Case Study Solution

September 30, 2022

We have provided two data sets - vector data containing the road network of Vietnam and the provincial boundaries of Vietnam - raster data containing the flood risk of Vietnam

In this assignment, you have to complete the following tasks: - Identify provinces that are vulnerable to flooding - Idenfiy roads that are vulnerable to flooding - Identify provinces that have vulnerable roads

1 Import libraries and read files

[ ]:
```
# Import packages
import geopandas
import folium
import rasterio.mask
from rasterio.plot import show
import numpy as np
```

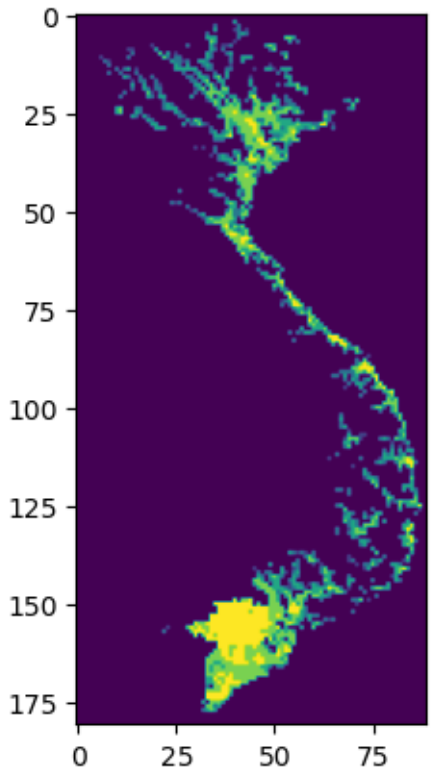
[ ]:
```
# Read geojson files
gdf_bound = geopandas.read_file('Data/case_study/vietnam_bound.geojson')
gdf_road = geopandas.read_file('Data/case_study/vietnam_roadnet.geojson')
dataset = rasterio.open('Data/case_study/geonode__fl1010irmt.tif')
ras = dataset.read(1)
```

1.1 Cleaning and change column

[ ]:
```
gdf_bound = gdf_bound[['NAME_1', 'geometry']]
gdf_bound.columns = ['prov_name', 'geometry']
gdf_road = gdf_road[['geometry']]
```

2 I. Identify provinces that are vulnerable to flooding

[ ]:
```
# Simple plot without mapping
out_image, out_transform = rasterio.mask.mask(dataset, gdf_bound['geometry'],␣
↪crop=True)
out_meta = dataset.meta
show(out_image)
```



[ ]: <AxesSubplot:>

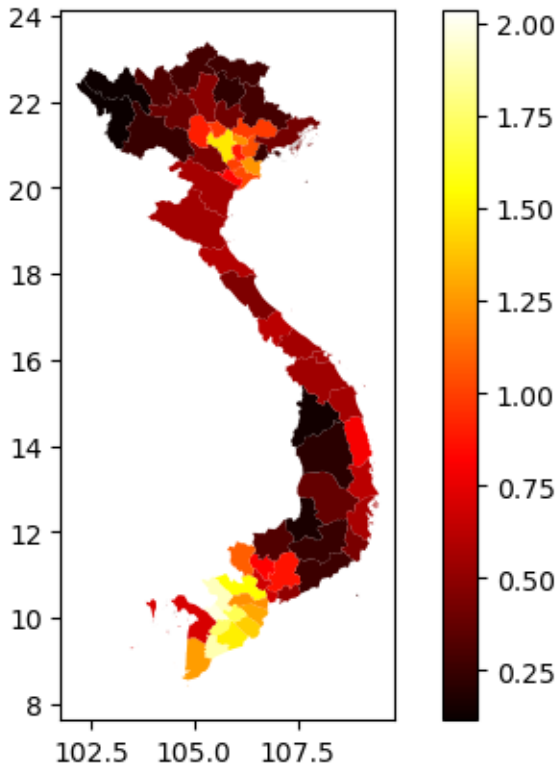
[ ]:
```
# Create a mask for each location and add the mean value as column
res_col = []
for prov_name in gdf_bound['prov_name'].unique():
    df_filt = gdf_bound[gdf_bound['prov_name'] == prov_name]
    gtraster, bound = rasterio.mask.mask(dataset, df_filt['geometry'],␣
 ↪crop=True)
    gdf_bound.loc[gdf_bound['prov_name'] == prov_name, 'avg'] = np.
 ↪mean((gtraster[0]))
```

[ ]:
```
gdf_prov_g = gdf_bound.groupby('prov_name').mean()
# Get 75 percentile of average risk
gdf_prov_g = gdf_prov_g[gdf_prov_g['avg'] > 1.06]
# Get list of high risk
high_average = list(gdf_prov_g.index)
print(f"The high risk areas are {high_average}")
```

```
The high risk areas are ['An Giang', 'Bạc Liêu', 'Bắc Ninh', 'Bến Tre', 'Cà
Mau', 'Cần Thơ', 'Hà Nội', 'Hậu Giang', 'Long An', 'Sóc Trăng', 'Thái Bình',
'Tiền Giang', 'Trà Vinh', 'Tây Ninh', 'Vĩnh Long', 'Đồng Tháp']
```

[ ]:
```
# Simple plot of heat data included
gdf_bound.plot(column='avg', cmap='hot', legend=True)
```

[ ]: <AxesSubplot:>



[ ]:
```
# Create index as string
gdf_bound = gdf_bound.reset_index()
gdf_bound['index'] = gdf_bound['index'].astype(str)
```

[ ]:
```
# Map with folium
geo = geopandas.GeoSeries(gdf_bound.set_index('index')['geometry']).to_json()
m = folium.Map(location=[14.0583, 108.2772], zoom_start=6)
folium.Choropleth(
    geo_data = geo,
    name = 'Choropleth',
    data = gdf_bound,
    columns = ['index','avg'],
    key_on = 'feature.id',
    fill_color = 'Reds',
    fill_opacity = 0.8,
    line_opacity = 1,
    legend_name = 'Average Flood Risk',
```

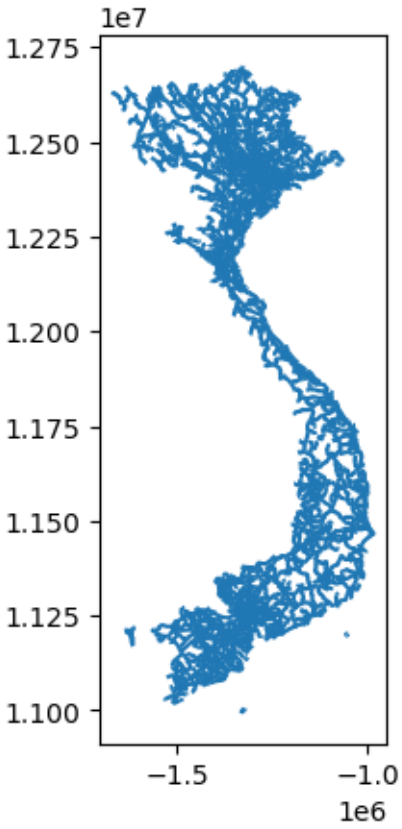
```
    smooth_factor=  1
).add_to(m)

m
```

[ ]: <folium.folium.Map at 0x7fd0f31608d0>

3 II. Idenfiy roads that are vulnerable to flooding

[ ]:
```
# Simple road plot
gdf_road.plot()
```

[ ]: <AxesSubplot:>



[ ]:
```
# create road index column
gdf_road = gdf_road.reset_index()
gdf_road['index'] = gdf_road['index'].astype(str)
```

[ ]:
```
# Set crs
gdf_road = gdf_road.to_crs("EPSG:4326")
```

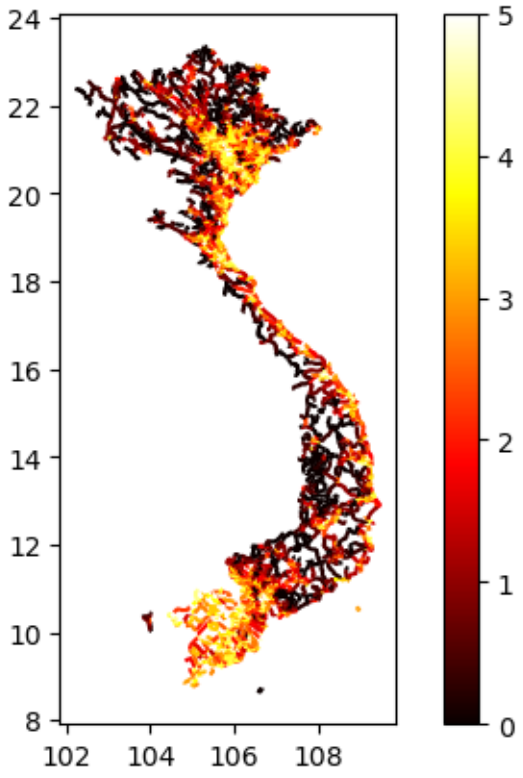
[ ]:
```
# Create a mask for each location and add the mean value as column
res_col = []
for road in gdf_road['index']:
    df_filt = gdf_road[gdf_road['index'] == road]
    gtraster, bound = rasterio.mask.mask(dataset, df_filt['geometry'],␣
 ↪crop=True)
    gdf_road.loc[gdf_road['index'] == road, 'avg'] = np.mean((gtraster[0]))
```

[ ]:
```
# Get 75 percentile of average risk
gdf_road_high = gdf_road[gdf_road['avg'] > 4]
# Get list of high risk
high_average = list(gdf_road_high.index)
print(f"There are {len(high_average)} high risk roads of the {len(gdf_road)}␣
 ↪roads")
```

```
There are 508 high risk roads of the 4535 roads
```

[ ]:
```
# Simple plot of heat data included
gdf_road.plot(column='avg', cmap='hot', legend=True)
```

[ ]:
```
<AxesSubplot:>
```



[ ]:
```
# Road to classify
gdf_road['avg_rounded'] = round(gdf_road['avg']).astype(int)
```

[ ]:
```
# Map with folium
colors = {0: 'Green', 1: 'Green', 2: 'Orange', 3: 'Orange', 4: 'Red', 5: 'Red'}
geo = geopandas.GeoSeries(gdf_road['geometry']).to_json()
m = folium.Map(location=[14.0583, 108.2772], zoom_start=6)
for flood_r in gdf_road['avg_rounded'].unique():
    gdf_f = gdf_road[gdf_road['avg_rounded'] == flood_r].copy()
    geo = geopandas.GeoSeries(gdf_f['geometry']).to_json()
    folium.Choropleth(geo, line_color= colors[flood_r], line_weight=1.4).
 ↪add_to(m)
m
```

[ ]:
```
<folium.folium.Map at 0x7fd0f3304850>
```

4 III. Identify provinces that have vulnerable roads

[ ]:
```
# Calculate weighted average (length times avg for road segment)
gdf_road['weight'] = gdf_road.length * gdf_road['avg']
```

```
/Users/localadmin/opt/anaconda3/envs/TIL6022/lib/python3.7/site-
packages/ipykernel_launcher.py:2: UserWarning: Geometry is in a geographic CRS.
Results from 'length' are likely incorrect. Use 'GeoSeries.to_crs()' to re-
project geometries to a projected CRS before this operation.
```

[ ]:
```
# Merge roads to corresponding boundary
gdf_merge = geopandas.sjoin(gdf_road, gdf_bound[['prov_name','geometry']],␣
 ↪how="left", op='intersects')

#gdf_merge = gdf_road.sjoin(gdf_bound[['prov_name','geometry']], how="left",␣
 ↪predicate='intersects')
```

[ ]:
```
# Drop impossible joins
gdf_merge = gdf_merge[gdf_merge.index_right.notnull()]
```

4.0.1 Plot polygons with weighted road index

[ ]:
```
# Average for each polygon
gdf_merge_pol = gdf_merge[['weight', 'index_right']].groupby('index_right').
 ↪mean().reset_index()
```

[ ]:
```
# Set index as integer and rename column.
gdf_merge_pol['index_right'] = gdf_merge_pol.index_right.astype(int)
gdf_merge_pol.columns = ['index', 'weight']
# Change gdf_bound index type to match gdf_merge index type
gdf_bound['index'] = gdf_bound.index.astype(int)
```

[ ]:
```
# Merge the boundaries with the weight dataframe
gdf_merge_pol = geopandas.pd.merge(gdf_merge_pol, gdf_bound, how='left',␣
 ↪on="index")
gdf_merge_pol = gdf_merge_pol[['geometry', 'weight', 'index', 'prov_name']]
```

[ ]:
```
# Set index as string for plot
gdf_merge_pol['index'] = gdf_merge_pol['index'].astype(str)
```

[ ]:
```
# Map with folium
geo = geopandas.GeoSeries(gdf_merge_pol.set_index('index')['geometry']).
 ↪to_json()
m = folium.Map(location=[14.0583, 108.2772], zoom_start=6)
folium.Choropleth(
    geo_data = geo,
    name = 'Choropleth',
    data = gdf_merge,
    columns = ['index','weight'],
    key_on = 'feature.id',
    fill_color = 'Reds',
    fill_opacity = 0.8,
    line_opacity = 1,
    legend_name = 'Average Flood Risk',
    smooth_factor=  1
).add_to(m)

m
```

[ ]:
```
<folium.folium.Map at 0x7fd0d9a0bf90>
```

[ ]:
```

```