# Import packages

import os

import glob

import matplotlib.pyplot as plt

from mpl\_toolkits.axes\_grid1 import make\_axes\_locatable

import numpy as np

import rasterio

from rasterio import transform

from rasterio.plot import show\_hist

from rasterio.plot import show

import xarray

**Question 1**

**a) What is the air temperature (in F) and cloud cover (in %) in Florence, OR (in 2020) on January 31, 2020?**

# Read data

xds = xarray.open\_dataset(filepath + 'era/usa\_t2m\_tcc\_2020.nc', decode\_coords='all')

xds\_daily = xds.resample(time='1D').mean()

# Next, find the index of the grid point nearest a specific lat/lon -- in this case, the city of Florence

florence\_weather = xds\_daily.sel(latitude=43.974659, longitude=-124.103142, method='nearest')

# Convert temperature values to Fahrenheit

fahrenheit = (florence\_weather['t2m'][31].values - 273.15) \* 9/5 + 32

# Note: Jan 31 is DOY 31

print('On January 31, 2020 air temperature in Florence, OR was %.2f F and the cloud cover was %.2f %%' % (fahrenheit, florence\_weather['tcc'][31].values \* 100))

On January 31, 2020 air temperature in Florence, OR was 54.04 F and the cloud cover was 99.88 %

**b) What is the air temperature (in F) and cloud cover (in %) in Eugene, OR (in 2020) on February 15, 2020?**

# Next, find the index of the grid point nearest a specific lat/lon -- in this case, the city of Eugene

eugene\_weather = xds\_daily.sel(latitude = 44.052071, longitude = -123.086754, method='nearest')

# Note: February 15th is DOY 46

# Convert temperature values to Fahrenheit

eug\_fahrenheit = (eugene\_weather['t2m'][46].values - 273.15) \* 9/5 + 32

print('On February 15, 2020 air temperature in Eugene, OR was %.2f F and the cloud cover was %.2f %%' % (eug\_fahrenheit, eugene\_weather['tcc'][46].values \* 100))

On February 15, 2020 air temperature in Eugene, OR was 43.79 F and the cloud cover was 97.51 %

**Question 2**

**a) Find grid cell and rough location of the highest average air temperature (i.e. hottest place)**

hottest\_place = np.mean(xds\_daily['t2m'], axis=0).argmax()

index = np.unravel\_index(hottest\_place, np.mean(xds\_daily['t2m'], axis=0).shape)

xds\_daily['t2m'].shape

(366, 99, 233)

np.mean(xds\_daily['t2m'], axis = 0).shape

(99, 233)

hottest\_long = np.mean(xds\_daily['t2m'], axis = 0)[81,55].longitude.values

hottest\_lat = np.mean(xds\_daily['t2m'], axis = 0)[81,55].latitude.values

print('The coordinates of the hottest place are %.2f, %.2f.' % (hottest\_lat, hottest\_long))

The coordinates of the hottest place are 28.99, -111.25.

**Reverse geocoding the above coordinates shows that the hottest place in this dataset is located at: Plan de Ayala, 83325, Hermosillo, Sonora, Plan de Ayala Hermosillo, México.**

**b) Find grid cell and rough location of the lowest average air temperature (i.e. coldest place)**

coldest\_place = np.mean(xds\_daily['t2m'], axis=0).argmin()

index = np.unravel\_index(coldest\_place, np.mean(xds\_daily['t2m'], axis=0).shape)

index

(21, 61)

coldest\_long = np.mean(xds\_daily['t2m'], axis = 0)[21,61].longitude.values

coldest\_lat = np.mean(xds\_daily['t2m'], axis = 0)[21,61].latitude.values

print('The coordinates of the coldest place are %.2f, %.2f.' % (coldest\_lat, coldest\_long))

The coordinates of the coldest place are 43.99, -109.75.

**Reverse geocoding the above coordinates shows that the hottest place in this dataset is located in Cody, WY, United States.**

**c) Find grid cell and rough location of the highest average cloudiness (i.e. cloudiest place)**

cloudiest\_place = np.mean(xds\_daily['tcc'], axis=0).argmax()

index = np.unravel\_index(cloudiest\_place, np.mean(xds\_daily['tcc'], axis=0).shape)

index

(0, 0)

cloudiest\_long = np.mean(xds\_daily['tcc'], axis = 0)[0,0].longitude.values

cloudiest\_lat = np.mean(xds\_daily['tcc'], axis = 0)[0,0].latitude.values

print('The coordinates of the cloudiest place are %.2f, %.2f.' % (cloudiest\_lat, cloudiest\_long))

The coordinates of the cloudiest place are 49.24, -125.00.

**Reverse geocoding the above coordinates shows that the cloudiest place in this dataset is located in Port Alberni V9Y, British Columbia, Canada**

**d) Find grid cell and rough location of the lowest average cloudiest (i.e. least cloudy place)**

least\_cloudy\_place = np.mean(xds\_daily['tcc'], axis=0).argmin()

index = np.unravel\_index(least\_cloudy\_place, np.mean(xds\_daily['tcc'], axis=0).shape)

index

(71, 41)

least\_cloudy\_long = np.mean(xds\_daily['tcc'], axis = 0)[71, 41].longitude.values

least\_cloudy\_lat = np.mean(xds\_daily['tcc'], axis = 0)[71, 41].latitude.values

print('The coordinates of the cloudiest place are %.2f, %.2f.' % (least\_cloudy\_lat, least\_cloudy\_long))

The coordinates of the cloudiest place are 31.49, -114.75.

**Reverse geocoding the above coordinates shows that the least cloudy place in this dataset is located off the coast of Baja California, Mexico in the Gulf of California.**

**e) Find the grid cell and rough location of the place with the highest range in daily air temperature**

# Resample the original dataset to reduce the time data dimension

xds\_daily\_min = np.min(xds\_daily['t2m'], axis = 0)

xds\_daily\_min

xarray.DataArray

't2m'

* latitude: 99
* longitude: 233
* array([[269.54614, 269.9083 , 268.75464, ..., 253.5631 , 254.69144,
* 254.49556],
* [270.14514, 267.34943, 265.0641 , ..., 254.31932, 253.95721,
* 253.46048],
* [272.68903, 270.9557 , 268.14398, ..., 252.37546, 251.43365,
* 250.67844],
* ...,
* [288.39587, 288.3816 , 288.36526, ..., 293.44495, 293.47928,
* 293.53958],
* [288.47852, 288.47052, 288.50317, ..., 293.71115, 293.74377,
* 293.7921 ],
* [288.5628 , 288.55115, 288.5858 , ..., 293.9367 , 294.00064,
* 294.00464]], dtype=float32)
* Coordinates:
  + longitude

(longitude)

float32

-125.0 -124.8 ... -67.25 -67.0

* + latitude

(latitude)

float32

49.24 48.99 48.74 ... 24.99 24.74

* Attributes: (0)

xds\_daily\_max = np.max(xds\_daily['t2m'], axis = 0)

xds\_daily\_max

xarray.DataArray

't2m'

* latitude: 99
* longitude: 233
* array([[294.49335, 294.26083, 294.98743, ..., 293.40433, 293.88705,
* 294.4807 ],
* [294.85352, 294.37445, 293.87637, ..., 294.7832 , 294.55133,
* 294.73587],
* [292.5385 , 293.26373, 293.71115, ..., 300.5082 , 300.63113,
* 299.97418],
* ...,
* [296.69775, 296.6911 , 296.75406, ..., 302.0663 , 302.0553 ,
* 302.08466],
* [296.8783 , 296.93097, 296.95294, ..., 302.03665, 301.99136,
* 302.025 ],
* [297.1285 , 297.12918, 297.1708 , ..., 301.992 , 301.9577 ,
* 301.9667 ]], dtype=float32)
* Coordinates:
  + longitude

(longitude)

float32

-125.0 -124.8 ... -67.25 -67.0

* + latitude

(latitude)

float32

49.24 48.99 48.74 ... 24.99 24.74

* Attributes: (0)

xds\_daily\_range = xds\_daily\_max - xds\_daily\_min

xds\_daily\_range

xarray.DataArray

't2m'

* latitude: 99
* longitude: 233
* array([[24.947205 , 24.35254 , 26.232788 , ..., 39.841232 , 39.195618 ,
* 39.985153 ],
* [24.708374 , 27.025024 , 28.812286 , ..., 40.463882 , 40.594116 ,
* 41.27539 ],
* [19.849487 , 22.308044 , 25.56717 , ..., 48.13275 , 49.19748 ,
* 49.295746 ],
* ...,
* [ 8.30188 , 8.309509 , 8.388794 , ..., 8.621368 , 8.576019 ,
* 8.545074 ],
* [ 8.39978 , 8.460449 , 8.449768 , ..., 8.3255005, 8.247589 ,
* 8.23288 ],
* [ 8.565704 , 8.578033 , 8.585022 , ..., 8.055298 , 7.957062 ,
* 7.9620667]], dtype=float32)
* Coordinates:
  + longitude

(longitude)

float32

-125.0 -124.8 ... -67.25 -67.0

* + latitude

(latitude)

float32

49.24 48.99 48.74 ... 24.99 24.74

* Attributes: (0)

max\_daily\_range = xds\_daily\_range.argmin()

index = np.unravel\_index(max\_daily\_range, xds\_daily\_range.shape)

index

(46, 7)

max\_range\_long = xds\_daily\_range[46, 7].longitude.values

max\_range\_lat = xds\_daily\_range[46, 7].latitude.values

print('The coordinates of the place with the highest daily range in temperature are %.2f, %.2f.' % (max\_range\_lat, max\_range\_long))

The coordinates of the place with the highest daily range in temperature are 37.74, -123.25.

**The coordinates of the location with the highest daily range (37.74, -123.25) are located in the Pacific Ocean just off the coast of San Francisco, CA.**

**f) Find the grid cell and rough location of the place with the absolute coldest temperature on a single day**

# Calculate the absolute min of any day from the minimums data array created in the previous question

xds\_abs\_min = xds\_daily\_min.argmin()

index = np.unravel\_index(xds\_abs\_min, xds\_daily\_min.shape)

index

(2, 55)

abs\_min\_long = xds\_daily\_min[2, 55].longitude.values

abs\_min\_lat = xds\_daily\_min[2, 55].latitude.values

print('The coordinates of the place with the coldest absolute temperature on a single day are %.2f, %.2f.' % (abs\_min\_lat, abs\_min\_long))

The coordinates of the place with the coldest absolute temperature on a single day are 48.74, -111.25.

**The coordinates of the location with the absolute coldest temperature on a single day (48.74, -111.25) are located near Chester, MT, USA just outside of Mt. Brown.**

**Question 3**

**a) Creating a color infrared composite of the Landsat image over Florence, OR**

# Define file path

filepath = '/Users/lily/Documents/GitHub/geospatial-data-science/labs/lab4/'

# Define list of Landsat bands

files = sorted(glob.glob(filepath + 'landsat/\*.tif'))

print(files)

# Open a single band

src = rasterio.open(files[0])

band\_1 = src.read(1)

# Find metadata (e.g. driver, data type, coordinate reference system, transform etc.)

print(src.profile)

# Get corners of dataset

full\_extent = [src.bounds.left, src.bounds.right, src.bounds.bottom, src.bounds.top]

print(full\_extent)

# Plot dataset

fig, ax = plt.subplots(figsize=(8,8))

im = ax.imshow(band\_1, cmap='gray', extent=full\_extent)

ax.set\_title("Band 1")

fig.colorbar(im, orientation='vertical')

plt.show()

# Open all bands in a loop

list\_bands = []

for file in files:

# Read band

src = rasterio.open(file)

band = src.read(1)

# Append to list

list\_bands.append(band)

# Convert from list of arrays to n-dimensional array

all\_bands = np.dstack(list\_bands)

all\_bands.shape

# Convert 16-bit values to an 8-bit range of 0-255

all\_bands\_image = np.uint8((all\_bands / 65536) \* 255)

# Produce a new array by stacking the RGB bands

color\_ir\_comp = np.dstack((all\_bands\_image[:,:,4],all\_bands\_image[:,:,2],all\_bands\_image[:,:,2]))

# Write an array as a raster band to a new 8-bit file. For the new file's profile,

# we start with the profile of the source

profile = src.profile

# And then change the band count to 3, set the dtype to uint8, and specify LZW compression.

profile.update(dtype=rasterio.uint8, count=3, compress='lzw')

with rasterio.open(filepath + 'landsat/color\_IR.tif', 'w', \*\*profile) as dst:

# Write array

dst.write(np.rollaxis(rgb, axis=2))

# Plot as Color Infrared Composite image

fig, ax = plt.subplots(figsize=(8,8))

im = ax.imshow(color\_ir\_comp, extent=full\_extent)

ax.set\_title("Color Infrared Composite")

plt.show()

A picture containing text

Description automatically generated

**b) Creating an NDVI image**

# Compute NDVI

np.seterr(divide='ignore', invalid='ignore')

ndvi = np.divide((all\_bands[:,:,4].astype(float) - all\_bands[:,:,3].astype(float)), \

(all\_bands[:,:,4].astype(float) + all\_bands[:,:,3].astype(float)))

# Plot NDVI image

fig, ax = plt.subplots(figsize=(8,8))

im = ax.imshow(ndvi, extent=full\_extent)

ax.set\_title("NDVI image")

fig.colorbar(im, orientation='vertical')

plt.show()

A picture containing text, monitor, screen, display

Description automatically generated

**Question 4**

**a) Produce an NDWI histogram for the Landsat image of Florence**

# Compute NDWI

np.seterr(divide='ignore', invalid='ignore')

ndwi = np.divide((all\_bands[:,:,2].astype(float) - all\_bands[:,:,4].astype(float)), \

(all\_bands[:,:,2].astype(float) + all\_bands[:,:,4].astype(float)))

ndwi

array([[ nan, nan, nan, ..., nan,

nan, nan],

[ nan, nan, nan, ..., nan,

nan, nan],

[ nan, nan, nan, ..., nan,

nan, nan],

...,

[ nan, nan, nan, ..., -0.29082536,

-0.31481248, -0.30104966],

[ nan, nan, nan, ..., -0.32622883,

-0.30386218, -0.27118251],

[ nan, nan, nan, ..., -0.34158455,

-0.27173558, -0.30858688]])

# Plot NDVI Histogram

fig, axes = plt.subplots(figsize=(12,8))

show\_hist(

ndwi, bins=50, lw=0.0, stacked=False, alpha=0.3,

histtype='stepfilled', ax = axes, title="NDWI Histogram")

axes.set\_xlabel("NWDI Value")

Text(0.5, 0, 'NWDI Value')

A screenshot of a computer

Description automatically generated with low confidence

**b) Choose a threshold and produce a binary water mask**

# Plot NDVI image

fig, ax = plt.subplots(figsize=(8,8))

im = ax.imshow(ndwi, extent=full\_extent)

ax.set\_title("NDWI image")

fig.colorbar(im, orientation='vertical')

plt.show()

A picture containing text, monitor, television, screen

Description automatically generated

# Note: The water in this image shows up in the 0.05-0.1 range

# Create a binary mask

ndwi\_mask = np.copy(ndwi)

not\_water = ndwi\_mask[ndwi\_mask < -0.05] = -1

water = ndwi\_mask[ndwi\_mask >= -0.05] = 1

# Plot NDWI binary mask and binary mask histogram

fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (16, 9))

show(ndwi\_mask, ax = ax1, title = "Florence, OR NDWI")

show\_hist(ndwi\_mask, bins = 50, histtype='stepfilled',

lw=0.0, stacked=False, alpha=0.3, ax= ax2, title = "NDWI Histogram" )

ax2.set\_xlabel("NWDI Value")

plt.show()

A picture containing text

Description automatically generated

**c) Compute the area of water in the image (including ocean)**

water\_px = np.sum(ndwi\_mask == 1)

water\_px

774639

# Note: Landsat spatial resolution is 30x30 m

# Calculate area of water px

water\_area = water\_px \* 30

# Convert to square km

water\_area\_km = water\_area / 1000000

water\_area\_km

23.23917

**In this Landsat image of Florence, OR, there is 23.23917 km2 of water (including the ocean).**