Final Project Report

City of Haverhill QAlert System Analysis

Team Members: Xiaochen Xue, Jing Yu, Zhitong Su, Kaijia You

Abstract

The city of Haverhill (the client) incorporates a 3-1-1 QAlert system, in which the residents report non-emergency issues through this hotline and the city government responds accordingly by departments. To further enhance the efficiency and to lower cost, the government of Haverhill would like to visualize the data in an interactive map, where the requests are categorized in layers to help decision makers isolate issues. Specifically, the client is interested in requests related to trash disposal and could possibly affect the city's trash collection routes.

Our team approach the project by first breaking down the objective into sub-tasks. We first performed preliminary analysis to understand each piece of data given in a static context. Then we determined which attributes of the dataset is necessary and cleaned the dataset accordingly. With a concise modification of the data, we then experimented and visualized three interactive maps: a heatmap of number of requests against each precinct, a map that display all requests sorted by department, and a map that display clusters of requests related to trash.

Objective

The main objective of the project is to answer "How is the QAlert calls distributed along the city of Haverhill, specifically the request types relating to trash collection?" To achieve a satisfactory answer, the team communicated with the client and the project coordinators, and learned the specific needs of the clients and summarized them as follows:

- Clean unnecessary data entries from the "Haverhill_Requests" csv file.
- Categorize and sort the entries in term of request type; the client is most interested in requests that affect the city's trash collection route.
- Merge such requests with the provided GeoJSON data and visualize its distribution on an interactive mapping context.

Data Preparation and cleaning

Our team started the data cleaning process by examining the GeoJSON JavaScript. We were surprised to see the coordinates were out of scale; they were unreasonably large, such as (7966550, -4238769). After online research, we determined that these coordinates in the JSON file were projected using the original shapefiles, using WGS84/Pseudo-Mercator projections (EPSG:3857). This means the coordinates provided by clients were essentially no use to the progress of the project. To generate a new JavaScript with GPS coordinates, we decided to work from the shapefiles provided in the packages, and re-project them using WGS84/World (EPSG:4326), which now the geographical data is ready to be merged with requests.

The CSV data containing the requests were then cleaned. First, the unnecessary entries with "informational" request types were deleted, as well as the ones lacking coordinates. This reduces the data size from ~80,000 to ~17,900. Second, the dimensions of the data were reduced. There are 36 attributes for each data point, and we only extracted the ones that can potentially affect the visualization, such as latitude, longitude, request type, department etc. Then, the dataset was converted to a data frame on python, sorted by departments that handled 311 QAlert calls, and exported as "department.csv" attached to this deliverable. From "department.csv", data with request types relating to trash, specifically "Illegal Dumping", "Barrel Overflow", "Enforcement", "Bag Request" and "Trash Cart Management" are taken to form a new dataset called "trash.csv".

Data Merging

To answer the project question, we decided to visualize in two maps: A map containing requests categorized by department, and a map containing requests related to planning of trash collecting route only. To merge data for the first map, all data points are used. We first convert both "department.csv" and JSON files into data frames, group by occurrence of requests by precinct, and merge with the geometry (the polygon from coordinates) attribute to get both the number of requests in each precinct and its geometry on GPS map. The same merging technique is used for the trash route map with "trash.csv".

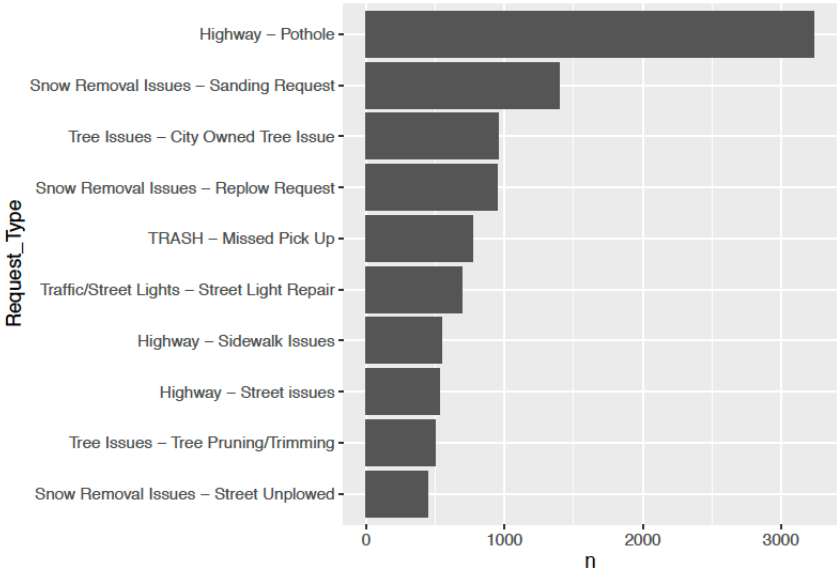
Besides, we merged the haverhill-request.csv dataset with the geographical information provided in the shp files such as Hav_Precincts_Wards_WGS84.shp. As mentioned above, the CSV dataset contains the information of each request being generated, including request type, latitude, and longitude. However, it is not sufficient for our project, since we are interested in the question such as "What kinds of services are being requested per ward/precinct along with their frequency?" We need one more feature or column that indicates which ward/precinct each request belongs to. Along with the CSV dataset, we are given several packages concerning the geographical information. One of them tells the geographical division of ward/precinct in Haverhill, which is stored at the form of multi polygon. Now that we have the latitude and longitude of each request and polygon of each ward/precinct, we should figure out a way to combine them and output a column of ward/precinct for each request. The technology we employ here is the sf package in R, which is designed for spatial analysis. In particular, we use the st_intersect() function, which inputs two geometries and output whether they intersect with each other. For our problem, the two inputs are the latitude and longitude of a certain request and the polygon of a certain ward/precinct. In Haverhill, the ward/precincts are labeled in a way such as "1-1", "3-4", "5-2', which are not informative. To make it more informative, we add one more column called "Polling_Station_Name", which is the address of the polling station of each ward/precinct. In addition, we perform the same workflow for the Hav_CDBG_Area_WGS84.shp file. CDBG stands for Community Development Block Grant Area, which is a central are of Haverhill. We add one more feature or column that indicates whether each request belongs to this central area. "1" stands for yes, while "0" stands for no.

Exploratory Data Analysis

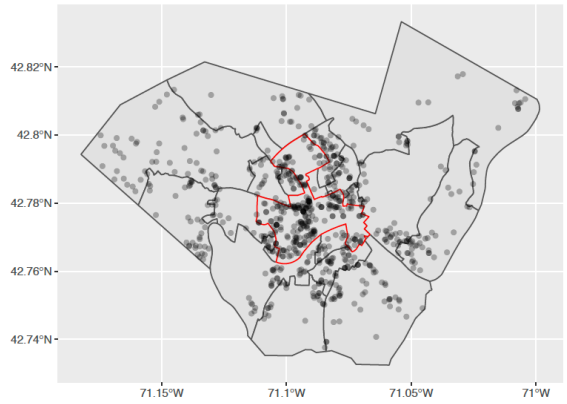
Based on the new CSV data set we constructed, we are able to answer some strategic questions, such as "what is the top 10 locations requesting services?", "What kinds of services are being requested per ward/precinct along with their frequency?", "What kinds of services are being requested by the Community Development Block Grant (CDBG) Area along with their frequency?", and etc. To answer these questions, we generate substantial amount of tables and graphs to better visualize the insights we acquire from the dataset. Example is shown below.

Haverhill Request Type Frequency

```
## # A tibble: 162 x 2
##    Request_Type                                  n
##    <chr>                                     <int>
##  1 Highway - Pothole                          3241
##  2 Snow Removal Issues - Sanding Request      1393
##  3 Tree Issues - City Owned Tree Issue         957
##  4 Snow Removal Issues - Replow Request        947
##  5 TRASH - Missed Pick Up                      772
##  6 Traffic/Street Lights - Street Light Repair 695
##  7 Highway - Sidewalk Issues                   542
##  8 Highway - Street issues                     529
##  9 Tree Issues - Tree Pruning/Trimming         497
## 10 Snow Removal Issues - Street Unplowed       448
## # ... with 152 more rows
```



For all precinct/wards, we create their corresponding tables and graphs that reflects the frequency of the request types, just like the example shown above. We achieve all these by simply using the filter and groupby function in R. The work that summarizing the request types within each ward/precinct is meaningful because it can help the city better predict and prepare for requests. The frequency of request types could be a good reference to allocate the labor resources. Apart from the analysis of request type frequency of each ward/precinct, we also categorize illegal dumping sites, trash missed pickups and recycling misses pickups, which are stored at three separate CSV files. By categorizing these locations, the City can revise their trash collecting route accordingly to avoid these requests.

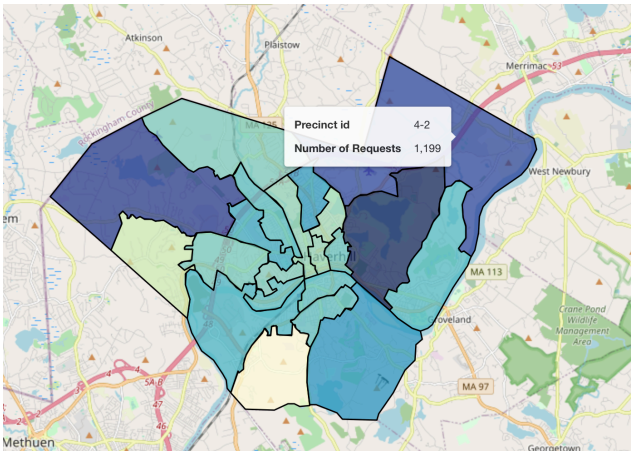


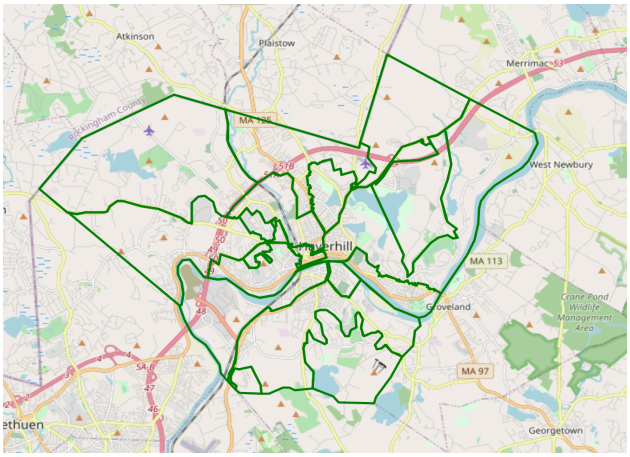
Trash Missed Pickups

```
## # A tibble: 16 x 2
##    Address                n
##    <chr>              <int>
##  1 101 BLAISDELL ST       5
##  2 134 WARRENTON RD       5
##  3 28 BROADWAY            4
##  4 37 WELLINGTON AVE      4
##  5 643 WASHINGTON ST      4
##  6 12 VARNUM ST           3
##  7 123 COGSWELL ST        3
##  8 257 LOWELL AVE         3
```

Visualize Interactive Map

The team has discussed on the platform to perform data visualization, as none of the teammates have a PC nor virtual environment, and it was inconvenient to use Boston University's machines due to remote learning, we have decided to create high quality interactive maps using Python's





Folium library, and export the final products in html format so it would be easy to use on the Client's end.

Figure 1: Heatmap of all requests with labels Figure 2: Haverhill Trash collecting Route

Figure 3: Wards/Precincts border with CDBG Figure 4: Wards/Precincts border

The first plot made according to client's need is a visualization of all requests. We used the merged data above and applied a base layer of heatmap with respect to wards/precincts, so the client would

get an intuitive sense of which area/precinct the 3-1-1 calls are more abundant. Then, the data points are grouped by their respective departments that answered the request, totaling 15 additional layers. The visualization technique used in this map is to plot every request as a size 1 point, and the color of the point vary with respect to the department. In this map, the highway department answered the most calls, and the plot is most dense around the CDBG area.

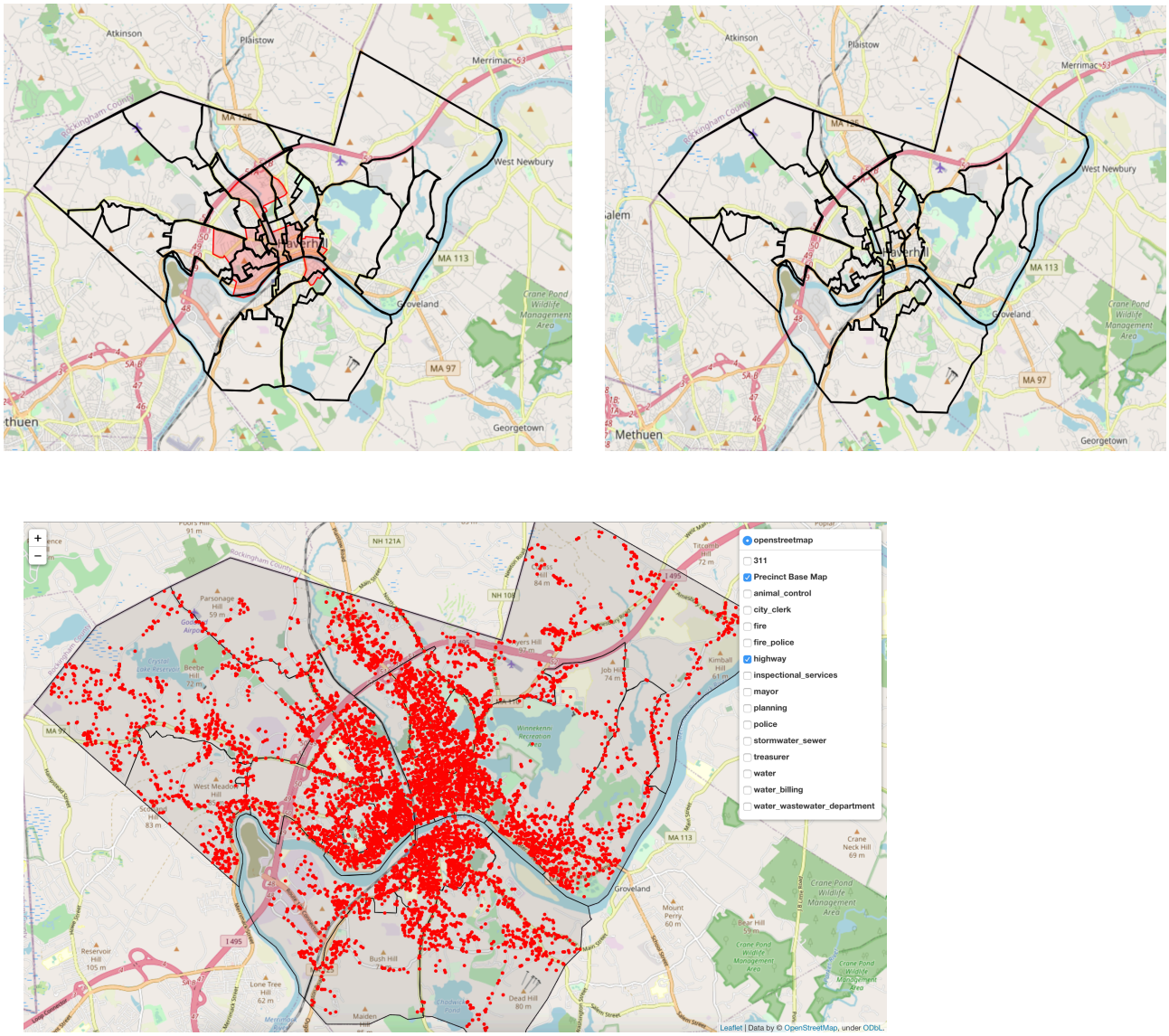


Figure 5: Sample Visualization of all requests answered by highway department

The second plot made is a visualization of requests relating to trash collection only. As with the first plot, a heatmap was plotted first as a base layer to visualize the density of trash-related requests

in each ward/precinct. Five additional layers were then plotted to visualize each of the five types trash-relating requests and their geographical location. We tried a new form of visualization here, because plotting all points for a type of request may have a disorganized first impression, so we produced a cluster technique, where the locations are grouped by clusters, and each cluster is labeled with its respective number of members; if the client wishes to see the exact locations, they can simply click on the cluster, or zoom in on the map, to examine sub-clusters, or actual data points. This way, the map is organized, and the visualization becomes more intuitive. When shown to client, it was evident that the client was satisfied with this six-layered map and it was clear that our team has achieved its goals.

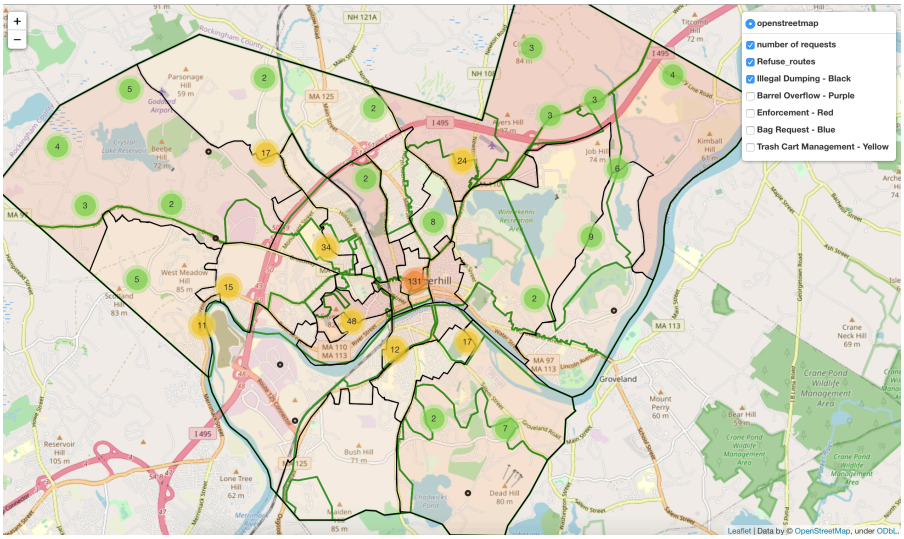
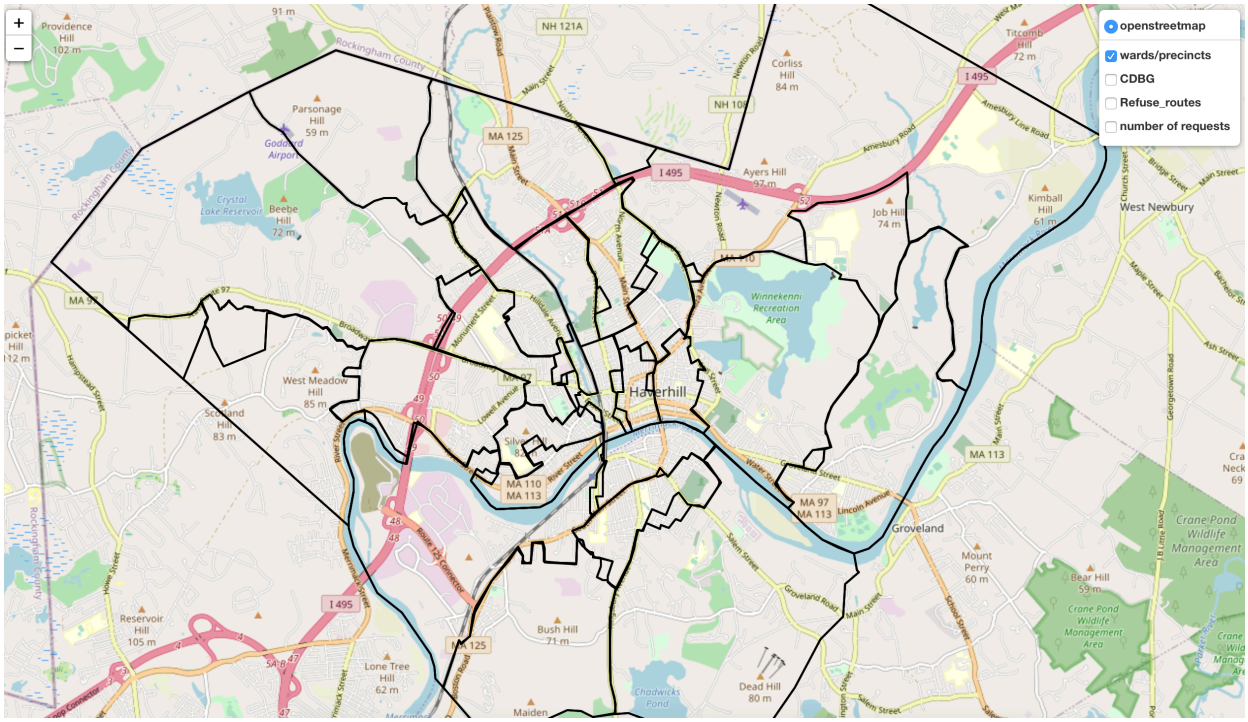


Figure 6: Sample Visualization of requests related to Trash Illegal Dumping

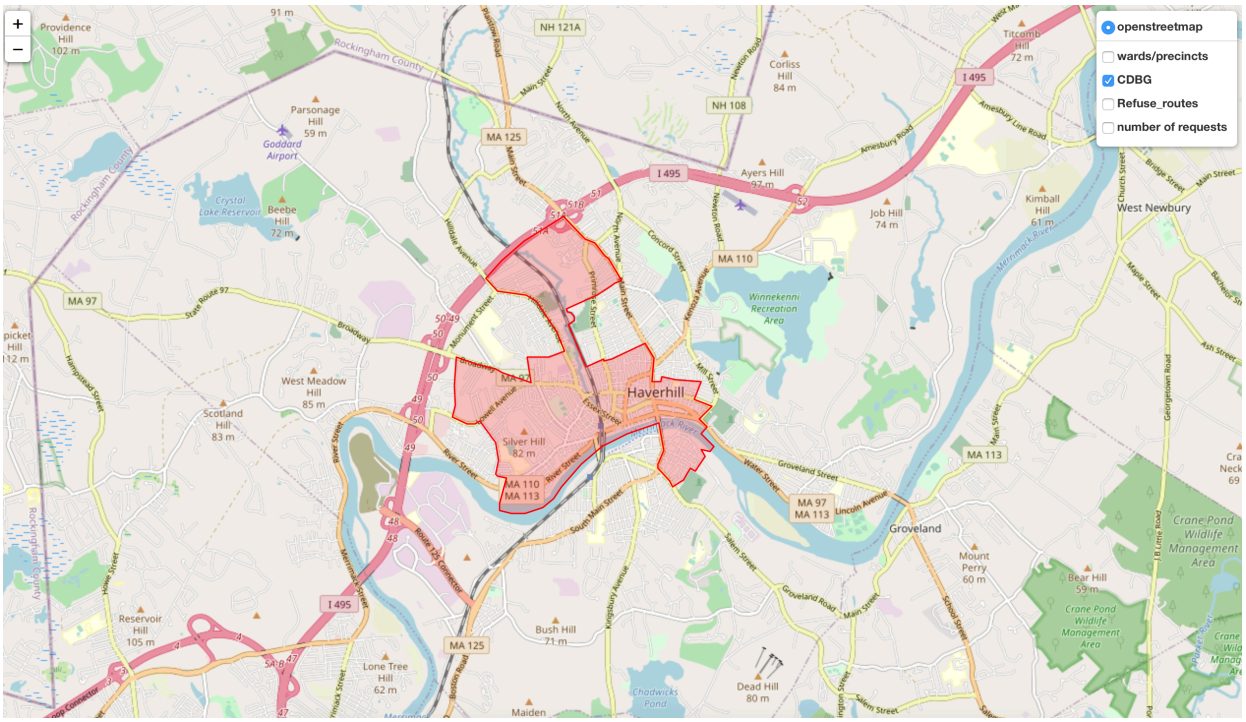
Limitations and Conclusion

The maps are rigid in a sense that we cannot add more layers to existing html maps. And the coding portion is not friendly to users with little python experience. But we have provided specific and explicit instructions to run these codes in instruction file and show the visualization of data and result of two. ipynb files later. I believe our group has completed the visualization needs of the client, and will take this further to enhance the hand-off experience by modifying the coding portion. We are more than happy to produce visualizations for the client over the summer to help them make decisions.

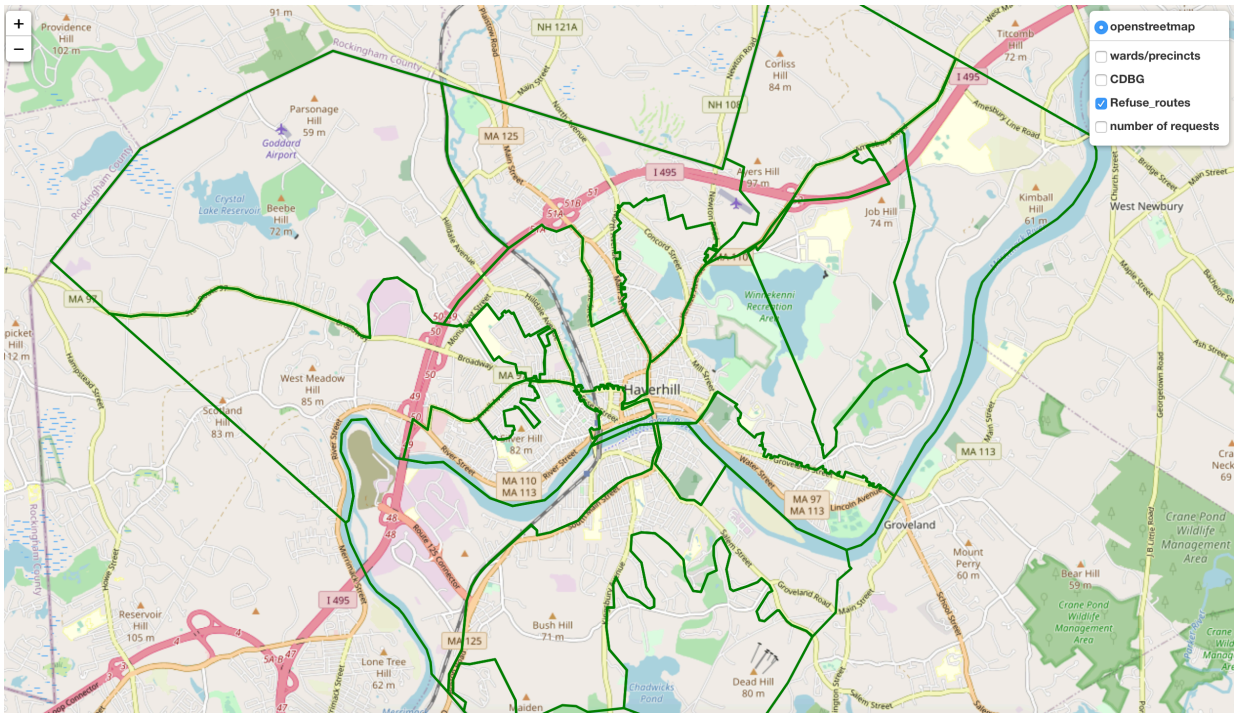
“final_project.ipynb” : This piece of code contains two components: creating the overall visualizations of the client’s data, and the visualization of sorted requests. In the first portion of the code, we first initialize a map of Haverhill by hardcoding the longitude and latitude of the city [42.7762, -71.0773] into a folium map. Then we created four additional layers based on this map. The first three layers are visualizations of the GeoJSON files. We first read the shapefile, re-project them into the GPS coordinates using EPSG code of 4326, then create a style function most appropriate for each of the three layers shown below (Wards/Precincts, Border of CDBG and Trash collecting Route).



Wards/Precincts



CDBG Area

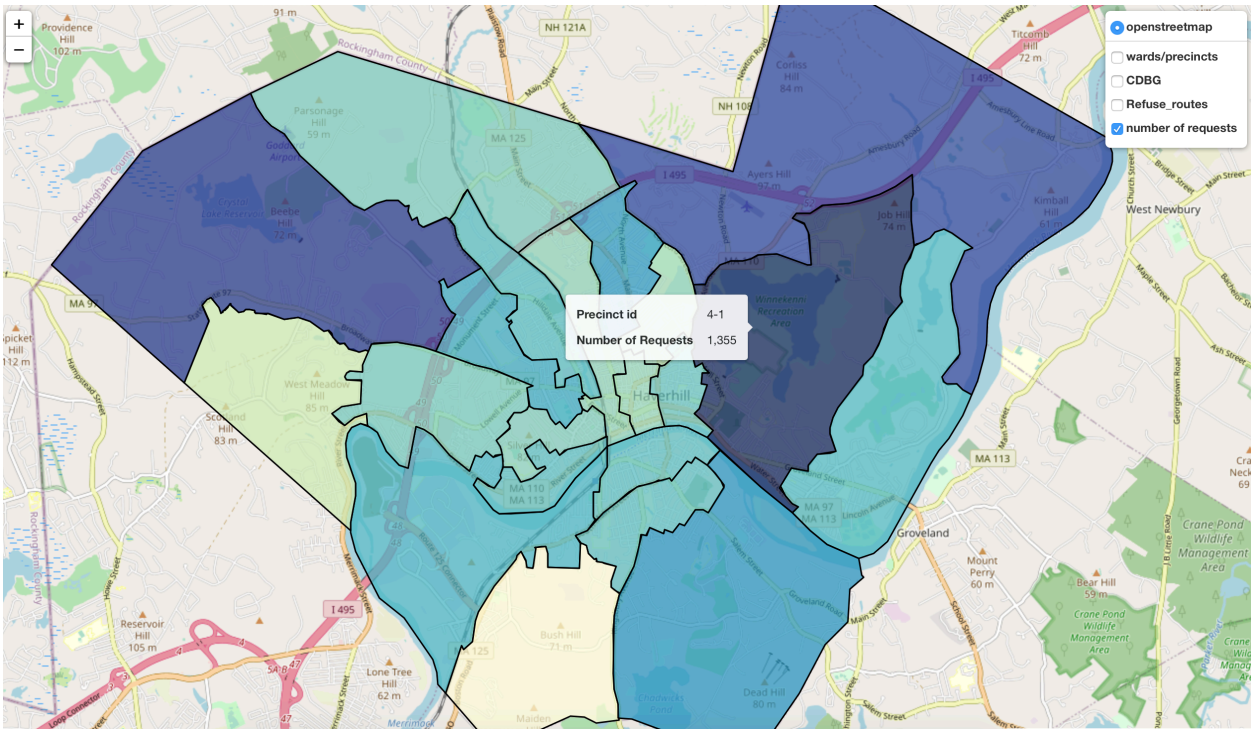


Refuse Collecting Routes

Now, a heatmap is created to showcase the density of requests handled by each Precinct/Ward. The JSON data was first converted to GPS coordinates. The cleaned 'department.csv' file was taken and the number of requests per precinct was collected using the df.groupby function. Then, the JSON data and the analyzed csv file were merged, so each row contains precinct ID, its polygon geometry, and the number of requests in it.

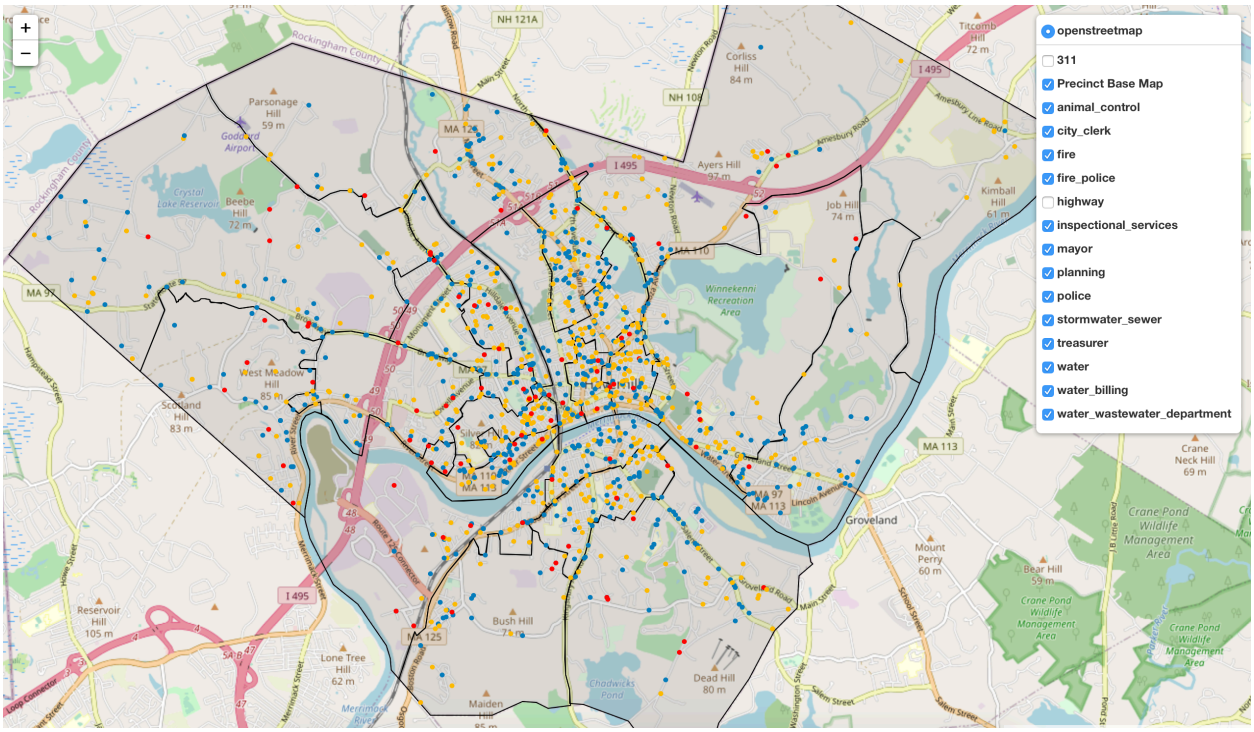
	Precinct	geometry	nb
0	1-1	POLYGON ((-71.09160 42.77516, -71.09232 42.774...	734
1	1-2	POLYGON ((-71.08679 42.78449, -71.08710 42.785...	706
2	1-3	POLYGON ((-71.10464 42.79081, -71.10478 42.790...	904
3	2-1	POLYGON ((-71.06884 42.77243, -71.06980 42.773...	815
4	2-2	POLYGON ((-71.12075 42.76855, -71.12138 42.768...	860

With this information, the heatmap is created as a layer and when the mouse is hovered on, the map will show exact numbers and precinct name.



Interactive Heatmap of Number of Requests per Precinct

Now, our team uses the heatmap as a base layer, unifying colors to grey while keeping the interactive feature, we plotted sorted requests as points on this map. From the "department.csv" dataset, we created 15 layers for each department in the city; in each layer, every request is plotted as a point with a radius of 1 on the map. This way, the client can see the exact distribution of every request and plan their response locations in accordance with the density of the points.



Example Requests per department visualization

"final_project_2.ipynb" : read the "trash.csv" file, and load all 480 rows of requests relating to Refuse Routes on to Jupyter Notebook as a data frame called "trash".

	Request_ID	Create_Date	Request_Type_ID	Request_Type	Department	Street_ID	Longitude	Latitude	Ward_Precinct
0	59225	2/13/19 16:37	341	Trash - Bag Request	Highway	647	-71.084017	42.751749	7_3
1	52598	11/20/18 17:53	341	Trash - Bag Request	Highway	305	-71.086570	42.767473	2_1
2	86005	1/21/20 15:10	341	Trash - Bag Request	Highway	630	-71.055604	42.748844	7_3
3	25958	1/19/18 16:42	341	Trash - Bag Request	Highway	559	-71.077883	42.789527	3_3
4	61378	3/7/19 16:35	342	Trash - Barrel Overflow	Highway	1061	-71.081813	42.777646	3_1

Then, same data merging technique is used to create the base map from the Wards/Precincts shapefile with the requests in the "trash.csv" file as shown below, note that the request numbers are significantly smaller than the previous.

	Precinct	geometry	nb
0	1-1	POLYGON ((-71.09160 42.77516, -71.09232 42.774...	45
1	1-2	POLYGON ((-71.08679 42.78449, -71.08710 42.785...	46
2	1-3	POLYGON ((-71.10464 42.79081, -71.10478 42.790...	24
3	2-1	POLYGON ((-71.06884 42.77243, -71.06980 42.773...	27
4	2-2	POLYGON ((-71.12075 42.76855, -71.12138 42.768...	10

This time, our team used a more convoluted approach to make the map more succinct and organized yet still expressing the same information as the last map. We implemented the MarkCluster function to organize data points into zoomable clusters; we incorporated this feature for every single layer, in this case a total of five as there are five specific issues related to trash collection. A sample layer with only Illegal dumping related clusters is shown below.