Meng Shen
Tsz Lung Chan

MCIT 594 Project Report

Additional Feature

The additional feature calculates the correlation between total fines per capita and total property value per capita. This analysis aims at verifying whether citizens in neighborhoods in higher property value tend to pay more parking violation fines.

This additional feature utilizes the data sets by:

i) Calculating the average fine per capita in each zip code by using the parking fine data and population data; and

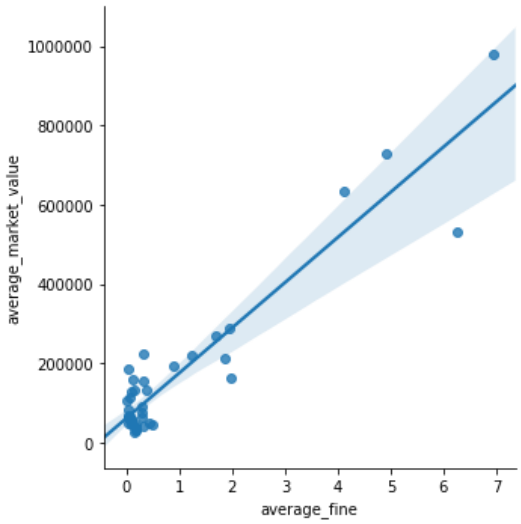
ii) Calculating the average property value per capita in each zip code by using the property data and population data

The correlation coefficient that indicates the strength of the relationship between two variables can be found using the following formula:

$$r_{xy} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{[n\sum x_i{}^2 - (\sum x_i)^2][n\sum y_i{}^2 - (\sum y_i)^2]}}$$

Correctness of the Additional Feature

We did a statistical calculation of the correlation between the two variables using Python Jupyter Notebook. The correlation we analyzed was 0.9392, which is consistent with our program output. The coefficient of 0.9392 indicates that total fines per capita and total property value per capita have a high positive correlation. This means that people in neighborhoods in higher property value tend to pay more parking violation fines.



	average_fine	average_market_value
average_fine	1.000000	0.939237
average_market_value	0.939237	1.000000

Meng Shen
Tsz Lung Chan

Use of Data Structures

Array List

We used an Array List to store the parking violation and the property value data parsed. See JSONParkingReader class, CSVParkingReader class and PropertyReader class.

The reasons for choosing Array List were:

- The primary goal was to store the data entries;
- We did not need to worry about if a particular element is contained at this stage;
- We did not know the exact number of data entries while reading inputs
- There was no need for sorting

Because we did not know the exact number of data entries while reading inputs, we decided to use dynamic Array List instead of standard array. Another data structure we considered was Linked List. However, Array List was more efficient with the get method, which runs in O(1) time.

Tree Map

We used Tree Map to store the population data. See PopulationReader class. We also used Tree Map to store processed results such as calculateFinePerCapita method, calcuateAverage method, and calculateValuePerCapita method.

The reason for choosing Tree Map were:

- Each zip code has a unique value and can be used as a key-value pair;
- We would like to sort the zip code keys by natural ordering to facilitate output

The population.txt data was arranged in ascending zip codes. This could potentially be arranged in an array of population if we could find a way to correspond each element at a particular array index to a particular zip code. However, this is never as efficient and semantically meaningful to use key-value pairs where the relationship can be easily understood, and the data is also more efficiently retrieved. In addition, we could not maintain any order by the zip code keys if Hash Map was used. In this case, the ordering was useful for sanity check and output display.

Tree Set

We used Tree Set as an interim step to find the common Zip codes when we would like to process two Maps jointly. See the commonZip variable in ParkingProcessor class, PropertyProcessor class, and UserInterface class.

The reason for choosing Tree Set were:

- We would like to find the intersection of two sets which contained zip codes present in different Maps;
- We would like to sort the zip code keys by natural ordering to facilitate transversal

We considered to use a Hash Set because it was more efficient and adding and retrieving data were done in O(1) time. However, it did not represent the data in any meaningful order. We preferred to ensure the zip codes were arranged in order for further handling in our codes.