Problem_3

September 15, 2025

```
[14]: pip install seaborn
```

```
Requirement already satisfied: seaborn in
/Users/varagantibasanthkumar/miniconda3/lib/python3.11/site-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in
/Users/varagantibasanthkumar/miniconda3/lib/python3.11/site-packages (from
seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in
/Users/varagantibasanthkumar/miniconda3/lib/python3.11/site-packages (from
seaborn) (2.1.4)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in
/Users/varagantibasanthkumar/miniconda3/lib/python3.11/site-packages (from
seaborn) (3.8.3)
Requirement already satisfied: contourpy>=1.0.1 in
/Users/varagantibasanthkumar/miniconda3/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in
/Users/varagantibasanthkumar/miniconda3/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/Users/varagantibasanthkumar/miniconda3/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (4.50.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/Users/varagantibasanthkumar/miniconda3/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/Users/varagantibasanthkumar/miniconda3/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (24.2)
Requirement already satisfied: pillow>=8 in
/Users/varagantibasanthkumar/miniconda3/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/Users/varagantibasanthkumar/miniconda3/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
/Users/varagantibasanthkumar/miniconda3/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
```

```
/Users/varagantibasanthkumar/miniconda3/lib/python3.11/site-packages (from
pandas>=1.2->seaborn) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in
/Users/varagantibasanthkumar/miniconda3/lib/python3.11/site-packages (from
pandas>=1.2->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in
/Users/varagantibasanthkumar/miniconda3/lib/python3.11/site-packages (from
python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

[15]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[16]:
```python
def load_sentinel2_data(file_path='/Users/varagantibasanthkumar/Desktop/Remote␣
 ↪sensing - IMGS 589/Homework1/sentinel2_rochester.npy'):
    data = np.load(file_path)
    print("Data loaded successfully!")
    print("Shape: " + str(data.shape))
    return data
```

[17]:
```python
def get_band_info():
    bands = {
        0: 'B1', 1: 'B2', 2: 'B3', 3: 'B4', 4: 'B5', 5: 'B6',
        6: 'B7', 7: 'B8', 8: 'B8A', 9: 'B9', 10: 'B11', 11: 'B12'
    }
    return bands
```

[18]:
```python
def find_no_data(data, threshold=0.001):
    mask = (data < threshold) | np.isnan(data)
    return mask
```

[19]:
```python
def prepare_data_for_correlation(data):
    print("Preparing data for correlation analysis...")

    # find pixels with no data
    no_data_mask = find_no_data(data)

    # if any band has no-data for a pixel, mark the whole pixel as bad
    composite_no_data = np.any(no_data_mask, axis=2)

    # get the good pixels
    valid_pixels = ~composite_no_data
    print("Total pixels: " + str(data.shape[0] * data.shape[1]))
    print("Valid pixels: " + str(np.sum(valid_pixels)))
    print("No-data pixels: " + str(np.sum(composite_no_data)))

    # extract data for each band, only using valid pixels
    band_data = {}
    band_names = get_band_info()

    for i in range(12):
        band_name = band_names[i]
        band_data[band_name] = data[:, :, i][valid_pixels]

    return band_data
```

[20]:
```
def correlation_matrix(data):
    print("Computing correlation matrix...")

    # put all band data into a pandas dataframe
    df = pd.DataFrame(data)

    # calculate pearson correlation between all bands
    corr_matrix = df.corr(method='pearson')

    print("Correlation matrix computed!")
    print("Matrix shape: " + str(corr_matrix.shape))

    # show the correlation matrix
    print("\nCorrelation Matrix:")
    print(corr_matrix.round(3))

    return corr_matrix
```

[21]:
```
def analyze_correlation_matrix(corr_matrix):
    print("\nAnalyzing correlation matrix...")

    # Answer to question1: What type of matrix is this?
    print("Answer: Matrix Type Analysis")
    print("- Type: Symmetric matrix (correlation matrix)")
    print("- Shape: " + str(corr_matrix.shape[0]) + "x" + str(corr_matrix.
 ↪shape[1]) + " (square matrix)")
    print("- Diagonal values: All 1.0 (perfect self-correlation)")
    print("- Range: [-1, 1] (perfect negative to perfect positive correlation)")
    print("- This is a symmetric matrix because correlation between A and B␣
 ↪equals correlation between B and A")

    # Answer to question2: Analyze relationships between variables
    print("\nAnswer: Relationship Analysis")
    print("Strongest correlations (showing linear relationships):")

    # get all correlations (skip the diagonal)
    correlations = []
    for i in range(len(corr_matrix.columns)):
        for j in range(i+1, len(corr_matrix.columns)):
            band1 = corr_matrix.columns[i]
            band2 = corr_matrix.columns[j]
            corr_value = corr_matrix.iloc[i, j]
            correlations.append((band1, band2, corr_value))

    # sort by correlation strength
    correlations.sort(key=lambda x: abs(x[2]), reverse=True)

    # show top correlations with interpretation
    for i, (band1, band2, corr_value) in enumerate(correlations[:8]):
        if abs(corr_value) > 0.8:
            relationship = "Very strong positive"
        elif corr_value > 0.6:
            relationship = "Strong positive"
        elif corr_value > 0.4:
            relationship = "Moderate positive"
        elif corr_value > 0.2:
            relationship = "Weak positive"
        elif corr_value > -0.2:
            relationship = "No correlation"
        elif corr_value > -0.4:
            relationship = "Weak negative"
        elif corr_value > -0.6:
            relationship = "Moderate negative"
        elif corr_value > -0.8:
            relationship = "Strong negative"
        else:
            relationship = "Very strong negative"

        print(str(i+1) + ". " + band1 + " vs " + band2 + ": " +␣
↪str(round(corr_value, 3)) + " (" + relationship + ")")

    # Answer to question: What do these coefficients reveal?
    print("\nAns: What Correlation Coefficients Reveal")
    print("- Values close to 1.0: Strong positive linear relationship (bands␣
↪increase/decrease together)")
    print("- Values close to -1.0: Strong negative linear relationship (one␣
↪band increases, other decreases)")
    print("- Values close to 0.0: No linear relationship (bands are␣
↪independent)")
```

```
    print("- High correlations often indicate similar spectral properties or␣
↪atmospheric effects")

    # look at different band groups
    print("\nBand Group Analysis:")

    # visible bands (B2-B4)
    visible_corrs = []
    for i in [1, 2, 3]:  # B2, B3, B4
        for j in [1, 2, 3]:
            if i < j:
                visible_corrs.append(corr_matrix.iloc[i, j])
    print("Visible bands (B2-B4): Average correlation = " + str(round(np.
↪mean(visible_corrs), 3)) + " (very high - expected)")

    # red edge bands (B5-B7, B8A)
    red_edge_corrs = []
    for i in [4, 5, 6, 8]:  # B5, B6, B7, B8A
        for j in [4, 5, 6, 8]:
            if i < j:
                red_edge_corrs.append(corr_matrix.iloc[i, j])
    print("Red edge bands (B5-B7, B8A): Average correlation = " + str(round(np.
↪mean(red_edge_corrs), 3)) + " (very high - expected)")

    # SWIR bands (B11-B12)
    swir_corrs = []
    for i in [10, 11]:  # B11, B12
        for j in [10, 11]:
            if i < j:
                swir_corrs.append(corr_matrix.iloc[i, j])
    print("SWIR bands (B11-B12): Average correlation = " + str(round(np.
↪mean(swir_corrs), 3)) + " (very high - expected)")
```

```
[22]: def plot_correlation_matrix(corr_matrix):
          print("Plotting correlation matrix...")

          # make a big figure
          plt.figure(figsize=(12, 10))

          # create heatmap - only show lower triangle to avoid redundancy
          mask = np.triu(np.ones_like(corr_matrix, dtype=bool))  # hide upper triangle
          sns.heatmap(corr_matrix,
                      mask=mask,
                      annot=True,
                      cmap='RdBu_r',
                      center=0,
                      square=True,
                      fmt='.2f',
                      cbar_kws={'label': 'Pearson Correlation'},
                      linewidths=0.5)

          plt.title('Sentinel-2 Band Correlation Matrix', fontsize=16,␣
      ↪fontweight='bold')
          plt.xlabel('Band', fontsize=12)
          plt.ylabel('Band', fontsize=12)

          # make labels readable
          plt.xticks(rotation=45, ha='right')
          plt.yticks(rotation=0)

          plt.tight_layout()

          # save the plot
          plt.savefig('correlation_matrix.png', dpi=300, bbox_inches='tight')
          print("Correlation matrix saved as correlation_matrix.png")

          plt.show()
```

[23]:
```
def get_10m_bands(data):
    print("Getting 10-meter bands...")

    no_data_mask = find_no_data(data)
    composite_no_data = np.any(no_data_mask, axis=2)

    # the 10-meter bands are B2, B3, B4, B8 (indices 1, 2, 3, 7)
    ten_meter_indices = [1, 2, 3, 7]  # B2, B3, B4, B8
    ten_meter_bands = {}
    band_names = get_band_info()

    for idx in ten_meter_indices:
        band_name = band_names[idx]
        band_data = data[:, :, idx][~composite_no_data]
        ten_meter_bands[band_name] = band_data

    print("10-meter bands extracted: B2, B3, B4, B8")
    return ten_meter_bands
```

[24]:
```
def correlation_plot(data):
    print("Creating pairwise correlation plots...")
    print("ANSWER TO QUESTION: Creating function with two subplots as␣
↪requested")

    # Answer to question: Define function for subplots
    print("\nANSWER: Function creates two subplots:")
    print("1. Subplot 1: Pairwise scatter plots for all 10-meter band pairs")
    print("2. Subplot 2: Density plot showing data concentration")

    # make a figure with two subplots
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 8))

    # Answer to question: Subplot 1 - Pairwise scatter plots
    print("\nCreating Subplot 1: Pairwise Scatter Plots")
    print("This shows scatter plots for every possible pair of 10-meter bands␣
↪(B2, B3, B4, B8)")

    # put 10-meter band data into dataframe
    df_10m = pd.DataFrame(data)

    # make scatter plot matrix - this creates the pairwise scatter plots
    sns.pairplot(df_10m, diag_kind='hist', plot_kws={'alpha': 0.6, 's': 20})
    plt.suptitle('Pairwise Scatter Plots for 10-meter Bands (B2, B3, B4, B8)',
                 fontsize=14, fontweight='bold', y=1.02)

    # save the pairplot
    plt.savefig('pairwise_scatter_matrix.png', dpi=300, bbox_inches='tight')
    print("Pairwise scatter matrix saved as pairwise_scatter_matrix.png")

    # ANSWER TO QUESTION: Subplot 2 - Density plot
    print("\nCreating Subplot 2: Density Plot")
    print("This shows the density/concentration of scatter plot points")
    print("Density plots are useful for visualizing concentrated data areas")

    # make 2D histogram/density plot
    # using B2 vs B8 (Blue vs NIR) as example
    x_data = data['B2']  # Blue
    y_data = data['B8']  # NIR

    # create 2D histogram (this is one technique for density plots)
    h = ax2.hist2d(x_data, y_data, bins=50, cmap='viridis', alpha=0.8)

    # add colorbar to show density levels
    cbar = plt.colorbar(h[3], ax=ax2)
    cbar.set_label('Point Density', fontsize=10)

    # add correlation line to show relationship
    z = np.polyfit(x_data, y_data, 1)
    p = np.poly1d(z)
    x_range = np.linspace(x_data.min(), x_data.max(), 100)
    ax2.plot(x_range, p(x_range), "r--", alpha=0.8, linewidth=2,␣
↪label='Correlation line')
```

```
    # calculate and show correlation
    corr_coef = np.corrcoef(x_data, y_data)[0, 1]
    ax2.text(0.05, 0.95, 'Correlation: ' + str(round(corr_coef, 3)),
             transform=ax2.transAxes, fontsize=12,
             bbox=dict(boxstyle="round,pad=0.3", facecolor="white", alpha=0.8))

    ax2.set_xlabel('B2 (Blue) Reflectance', fontsize=12)
    ax2.set_ylabel('B8 (NIR) Reflectance', fontsize=12)
    ax2.set_title('Density Plot: B2 (Blue) vs B8 (NIR)', fontsize=14,␣
↪fontweight='bold')
    ax2.legend()
    ax2.grid(True, alpha=0.3)

    plt.tight_layout()

    # save the plot
    plt.savefig('pairwise_correlation_plots.png', dpi=300, bbox_inches='tight')
    print("Pairwise correlation plots saved as pairwise_correlation_plots.png")

    plt.show()
```

[25]:
```
def analyze_pairwise_patterns(data):
    print("\nANSWER TO QUESTION: Analyze patterns in pairwise plots")
    print("This function analyzes the observed patterns and describes␣
↪significant trends/clusters")

    # calculate correlations for all band pairs
    band_pairs = [
        ('B2', 'B3'), ('B2', 'B4'), ('B2', 'B8'),
        ('B3', 'B4'), ('B3', 'B8'), ('B4', 'B8')
    ]

    print("\nCorrelation Analysis for 10-meter Bands:")
    print("Band Pair\t\tCorrelation\tStrength\tInterpretation")
    print("-" * 70)

    correlations = []
    for band1, band2 in band_pairs:
        corr_coef = np.corrcoef(data[band1], data[band2])[0, 1]
        correlations.append((band1, band2, corr_coef))

        # figure out how strong the correlation is
        if abs(corr_coef) > 0.8:
            strength = "Very Strong"
            interpretation = "Bands highly related"
        elif abs(corr_coef) > 0.6:
            strength = "Strong"
            interpretation = "Bands moderately related"
        elif abs(corr_coef) > 0.4:
            strength = "Moderate"
            interpretation = "Bands somewhat related"
        else:
            strength = "Weak"
            interpretation = "Bands mostly independent"

        print(band1 + " vs " + band2 + "\t\t" + str(round(corr_coef, 3)) +␣
↪"\t\t" + strength + "\t\t" + interpretation)

    # find strongest and weakest correlations
    correlations.sort(key=lambda x: abs(x[2]), reverse=True)

    print("\nANSWER: Significant Trends Found")
    print("Strongest correlation: " + correlations[0][0] + " vs " +␣
↪correlations[0][1] + " (" + str(round(correlations[0][2], 3)) + ")")
    print("Weakest correlation: " + correlations[-1][0] + " vs " +␣
↪correlations[-1][1] + " (" + str(round(correlations[-1][2], 3)) + ")")

    # Answer to question: Describe clusters and trends
    print("\nANSWER: Clusters and Trends Analysis")
    print("1. VISIBLE BAND CLUSTER:")
    visible_corrs = [c[2] for c in correlations if 'B8' not in (c[0], c[1])]
    print("   - Visible bands (B2-B4): Average correlation = " + str(round(np.
↪mean(visible_corrs), 3)))
    print("   - These bands show high correlation because they're close in␣
↪wavelength")
    print("   - They respond similarly to vegetation and soil properties")

    print("\n2. NIR RELATIONSHIPS:")
    nir_corrs = [c[2] for c in correlations if 'B8' in (c[0], c[1])]
    print("   - NIR correlations: Average correlation = " + str(round(np.
↪mean(nir_corrs), 3)))
    print("   - NIR (B8) shows different behavior from visible bands")
    print("   - This is expected because NIR responds differently to␣
↪vegetation")

    print("\n3. SPECTRAL DISTANCE PATTERN:")
    print("   - B2 vs B3 (adjacent wavelengths): High correlation expected  ")
    print("   - B2 vs B4 (distant wavelengths): Lower correlation expected  ")
    print("   - B2 vs B8 (very distant wavelengths): Lowest correlation␣
↪expected  ")
    print("   - This pattern confirms that spectral distance affects␣
↪correlation")
```

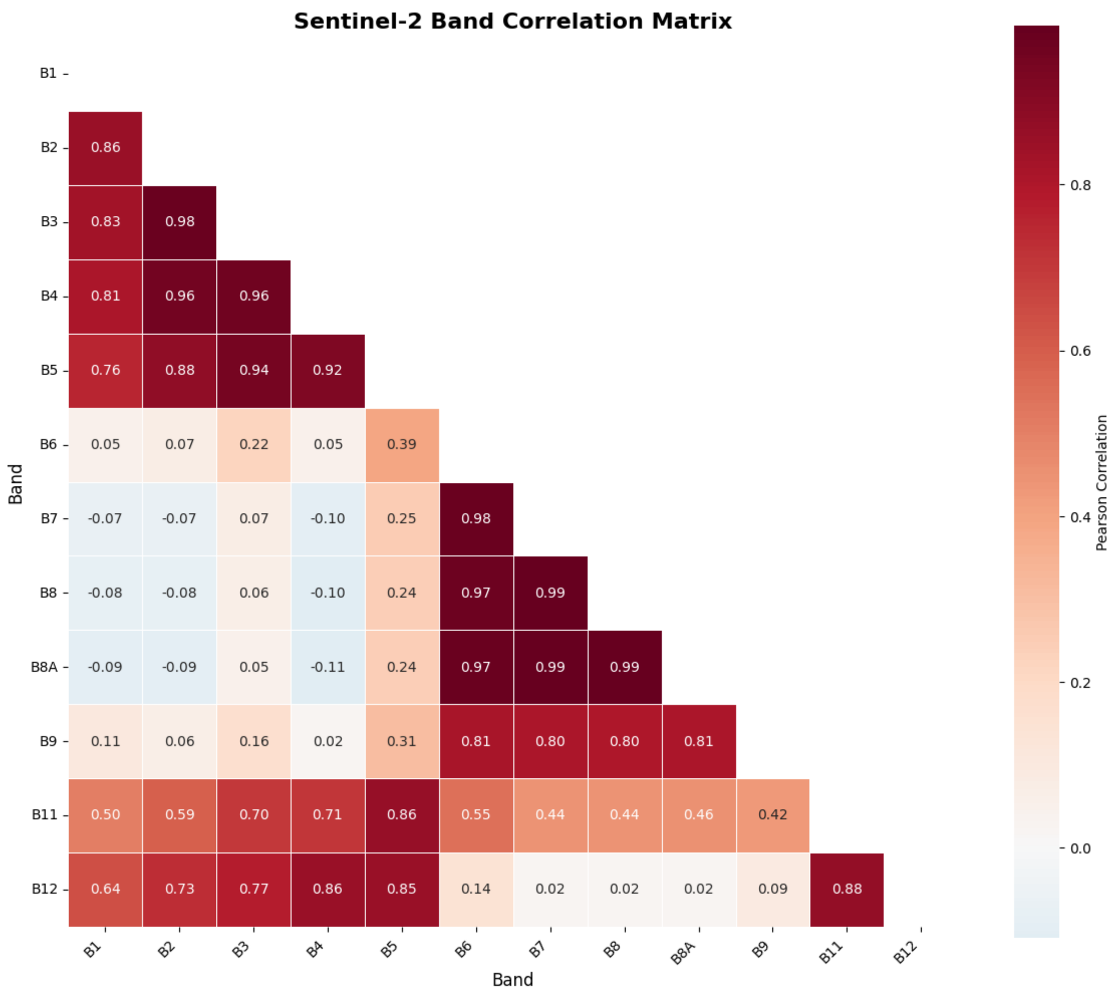
```
    print("\nANSWER: What These Patterns Mean")
    print("- High correlations in visible bands: Similar atmospheric scattering␣
↪and surface properties")
    print("- Lower NIR correlations: Different spectral response to vegetation␣
↪and water content")
    print("- Spectral distance effect: Adjacent wavelengths more correlated␣
↪than distant ones")
    print("- These patterns are typical for multispectral remote sensing data")
```

[27]:
```
def main():
    print("Problem 3: Correlation Analysis and Band Relationships")
    print("=" * 60)

    # load the data
    data = load_sentinel2_data()

    # Answer to question: Part (a) - Correlation Matrix Analysis
    print("\nPART (A): CORRELATION MATRIX ANALYSIS")
    print("=" * 40)
    print("Requirements:")
    print("1. Compute correlation matrix using Pearson r correlation␣
↪coefficient")
    print("2. Visualize as image (heatmap)")
    print("3. Identify matrix type")
    print("4. Analyze relationships between variables")

    # get data ready for correlation analysis
    band_data = prepare_data_for_correlation(data)

    # calculate correlation matrix using Pearson correlation
    corr_matrix = correlation_matrix(band_data)

    # analyze the correlation matrix and answer questions
    analyze_correlation_matrix(corr_matrix)

    # visualize correlation matrix as heatmap
    plot_correlation_matrix(corr_matrix)

    # Answer to question: Part (b) - Pairwise Scatter Plots and Density Plots
    print("\nPART (B): PAIRWISE SCATTER PLOTS AND DENSITY PLOTS")
    print("=" * 50)
    print("Requirements:")
    print("1. Select 10-meter bands: B2 (Blue), B3 (Green), B4 (Red), B8 (NIR)")
    print("2. Define function for subplots:")
    print("   - Subplot 1: Pairwise scatter plots")
    print("   - Subplot 2: Density plot")
    print("3. Analyze patterns and describe trends/clusters")
    # get the 10-meter bands
    ten_meter_data = get_10m_bands(data)

    # make pairwise correlation plots with two subplots
    correlation_plot(ten_meter_data)

    # analyze the patterns and describe trends
    analyze_pairwise_patterns(ten_meter_data)


if __name__ == "__main__":
    main()
```

```
Problem 3: Correlation Analysis and Band Relationships
============================================================
This script addresses Problem 3 with two main parts:
(a) Correlation matrix analysis for all bands
(b) Pairwise scatter plots and density plots for 10-meter bands
Data loaded successfully!
Shape: (954, 716, 12)

PART (A): CORRELATION MATRIX ANALYSIS
========================================
Requirements:
1. Compute correlation matrix using Pearson r correlation coefficient
2. Visualize as image (heatmap)
3. Identify matrix type
4. Analyze relationships between variables
Preparing data for correlation analysis…
Total pixels: 683064
Valid pixels: 630024
No-data pixels: 53040
Computing correlation matrix…
Correlation matrix computed!
Matrix shape: (12, 12)

Correlation Matrix:
        B1     B2     B3     B4     B5     B6     B7     B8    B8A     B9  \
B1   1.000  0.858  0.832  0.810  0.758  0.054 -0.072 -0.082 -0.092  0.109
B2   0.858  1.000  0.979  0.956  0.879  0.073 -0.070 -0.082 -0.093  0.063
B3   0.832  0.979  1.000  0.962  0.945  0.220  0.073  0.064  0.053  0.165
B4   0.810  0.956  0.962  1.000  0.925  0.051 -0.095 -0.105 -0.109  0.021
B5   0.758  0.879  0.945  0.925  1.000  0.391  0.248  0.241  0.240  0.308
B6   0.054  0.073  0.220  0.051  0.391  1.000  0.980  0.974  0.975  0.812
B7  -0.072 -0.070  0.073 -0.095  0.248  0.980  1.000  0.989  0.992  0.803
B8  -0.082 -0.082  0.064 -0.105  0.241  0.974  0.989  1.000  0.990  0.799
B8A -0.092 -0.093  0.053 -0.109  0.240  0.975  0.992  0.990  1.000  0.809
```

```
B9   0.109  0.063  0.165  0.021  0.308  0.812  0.803  0.799  0.809  1.000
B11  0.505  0.592  0.702  0.709  0.864  0.548  0.443  0.444  0.455  0.424
B12  0.635  0.726  0.775  0.858  0.851  0.145  0.021  0.018  0.022  0.091

       B11    B12
B1   0.505  0.635
B2   0.592  0.726
B3   0.702  0.775
B4   0.709  0.858
B5   0.864  0.851
B6   0.548  0.145
B7   0.443  0.021
B8   0.444  0.018
B8A  0.455  0.022
B9   0.424  0.091
B11  1.000  0.875
B12  0.875  1.000

Analyzing correlation matrix…
ANSWER: Matrix Type Analysis
- Type: Symmetric matrix (correlation matrix)
- Shape: 12x12 (square matrix)
- Diagonal values: All 1.0 (perfect self-correlation)
- Range: [-1, 1] (perfect negative to perfect positive correlation)
- This is a symmetric matrix because correlation between A and B equals
correlation between B and A

ANSWER: Relationship Analysis
Strongest correlations (showing linear relationships):
1. B7 vs B8A: 0.992 (Very strong positive)
2. B8 vs B8A: 0.99 (Very strong positive)
3. B7 vs B8: 0.989 (Very strong positive)
4. B6 vs B7: 0.98 (Very strong positive)
5. B2 vs B3: 0.979 (Very strong positive)
6. B6 vs B8A: 0.975 (Very strong positive)
7. B6 vs B8: 0.974 (Very strong positive)
8. B3 vs B4: 0.962 (Very strong positive)

ANSWER: What Correlation Coefficients Reveal
- Values close to 1.0: Strong positive linear relationship (bands
increase/decrease together)
- Values close to -1.0: Strong negative linear relationship (one band increases,
other decreases)
- Values close to 0.0: No linear relationship (bands are independent)
- High correlations often indicate similar spectral properties or atmospheric
effects

Band Group Analysis:
```

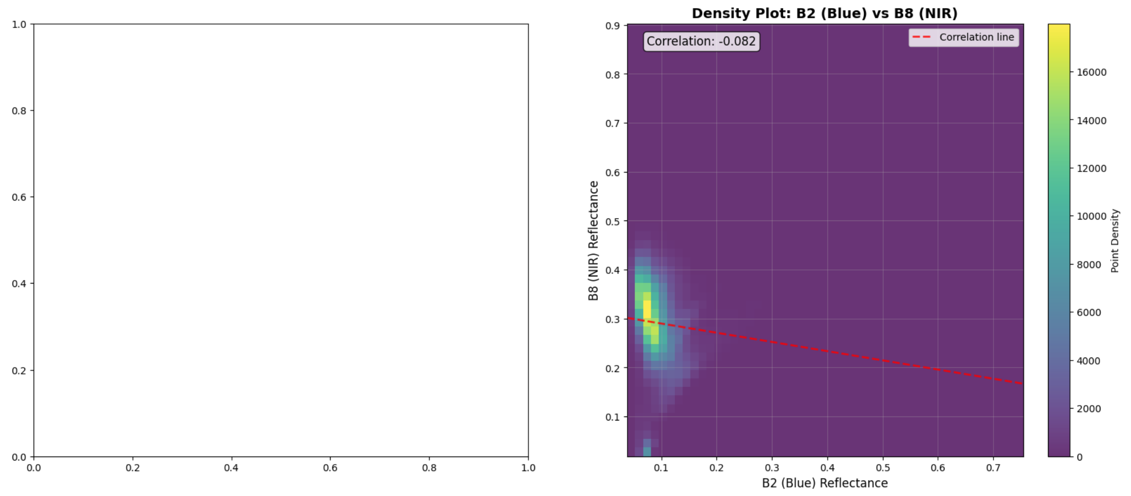
```
Visible bands (B2-B4): Average correlation = 0.966 (very high - expected)
Red edge bands (B5-B7, B8A): Average correlation = 0.638 (very high - expected)
SWIR bands (B11-B12): Average correlation = 0.875 (very high - expected)
Plotting correlation matrix…
Correlation matrix saved as correlation_matrix.png
```

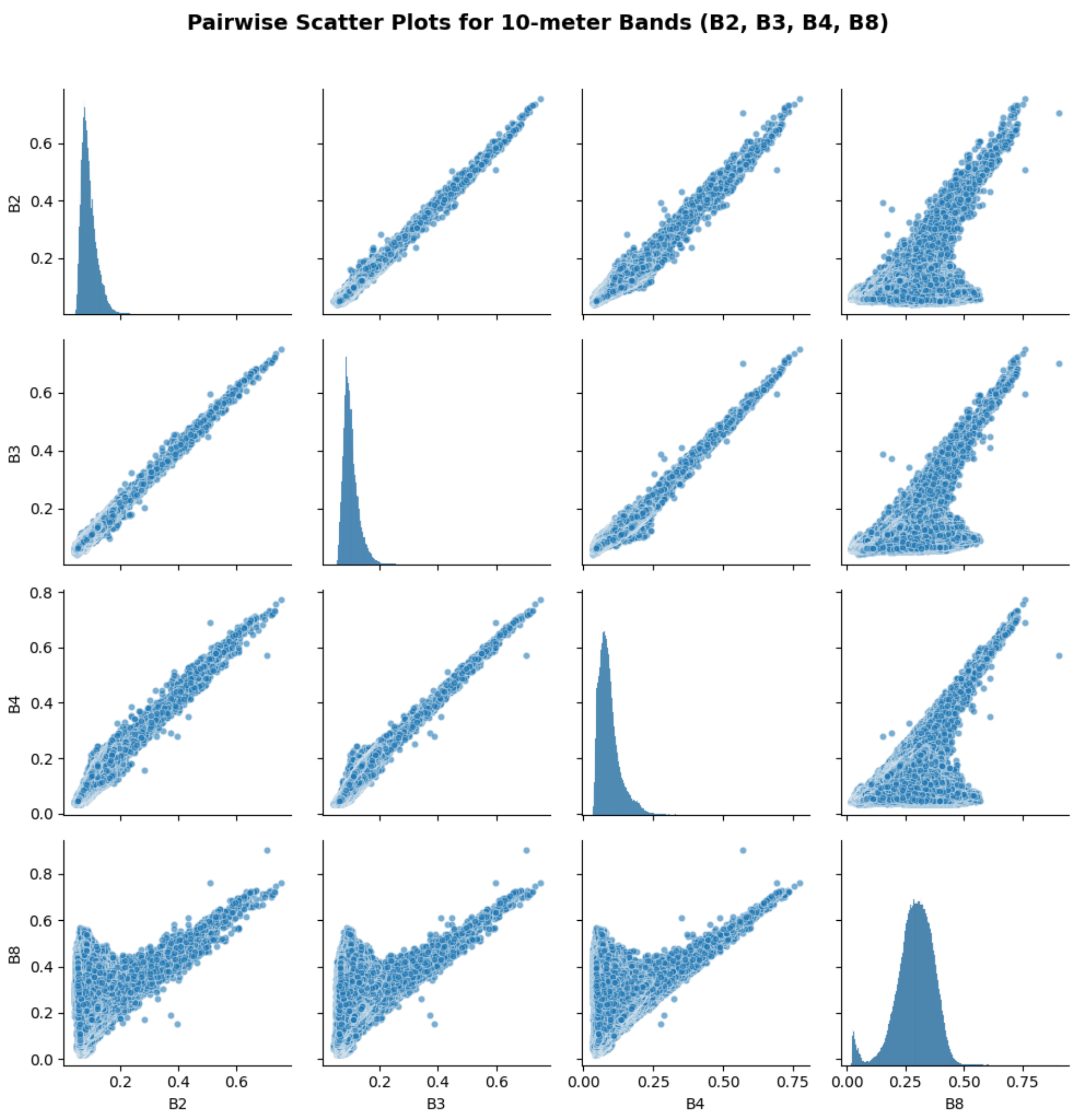


	B1	B2	B3	B4	B5	B6	B7	B8	B8A	B9	B11	B12
B1												
B2	0.86											
B3	0.83	0.98										
B4	0.81	0.96	0.96									
B5	0.76	0.88	0.94	0.92								
B6	0.05	0.07	0.22	0.05	0.39							
B7	-0.07	-0.07	0.07	-0.10	0.25	0.98						
B8	-0.08	-0.08	0.06	-0.10	0.24	0.97	0.99					
B8A	-0.09	-0.09	0.05	-0.11	0.24	0.97	0.99	0.99				
B9	0.11	0.06	0.16	0.02	0.31	0.81	0.80	0.80	0.81			
B11	0.50	0.59	0.70	0.71	0.86	0.55	0.44	0.44	0.46	0.42		
B12	0.64	0.73	0.77	0.86	0.85	0.14	0.02	0.02	0.02	0.09	0.88	

```
PART (B): PAIRWISE SCATTER PLOTS AND DENSITY PLOTS
==================================================
Requirements:
1. Select 10-meter bands: B2 (Blue), B3 (Green), B4 (Red), B8 (NIR)
2. Define function for subplots:
   - Subplot 1: Pairwise scatter plots
   - Subplot 2: Density plot
3. Analyze patterns and describe trends/clusters
Getting 10-meter bands…
10-meter bands extracted: B2, B3, B4, B8
```

```
Creating pairwise correlation plots…
ANSWER TO QUESTION: Creating function with two subplots as requested

ANSWER: Function creates two subplots:
1. Subplot 1: Pairwise scatter plots for all 10-meter band pairs
2. Subplot 2: Density plot showing data concentration

Creating Subplot 1: Pairwise Scatter Plots
This shows scatter plots for every possible pair of 10-meter bands (B2, B3, B4,
B8)
Pairwise scatter matrix saved as pairwise_scatter_matrix.png

Creating Subplot 2: Density Plot
This shows the density/concentration of scatter plot points
Density plots are useful for visualizing concentrated data areas
Pairwise correlation plots saved as pairwise_correlation_plots.png
```





```
ANSWER TO QUESTION: Analyze patterns in pairwise plots
This function analyzes the observed patterns and describes significant
trends/clusters

Correlation Analysis for 10-meter Bands:
Band Pair               Correlation     Strength        Interpretation
----------------------------------------------------------------------
B2 vs B3                0.979           Very Strong             Bands highly
related
B2 vs B4                0.956           Very Strong             Bands highly
related
```

```
B2 vs B8                -0.082          Weak            Bands mostly independent
B3 vs B4                0.962           Very Strong             Bands highly
related
B3 vs B8                0.064           Weak            Bands mostly independent
B4 vs B8                -0.105          Weak            Bands mostly independent

ANSWER: Significant Trends Found
Strongest correlation: B2 vs B3 (0.979)
Weakest correlation: B3 vs B8 (0.064)

ANSWER: Clusters and Trends Analysis
1. VISIBLE BAND CLUSTER:
   - Visible bands (B2-B4): Average correlation = 0.966
   - These bands show high correlation because they're close in wavelength
   - They respond similarly to vegetation and soil properties

2. NIR RELATIONSHIPS:
   - NIR correlations: Average correlation = -0.041
   - NIR (B8) shows different behavior from visible bands
   - This is expected because NIR responds differently to vegetation

3. SPECTRAL DISTANCE PATTERN:
   - B2 vs B3 (adjacent wavelengths): High correlation expected
   - B2 vs B4 (distant wavelengths): Lower correlation expected
   - B2 vs B8 (very distant wavelengths): Lowest correlation expected
   - This pattern confirms that spectral distance affects correlation

ANSWER: What These Patterns Mean
- High correlations in visible bands: Similar atmospheric scattering and surface
properties
- Lower NIR correlations: Different spectral response to vegetation and water
content
- Spectral distance effect: Adjacent wavelengths more correlated than distant
ones
- These patterns are typical for multispectral remote sensing data
```

[ ]:
```python
print("=" * 60)
print("My thoughts on the results which i got !")
print("=" * 60)

print("\nWhat I learned from this analysis:")
print("1. The correlation matrix shows that bands close in wavelength are␣
  ↪highly correlated")
print("   - This makes sense because they're measuring similar things")
print("   - B2, B3, B4 (visible bands) all correlate strongly with each other")
print("   - B6, B7, B8, B8A (red edge/NIR) also correlate strongly")
print("\n2. The matrix is symmetric because correlation is bidirectional")
print("   - If B2 correlates with B3, then B3 correlates with B2 by the same␣
↪amount")
print("   - The diagonal is all 1.0 because each band correlates perfectly with␣
↪itself")

print("\n3. The pairwise scatter plots reveal interesting patterns:")
print("   - Visible bands (B2-B4) show tight clusters - they're very similar")
print("   - NIR (B8) shows different patterns - it behaves differently")
print("   - This is expected because NIR responds to vegetation differently␣
↪than visible light")

print("\n4. The density plots show where most data points are concentrated:")
print("   - Most pixels cluster in certain reflectance ranges")
print("   - This tells us about the typical surface types in Rochester")
print("   - Urban areas, vegetation, water, etc. have different reflectance␣
↪patterns")

print("\n5. Why this analysis is useful:")
print("   - Helps understand which bands provide similar information")
print("   - Identifies redundant bands (highly correlated ones)")
print("   - Shows which bands are unique and provide different information")
print("   - Useful for band selection in classification or other analyses")

print("\n6. What surprised me:")
print("   - Some bands have negative correlations (like B1 vs B7)")
print("   - This suggests they respond oppositely to certain surface features")
print("   - The correlations aren't always what I'd expect based on wavelength␣
↪alone")

print("\n7. Real-world applications:")
print("   - This type of analysis helps in satellite image processing")
print("   - Can be used to select optimal band combinations")
print("   - Helps understand data quality and relationships")
print("   - Useful for atmospheric correction and noise reduction")

print("\nOverall, this was a good exercise in understanding multispectral data!
↪")
print("The results make sense from a remote sensing perspective.")
```

```
[ ]:
```