CIT 594 Group Project Report

Yuyuan Lin(linyuy@seas.upenn.edu), Zhong Liu(zhongliu@seas.upenn.edu)

Additional Feature

The topic of the additional feature in our final project is the relationship between the Total Fines Per Capita and the Total Residential Market Value Per Capita. In statistics, we can use the cross-correlation coefficient to quantify the relationship. There are several types of correlation coefficient formulas. One of the most used is the Pearson's correlation coefficient formula, which is shown below.

$$CC = \frac{\sum_i (I_1(i) - \overline{I_1}) \cdot (I_2(i) - \overline{I_2})}{\sqrt{\sum_i (I_1(i) - \overline{I_1})^2 \sum_i (I_2(i) - \overline{I_2})^2}}$$

In the equation, let us use $I_1(i)$ as the Total Fines Per Capita in each zip code, and $I_2(i)$ as the Total Residential Market Value Per Capita. $\overline{I_1}$ is the average Total Fines Per Capita in all zip codes of interest. And $\overline{I_2}$ is the average Total Residential Market Value Per Capita in all the zip codes of interest. Therefore, the entries of I_1 have used the datasets of population and parking and those of I_2 have used the datasets of population and property.

In the implantation, for simplicity, we have used a derived form of the Pearson's correlation coefficient formula, which is shown below:

$$CC = \frac{cov(I_1, I_2)}{\sigma_{I_1} \sigma_{I_2}}$$

, where $cov(I_1, I_2)$ is the covariance, σ_{I_1} and σ_{I_2} are the standard deviations. Essentially the calculation of covariance, standard deviation, variance is similar. Hence using this equation saves some time.

Now let us explain the physical meaning of the cross-correlation coefficient. The range of the coefficient is [-1, 1]. If the coefficient is close to 1, then the two features are positively correlated, i.e. a larger value of feature 1 results in a larger value of feature 2. If the coefficient is close to -1, then the two features are negatively correlated, i.e. a smaller value of feature 1 results in a larger value of feature 2. If the value of the coefficient is close to 0, then the two features are almost independent.

We verified the results of our implementation with a few online Pearson's correlation coefficient calculators with some data sets. The results are the same. For example, using the artificial data below:

Zip Code	Total Fines Per Capita	Total Residential Market Value Per Capita
19103	0.9	62.5
19104	1.69	40
19106	0.72	20
19107	3.6	600
19123	1.8	250

Intuitively we can see that the larger value of total residential market value per capita results in a larger total fine per capita, there must be a positive correlation. Using the on-line correlation coefficient calculator[1], the coefficient is 0.944. In our implementation, we have 0.944.

Use of Data Structures

Array

We have used the array data structure for a temporary storage of the data line entries for csv type files in the reader classes. We have used the array data structure for the following reasons:

(1) We only need a simple temporary storage and access data structure. There is no need of other operations such as insertion, search, and deletion afterwards once the data of interest of retrieved.
(2) Array is associated with the string split() method in java API. So it is straightforward and convenient.
(3) Our input file has a relatively fixed format, the meaning of each column in the csv files are fixed. So we can quickly access to the data information of interest with the O(1) time complexity. The data of interest is then saved in the data object.

HashMap

We have used HashMap to store and update the values associated with certain keys. For examples, in the property reader class, we have used the zip codes as the keys and the total market values as the values. The total market values are updated for each new entry. The memorization or cache of answers to several of the fix problems also uses HashMap where we cache calculation result by each zip code as key in the HashMap, which provides quick access to result later on. See **UserInterface** class for details.

We have used the HashMap data structure for the following reasons:

(1) We need a data structure that can store and update key-value pairs. And the order of the key-value pairs is not of interest.
(2) We need to do access and insertion for many times. The time complexity is extremely important. The HashMap has complexity of O(1) for insertion and lookup, which is perfect for our purposes.
(3) We can use LinkedHashMap for the similar time complexity in terms of the insertion and lookup. The LinkedHashMap will keep the order in which key-value pairs are inserted. But this is not necessary.

TreeMap

We have used tree map for question 2-- Total Fines Per Capita in the ParkingViolationProcessor class. The information must be displayed on screen in the ascending order. And the information displayed is a key-value pair.

We have used the data structure for the following reasons:

(1) We need a data structure that can store and update key-value pairs. And the order of the key-value pairs is of interest.

(2) We need to do access and insertion for many times. The time complexity is extremely important. TreeMap has complexity of O(logN) for insertion and lookup. So it is not very efficient if we have a dataset with a size of half a million. However, the HashMap has complexity of O(1) for insertion and lookup. Therefore, we have decided to use the HashMap to create the complete set of key-value sets with quick accessibility first. The HashMap only have a size about 50. And then we convert the HashMap to TreeMap by looping over the entire data structure. The TreeMap is used for display in the end.

Reference:

[1] https://www.socscistatistics.com/tests/pearson/