MVP 1

Advanced Geocomputation

Matt Braaksma

Table of contents

Comparing Land Use Data: OSM vs NLCD

Description

This Minimum Viable Product (MVP) focuses on comparing land use data from OpenStreetMap (OSM) with remote sensing land use/land cover data from the National Land Cover Database (NLCD) for Lakeville, Minnesota. The comparison involves the following key tasks:

- Access OSM Landuse Data: Using the `osmnx` module, OSM landuse polygons will be extracted specifically for Lakeville, MN.
- Rasterization of OSM Polygons: The `rasterio` module will be used to convert OSM landuse polygons into raster format, making them compatible with the NLCD dataset for pixel-by-pixel comparison.
- Class Correspondence: A mapping will be created to align OSM land use categories with the corresponding NLCD land use/land cover categories.
- Comparison with NLCD Data: After spatially and categorically aligning the two datasets, comparisons will be made between the OSM and NLCD landuse representations for the Lakeville region.

By using this methodology, the MVP will evaluate how OSM landuse data compares with NLCD remote sensing data in Lakeville, MN.

Code

1. Import modules

Import the necessary data, geospatial, and visualization modules.

```
# Standard Library Imports
import os
import numpy as np
import pandas as pd
import warnings

# Data Visualization Imports
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
from matplotlib.patches import Patch
import seaborn as sns

# Geospatial Data Imports
from osgeo import gdal, ogr
import geopandas as gpd
import rasterio
from rasterio.features import rasterize
import pygeoprocessing as pygeo
import osmnx as ox

# Set dir
# Define the path to your working directory
data_dir = '../../../base_data/advgeocomp2024/mvp01/data'
os.makedirs(data_dir, exist_ok=True)

# Suppress specific warnings from shapely (osmnx output)
warnings.filterwarnings("ignore", category=RuntimeWarning, module="shapely")
```

2a. NLCD Data

The NLCD data is for the entire continental United States, so we need to clip it to to our area of interest.

Download Lakeville, MN Borders from OSM

```
# Get the geographic boundary for Lakeville, Minnesota
gdf_lakeville = ox.geocode_to_gdf("Lakeville, Minnesota, USA")

# Export
gdf_lakeville_path = os.path.join(data_dir, 'lakeville.gpkg')
gdf_lakeville.to_file(gdf_lakeville_path, driver='GPKG')
```

Import NLCD Raster and Clip to Lakeville

```
# File paths
nlcd_path = '/Volumes/T7 Touch/Research/Reservation_Land/NLCD/raw/nlcd_2021_land_cover_l48_2
input_vector = gdf_lakeville_path
nlcd_lakeville_path = os.path.join(data_dir, 'nlcd_lakeville.tif')

if not os.path.exists(nlcd_lakeville_path):
    # Open the raster
    raster = gdal.Open(nlcd_path)

    # Use gdal.Warp to clip raster with the vector file
    gdal.Warp(
        nlcd_lakeville_path,
        raster,
        format='GTiff',
        cutlineDSName=input_vector,  # Path to vector file
        cropToCutline=True,          # Only crop the raster to the extent of the    vector
        dstNodata=-9999,             # Set nodata value for the output raster
    )

    # Close the dataset
    raster = None
```

2b. OSM Data

Import and Clean OSM

```
# Get OSM landuse data
gdf_landuse_full = ox.features_from_place("Lakeville, Minnesota, USA", tags = {"landuse": Tr

# Keep only polygons (Polygon or MultiPolygon)
gdf_landuse = gdf_landuse_full[gdf_landuse_full['geometry'].geom_type.isin(['Polygon', 'Mult

# Drop all columns except 'geometry' and 'landuse'
gdf_landuse = gdf_landuse[['geometry', 'landuse']]

# Reset index to remove the multi-index structure
gdf_landuse = gdf_landuse.reset_index(drop=True)
```

Create Class Correspondence

```
# Define the mapping from landuse to landuse_esa
landuse_to_landuse_esa = {
    "grass": 21,                # Developed, Open Space
    "residential": 22,          # Developed, Low Intensity
    "commercial": 24,           # Developed High Intensity
```

```
    "forest": 41,                # Deciduous Forest
    "farmland": 82,              # Cultivated Crops
    "industrial": 24,           # Developed High Intensity
    "farmyard": 82,              # Cultivated Crops
    "religious": 24,            # Developed High Intensity
    "recreation_ground": 21,    # Developed, Open Space
    "basin": 12,                # Perennial Ice/Snow
    "cemetery": 21,             # Developed, Open Space
    "meadow": 21,               # Developed, Open Space
    "retail": 24,               # Developed High Intensity
    "quarry": 31,               # Barren Land (Rock/Sand/Clay)
    "plant_nursery": 22,         # Developed, Low Intensity
    "brownfield": 82,           # Cultivated Crops
    "greenfield": 82,            # Cultivated Crops
    "construction": 24           # Developed High Intensity
}

# Create a new column 'landuse_esa' in the GeoDataFrame
gdf_landuse['landuse_esa'] = gdf_landuse['landuse'].map(landuse_to_landuse_esa)
```

Rasterize OSM

```
# Load the nlcd raster to get its metadata
with rasterio.open(nlcd_lakeville_path) as src:
    meta = src.meta
    transform = src.transform

# Reproject the GeoDataFrame if necessary
if gdf_landuse.crs != src.crs:
    gdf_landuse = gdf_landuse.to_crs(src.crs)

# Create an empty array for the new raster
out_array = np.zeros((meta['height'], meta['width']), dtype=np.float32)

# Prepare geometries and attributes for rasterization
geometries = [(geom, int(row.landuse_esa)) for _, row in gdf_landuse.iterrows() for geom in

# Rasterize the geometries
rasterized = rasterize(
    geometries,
    out_shape=out_array.shape,
    transform=transform,
    fill=0,  # Fill value for no data
    all_touched=True,  # Rasterize all pixels touched by geometries
    dtype='float32'
```

```
)

# Write the rasterized output to a new file
osm_raster_path = os.path.join(data_dir, 'osm_raster.tif')
with rasterio.open(osm_raster_path, 'w', **meta) as dst:
    dst.write(rasterized, 1)
```

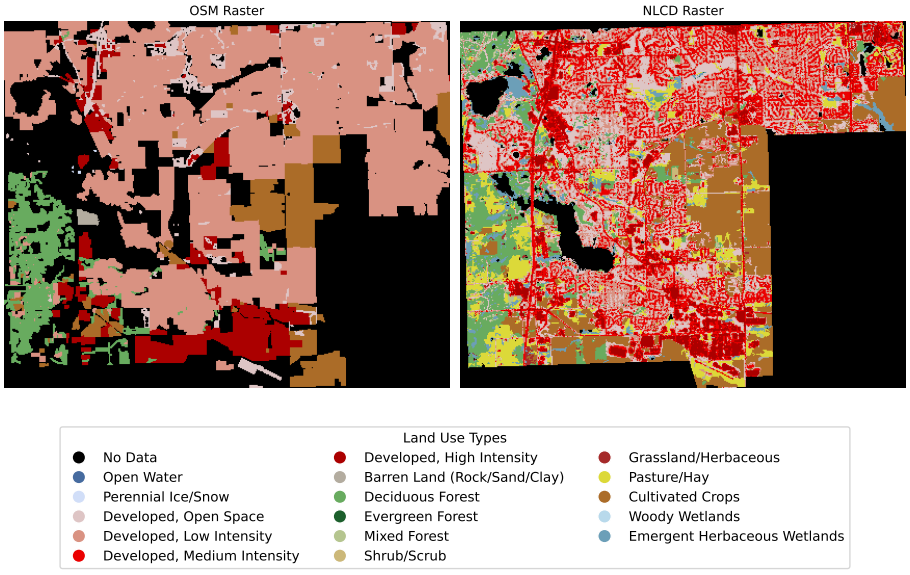
3. Compare Data

OSM vs NLCD Plot

```
# Function to read raster and convert to array
def read_raster_as_array(raster_path):
    dataset = gdal.Open(raster_path)
    array = dataset.ReadAsArray()
    dataset = None
    return array

# Read the raster files
osm_raster = read_raster_as_array(osm_raster_path)
nlcd_raster = read_raster_as_array(nlcd_lakeville_path)

# Define the land use colors and labels
landuse_colors = {
    0: '#000000',      # No data (zero) is black
    11: '#466b9f',
    12: '#d1def8',
    21: '#dec5c5',
    22: '#d99282',
    23: '#eb0000',
    24: '#ab0000',
    31: '#b3ac9f',
    41: '#68ab5f',
    42: '#1c5f2c',
    43: '#b5c58f',
    51: 'lightgray',
    52: '#ccb879',
    71: 'brown',
    81: '#dcd939',
    82: '#ab6c28',
    90: '#b8d9eb',
    95: '#6c9fb8',
}

landuse_mapping = {
    0: "No Data",  # Label for no-data values
```

```
    11: "Open Water",
    12: "Perennial Ice/Snow",
    21: "Developed, Open Space",
    22: "Developed, Low Intensity",
    23: "Developed, Medium Intensity",
    24: "Developed, High Intensity",
    31: "Barren Land (Rock/Sand/Clay)",
    41: "Deciduous Forest",
    42: "Evergreen Forest",
    43: "Mixed Forest",
    51: "Dwarf Scrub",
    52: "Shrub/Scrub",
    71: "Grassland/Herbaceous",
    72: "Sedge/Herbaceous",
    73: "Lichens",
    74: "Moss",
    81: "Pasture/Hay",
    82: "Cultivated Crops",
    90: "Woody Wetlands",
    95: "Emergent Herbaceous Wetlands",
}

# Get the unique land use codes from the raster data
unique_values = np.unique(np.concatenate((osm_raster.flatten(), nlcd_raster.flatten())))

# Filter landuse colors and labels based on the unique land use codes
filtered_landuse_colors = {code: landuse_colors[code] for code in unique_values if code in l
filtered_landuse_mapping = {code: landuse_mapping[code] for code in unique_values if code in

# Create a custom colormap and normalizer
cmap = mcolors.ListedColormap([filtered_landuse_colors[code] for code in filtered_landuse_co
norm = mcolors.BoundaryNorm(unique_values, cmap.N)

# Plot the two rasters side by side
plt.figure(figsize=(12, 6))

# Plot the OSM raster
plt.subplot(1, 2, 1)
plt.imshow(osm_raster, cmap=cmap, norm=norm)  # Apply custom colormap and normalization
plt.title('OSM Raster')
plt.axis('off')

# Plot the NLCD raster
plt.subplot(1, 2, 2)
plt.imshow(nlcd_raster, cmap=cmap, norm=norm)  # Apply custom colormap and normalization
```

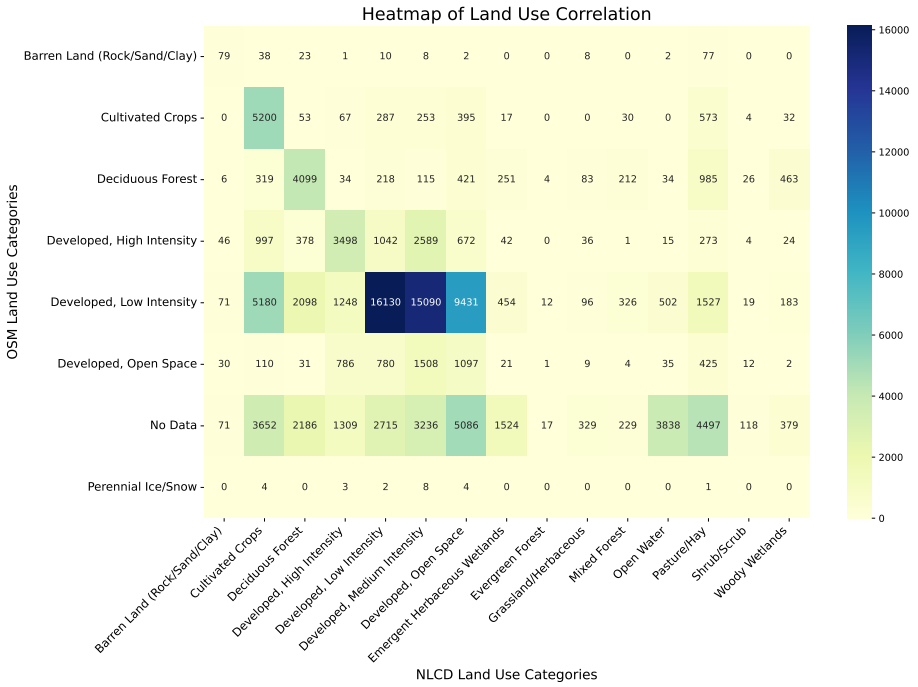
```
plt.title('NLCD Raster')
plt.axis('off')

plt.tight_layout()

# Create legend entries
legend_handles = [plt.Line2D([0], [0], marker='o', color='w', label=filtered_landuse_mapping
                             markerfacecolor=filtered_landuse_colors[key], markersize=10)
                  for key in filtered_landuse_mapping]

# Add legend below the plot
plt.figure(figsize=(12, 2))  # Create a new figure for the legend
plt.legend(handles=legend_handles, title='Land Use Types', loc='center', bbox_to_anchor=(0.5
plt.axis('off')  # Hide axis for the legend
plt.show()
```



OSM vs NLCD Cross Tabulation

```
# Flatten raster arrays for cross tabulation
osm_flat = osm_raster.flatten()
nlcd_flat = nlcd_raster.flatten()

# Map values to labels
osm_labels = [landuse_mapping.get(value, 'Unknown') for value in osm_flat]
nlcd_labels = [landuse_mapping.get(value, 'Unknown') for value in nlcd_flat]
```

```
# Create a DataFrame for cross tabulation
cross_tab_df = pd.DataFrame({'OSM Land Use': osm_labels, 'NLCD Land Use': nlcd_labels})

# Drop 'Unknown' from both land use categories
cross_tab_df = cross_tab_df[cross_tab_df['NLCD Land Use'] != 'Unknown']
cross_tab_df = cross_tab_df[cross_tab_df['NLCD Land Use'] != 'No Data']  # Exclude No Data v

# Create cross tabulation
cross_tab = pd.crosstab(cross_tab_df['OSM Land Use'], cross_tab_df['NLCD Land Use'])

# Create a heatmap
plt.figure(figsize=(14, 10))  # Increase figure size for better spacing
sns.heatmap(cross_tab, annot=True, fmt='g', cmap='YlGnBu', cbar=True)

# Customize the plot
plt.title('Heatmap of Land Use Correlation', fontsize=18)  # Larger title font size
plt.xlabel('NLCD Land Use Categories', fontsize=14)  # Larger x-axis label font size
plt.ylabel('OSM Land Use Categories', fontsize=14)  # Larger y-axis label font size
plt.xticks(rotation=45, ha='right', fontsize=12)  # Rotate x-axis labels and adjust alignmen
plt.yticks(rotation=0, fontsize=12)  # Y-axis label font size

# Adjust layout to make room for labels
plt.subplots_adjust(bottom=0.2, left=0.15)

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



Design Framework

Problem

OpenStreetMap provides layer of land use/land cover data. How does this compare to land use/land cover data derived from remote sensing data?

Solution

Answering this question requires accessing both OSM LULC data and a remote sensing data set. Next a correspondence is required to ensure the proper classes are being compared. Then spatial correlation analysis can be conducted. Finally, other data could be included to determine whether the correlations are driving by other socioeconomic or geographic determinants.

Challenge

Sourcing the data and ensuring that the different pieces work well together.

Spec list

Spec	Value (H, M, L)	Effort (H, M, L)
Get OSM data	H	L

Spec	Value (H, M, L)	Effort (H, M, L)
Get remote sensing data	H	M
Create class correspondence	M	M
Rasterize OSM data	L	H
Compare OSM and remote sensing rasters	H	M
Plot OSM and remote sensing rasters	H	M
Combine with other data	H	H

Success

Comparing OSM and remote sensing rasters is the key metric. If am able able to reach that point, it will be a success.

Reflection

Developing and running this MVP provided valuable insights into the challenges of comparing crowd-sourced data like OpenStreetMap (OSM) with remote sensing datasets such as the National Land Cover Database (NLCD). A significant success was the extraction, rasterization, and alignment of OSM land use polygons with the NLCD data, allowing for landuse comparisons in Lakeville, MN. However, the lack of rural land use data in the OSM dataset resulted in significant data gaps. Despite this limitation, I would classify the MVP as a success because I met the project's objectives and gained an understanding of the key tasks required to complete such analyses.

One of the main takeaways is that the focus of OSM data is *urban landuse*, while remote snesing data is typically focused on *rural landcover*. In this sense the data are better seen as complements with different uses rather than competing data products.