LAI

December 29, 2024

Import modules

[1]:
```
import os
import rasterio
import geopandas as gpd
import numpy as np
import matplotlib.pyplot as plt
from rasterio.plot import show
from mapclassify import NaturalBreaks
from rasterio.merge import merge
from datetime import datetime
from netCDF4 import Dataset
import cartopy.crs as ccrs
import cartopy.feature as cfeature
from datetime import datetime, timedelta
```

Merge the individual tiff files to create NC file

[2]:
```
# Define the directory containing the LAI images
lai_dir = r'C:\test_env\VegetationAnalysis\LAI'

# List all the files in the directory
lai_files = [os.path.join(lai_dir, f) for f in os.listdir(lai_dir) if f.
↪endswith('.tif')]

# Function to replace '02' with '16' in the date part of the filename
def replace_date(filename):
    date_str = filename.split('\\')[-1].split('.')[0]
    if date_str.endswith('02'):
        date_str = date_str[:-2] + '16'
    return date_str

# Read and merge the images
src_files_to_mosaic = []
dates = []

for file in lai_files:
    date_str = replace_date(file)
```

```
    dates.append(datetime.strptime(date_str, '%Y%m%d'))
    src = rasterio.open(file)
    src_files_to_mosaic.append(src)

# Create an empty array to hold the mosaic data
fill_value = -3.4028235e+38
mosaic_shape = (len(dates), src_files_to_mosaic[0].height,␣
 ↪src_files_to_mosaic[0].width)
mosaic = np.full(mosaic_shape, fill_value, dtype=np.float32)

# Fill the mosaic array with data from each image
for i, src in enumerate(src_files_to_mosaic):
    data = src.read(1)
    data[data == fill_value] = np.nan  # Replace fill value with NaN for masking
    mosaic[i] = data

# Create a NetCDF file
nc_path = r'C:\test_env\VegetationAnalysis\merged\lai_mosaic.nc'
nc_file = Dataset(nc_path, 'w', format='NETCDF4')

# Create dimensions
time_dim = nc_file.createDimension('time', len(dates))
lat_dim = nc_file.createDimension('lat', mosaic.shape[1])
lon_dim = nc_file.createDimension('lon', mosaic.shape[2])

# Create variables
times = nc_file.createVariable('time', np.float64, ('time',))
latitudes = nc_file.createVariable('lat', np.float32, ('lat',))
longitudes = nc_file.createVariable('lon', np.float32, ('lon',))
lai = nc_file.createVariable('lai', np.float32, ('time', 'lat', 'lon',),␣
 ↪fill_value=fill_value)

# Assign data to variables
times[:] = np.array([date.timestamp() for date in dates])
latitudes[:] = np.linspace(src.bounds.top, src.bounds.bottom, mosaic.shape[1])
longitudes[:] = np.linspace(src.bounds.left, src.bounds.right, mosaic.shape[2])
lai[:, :, :] = mosaic

# Add attributes
nc_file.description = 'LAI Mosaic Time Series'
nc_file.history = 'Created ' + datetime.now().strftime('%Y-%m-%d %H:%M:%S')
nc_file.source = 'LAI data from raster files'

# Close the NetCDF file
nc_file.close()

print(f"NetCDF file created and saved to {nc_path}")
```

```
NetCDF file created and saved to
C:\test_env\VegetationAnalysis\merged\lai_mosaic.nc
```

Load the NC file

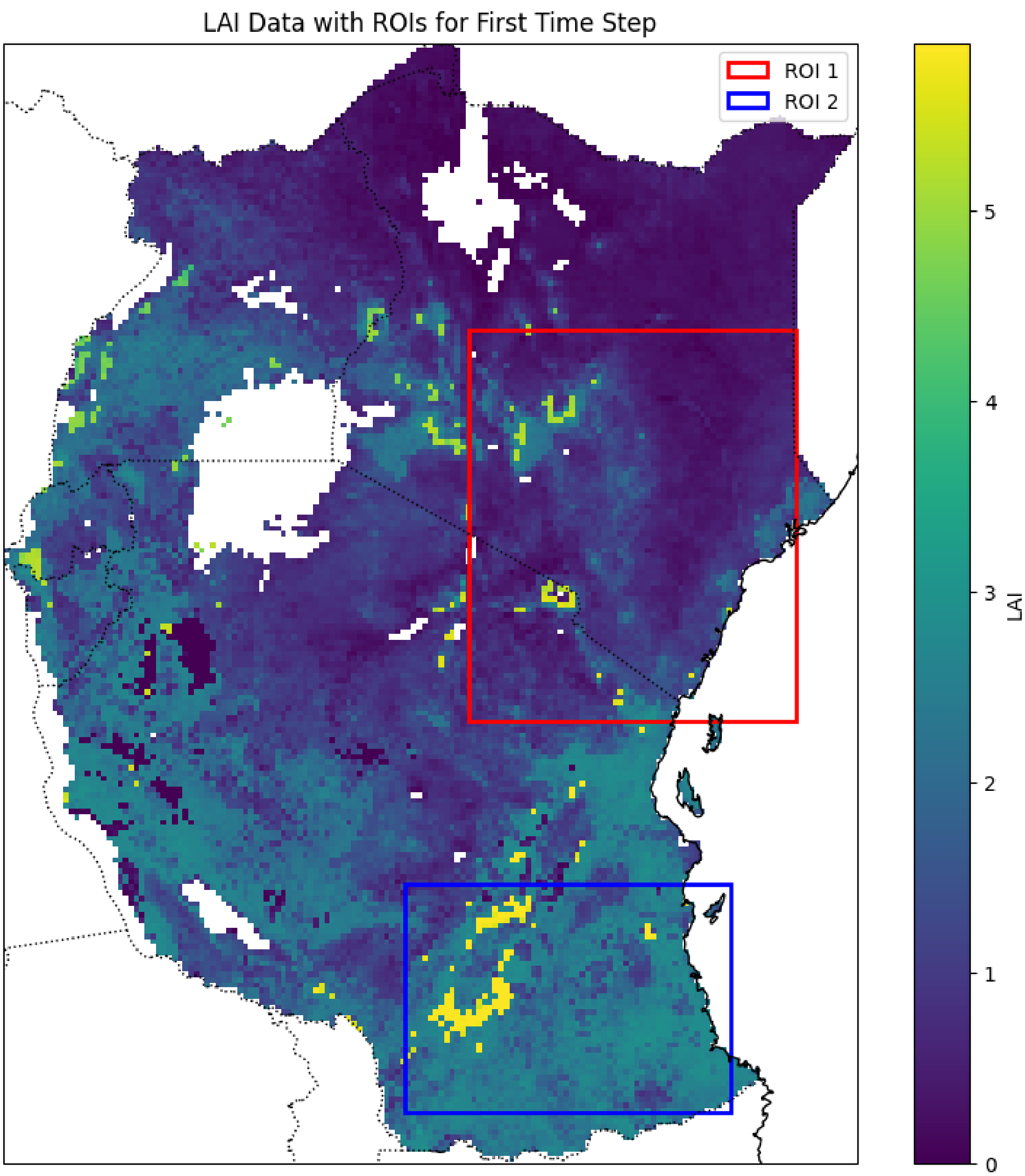
[3]:
```python
# Open the NetCDF file
nc_path = r'C:\test_env\VegetationAnalysis\merged\lai_mosaic.nc'
nc_file = Dataset(nc_path, 'r')
```

Plot ROIs over the first timestep

[4]:
```python
# Extract the variables
times = nc_file.variables['time'][:]
latitudes = nc_file.variables['lat'][:]
longitudes = nc_file.variables['lon'][:]
lai = nc_file.variables['lai'][:]

# Define the ROIs
roi1 = {'lat_min': -5, 'lat_max': 1, 'lon_min': 36, 'lon_max': 41}
roi2 = {'lat_min': -11, 'lat_max': -7.5, 'lon_min': 35, 'lon_max': 40}

# Plot the LAI data for the first time step with ROIs
fig, ax = plt.subplots(figsize=(10, 10), subplot_kw={'projection': ccrs.
 ↪PlateCarree()})
cmap = plt.cm.viridis
cmap.set_bad(color='white')  # Set the color for masked values to white

# Plot the first time step
im = ax.imshow(lai[0, :, :], cmap=cmap, extent=[longitudes.min(), longitudes.
 ↪max(), latitudes.min(), latitudes.max()], origin='upper')
ax.set_title('LAI Data with ROIs for First Time Step')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
fig.colorbar(im, ax=ax, orientation='vertical', label='LAI')

# Add ROIs
ax.add_patch(plt.Rectangle((roi1['lon_min'], roi1['lat_min']), roi1['lon_max']␣
 ↪- roi1['lon_min'], roi1['lat_max'] - roi1['lat_min'], fill=False,␣
 ↪edgecolor='red', linewidth=2, label='ROI 1'))
ax.add_patch(plt.Rectangle((roi2['lon_min'], roi2['lat_min']), roi2['lon_max']␣
 ↪- roi2['lon_min'], roi2['lat_max'] - roi2['lat_min'], fill=False,␣
 ↪edgecolor='blue', linewidth=2, label='ROI 2'))

# Add features
ax.add_feature(cfeature.COASTLINE)
ax.add_feature(cfeature.BORDERS, linestyle=':')

# Add legend
```

```
ax.legend(loc='upper right')

plt.show()
```

```
C:\Users\Rae\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2k
fra8p0\LocalCache\local-packages\Python311\site-
packages\matplotlib\colors.py:744: RuntimeWarning: overflow encountered in
multiply
  xa *= self.N
```



SECTION 1

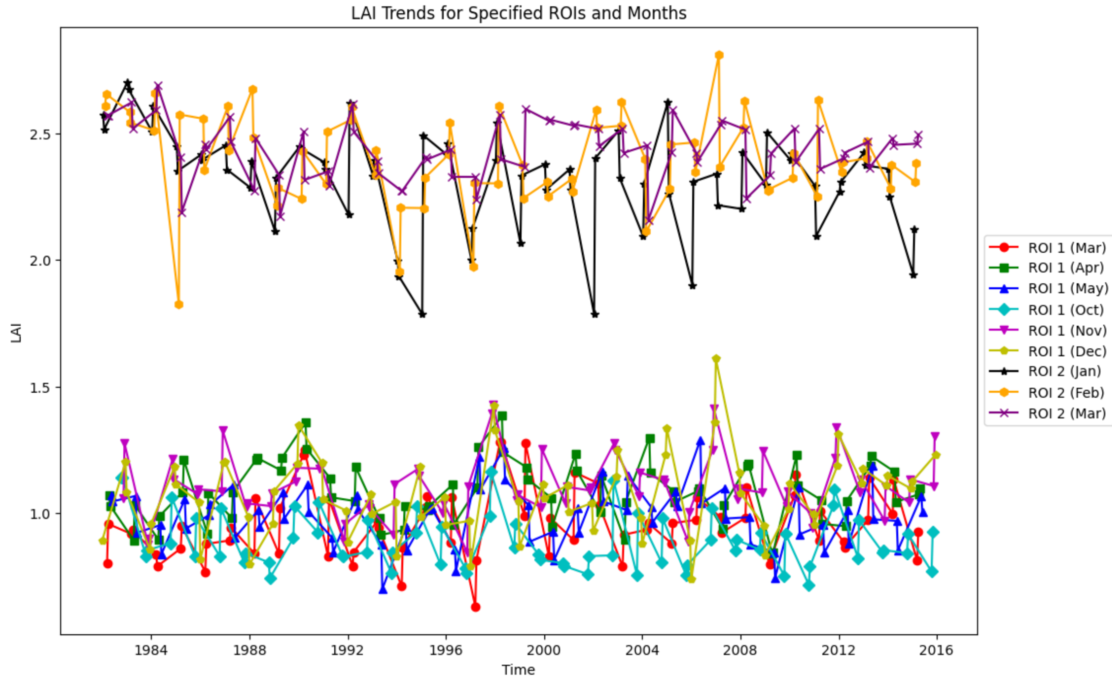
Analysis over ROIs

[5]:
```
# Define the ROIs and months
roi1 = {'lat_min': -5, 'lat_max': 1, 'lon_min': 36, 'lon_max': 41}
roi2 = {'lat_min': -11, 'lat_max': -7.5, 'lon_min': 35, 'lon_max': 40}
months_roi1 = [3, 4, 5, 10, 11, 12]
months_roi2 = [1, 2, 3]

# Function to open and read the NetCDF file
def read_nc_file(nc_path):
    nc_file = Dataset(nc_path, 'r')
    times = nc_file.variables['time'][:]
    latitudes = nc_file.variables['lat'][:]
    longitudes = nc_file.variables['lon'][:]
    lai = nc_file.variables['lai'][:]
    dates = [datetime(1970, 1, 1) + timedelta(seconds=time) for time in times]
    return nc_file, dates, latitudes, longitudes, lai

# Function to calculate spatial anomalies for a given ROI and months
def calculate_spatial_anomalies(dates, latitudes, longitudes, lai, roi, months):
    lat_indices = np.where((latitudes >= roi['lat_min']) & (latitudes <=␣
 ↪roi['lat_max']))[0]
    lon_indices = np.where((longitudes >= roi['lon_min']) & (longitudes <=␣
 ↪roi['lon_max']))[0]
    anomalies = {month: [] for month in months}
    for month in months:
        month_indices = [i for i, date in enumerate(dates) if date.month ==␣
 ↪month]
        month_lai = lai[month_indices, :, :]
        long_term_avg = np.mean(month_lai, axis=0)
        anomalies[month] = month_lai - long_term_avg
    return anomalies, lat_indices, lon_indices

# Function to extract LAI values for a given ROI and months
def extract_lai(dates, latitudes, longitudes, lai, roi, months):
    lat_indices = np.where((latitudes >= roi['lat_min']) & (latitudes <=␣
 ↪roi['lat_max']))[0]
    lon_indices = np.where((longitudes >= roi['lon_min']) & (longitudes <=␣
 ↪roi['lon_max']))[0]
    lai_values = {month: [] for month in months}
    lai_dates = {month: [] for month in months}
    for i, date in enumerate(dates):
        if date.month in months:
            lai_values[date.month].append(np.mean(lai[i, lat_indices, :][:,␣
 ↪lon_indices]))
```

```
            lai_dates[date.month].append(date)
    return lai_dates, lai_values
```

Timeseries over the ROIs

[6]:
```
# Read the NetCDF file
nc_file, dates, latitudes, longitudes, lai = read_nc_file(nc_path)

# Extract LAI values for each ROI
dates_roi1, lai_roi1 = extract_lai(dates, latitudes, longitudes, lai, roi1,␣
 ↪months_roi1)
dates_roi2, lai_roi2 = extract_lai(dates, latitudes, longitudes, lai, roi2,␣
 ↪months_roi2)

# Plot the trends for each month
fig, ax = plt.subplots(figsize=(12, 8))
colors = ['r', 'g', 'b', 'c', 'm', 'y', 'k', 'orange', 'purple']
markers = ['o', 's', '^', 'D', 'v', 'p', '*', 'h', 'x']

# Plot for ROI 1
for i, month in enumerate(months_roi1):
    ax.plot(dates_roi1[month], lai_roi1[month], label=f'ROI 1 ({datetime(1970,␣
 ↪month, 1).strftime("%b")})', color=colors[i], marker=markers[i])

# Plot for ROI 2
for i, month in enumerate(months_roi2):
    ax.plot(dates_roi2[month], lai_roi2[month], label=f'ROI 2 ({datetime(1970,␣
 ↪month, 1).strftime("%b")})', color=colors[i + len(months_roi1)],␣
 ↪marker=markers[i + len(months_roi1)])

ax.set_title('LAI Trends for Specified ROIs and Months')
ax.set_xlabel('Time')
ax.set_ylabel('LAI')

# Place the legend outside the graph
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))

plt.show()
```

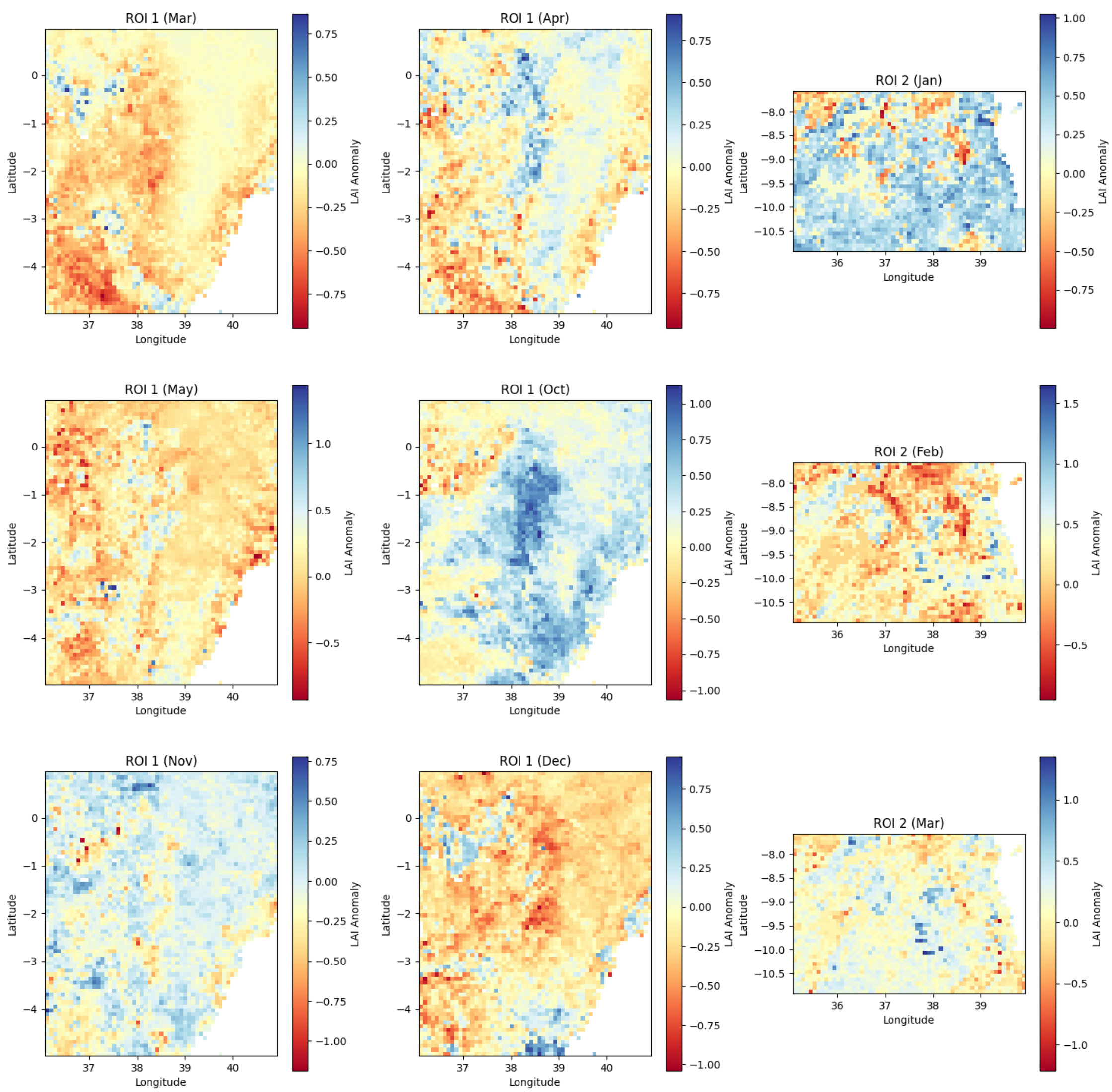


Spatial Anomalies over ROIs

[7]:

```
# Calculate spatial anomalies for each ROI
anomalies_roi1, lat_indices_roi1, lon_indices_roi1 =␣
  ↪calculate_spatial_anomalies(dates, latitudes, longitudes, lai, roi1,␣
  ↪months_roi1)
anomalies_roi2, lat_indices_roi2, lon_indices_roi2 =␣
  ↪calculate_spatial_anomalies(dates, latitudes, longitudes, lai, roi2,␣
  ↪months_roi2)

# Plot the spatial anomalies for each month and region
fig, axes = plt.subplots(3, 3, figsize=(15, 15))
cmap = plt.cm.RdYlBu
cmap.set_bad(color='white')

# Plot for ROI 1
for i, month in enumerate(months_roi1):
    row = i // 2
    col = i % 2
    ax = axes[row, col]
    im = ax.imshow(anomalies_roi1[month][0, lat_indices_roi1, :][:,␣
  ↪lon_indices_roi1], cmap=cmap, extent=[longitudes[lon_indices_roi1].min(),␣
  ↪longitudes[lon_indices_roi1].max(), latitudes[lat_indices_roi1].min(),␣
  ↪latitudes[lat_indices_roi1].max()], origin='upper')
    ax.set_title(f'ROI 1 ({datetime(1970, month, 1).strftime("%b")})')
```

```
    ax.set_xlabel('Longitude')
    ax.set_ylabel('Latitude')
    fig.colorbar(im, ax=ax, orientation='vertical', label='LAI Anomaly')

# Plot for ROI 2
for i, month in enumerate(months_roi2):
    ax = axes[i, 2]
    im = ax.imshow(anomalies_roi2[month][0, lat_indices_roi2, :][:,␣
 ↪lon_indices_roi2], cmap=cmap, extent=[longitudes[lon_indices_roi2].min(),␣
 ↪longitudes[lon_indices_roi2].max(), latitudes[lat_indices_roi2].min(),␣
 ↪latitudes[lat_indices_roi2].max()], origin='upper')
    ax.set_title(f'ROI 2 ({datetime(1970, month, 1).strftime("%b")})')
    ax.set_xlabel('Longitude')
    ax.set_ylabel('Latitude')
    fig.colorbar(im, ax=ax, orientation='vertical', label='LAI Anomaly')

plt.tight_layout()
plt.show()
```



SECTION II

Anaysis for each month over the whole region*

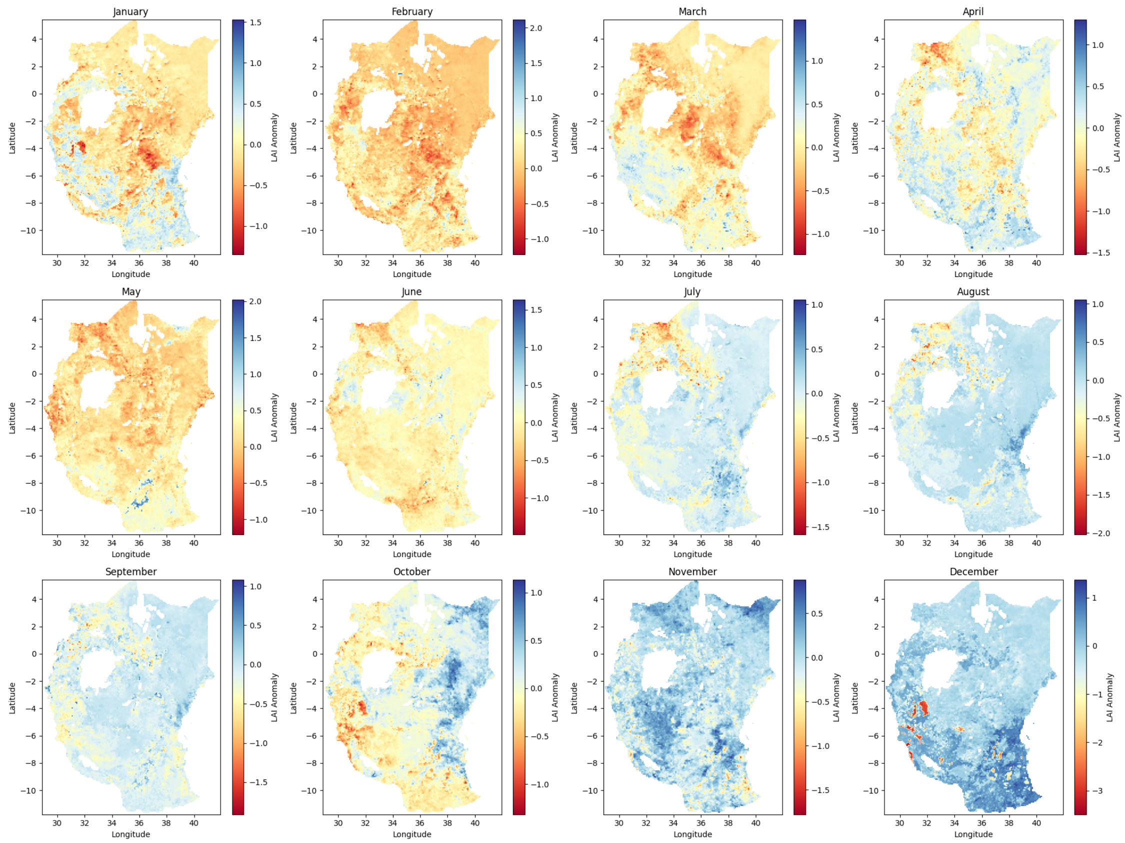
[8]:
```python
# Function to calculate spatial anomalies for each month
def calculate_spatial_anomalies(dates, latitudes, longitudes, lai):
    anomalies = {}
    for month in range(1, 13):
        month_indices = [i for i, date in enumerate(dates) if date.month == month]
        month_lai = lai[month_indices, :, :]
        long_term_avg = np.mean(month_lai, axis=0)
        anomalies[month] = month_lai - long_term_avg
    return anomalies
```

```
# Function to calculate statistics for anomalies
def calculate_statistics(anomalies):
    stats = {}
    for month, anomaly in anomalies.items():
        mean_anomaly = np.mean(anomaly)
        std_anomaly = np.std(anomaly)
        min_anomaly = np.min(anomaly)
        max_anomaly = np.max(anomaly)
        stats[month] = {
            'Mean': mean_anomaly,
            'Standard Deviation': std_anomaly,
            'Min': min_anomaly,
            'Max': max_anomaly
        }
    return stats

# Function to create a table from statistics
def create_table(stats):
    table = "Month\tMean\tStandard Deviation\tMin\tMax\n"
    for month, stat in stats.items():
        table += f"{datetime(1970, month, 1).strftime('%b')}\t{stat['Mean']:.
↪4f}\t{stat['Standard Deviation']:.4f}\t{stat['Min']:.4f}\t{stat['Max']:.
↪4f}\n"
    return table
```

Monthly Spatial Analysis

[9]:
```
# Read the NetCDF file
nc_file, dates, latitudes, longitudes, lai = read_nc_file(nc_path)

# Calculate spatial anomalies for each month
anomalies = calculate_spatial_anomalies(dates, latitudes, longitudes, lai)

# Plot the spatial anomalies for each month
fig, axes = plt.subplots(3, 4, figsize=(20, 15))
cmap = plt.cm.RdYlBu
cmap.set_bad(color='white')

for month in range(1, 13):
    row = (month - 1) // 4
    col = (month - 1) % 4
    ax = axes[row, col]
    im = ax.imshow(anomalies[month][0, :, :], cmap=cmap, extent=[longitudes.
↪min(), longitudes.max(), latitudes.min(), latitudes.max()], origin='upper')
    ax.set_title(f'{datetime(1970, month, 1).strftime("%B")}')
    ax.set_xlabel('Longitude')
```

```
    ax.set_ylabel('Latitude')
    fig.colorbar(im, ax=ax, orientation='vertical', label='LAI Anomaly')

plt.tight_layout()
plt.show()
```



Mothly statistics

[10]:
```
# Read the NetCDF file
nc_file, dates, latitudes, longitudes, lai = read_nc_file(nc_path)

# Calculate spatial anomalies for each month
anomalies = calculate_spatial_anomalies(dates, latitudes, longitudes, lai)

# Calculate statistics for each month
stats = calculate_statistics(anomalies)

# Create a table for the statistics
table = create_table(stats)

# Print the table
```

```
print(table)
```

```
Month   Mean    Standard Deviation      Min     Max
Jan     -0.0000 0.3207  -3.3477 2.8779
Feb     -0.0000 0.3003  -2.7816 2.6776
Mar     0.0000  0.2845  -3.3171 2.6156
Apr     -0.0000 0.2711  -3.0145 2.5864
May     0.0000  0.2467  -3.3723 2.1276
Jun     0.0000  0.2177  -2.6523 2.3615
Jul     -0.0000 0.2001  -2.6338 2.6258
Aug     -0.0000 0.1962  -3.1355 2.9825
Sep     -0.0000 0.2063  -3.3743 2.2776
Oct     0.0000  0.2458  -2.2122 2.7381
Nov     0.0000  0.2909  -2.5812 2.8762
Dec     0.0000  0.3490  -3.5729 3.1615
```

SECTION III

Seasonal Analysis over the whole region

[11]:
```
# Function to calculate spatial anomalies for each season
def calculate_seasonal_anomalies(dates, latitudes, longitudes, lai):
    seasons = {
        'NDJ': [11, 12, 1],
        'MAM': [3, 4, 5],
        'JJA': [6, 7, 8]
    }
    anomalies = {}
    for season, months in seasons.items():
        season_indices = [i for i, date in enumerate(dates) if date.month in months]
        season_lai = lai[season_indices, :, :]
        long_term_avg = np.mean(season_lai, axis=0)
        anomalies[season] = season_lai - long_term_avg
    return anomalies

# Function to calculate statistics for anomalies
def calculate_statistics(anomalies):
    stats = {}
    for season, anomaly in anomalies.items():
        mean_anomaly = np.mean(anomaly)
        std_anomaly = np.std(anomaly)
        min_anomaly = np.min(anomaly)
        max_anomaly = np.max(anomaly)
        stats[season] = {
            'Mean': mean_anomaly,
            'Standard Deviation': std_anomaly,
```

```
            'Min': min_anomaly,
            'Max': max_anomaly
        }
    return stats

# Function to create a table from statistics
def create_table(stats):
    table = "Season\tMean\tStandard Deviation\tMin\tMax\n"
    for season, stat in stats.items():
        table += f"{season}\t{stat['Mean']:.4f}\t{stat['Standard Deviation']:.
 ↪4f}\t{stat['Min']:.4f}\t{stat['Max']:.4f}\n"
    return table
```

Statistical output

[12]:

```
# Read the NetCDF file
nc_file, dates, latitudes, longitudes, lai = read_nc_file(nc_path)

# Calculate seasonal anomalies
seasonal_anomalies = calculate_seasonal_anomalies(dates, latitudes, longitudes,␣
 ↪lai)

# Calculate statistics for each season
seasonal_stats = calculate_statistics(seasonal_anomalies)

# Create a table for the statistics
seasonal_table = create_table(seasonal_stats)

# Print the table
print(seasonal_table)
```

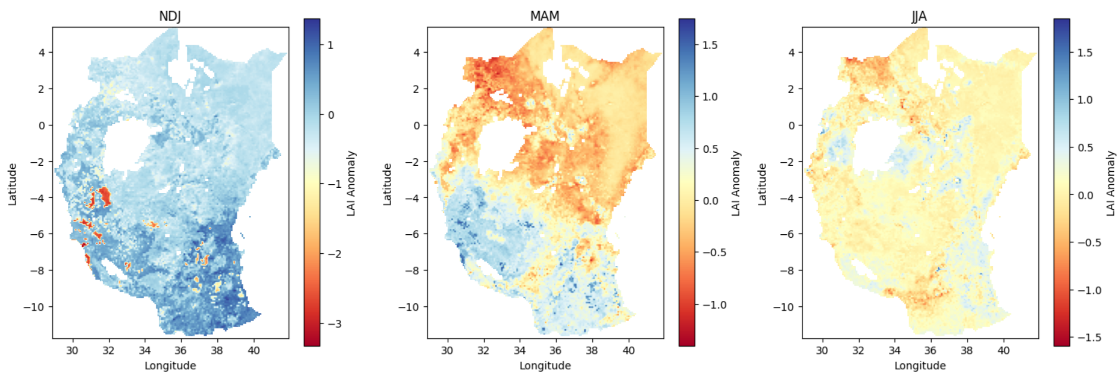
```
Season  Mean    Standard Deviation      Min     Max
NDJ     0.0000  0.3843  -3.4493 3.3029
MAM     0.0000  0.3453  -3.5092 3.0007
JJA     -0.0000 0.2291  -2.9408 3.3310
```

Seasonal spatial anomalies

[13]:

```
# Plot the seasonal anomalies
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
cmap = plt.cm.RdYlBu
cmap.set_bad(color='white')

seasons = ['NDJ', 'MAM', 'JJA']
for i, season in enumerate(seasons):
    ax = axes[i]
```

```
    im = ax.imshow(seasonal_anomalies[season][0, :, :], cmap=cmap, extent=[longitudes.min(), longitudes.max(), latitudes.min(), latitudes.max()], origin='upper')
    ax.set_title(f'{season}')
    ax.set_xlabel('Longitude')
    ax.set_ylabel('Latitude')
    fig.colorbar(im, ax=ax, orientation='vertical', label='LAI Anomaly')

plt.tight_layout()
plt.show()
```



Close the NC file

[14]:
```
nc_file.close()
```

[ ]:
```

```