A Web-based Visualization Tool for Moving Violation Data in Washington D.C. 2015

Zhiqi Yu
George Mason University
Dept. of Geography and Geoinformation Science
4400 University Drive, MS 6C3
Fairfax, VA 22030
zyu2@gmu.edu

ABSTRACT

In this paper, we present a web-based visualization tool for visualizing and mining moving violation data in Washington D.C. in 2015. This web application is developed using open source software, such as Leaflet and D3.js. We stored our dataset on the remote server powered by Carto. The presented web application provided functionalities include querying the dataset based on the selected time frame, and querying the attribute table of specific street segments by clicking on features on the map. We used a sequential multi-hue color scheme to color different street segments based on how many violations were issued in the given time frame. The presented web application can be used to explore the moving violation data and find interesting patterns for decision makers to use as reference for future planning.

Keywords

Web-GIS; Data visualization; Spatial-temporal data mining; Web mapping; Web frontend design

1. INTRODUCTION

Moving violations are traffic violations that are issued to moving vehicles. Some types of moving violations such as dangerous driving and speeding can lead to serious consequences if such violations happen in heavily populated area. Washington D.C, the capital of the United States, is both the center of politics and a traveling resort that attracts thousands of people to visit every day. With such a small area geographically, the Washington metropolitan area is one of the most heavily populated area (over 6 million people as of 2016) in the country [1]. Therefore, traffic safety is of great importance in the Washington D.C. And investigating the spatial and temporal patterns of moving violations are essential to understand the variation of traffic safety both geographically and temporally.

To address the intention mentioned above, we presented a web-based visualization tool that enables users to interact with the moving violation dataset to discover patterns and information, such as which streets tend to have higher violations and in which week are there more violations etc.

By presenting this visualization demo, we believe that this work will provide a tool that can give policy makers insights about transportation safety in Washington D.C. and further help keep people safe in the capital of the US.

The rest of the paper is organized as follows: Section 2 provides description of the dataset we visualized. Section 3 describes the structure of the application. Section 4 introduces the interface of the application. Section 5 outlines the demo that will be provided, and Section 6 discusses future development for this application.

PLDI '13 June 16–19, 2013, Seattle, WA, USA
© 2017 ACM. ISBN 978-1-4503-2138-9…$15.00
DOI: 10.1145/1235

2. DATASET

With the trend of open data, many governmental departments have provided open access to different types of interesting datasets. Open Data DC [2] is the official open data portal for all the open datasets about Washington D.C.

We downloaded the Moving Violations Summary for 2015 dataset from the Open Data DC. This dataset is summarized by ticket counts based on week of year, and is aggregated to the street segments where the violation happens. The dataset is in polyline Shapefile format. The bulk of the attribute table is the set of conditions under which violations happen, such as low speeding during rush hours. The conditions are organized as [violation type] + [time]. The violation types include:

- Low Speeding (Under 20mph) - speed violations under 20mph
- High Speeding (above 20mph) - speed violations over 20 mph including reckless driving
- Unsafe Driving - violations for driving maneuvers unsafe to traffic
- Unsafe Vehicle - violations for vehicle characteristics unsafe to traffic
- Unsafe Operator - violations for operator (driver) char-

acteristics unsafe to traffic

- Other - miscellaneous violations

The time is given as:

- Am no rush - in the morning but not rush hours
- Am rush - in the rush hours in the morning
- Pm no rush - in the afternoon but not rush hours
- Pm rush - in the rush hours in the afternoon
- Evening - in the evening
- Overnight - in the late night
- No time - no time given for this violation

The combination of these two conditions makes 42 different conditions that categorize violations. For each row in the dataset, the combination of street segment id and the week number in the year uniquely identifies the row, and in each of the 42 violation condition fields of one row, the number of violations issued under this condition in the given combination of week number and street segment id is recorded.

Unlike many other geospatial dataset that one object is represented by one feature, our dataset is organized in a way that many rows can represent a same physical object in different times. This is because the rows in our dataset is uniquely identified by the combination of street segment id and week number. Therefore, many rows will have same street segment id while having different week number. Thus, there are overlapping features when the data is mapped, and each layer of the overlapped feature represent one week summary of moving violations on the corresponding street segment.

3. APPLICATION STRUCTURE

3.1 Application Stack

Database: Carto is an online platform for users to store and visualize geospatial dataset. It is based on PostgreSQL and PostGIS [3], and it supports standard SQL queries as well as extended spatial queries with OGC OpenGIS standards [4]. The platform provides users with limited storage space and functions including all kinds of APIs (application programming interfaces). In our development, we only use Carto as a spatial database. And we use the provided SQL API [3] to query the database remotely. By using Carto, we can simply drag-and-drop our shapefile to the browser and the server will create the database automatically.

Web mapping: We used the open source library, Leaflet [5] for web mapping.

Data manipulation: We used the D3.js (Data Driven Document) [6] open source library to load JSON data and we used the built-in brushing function to develope the interaction on the temporal dimension of the data.

3.2 Interaction Logic

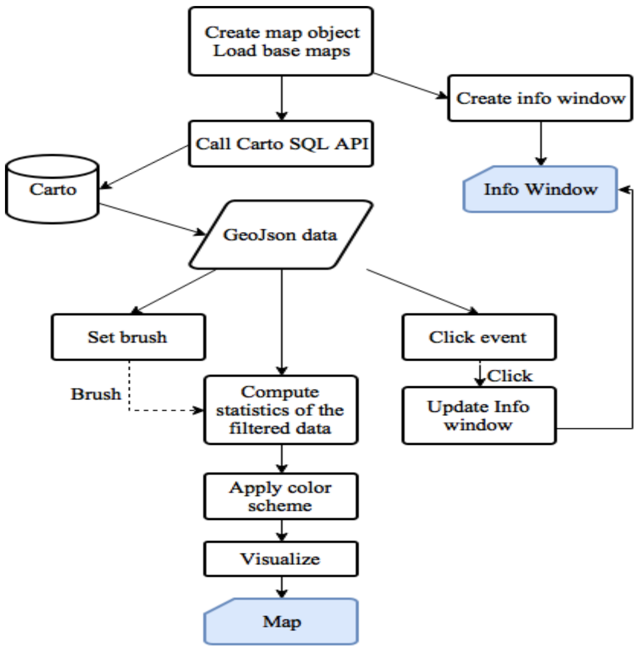


Figure 1: Application Flow

The presented application is an interactive data analytic tool. The interaction follows the following logics: Select by location: this interaction enables users to move mouse freely on the map to select street segments of interest. The selection is based on feature, and selection will update the info window. Select by time: this interaction enables users to make selection on the time bar to limit data to the selected time frame shown on the map. The selection is based on the whole data and thus the map will be repainted after each selection.

3.3 Application workflow

Our web application was designed to work in a flow described in the Figure 1. The application starts with using Leaflet library to create a map object on the page, and load predefined WMS tile layers as base maps. Then we create an info window that overlays on the map to show information about selected features. Next, we use Carto SQL API to query the database we created on Carto to retrieve data for visualization. At the startup, we use a simple query,

```
SELECT * FROM moving_violations_summary_for_2015
```

to retrieve all the data. The later process will all be done based on this GeoJSON data.

As we have the data in GeoJSON format, we load it as a GeoJSON layer, and we compute statistics of the data. This step is critical because as we mentioned before, there are overlaying features in the data, which results in that when users do mouse selection on the map, there will only be the top one feature selected while the other overlapping ones will not get selected. Whereas, when we do mouse selections, we do selections on street segments, which means we need to select every overlapping feature. Thus, we need to select every row in the dataset with the selected street segment id. To do that, we implemented a computation module that when user select one feature, it retrieves the street segment id of the selected feature, and use it to filter the whole dataset to get a sub dataset that contain all data entries with the selected street segment id. Later on, we compute summations over each condition across all the data entries in the sub dataset to get overall counts for this street segment, and we use them to update the table in the info window.

To represent the magnitude of numbers of violations on each street segment, we applied a sequential multi-hue color scheme generated by Color Brewer [7]. Additionally, we use the opacity of the street segments to represent how many weeks does the street segments have violations. This is achieved by assigning each feature a small opacity value, so that the more overlapping features, the less opaque of the feature.

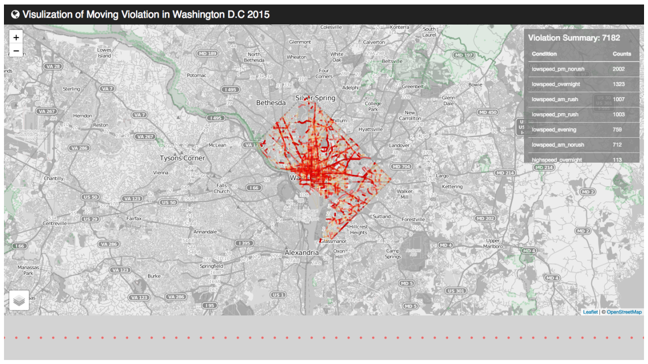
We implemented several events for every feature. First, we implemented a 'mouse over' event that will be invoked when the mouse is on the feature, second, we implemented a 'mouse leave' that will be invoked when the mouse leaves the feature that was previously selected. Last, we implemented a 'click' event that invokes when users click on the feature. Specifically, when users hover mouse on one feature, the 'mouse over' event will be triggered and the selected feature will be highlighted using a bright color, and the info window will be updated to show the violation summaries of the selected feature. When users move mouse out of the selected feature, the previously selected feature will be reset to the original color. When user want to take a close look at the summaries for a feature, user can click on the feature then the selection will be locked so that when the mouse leaves the feature, the selection won't change. To release the lock, we implemented another event which will be invoked when users press 'Enter' button, and the lock will be gone so that users can browse other features freely.

We implemented a brushing function that is provided in D3.js library. Specifically, when users make a selection on the time bar, the selected time frame will be applied on the filter on the original data so that only the violations issued in the given time frame will be visualized.

4. INTERFACES

The web page of the application is built with components from Bootstrap library [8]. The overall look includes a simplified navigation bar with only a title of the application, a map container that takes the bulk of the page, and a time bar right below the map container. The color theme of the page is tuned to light grey.

Specifically, the map container has a zoom control on the top left, and the layer control is on the bottom left. There are 3 base layers can be chosen from the layer control. The info window is on the top right and it contains an overall count of violations on the selected street segment and a table that show in detail the count of violations of different conditions. The table is automatically sorted in descent order. In the middle of the time bar at the bottom of the page is a series of dots with each one of them represent one week in the year.


Visulization of Moving Violation in Washington D.C 2015

Violation Summary: 7182

Condition	Counts
lowspeed_pm_norush	2002
lowspeed_overnight	1323
lowspeed_am_rush	1007
lowspeed_pm_rush	1003
lowspeed_evening	759
lowspeed_am_norush	712
highspeed_overnight	113

Figure 2: Overall look of the application

The GeoJSON layer is styled with customized color scheme and opacity. Specifically, the street segments that have low violation numbers will be colored close to yellow, and the street segments with high violation numbers will be colored close to red. The quantiles of the violation numbers are used to apply different colors to data that fall into different quantiles. Instead of using the 25%, 50%, 75% as divide for quantile method, we used the 20%, 40%, 60%, 80% and 99% to divide the data. This is because the data is extremely skewed. The original quantile method will not be able to capture the outliers because the outliers will be colored the same as the normal high values. We added a 99% divide so that the extremely high values will be colored differently against the normal high values. The opacity is set to a low value so that street segments that with violations in more weeks will be less opaque because the opacity will be sumed based on how many features are overlapped.

Selections made by moving mouse on features will highlight the feature with a solid gold color to separate it from the rest of the features.

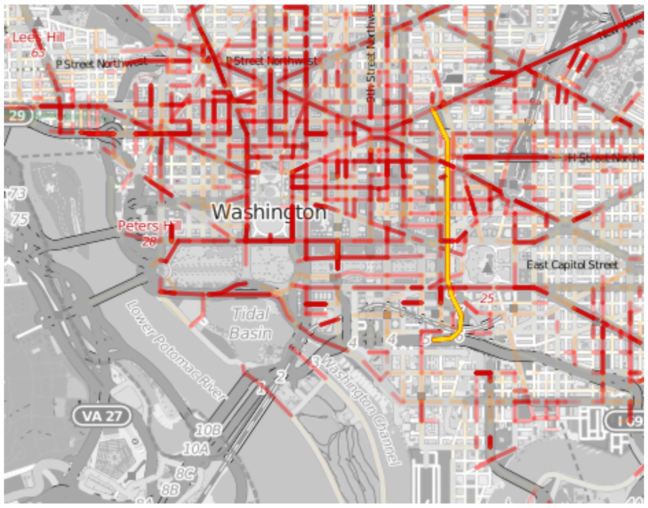
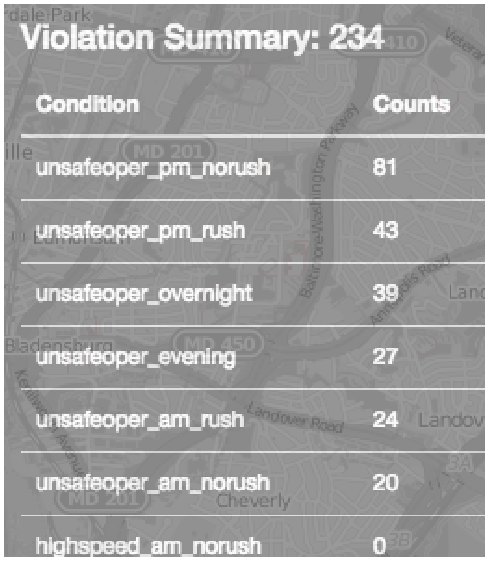


Figure 3: Highlight features

If users click on the feature, the selection will be locked and users can scroll down the table in the info window to see details of the selected street segments. Specifically in the info window, we used a title 'Violation Summary' and followed by a total count of moving violations. Below the title is a table with two columns, one is the name of the condition, the other is the count of moving violations issued under the corresponding condition. The table is sorted in descent order so that the higher number of moving violation and their corresponding condion will displayed at top.



Violation Summary: 234

Condition	Counts
unsafeoper_pm_norush	81
unsafeoper_pm_rush	43
unsafeoper_overnight	39
unsafeoper_evening	27
unsafeoper_am_rush	24
unsafeoper_am_norush	20
highspeed_am_norush	0

Figure 4: Info Window

Brushing on the time bar will result in leaving a selection rectangle on the time bar and the layer will be refreshed.

5. APPLICATION DEMO

The demo will show the interface of our application and major functionalities mentioned in the previous sections. We hope the demo will serve to show how our application can be used to explore the moving violation data.

We will first open the web page, zoom in and out to see how the data is colored. Then we will move mouse on the map to select a few features with different colors to see how the info window changes. We will then click on features to enable scrolling in the info window for browsing more information about the selected feature. Afterwards, we will demo the brushing function. We will select different time frames, and move the selections to see how the visualization changes, thus indicate how the moving violations change overtime.

6. FUTURE DEVELOPMENT

The presented web application is still in its early stage of evolvement. To enable the full potential of interactive data analysis, we believe several things can be improved or added in the future development of this application:

- Add a visual modality to the time bar such as the size of the dots, or the height of bars located at the dots to represent how many violations in total in each week.
- Add a function that enable users to check only one or a few conditions under which violations were issued so that users can explore the spatial temporal pattern of the specified condition.
- Add a chart to show the distribution of the moving violations in the selected time frame to discover if the violations are normally distributed or skewed.

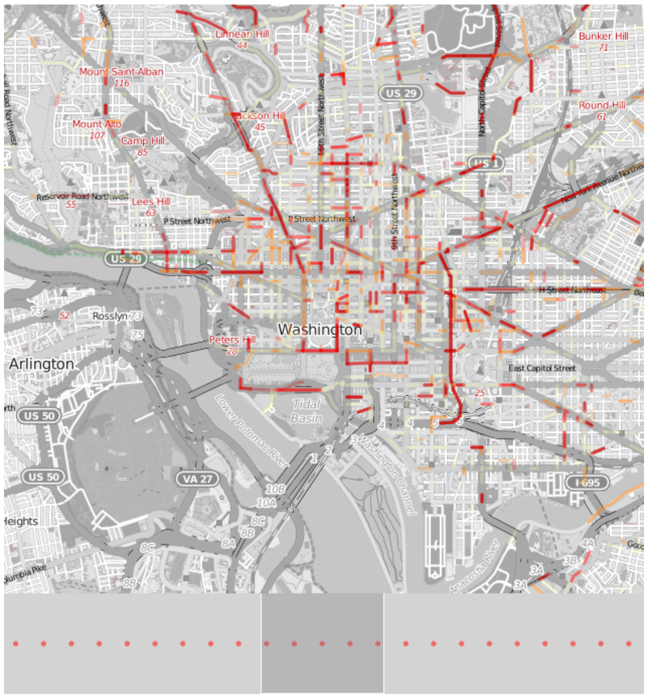


Figure 5: Time bar brushing

7. REFERENCES

[1] Wikipedia. Washington, D.C. — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Washington\%2C\%20D.C.&oldid=809875155, 2017. [Online; accessed 04-December-2017].

[2] Open data dc. http://opendata.dc.gov/.

[3] Making calls to the sql api. https://carto.com/docs/carto-engine/sql-api/making-calls/.

[4] Open Geospatial Consortium et al. Opengis implementation specification for geographic information-simple feature access-part 2: Sql option. *OpenGIS Implementation Standard*, 2010.

[5] Leaflet. http://leafletjs.com/.

[6] Mike Bostock. Data-driven documents. https://d3js.org/.

[7] Cynthia A. Brewer. Colorbrewer. http://colorbrewer2.org/.

[8] Jacob Thornton Mark Otto and Bootstrap contributors. Bootstrap. https://getbootstrap.com/.