EDS 220: Data Wrangling with Rasters and False Color Imagery

Author: Henry Oliver

Github Repository: https://github.com/ht-oliver/eds220-hwk4

The purpose of this assignment to utilize false color satellite imagery to investigate the extent of wildfires in California in January, 2025. This anlaysis will walk through the steps necessary to display Landsay 8 satellite imagery overlayed with estimated perimeters of the 2025 LA County Palisades and Eaton Fires.

Background

The Palisades and Eaton fires burned across parts of Los Angeles County, leaving visible scars on the landscape. Using Landsat satellite imagery, we can highlight burn areas, compare pre- and post-fire conditions, and better understand the extent and distribution of damage. Remote sensing provides an objective, large-scale view that complements on-the-ground assessments and helps support recovery planning and ecological monitoring.

Highlights

- Wrangling and displaying `.shp` , file types `.nc`
- Producing True and False color imagery
- Creating useful, intuitive, and accurate visualizations

Part 1: Installing libraries

Performing the data download, manipulation, and displays in this analysis requires the installation of several publicly-available software packages.

In [111…
```
import os                         # file and path handling
import pandas as pd               # tabular data analysis
import matplotlib.pyplot as plt   # plotting and visualization
import xarray as xr               # working with labeled multi-dimensional data (e.g., rasters)
import rioxarray as rio           # geospatial raster I/O and spatial operations
import netCDF4                    # reading NetCDF datasets
import geopandas as gpd           # vector geospatial data (shapefiles, geodataframes)
import numpy as np                # numerical operations and arrays
```

Part 2: Fire Perimeter data

Shapefiles for the outline of LA County fires are provided by LA County. This step contains a method to download, join, project, and display fire perimeter data. Make sure to carefully read comments for reasoning behind each code chunk.

Below are the data sources for each perimeter shapefile.

Eaton Fire

- **File Name**: Eaton_Perimeter_20250121.shp
- **Source**: https://egis-lacounty.hub.arcgis.com/datasets/ lacounty::palisades-and-eaton-dissolved-fire-perimeters-2025/explore?layer=0
- **Publisher**: County of Los Angeles
- **Date**: February 26, 2025

Palisades Fire

- **File Name**: Palisades_Perimeter_20250121.shp
- **Source**: https://egis-lacounty.hub.arcgis.com/datasets/lacounty::palisades-and-eaton-dissolved-fire-perimeters-2025/explore?layer=1&location=34.133066%2C-118.349606%2C9.60
- **Publisher**: County of Los Angeles
- **Date**: February 26, 2025

In [112…
```
# Download Eaton shapefile
eaton = gpd.read_file(os.path.join('data',
                                   'Eaton_Perimeter_20250121',
                                   'Eaton_Perimeter_20250121.shp'))

# Download Palisades shapefile
palisades = gpd.read_file(os.path.join('data',
                                   'Palisades_Perimeter_20250121',
                                   'Palisades_Perimeter_20250121.shp'))
```

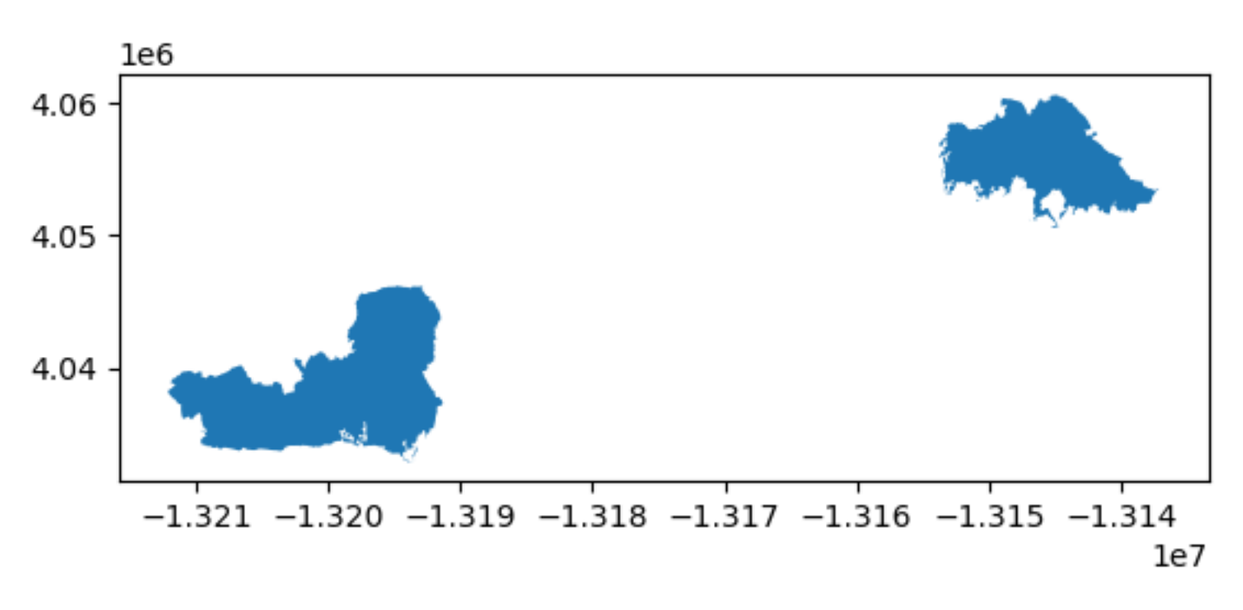
We want to join these two files. In order to do so we must confirm they have identical Coordinate Reference Systems (CRS), and are both projected.

In [113…
```
# Check CRS
print(f"Eaton CRS is: {eaton.crs}")
print(f"Palisades CRS is: {palisades.crs}")
assert eaton.crs == palisades.crs # Returns error if CRS don't match

# Check if projected
print(f"Eaton CRS is projected. {eaton.crs.is_projected}")
print(f"Palisades CRS is projected. {palisades.crs.is_projected}")
assert (eaton.crs.is_projected and palisades.crs.is_projected) == True # Returns error if not projected
```

```
Eaton CRS is: EPSG:3857
Palisades CRS is: EPSG:3857
Eaton CRS is projected. True
Palisades CRS is projected. True
```

In [114…
```
# Combine eaton and palisades shapefiles
fires = gpd.GeoDataFrame(pd.concat([eaton, palisades]))

# Confirm successful combination by plotting
fires.plot()
```

Out[114… `<Axes: >`



Part 3: NetCDF data import and exploration

Our Landsat 8 data is in NetCDF format, so we'll have to take specific steps to read it in correctly.

Below is the data source for Landsat 8 imagery

Landsat 8 Imagery of LA County

- **File Name**: 'landsat8-2025-02-23-palisades-eaton.nc'
- **Source**: https://planetarycomputer.microsoft.com/dataset/landsat-c2-l2
- **Publisher**: Microsoft Planetary Computer
- **Date**: February 26, 2025

In [115…
```
# Create filepath
path = ('data/landsat8-2025-02-23-palisades-eaton.nc')

# Read in landsat netCDF4 data package with xarray
landsat = xr.open_dataset(path)

# Print data type
print(f"landsat data type is: {type(landsat)}")

# View bands, dimensions & coordinates
landsat
```

```
landsat data type is: <class 'xarray.core.dataset.Dataset'>
```

Out[115… xarray.Dataset

▶ Dimensions: (**y**: 1418, **x**: 2742)

▼ Coordinates:

y	(y)	float64	3.799e+06 3.799e+06 ... 3.757e+06
x	(x)	float64	3.344e+05 3.344e+05 ... 4.166e+05
time	()	datetime64[ns]	...

▼ Data variables:

red	(y, x)	float32	...
green	(y, x)	float32	...
blue	(y, x)	float32	...
nir08	(y, x)	float32	...
swir22	(y, x)	float32	...
spatial_ref	()	int64	...

▶ Indexes: (2)

▶ Attributes: (0)

By printing `landsat` I was able to access the number of bands, the dimensions, and the CRS for x and y coordinate. The CRS for the x and y coordinates is EPSG:32611, and it is projected. I'm also able to tell that this data has 5 spectral bands including: red, green, nir08, and swir22. I was also able to see that the units of the CRS are meters, and the resolution is 30x30 meters.

Part 4: Restoring geospatial information

Right now, our Landsat ratio does not have an assigned CRS, though it does have a reference CRS. We need to fix that so our data can be displayed with our fire perimeters.

In [116…
```
# Print CRS of landsat data
print(landsat.rio.crs)
```

```
None
```

a) Is this a geospatial object? **It's almost a geospatial object, but it doesn't have a CRS assigned right now so technically it is not**

b) Print the CRS by using accesing the `spatial_ref.crs_wkt` attribute of the dataset

In [117…
```
# Print CRS
print(landsat.spatial_ref.crs_wkt)
```

```
PROJCS["WGS 84 / UTM zone 11N",GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS 84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","890
1"]],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-117],PARAMET
ER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Northing",NORTH],AUTHORITY["EPSG","3261
1"]]
```

c) Recover the geospatial information by using rio.srite_crs() and the spatial reference information from part b.

In [118…
```
# Write CRS to WGS 84 / UTM zone 11N
landsat.rio.write_crs(["EPSG", "32611"], inplace=True)

# Print
print(landsat.rio.crs)
```

```
EPSG:32611
```

Part 5: True color image

'True Color Images' display the 'red', 'blue' and 'green' light as detected by the satellite, which is similar to the way the image would be interpreted by our eyes. In order to display this information, we'll have to specify why pieces of the data we want to plot, and how we want to plot it. This won't work on the first try, but following the instructions below will help the process make sense.

a) Without creating any new variables:

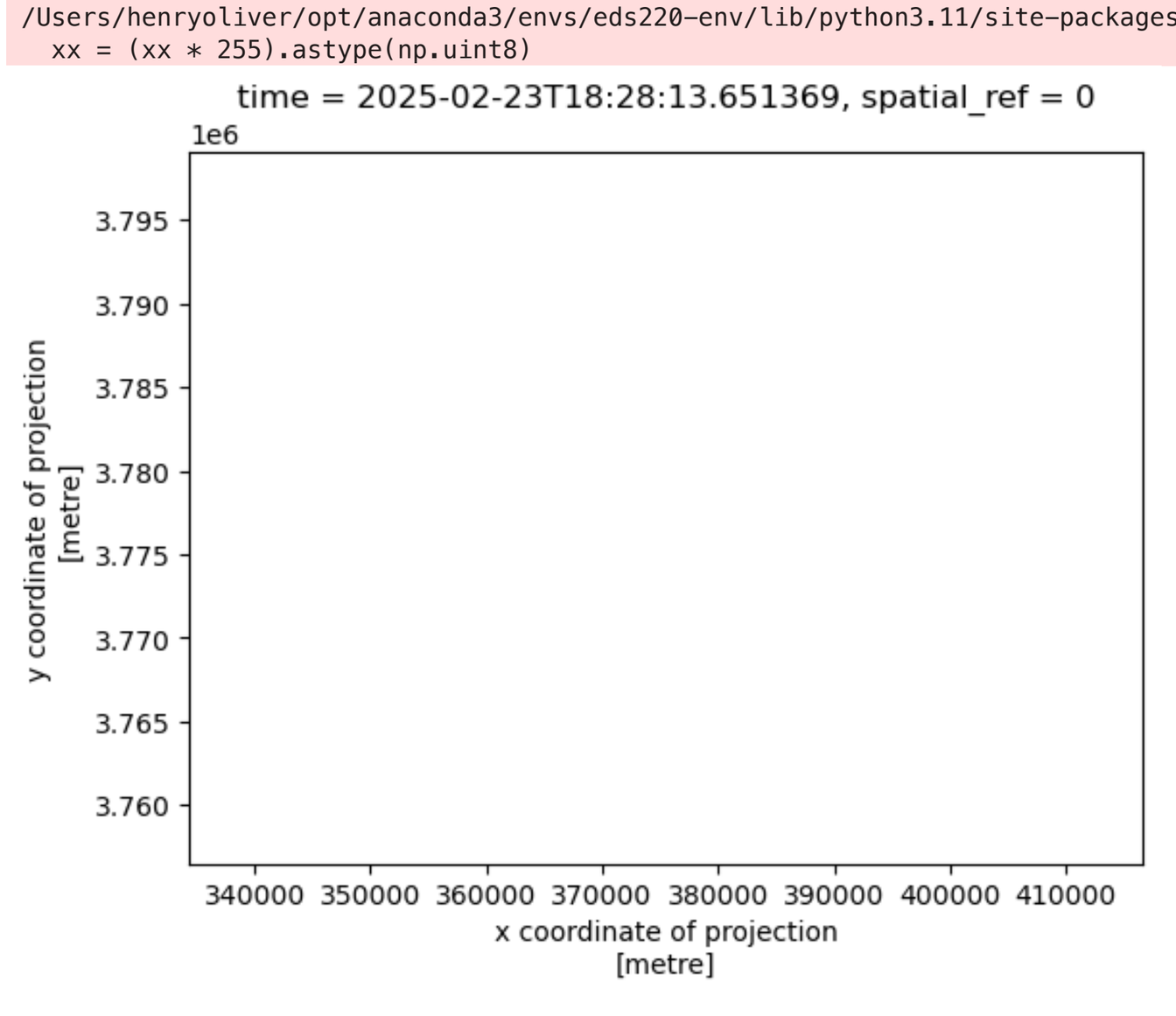
- select the red, green and blue variables (in that order) of the `xarray.Dataset` holding the Landsat data,
- convert it to a `numpy.array` using the `to_array()` method, and then
- use `.plot.imshow()` to create an RGB image with the data. There will be two warnings, that's ok

In [119…
```
# Attempt to print true color image
landsat[['red', 'green', 'blue']].to_array().plot.imshow()
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

Out[119… `<matplotlib.image.AxesImage at 0x366cb3990>`

```
/Users/henryoliver/opt/anaconda3/envs/eds220-env/lib/python3.11/site-packages/matplotlib/cm.py:478: RuntimeWarning: invalid value encountered in cast
  xx = (xx * 255).astype(np.uint8)
```

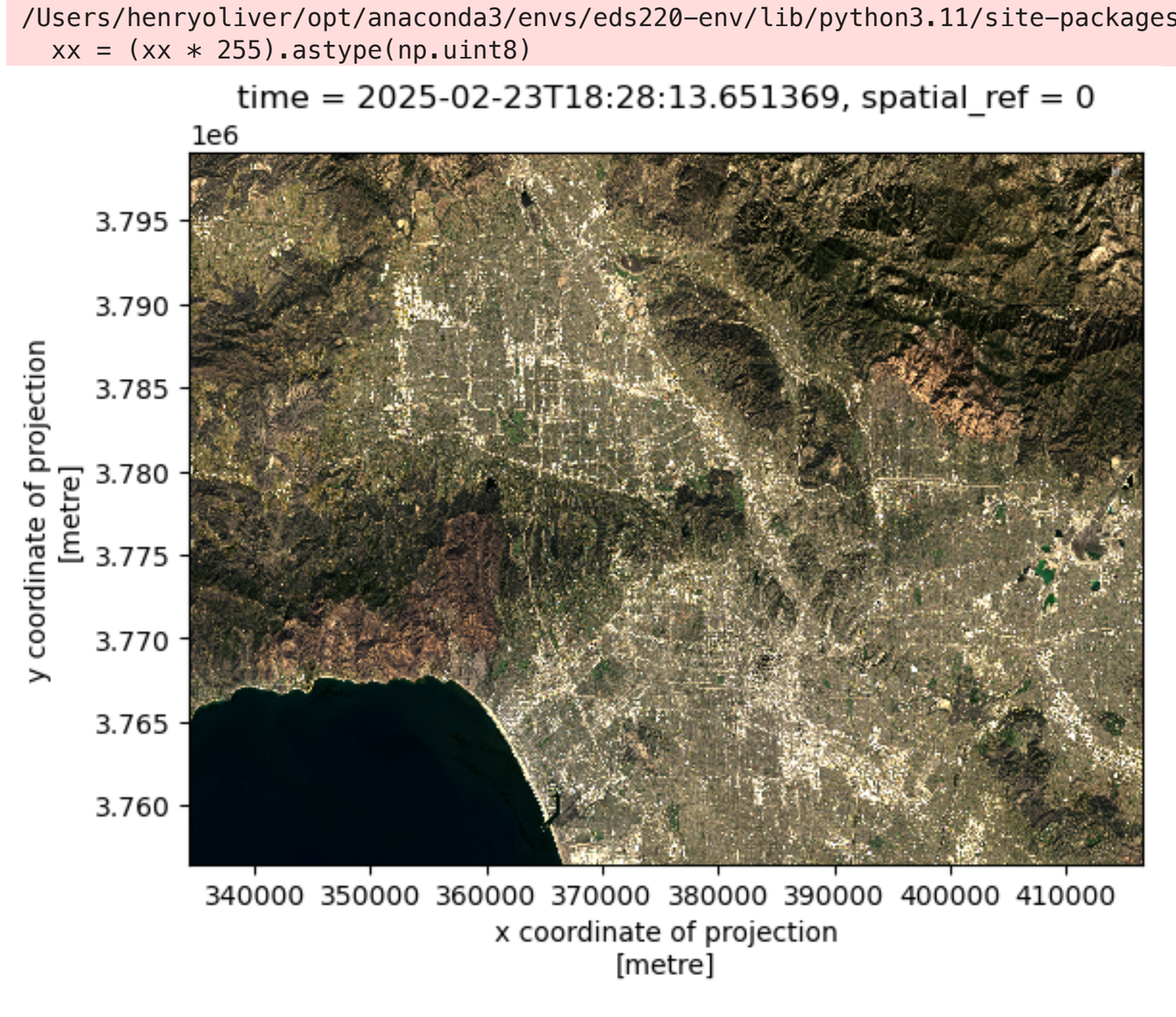


b) Adjust the scale used for plotting the bands to get a true color image. HINT: Check the robust parameter. The issue here is the clouds: their RGB values are outliers and cause the other values to be squished when plotting.

In [120…
```
# Print true color image
landsat[['red', 'green', 'blue']].to_array().plot.imshow(robust=True)
```

Out[120… `<matplotlib.image.AxesImage at 0x366c61610>`

```
/Users/henryoliver/opt/anaconda3/envs/eds220-env/lib/python3.11/site-packages/matplotlib/cm.py:478: RuntimeWarning: invalid value encountered in cast
  xx = (xx * 255).astype(np.uint8)
```



c) To resolve the other warning, identify which bands have `nan` values. HINT: There are many ways of doing so. ne option is to use the `numpy.isnan()`.

In [121…
```
# Access bands with NA values
np.isnan(landsat)
```

Out[121… xarray.Dataset

▶ Dimensions: (**y**: 1418, **x**: 2742)

▼ Coordinates:

y	(y)	float64	3.799e+06 3.799e+06 ... 3.757e+06
x	(x)	float64	3.344e+05 3.344e+05 ... 4.166e+05
time	()	datetime64[ns]	2025-02-23T18:28:13.651369
spatial_ref	()	int64	0

▼ Data variables:

red	(y, x)	bool	False False False ... False False
green	(y, x)	bool	False False False ... False False
blue	(y, x)	bool	False False False ... False False
nir08	(y, x)	bool	False False False ... False False
swir22	(y, x)	bool	False False False ... False False

▶ Indexes: (2)

▶ Attributes: (0)

d) use `.fillna()` method for `xarray.Dataset` to substitute the any `nan` values in the Landsat data for zero

In [122…
```
landsat.fillna(0)
```

Out[122… xarray.Dataset

▶ Dimensions: (**y**: 1418, **x**: 2742)

▼ Coordinates:

y	(y)	float64	3.799e+06 3.799e+06 ... 3.757e+06
x	(x)	float64	3.344e+05 3.344e+05 ... 4.166e+05
time	()	datetime64[ns]	2025-02-23T18:28:13.651369
spatial_ref	()	int64	0

▼ Data variables:

red	(y, x)	float32	1.024e+04 9.886e+03 ... 9.967e+03
green	(y, x)	float32	9.93e+03 9.687e+03 ... 9.662e+03
blue	(y, x)	float32	9.29e+03 9.183e+03 ... 9.187e+03
nir08	(y, x)	float32	1.331e+04 1.313e+04 ... 1.306e+04
swir22	(y, x)	float32	1.43e+04 1.437e+04 ... 1.329e+04

▶ Indexes: (2)

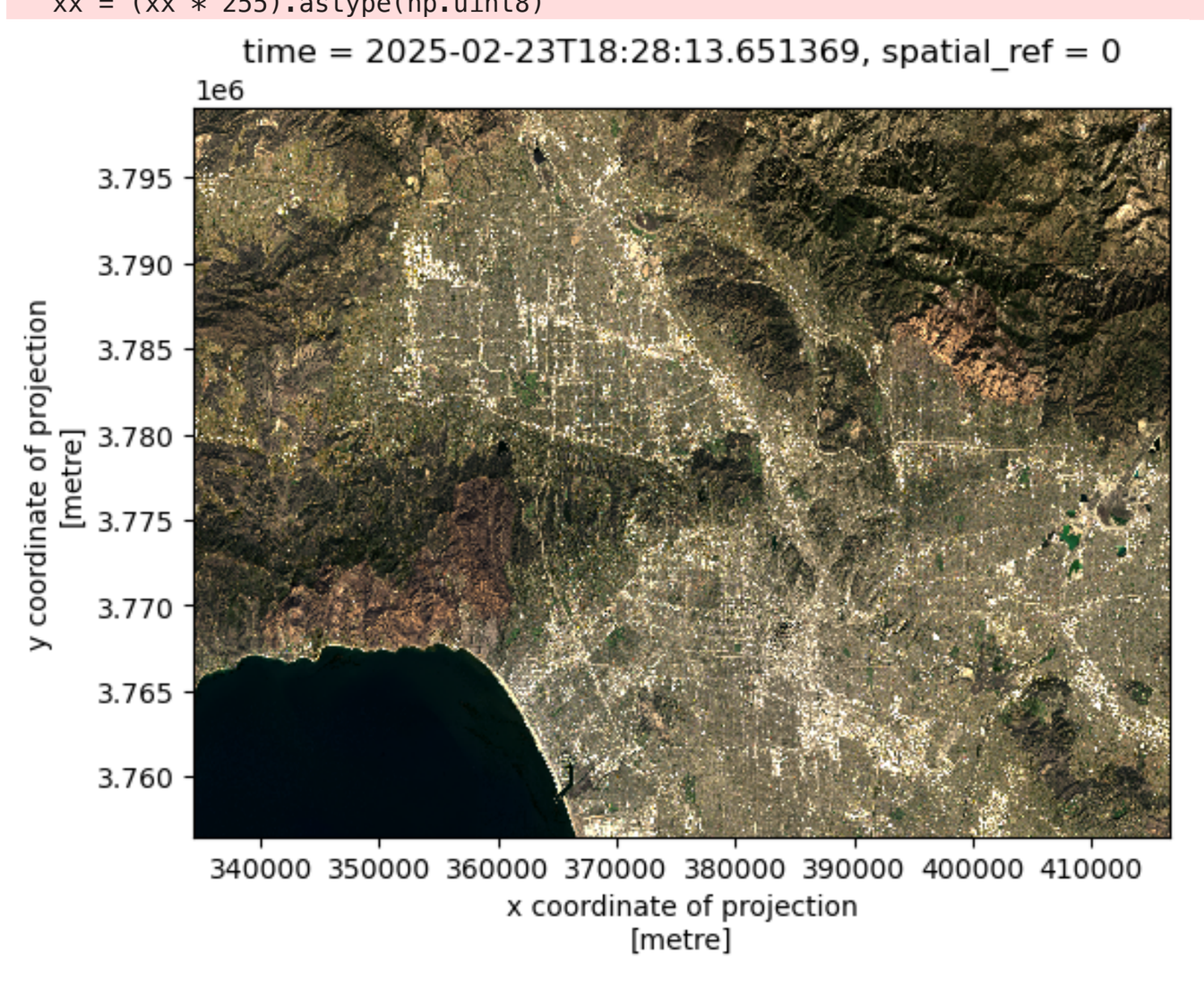
▶ Attributes: (0)

e) Create a new true color image that gets plotted without warnings

In [123…
```
landsat[['red', 'green', 'blue']].to_array().plot.imshow(robust=True)
```

Out[123… `<matplotlib.image.AxesImage at 0x386e3b990>`

```
/Users/henryoliver/opt/anaconda3/envs/eds220-env/lib/python3.11/site-packages/matplotlib/cm.py:478: RuntimeWarning: invalid value encountered in cast
  xx = (xx * 255).astype(np.uint8)
```



Why weren't we getting an image at the beginning? What changed to give us our True Color image?

The output in part a) showed a blank plot, which was not displaying any of the bands that I had selected. This is because once the minimum, maximum and NA data for each band were not specified, and as a result the bands could not be printed. Consequently, only the x & y data was plotted. In part e) my plot shows a realistic true color image without warnings. This is because by adding the `robust = True` argument to my `im.show()` method, the vmin and vmax values for each band are specified, which allows the bands to be displayed accurately. Also, by turning the NA values to 0, we can eliminate the warning about NA values, without comprimising our image.

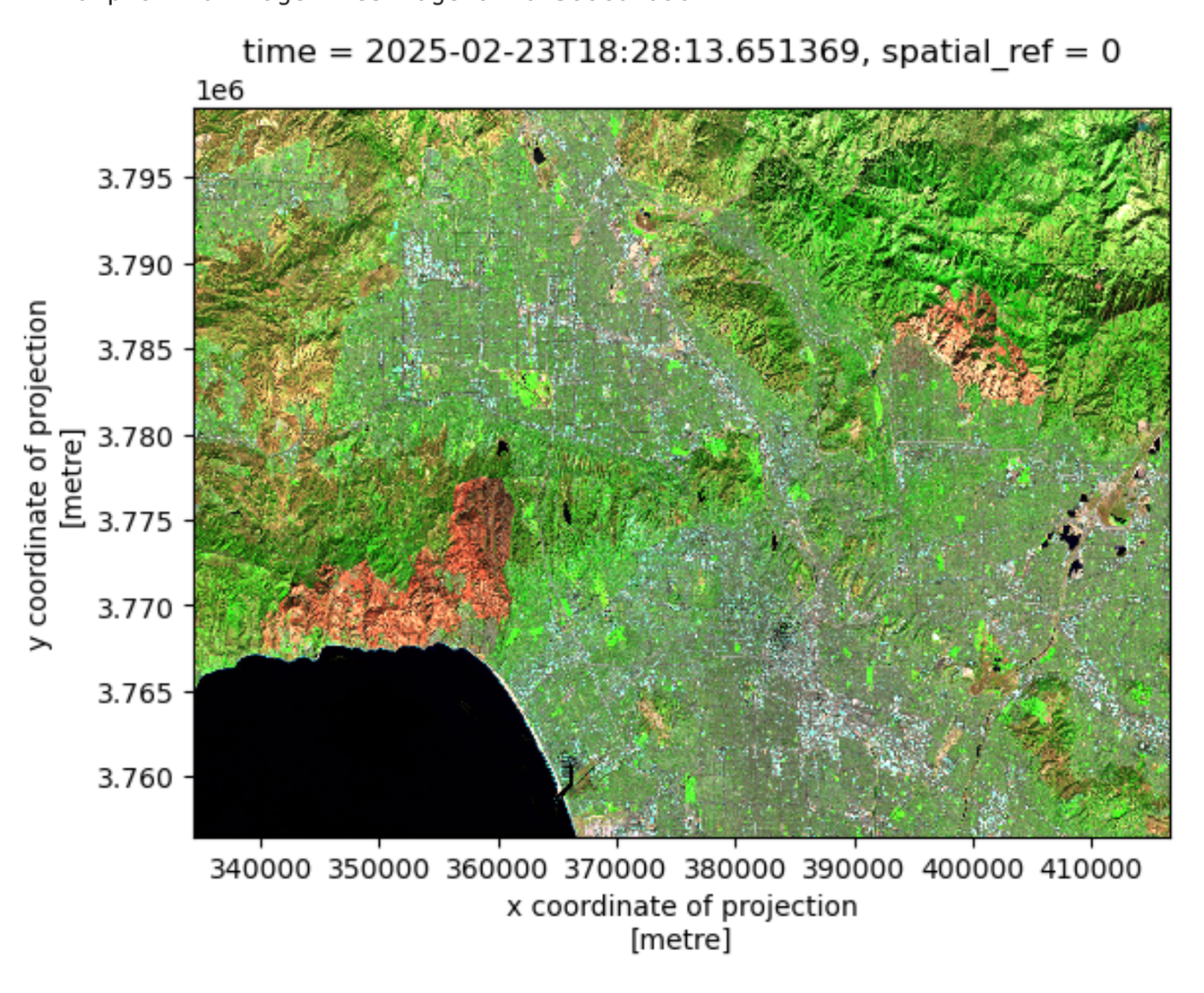
Step 6: False color image

'False color' images display short-wave infrared, near-infrared, and red light as detected by the satellite. While these images are less accurate to our visual interpretation of a landscape, they help highlight certain difficult-to-see characteristics of a landscape.

a) Without creating any new variables, create a false color image by plotting the short-wave infrared (swir22), near-infrared, and red variables (in that order)

In [124…
```
# Display false color image
landsat[['swir22', 'nir08', 'red']].to_array().plot.imshow(robust=True)
```

Out[124… `<matplotlib.image.AxesImage at 0x366c0fd90>`



Step 7: Map

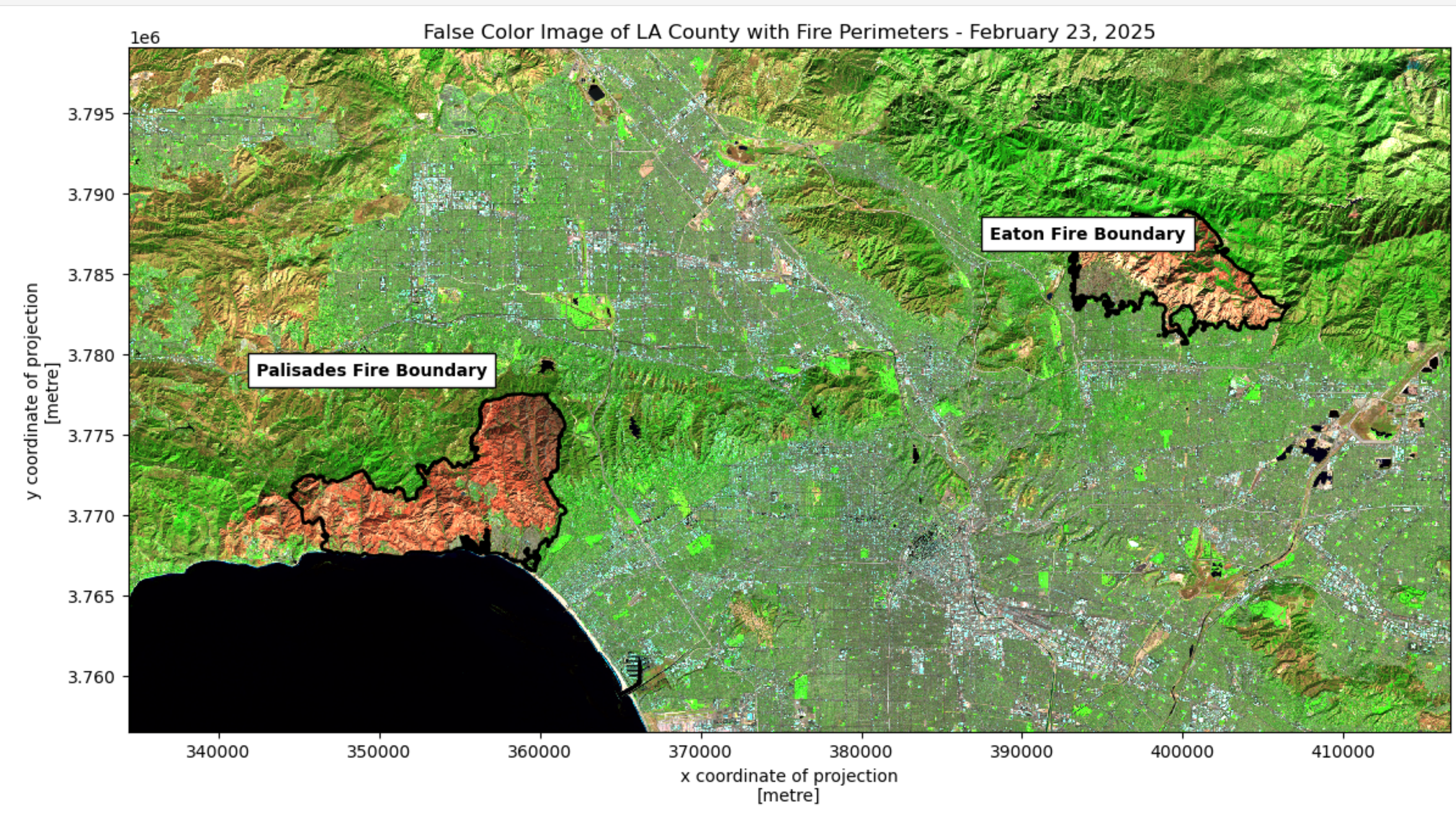
Now we can put it all together. In order to do so, we'll need to make sure that our imagery and perimeters share a CRS

a) Create a map showing the shortwave infrared/near-infrared/red false color image together with both fire perimeters.

- Add a title and annotations to your figure

In [125…
```
# Make sure our fires and imagery have the same CRS
fires = fires.to_crs(landsat.rio.crs)
assert(fires.crs == landsat.rio.crs)
```

In [126…
```
# Create plot
fig, ax = plt.subplots(1, 1, figsize=(14, 8))

# Plot LANDSAT false color imagery
landsat[['swir22', 'nir08', 'red']].to_array().plot.imshow( # Select false color bands
    robust=True,
    ax=ax,
    add_colorbar = False,
    zorder = 0) # Specify this is the borrom layer of our plot
# Add title
ax.set_title("False Color Image of LA County with Fire Perimeters - February 23, 2025")

# Plot fire perimeters
fires.plot(
    ax=ax,
    edgecolor='black',
    facecolor= "none",
    linewidth=2,
    alpha= 1,
    zorder=1 # Specify plot to be on TOP of satellite imagery
)

# Add Map data labels
plt.figtext(x=.63, y=.65, s="Eaton Fire Boundary", weight='bold',
            bbox = {'facecolor': 'white', #Add text border
                    'pad':5})
plt.figtext(x=.2,  y=.51, s="Palisades Fire Boundary", weight='bold',
            bbox = {'facecolor': 'white', #Add text border
                    'pad':5})

# Add Data Citation
fig.text(
    0.01, 0.01,
    "Data Sources: Landsat 8 (USGS), Fire Perimeters (County of Los Angeles)",
    fontsize=10,
    ha='left',
    va='bottom',
    style = 'italic'
)

# Add Timestamp
fig.text(
    1, 0.01,
    "Image Taken 2025-02-23 18:28:13 UTC",
    fontsize=10,
    ha='right',
    va='bottom',
    style = 'italic'
)
plt.show()
```


False Color Image of LA County with Fire Perimeters - February 23, 2025
1e6
3.795
3.790
3.785
3.780
3.775
3.770
3.765
3.760
y coordinate of projection [metre]
Eaton Fire Boundary
Palisades Fire Boundary
340000 350000 360000 370000 380000 390000 400000 410000
x coordinate of projection [metre]

Data Sources: Landsat 8 (USGS), Fire Perimeters (County of Los Angeles)   *Image Taken 2025-02-23 18:28:13 UTC*

Interpretation:

The figure shows false-color Landsat imagery of Los Angeles County, with the Palisades and Eaton fire perimeters outlined in black. In this image, blue and green light are substituted with reading for short-wave (SWIR) and near-infrared (NIR) light respectively. As a result, healthy vegetation appears in shades of green (healthy vegetation has relatively high reflectence of NIR), bare ground or urban areas appear in gray to brown, and the areas affected by fire stand out in red (low NIR reflections, high SWIR), highlighting burned vegetation. By overlaying the fire perimeters, we can clearly see how the fire boundaries correspond to the affected areas, providing a visual confirmation of the fire extent.

References:

County of Los Angeles. (2025). Eaton fire perimeter (Version 2025-02-21) [Shapefile]. Los Angeles County GIS Hub. https://egis-lacounty.hub.arcgis.com/datasets/lacounty::palisades-and-eaton-dissolved-fire-perimeters-2025/explore?layer=0

County of Los Angeles. (2025). Palisades fire perimeter (Version 2025-02-21) [Shapefile]. Los Angeles County GIS Hub. https://egis-lacounty.hub.arcgis.com/datasets/lacounty::palisades-and-eaton-dissolved-fire-perimeters-2025/explore?layer=1&location=34.133066%2C-118.349606%2C9.60

Microsoft Planetary Computer. (2025). Landsat 8 imagery of Los Angeles County (February 23, 2025) [NetCDF]. https://planetarycomputer.microsoft.com/dataset/landsat-c2-l2