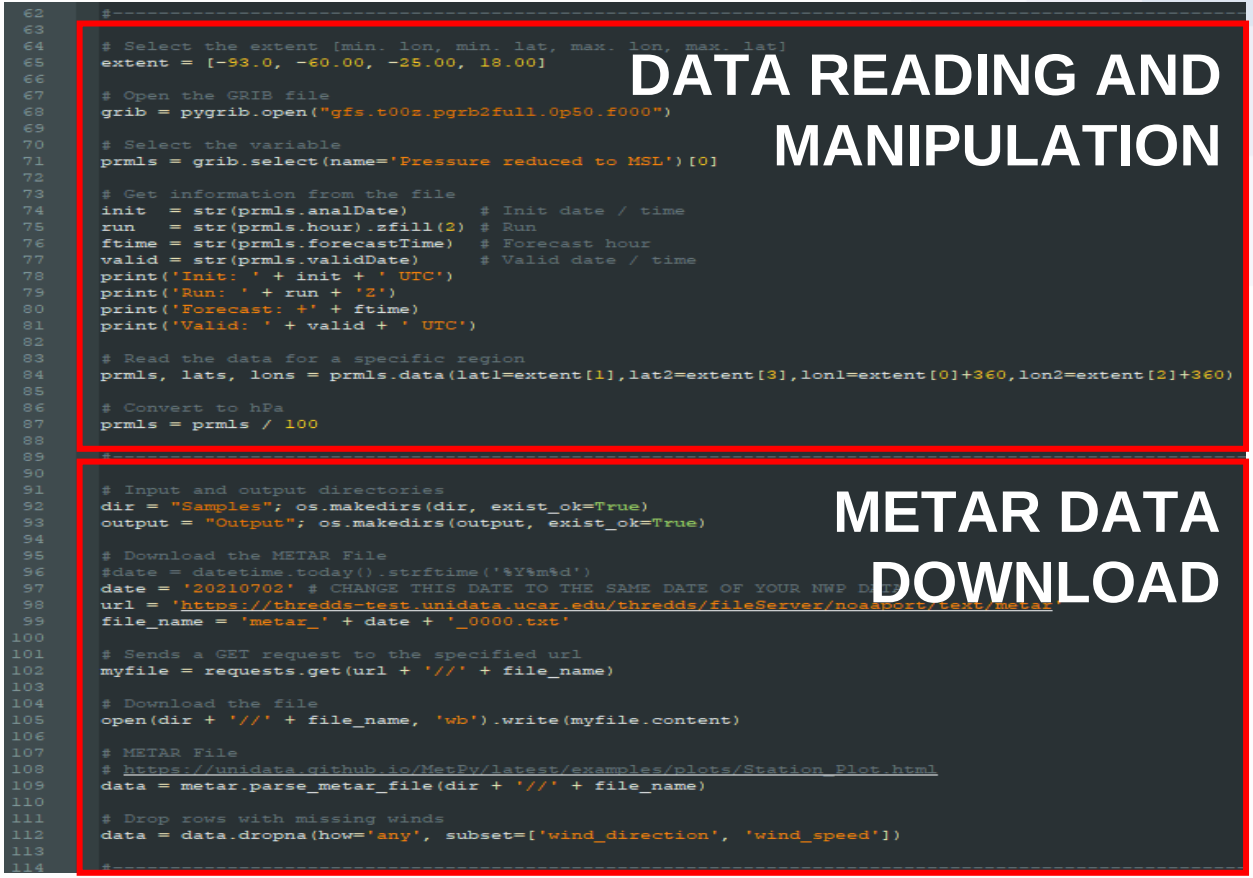

```python
#-------------------------------------------------------------------------------
# INPE / CPTEC - Training: Python and GOES-R Imagery: Script 26 - METAR + NWP Plot
# Author: Diego Souza
#-------------------------------------------------------------------------------

# Adapted from: https://unidata.github.io/MetPy/latest/examples/plots/Station_Plot.html

import matplotlib.pyplot as plt                 # Plotting library
import cartopy, cartopy.crs as ccrs             # Plot maps
import cartopy.io.shapereader as shpreader      # Import shapefiles
import cartopy.feature as cfeature              # Common drawing and filtering operations
import os                                       # Miscellaneous operating system interfaces
import numpy as np                              # Scientific computing with Python
import requests                                 # HTTP library for Python
from datetime import timedelta, date, datetime  # Basic Dates and time types
from metpy.calc import reduce_point_density     # Provide tools for unit-aware, meteorological calculations
from metpy.io import metar                      # Parse METAR-formatted data
from metpy.plots import current_weather, sky_cover, StationPlot # Contains functionality for making meteorological plots
import pygrib                                   # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files

#-------------------------------------------------------------------------------

def plot_maxmin_points(lon, lat, data, extrema, nsize, symbol, color='k',
                       plotValue=True, transform=None):
    """
    This function will find and plot relative maximum and minimum values for the data
    can be used to plot an H for maximum values (e.g., High pressure) and an L for minimum
    values (e.g., low pressue). It is best to used filetered 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
    nsize = Size of the grid box to filter the max and min values to plot a reason
    symbol = String to be placed at location of max/min value
    color = String matplotlib colorname to plot the symbol (and numerica 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, nsize, mode='nearest')
    elif (extrema == 'min'):
        data_ext = minimum_filter(data, nsize, mode='nearest')
    else:
        raise ValueError('Value for hilo must be either max or min')

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

    for i in range(len(mxy)):
        txt1 = ax.annotate(symbol, xy=(lon[mxy[i], mxx[i]], lat[mxy[i], mxx[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], mxx[i]])), xy=(lon[mxy[i], mxx[i]], lat[mxy[i], mxx[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 THE HIGH AND LOW PRESSURE LABELS

Script 26: METAR + NWP






```
#----------------------------------------------------------------------------

# 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
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 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/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.parse_metar_file(dir + '//' + file_name)

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

#----------------------------------------------------------------------------
```

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 GENEATION

PLOT CONFIGURATION (SAT + NWP)

```
#-------------------------------------------------------------------------------

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

# Set up the map projection
proj = ccrs.PlateCarree()

# Use the Geostationary projection in cartopy
ax = plt.axes(projection=proj)

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]
ax.set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())

# Change the DPI of the resulting figure. Higher DPI drastically improves the
# look of the text rendering.
plt.rcParams['savefig.dpi'] = 255

# Use the Cartopy map projection to transform station locations to the map and
# then refine the number of stations plotted by setting a minimum radius
point_locs = proj.transform_points(ccrs.PlateCarree(), data['longitude'].values, data['latitude'].values)
data = data[reduce_point_density(point_locs, 3)]

# Add some various map elements to the plot to make it recognizable.
ax.add_feature(cfeature.LAND)
ax.add_feature(cfeature.OCEAN)

# Define de contour interval
data_min = 500
data_max = 1050
interval = 2
levels = np.arange(data_min,data_max,interval)

# Plot the contours
img1 = ax.contour(lons, lats, prmls, colors='gray', linewidths=0.7, levels=levels)
ax.clabel(img1, inline=1, inline_spacing=0, fontsize='10',fmt = '%1.0f', colors= 'gray')

# Use definition to plot H/L symbols
plot_maxmin_points(lons, lats, prmls, 'max', 50, symbol='H', color='b',  transform=ccrs.PlateCarree())
plot_maxmin_points(lons, lats, prmls, 'min', 25, symbol='L', color='r', transform=ccrs.PlateCarree())

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.gridlines(crs=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)
gl.top_labels = False
gl.right_labels = False

#-------------------------------------------------------------------------------
```

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

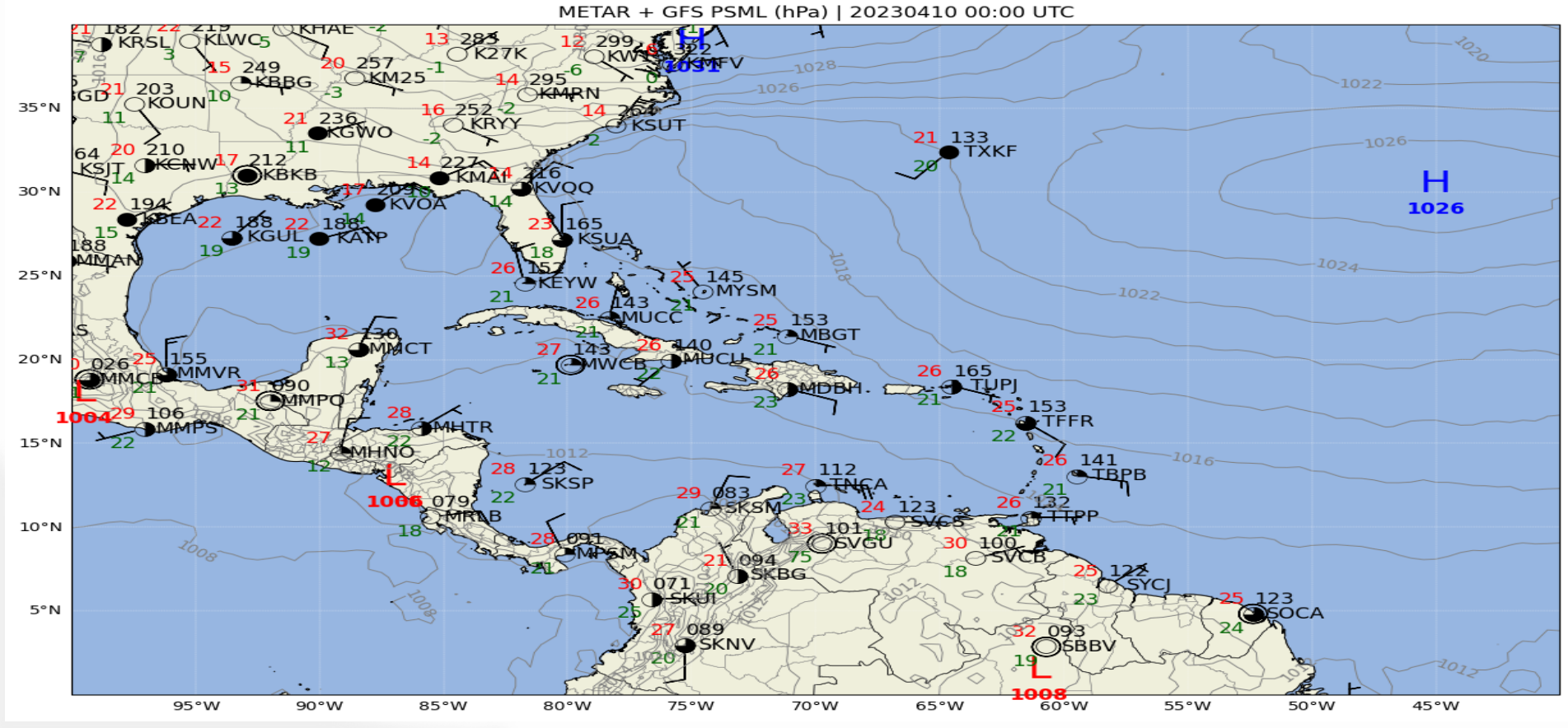
IMAGE GENEATION

METAR PLOT

IMAGE GENERATION

```
167
168 # Station Plot
169
170 # Start the station plot by specifying the axes to draw on, as well as the
171 # lon/lat of the stations (with transform). We also the fontsize to 12 pt.
172 stationplot = StationPlot(ax, data['longitude'].values, data['latitude'].values,
173                           clip_on=True, transform=ccrs.PlateCarree(), fontsize=8)
174
175 # Plot the temperature and dew point to the upper and lower left, respectively, of
176 # the center point. Each one uses a different color.
177 stationplot.plot_parameter('NW', data['air_temperature'].values, color='red')
178 stationplot.plot_parameter('SW', data['dew_point_temperature'].values,
179                           color='darkgreen')
180
181 # A more complex example uses a custom formatter to control how the sea-level pressure
182 # values are plotted. This uses the standard trailing 3-digits of the pressure value
183 # in tenths of millibars.
184 stationplot.plot_parameter('NE', data['air_pressure_at_sea_level'].values,
185                           formatter=lambda v: format(10 * v, '.0f')[-3:])
186
187 # Plot the cloud cover symbols in the center location. This uses the codes made above and
188 # uses the `sky_cover` mapper to convert these values to font codes for the
189 # weather symbol font.
190 stationplot.plot_symbol('C', data['cloud_coverage'].values, sky_cover)
191
192 # Same this time, but plot current weather to the left of center, using the
193 # `current_weather` mapper to convert symbols to the right glyphs.
194 stationplot.plot_symbol('W', data['current_wx1_symbol'].values, current_weather)
195
196 # Add wind barbs
197 stationplot.plot_barb(data['eastward_wind'].values, data['northward_wind'].values)
198
199 # Also plot the actual text of the station id. Instead of cardinal directions,
200 # plot further out by specifying a location of 2 increments in x and 0 in y.
201 stationplot.plot_text((2, 0), data['station_id'].values)
202
203 # Add a title
204 plt.title('METAR + GFS PSML (hPa) | ' + date + ' 00:00 UTC', fontsize=8, loc='center')
205 #-------------------------------------------------------------------------------
206
207 # Save the image
208 plt.savefig(f'{output}/image_26.png', bbox_inches='tight', pad_inches=0, dpi=[illegible]
209
210 plt.show()
```

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

```
#-------------------------------------------------------------------------------
# INPE / CPTEC - Training: Python and GOES-R Imagery: Script 27 - METAR + NWP + Satellite Plot
# Author: Diego Souza
#-------------------------------------------------------------------------------

# Required modules
from netCDF4 import Dataset                                 # Read / Write NetCDF4 files
from osgeo import gdal                                      # Python bindings for GDAL
import matplotlib.pyplot as plt                             # Plotting library
import cartopy, cartopy.crs as ccrs                         # Plot maps
import cartopy.io.shapereader as shpreader                  # Import shapefiles
import os                                                   # Miscellaneous operating system interfaces
import numpy as np                                          # Scientific computing with Python
from matplotlib import cm                                   # Colormap handling utilities
from utilities import download_CMI                          # Our function for download
from utilities import reproject                             # Our function for reproject
from utilities import loadCPT                               # Import the CPT convert function
import requests                                             # HTTP library for Python
from datetime import timedelta, date, datetime              # Basic Dates and time types
from metpy.calc import reduce_point_density                 # Provide tools for unit-aware, meteorological calculations
from metpy.io import metar                                  # Parse METAR-formatted data
from metpy.plots import current_weather, sky_cover, StationPlot # Contains functionality for making meteorological plots
import pygrib                                               # Provides a high-level interface to the ECWMF ECCODES C library for reading GRIB files
gdal.PushErrorHandler('CPLQuietErrorHandler')               # Ignore GDAL warnings

#-------------------------------------------------------------------------------

def plot_maxmin_points(lon, lat, data, extrema, nsize, symbol, color='k',
                       plotValue=True, transform=None):
    """
    This function will find and plot relative maximum and minimum for a 2D grid. The function
    can be used to plot an H for maximum values (e.g., High pressure) and an L for minimum
    values (e.g., low pressue). It is best to used filetered 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
    nsize = Size of the grid box to filter the max and min values to plot a reasonable
    symbol = String to be placed at location of max/min value
    color = String matplotlib colorname to plot the symbol (and numerica value, if plotted)
    plot_value = Boolean (True/False) of whether to plot the numeric value of max/min points
    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, nsize, mode='nearest')
    elif (extrema == 'min'):
        data_ext = minimum_filter(data, nsize, mode='nearest')
    else:
        raise ValueError('Value for hilo must be either max or min')

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

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

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

#-------------------------------------------------------------------------------
```

LIBRARIES

FUNCTION TO PLOT HIGH AND LOW PRESSURE LABELS

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

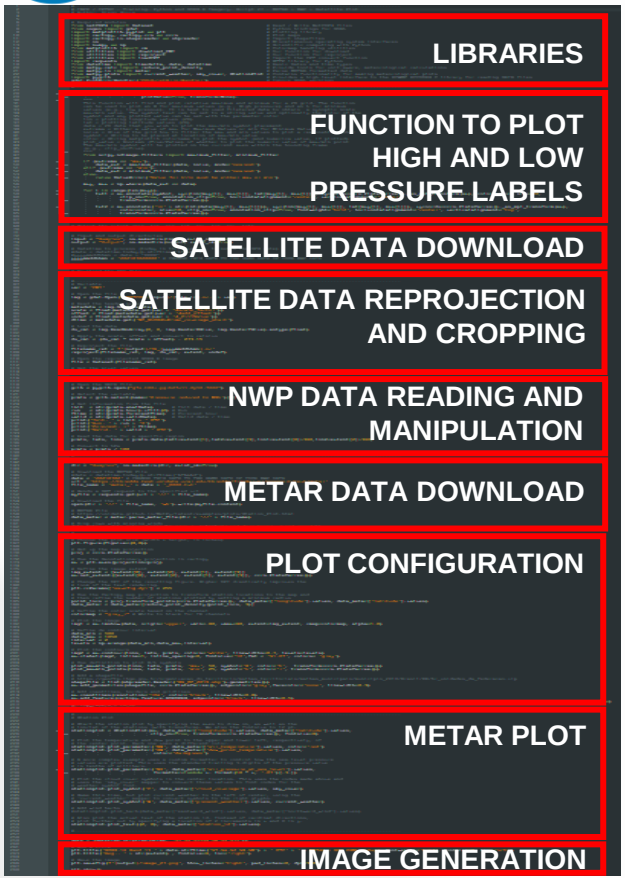
SATELLITE DATA DOWNLOAD

```
#----------------------------------------------------------------------------

# 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')
#yyyymmddhhmn = date + '0000'
yyyymmddhhmn = '202107020000' # CHANGE THIS DATE TO THE SAME DATE OF YOUR NWP DATA

#----------------------------------------------------------------------------

# Download the ABI file
file_ir = download_CMI(yyyymmddhhmn, 13, input)
```

SATELLITE DATA REPROJECTION AND CROPPING

```
# 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_{yyyymmddhhmn}.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'][:]

#----------------------------------------------------------------------------
```

Script 27: METAR + NWP + Satellite



NWP DATA READING AND MANIPULATION

```
# 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
```

METAR DOWNLOAD

```
# 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'])
```

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

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

# Set up the map projection
proj = ccrs.PlateCarree()

# Use the Geostationary projection in cartopy
ax = plt.axes(projection=proj)

# Define the image extent
img_extent = [extent[0], extent[2], extent[1], extent[3]]
ax.set_extent([extent[0], extent[2], extent[1], extent[3]], ccrs.PlateCarree())

# Change the DPI of the resulting figure. Higher DPI drastically improves the
# look of the text rendering.
plt.rcParams['savefig.dpi'] = 255

# Use the Cartopy map projection to transform station locations to the map and
# then refine the number of stations plotted by setting a minimum radius
point_locs = proj.transform_points(ccrs.PlateCarree(), data_metar['longitude'].values, data_metar['latitude'].values)
data_metar = data_metar[reduce_point_density(point_locs, 3)]

# Define the color scale based on the channel
colormap = "gray_r" # White to black for IR channels

# Plot the image
img1 = ax.imshow(data, origin='upper', vmin=-80, vmax=60, extent=img_extent, cmap=colormap, alpha=1.0)

# Define de contour interval
data_min = 500
data_max = 1050
interval = 2
levels = np.arange(data_min,data_max,interval)

# Plot the contours
img2 = ax.contour(lons, lats, prmls, colors='white', linewidths=0.7, levels=levels)
ax.clabel(img2, inline=1, inline_spacing=0, fontsize='10',fmt = '%1.0f', colors= 'gray')

# Use definition to plot H/L symbols
plot_maxmin_points(lons, lats, prmls, 'max', 50, symbol='H', color='b',  transform=ccrs.PlateCarree())
plot_maxmin_points(lons, lats, prmls, 'min', 25, symbol='L', color='r', transform=ccrs.PlateCarree())

# Add a shapefile
# https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2019/Brasil/BR/br_unidades_da_federacao.zip
shapefile = list(shpreader.Reader('BR_UF_2019.shp').geometries())
ax.add_geometries(shapefile, ccrs.PlateCarree(), edgecolor='gray',facecolor='none', linewidth=0.3)

# Add coastlines, borders and gridlines
ax.coastlines(resolution='10m', color='black', linewidth=0.8)
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
gl = ax.gridlines(crs=ccrs.PlateCarree(), color='white', alpha=1.0, linestyle='--', linewidth=0.25, xlocs=np.arange(-180, 180, 5), ylocs=np.arange(
gl.top_labels = False
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

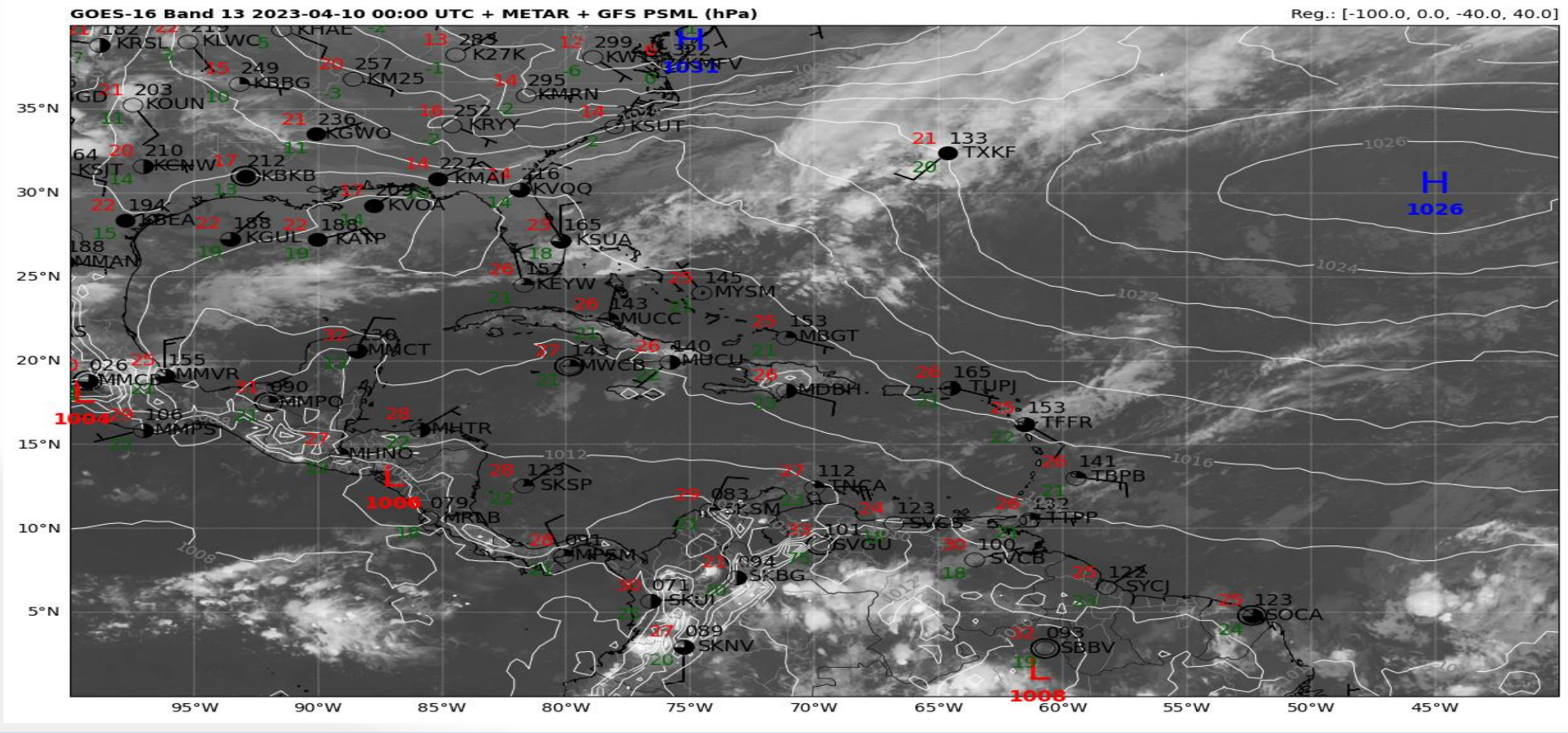
METAR PLOT

```
219
220     # Station Plot
221
222     # Start the station plot by specifying the axes to draw on, as well as the
223     # lon/lat of the stations (with transform). We also the fontsize to 12 pt.
224     stationplot = StationPlot(ax, data_metar['longitude'].values, data_metar['latitude'].values,
225                               clip_on=True, transform=ccrs.PlateCarree(), fontsize=8)
226
227     # Plot the temperature and dew point to the upper and lower left, respectively, of
228     # the center point. Each one uses a different color.
229     stationplot.plot_parameter('NW', data_metar['air_temperature'].values, color='red')
230     stationplot.plot_parameter('SW', data_metar['dew_point_temperature'].values,
231                                color='darkgreen')
232
233     # A more complex example uses a custom formatter to control how the sea-level pressure
234     # values are plotted. This uses the standard trailing 3-digits of the pressure value
235     # in tenths of millibars.
236     stationplot.plot_parameter('NE', data_metar['air_pressure_at_sea_level'].values,
237                                formatter=lambda v: format(10 * v, '.0f')[-3:])
238
239     # Plot the cloud cover symbols in the center location. This uses the codes made above and
240     # uses the `sky_cover` mapper to convert these values to font codes for the
241     # weather symbol font.
242     stationplot.plot_symbol('C', data_metar['cloud_coverage'].values, sky_cover)
243
244     # Same this time, but plot current weather to the left of center, using the
245     # `current_weather` mapper to convert symbols to the right glyphs.
246     stationplot.plot_symbol('W', data_metar['current_wx1_symbol'].values, current_weather)
247
248     # Add wind barbs
249     stationplot.plot_barb(data_metar['eastward_wind'].values, data_metar['northward_wind'].values)
250
251     # Also plot the actual text of the station id. Instead of cardinal directions,
252     # plot further out by specifying a location of 2 increments in x and 0 in y.
253     stationplot.plot_text((2, 0), data_metar['station_id'].values)
254
255     #
256
```

IMAGE GENERATION

```
257     # Extract the date from the satellite data
258     date = (datetime.strptime(dtime, '%Y-%m-%dT%H:%M:%S.%fZ')
259
260     # Add a title
261     plt.title('GOES-16 Band 13 ' + date.strftime('%Y-%m-%d %H:%M') + ' UTC' + ' + METAR + GFS PSML
262     plt.title('Reg.: ' + str(extent) , fontsize=6, loc='right')
263
264     # Save the image
265     plt.savefig(f'{output}/image_27.png', bbox_inches='tight', pad_inches=0, dpi=300)
266
267     plt.show()
```

Script 27: METAR + NWP + Satellite



“Output” Directory With Imagery

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

Search Output

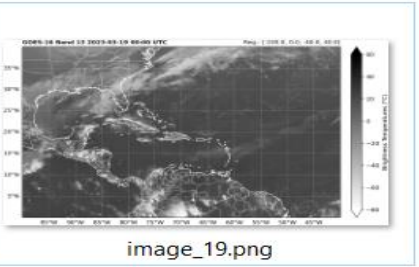


image_19.png

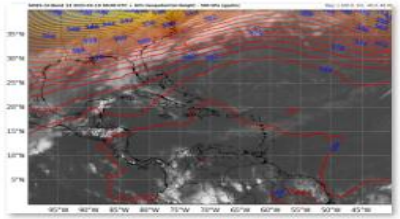


image_20.png

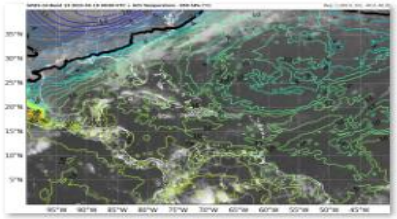


image_21.png

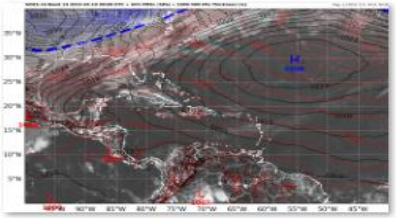


image_22.png

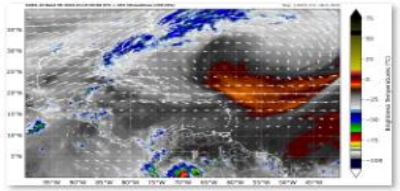


image_23.png

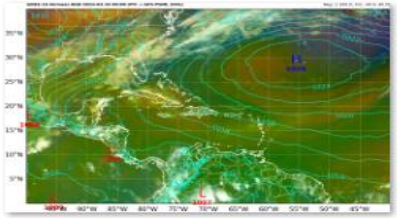


image_24.png


image_25.png


image_26.png

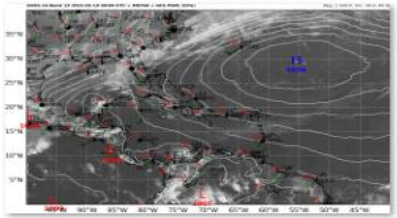


image_27.png

Arquivos

- Output
  - IR_202007301200.nc
  - IR_202107130000.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

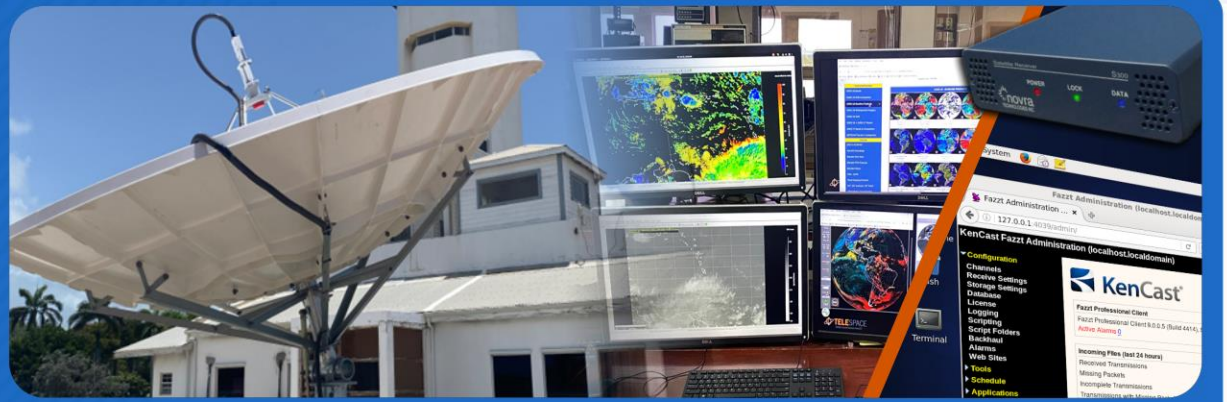
GEONETCast-Americas Training for the Eastern Caribbean States

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)