Project Report
for
Taxi Trajectory Visual Analytics System

Davneet Kaur, *dkaur1@kent.edu*

1. Introduction

The Taxi Trajectory Visual Analytics System is based on the taxi trajectory data from the city of Porto. This dataset consists of real-time data associated with vehicles and road networks. Analyzing and visualizing this data can help to obtain some useful insights from data.

As part of this Visual Analytics system, I have implemented an interactive dashboard using D3.js, Leaflet.js and Keen.io. D3.js is a JavaScript library for visualizations using HTML, SVG and CSS. Leaflet.js is a JavaScript library provides interactive maps. Keen.io provides responsive templates for analytics dashboards.
Figure 1 shows the initial loading page for the dashboard.

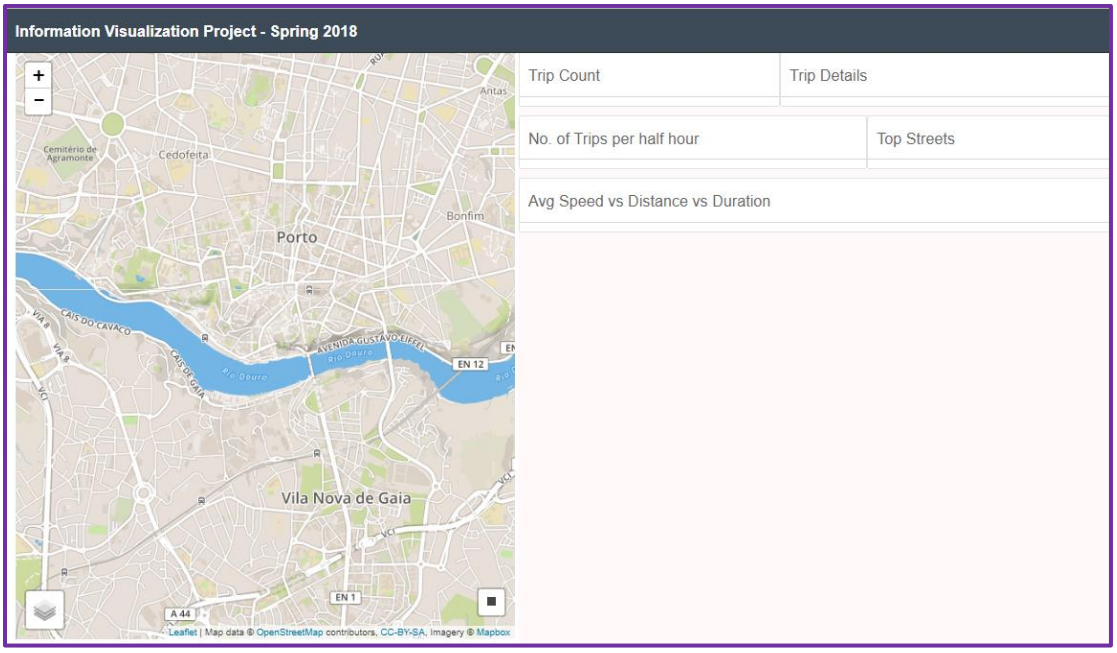

Figure 1: Initial loading page of dashboard

I have used some examples from the D3.js website as references to create different types of visuals in the dashboard. I have also done some formatting in the data that I got as a result of the initial bounding box query. This includes formatting the date and time attributes as well as adding a new attribute called “timeslot” that assigns a timeslot of half hour to each trip depending upon the trip start time. Also, the distance is changed from meters to miles and duration from seconds to minutes. All the decimal values are rounded to two decimal precisions. Figure 2 shows the code for format data function.

```
//*******************************************************************************************************************************
// FormatData Function:
//*******************************************************************************************************************************
function FormatData(trips) {
    var hours, minutes;
    trips.forEach(function (d) {
        d["avspeed"] = +d3.format(".2f")(d["avspeed"]);
        d["distance"] = +d3.format(".2f")(d["distance"] * 0.00062137);
        d["duration"] = +d3.format(".2f")(d["duration"] / 60);
        d["endtime"] = dateFormat.parse(d["endtime"]);
        d["starttime"] = dateFormat.parse(d["starttime"]);
        d["starttime"].setSeconds(0);
        if (d.starttime.getMinutes() <= 30) {
            d["timeslot"] = (("0" + d.starttime.getHours()).slice(-2) + ":" + "00" + "-" + d.starttime.getHours() + ":" + "30");
        }
        else {
            d["timeslot"] = (("0" + d.starttime.getHours()).slice(-2) + ":" + "30" + "-" + (d.starttime.getHours() + 1) + ":" + "00");
        }
    });
}
```

Figure 2: Format Data function

Figure 3 and 4 show the dashboard view upon running a query. The dashboard includes heatmap, trip count, list view, bar chart, donut chart and scatterplot matrix. Each of these visualizations are explained in section 2. It also includes a detail about what all interactions have been implemented in the dashboard.

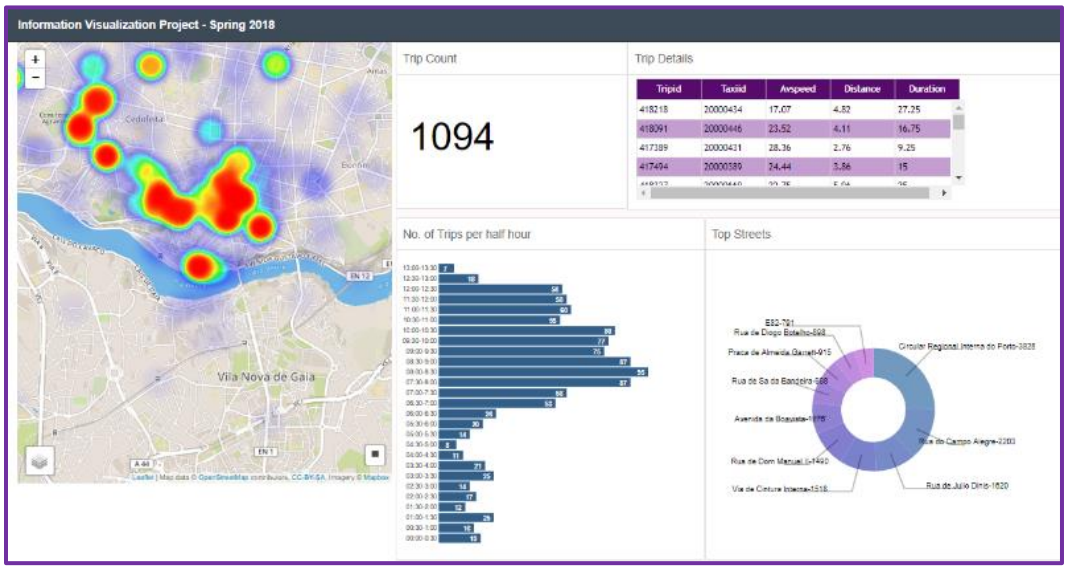

Figure 3: Dashboard view with queried data

2. Project implementation details

Below are the various visualizations from the dashboard:

- **Heat Map**

A heat map showing the hot pickup locations is shown. The selected trips from the bounding box query are passed to a DrawHeatMap() function. In this function, the pickup latitude and longitude of all selected trips are filtered from the TArr.js file and used to create the heatmap.

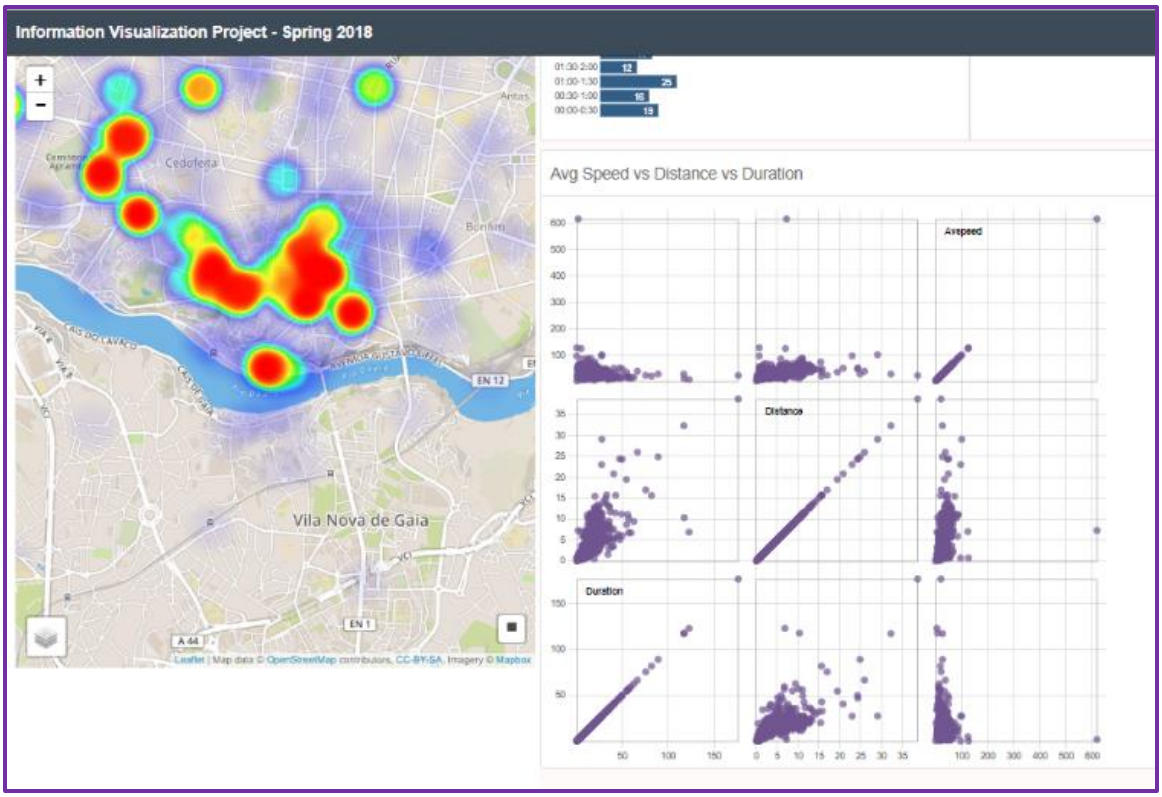

Figure 4: Dashboard view with queried data

- **Trip Count**

A trip count of all the trips in the selected bounding box is displayed. The selected trips from the bounding box query are passed to DrawTextbox() function that counts and displays the total number of trips. This count changes according to the number of trips when clicking on any bar in the bar chart or any arc in the donut chart. These interactions are explained in detail in the next section.

- **List View**

A list view showing the trip details of the selected bounding box visualized. The selected trips from the bounding box query are passed to DrawList() function. List view shows the Tripid, Taxiid, Average speed, Distance and Duration of the trips. Average speed is in km/hr, Distance is in miles and Duration in mins. These are rounded off to 2 decimal points for better visualization and clarity.

- **Bar chart**

A bar chart showing the number of trips per half hour is visualized. The selected trips from the bounding box query are passed to DrawBarChart() function. The new attribute “timeslot” is used in this visualization. The time slots range from 00:00 - 00:30 to 13:00-13:30 as there are no trips after 13:30 till midnight. The time slots in which there is at least one trip are only visualized. If there is no trip in certain time slot, it is not visualized in the bar chart.

- **Donut chart**

A Donut chart shows the top streets and trips associated with that street in the selected bounding region with respect to street names and their occurrences. The selected trips from the bounding box query are passed to DrawDonutChart() function. To get a better visualization nest() function is used to groupby street names along associated trips and later top 10 streets are picked if the number of streets is more than 10 or less otherwise.

- **Scatter Matrix**

A scatter plot matrix showing Avg speed, Distance and Duration with respect to trips selected in that bounding box region is visualized. The selected trips from the bounding box query are passed to DrawSactterMatrix() function. In matrix from top to bottom one can see the trips plotted based on Avg speed vs Distance vs Duration to obtain correlation between variables over a single view.

Interactions

There are mainly two types of filters in the dashboard. One is the bar chart and other is the donut chart. All the filtrations are associated with the mouse click events.

Bar chart interaction

Clicking on any bar in the bar chart updates the list view, donut chart, heat map and scatterplot. When a bar is clicked, all the trips within that time slot and from the initial bounding box are filtered and the other views are rendered accordingly.

For example, when clicking on bar with time slot - 9:30 - 10:00:

- The trip count column shows the number of trips that started between 9:30 - 10:00 time slot
- The list view is updated to show only those trips whose trip start time is within 9:30 - 10:00 time slot
- The Heat map is updated to show the pickups of only those trips whose trip start time is within 9:30 - 10:00 time slot
- The donut chart is updated to show the top 10 streets that are visited within 9:30 - 10:00 time slot

Donut chart interaction

Clicking on any arc on the donut chart updates the trip count, list view and the map view. When an arc is clicked, all the trips from the current view are filtered to get only those trips that have the same trip start street or end trip as the clicked arc. Then the trip count and list view are updated to show the filtered trips.

Also, the complete trajectories of the filtered trips are displayed on the map (figure 5). The start point of each drawn trip is highlighted using a green dot on the map.

For example, when clicking on an arc in donut chart with street name “Circular Regional Interna do Porto”:

- The Trip count column is updated to show the number of trips that have the starting or ending street as “Circular Regional Interna do Porto”.
- The list view is updated to show only those trips that have the starting or ending street as “Circular Regional Interna do Porto”.

- The map is updated to show the trajectories of trips starting or ending at “Circular Regional Interna do Porto”

The important thing to note here is that these interactions pertain to the current view of the data i.e., the trips from the initial bounding box query, or the trips from the particular time slot, depending upon what interactions have resulted into the current donut chart.

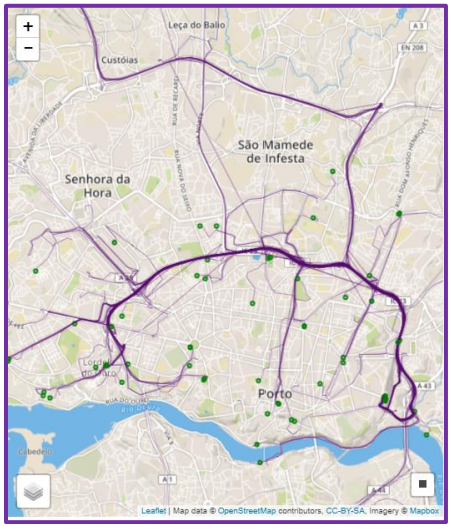

Figure 5: Map showing trajectories with starting points

Scatterplot

As mentioned over bar chart interaction, scatter plot matrix will be updated with reference to number of trips rendered within that time period of individual bar for better visualization and details.

“mouseover” function is implemented so that whenever mouse is hovered over any circle plotted, it will show the details of the trips average speed, distance and duration which provides a clear view of plotted region with details and helps avoid the clutter and differentiate between multiple trips concentrated over a single region.

“mouseout” function is necessary to remove the displayed “mouseover” values which would otherwise over shadow the further pointed views affecting the purpose of mouseover.

List

Clicking on any row in the list view draws a trajectory of that trip over the map (figure 6). The start the trip is shown using a “green dot”.

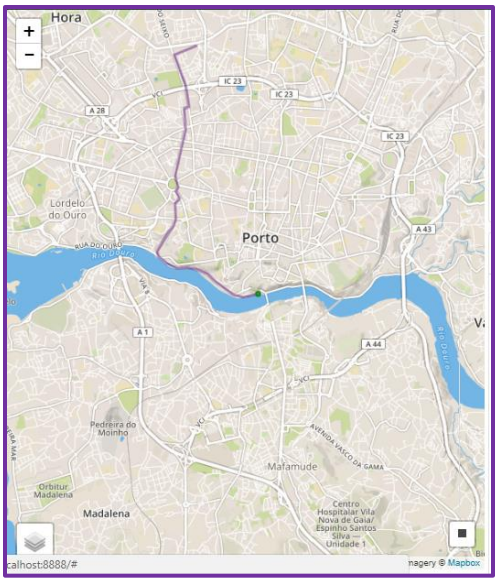

Figure 6: Map showing trajectories with starting points

References

[1] Bostock, M. (2018). D3.js - Data-Driven Documents. [online] D3js.org. Available at: https://d3js.org/ [Accessed 10 Apr. 2018].

[2] GitHub. (2018). d3/d3. [online] Available at: https://github.com/d3/d3/wiki/Gallery [Accessed 8 Apr. 2018].

[3] Adilmoujahid.com. (2018). Interactive Data Visualization of Geospatial Data using D3.js, DC.js, Leaflet.js and Python // Adil Moujahid // Data Analytics and more. [online] Available at: http://adilmoujahid.com/posts/2016/08/interactive-data-visualization-geospatial-d3-dc-leaflet-python/ [Accessed 21 Apr. 2018].

[4] Adilmoujahid.com. (2018). Interactive Data Visualization with D3.js, DC.js, Python, and MongoDB // Adil Moujahid // Data Analytics and more. [online] Available at: http://adilmoujahid.com/posts/2015/01/interactive-data-visualization-d3-dc-python-mongodb/ [Accessed 24 Apr. 2018].

[5] Bl.ocks.org. (2018). *Scatterplot Matrix*. [online] Available at: https://bl.ocks.org/mbostock/3213173 [Accessed 20 Apr. 2018].