Project 09. Spatial Analysis – a Review

Abstract

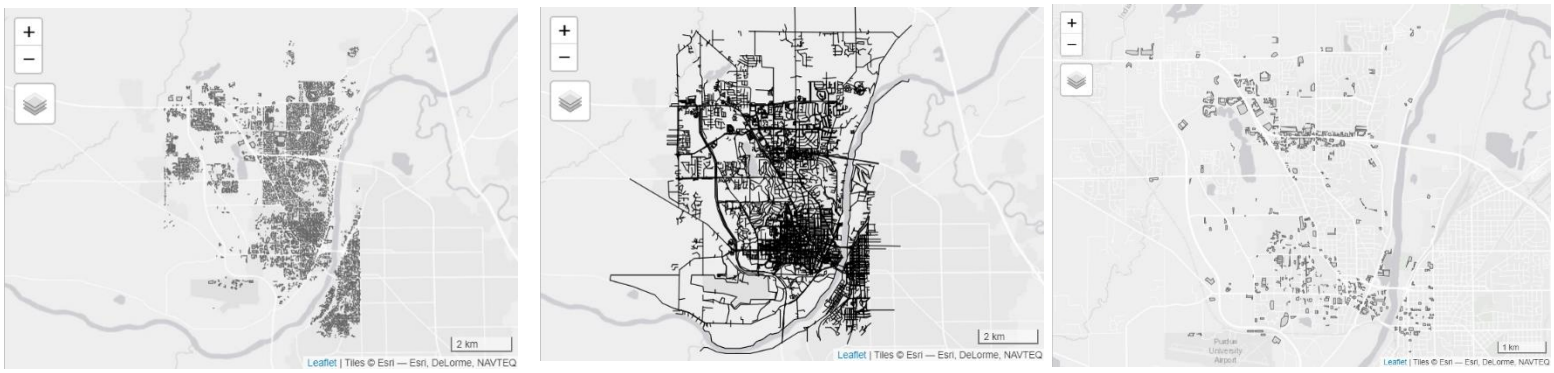
This project entails collecting and processing geospatial data for West Lafayette. It involved extracting features such as buildings, roads, and parking lots (self-interest) from OpenStreetMap (OSM) data and conducting analysis on the length, area, and centroid of buildings. The project also required an analysis of Purdue campus buildings, focusing on the differences between their states in 2014 and the present. Finally, the changes in the buildings on the Purdue campus were mapped out for comparison.

1. Make an R program to read in the following data for West Lafayette of Indiana from OSM website, and store them to your local drive as shapefiles.

The main objective of this was to gather geospatial data for West Lafayette, USA, and perform various data processing and visualization tasks. The process began by defining a bounding box that encompassed the area of interest. Building features were extracted from OpenStreetMap data using the 'osmdata' package. This involved opening a query within the specified bounding box, adding the 'building' feature, and converting the obtained data into a spatial object (sf) for further analysis and visualization. The resulting building polygons were visualized. Next, road features were extracted in a similar manner, but this time using the 'highway' key. The road data was also converted into an sf object and visualized using 'tmap.' In addition, Parking Lot features were also extracted using the 'amenity' key with the value 'parking.' These parking lots were visualized on the map. The following figures show the example of buildings, roads, and parking lots (figure 01, the key coding parts are placed in the appendix).

Figure 01

Building, Roads, and Parking Lots in West Lafayette



After that, the bounding boxes for the building, road, and parking lot data were obtained and printed to understand the spatial extent of each dataset. Figure 02 shows the result of the bounding boxes for all these three features.

Figure 02

Bounding Boxes for All Three Features

```
> print(bbox_buildings)
     xmin      ymin      xmax      ymax
-86.96273  40.39988 -86.88636  40.48680
> # Get bounding box for roads
> bbox_roads <- st_bbox(wl_rds_sf$osm_lines)
> print(bbox_roads)
     xmin      ymin      xmax      ymax
-86.98921  40.39325 -86.86987  40.50416
> # Get bounding box for parking lots
> bbox_parking <- st_bbox(wl_parking_sf$osm_polygons) #
> print(bbox_parking)
     xmin      ymin      xmax      ymax
-86.96188  40.40787 -86.88667  40.48589
>
```

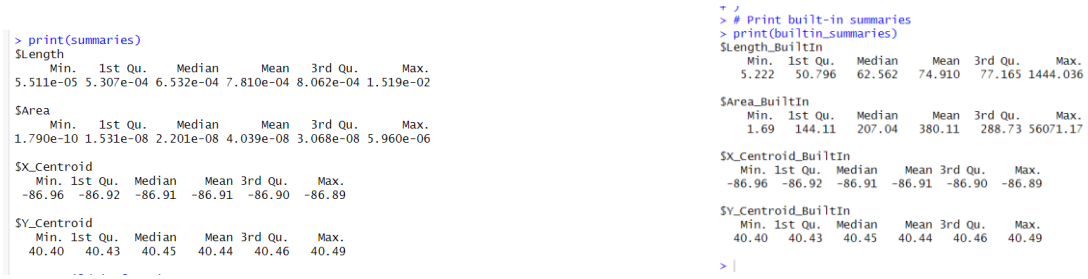
Finally, the gathered geospatial data was saved as shapefiles, including building polygons, road lines, and parking lot polygons. These shapefiles were written to specified file paths with overwrite capability (Reference).

2. **Make an R program to calculate (you need to implement/program the formulas**

During this section, the goal was to calculate the length (perimeter), area, and centroid (x, y coordinates) of each building within the given geospatial dataset by using both manual and built in functions. For the manual calculation, it was accomplished by extracting the coordinates of the building's geometry, and then summing the distances between consecutive vertices using the Euclidean distance formula. For the area, the shoelace formula works by taking the coordinates of the vertices of the polygon and summing the products of the x-coordinates of consecutive vertices with the y-coordinates of the next consecutive vertices, and then subtracting the sum of the products of the y-coordinates of consecutive vertices with the x-coordinates of the next consecutive vertices. The absolute value of this result is half the area of the polygon. For the centroid, it calculated the average x and y coordinates of the building's vertices, providing the center point of the polygon. After the calculation, the results of the manual and built-in for the length, area, and centroid is shown below (Figure 03).

Figure 03

Result of the Manual vs Built-in Codes



```
> print(summaries)
$Length
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
5.511e-05 5.307e-04 6.532e-04 7.810e-04 8.062e-04 1.519e-02

$Area
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
1.790e-10 1.531e-08 2.201e-08 4.039e-08 3.068e-08 5.960e-06

$X_Centroid
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -86.96  -86.92  -86.91  -86.91  -86.90  -86.89

$Y_Centroid
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  40.40   40.43   40.45   40.44   40.46   40.49
```

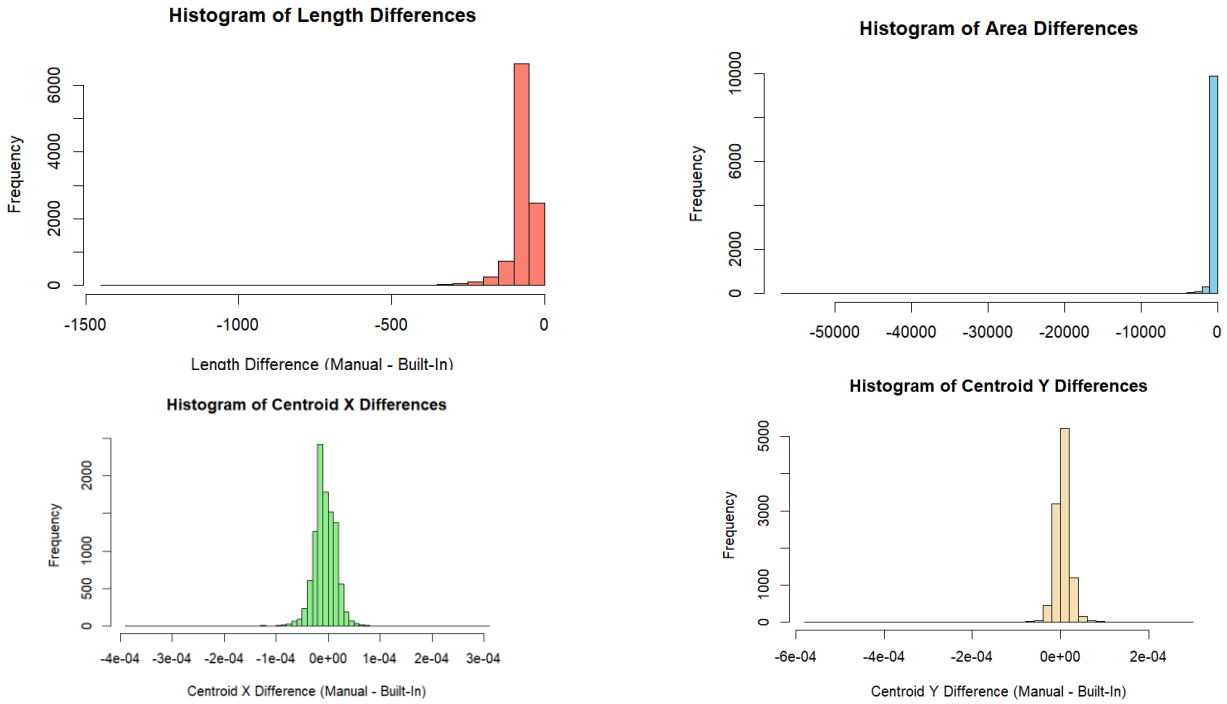
```
> # Print built-in summaries
> print(builtin_summaries)
$Length_BuiltIn
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
   5.222   50.796   62.562   74.910   77.165 1444.036

$Area_BuiltIn
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
    1.69   144.11   207.04   380.11   288.73 56071.17

$X_Centroid_BuiltIn
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -86.96  -86.92  -86.91  -86.91  -86.90  -86.89

$Y_Centroid_BuiltIn
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  40.40   40.43   40.45   40.44   40.46   40.49

>
```

The discrepancies between the manually calculated and built-in function results in R for the geometric properties of buildings can be attributed to various factors. For instance, the manual codes may lack the sophistication to address the complexities that some buildings exhibit. They might calculate perimeter and area based on simpler assumptions that do not account for potential anomalies in shapes. For example, the manual method may assume a planar surface and straight-line distances between points, which could result in significant inaccuracies when applied to spherical coordinates. However, most of the built-in functions are generally fine-tuned to handle intricate geometries, which might include polygons with holes or self-intersecting boundaries.

To conduct further analysis, the differences between the manual and built-in results were also plotted in a histogram. Figure 04 below displays the differences between manual and built-in calculations of geometric properties (length, area, and centroids) of buildings.

Figure 04

Difference Histograms (Manual vs Built-in)



For the histogram of length differences, it reveals a concentration of differences close to zero, with a long tail extending towards the negative values. This suggests that manual length calculations are generally shorter than the built-in calculations. Additionally, there is a significant number of lengths where the discrepancy is quite large.

The area differences histogram shows that although most of the differences are nearly zero, there is still a long tail extending to the negative side. This implies that there are still some buildings with significantly different areas between the manual and built-in calculations.

The differences in the X-coordinate of the centroid exhibit a bell-shaped distribution centered very close to zero. This suggests that, for the X-coordinate of the centroid, the manual and built-in calculations are quite similar for most buildings, with slight variations on either side of zero.

Similarly, the Y-coordinate histogram also forms a bell-shaped distribution centered around zero, but it is more concentrated. This indicates a close agreement between manual and built-in calculations for the Y-coordinate of the centroid, with small discrepancies fairly evenly distributed around zero.

After conducting a spatial analysis of building geometries, it was observed that there are a number of buildings for which the calculated centroid does not lie within the physical boundaries. This phenomenon can occur in buildings with irregular shapes, such as L-shaped structures or some

buildings with holes in the middle. Figure 05 provides examples of such buildings with centroids located outside the boundaries.

Figure 05

Buildings with Centroid Outside

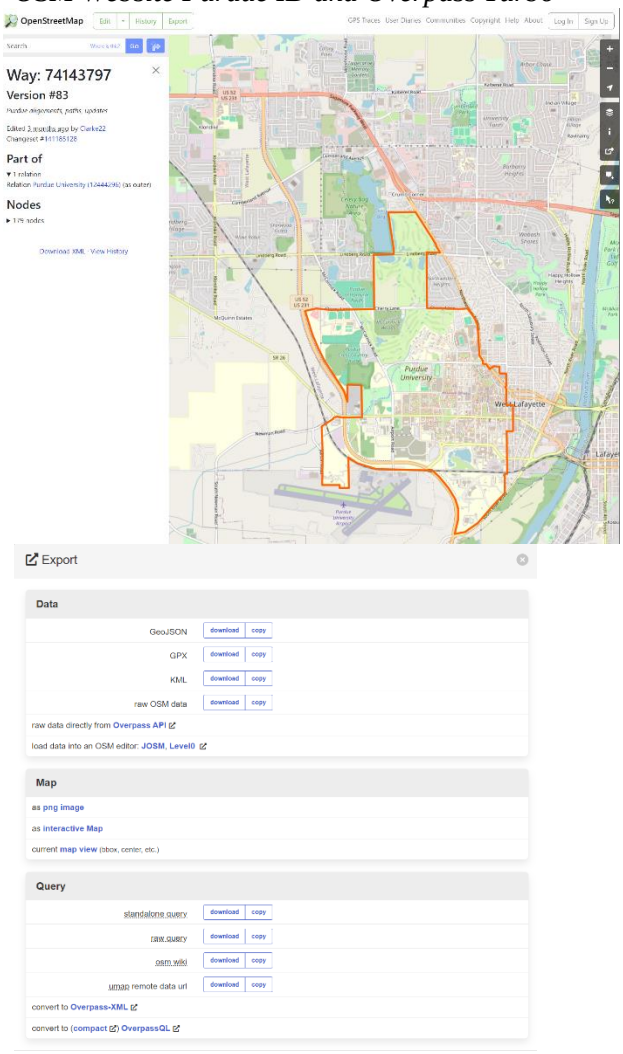
3. Compare the OSM building data and the provided campus building data.

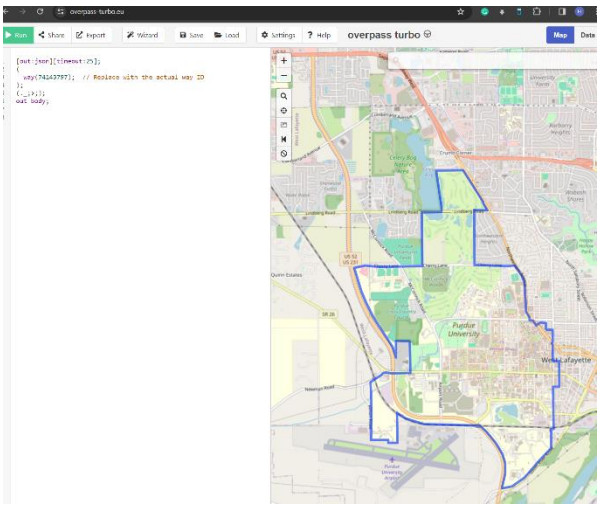
To compare the OSM building data with the provided campus building data, the first step is to establish a boundary that encompasses Purdue's main campus. Using the OpenStreetMap (OSM) website, we located the boundary of Purdue's main campus. Please note that there are four boundaries available, but we selected one specific boundary. Figure 06 displays a screenshot showing the boundary we found on the OSM website.

Subsequently, I used the unique ID (74143797) to this boundary to extract the GeoJSON file through the Overpass Turbo interface, as depicted in Figure 06 on the right. Overpass Turbo is a web-based tool designed for querying OSM data. It allows for precise and customized queries, facilitating the download of a GeoJSON file containing the detailed campus boundary.

Figure 06

OSM Website Purdue ID and Overpass Turbo

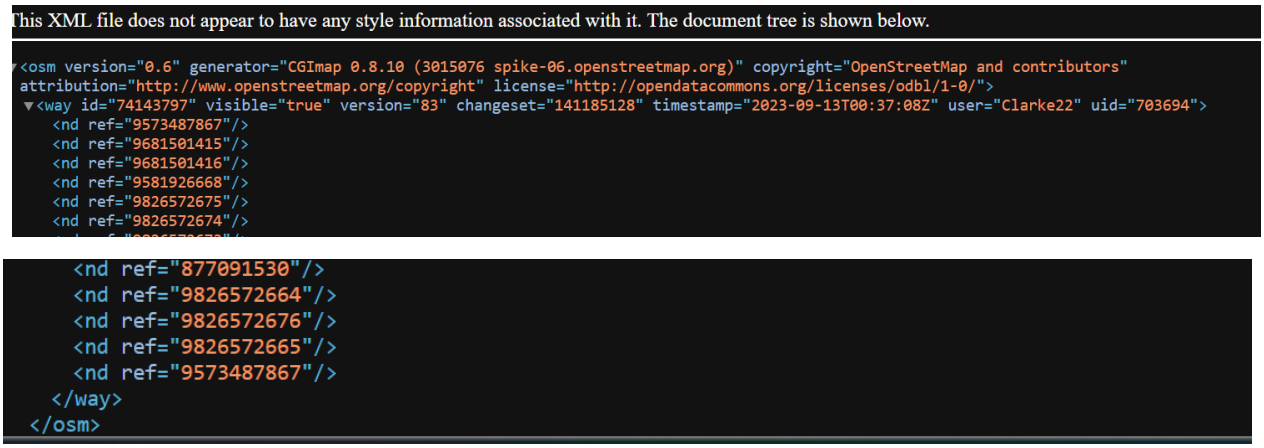




After downloading the boundary, since the downloaded file is a linestring, it is necessary to convert it into polygon for further analysis. To convert the line string to polygon, it is necessary to make sure that the lines are connected head to tail so that it is able to transform to a polygon. One simple way is to check whether the initial and the last nodes are the same. Through the boundary provided by the OSM, we can notice that it has 179 nodes, and we can find out that the initial and the last nodes are the same, meaning that this boundary is able to generate a polygon. Figure 07 shows that the initial node has a reference id of "9573487867" which matches the last node's id.

Figure 07

Node Connection Check (Initial and Final Node ID)



After obtaining the boundary, the next step is to identify changes in the buildings within the Purdue boundary. Several data cleaning and preprocessing steps were performed on the 2014 Purdue campus building data. Initially, the building shapefile was loaded into the R environment. The script then removed any features with empty geometries to ensure that all spatial data is valid. Additionally, further cleaning was necessary to eliminate self-intersections or other geometric issues. Rows lacking a GIS_BID, assumed to be a unique identifier for each building, were filtered out to remove incomplete records. The script also removed duplicates by retaining only distinct GIS_BID entries, ensuring that each building was represented only once in the dataset.

Subsequently, the data was processed to dissolve overlapping polygons based on the GIS_BID. This step aggregated buildings with the same identifier, which might represent different parts or sections of the same building, into a single geometric feature. However, there were still several buildings that contained overlapping, potentially leading to inaccurate results. Figure 08 displays the buildings within the Purdue boundary after data cleaning. The figure also illustrates an example of a building (COREC) for which overlapping could not be dissolved after cleaning.

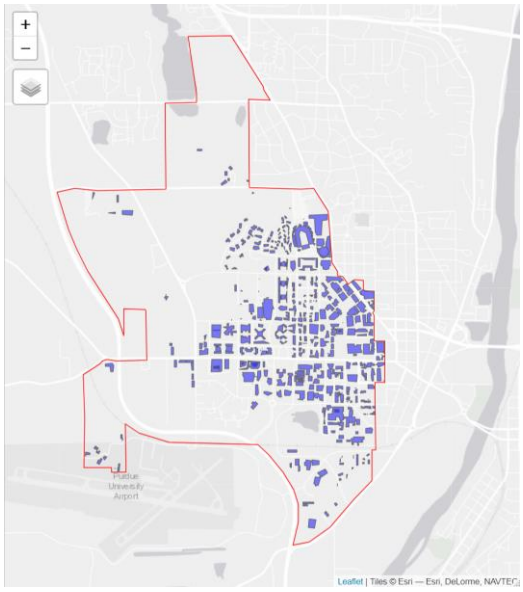
Figure 08

Buildings in Purdue Boundary (2014), Example of Building Unable to Dissolve Overlapping

After cleaning the building data of 2014, the current building data were implemented following the same process: Validation check, remove duplicate polygon with same ID, dissolve overlapping polygons. Figure 09 shows the current building in Purdue boundary from OSM.

Figure 09

OSM Buildings in Purdue Boundary (Current)



After all the data (2014 before vs. OSM current) is implemented and cleaned, the total area was calculated to measure the changes of the building. The result showed that there was a total of 61,9337 m^2 of the building in 2014, and there are 69,9517.7 m^2 of the building present. There is a total of 8,0180.71 m^2 of the increase in total area, which is 12.95% total increase. Figure 10 shows the result of analysis.

Figure 10

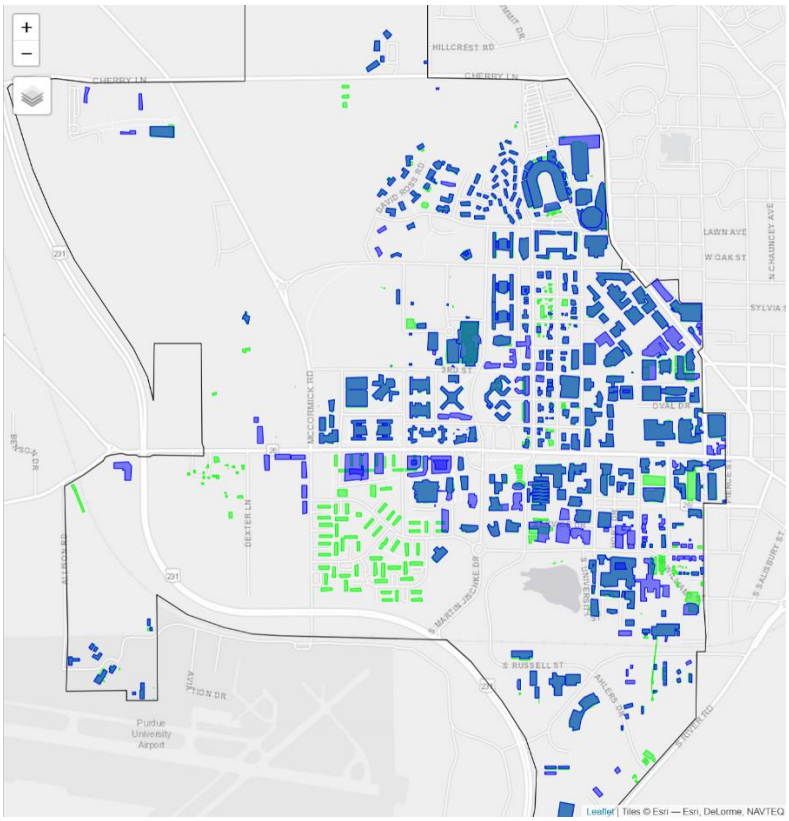
Changes of Building Areas

```
> # Report the results
> cat("Total area of Purdue buildings in 2014 (sq meters):", total_area_2014, "\n")
Total area of Purdue buildings in 2014 (sq meters): 619337
> cat("Total area of current OSM Purdue buildings (sq meters):", total_area_current, "\n")
Total area of current OSM Purdue buildings (sq meters): 699517.7
> cat("Area change (sq meters):", area_change, "\n")
Area change (sq meters): 80180.71
> cat("Percentage change in total area:", area_change_percentage, "%\n")
Percentage change in total area: 12.94622 %
> cat("Number of buildings removed since 2014:", number_removed_buildings, "\n")
```

To have a better visualization, both buildings from 2014 and OSM has been plotted into the same map (Figure 11, where green is 2014, and blue is current)

Figure 11

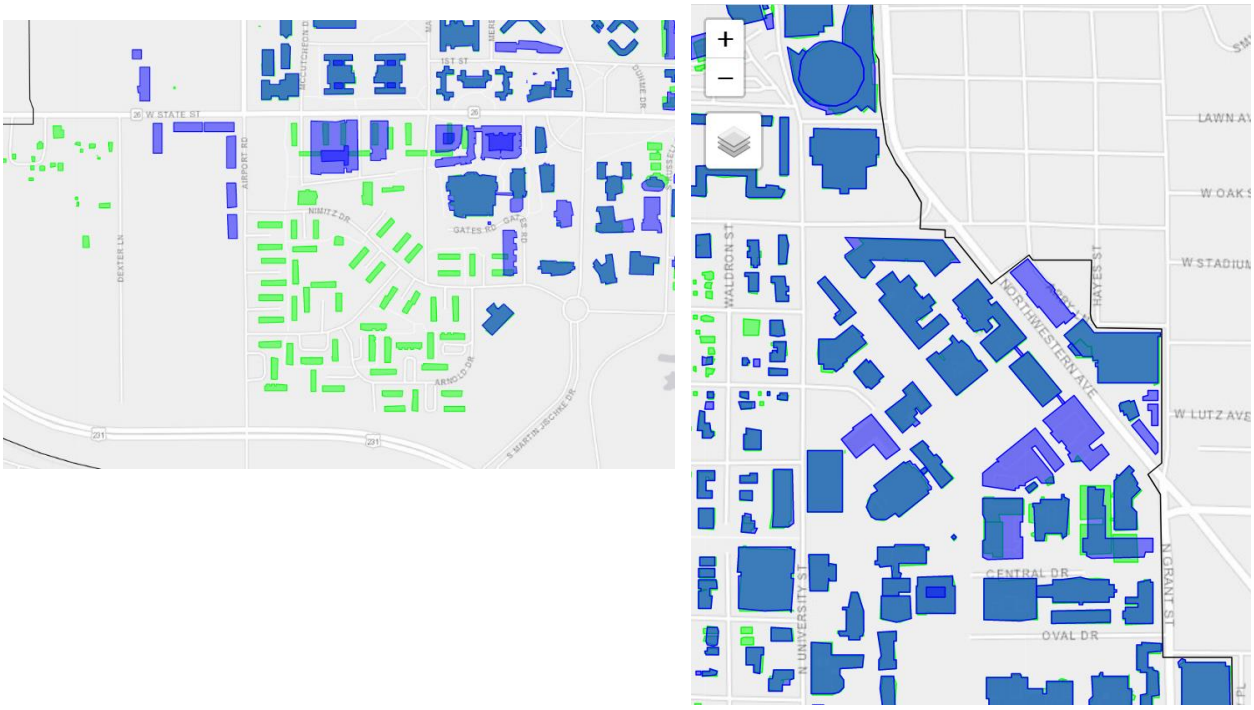
Purdue Campus 2014 vs. OSM (Current)



It is clear to see that there have been many changes on the Purdue campus. One notable change is that Purdue Village no longer exists (Figure 12, left). Additionally, there are several new buildings on the Purdue campus. For instance, Wang Hall, which began construction in 2014, stands out as one of the iconic new buildings compared to the older campus (Figure 12, right).

Figure 12

Purdue Village (old) and Wang Hall (new)



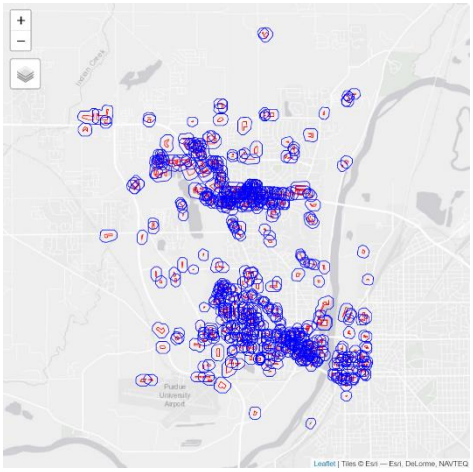
4. Extra Layer and Spatial Analysis

For my personal point of interest, I chose parking features and conducted a buffer analysis around parking lots to assess their accessibility in relation to other urban elements. I established a buffer distance of 100 meters and utilized the st_buffer function to create buffer zones around each parking lot. This process generated new geometries that delineate areas extending 100 meters from the edge of each parking lot. Assuming that 100 meters is a reasonable walking distance, Figure 12 illustrates the buffered zones of parking lots. The figure clearly demonstrates that the Purdue campus has numerous parking spaces distributed throughout, mostly within a walkable distance. However, what the analysis does not reveal is the number of available spots in each parking lot, as they can vary significantly. This information gap does not aid students in locating available parking near the

campus. Additionally, it is noteworthy that there are many parking lots along Sagamore Parkway, likely due to the abundance of restaurants and grocery stores in that area.

Figure 12

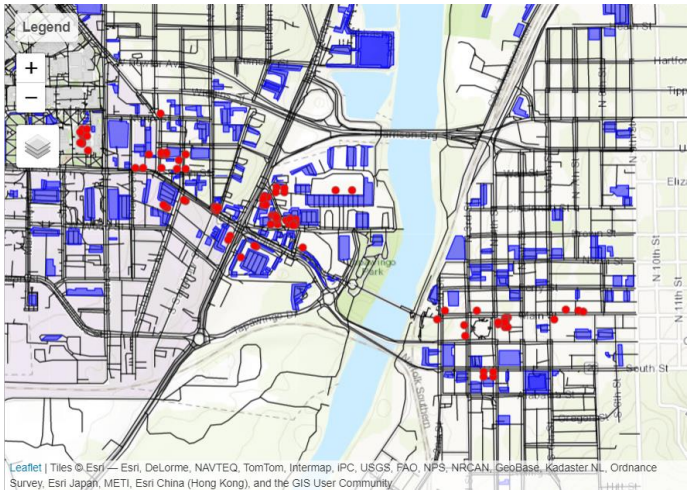
Parking Lots Buffered 100m



In addition, Figure 13 shows the map of the restaurant, parking lots, and the road of West Lafayette area. Where the red points indicate the restaurant, the blue polygons indicate the parking lots and the black lines indicate the roads. It is clear to see that the restaurant at West Lafayette will have some parking lots near them. In contrast, for the restaurant at Lafayette downtown area, most of the parking lots located further.

Figure 13

Restaurant, Parking, and Road of West Lafayette



5. Summary/Conclusion/Concluding Remarks

This project was conducted as a comprehensive review that integrated key aspects from previous projects, such as data acquisition from OpenStreetMap (OSM), and the calculation of length, area, and centroids. A significant challenge in this project was data acquisition and cleaning. Since the dataset did not have a predetermined boundary, it was essential first to delineate the bounds of the Purdue campus before proceeding with the analysis. Moreover, the building data contained numerous duplicates and invalid entries, necessitating thorough cleansing to ensure the accuracy of the analysis. However, certain buildings presented challenges that could not be resolved, such as the inability to dissolve overlapping polygons, which could impact the precision of the spatial analysis.

Acknowledgement

References

OSM Data Codings

```
available_features()

#    Task 01 read data for West Lafayette
bbox <- getbb('west lafayette, usa')
typeof(bbox)
class(bbox)
bbox

# Add building features
wl_blds <- opq(bbox) # Open a query based on the bbox
wl_blds <- add_osm_feature(wl_blds, key = 'building')
wl_blds_sf<- osmdata_sf(wl_blds) # Convert osm to sf

ttm()
qtm(wl_blds_sf$osm_polygons) # Plot the polygons

# Add roads features
wl_rds <- opq(bbox)
wl_rds <- add_osm_feature(wl_rds, key = 'highway')
wl_rds_sf<- osmdata_sf(wl_rds)

qtm(wl_rds_sf$osm_lines) # Plot the roads
```

```

# Add parking lot features
wl_parking <- opq(bbox)
wl_parking <- add_osm_feature(wl_parking, key = 'amenity', value = 'parking')
wl_parking_sf <- osmdata_sf(wl_parking)

# Plot the parking lots
qtm(wl_parking_sf$osm_polygons) # or osm_points, depending on the data

# Get bounding box for buildings
bbox_buildings <- st_bbox(wl_blds_sf$osm_polygons)
print(bbox_buildings)

# Get bounding box for roads
bbox_roads <- st_bbox(wl_rds_sf$osm_lines)
print(bbox_roads)

# Get bounding box for parking lots
bbox_parking <- st_bbox(wl_parking_sf$osm_polygons) # or osm_points, depending on th
print(bbox_parking)
```

Shapefile saving to local.

```
#  Save shapefile
# Define file paths for the shapefiles
path_to_buildings_shp <- "./buildings.shp"
path_to_roads_shp <- "./roads.shp"
path_to_parking_shp <- "./parking.shp"

# Write the datasets to shapefiles with overwrite capability
st_write(wl_blds_sf$osm_polygons, path_to_buildings_shp, delete_layer = TRUE)
st_write(wl_rds_sf$osm_lines, path_to_roads_shp, delete_layer = TRUE)
st_write(wl_parking_sf$osm_polygons, path_to_parking_shp, delete_layer = TRUE)
```