SatPy A Python Library for Weather Satellite Processing

SatPy - Part 3: Exploring The Geostationary Operational Environmental Satellite (GOES) – R Data

For detailed documentation, visit SatPy Documentation.

SatPy is a powerful library for **reading**, **manipulating**, and **displaying** data from remote sensors, primarily related to meteorology. It also provides the capability to **save** this data as images or in various formats.

SatPy excels at generating images with **individual channels or bands** and creating **RGB composites** directly from satellite instrument data.

The pyresample library is used for **resampling data** in different areas with specific projections or uniform grids.

Additionally, Satpy offers various **atmospheric corrections** and **visual enhancements**, either directly within Satpy or through the PySpectral and TrollImage packages.

The Geostationary Operational Environmental Satellite (GOES) – R Data

https://www.goes-r.gov/

Loading and Visualizing Satellite Data

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
from satpy.scene import Scene
from satpy import find_files_and_readers
```

Searching for GOES-R L1B Data

In [7]:
```
import os

# Get the absolute path of the current script or notebook's directory.
current_directory = os.getcwd()

# Construct the absolute path to the target directory by appending the specific rel
# to the current directory. This ensures the path is always correct regardless of w
```

```
base_dir = os.path.join(current_directory, "Input_data", "ABI-L1b-RadF", "s20210992
# Use the corrected path and reader to find files and readers
fGRl1b = find_files_and_readers(base_dir=base_dir, reader='abi_l1b')

# List the found files and associated readers
fGRl1b
```

Out[7]:

```
{'abi_l1b': ['c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub
\\Module_2\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C01_G16_s
20210992350171_e20210992359479_c20210992359525.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2
\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C02_G16_s2021099235
0171_e20210992359479_c20210992359522.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2
\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C03_G16_s2021099235
0171_e20210992359479_c20210992359530.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2
\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C04_G16_s2021099235
0171_e20210992359479_c20210992359514.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2
\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C05_G16_s2021099235
0171_e20210992359479_c20210992359533.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2
\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C06_G16_s2021099235
0171_e20210992359484_c20210992359528.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2
\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C07_G16_s2021099235
0171_e20210992359490_c20210992359539.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2
\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C08_G16_s2021099235
0171_e20210992359479_c20210992359535.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2
\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C09_G16_s2021099235
0171_e20210992359484_c20210992359551.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2
\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C10_G16_s2021099235
0171_e20210992359490_c20210992359536.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2
\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C11_G16_s2021099235
0171_e20210992359479_c20210992359545.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2
\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C12_G16_s2021099235
0171_e20210992359484_c20210992359539.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2
\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C13_G16_s2021099235
0171_e20210992359490_c20210992359555.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2
\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C14_G16_s2021099235
0171_e20210992359479_c20210992359547.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2
\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C15_G16_s2021099235
0171_e20210992359484_c20210992359557.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2
\\Input_data\\ABI-L1b-RadF\\s20210992350171\\OR_ABI-L1b-RadF-M6C16_G16_s2021099235
0171_e20210992359490_c20210992359542.nc']}
```

Searching for GOES-R L2 CMIPC Data

In [8]:
```python
import os

# Get the absolute path of the current script or notebook's directory.
current_directory = os.getcwd()

# Construct the absolute path to the target directory by appending the specific rel
# to the current directory. This ensures the path is always correct regardless of w
base_dir = os.path.join(current_directory, "Input_data", "ABI-L2-CMIPC", "s20180471


# Use the defined path and reader ('abi_l2_nc') to find files and associated reader
fGRl2 = find_files_and_readers(base_dir=base_dir, reader='abi_l2_nc')

# List the found files and associated readers
fGRl2
```

Out[8]:
{'abi_l2_nc': ['c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C01_G16_s20180471917196_e20180471919570_c20180471920048.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C02_G16_s20180471917196_e20180471919569_c20180471920080.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C03_G16_s20180471917196_e20180471919570_c20180471920045.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C04_G16_s20180471917196_e20180471919569_c20180471920036.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C05_G16_s20180471917196_e20180471919569_c20180471920050.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C06_G16_s20180471917196_e20180471919575_c20180471920036.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C07_G16_s20180471917196_e20180471919581_c20180471920040.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C08_G16_s20180471917196_e20180471919569_c20180471920033.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C09_G16_s20180471917196_e20180471919575_c20180471920022.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C10_G16_s20180471917196_e20180471919581_c20180471920040.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C11_G16_s20180471917196_e20180471919569_c20180471920038.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C12_G16_s20180471917196_e20180471919576_c20180471920042.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C13_G16_s20180471917196_e20180471919581_c20180471920028.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C14_G16_s20180471917196_e20180471919569_c20180471920034.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C15_G16_s20180471917196_e20180471919575_c20180471920042.nc',
  'c:\\Users\\moham\\OneDrive\\Desktop\\Stevens\\Projects\\FACT\\GitHub\\Module_2\\Input_data\\ABI-L2-CMIPC\\s20180471917\\OR_ABI-L2-CMIPC-M3C16_G16_s20180471917196_e20180471919581_c20180471920042.nc']}

In [9]:
```
from datetime import datetime
# Create a datetime object representing October 28, 2023, 3:30 PM
date_time = datetime(2023, 10, 28, 15, 30)
```

In [10]:
```
# Import the 'glob' function from the 'glob' module, which is used for file and dir
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 [11]:
```
# Create a Scene object 'scn' using a list of filenames 'fGRl1b'
scn = Scene(filenames=fGRl1b)
```

In [12]:
```
# Retrieve and display the attributes of the 'scn' Scene object
scn.attrs
```

Out[12]:
```
{}
```

Below code shows differentbands avaialble in the image file corresponding to different wavelengths

In [13]:
```
# Retrieve and list all the dataset names available within the 'scn' Scene object
scn.all_dataset_names()
```

Out[13]:
```
['C01',
 'C02',
 'C03',
 'C04',
 'C05',
 'C06',
 'C07',
 'C08',
 'C09',
 'C10',
 'C11',
 'C12',
 'C13',
 'C14',
 'C15',
 'C16']
```

In [14]:
```
# Retrieve and list the keys (dataset names) available within the 'scn' Scene objec
# These keys correspond to different bands or datasets. For more details about thes
# http://www.geo-web.org.uk/sats.php
scn.keys()
```

Out[14]:
```
[]
```

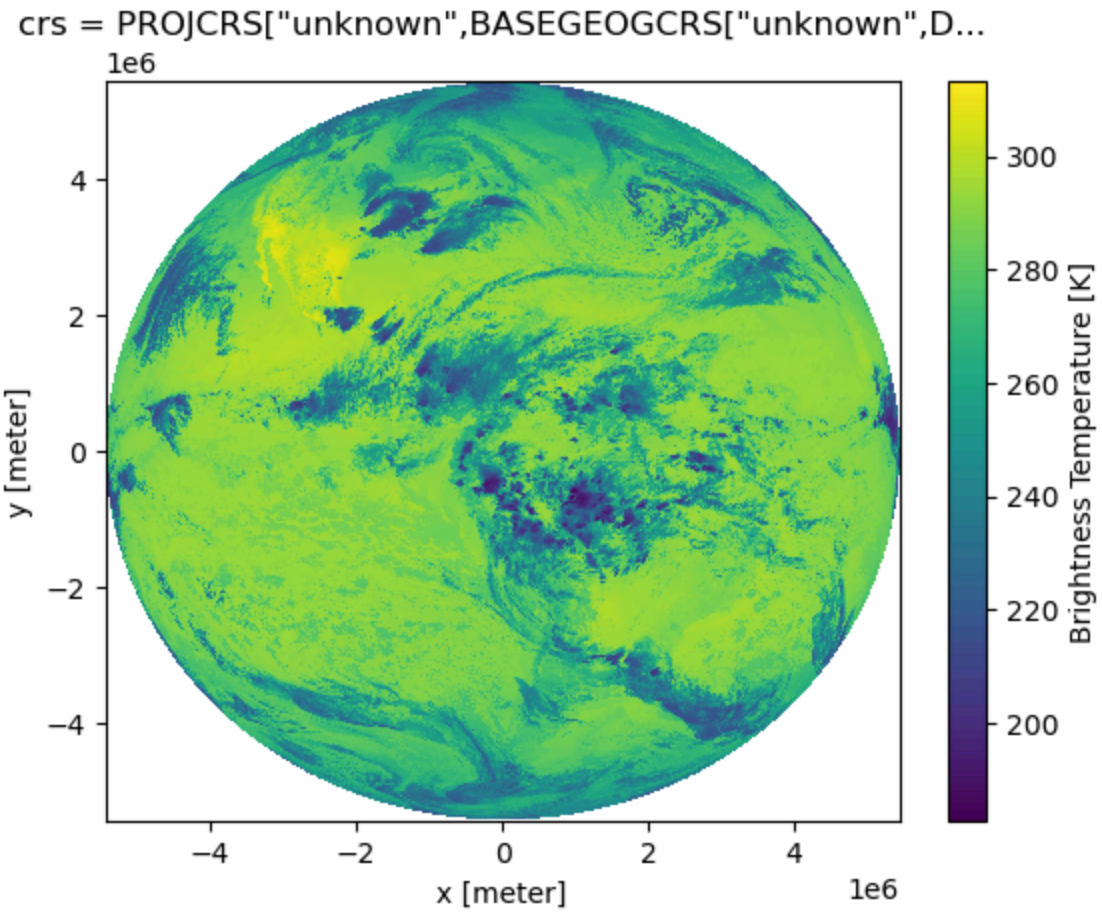
In [15]:
```
# Load the dataset named "C13" into the 'scn' Scene object.
scn.load(["C13"])
```

In [16]:
```
# Display the data from the "C13" dataset within the 'scn' Scene object.
scn.show("C13")
```

Out[16]:

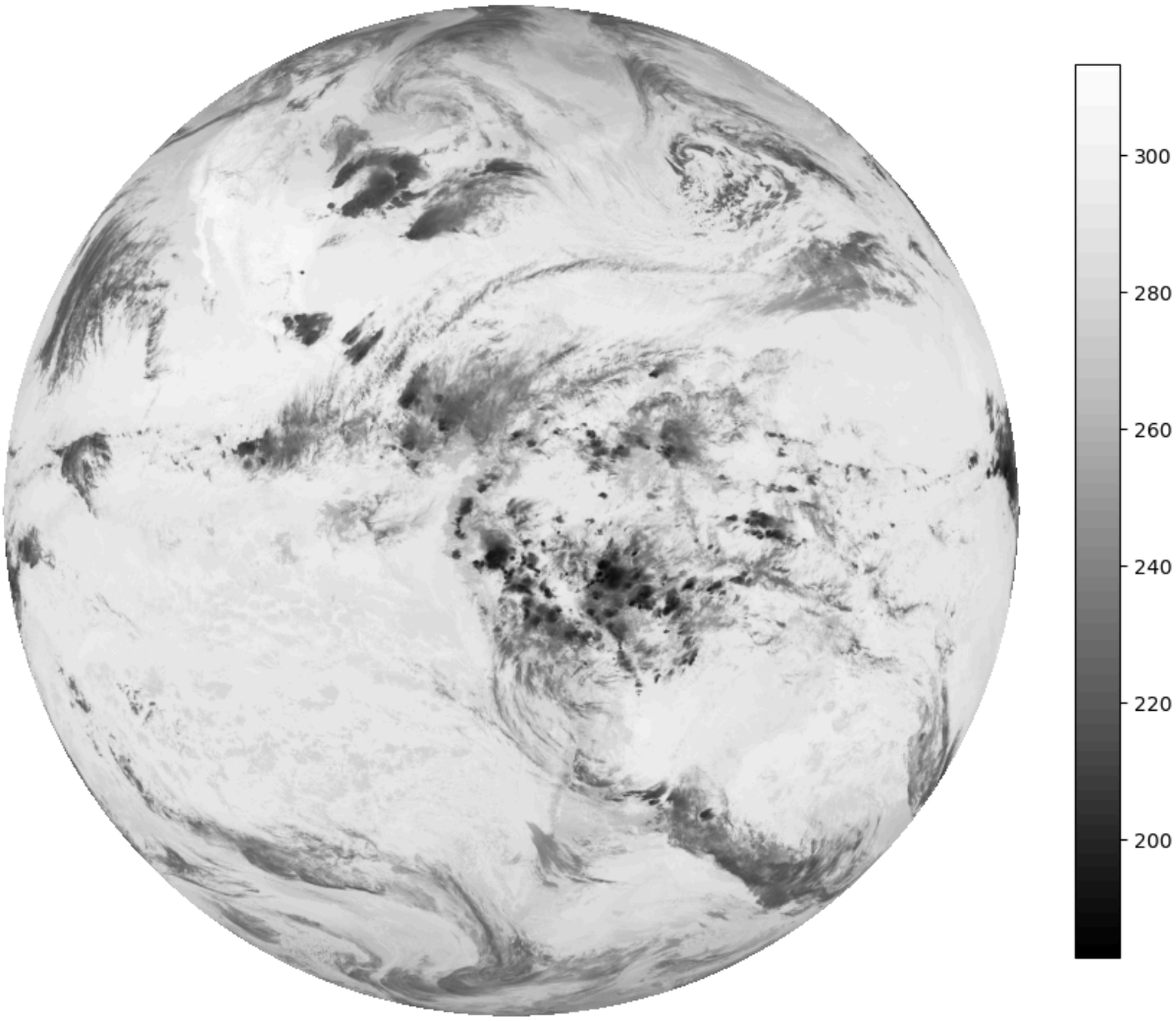
In [17]:
```
# Enable inline plotting using Matplotlib for the Jupyter Notebook (%matplotlib inl
%matplotlib inline

# Plot and display the "C13" dataset/band from the 'scn' Scene object as an image.
scn["C13"].plot.imshow()
```

Out[17]:
```
<matplotlib.image.AxesImage at 0x216e4373890>
```



In [18]:
```python
# Import the Matplotlib library as 'plt' for creating plots and visualizations
import matplotlib.pyplot as plt
```

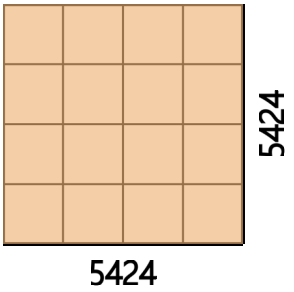
In [19]:
```python
# Create a Matplotlib figure and axis with a specified size (10x10 inches)
fig, ax = plt.subplots(figsize=(10, 10))

# Display the data from the "C13" dataset in grayscale ("Greys_r" colormap)
plt.imshow(scn["C13"].values, cmap="Greys_r")

# Turn off the axis labels and ticks
ax.set_axis_off()

# Add a colorbar to the plot with a specified fraction of the original size
plt.colorbar(fraction=0.04)

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



In [20]:
```
# Access and retrieve the data from the "C13" dataset within the 'scn' Scene object
scn["C13"]
```

Out[20]: xarray.DataArray 'Rad' (**y**: 5424, **x**: 5424)

	Array	Chunk
Bytes	112.23 MiB	7.01 MiB
Shape	(5424, 5424)	(1356, 1356)
Dask graph	16 chunks in 14 graph layers	
Data type	float32 numpy.ndarray	



▼ Coordinates:

crs	()	object	PROJCRS["unknown",BASEGEOGCRS["u...	
y	(y)	float64	5.434e+06 5.432e+06 ... -5.434e+06	
x	(x)	float64	-5.434e+06 -5.432e+06 ... 5.434e+06	

► Indexes: (2)

► Attributes: (29)

In [21]:
```
# Load the band at 10.3µm as radiances (in mW m-2 sr-1(cm-1)-1):
scn.load([10.3], calibration=["radiance"])

# Load the band at 10.3µm as brightness temperatures (in Kelvin, K):
scn.load([10.3], calibration=["brightness_temperature"])
```

In [22]:
```
# Retrieve and list the keys (dataset names) available within the 'scn' Scene objec
scn.keys()
```

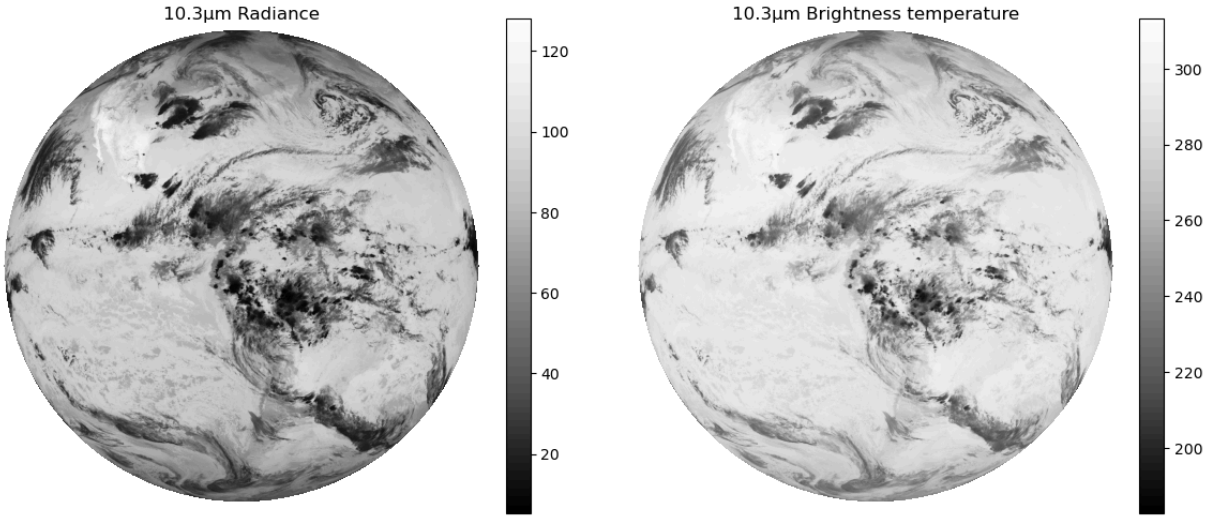
Out[22]:
```
[DataID(name='C13', wavelength=WavelengthRange(min=10.1, central=10.35, max=10.6,
unit='µm'), resolution=2000, calibration=<2>, modifiers=()),
 DataID(name='C13', wavelength=WavelengthRange(min=10.1, central=10.35, max=10.6,
unit='µm'), resolution=2000, calibration=<3>, modifiers=())]
```

In [23]:
```
# Retrieve the wavelength and calibration information for the second dataset (index
# The [1] index refers to the second dataset, and ['wavelength'] and ['calibration'
scn.keys()[1]['wavelength'][1], scn.keys()[1]['calibration']
```

Out[23]:
```
(10.35, <3>)
```

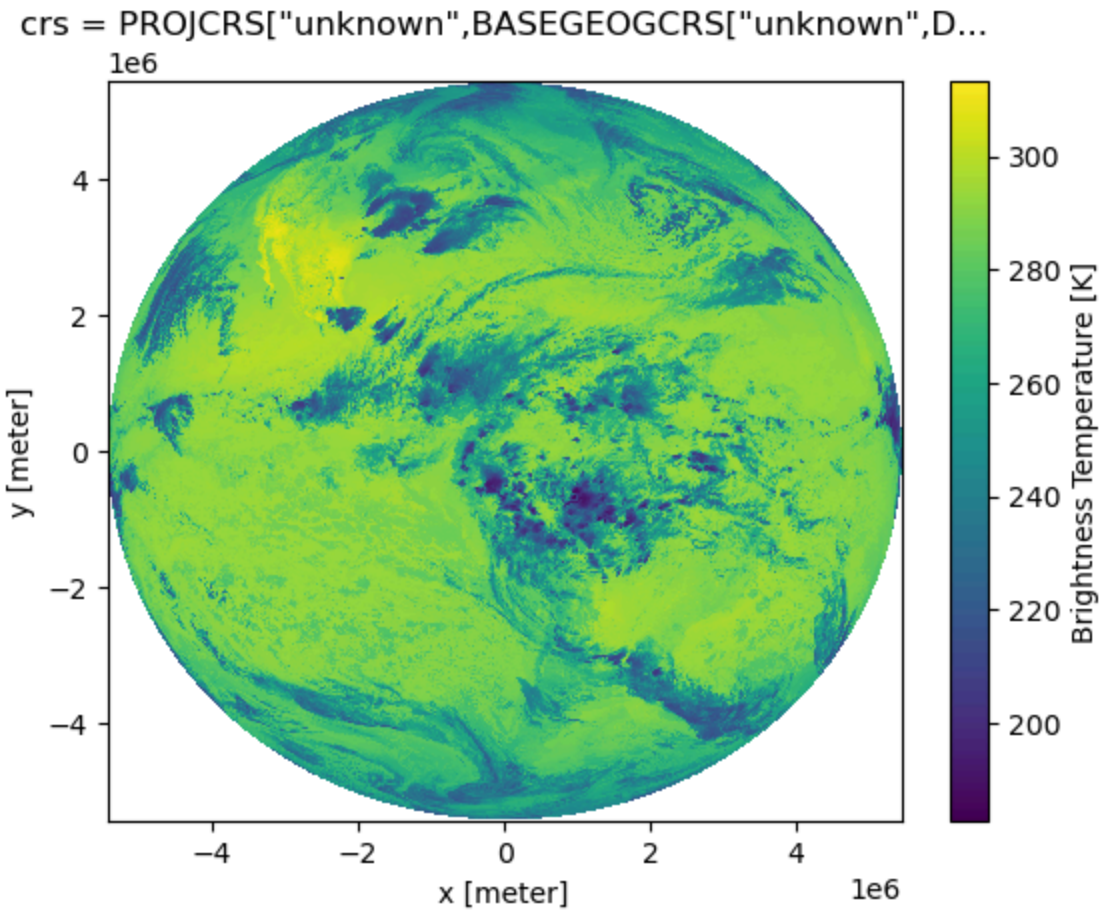
In [24]:
```
# Create a Matplotlib figure with two subplots side by side (1 row, 2 columns), spe
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

# Display the data from the second dataset (index 1) as an image in the first subpl
im1 = ax1.imshow(scn[scn.keys()[1]].values, cmap="Greys_r")
ax1.set_axis_off()  # Turn off axis labels and ticks for the first subplot

# Display the data from the first dataset (index 0) as an image in the second subpl
im2 = ax2.imshow(scn[scn.keys()[0]].values, cmap="Greys_r")
ax2.set_axis_off()  # Turn off axis labels and ticks for the second subplot
```

```
# Set titles for the subplots to indicate the type of data displayed.
ax2.set_title("10.3µm Brightness temperature")
ax1.set_title("10.3µm Radiance")

# Add colorbars to the subplots with a specified fraction of the original size.
fig.colorbar(im1, ax=ax1, fraction=0.05)
fig.colorbar(im2, ax=ax2, fraction=0.05)

# Show the figure with the two subplots.
plt.show()
```



Question: How can you plot or access the data for the 'C13' dataset?"

In [25]:
```
# Solution:
# To plot the 'C13' dataset:
scn['C13'].plot.imshow()

# To access the data values for the 'C13' dataset:
c13_data = scn['C13'].values
```



In [26]:
```
# Retrieve the 'x' coordinate information for the 'C13' dataset.
x_coordinates = scn["C13"].x
```

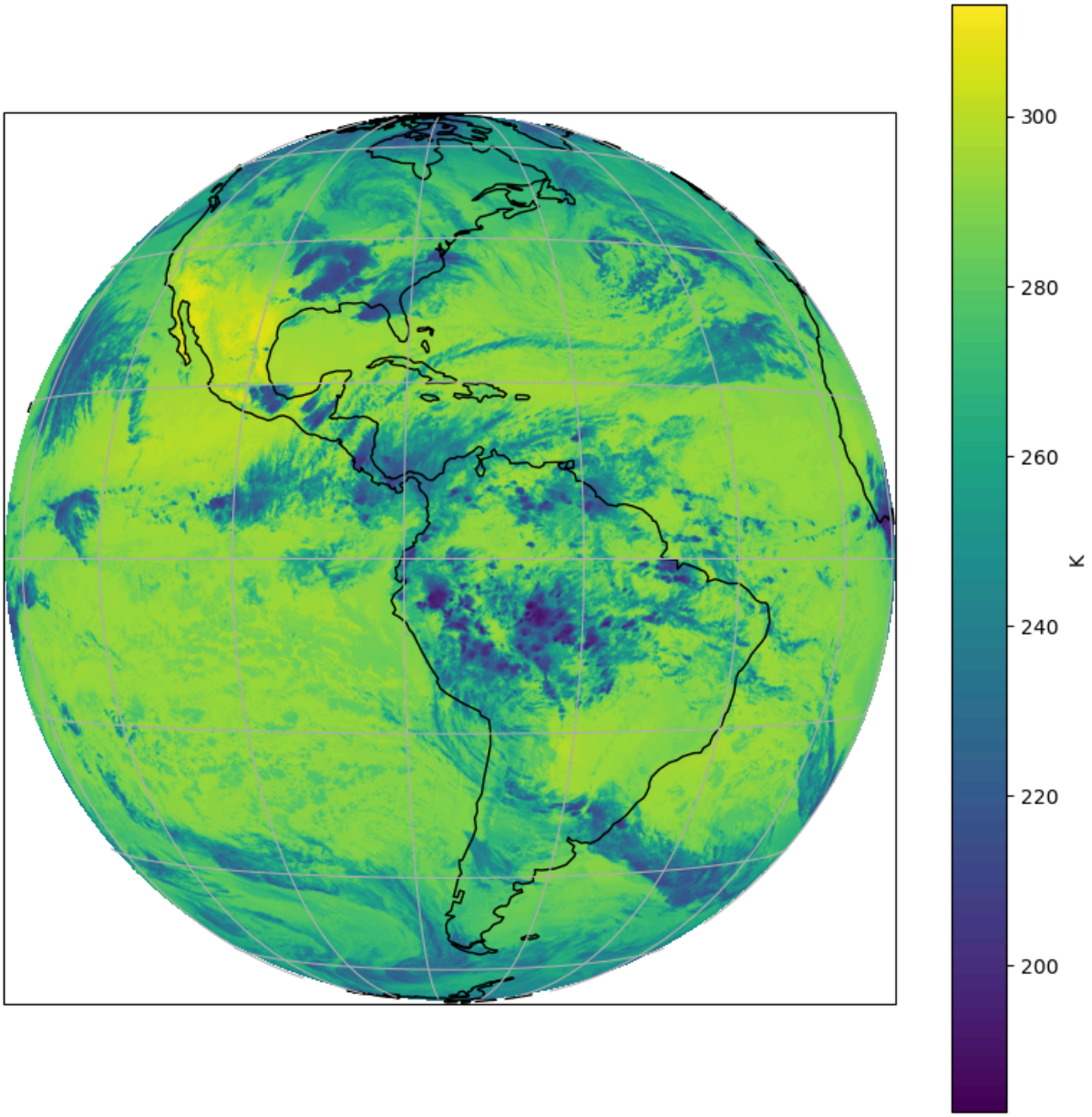
In [27]:
```
# define palette (matplotlib style)
cmap = ['#ffffff', '#ffffff', '#ffffff', '#ffffff', '#ffffff', '#b6ffb6', '#79ff79'
        '#0028a2', '#000079', '#fbfb00', '#e7e700', '#d2d200', '#baba00', '#a6a600'
        '#aaaaaa', '#a6a6a6', '#9e9e9e', '#969696', '#8e8e8e', '#868686', '#7d7d7d'
        '#313131', '#282828', '#202020', '#181818', '#141414', '#000000', '#000000'
```

In [28]:
```
# Retrieve the area definition (spatial information) associated with the 'C13' data
area_def = scn['C13'].attrs['area']
```

In [29]:
```
# Convert the area definition to a Cartopy Coordinate Reference System (CRS).
cartopy_crs = area_def.to_cartopy_crs()
```

In [30]:
```
# Display the results of the 'C13' dataset on a map using Cartopy and Matplotlib.
# Convert the area definition to a Cartopy CRS for proper georeferencing.
crs = area_def.to_cartopy_crs()

# Create a Matplotlib figure with a specified size.
fig = plt.figure(figsize=(10, 10))

# Create a map axis using Cartopy with coastlines and gridlines.
ax = plt.axes(projection=crs)
ax.coastlines()
ax.gridlines()
```

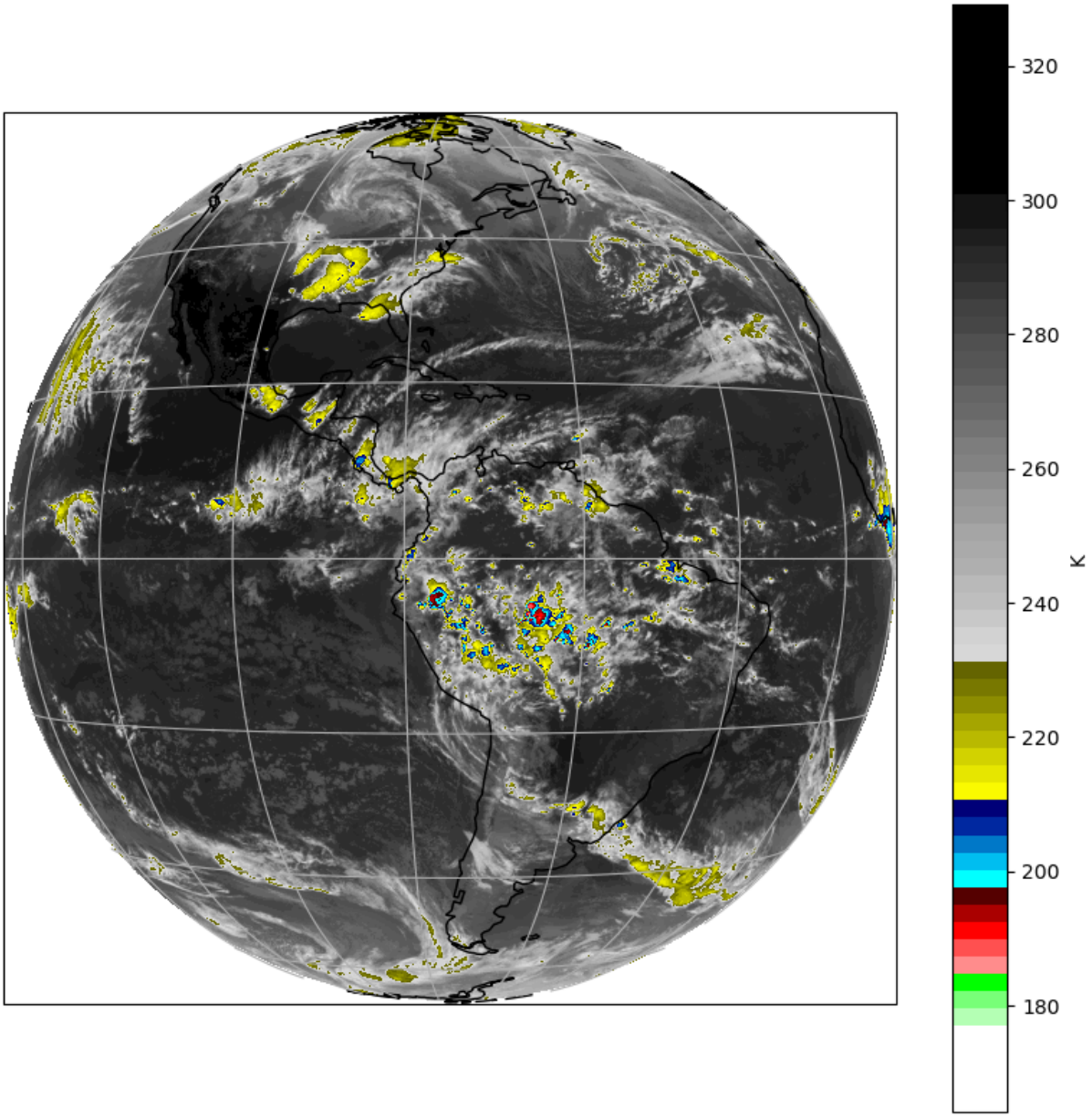
```
# Display the 'C13' dataset on the map with the correct georeferencing.
plt.imshow(scn['C13'], transform=crs, extent=crs.bounds, origin='upper')

# Add a colorbar to the plot with units information from the dataset's attributes.
plt.colorbar(label=scn['C13'].attrs['units'])

# Show the map with the dataset.
plt.show()
```



In [31]:
```
import numpy as np
```

In [32]:
```
# Define custom color levels and colormap for visualization.
# This code sets up custom levels, a normalization method, and a colormap for the v
levels = np.linspace(-109, 56, num=len(cmap))  # Define custom levels for color map
norm = plt.cm.colors.BoundaryNorm(levels, len(levels))  # Normalize data based on l
irmap = plt.cm.colors.ListedColormap(cmap)  # Create a custom colormap.
```

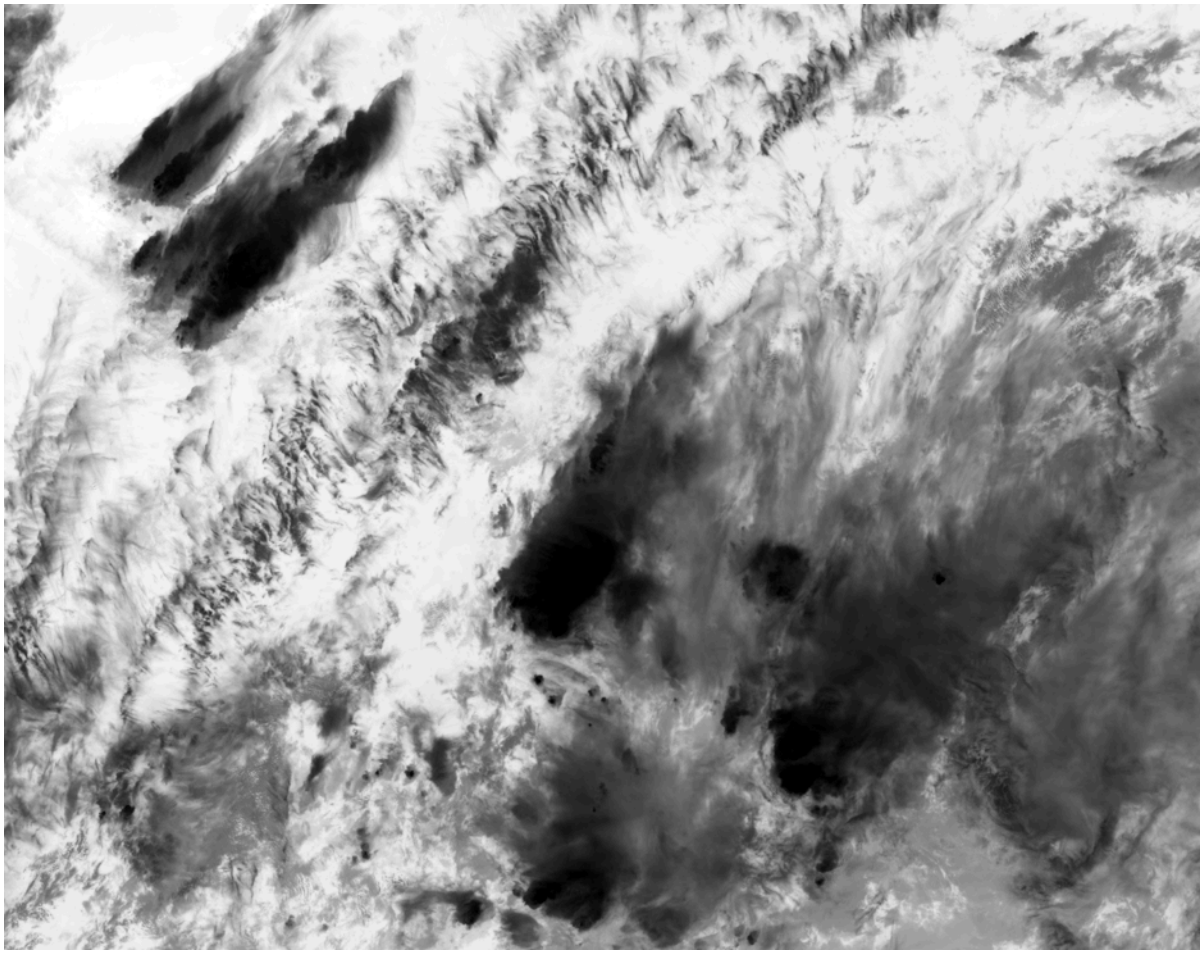
```
# Display the results of the 'C13' dataset on a map using Cartopy and Matplotlib.
# Convert the area definition to a Cartopy CRS for proper georeferencing.
crs = area_def.to_cartopy_crs()

# Create a Matplotlib figure with a specified size.
fig = plt.figure(figsize=(10, 10))

# Create a map axis using Cartopy with coastlines and gridlines.
ax = plt.axes(projection=crs)
ax.coastlines()
ax.gridlines()

# Display the 'C13' dataset on the map with the correct georeferencing and custom c
# Set the minimum and maximum values for color mapping using 'vmin' and 'vmax'.
plt.imshow(scn['C13'], transform=crs, extent=crs.bounds, origin='upper',
           vmin=-109 + 273.15, vmax=56 + 273.15, cmap=irmap)

# Add a colorbar to the plot with units information from the dataset's attributes.
plt.colorbar(label=scn['C13'].attrs['units'])

# Show the map with the dataset.
plt.show()
```



Cropping

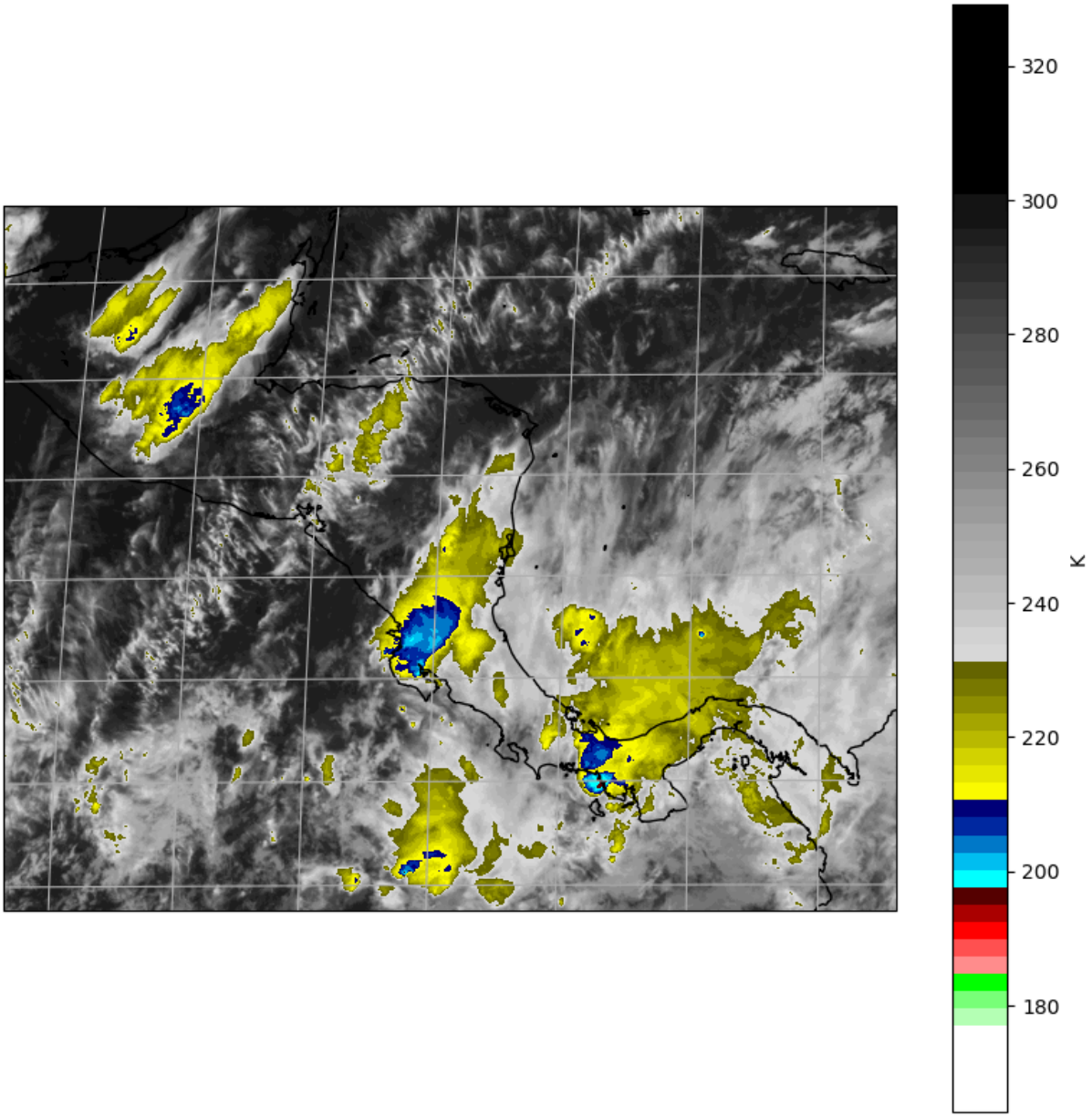
In [33]:
```
# Crop the scene to a specific region defined by latitude and longitude values.
# The bounding box is defined as (lon_min, lat_min, lon_max, lat_max).
# In this case, the region spans from longitude -93.5° to -76.0° and from latitude

# Crop the scene to the specified bounding box.
scn_cropped = scn.crop(ll_bbox=(-93.5, 5.5, -76.0, 19.5))

# Display the cropped dataset for the 'C13' channel.
scn_cropped.show("C13")
```

Out[33]:



In [34]:
```
# Retrieve the area definition (spatial information) associated with the 'C13' data
area_def_c = scn_cropped['C13'].attrs['area']
```

In [35]:
```
# Display the results of the 'C13' dataset within the cropped region using Cartopy
# Convert the area definition of the cropped scene to a Cartopy CRS for proper geor
crs = area_def_c.to_cartopy_crs()

# Create a Matplotlib figure with a specified size.
fig = plt.figure(figsize=(10, 10))

# Create a map axis using Cartopy with coastlines and gridlines.
ax = plt.axes(projection=crs)
ax.coastlines()
ax.gridlines()

# Display the 'C13' dataset within the cropped region on the map.
# Set the minimum and maximum values for color mapping using 'vmin' and 'vmax'.
# Apply the custom colormap 'irmap' defined earlier.
plt.imshow(scn_cropped['C13'], transform=crs, extent=crs.bounds, origin='upper',
           vmin=-109 + 273.15, vmax=56 + 273.15, cmap=irmap)

# Add a colorbar to the plot with units information from the dataset's attributes.
plt.colorbar(label=scn_cropped['C13'].attrs['units'])

# Show the map with the 'C13' dataset within the cropped region.
plt.show()
```


320
300
280
260
240
220
200
180
K