Xtern Work Sample Assessment

Role: Data Science

By: Michael Napoli

Submitted on: October 13, 2019

Assumptions used for Data Analysis:

1. A given unit on the xy-plane is equal to 1 miles geographically.
2. There is only one mega charge bus in operation for the time being.
3. The bus can house and charge 100 scooters at a given time.
4. The bus will travel between clusters at an average close to its maximum speed (50 mph).
5. Travel time between scooters within a given cluster is negligible.
6. The time required to charge the scooter population is equal to:

$$charge\ time = \frac{(5 - battery\ value\ of\ scooter) * (number\ of\ scooters)}{(charge\ capabilities)}$$

7. A cluster is defined as a group of scooters within 0.4 radial units of the central scooter location.

Introduction:

Note: All programs and data files mentioned are located in the data_manip_files folder within this github repository.

Scooters are a fast and energy efficient way of moving around a busy city landscape. They provide the consumers affordable and easily available transportation without adding to traffic congestion. The only problem with these scooters is the fact that they run on a finite energy supply stored within their batteries and must be charged in order to stay operational. In order to do so, the Xtern Xpress ridesharing service needs an efficient plan to charge scooters after consumers are done using them.

The scooter location and battery power data, given by the TechPointX team, was utilized in calculating the number of scooters within a given cluster and the subsequent battery power average of said clusters. The data of distance from each scooter to the mega charging bus was also calculated but later realized to be irrelevant to the findings of the evaluation. Therefore, the route by the mega bus and the overall strategy of the charging system was found to be connected to specific scooter distance from the bus parking location.

Programming Used for Data Manipulation (C++):

The data calculations and final results found throughout this analysis were calculated through programs written in C++ code for the evaluation of large data samples. These files include distance_eq.cpp, cluster_mapping.cpp and time_charge.cpp.

In the distanc_eq.cpp file, the class for point values was created as seen below. It was made up of two variables of type double and consisted of the proper get and set member functions as seen in Figure 1 on the next page. The point class was essential to evaluating the point coordinates of the data set given using a single vector for all scooter point values. The ID of each scooter was then understood to be the vector location of the scooter.

```
17  // POINT CLASS TYPE
18  // class type made for the calculation of distance values and coordinate manipulation
19  class Point
20  {
21     private:
22     double x;  // x-coordinate
23     double y;  // y-coordinate
24
25     public:
26
27     // set class functions (sets the x and y coordinate respectively)
28     void setX(const double temp_x) { x = temp_x; }
29     void setY(const double temp_y) { y = temp_y; }
30     void setXY(const double temp_x, const double temp_y) {
31        x = temp_x;
32        y = temp_y;
33     }
34
35     // returns the appropriate value
36     double getX() { return x; }
37     double getY() { return y; }
38     void getCoordinates() { cout << "(" << x << "," << y << ")"; }
39  };
```

Figure 1: Point Class

To read in the scooter data values, a function was written that used the simplified data table (put into .txt file format) to add elements into proper vectors. This function disregarded the scooter ID because it was understood that the vector location given by the program would be equal to its ID number. This can be seen in Figure 2 below.

```
104  // FUNCTION: open_scooter_data()
105  // open the appropraite data file and read in x-y coordinates
106  bool open_scooter_data(const string &filename, vector<Point> &pt_vect, vector<int> &batt_vect) {
107     ifstream fin;  // stream variable for reading from data file
108     int scooter_ID(0), temp_batt(0);  // temporary values for battery level and scooter ID
109     double temp_x(0), temp_y(0);  // temporary variables for x-y cooridnates
110     Point temp_pt;  // temporary variabled for Point to be added to vector
111
112     fin.open(filename, ios::in);
113
114     // if file is not opened, exit function with error message
115     if (!fin.is_open()) {
116        cout << "ERROR: Data file, " << filename << ", not accessible." << endl;
117        return false;
118     }
119
120     while (!fin.eof()) {
121        fin >> scooter_ID;  // scooter ID number (irrelevant to data manipulation)
122        fin >> temp_x;  // scooter x-coordinate
123        fin >> temp_y;  // scooter y-coordinate
124        fin >> temp_batt;  // scooter battery level
125
126        temp_pt.setX(temp_x);  // apply changes to the temporaray Point variable
127        temp_pt.setY(temp_y);
128        pt_vect.push_back(temp_pt);  // add point to appropriate vecotor
129        batt_vect.push_back(temp_batt);  // add to battery level vector
130     }
131
132     fin.close();
133
134     return true;
135  }
```

Figure 2: open_scooter_data() Function Definition

Once the data from the files was read into the program and assigned to the appropriate vectors. The program calculated the distance of each scooter from the bus and wrote those values to a .csv file so that the values could be graphed. Unfortunately, after calculation and subsequent transfer of values to the main data set excel file, it was observed these distance values played no role in what the route of the bus and ultimate charging method for the scooters would be. That being said, the point class, distance equation, and the reading and writing functions would be utilized to some extent in the cluster_mapping.cpp file which yielded important results.

The file cluster_mapping.cpp was written to evaluate the number of scooters and average battery value of each of the 19 clusters. This was done by approximating the center point for all of the clusters seen in Figure X (figure shown in the Data Analysis and Results section) and calculating all points within these clusters using a cluster radii of 0.04 coordinate units and the distance function shown in Figure 3 below. This was done after the values of the scooters was read in using the open_scooter_data() function shown in Figure 2.

```
121  // FUNCTION: distance()
122  // calculate the distance between two point values (class type) and returns it
123  double distance(Point &pt1, Point &pt2) {
124     // use pythagorean's theorem to calculate distance
125     return (sqrt(pow((pt1.getX() - pt2.getX()), 2) + pow((pt1.getY() - pt2.getY()), 2)));
126  }
```

Figure 3: distance() Function Definition

The scooter locations were categorized into the appropriate their cluster by comparing their respective location to that of the central coordinate of all the clusters. If they were within 0.04 units of the central coordinate, they were understood to be located in that cluster. This process can be seen in the Figure 4 code below.

```
82   if (open_scooter_data("data_set_simplified_format.txt", scooter_location, battery_level)) {
83      // for all scooter locations
84      for (int i(0); i < scooter_location.size(); ++i) {
85         // check to see what cluster they are located in
86         for (int k(0); k < cluster.size(); ++k) {
87            // if they are in the given cluster
88            if (distance(cluster[k], scooter_location[i]) < CLUSTER_RADIUS) {
89               cluster_battery_total[k] += battery_level[i];  // add their battery value to the cluster total
90               ++cluster_scooter_count[k];  // iterate the cluster's population value
91               break;  // break loop and continue to next scooter
92            }
93         }
94      }
95   }
```

Figure 4: Cluster Mapping for Loops and if Statement

The last program written for data analysis was time_charge.cpp. This program used the cluster data found via the cluster mapping program to evaluate the time needed to charge all scooters individually in each cluster. These values were instrumental in calculating the Operating Time Cost of the scooter ridesharing system.

Within the program, the reading function previously used was scaled down in order to read different vector values from the cluster data as opposed to reading in the entire scooter data file. This decreased the computation of the program from around 26000 values to 19 and removed the need for the Point class type. After loading in the 19 cluster populations and battery averages, the battery average values were converted to time and multiplied by the total number of scooters located in each cluster to calculate the time required to charge all the scooters within the cluster individually. This can be seen in Figure 5 below.

```
29    // if data file is opened and read successfully
30    if (open_cluster_data("simplified_cluster_data.txt", cluster_scooters, battery_level)) {
31       // loop through all cluster elements
32       for (int i(0); i < battery_level.size(); ++i) {
33          // calculate total time to charge all scooters individually
34          temp_charge_time = (5 - battery_level[i]) * (double)cluster_scooters[i];
35          cluster_charge_time.push_back(temp_charge_time);  // add to cluster time vector
36       }
37    }
```

Figure 5: Calculating Scooter Charge Time for Clusters

Once the time vectors for individual scooter charging were put together, the values were outputted to a comma separated value file called time_per_cluster.csv for manipulation in an excel formatted file.

Data Analysis and Results:

It should be noted that the total scooter population calculated by finding all scooters within each cluster is roughly 99.96% the actual number of scooters owned by the Xtern Xpress ridesharing company. This is because of the error associated with the cluster radius size and its inability to account for all scooters within every cluster. That being said, the data and percentages calculated by this process should be considered close enough to the actual data of each cluster because of the extremely small error found in the exclusion of only 0.04% of scooters.

The scooter data given contained xy-coordinates for more than 25000 individual scooters along with their ID numbers and their respective battery charge. These scooters were found to be organized across a coordinate plane in 19 differing clusters, or regions of high scooter concentration. The Figure 6 plot displays this trend along with the cluster ID numbers used in the data manipulation and route planning.

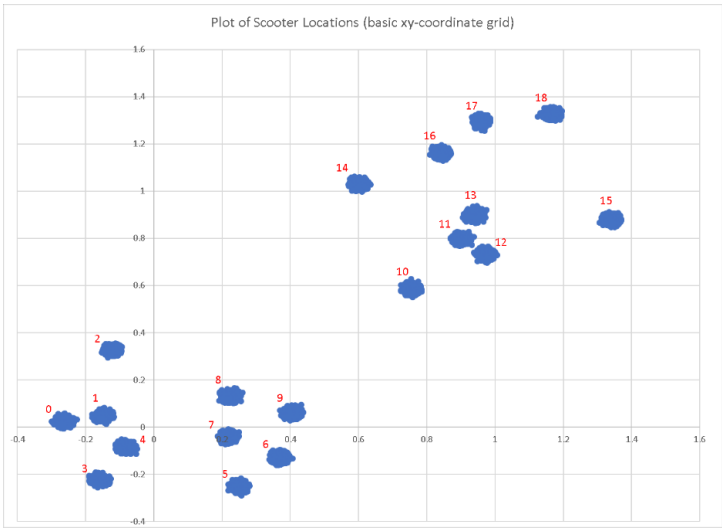

Figure 6: Scooter Locations and Cluster ID Numbers

The 19 clusters displayed in the plot have a relatively equal distribution of scooters apart from that of cluster 8, which contains 8.58% of the total scooter population, making it the most densely packed of the 19 clusters as seen in Figure 7 on the next page. All other clusters had a scooter population equal to roughly 5-6% with a few small deviations. This data can be found in Table 1, located on the next page.

Although cluster 8 has the highest specific cluster percentage of scooters, the clusters 11-13 are close enough to one another that travel between them would be relatively short and therefore negligible to the travel time of the mega charging bus. The combination of scooters within these three clusters is equal to 4070 scooters, or 15.86% the total population.

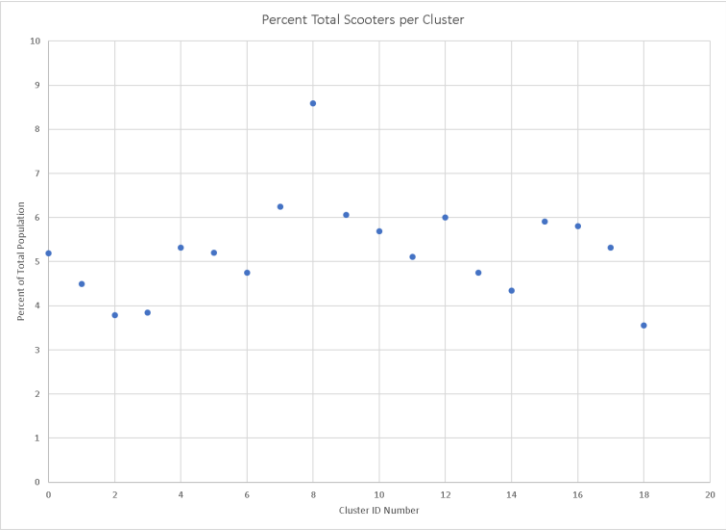

Figure 7: Scooter Percentage per Cluster

	cluster_ID	cluster_total	percent_scooters	battery_average	scooter_charge_time	bus_charge_time
	0	1333	5.19303	2.52138	3304	33.04
	1	1153	4.4918	2.48742	2897	28.97
	2	971	3.78277	2.43769	2488	24.88
	3	988	3.849	2.49393	2476	24.76
	4	1366	5.32159	2.41581	3530	35.3
	5	1334	5.19693	2.52999	3294.99	32.9499
	6	1219	4.74892	2.52994	3011	30.11
	7	1603	6.24489	2.48534	4031	40.31
	8	2204	8.58623	2.52768	5448.99	54.4899
	9	1557	6.06568	2.53308	3840.99	38.4099
	10	1459	5.6839	2.42358	3759	37.59
	11	1312	5.11122	2.45732	3336	33.36
	12	1540	5.99945	2.53182	3801	38.01
	13	1218	4.74502	2.51067	3032	30.32
	14	1115	4.34376	2.53453	2749	27.49
	15	1517	5.90985	2.50758	3781	37.81
	16	1491	5.80856	2.444	3811	38.11
	17	1366	5.32159	2.53075	3373	33.73
	18	913	3.55682	2.49179	2290	22.9
Averages		1350.47368	5.261105789	2.494436842	3381.787895	33.81787895

Table 1: Cluster Data

Within the cluster data table, the average battery power contained in each scooter was calculated in order to evaluate what clusters need the most initial attention by the Xtern Xpress charging team. With all clusters having a rough 2.5 charge rating, this data has no impact on the final findings of the charging system put in place by the company and the route taken by the bus. That being said, using this data, it is possible to calculate the average time needed to charge each bike which is proportional to (5 – average battery power). Therefore, the average bike needs approximately 2.5 hours of charge in order to reach full capacity.

Using the calculated cluster data, the time per cluster needed to charge all scooters individually could be found. The assumption of the bus being able to hold and charge up to 100 scooters at any given time was also utilized to understand the time needed to charge all scooters in a given cluster with the premise that the bus would be at full capacity the entire time the scooters were charging. This data can be seen in Table 1.

With this data the following conclusions were made. First, the mega charging bus would spend time in clusters with the largest proportion of scooters, these clusters include 8 and the collective clusters 11, 12 and 13 because of their close proximity to one another. For all other clusters, another form of charging would be necessary to meet the requirements of constantly having charged scooters available to consumers.

Conclusions and Charging Strategy:

Note: All data referred to here can be found in Table 2 on the next page.

With the limited availability of a single mega charge bus, the clusters it spends its time operating in must be prioritized based on the necessity for extra charging power and the proximity of the surrounding clusters so that it can move between clusters if necessary. Because cluster 8 contains the most scooters per single cluster, 8.58% of the total scooter population, it makes sense to have the bus operate within the limits of the cluster 8 scooters. In this cluster, with the bus operating at maximum capacity, it should take the bus 54.5 hours to completely charge all scooters. Because the bus cannot operate that long without running out of gas and/or power to give the scooters, there must be another charging method utilized alongside this.

Another grouping of clusters that make up a large percentage of the total scooter population is the clusters 11, 12 and 13. Together these clusters contain 15.86% of all scooters and are located near enough to one another for the bus to navigate easily between them. If the bus were operating within these limits, it would take approximately 101 hours for all the scooters to be fully charged. Therefore, another method is needed for these clusters to maintain operation.

Along with the aforementioned clusters, there are still another 15 that the bus will not be visiting and individually contain approximately 5% of the scooter population each. The bus cannot be expected to charge all the Xtern Xpress scooters as it would be unrealistic to expect a single bus to navigate throughout the given clusters without stopping for gas and/or refueling the power needed to supply the scooters.

To solve this issue, the charging port incentive program was created. This option details to the consumer the location of charging ports for scooters, most likely in areas of dense civilian traffic. If the consumer drops their scooter off at these ports before disembarking their ride, they may attain some form of gift. Whether that be a price decrease on that specific ride or some other form of incentive for the consumer, this option would definitely be beneficial to the program as well as give the consumer a goal during rides. The scooters would also all be located in one area so that they are more easily found by consumers and the company has the option to go to the location and check if any need repairs etc. Ideally, these ports would be able to charge 50 scooters at a given time and there would be about 4 placed throughout every cluster.

Another option are pick-up drivers. These employees of the ride sharing program operate at night and drive through the city using their scooter locaters to pick up scooters to move to some charging location. These locations can also be nondisclosed so that repairs can be done on broken and malfunctioning scooters.

Through the combination of port service and mega charge bus utilization the time needed to charge scooters in cluster 8 decreases to 18.16 hours and in the cluster 11, 12 and 13 grouping the time decreases to 33.90 hours. Although there would undoubtedly be problems with pedestrians attempting to use scooters that are out of charge, these efforts should decrease the problem significantly.

cluster_ID	percent_scooters	charge_time (hrs)	bus_charge_time (hrs)	port_charge_time (hrs)	combined_port_bus (hrs)
0	5.19	3304	33.04	16.52	11.01
1	4.49	2897	28.97	14.49	9.66
2	3.78	2488	24.88	12.44	8.29
3	3.85	2476	24.76	12.38	8.25
4	5.32	3530	35.30	17.65	11.77
5	5.20	3295	32.95	16.47	10.98
6	4.75	3011	30.11	15.06	10.04
7	6.24	4031	40.31	20.16	13.44
8	8.59	5449	54.49	27.24	18.16
9	6.07	3841	38.41	19.20	12.80
10	5.68	3759	37.59	18.80	12.53
11	5.11	3336	33.36	16.68	11.12
12	6.00	3801	38.01	19.01	12.67
13	4.75	3032	30.32	15.16	10.11
14	4.34	2749	27.49	13.75	9.16
15	5.91	3781	37.81	18.91	12.60
16	5.81	3811	38.11	19.06	12.70
17	5.32	3373	33.73	16.87	11.24
18	3.56	2290	22.90	11.45	7.63
Averages	5.26	3381.79	33.82	16.91	11.27

Table 2: Charge Times Among Clusters Data