Final Project Deliverable 3- Project Report Draft

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.

CSV Data

Haverhill.csv: This is the reduced version of the original csv. We simply deleted all rows with "informational" data types, and the rows with no geographical coordinates. Besides, it contains the additional columns including "Ward", "Ward_Precinct", "Polling_Station_Name", "CDBG".

Department.csv: This is a modified version of the "Haverhill.csv" file, where only 9 attributes are kept for visualization purposes, and the data are sorted by "Department" attribute.

Trash.csv: This is the file containing only trash-related requests modified from "department.csv". The 480 rows of data are sorted by "Request_Type" and are all important to trash collecting routes.

Trash: This is a document containing three csv files: "illegal_dumping_sites.csv", "recycling_missed_pickups.csv", "trash_missed_pickups.csv", which are the categorization of illegal dumping sites, recycling _missed_pickups, and trash_missed_pickups separately. All three

csv files contain two columns, where the first column is address and the second column is frequency.

JSON Data

The GeoJSON data provided by the client was projected through the Pseudo-Mercator Projection type (EPSG: 3857), which is not suitable for the World GPS coordinates. We re-projected suitable JSON files to use, and the process is explained in the coding section.

Codes

The codes of data cleaning, feature creation, and exploratory data analysis are written in R with tidyverse and sf packages installed.

The codes of interactive maps are written in Jupyter Notebook, under Python 3 environment, with folium and geopandas libraries installed.

"Haverhill_Data_Cleaning.Rmd" : This is a R markdown document that contains text, codes, and results. This piece of codes create a new csv file of requests that filters out all "informational" request types, as well as the ones lacking coordinates. This document of codes can be reused to clean any future request data stored in the haverhill-request.csv file. The procedures of reuse would be: Download R and R Studio → Type install_packages("tidyverse") in the console → Store "Haverhill_Data_Cleaning.Rmd" and "haverhill-request.csv" in the same directory" → Open the directory through "Open Project" → Run all chunks of codes → A new csv file with all irrelevant rows removed is created in the same directory.

"Haverhill_Ward&Precinct.Rmd" : This is a R markdown document that contains text, codes, and results. This piece of codes contains two components: mapping all requests onto the map of Haverhill with the borders of every ward/precinct, creating three new columns "Ward", "Ward_Precinct", "Polling_Station_Name" in the original "haverhill-request.csv" file. This document of codes can be reused to process any future request data stored in the haverhill-request.csv file. The procedures of reuse would be: Download R and R Studio → Type install_packages("tidyverse") and install_packages("sf") in the console → Store "Haverhill_Ward&Precinct.Rmd" and "haverhill-request.csv" in the same directory" → Open the

directory through “Open Project” → Run all chunks of codes → A new csv file with additional columns is created in the same directory. Although the codes are reusable with proper operation, we recommend the client to record the ward/precinct information in the process of data collection.

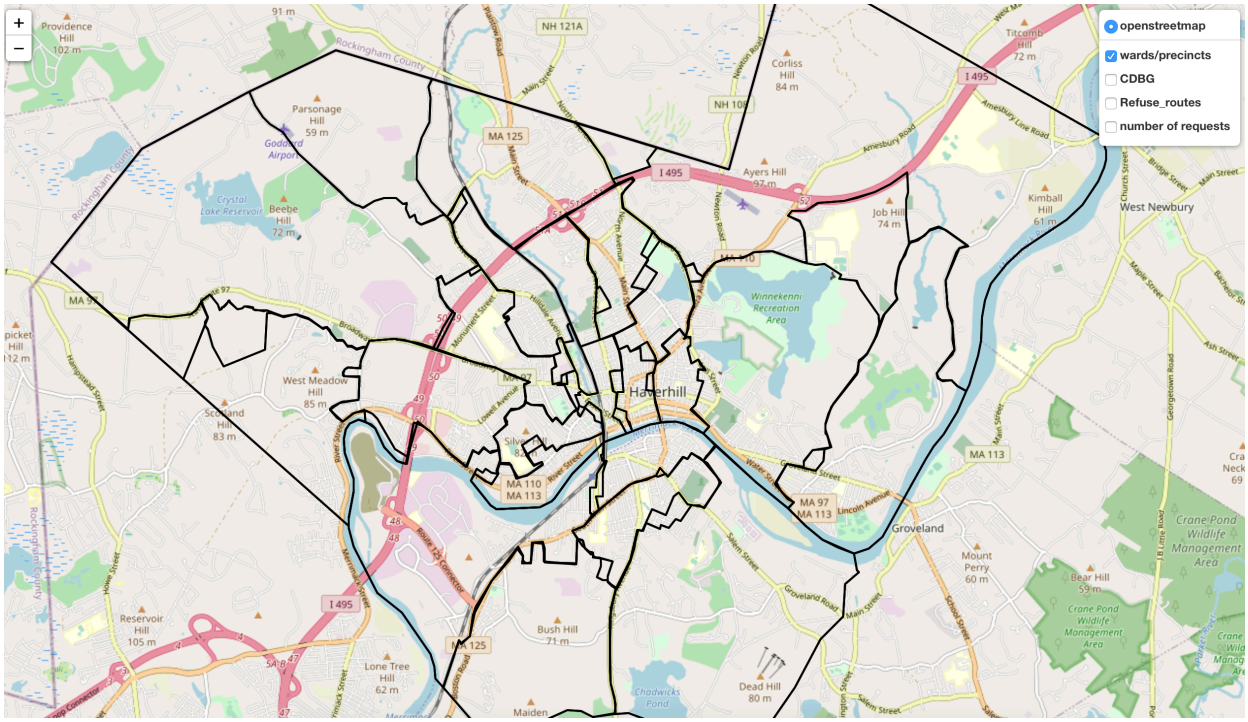
“Haverhill_CDBG.Rmd” : This is a R markdown document that contains text, codes, and results. This piece of codes contains two components: mapping all requests onto the map of Haverhill with the borders of CDBG, creating a new columns “CDBG” in the original “haverhill-request.csv” file. This document of codes can be reused to process any future request data stored in the haverhill-request.csv file. The procedures of reuse would be: Download R and R Studio → Type install_packages(“tidyverse”) and install_packages(“sf”) in the console → Store “Haverhill_CDBG.Rmd” and “haverhill-request.csv” in the same directory” → Open the directory through “Open Project” → Run all chunks of codes → A new csv file with an additional column is created in the same directory. Although the codes are reusable with proper operation, we recommend the client to record the CDBG information in the process of data collection.

“Haverhill_EDA.Rmd” : This is a R markdown document that contains text, codes, and results. This piece of codes shows all of the exploratory analysis and static visualizations. By clicking the button “Knit” on the top of R studio, it will automatically generate a pdf version with only text and graphics in the same directory. To make it work, a concrete procedures would be: Download R and R Studio → Type install_packages(“tidyverse”) and install_packages(“sf”) in the console → Store “Haverhill_EDA” and “haverhill-request.csv” in the same directory” → Open the directory through “Open Project” → Click the button “Knit” on the top of R Studio → A new pdf file with only text and graphics is created in the same directory.

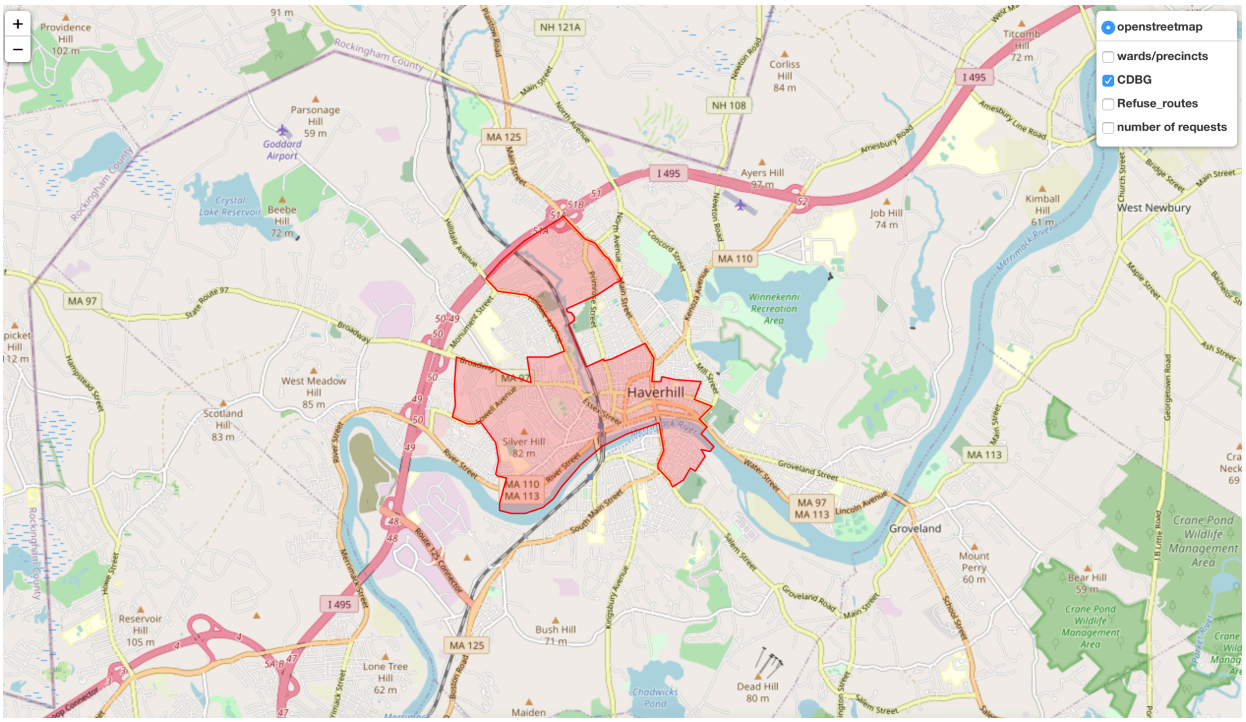
Notes:

1. Storing all R codes (Rmd files) and the CSV file (haverhill-request.csv) in a same folder is recommended.
2. The Procedures “Download R and R Studio” and “Type install_packages(“tidyverse”) and install_packages(“sf”) in the console” only need to be done once, especially when it is the first time using R.
3. To make sure the read_csv() work, please manually rename all variable names in a way such that connect all words by “_” in Excel.

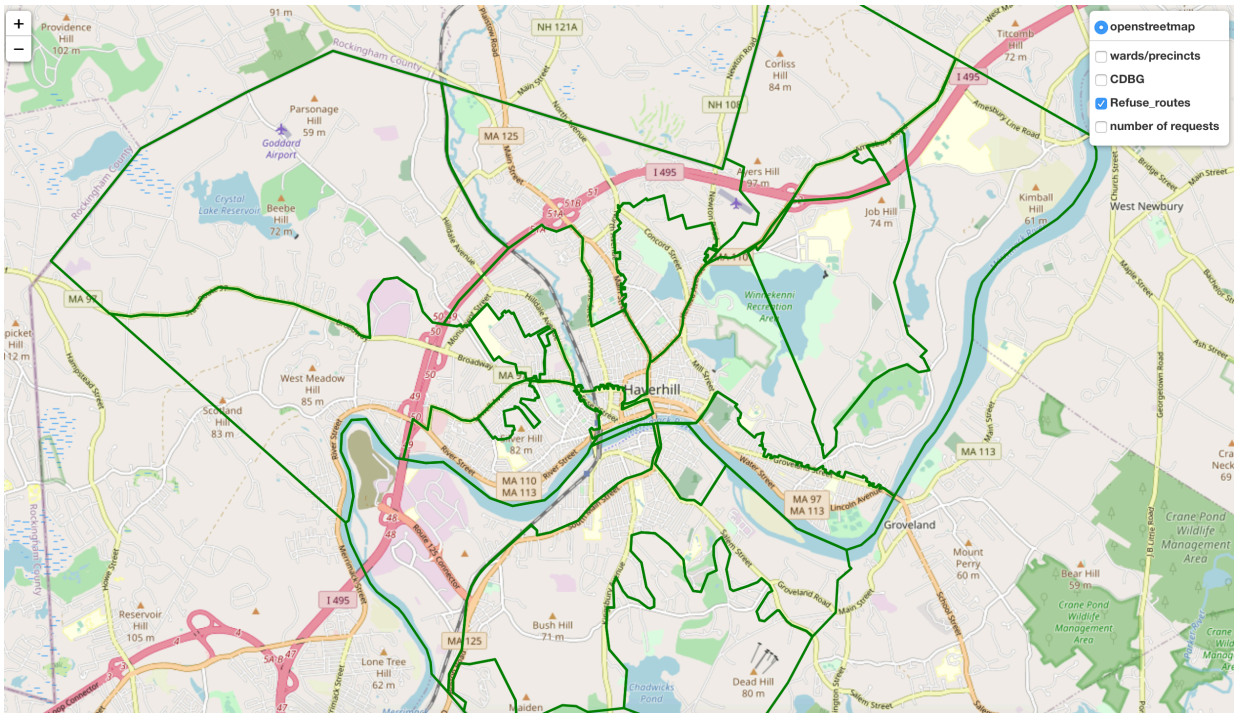
“final_project.ipynb” : This piece of code contains two components: creating the overall visualizations of the client’s data, and the visualization of sorted requests. In the first portion of the code, we first initialize a map of Haverhill by hardcoding the longitude and latitude of the city [42.7762, -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

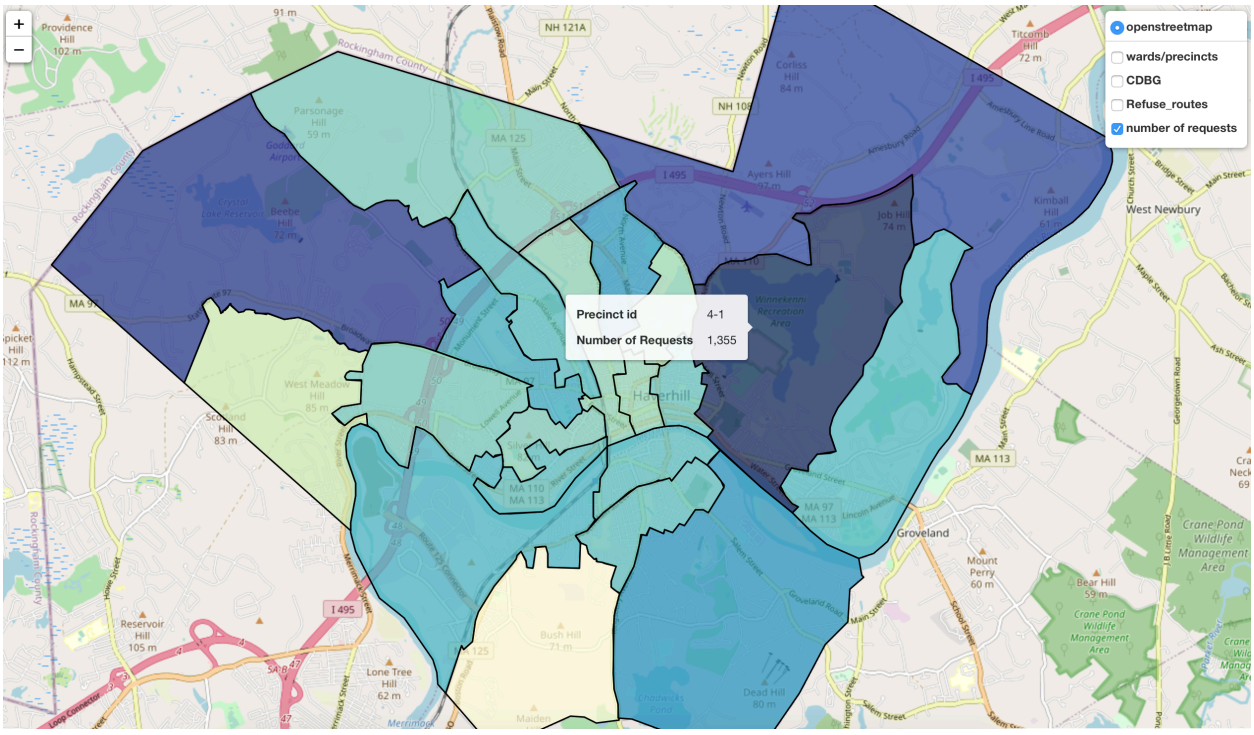


Refuse Collecting Routes

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.

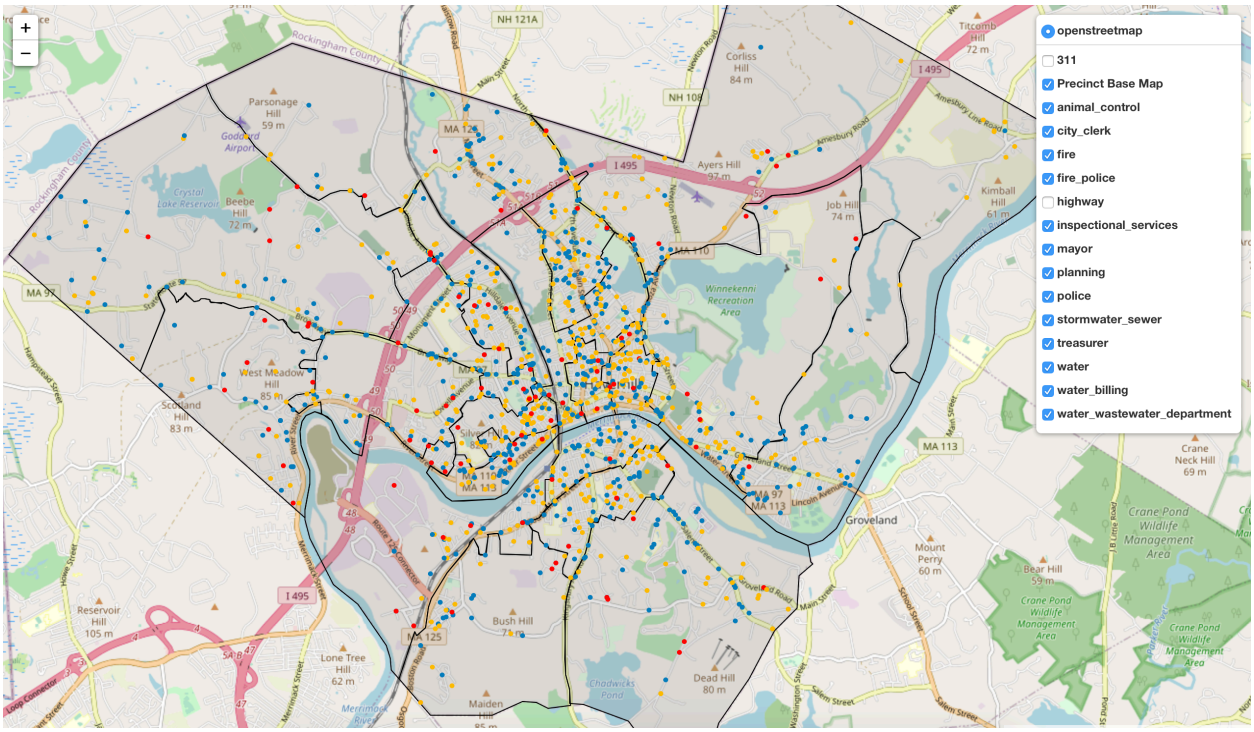
	Precinct	geometry	nb
0	1-1	POLYGON ((-71.09160 42.77516, -71.09232 42.774...	734
1	1-2	POLYGON ((-71.08679 42.78449, -71.08710 42.785...	706
2	1-3	POLYGON ((-71.10464 42.79081, -71.10478 42.790...	904
3	2-1	POLYGON ((-71.06884 42.77243, -71.06980 42.773...	815
4	2-2	POLYGON ((-71.12075 42.76855, -71.12138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