Weather Hazards Monitoring with GOES-R Data

SatPy is a versatile library for **reading**, **manipulating**, and **visualizing** data from weather satellites. In this section, we will explore more advanced capabilities of SatPy for processing weather satellite data, enabling us to work effectively with meteorological datasets.

Loading and Visualizing Satellite Data

In [ ]:
```
urls2dwn = [ 'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-
    'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-L1b-RadF-M6
    'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-L1b-RadF-M6
    'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-L1b-RadF-M6
    'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-L1b-RadF-M6
    'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-L1b-RadF-M6
    'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-L1b-RadF-M6
    'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-L1b-RadF-M6
    'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-L1b-RadF-M6
    'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-L1b-RadF-M6
    'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-L1b-RadF-M6
    'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-L1b-RadF-M6
    'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-L1b-RadF-M6
    'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-L1b-RadF-M6
    'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-L1b-RadF-M6
    'https://noaa-goes16.s3.amazonaws.com/ABI-L1b-RadF/2021/099/23/OR_ABI-L1b-RadF-M6
```

In [ ]:
```
import requests
import os

# Specify the local directory where you want to save the files.
# local_directory = input("Enter the path to the download folder: ")
local_directory = "Output_data/ABI-L1b-RadF/s20210992350171"
# Ensure that the local directory exists; create it if it doesn't.
os.makedirs(local_directory, exist_ok=True)

# Iterate through the URLs and download files.
for urld in urls2dwn:
    # Extract the filename from the URL.
    ntw = urld.split('/')[-1]

    # Construct the complete path to save the file in the local directory.
    file_path = os.path.join(local_directory, ntw)

    # Send an HTTP GET request to the URL.
    resp = requests.get(urld)

    # Check if the response is successful (status code 200).
    if resp.status_code == 200:
        # Write the content to the file in binary mode.
```

```
        # with open(file_path, "wb") as file:
        #     file.write(resp.content)
        print(f"File '{ntw}' downloaded and saved to '{local_directory}'.")
    else:
        print(f"Failed to download '{ntw}' from the URL: {urld}")
```

In [ ]:
```
# Import the warnings library, which allows control over warning messages.
import warnings

# Disable all warning messages to keep the output clean.
# This is useful in a lecture setting to avoid distracting students with non-critic
# that might arise from library functions used in processing geostationary satellit
warnings.filterwarnings('ignore')
```

In [ ]:
```
# Import necessary components from the satpy library, which is crucial for satellit
from satpy.scene import Scene  # Scene is a central object in Satpy used to represe

# The 'find_files_and_readers' function automates the discovery of satellite data f
# This simplifies loading and processing satellite imagery, which is essential for
from satpy import find_files_and_readers
```

Searching for GOES-R L1B Data

In [ ]:
```
# Import the os module to interact with the operating system and the glob module to
import os
import glob

# Define the base directory for satellite data relative to the current script's loc
# This is crucial for handling data files in a way that remains functional across d
base_dir = os.path.join('Output_data', 'ABI-L1b-RadF')

# Construct a pattern to find specific subdirectories under the base directory.
# This pattern targets directories starting with 's20210992350171', which could be
pattern = os.path.join(base_dir, 's20210992350171*')
directories = glob.glob(pattern)  # Use glob to search for directories that match t

# Select the first directory found that matches the pattern. This directory will be
# It's common in data processing workflows to automate the selection of data subset
directory = directories[0]

# Load the satellite data using Satpy's find_files_and_readers function, specifying
# 'abi_l1b' refers to the reader for Level 1b data from the ABI instrument, which i
fGRl1b = find_files_and_readers(base_dir=directory, reader='abi_l1b')

# Output the result to show what files and readers are being used. This helps in de
fGRl1b
```

Searching for GOES-R L2 CMIPC Data

In [ ]:
```
import os
import glob
```

```
# Define the base directory for Level 2 satellite data relative to the current scri
# This is pivotal for handling files in a way that ensures portability across diffe
base_dir = os.path.join('input_data', 'ABI-L2-CMIPF')

# Construct a pattern to find specific subdirectories under the base directory.
# Here, the pattern targets directories starting with 's20200621430', likely corres
pattern = os.path.join(base_dir, 's20200621430*')
directories = glob.glob(pattern)  # Use glob to search for directories that match t

# Select the first directory found that matches the pattern. This directory will be
# Automating the selection of specific data subsets simplifies the analysis process
directory = directories[0]

# Load the satellite data using Satpy's find_files_and_readers function, specifying
# 'abi_l2_nc' indicates a reader for Level 2 data from the ABI instrument, formatte
fGRl2 = find_files_and_readers(base_dir=directory, reader='abi_l2_nc')

# Output the result to show what files and readers are being used. This is useful f
fGRl2
```

In [ ]:
```
# Import the datetime module to handle date and time data.
# This is essential for processing time-stamped satellite data, allowing for precis
from datetime import datetime

# Import the glob function directly from the glob module.
# This function is used to find all the file paths that match a specific pattern, w
# Using glob allows for efficient and flexible file handling, especially when deali
from glob import glob
```

> **Attention:**
>
> **SatPy** always expects the original file names!
>
> So, do not change them when saving the data on your local machine.
> Otherwise, SatPy will not be able to open the files.

In [ ]:
```
# Create a Scene object named 'scn' by providing a list of filenames obtained from
# The Scene object is a core part of the Satpy library, which organizes and manages
scn = Scene(filenames=fGRl1b)

# Retrieve the names of all datasets available in the 'scn' object, which represent
# This information is crucial for understanding what types of data are available fo
dataset_names = scn.all_dataset_names()

# Output the list of dataset names. This is useful for educational purposes to show
# and to select specific datasets for further analysis in practical exercises.
print(dataset_names)
```

In [ ]:
```
# The magic command '%matplotlib inline' configures the Jupyter Notebook to display
# This setting is essential for interactive data visualization, especially when plo
%matplotlib inline

# Import the matplotlib.pyplot module under the alias 'plt'.
```

```
# Matplotlib is a comprehensive library for creating static, interactive, and anima
# It is particularly useful in satellite data analysis for plotting images, graphs,
import matplotlib.pyplot as plt
```

In [ ]:
```
# Load multiple datasets using a list comprehension to generate dataset names.
# List comprehensions provide a concise way to create lists based on existing lists
# In this case, we generate names for the datasets 'C01' to 'C16', which are typica
scn.load([f'C{x:02d}' for x in range(1, 17)])

# Explanation of the list comprehension:
# [f'C{x:02d}' for x in range(1, 17)] creates a list of strings from 'C01' to 'C16'
# 'f' before the string starts an f-string, allowing us to insert variables directl
# '{x:02d}' formats the number 'x' as a two-digit decimal, padding with zeros if ne

# The 'scn.load' function is then used to load these specific datasets into the Sce
# Loading multiple channels like this is common in the analysis of satellite imager
# where each channel can represent different spectral bands and contain different t
```

In [ ]:
```
# The method 'available_composite_names' is called on the 'scn' object.
# This method retrieves a list of all the composite images that can be created usin
# Composite images are made by combining multiple data channels to enhance the visu
# This feature is particularly useful in the study of geostationary satellites, as

# Retrieve and print the list of available composite names, providing a crucial ins
print(scn.available_composite_names())
```

In [ ]:
```
# Assign the dataset name 'airmass' to the variable 'rgb_im'.
# This variable naming provides clarity when referencing the dataset in multiple pl
# ensuring consistency and reducing the likelihood of errors in dataset identificat
rgb_im = 'airmass'

# Load the dataset named 'airmass' using the 'scn.load' method.
# The 'airmass' composite is particularly useful in meteorology as it combines seve
# making it easier to analyze atmospheric conditions from geostationary satellite d
scn.load([rgb_im])

# Display the loaded 'airmass' dataset using 'scn.show'.
# This method visualizes the specified dataset, allowing students to see the practi
# and understand their relevance in real-world atmospheric monitoring and analysis.
scn.show(rgb_im)
```

In [ ]:
```
# Access the dataset named 'airmass' from the 'scn' object using the previously def
# The variable 'rgb_im' holds the string 'airmass', which acts as a key to retrieve
# This operation is critical in satellite data processing as it allows for direct m
result = scn[rgb_im]

# The retrieved dataset can then be used for further analysis, visualization, or pr
# It's important for students to understand how to efficiently access and work with
result
```

In [ ]:
```
# Retrieve the keys (dataset names) available in the 'scn' object
# This function lists all dataset identifiers stored in the Scene object, each repr
keys = scn.keys()

# The output is a list of DataID objects, each encapsulating the metadata for a dif
```

```
# These DataIDs include crucial information such as:
# - name: The identifier of the dataset, like 'C01', 'C02', ..., 'C16', 'airmass'.
# - wavelength: A WavelengthRange object indicating the spectral range each channel
# - resolution: The spatial resolution of the data in meters, which affects the det
# - calibration: The type of calibration applied to the data, which could be reflec
# - modifiers: Any additional processing applied to the data channel.

# Print the keys to show the available datasets and their properties, aiding in und
keys
```

In [ ]:
```
# Access the area information associated with the 'C13' dataset in the 'scn' object
# The area property of a dataset within a Scene object provides geographical and ge
# This includes information such as the projection, extent, resolution, and size of

# For the 'C13' dataset, which typically represents an infrared channel on geostati
# understanding its area is essential for accurately interpreting spatial phenomena
area_info = scn["C13"].area

# Print the area information to provide insights into the spatial characteristics o
# This information supports tasks such as mapping, data integration with other geos
area_info
```

In [ ]:
```
# Access the area definition information for the 'C01' dataset in the 'scn' object.
# The 'area' property of a dataset provides detailed geographic and geometric infor
# This includes details such as projection type, coordinate reference system, image

# The 'C01' channel, which often captures data in a visible light wavelength (appro
# provides high-resolution imagery useful for detailed visual inspections of cloud
area_info = scn["C01"].area

# Print the area information to give insights into the geographic scope and detail
# This is crucial for applications such as mapping, tracking environmental changes,
area_info
```

In [ ]:
```
# Access the area definition information for the 'C02' dataset in the 'scn' object.
# The 'area' property of a dataset provides crucial geographic and geometric inform
# This includes the projection type, coordinate system, extent of the image, and pi

# The 'C02' channel is typically in the visible spectrum (centered around 0.64 µm),
# cloud formations, and atmospheric phenomena. This channel is instrumental in mete
area_info = scn["C02"].area

# Print the area information to provide insights into the geographic scope and reso
# Understanding this information is vital for accurate mapping, environmental monit
area_info
```

In [ ]:
```
# Load the "natural_color" dataset using the 'scn.load' method.
# The "natural_color" composite is a popular visual representation that combines mu
# to produce an image that approximates what the human eye would see from space.
# This is particularly useful in earth observation for visualizing land cover, wate

# The composite typically leverages red, green, and blue spectral bands to enhance
# which makes it ideal for presentations, educational purposes, and initial visual
scn.load(["natural_color"])
```

```
# This step is crucial for subsequent visualization and analysis tasks, as it prepa
```

In [ ]:
```
# Get the area definition of the "C13" dataset from the 'scn' object.
# The 'area' property contains critical geographic and geometric information, inclu
# which is essential for accurate geographic referencing in satellite imagery analy
rs = scn["C13"].area

# Resample the scene to the specified area definition.
# Resampling is a critical process in satellite data handling, allowing datasets to
# This standardization is necessary for accurate comparison and integration of diff

# Here, 'scn.resample(rs)' adjusts all data in the scene to match the area definiti
# This is particularly useful when preparing data for detailed analysis or visualiz
# ensuring consistency across different datasets within the same scene.
lscn = scn.resample(rs)

# The resulting 'lscn' is a new Scene object containing all the original data,
# but now aligned to the same geographic grid as the 'C13' dataset.
# This uniformity is crucial for subsequent processing steps, such as creating comp
```

In [ ]:
```
# Load the "natural_color" dataset for the resampled scene.
# The 'natural_color' composite is designed to mimic the colors visible to the huma
# Loading this dataset into the resampled scene ('lscn') ensures that the data is p

lscn.load(["natural_color"])

# Display the "natural_color" dataset.
# This step involves visualizing the loaded dataset using the 'show' method, which
# Visualizing data in this way is particularly useful for presentations and educati

lscn.show("natural_color")

# This visualization step is crucial for analyzing environmental and atmospheric co
# as it allows observers to easily identify and assess visible features without the
```

In [ ]:
```
# Crop the resampled scene (lscn) to a specific geographic region defined by latitu
# The bounds are given as a tuple in the format (lon_min, lat_min, lon_max, lat_max
# This operation is useful for focusing the analysis on a particular area of intere
scn_c1 = lscn.crop(ll_bbox=(-65.7, 10.7, -55.9, 20.1))

# Display the "C13" dataset from the cropped scene (scn_c1).
# The "C13" channel typically represents an infrared wavelength used for observing
# crucial for meteorological studies and weather forecasting.
# Displaying this dataset allows for detailed observation of atmospheric conditions
scn_c1.show("C13")

# Visualizing specific channels like "C13" in defined geographic regions helps in t
# such as monitoring storm development or evaluating climate patterns in detail.
# This capability is particularly valuable in educational settings for demonstratin
```

In [ ]:
```
# Load the "ash" dataset from the cropped scene (scn_c1).
# The "ash" composite is specifically designed to detect and visualize volcanic ash
# This dataset is crucial for monitoring volcanic activity and assessing the distri
# which can have significant impacts on air quality and aviation safety.
```

```
scn_c1.load(['ash'])

# Display the "ash" dataset from the cropped scene (scn_c1).
# Displaying this dataset allows for visual assessment of ash presence within the s
# The visualization is particularly useful in educational and operational settings
scn_c1.show('ash')

# This step not only aids in the educational demonstration of satellite capabilitie
# Such visualizations are essential tools in emergency response planning and enviro
```

In [ ]:

```
# Load the "so2" dataset from the cropped scene (scn_c1).
# The "so2" dataset is designed to detect sulfur dioxide (SO2) concentrations in th
# SO2 is a significant volcanic gas and industrial pollutant, making this dataset c
scn_c1.load(['so2'])

# Display the "so2" dataset from the cropped scene (scn_c1).
# Displaying this dataset allows for a visual assessment of SO2 distribution within
# The visualization helps in understanding the spatial extent and concentration of
scn_c1.show('so2')

# This step not only aids in the educational demonstration of how satellite imagery
# Such visualizations are valuable tools for researchers, policymakers, and educato
```

In [ ]:

```
# Crop the lscn (resampled scene) to a specific region defined by the latitude and
# The bounding box coordinates are specified as (lon_min, lat_min, lon_max, lat_max
# This operation focuses the analysis on a particular geographic area, which is ess
scn_c2 = lscn.crop(ll_bbox=(-95.0, 15.3, -90.8, 18.7))

# Show the "C13" dataset from the cropped scene (scn_c2).
# The "C13" channel typically represents an infrared wavelength useful for observin
# Displaying this dataset allows for detailed observation of atmospheric conditions
scn_c2.show("C13")

# This step not only provides visual feedback for the cropped area but also highlig
```

In [ ]:

```
# Load the "cira_fire_temperature" dataset into the cropped scene (scn_c2).
# The "cira_fire_temperature" dataset is specifically designed to detect and visual
# Loading this dataset is crucial for monitoring active fire areas and assessing th
scn_c2.load(['cira_fire_temperature'])

# Show the "cira_fire_temperature" dataset from the cropped scene (scn_c2).
# Displaying this dataset allows for a visual assessment of fire intensity and spre
# The visualization helps in understanding the spatial distribution of fires and ca
scn_c2.show('cira_fire_temperature')

# This visualization step is particularly valuable in educational settings to demon
# It also provides practical insights for emergency responders and environmental re
```

In [ ]:

```
# Load the "land_cloud_fire" dataset into the cropped scene (scn_c2).
# The "land_cloud_fire" composite is designed to provide a comprehensive view that
# This dataset is particularly useful for simultaneous monitoring of different envi
scn_c2.load(['land_cloud_fire'])

# Show the "land_cloud_fire" dataset from the cropped scene (scn_c2).
# Displaying this dataset allows for a visual assessment that integrates the visibi
```

```
# This type of visualization is crucial for understanding interactions between vari
scn_c2.show('land_cloud_fire')

# This step is highly beneficial in educational settings to demonstrate the versati
# It also provides practical insights for students and professionals in fields like
```