GEOG5870 Assessment 3 Technical Report

# Introduction

My website explores the issue of presenting large amounts of point data on an online map. Originally it aimed to plot all of the crime reports in Northumbria for February 2017 and allow for the manipulation and analysis of this data. However, at over 15,000 individual points the performance issues of trying to work with such a large data set became evident.

Instead I decided to base my project on a comparison between four different methods of presenting large amounts of data on an online mapː as basic markers, as clustered markers, as a choropleth map, and as a heat map. Accompanying the implementation of each method on the website is a short description which briefly discusses the utility and limitation of each method in terms of use in visualising the data.

# Data utilised

The primary data source utilised by this website is a subset of a crime data set for crime reports in Northumbria for February 2017. The original data set has over 15,000 entries but this website utilises only the crime events categorised as ‘criminal damage and arson’, of which there are 1,675 individual events. The data was sourced from data.police.uk and is available under the Open Government Licence (Police.uk, No date).

Additionally the Postcode District boundaries from 2012 are utilised for the choropleth map and these were sourced from the UK Data Service (UK Data Service, No date).

# Implementation

## HTML, CSS, and JQuery

The HTML and CSS of the website are fairly basic. The only aspect of serious note are a set of four HTML buttons and description Divs for each of the different visualisation options. These allow the user to move between different visualisations and descriptions by clicking the buttons.

They are implemented with JQuery in the Setup() function. The JQuery finds which button was clicked and runs a set of methods in order to display the correct map. Additionally it also hides the description Divs except for the one relating to visualising option that the user selected.

## Point map

There are a number of stages to the implementation of the point map:

1. The data was converted from CSV format into JSON format using ConvertCSV.com (Data Design Group, 2017).
2. The JSON was stored in a JavaScript file (crimDamage.js) as a variable for easy access by the rest of the code.
3. Each feature of the JSON is looped through and assigned to a Leaflet marker along with relevant coordinate and popup information. Each marker is then added to a Leaflet LayerGroup.
4. The LayerGroup is added to the map.

A major issue encountered when implementing this map was that, given the size of the data set and the number of markers that were required to be drawn, the map became very slow and unresponsive. While this is to some extent just an unfortunate reality of attempting to display thousands of points at once, the responsiveness can be increased by displaying the points as Leaflet DivIcons rather than traditional Markers.

A Marker is an image whereas a DivIcon is simply a CSS shape and therefore significantly easier for the computer to load. Changing the marker icon to a DivIcon significant improved the performance of this map.

## Clustered point map

The implementation of the clustered point map is vritually identical to that of the basic point map, except that, instead of assigning each marker to a LayerGroup, each marker is instead assigned to a L.markerClusterGroup.

This map significant improves the performance at the expense of small-scale geographical visibility of the data.

## Choropleth map

The Choropleth map is perhaps the most complicated form of visualisation on the website. It works by loading a GeoJSON containing the 2012 Postcode District boundaries for relevant postcodes into the map, calculating how many points intersect with each polygon, adding this number to a property of each polygon, and then colouring the polygons based on the count of that property.

While the intersecting calculation could be done very easily with a traditional GIS package such as ArcMap, for this project I was interested to see whether it could be done online using JavaScript. A successful implementation was eventually reached but it is worth discussing some of the difficulties which were encountered along the way.

The first issue relates to projecting data from the UK Data Service onto a Leaflet map. The Shapefile was first converted into a GeoJSON using mapshaper (mapshaper, 2017) and then stored in postcode.js as a variable for easy access by other JavaScript methods. Simply adding this GeoJSON to the map would not work however as the shapefile from UK Data Service was projected with the British National Grid rather than the EPSG:4326 projection which Leaflet uses by default. The shapefile therefore had to be re-projected into the correct projection before being converted into GeoJSON. In this case this was done within ArcMap but is also possible with free software such as QGIS or the GeoTools Java library.

The second issue was writing an algorithm which, given a point and a polygon, was able to figure out whether the point was within the polygon. This calculation (referred to as PointInPolygon, or PIP) was implemented based on an answer by user VitalyZ on StackOverflow (VitalyZ, 2017). This calculation uses a Ray Casting algorithm for checking PIP.

Thirdly, while the PIP method superficially worked, I noted that many polygons which I would expect to have high crime numbers were being assigned a value of 0 despite having overlapping points. On investigation it turned out that this was due to the polygons being non-contiguous which was not supported by the PIP algorithm. The solution was to manually simplify the polygon layer and remove unimportant islands and connecting other non-contiguous areas together.

At this stage the Choropleth map was successfully implemented and calculated PIP accurately. A style was attributed to each polygon of the GeoJSON feature using Leaflet’s onEachFeature method and a series of functions which assigned a specific colour based on the number of crimes aggregated to each polygon.

## Heat map

The heat map layer was relatively simple to produce. It utilised the leaflet.heat plugin (Agafonkin, 2017)

References

Police.uk. No date. All crimes in February 2017. Northumbria Police. [Online]. [Accessed 07.05.2017]. Available online at <https://data.police.uk/data/>

UK Data Service. No date. 2012 UK Postcode District Boundaries. [Online]. [Accessed 07.05.2017]. Available online at <https://www.ukdataservice.ac.uk/>

Data Design Group. 2017. Convert CSV to JSON. [Online]. [Accessed 07.05.2017]. Available online at <http://www.convertcsv.com/csv-to-json.html>

VitalyZ. 2017. Determine if a point reside inside a leaflet polygon. Stack Overflow. [Online]. [Accessed 07.05.2017]. Available online at <https://stackoverflow.com/a/42532563>

Mapshaper. 2017. mapshaper. [Online]. [Accessed 07.05.2017]. Available online at <http://www.mapshaper.org/>

Agafonkin, V. 2017. Leaflet.heat. [Online]. [Accessed 07.05.2017]. Available online at <https://github.com/Leaflet/Leaflet.heat>