Heriot-Watt University

Masters Thesis

Interface With Google Maps

Author:
Abdur Rehman Shahid

Supervisor:
Dr. Murdoch Gabbay

2nd Marker:
Prof. Just Mike

A thesis submitted in fulfillment of the requirements for the degree of MSc. Data Science

in the

School of Mathematical and Computer Sciences

August 2018

Declaration of Authorship

I, Abdur Rehman Shahid, declare that this thesis titled, 'Interface With Google Maps' and the work presented in it is my own. I confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: _____

Date: 12th April, 2018 _____

“Data at the moment is about developers and it needs to be about communities."

Leigh Dodds

Abstract

The aim of this project is to create a exploratory-interactive geospatial-information visualization application using heat maps created from a combination of disparate open data sources and open source mapping technology such as Google Maps, to exploit the potential socio-economic value buried inside monotonously large datasets. The data gathered for this application is a combination of non-state open data and state owned Open Government Data(OGD) sources.

This application lets a user discover new knowledge or update an old one. The idea here is that no single component, neither application nor the map interface, is complete on its own. We need to have an application with all its components working together, i.e. map interface and rest of application interface, to assist a user create knowledge.

The idea here is to not just visualize regional trends but to make a user explore and analyze using interactivity to enhance the user experience and to control the perspective of information representation by switching themes of visualization.

The web application will offer two different functionalities served in a single view. One of them is exploratory visualization as a user can explore thematic data such as health, crime, employment and education, through interactive heat map visualizations. The other functionality, loading news section, will allow a user to click a location on the (already loaded heat) map and list news regarding that area and the selected theme of deprivation index, e.g health, crime etc.

After introduction in chapter 1 this dissertation report discusses the background and context of this project in chapter 2 which will focus on the subjectivity of this project. Chapter 3 formally lays down the project requirements.

Acknowledgements

I like to thank my supervisor Dr. Murdoch Gabbay, it would not have been possible without his kind support. I will also like to thank the second marker Mr. Just Mike for his useful feedbacks and suggestions. Finally I would like to thank all my family and friends for their prayers and support.

Contents

Declaration of Authorship i

Abstract iii

Acknowledgements iv

Contents v

List of Figures vii

List of Tables ix

Abbreviations x

1 Introduction 1
1.1 Objectives . . . 1

2 Literature Review 3
2.1 Geographic Visualization . . . 3
2.1.1 Heat Maps . . . 4
2.2 Google Maps . . . 9
2.2.1 Creating Heat Maps With Google Maps Javascript API . . . 11
2.2.2 Interaction with Google Maps . . . 13
2.2.2.1 Interaction with the data . . . 14
2.2.2.2 Interaction with the data representation . . . 14
2.2.2.3 Interaction with the Temporal Dimension . . . 14
2.2.2.4 Contextualizing Interaction . . . 14
2.2.3 Reverse Geo coding . . . 16
2.2.3.1 Address Components . . . 17
2.2.3.2 Formatted address . . . 17
2.2.3.3 Geometry: . . . 17
2.2.3.4 Place ID: . . . 18
2.2.3.5 Types: . . . 18
2.2.4 Places Search Box . . . 19
2.3 Other mapping technologies . . . 21
2.4 Data . . . 23

2.4.1 Open Data: Comparing Open Government, Open Data and Big Data . . . 24
2.4.2 Geospatial Open Government Data and Its Utilization . . . 27
2.4.3 Geospatial non Government Open Data and Its Utilization . . . 29
2.4.4 Data Preprocessing and Management . . . 30
2.4.4.1 Data Preprocessing . . . 30
2.4.4.2 Data Management . . . 31

3 Requirement Analysis **33**
3.1 Project Scope . . . 33
3.2 Project Objectives . . . 34
3.3 Project Deliverables . . . 34
3.4 Project Risk Assessment . . . 35
3.5 Requirements . . . 36
3.5.1 Heat Map Visualizations . . . 37
3.5.2 News . . . 37

4 Professional, Legal and Ethical Issues **38**
4.1 Professional . . . 38
4.2 Legal . . . 38
4.3 Ethical . . . 39

5 Implementation **40**
5.1 Data Requirement . . . 41
5.2 Data Acquisition . . . 41
5.3 Data Preprocessing . . . 42
5.4 Application . . . 45
5.4.1 Server Side . . . 46
5.4.2 Client Side . . . 50

6 Evaluation And Results **59**
6.1 Experiments . . . 60
6.1.1 What is the Country Level Health Situation? . . . 60
6.1.2 What is the crime situation in "City of London"? . . . 62
6.1.3 what is the Employment situation in or around an address? . . . 62
6.1.4 compare the Education situation between neighboring districts "Hounslow" and "City of West Minster" . . . 62
6.1.5 User wants to read "Crime" news related to Bradford . . . 66

7 Conclusion and Future Work **70**

A Reverse Geo Code Result **73**
A.1 Google Heat Map Layer Weighted Locations . . . 73
A.2 Google Search place Sample . . . 74

Bibliography **76**

List of Figures

2.1 A. Population By Ethnicity . . . 5
2.2 B. English IMD map . . . 5
2.3 C. Brexit voter distribution . . . 5
2.4 East Lindsey: Comparing Maps . . . 5
2.5 A. Population by ethnicity . . . 6
2.6 B. English IMD map . . . 6
2.7 C. Brexit voter distribution . . . 6
2.8 Eden: Comparing maps . . . 6
2.9 Showing a feature on map(grid). Source: Ginner and Henderson [2018] 8
2.10 Showing feature overlap. Source: DeBoer [2015] . . . 8
2.11 Different zoom levels(zooming out left to right). Source: DeBoer [2015] 9
2.12 Normal radius - 10px Source: Google Developer Console [2018] . . . 9
2.13 Normal radius - 20px Source: Google Developer Console [2018] . . . 9
2.14 A festival seating plan . . . 10
2.15 Geo Spatial Temporal Map Source: Vieira et al. [2008] . . . 15
2.16 Geo Code . . . 19
2.17 Reverse Geo Code . . . 19
2.18 Reverse Geo Code Result: Premise . . . 20
2.19 Reverse Geo Code Result: Street Address . . . 20
2.20 Reverse Geo Code Result: Postal Code . . . 20
2.21 Reverse Geo Code Result: Postal Code Prefix . . . 20
2.22 Reverse Geo Code Result: Postal Town . . . 20
2.23 Reverse Geo Code Result: Administrative level 2 . . . 20
2.24 Reverse Geo Code Result: Administrative level 1 . . . 20
2.25 Reverse Geo Code Result: Country . . . 20
2.26 Source: Google Developer Guide [2018a] . . . 20
2.27 Heat Map made with Arc GIS Javascript API . . . 22
2.28 Venn Diagram differentiating between apparently similar types of open source data. Source Joel [2014] . . . 25
2.29 English IMD map . . . 28
2.30 Live UA interactive map showing twitter news mapped on Syrian map 29

6.1 First page . . . 61
6.2 Health heatmap layer . . . 61
6.3 Crime heatmap layer . . . 63
6.4 Crime in City of London . . . 63
6.5 3a. . . . 64
6.6 3b. . . . 64

6.7 4a. 65
6.8 4b 65
6.9 5a. 67
6.10 5b. 67
6.11 5c. Crime News for "Bradford" 68
6.12 5c. Changed Radius 68

List of Tables

3.1 Risk and Mitigation 36

5.1 Datasets Summary 41

Abbreviations

OGD	**O**pen **G**overnment **D**ata
API	**A**pplication **P**rogram **I**nterface
ETL	**E**xtract **T**ransform **L**oad
LSAO	**L**ower Layer **S**uper **O**utput **A**reas
IMD	**I**ndex of **M**ultiple **D**eprivation
GIS	**G**eographic **I**nformation **S**ystem
lat	Latitude
long	Longitude
<lat,long>	Latitude Longitude pair representing a single location
RDBMS	**R**elational **D**ata **B**ase **M**anagement **S**ystem

Dedicated to my Father, Dr. Shahid Mahmood

Chapter 1

Introduction

1.1 Objectives

1. To create a **visualization**: Where data mining is for computer systems to analyze the large datasets, data visualizations are for humans to analyze hidden trends in the data(Alazmi and Alazmi [2012]). This idea comes from grounded theory and visualizations, where grounded theory is 'the discovery of theory from data systematically obtained from social research' (Glaser and Strauss [2017]). The social research can be informal or a formal one where an informal social researcher nowadays is anyone with a social thought who is exploring articles on the Internet to form a unique hypothesis of his own (Lee et al. [1995];Smith [2016]). Data visualizations like any other data analytics help with decision making or knowledge creation or updation on top of existing perceived knowledge by social research or years of personal experiences(Lee et al. [1995];Dupin-Bryant and Olsen [2014]).

2. which is **Exploratory**: The datasets at our disposal are huge. They have to be dissected to get valuable information out of them. A human can normally absorb only a selection of whole data so the visualizations display one IMD theme at a time. This can also be taken as a single perspective of a multi dimensional dataset. So, the objective of any such exploratory visualization is to extend visual thinking and visual communication where a user asks the GIS system a question via interaction while the system changes its visual representation as an answer(Smith [2016];Koua et al. [2006]).

3. which is **Interactive**: the exploration cannot occur unless a user interacts with the system. So, the resulting mapping application must provide it's user

with interactivity which acts as a tool to visually communicate with our GIS to explore the datasets behind the visualization(Smith [2016];Roth [2013],2015).

To achieve the above objectives of creating a geographic information visualization that would unearth regional trends, the first task is to choose an appropriate dataset. Second, an appropriate type of visualization to represent that dataset is required. A visualization that can efficiently represent and explore different sections of the data with different perspectives. Given this condition and some research on types of visualizations I chose 2 datasets 1) English Indices of Multiple deprivation (data.gov.uk [2018]) and 2) National Survey Postcode Lookup (ONS Geography [2018]), both are available free of cost and open to use under the Open Government License (The National Archives UK [2015]). These are UK open government data sets and their combination makes up a single huge data set with 1.4 million rows. This data is used to populate each heat map layer, 1 for each of the 4 themes of IMD, i.e. Health, Crime, Education and Employment. The detail of creating the data mashup and visualizing it in my application is given in Chapter 5

The interactivity gets covered by choosing Google Maps interface, the market leader in online map making. By default it provides some interactions like zooming, panning, toggling, changing display which combined with interface of the application, fits all four dimensions of interactivity discussed in Chapter 2. Next chapter 2 discusses the subject and related work of this dissertation project, i.e. online-exploratory-interactive-geographic-visulaization applications presenting large and open datasets of socio economic value visualizing general to local trends.

The web application can be downloaded from `https://github.com/abdurrehmanshahid315/maps.git`. The instructions to run this application are found in the read me file. The data preprocessing file and datasets can be found at `https://drive.google.com/open?id=1UpZYyLr9zgasqX8nKOet86Tg6dK8wV-M`. The data sets need to be in the same folder as the "datapreprocessing.py" file.

Chapter 2

Literature Review

"The objective of this project is to create an **online-exploratory- interactive- geographic visualization** application representing **large** and **open datasets of socio economic value** depicting general to local trends". This is the total description that would be enough to define the scope of this project. Hence, this chapter tries to subjectively locate this application by breaking down this project description.

This section first of all discusses geographic visualizations followed by discussion on google maps as a map technology and it's services that we used in our application, it further compares google maps with other mapping technologies. This will include the interactions provided by these mapping applications, that allows the exploration of visualization as proposed in objectives. Finally the backbone of this application, the large (and open) data is discussed and how to manage such large data with a open/big data compatible DBMS that is scalable and performs well in crunch (run time) moments.

2.1 Geographic Visualization

Data Visualization, our first dimension, of the application is a "data mining application" by which any data may be represented in an appealing and suitable graphical manner(Alazmi and Alazmi [2012]). For example we have a data mashup,i.e. a combination of 2 data sets(IMD and NSPL), a single table which contains information about locations that are (almost) continuous. Tobler's first law of geography (Sui [2004]) "all things are related but near things are more related than distant things" extends this idea of continuity which clarifies the ultimate goal of visualization. This continuity is also a defining condition for choosing a suitable visualization technique, which according to DeBoer [2015] is a heat map.

According to (Sen et al. [2017]) thematic maps, such as heat maps can provide a viewer with 3 things:

1. **Specific Value:** As mentioned earlier Google maps heat map layer knows this specific value, as a weight, for each unique geographic location <lat,long,weight>. For example there's a heat map showing distribution of immigrants living in various locations in England. In this case a population percentage will be a specific value, i.e. weight in Google heat map.

2. **Regional Trends:** On exploring a map such as the heat map one can understand the visual trends. For e.g. in the map below one can see that the closest regions outside central London(City Of London) are the ones that are worse at health. Or comparing this map with a map representing population density by ethnicity, we can see the trend, more immigrant or non white the area is more worse the health situation is.

3. **Mental Maps:** The map you explored for visualizing regional trends was from an authentic source or not, psychologically, a user's mental map will be strengthened. For example, I already had this idea that most voters in favor of Brexit were from predominantly white majority areas, by comparing the map showing Brexit voter distribution (theguardian [2016]),English IMD map(data.gov.uk [2018]) for overall deprivation and population by ethnicity map(theguardian [2011]) I can add this information to my previous knowledge, strengthening my mental map.

   For example let's compare these maps shown in Figure 2.2 and 2.4 together for spatial analysis. What we see here is that East Lindsey has 81 local authorities (LSOA) out of which most are at least 40 percent most deprived localities. While in Eden only 1 locality is in overall 40 percent least deprived group. 70% East Lindsey voters voted in favor of Brexit. Maybe because their living conditions are most poor nationally (as shown by IMD). Comparing the three maps visualizing regional trends for a given perspective I can test my hypothesis and strengthen my mental map that mostly white people living in poor areas voted for Brexit.

2.1.1 Heat Maps

Heat maps display difference in density, intensity, value or rank "via differences in color" to let a user visualize the data, test hypotheses, and un earth trends that may otherwise be buried within large data sets, while maintaining measurement

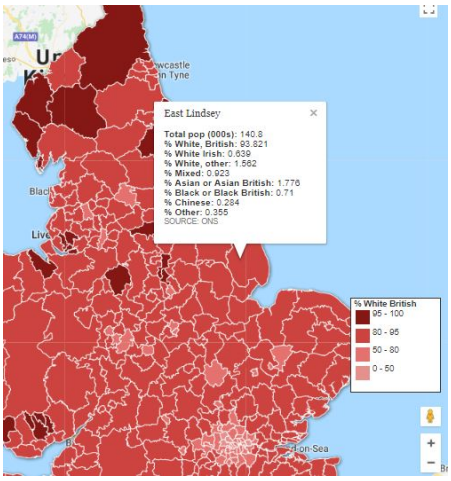

Figure 2.1: A. Population By Ethnicity

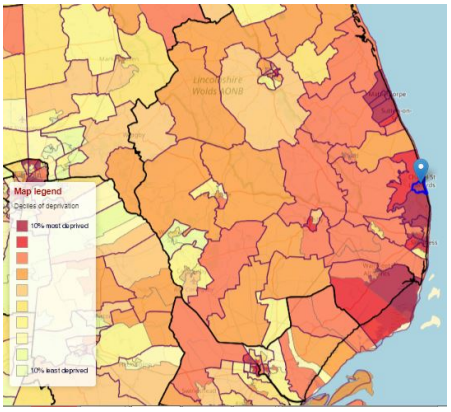

Figure 2.2: B. English IMD map

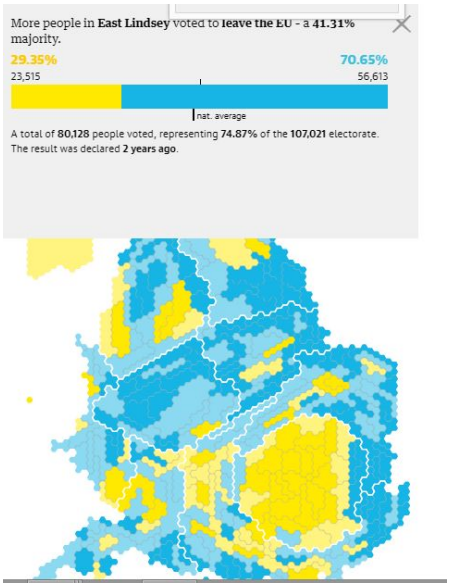

Figure 2.3: C. Brexit voter distribution

Figure 2.4: East Lindsey: Comparing Maps

relationships and data integrity (Paul [2015]; Dupin-Bryant and Olsen [2014];Ginner and Henderson [2018]).

Heatmaps are also known as "point density interpolation" in more technical language (DeBoer [2015]) and is very identical to hot spot mapping (ESRI [2018]). In fact, the google heatmap layer has a variable "maxIntensity" if given a value that changes the map behavior from a heat map (that disappears on zooming in) to a hot spot map where the individual data points do not disappear(on zooming in) and can be seen with a given radius.

Heat maps and hot spot mapping both show the statistical significance of an area on

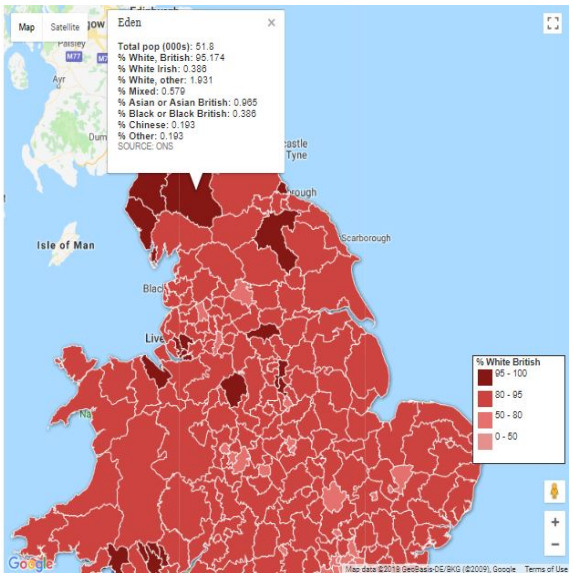

Figure 2.5: A. Population by ethnicity

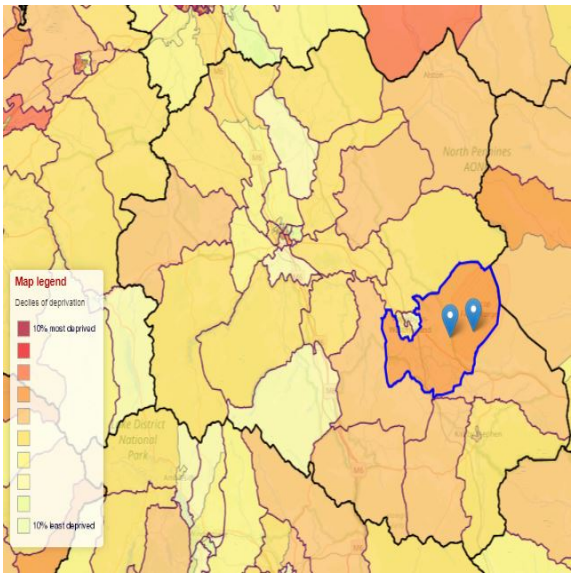

Figure 2.6: B. English IMD map

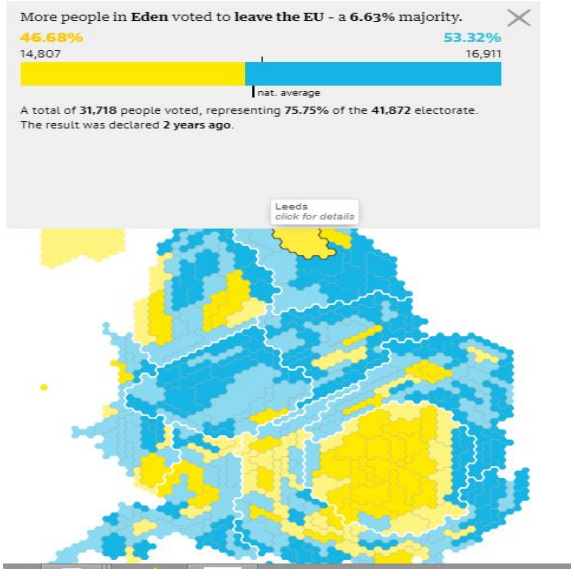

Figure 2.7: C. Brexit voter distribution

Figure 2.8: Eden: Comparing maps

a raster. The regions with high significance are marked with a higher intensity color (most hot e.g. red) while regions of low statistical significance are marked with a low intensity color(e.g. green). A heat map or a hot spot map can be considered as a grid with each pixel being a unique feature(such as each pixel is treated like a unique lat,long pair in GIS) with a buffer around it. These buffer circles when lie next to each other, they overlap, and form a resultant color. The intensity of the color changes with respect to average result of overlapping as seen in the figure below 2.9 (Ginner and Henderson [2018];DeBoer [2015]). There is one problem with heat maps that the color produced by overlaps might be misleading. This phenomena can be noticed in the figure 2.10 where neighboring buffers overlap and give a shade while there is no data point there. But this can be minimized by zoom in or keeping smaller radii for each data point. The individual data points get separated and the buffers overlap less than before. The effect of zooming in and changing radii can be seen in the figures below 2.11 and 2.12.

Heatmaps are often confused with other types of visualizations such as choropleth (DeBoer [2015]) which use a similar intensity scale for color palette, giving a feeling of rising "heat" for example green to red or gray scale, but the transition is not smooth between colors traveling longitudinally. The smoothness in transition, is related to the smoothness or continuity of the data set. The figures 2.4 show choropleth maps from different sources. Hence, a heatmap is maintaining the data integrity depicting a continuous data with a smooth(continuous) transition of colors.

"A heatmap is a visualization used to depict the intensity of data at geographical points. When the heatmap layer is enabled, a colored overlay will appear on top of the map. By default, areas of higher intensity will be colored red, and areas of lower intensity will appear green" (Google Developer Console [2018]). In our project I had to reverse this scale where a lower value depicts higher intensity of problems in a region. So red represents most deprived regions and green represents least deprived regions.

Exploratory visualization is useful in any case where there is a large dataset with multi dimensional data. For e.g. IMD dataset has crime,health score and rank values and so on, these columns act like different perspectives or dimensions of data which can be switched by interaction. Open Government datasets are a product of "Big Data" era so they should be presumed to be large and the application uses this multidimensional property of large datasets for bringing the exploratory nature of our visualization to the forefront.

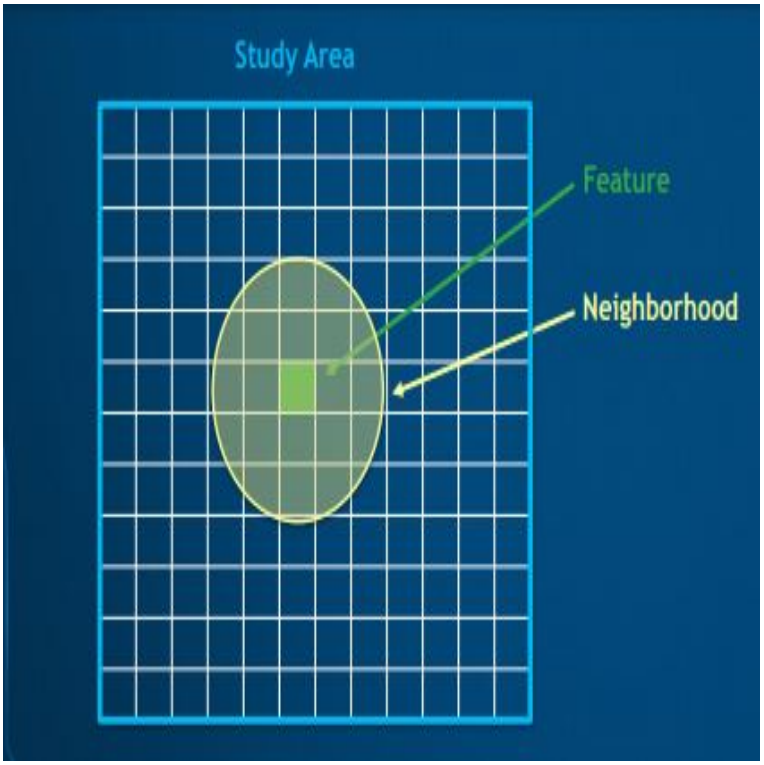

Figure 2.9: Showing a feature on map(grid). Source: Ginner and Henderson [2018]

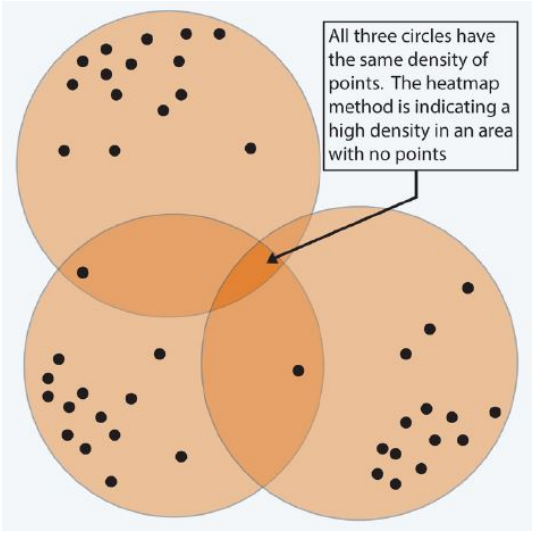

Figure 2.10: Showing feature overlap. Source: DeBoer [2015]

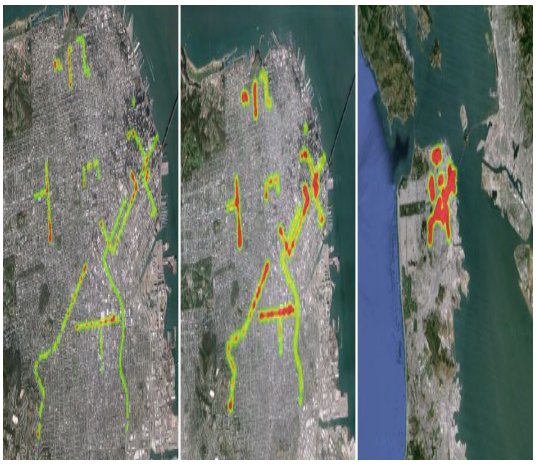

Figure 2.11: Different zoom levels(zooming out left to right). Source: DeBoer [2015]

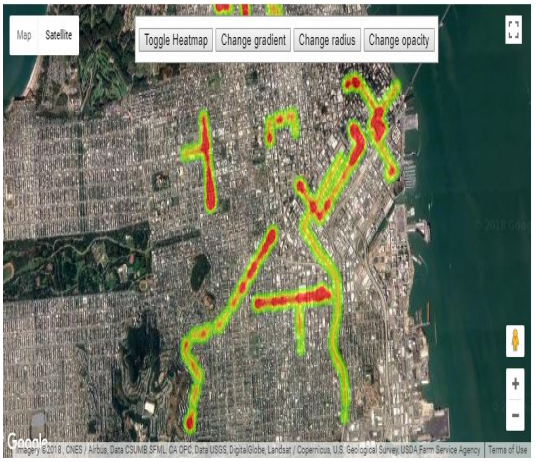

Figure 2.12: Normal radius - 10px Source: Google Developer Console [2018]

Figure 2.13: Normal radius - 20px Source: Google Developer Console [2018]

Exploration of geographic data can be achieved by enabling interactions, for example selecting data, changing representation by zooming in and out. These interactions are provided by the mapping technologies, in our case that is google maps. Hence, it is important to discuss them here.

2.2 Google Maps

Google Maps launched in 2005, has revolutionized the usage of geographic information (Dodsworth and Nicholson, 2013) for creating interesting visualizations. Due to its, easy to use interface, people have used it for purposes beyond map. They

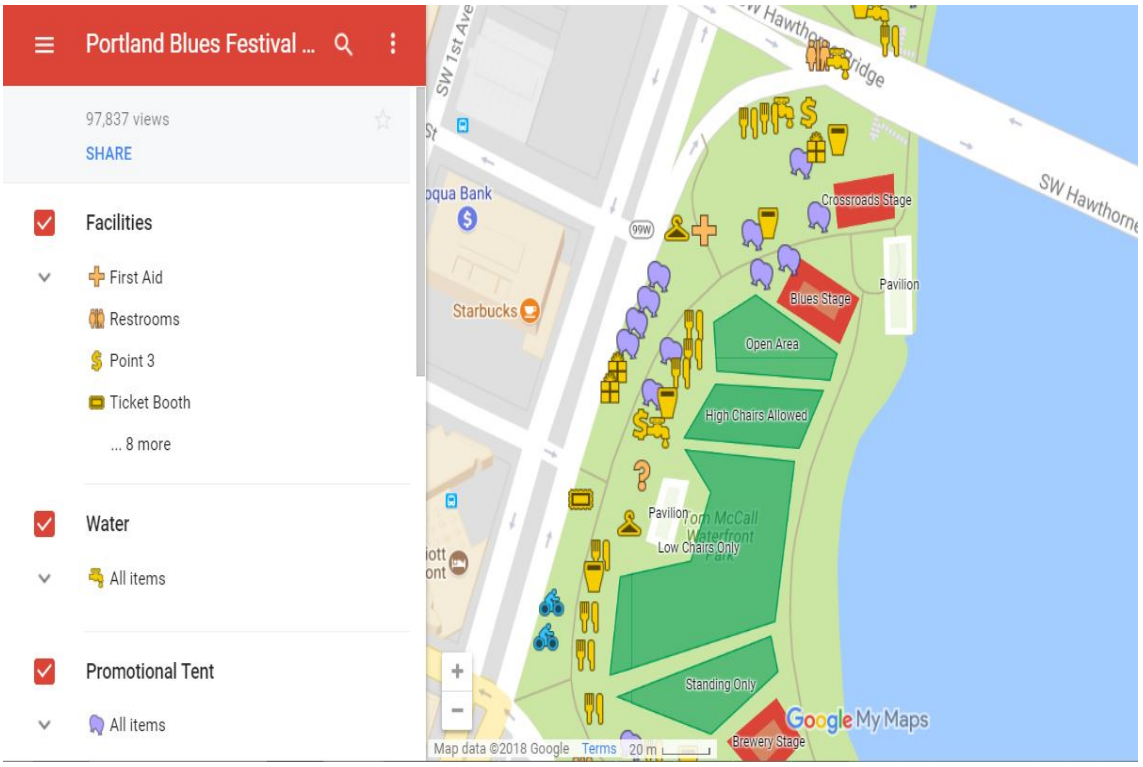

Figure 2.14: A festival seating plan

have used it for mapping geographic information of any kind, be it financial, socio-economic or environmental(Powney and Isaac, 2015), even a vacation tool for example creating tour plan around Australia's beaches using personalized kangaroo markers(ordnancesurvey.co.uk, 2018) (Google My Maps, 2018). Figure 2.14 shows a seating plan for a festival. Hence, a user of My Maps can put any kind of non spatial data on to a map giving it a geographic dimension. For example these marker png images had no value on their own but when on map they can provide a whole new meaning.

Plantin [2018] notes that google maps is a cartographic infrastructure created from participation and crowd sourcing to organize cartographic knowledge in society. Plantin further notes that google maps previously just provided means of public participation so everyone can take advantage of the spatial information, now it provides means of database handling and maintenance such as through tableau and fusion tables. This makes the google maps more participatory: such as providing means of public participation where people can report a closed business that still shows on the map and more enclosed where we can use organizational or personal spatial data for creating mashups on top of google map. For example Google My maps that can visualize any geo spatial information as shown in figure 2.14.

The most basic purpose it can serve is the ancient use of cartography, for example,

using maps, for crafting a trading route or during medieval war. With the excessive data at hand and that of potentially high socio-economic value, such as OGD, the ease of use with Google maps has taken this ancient use of layering maps with information, to a whole new level.

Google maps API gives us a method to automate the My Maps functionalities for e.g. it allows displaying large data sets as data layers, it also allows overriding default interactions and so on.

2.2.1 Creating Heat Maps With Google Maps Javascript API

Creating a heat map layer is not a very difficult task but also not as trivial as it seems. As pointed out by heat map layer documentation (Google Developer Console [2018]) larger datasets take exponentially longer time to load heat map layer. I had to reiterate my data preprocessing part of development several times just to minimize the time taken to display the map. This is the only problem that my application still faces, a work around should be expected in any future work.

In my opinion visualization, such as heat maps, act as a medium of data (re)presentation, while the google base map will act as a reference to the data representation and google maps API provides us with both.

The "heatmap layer" module in google maps is a seperate module and is not a part of the javascript API, rather it is a part of the visualization library and must be called seperately (Google Developer Console [2018]). This can be achieved by including a simple script tag in the javascript code to load maps:

```
1 <script type="text/javascript"
2   src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&
     libraries=visualization">
3 </script>
```

Listing 2.1: Calling google visualization library

As can be seen in Appendix: A.1 heat map layer is created separately out of an array of "weighted location" objects and then passed to the map object. Each "weighted location" object has a location and weight attached to it<lat,long,weight>. The weight in our case represents the IMD score e.g. health score, crime score etc.

The weighted locations can be useful when:

1. we have to emphasize the number of spots mapped at a single location, or;

2. we have to emphasize the "data based on arbitrary values" just like we have arbitrary IMD scores like crime score, health score etc.

Google provides options for customizing our heat map layer. These properties that I customized are:

1. **gradient**: It is a CSS color array which by default goes from green to red for rising intensity(of heat) for e.g. 0 to 100. We can modify the color gradient as it suits our data representation. Google maps, by default perceives higher number more intense. This can be appropriate if we are measuring magnitude, or population but not in our case.

   In our case the "cooler" regions represent better situations and "hotter" regions are more deprived. In our case "heat" refers to deprivation. Deprivation is more intense when an IMD score is lower. Hence, we have to modify the gradient. Rather simply reversing the default array is good enough.

   For e.g. a crime score of -3.5 is an alarming situation and historically red being sign for alarm, i had to reverse the default array.

```
["rgba(102, 255, 0, 0)",
"rgba(102, 255, 0, 1)",
"rgba(147, 255, 0, 1)",
"rgba(193, 255, 0, 1)",
"rgba(238, 255, 0, 1)",
"rgba(244, 227, 0, 1)",
"rgba(249, 198, 0, 1)",
"rgba(255, 170, 0, 1)",
"rgba(255, 113, 0, 1)",
"rgba(255, 57, 0, 1)",
"rgba(255, 0, 0, 1)"]
```

2. **maxIntensity**: This is the maximum intensity that a heat map can have at any single point this will put an upper bound so that the outliers can be ignored. By default, "heatmap colors are dynamically scaled" which means that any outliers with unreal high intensity will be ignored or averaged, nullifying or minimizing the outliers.

   One interesting thing to note here is that when a max intensity is explicitly set by us, the resulting map looks more like a hot spot map. This behavior is same if we don’t use weighted locations. The locations are taken as spots (lat,long) and the concentration of spots at a single location makes the colors of heat map.

   If we keep "maxIntensity" unset, the map acts like a heat map and it disappears on zooming in. This is a common feature in all mapping technologies including

ARC GIS and others. In my opinion this happens because the data point buffers are separated from their neighbors and the resulting color keeps fading away till its completely gone.

If we want to stop the above from happening we can set this to the maximum value of our IMD scores data set.

3. **radius**: The radius of influence for each data point, in pixels. In the sample there is an interaction provided to change radius. By default,i.e. if we don't explicitly set a radius, the radius is 10px.

   Each data point in our case is a weighted location <lat,long,weight> which represents each row of our dataset. So when a radius is changed it is a change in the data representation.

   Changing radius means changing the size of buffer around the data point. If it is higher it will influence the neighboring buffers too. But the accuracy largely remains the same because all data points increase their buffer size.

   This is useful when the heat map starts disappearing on zooming in.

4. **opacity**: The opacity range is 0-1. This changes the heat map layers opacity. This is a change in display brought by interacting with the system.

It is interesting to note that these options can be changed by interacting with the interface of mapping application. Google maps heat map layer sample includes javascript functions written already with a minimal functionality just to display the various possibilities(Google Developer Console [2018];Smith [2016];Jeremy [2002]).

2.2.2 Interaction with Google Maps

As mentioned earlier Google Maps offer sample interactions that are by default a part of the map for example zooming and panning, or, they are only related to a certain type of map for example heat map sample in JavaScript API comes with toggle gradient, toggle map, toggle/change radius sample functionalities with default behavior, which can be overridden or extended as we have done in our application.

Most of online map applications are exploratory because the information they represent is huge and we need to interact with the map to extract a specific selection of information we want to view.

In most online interactive map applications interactions are a combination of 4 types of interactions (Jeremy [2002]). These 4 types are:

2.2.2.1 Interaction with the data

The data for whole country can be huge as it is in our application. So, to divide the data into smaller chunks is the only basic solution. In the Figure 2.29 you can see a panel on top left. It contains buttons that are used to filter out other deprivation information, like by default the map loads with overall IMD score, then we can select health theme, selecting another chunk with same amount of information and so on. This can be a select query to the data base at client or server side.

2.2.2.2 Interaction with the data representation

It can be noted that information shown on the map can be viewed as a country wide information or a summary of regional trends. This information can be just enough for a user who is interested in for example north or south of England. On the other hand this is also inviting a user who wants to dig a bit deeper to focus on a more local trend. For this he may want to adjust the viewpoint zooming in/out and panning or by searching. This interactivity can also be seen as a navigational interactivity (Smith [2016]) as the user interacts with the map to reach a certain viewpoint or an area of interest.

2.2.2.3 Interaction with the Temporal Dimension

This category is related with dynamic maps for example the google directions app, or an analytical map where you can toggle a layer on and off. So, essentially this category is just about "displays that change continuously, either with or without user control" for example this map below is a Spatial-temporal analysis of breast cancer risk during 1983–1993 and it shows map frames of a map that is moving through time showing "changing patterns based on participants' historical residences". This also includes sorting the data or toggling data layers or manipulating map presentation for analytical benefits. So to analyze any hidden trends or to bring a regional trend to the forefront of the representation.

2.2.2.4 Contextualizing Interaction

In my opinion this is also serving the analytical purpose. This category of interaction is about analyzing by not limiting the application to a single map face, we can use the total interface of the application to create meaning for the map so it is more useful than it could have been on its own.

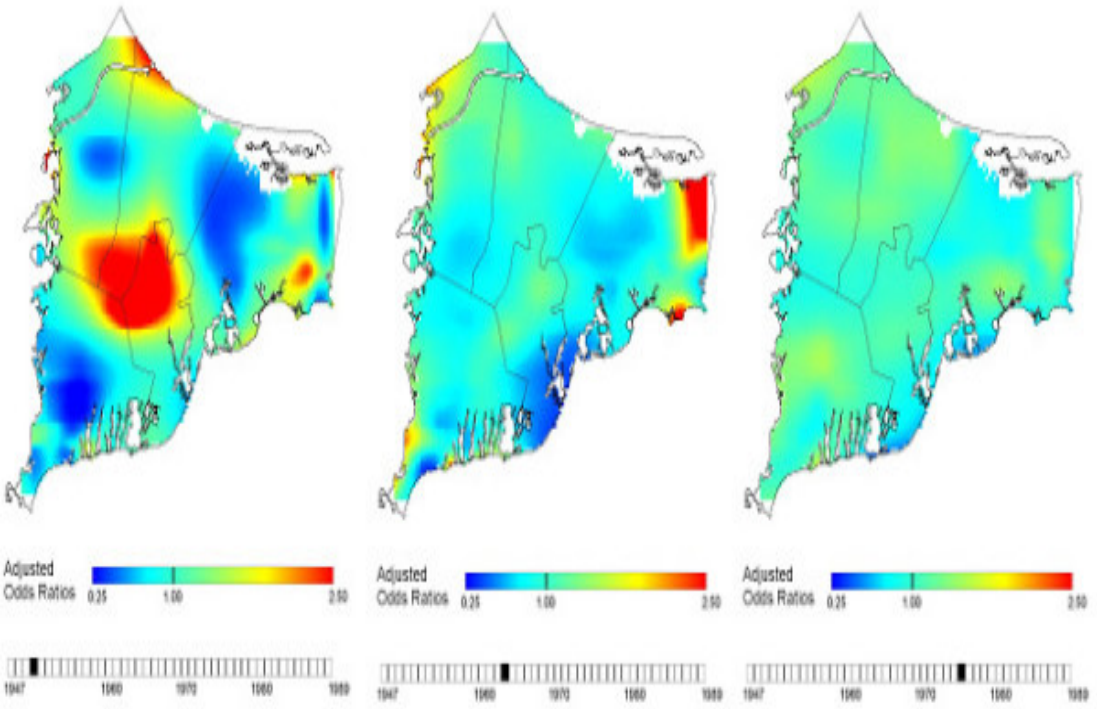

Figure 2.15: Geo Spatial Temporal Map Source: Vieira et al. [2008]

This includes map legends, using popups on map click to compare two neighboring regions, map tours or maps with stories, loading more than one map with different data layers, for example we could have allowed a user to compare for e.g. health and crime heat map layers. This calls for the map makers to not just focus on the map but taking an application as a whole aiming at proving a point for some value for the user and to be analytical (Smith [2016]).

Smith further suggests that static maps have been historically used for scientific research but the interactivity that is offered by contemporary open source mapping technologies, require a "two way relationship" between a user and the map. This advancement in the map making where a map is now exploratory rather than static has opened doors for advancement of spatial analysis at the cost of user interaction. This makes the map exploratory as there is a lot of layered information at hand, and a user can explore focusing on only a portion of total information.

We can conclude from above that, generally there are four types of interactivity supported by map interfaces. The applications I have mentioned to present the dimensions of interactivity although use a combination of interactions and so have their roots in the laws of human computer interactions(Roth et al. [2017]) and knowledge perception of a user(Lee et al. [1995]). A user is not just a map reader he must be treated like an application user, he can be anyone who is interested in knowing

more with an existing knowledge that came from his society or his own experiences. Through application we intend to facilitate a user, that is why interactivity is so important to build a map application which can have a map that visualizes one trend while rest of application facilitate his knowledge update through interactions. This idea of knowledge update lies at the core of data science where we are taught how to effectively create applications which can transform data into knowledge, which makes little to no sense to a target user on it's own. These interactions facilitate a user, asking questions from the application about data, updating his knowledge step by step (Roth [2015];Roth [2013]).

2.2.3 Reverse Geo coding

Geo Coding is to convert an address to its corresponding latitude and longitude pair. An example of geocoding is shown in Figure 2.16. It shows "Robin Smith Hall res 3" provided as parameters to geo code request, it then converts to a pinpoint location i.e. latitude longitude pair.

As the name implies, **reverse geo coding** is to convert a latitude longitude pair to an address (Google Developer Guide [2018a]). It is also known as address lookup. An example of reverse geocoding is shown in Figure 2.17.

Reverse geocoding, in the context of this project, provides a way to translate a map "click" interaction, which triggers a response to an interaction. It takes the <lat,long> of the clicked location and tries to translate it into an address. This results in a number of places as shown in the figure 2.18. Starting from the smallest local authority, e.g. a street address to the largest e.g. country.

The required parameters for making a reverse geocode request are:

1. **"latlng" (location)**: This a location object made up of latitude and longitude pair.
2. **key**: This is the API key. This key authenticates an application for purposes of quota management. This can be obtained from the developer console and a detail procedure can be found here: Google Developer Guide.

An example request url looks like this:

`https://maps.googleapis.com/maps/api/geocode/json?latlng=55.90966,-3.320349&key=YOUR_API_KEY`

"YOUR_API_KEY" will be replaced by the application key we manually obtain from the developer console.

The response to the above query is a JSON or XML multi dimensional array named "results", which contains multiple addresses. Each address in the results array is made up of smaller entities. Giving it a hierarchical structure. These addresses are sorted as smallest e.g. street address, to the largest e.g. Country. Each address in the results array is made up of 5 components:

2.2.3.1 Address Components

This is made of 3 things

1. long name: of the entity
2. short name: of the entity
3. types: These are many address component types discussed below 2.2.3.5

An address component array is a breakdown of a particular address, for example the response shows "premise" and its formatted address is "The City of Edinburgh Bypass, Currie EH14 4AS, UK". Apparently, this address is made of more than one type of address components as can be seen in Figure2.18.

2.2.3.2 Formatted address

This is a clean version of the combination of address components of a place.

2.2.3.3 Geometry:

This stores the geo location information of a particular address in context of the map. For example:

```
1
2 "bounds" : {
3                "northeast" : {
4                   "lat" : 55.909982,
5                   "lng" : -3.3185034
6                },
7                "southwest" : {
```

```
                  "lat" : 55.907716,
                  "lng" : -3.3227048
               }
            },
            "location" : {
               "lat" : 55.90972379999999,
               "lng" : -3.3203389
            },
            "location_type" : "ROOFTOP",
            "viewport" : {
               "northeast" : {
                  "lat" : 55.9101979802915,
                  "lng" : -3.3185034
               },
               "southwest" : {
                  "lat" : 55.9075000197085,
                  "lng" : -3.3227048
               }
            }
```


2.2.3.4 Place ID:

This is a unique string literal to identify a unique place on the google map.

2.2.3.5 Types:

There are many address component types but only some of them are related to our application. These are:

1. administrative_area_level_1: indicates a first-order civil entity below the country level. Within the United Kingdom, these administrative levels are states like England, Wales, Scotland and Northern Ireland, for example administrative_level_1 value for "3 Robin smith hall"'s is Scotland, UK.
2. administrative_area_level_2: "indicates a second-order civil entity below the country level". Within the UK this is equal to city level information.
3. administrative_area_level_3: This type is equal to the local district authority in the UK and so the IMD data set.

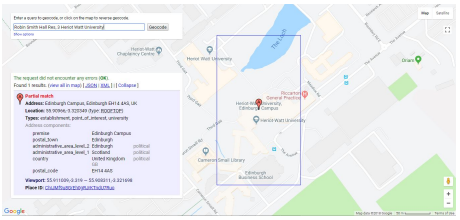

Figure 2.16: Geo Code

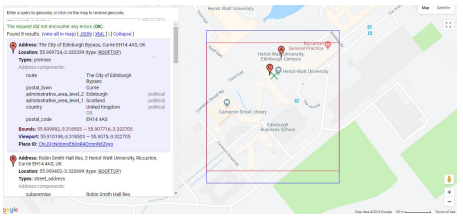

Figure 2.17: Reverse Geo Code

4. postal_town: This is the town information found in a postal address.

It is worth noting that the list of result types given above is mostly applicable to England, UK. Not all nations exhibit these administrative levels. Not all countries have the same division because all countries have different address representation. Not all reverse geocoding response contains all of these types of addresses. But the hierarchy remains. For example when I reverse geocode a region around Hounslow on the map(by clicking it) the administrative_area_level_3 value is "Hounslow" while if I reverse geocode a place near "Brighton and Hove" its administrative_level_3 does not exist (null values do not appear in results) and the "administrative area level 2" is the local authority district name(in IMD data set) and is identified as "Brighton and Hove" same is the reason why there was no administrative area level 3 in 2.2.3.5.

In our application we reverse geocode a place on map click and extract district level information only, to match it with our IMD data set(’s local authority district name) and show the information in an info window and load news section based on selected IMD and the extracted "postal_town" information. Some times this matched value is "administrative area level 3", in absence of it, the matching value is administrative area level 2 information. Mostly, it is found to be administrative area level 3 that matches district level information in England.

2.2.4 Places Search Box

"The SearchBox allows users to perform a text-based geographic search, such as ’police stations in City of London’. You can attach the SearchBox to a text field and, as text is entered, the service will return predictions in the form of a drop-down pick list" (Google Developer Guide Places Search Box [2018]). Search box is an extension of the places auto complete functionality. But the list that is shown in search box can be restricted by providing bounds or strict bounds in options parameter or area type for example administrative_level_1 or locality. We can also bias the results. For example there is a parameter "componentRestrictions" which can be biased to

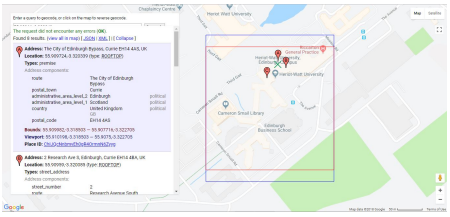

Figure 2.18: Reverse Geo Code Result: Premise

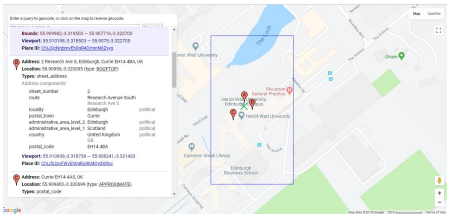

Figure 2.19: Reverse Geo Code Result: Street Address

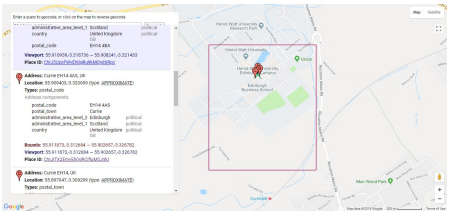

Figure 2.20: Reverse Geo Code Result: Postal Code

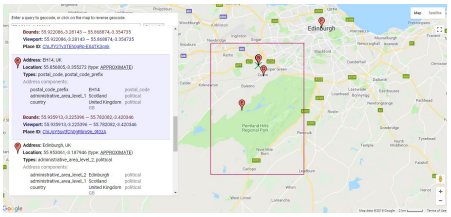

Figure 2.21: Reverse Geo Code Result: Postal Code Prefix

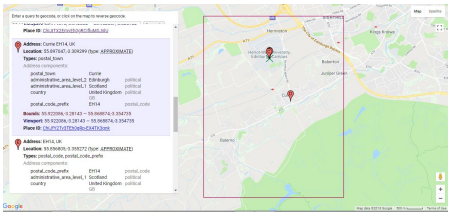

Figure 2.22: Reverse Geo Code Result: Postal Town

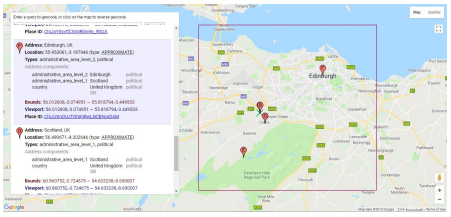

Figure 2.23: Reverse Geo Code Result: Administrative level 2

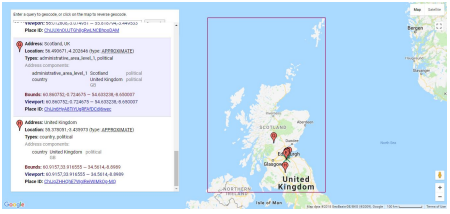

Figure 2.24: Reverse Geo Code Result: Administrative level 1

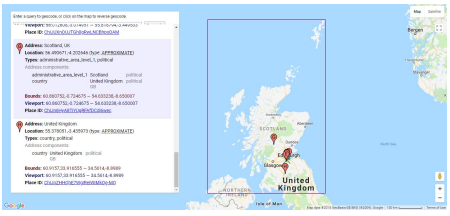

Figure 2.25: Reverse Geo Code Result: Country

Figure 2.26: Source: Google Developer Guide [2018a]

show only places in "UK". They will only appear first in the predictive list. It can take upto five countries as input.

This is another useful google web service that I used for the intention of providing interaction, primarily to navigate to a place of interest. According to (Google Developer Guide Places Search [2018]) "the Places API allows you to query for place information on a variety of categories, such as: establishments, prominent points of interest, geographic locations, and more. You can search for places either by proximity or a text string. A place search returns a list of places along with summary information about each place; additional information is available via a place details query". This has been used in the app in 2 ways:

1. To Search a place on map.
2. To Display establishments or buildings on top of map data layer. For example we can query "NHS near Hounslow" followed by "NHS near Richmond" to compare number of NHS facilities in each region.

2.3 Other mapping technologies

With the advancement of internet there have been a significant rise in mapping technologies and the web based services. These competitive spatial service providers have their own sources of spatial data for example reference databases and technologies. Most of these are open source and hence can be used by anyone with ease. (Roongpiboonsopit and Karimi [2010]) suggests that it is vital to understand how accurate they are because any error in geocoding is carried forward into rest of application. Apart from a few qualitative differences these online geocoding service providers work very similar to each other. For example all these providers have their own reference tables and have a similar functions. For example they just take an address as input and provide a few optional inputs and have similar outputs.

As the focus of our project is more towards the interactivity provided by the mapping tochnologies it is important to shift our focus towards the online/web mapping technologies.

When exploring many different applications using open source mapping platforms it is notable that these applications are a combination of traditional computer science and human computer interaction (Ballatore et al. [2011]). It is also worth noticing that the online mapping technologies of today like google maps and open street maps, provide all the necessary tools required to implement an interactive viewing.

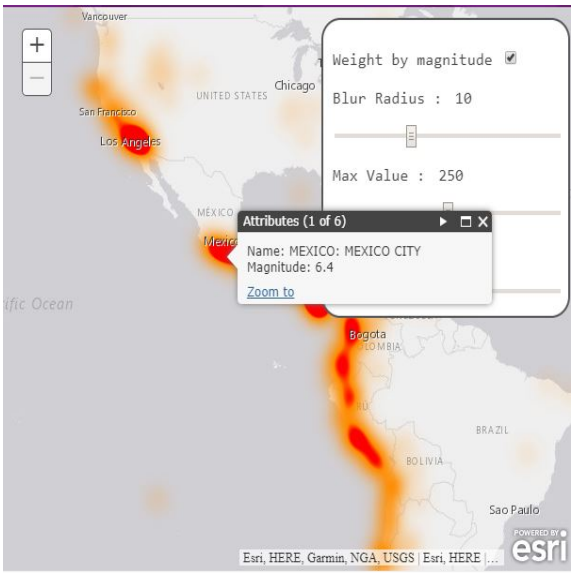

Figure 2.27: Heat Map made with Arc GIS Javascript API

There are many alternatives to google maps javascript API. There are a few factors on which we can decide which one to choose.

Firstly, the API's stability must be taken into account. Secondly it's API keeps on updating while previous functionalities are deprecated (Ballatore et al. [2011]). These alternatives must enable us to :

1. visualize customized data layers;
2. add new or modify default/existing, inter-activities as discussed in the interactivity section i.e. it must provide: zoom and panning, changing display, toggling layers, clicking on the map loading related information and searching (Fernandes et al. [2013]).

The criteria to select an API is:

1. API **size**: Google Maps Javascript API is the leanest of all its alternatives in the market. The size is made of objects, models and properties.
2. **Stability** with respect to added or deprecated functionalities in subsequent versions. This means that rate of objects models and properties added, modified or deleted must be low. We must understand that as GIS is an evolving field these APIs are always subject to change. Given this understanding and noting that fewer the objects models and properties in an API fewer the change. In addition to being subject to change, because GIS is an evolving science, google maps provide migration help to help a development team to migrate to the next version of API without clients getting disruption in services.

For example they give notifications if some service is going to be deprecated or changed in the future with help documentation on how to update to newer versions e.g. https://developers.google.com/admin-sdk/directory/v1/guides/migrate provides help to update your app which was using "Profiles API" currently, to "Directory API".

3. **Expressiveness** of the API: Expressiveness is a measure of effectiveness of the API within fewer calls to the API. Again because google maps api provides fewer objects, models and properties, at our disposal the expressiveness is more. On the contrary we cannot modify or override the default behavior for example the heat map layer functionality is taken care of by maps internal functions we can only provide inputs.

4. **Ease of understanding and implementation**: Google Maps, Arc GIS and Open Layer all have proper documentation and open help forums. And it is very easy to get help online. On the other hand more objects and models at hand can make the development a very complex task. Google is probably the simplest with only a fewer possibilities of changing the default behavior making it simpler to understand.

5. **Qualitative analysis**: Google Maps Api does not provide a way to manipulate the data layers. While Openlayers does not have geocoding while Arc GIS provides both.

6. **Pricing**: From what I have explored there is no professional mapping API that can replace google maps api and is free of cost. Each of these API's pricing is competitive with each other.

   In my opinion ARC GIS has much more potential to replace google maps than any other. It holds a middle ground, its not as bulky as openLayers but not as small as google maps. It's not that bad considering there are ways provided by Arc GIS to modify the data layer representation or change a default behavior while google does not allow this, because it considers it's default behavior as a "unique selling point" (Michael [2015];Fernandes et al. [2013]; Ayala et al. [2011]; CHIKKALA [2017];Ballatore et al. [2011]).

2.4 Data

To create a visualization as claimed in objectives in Chapter: 1 we have to understand the what data will eventually create a map visualization such as proposed in this project. The data requirement is to find a data that is

1. large: to enable exploration
2. geospatial:We need to visualize the data on a map interface so it has to be geo spatial.
3. quantifiable: Any visualization needs a certain value to analyze the data. The google's "weighted locations" need a "weight" numeric value to create a heat map.

The application aims to visualize a data that is socio economically related to a region. Coincidently, these kinds of data are usually served by open governments and are large and often multi dimensional just like IMD data set so they can only be analyzed by exploration.

This chapter first discusses open government data and particularly geosaptial open government and non government data in the light of its utilization by competitive mapping applications found on-line these days. It further details the data processing to be visualized on a map interface.

2.4.1 Open Data: Comparing Open Government, Open Data and Big Data

International [2015] is an accord that lays out certain standards for open data disposal. It suggest that just making data openly accessible is not enough and that it should have the following properties:

Data should be:

•**discoverable**, i.e. it should be search-able on the web for e.g. through google search engine one can easily find the IMD and NSPL datasets. Linked data

•**accessible**: The data should be machine accessible and readable. For example, IMD can be found at ukdataservice.ac.uk and data.gov.uk while NSPL is available at ukdataservice.ac.uk and national survey websites. In the day and age of linked data and web 2.0, the machine readability and understandability is much easier as this data gets updated from time to time, in which case the application won't remain valid. For example, NSPL dataset is update quarterly. It's not a productive approach to download large csv files manually and go through all the data cleaning transforming and extraction processes every few months. In this regard ukdataservice.ac.uk is a better option as it is built for the sole purpose of serving data that is machine readable data formats like SPSS, Stata and tab-delimited formats.

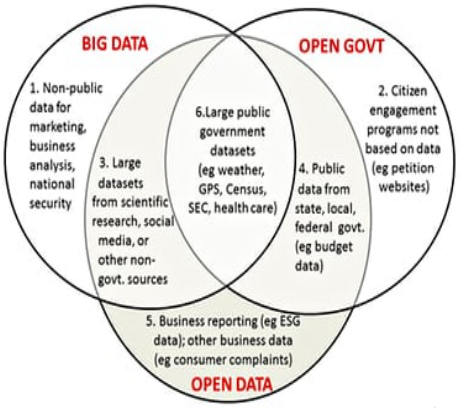

Figure 2.28: Venn Diagram differentiating between apparently similar types of open source data. Source Joel [2014]

•**intelligible**: The data that is uploaded by governments is backed by research papers for guidance about data at a very minute level which includes detailed information like how was the data collected what does each column and the scale of values used and these values mean. For example English IMD dataset can be found at Ministry of Housing, Communities and Local Government [2015]. This webpage displays a csv data file and tens of other documents for guidance on the data.

•**assessable**: The authority that uploads a data set must also allow a user to assess the origin of it, or if the data is useful enough or just be able to ask questions on the quality of data and its trustworthiness. So that in future if there is an application created on a fraudulent data then not only it will be useless but also misleading. But because a government has the resources to fund a data collection such as census level collection followed by mass storage in a fault tolerant system. Which is backed by ample documentation for making an assessment on the data.

The application uses 2 datasets NSPL(ONS Geography [2018]) and English IMD(data.gov.uk [2018]) under the open government license (The National Archives UK [2015]), these were obtained manually from data.gov.uk, as CSV files which were joined by a python script. Looking at the resulting size of data, which was extracted transformed and put into mongodb, one can easily relate with Big data. Open government data movement came into being with the advent of big data. The venn diagram differentiates between the 3 seemingly similar kinds of data: open data, open government data and big data 2.28. According to this diagram our application's data can be located in

- **3**: Large data sets from non government open data sources e.g. social media :

This is a combination of data that is big and open. This essentially means that it is huge in size and is free to view and use by anyone whoever wills to do so but it is not from government.

As our application uses non government data source such as newsapi.org which provides json data in response to http requests, which is not free of cost but does not discriminate between the user, which makes it open. And the response can be huge in size, for example, a query such as select all news items from 10 different news sources will result in 10s of thousands of news items. This makes it relate to the volume of big data. Also they update with new content every second, which is the idea of Velocity in Big Data.

The Rozhkovsky and Bilchenko [2018] uses social media (twitter) news feeds which is from the same group of data kind. It updates regularly so I presume the data at use is big data and because the map itself is open to share and they accredit the information on display to be from open source Big data such as twitter feeds.

- **6**: Large public Government data sets This section in the diagram represents the combination of Big, Open and government data. As mentioned earlier the data used for heat maps is a combination of datasets from open government.

OGD is not big data because it does not have:

Velocity: large number of data coming in every second, for e.g. streaming data being generated from a location enabled sensor, as such; for example the latest IMD data set was created in 2015 while NSPL is updated quarterly.

But it does have **Volume:** huge numbers of data such as crime incident data set has incidents listed from 2009 till last month.

and **Variety:** data coming from different sources.

There are very few chances of **Veracity:** because the data is collected by the government and there is a very strict protocol with humans involved in making sure there are no exceptions or errors uncaught. This is not the case with big data. This provides much ease with trust in data collection (Gandomi and Haider [2015];Gurin [2014]).

We can safely presume that we have to have a strong preplanned system for handling data. The usability is similar because as open data, big data is huge in numbers and has no meaning on its own.

2.4.2 Geospatial Open Government Data and Its Utilization

Utilizing OGD is a considerably new area of research and academics have questioned the disclosure of millions of datasets by governments and its lack of usage (Safarov et al., 2017). OGD is in its infancy, hence,it is mostly a focus of academia or a very few people interested, and only provides motivation and direction for further research and actual applications that can make the desired impact in a number of ways.

Safarov et al. further notes that there are 4 dimensions of utilization. First is "utilization" itself which I will discuss further in this section, second is "conditions" for example what is disclosed as open data by the government, the political barriers that hamper the openness and meaningfulness of the data. Third dimension is "Effects" which are ”the potential results and outcomes of OGD utilization from regional and social, economic or good governance perspectives"(Safarov et al., 2017). Fourth dimension is "users" for e.g. a citizen or a government employee. Each of these dimensions lay foundation for different context and motivations for the application.

In the context of my project the most desired effect of a utilization as such, is to enhance ease of exploration of ’boring and difficult to understand’ datasets for novice general public (Fung, 2013). This makes it a social value effect (Matheus et al., 2015) and if the user is a public official and reports it to higher authority or starts working to make a change, then it will be a public services development effect, if the citizen user sees the change and that develops his trust in his government, a citizen trust can be accomplished (AlAnazi and Chatfield, 2012).

According to (Graves and Hendler, 2013) the most basic tool that can utilize OGD to potential is visualization and as our focus is spatial analysis, we can note that map visualizations are the best way to use it. The closest competitive example of geospatial OGD visualization is the English IMD exploratory map. It’s a choropleth map such as shown in the figure 2.29. This application lets a user search a place or zoom and pan to it and view its corresponding deprivation information allowing to explore the huge data each time selecting a part of information to map. This is the closest example because my application’s heat map functionality uses the same IMD data set with the exception of joining it to post code point locations giving access to the street information. Another difference is the mapping technology used. My application uses google maps API while it uses leaflet and open street map. Similarity is the interaction as described earlier, i.e. searching a place, zooming in and out, panning and clicking to load more information about the region or switching a region.

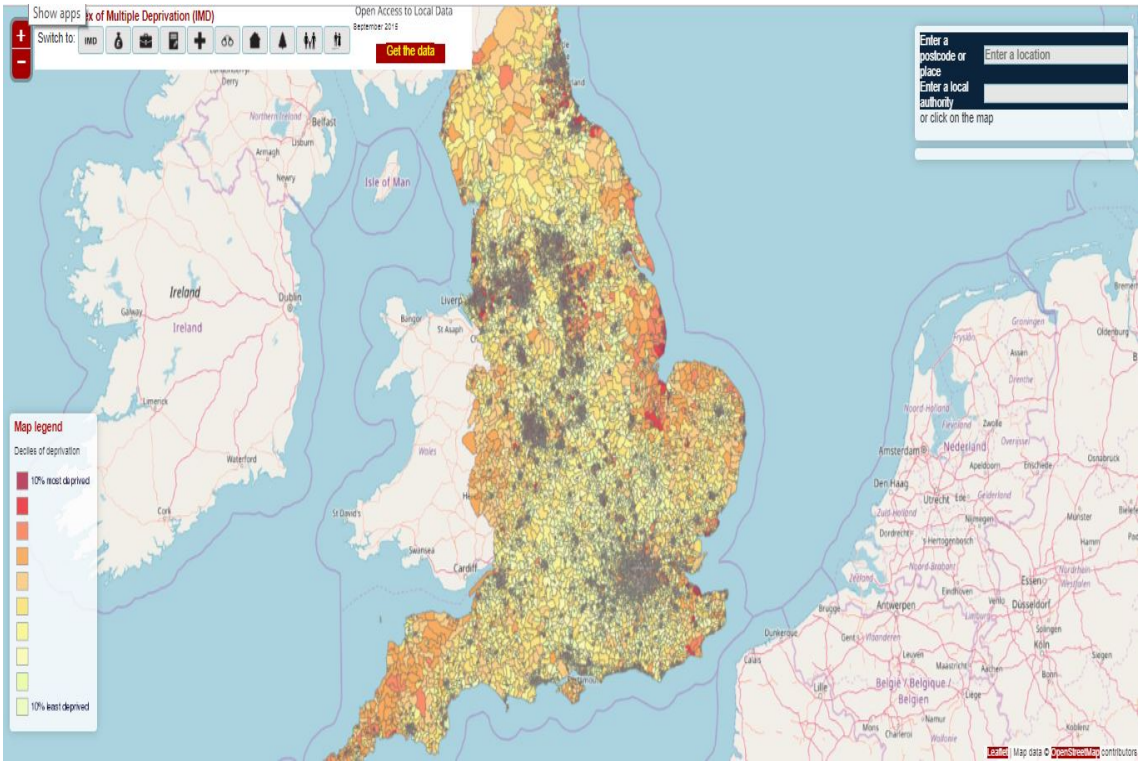

Figure 2.29: English IMD map

Another difference can be seen here, my application loads district level information inside google infowindow popups,such as district mean crime decile, which is averaged out, for simplicity, while IMD map application can show lsoa information.

(The New Inquiry, 2018) is an example of geospatial visualization of suspected areas for white collar crime if the user is a public official from say legislation or taxation department he can certainly put this visualization to use and keep his focus on finding possible suspects, marking areas where they can be found. This system uses US government crime incident data regarding white collar crime and in its introductory hypothesis it states that the police department is more focused on street crime while not putting much needed effort into the other category.

The spatial analysis of who exactly voted for Brexit as mentioned above(Figure 2.2) could not have been possible for me as an outsider, without open data. We compared state powered open government data with open source data to prove a hypothesis and strengthening my mental maps.

(Parasie and Dagiral, 2013) notes that the pioneers of utilizing open government data for mapping, are the newsrooms, using the open availability of data and mapping technologies, to further their regional or local political agendas. Giving this power to citizens themselves there is even more chance of social value being extracted from geospatial OGD map visualizations.

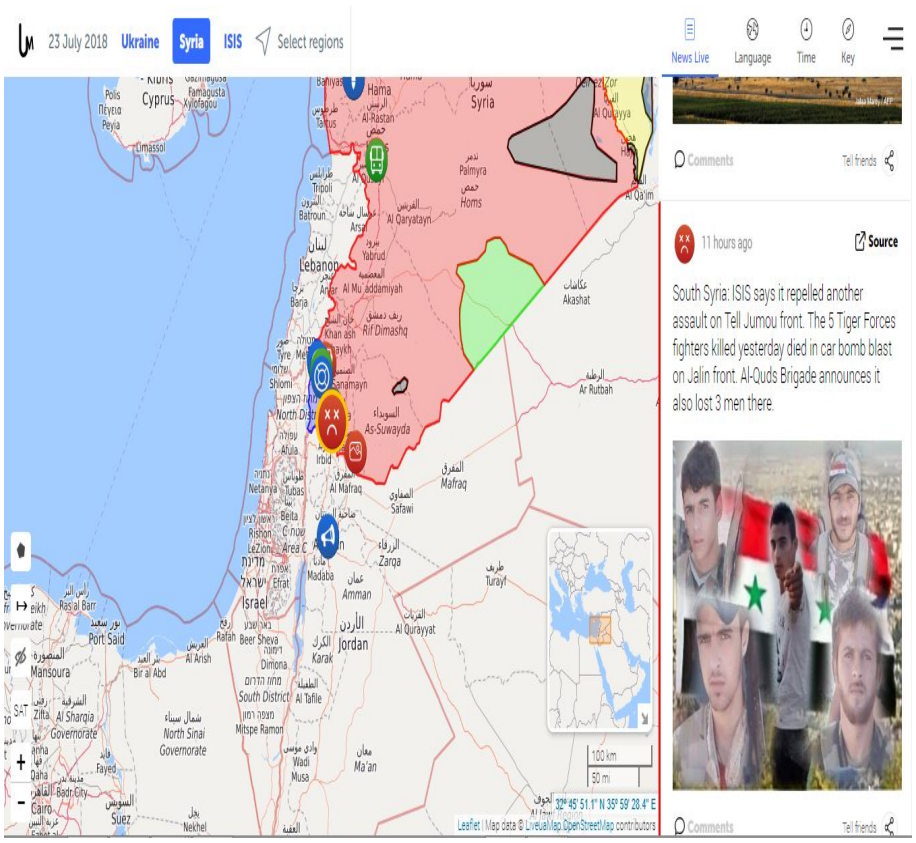

Figure 2.30: Live UA interactive map showing twitter news mapped on Syrian map

2.4.3 Geospatial non Government Open Data and Its Utilization

One other (non government) open data source is the newsapi.org (newsapi.org [2018]) loaded in response to a HTTP request. Although this functionality was added to display the click interactivity provided by Google maps JavaScript api, however many other online mapping applications display news events on the face of the map with Google cluster markers. For example liveua.com (Rozhkovsky and Bilchenko [2018]) displays Syrian conflict news events mentioning a spatial entity such as "Damascus" which is then reverse geocoded and shown at the location as a marker, a brief description then pops up on clicking the marker icon. An example can be seen below.

As can be seen this is again a mapping application for social cause. This war is a human catastrophe and universal awareness is just one of many themes of map making, made possible by data that was openly available this time because it is social media(twitter), which makes it big data, but the relatability with open data is evident.

2.4.4 Data Preprocessing and Management

2.4.4.1 Data Preprocessing

To create the visualization as ours data from different raw datasets in csv format had to be combined and preprocessed. The preprocessing is defined by the final data requirements. The data which will populate the heat map layer. We require 6 numeric columns(lat,long, crime score, health score, Employment score, education score) in the final data set, of which only 2 location columns(lat,long) and one IMD score column are selected each time a particular theme of IMD is selected and corresponding heat map layer loaded. The required columns for heat map layer are for England only. Our data preprocessing is directed towards this requirement. Data preprocessing is a well known concept for anyone who wants to analyze the data. The first step is to understand our data, as detailed in previous sections. Continuing from the data we understand and the data required for visualizations we now have to process it in a way that it matches this required data.

To achieve this we have to look into the guidance we have on this subject. After going through various research papers(Castanedo [2015];Han et al. [2011];Diehl [2004]) I have noticed that they all go through the same processes as listed below:

1. **Data Cleaning:** Both (IMD and NSPL) datasets are consistent and data collection is standardized and well maintained that is why the problems of missing values and noisy data does not appear. If it was big data sets from social media or survey forms then we might have needed proper automated data cleaning.

2. **Data Integration:** It is already clear that we need a single data set to create the visualization. This is achieved by combining IMD dataset, which provides value, and NSPL dataset, which provides geo spatial information. This process of fusing two datasets into a single entity is known as data integration(Ziegler and Dittrich [2007]).

3. **Data transformation:** Data visualization is an analytic application so it requires the data in a particular state before it can be analyzed(Castanedo [2015];Diehl [2004];Manikandan [2010]). Manikandan further states that transformation can be done simply for ease of comparison. In our case it is the minimization of time by letting go some accuracy such as rounding off <lat,long> to 4 decimal places.

4. **Data reduction:** The data has to be reduced because analytics with data mining or visualization takes time with rising number of columns. Looking

at the final shape of data we had to choose columns by *forward selection*:I knew lat,long pair and IMD values will be required and sometimes we reduced columns through *backward elimination*: The initial plan was to populate 8 heat maps(each map for each IMD) but due to time restriction I reduced the score columns to 4.

2.4.4.2 Data Management

Once the data has been preprocessed the same dataset is then transferred into mongodb, a nosql document store database management system. This choice is made on a few considerations:

1. The application has to deal with **large** amounts of data.
2. The application is a mere visualization, which is a **read intensive task**, no data is ever written into db from application. It's only written once in lifetime in the scope of my project i.e. during data pre processing where it is written to mongodb server.
3. The application uses web services which respond with (sometimes huge) data written in JSON format, which is just like a document store.
4. It is important to minimize data manipulation at run time, this calls for having the data preprocessed, so that we don't have to go through the data transformation at run time.

Keeping in view, the issues presented above, in my opinion mongo db is the best option for a database management system.

1. **Scalability**: NoSql provides horizontal scalability because of it's flexible schema or schema free structure. Mongodb can be seen as having a semi structured schema. For example it can store data types. RDBMS like MySql is appropriate where there is normalized data with several tables connected with each other through foreign keys. In our case we have 2 data sets that took 2 mins to be joined in python if we do it on our server at run time this will be a huge problem. Big data has a lot of inconsistencies and it will be increasingly difficult with SQL and exponentially difficult with larger datasets. The problem is that SQL follows a strict schema while Big data can be inconsistent. Which means that a schema less dbms or flexible schema dbms will be a better option. For e.g. a data set is 2 million rows long with many missing columns.

With nosql these columns will not exist while mysql will have to save null values because of it's fixed schema. While importing data into fixed schemas we can face lots of problems or it will have data integrity problems.

2. **Performance**: Big or open data can be a single table with many property columns. It's most commonly found as denormalized. With RDBMS we will have to follow a strict procedure to preprocess this data before inserting in the data base management system. If the schema is strict the data transfer process will break at every inconsistency. This is a huge blow to the performance.

   Even if we ignore write problems mongodb is very modern in a way that it provides API functions to query the data while RDBMS requires complex queries to read data. With mongodb we can run aggregate queries on large datasets which usually respond within seconds (Hadjigeorgiou et al. [2013]).

3. **CAP Theorem**: CAP theorem suggests that in case of a fault in a network(**P**) of database partitions we can trade off consistency for availability and vice versa. Different DBMS's choose one over the other. A system that choses consistency will reject any read or write to keep every read consistent with the most recent update.

   (a) **C**onsistency: In our case, there is nothing getting written into db from application in response to user interactions, therefore write consistency is not a concern for us. In the future if there is a functional requirement where data is written in large numbers and the results will get affected with inconsistent data then this will be a concern.

   (b) **A**vailability: On the other hand availability is required, because in case of nonavailability of data the map in application will not load. We can safely assume that the data is ever consistent, because rate of change of data is nil. So when choosing a dbms, availability is our main concern, data must always be available. Hence we will choose availability over consistency in case of a faulty partition. The systems that chooses availability also means eventual consistency once the fault is taken care of (Gilbert and Lynch [2012]).

We can conclude that we need a DBMS that can guarantee high availability and scalability. In addition, it is worth mentioning that data mining and visualization efficiency depends on the type of DBMS, and the processing power available for the application (Alazmi and Alazmi [2012]).

Chapter 3

Requirement Analysis

3.1 Project Scope

As set out in the objectives of this project, the application is an online, exploratory and interactive web mapping application. According to my exploration of existing works there are certain pre defined or most commonly followed activities to create such an application. So, our project inherits all those activities in our project scope.

The project scope includes:

1. **Preprocessing data** which will output a combination of 2 large open government datasets. Preprocessing includes data extraction, data reduction, data transformation and uploading data on to Mongodb. Please learn more about this in Chapter: 5.

2. **Interactive Visualization with Google Maps API** The visualization is representing a large multi perspective data set that needs to be explored. This exploration will come at the expense of interaction.

   To make it interactive Google Map's Javascript API is used. The default functionalities provided by Google's sample codes are modified. So they have to be overridden in support of our application.

To give more meaning to the map visualization, firstly, rest of application has to play a supportive part. So, the project scope includes designing an interface such that the application interface changes the visualization on the map interface and an interaction on google map interface changes the application.

This project will allow a user to search a place or browse the map, given heat map loaded for a specific socio economic indicator such as health or crime etc. In a way that user can understand the living conditions in or around his city or location of interest.

To restrict the aim for higher precision I have restricted the depth of information by reducing some data points. But that is still a lot of data and at run time the google maps javscript API takes almost a minute to load. Technically reasonable effort has to be put in to reduce data selecting only lat long pair and one score value such as health score, e.g. tuple <lat,long,healthScore> removing unwanted rows and columns.

The project will be deemed complete if the heat map made of data mashup is created using open data sources and Google Maps visualizations.

The deadline is 16th August, 2018. All work detailed in later sections will be submitted at this point.

3.2 Project Objectives

The project objective is to create online, interactive map visualization application using large and open data sources and google maps javascript API by 16th August, 2018.

3.3 Project Deliverables

Following are the project deliverables:

- A dissertation
- A poster session
- Data Preprocessing can be found on Google Drive: https://drive.google.com/open?id=1UpZYyLr9zgasqX8nKOet86Tg6dK8wV-M It includes:
  1. National Survey Post Code Lookup data file named “NSPL_FEB_2018.csv”
  2. English Indices of Multiple Deprivation data file “imd.csv”
  3. dataPreprocessing.py (to test this file, files 1, 2 and 3 must be in the same folder)

Application (maps) can be found at: https://github.com/abdurrehmanshahid315/maps.git

1. maps.zip containing application code For Instructions to run the software please have a look at the read me file.

A tight schedule is followed to make sure these are delivered in time.

3.4 Project Risk Assessment

The risks in my project are really minimal but in some cases can pose a threat to the project time-line, for example during producing application code there was always a chance to get stuck at some technicality that I haven't gone through in the past. In this case there exist a need for research and development which takes more time than just development. For e.g. I started with putting data on client side for the first time, which caused problems, I had to look into the data ever more closely and I found out that all string data had to be converted to float at run time which was a bad idea so I wrote another python script to convert it to numeric. I reached this point at every iteration of this project I always felt at each of these points in time that I have to change the data reduce it as much as I could. So I came to this point that only 3 columns (lat,long,imd score value) all float would be prepared in server and sent to client so that browser does the minimal job.

Even after making browser handle the minimum data, i.e. only "weighted locations" (3 columns) for heat map, the google map took 48 seconds to load which can get extended with a slower processor and slower internet connection. This can make a user uninterested in my application. It took much research and development just to reduce the load time.

This still is a project risk, and I concluded that I have to give up precision to get better loading time.

In a nutshell time management is very crucial. Although I had prior experience with google Maps, I had very little experience of working with this much huge data. I did not write huge amounts of code in the application, on the contrary most of the work is done in the preprocessing of data. At points during development I had to go back to reduce or shape up the data a bit more.

Therefore, time management is a risk that can be mitigated early on through proper planning. To mitigate this or to be able to recover at any point I started development

Table 3.1: Risk and Mitigation

Risk Situation	Impact	Mitigation
Getting stuck on technical Issues	High	Started development early
Time required to research and develop	High	The incremental approach worked to mitigate this issue
My Illness	High	Follow University mitigation recommendations
Mr. Gabbay's unavailibility	Medium	Alternate ways to make a contact for example by phone
Value is not evident	High	Evaluation will be conducted by me and supervisor at every iteration to make sure visualizations are valuable for the user
Data is lost	Medium	Due to data being maintained by their respective owners as Open Data its extremely rare
Latency Issues	High	The bulk datasets will be cleaned and loaded in the server to manage excessive load at data selection
Network and processor speed restrictions at run time	High	I have tested location<lat,long> close to 4, 3 and 2 decimal places. This will effect the depth of information and will decrease the accuracy as discussed earlier. 4 decimal places data was finally chosen

as early as I could. I also learned from past research report feedback that I had to be quick and not worry about the form of application too much, this was excessively hard at times, because when I wanted to take another look if every interaction with the map works, I many times found out that district names in google maps and IMD dataset do not match which I later corrected but at that moment seemed to be a big problem.

Higher the impact of a situation more rigorous would be the mitigation strategy. Table 3.1 sums up the risks, their possible impact (high,Medium,low) and their mitigation strategy.

3.5 Requirements

This part of the report is in conjunction with the RFC software engineering best practices (Bradner, S., 1997). The data mashup web application is made out of two separate parts:

3.5.1 Heat Map Visualizations

The application must let a user explore IMD(Ministry of Housing, Communities and Local Government, 2015) dataset by switching between following themes:

- **Health:** This will use the aggregated data to visualize health score by neighborhood(LSOA) with heat map layer over google map's basemap.
- **Education:** This will use the aggregated data to visualize education score by neighborhood(LSOA) with heat map layer over google map's basemap.
- **Employment:** This will use the aggregated data to visualize employment score by neighborhood(LSOA) with heat map layer over google map's basemap.
- **Crime:** This will use the aggregated data to visualize employment score by neighborhood(LSOA) with heat map layer over google map's basemap.

3.5.2 News

This part can be related to interactivity and open data. The idea here is to display response of interaction on the application rather than using application interface to change map, now map interface is interacted with and the application other than map interface receives the response. So, the application "MUST" allow a user to choose a location on the map given an heat map(crime, health, employment, education) is already selected and the heat map is already loaded and the news related to the chosen area will be listed in a physical section below the map.

Chapter 4

Professional, Legal and Ethical Issues

4.1 Professional

All coding practices for this dissertation follow British Computer Society standards. I have followed the project plan as much as I could. All third-party libraries and code reused are properly referenced. All reused material, for example, from stackoverflow.com help, if any, is properly attributed.

4.2 Legal

All the bulk datasets, API data sources and API web services are properly attributed according to the terms and conditions of their respective owners. Open Government Data that is used in this web application is made available for public use under the Open Government License (The National Archives UK, 2015) and is available in open formats like csv, JSON etc, at state sponsored websites. The Open Government License clearly states that anyone can:

- copy, publish, distribute and transmit the Information;
- adapt the Information;
- exploit the Information commercially and non-commercially for example, by combining it with other Information, or by including it in your own product or application.

-*The National Archives UK, 2015*

4.3 Ethical

These days the society has become very complex. The objective of our application, as mentioned earlier, is visualizing some very core societal values. Such as living conditions. Today's complex society reacts in a very complex way to anything targeting the socio economics of a society, for e.g. reaction to News, where its easy to deny news, as most than often they are biased views of individuals.

Still, some might be impressed by our visualization and would want to explore more to benefit from the application to the best of their interest and capacity e.g. they might use the information provided to assist their search for better place to live or to confirm the claims of their politicians. They might filter by searching a place for e.g. a street address, area name or post code or by zooming to closer proximity of a region city or a street.

The objective of this app is to enhance the socio economic knowledge of a user about his community. Their is no political motive what so ever. The information a user sees on the map, is only a translation of data that was provided by state institutions not a single column's value or scale is changed.

Having said that, as mentioned earlier the rounding off of <lat,long> can have an adverse affect on the accuracy but it is too small to have an impact on the wider picture. To mitigate this problem I have restricted the zoom so any inaccuracy can't be reached by zooming in.

We are in middle of a revolution when it comes to thematic mapping using open government data and open source mapping technologies. This will drive more and more people to join in and give their inputs. In that case our application is a perfect example for future. In the future this information will not just be accessible to men in suits in closed rooms but it will also be transparent to the ordinary people which is the first and basic right in democracy.

So, ethically my application disseminates information that is not secretive but very crucial to a society. The app's objective is to strike some chords to draw attention from every corner using crucial information legally available to view and use in any form.

Chapter 5

Implementation

This section details the implementation and the creation of the web application that is mentioned in project requirements. This section mostly answers the "How" part of the project. This section also maintains it's relevance with literature review in Chapter 2.

The task at hand is to display mashup of valuable information on google maps interface. Few things are immediately confirmed here:

1. **Availability** of **valuable** information
2. information has to be a **combination** of disparate datasets.
3. The information has to be **geo spatial**
4. The mapping API has to be **google maps**

This means that a combination of large datasets is required, one of the dataset provides value i.e. IMD dataset and the other provides large geo spatial information i.e. NSPL dataset.

In my opinion even before finding datasets we have to understand what data will eventually create a heat map. Hence, understanding data requirement, is a prerequisite for the next steps i.e. data acquisition and data preprocessing.

DataSet	Number of Rows	Number of columns	Kept no. of rows	Kept no. of Columns
NSPL	2608956	39	1452285	3
IMD	32844	11	32844	5
Merged	1452285	6	1452285	6

Table 5.1: Datasets Summary

5.1 Data Requirement

The main requirement of this application is to create google heatmaps. The map needs an array of "weighted locations" to create the heat map layer. The weighted location is made of 3 properties

1. latitude - Data Type: Numeric
2. longitude - Data Type: Numeric
3. Weight - Data Type: Numeric

Latitude and longitude together make one location. So they are to be taken as a single composite index. This is why google combines them in a single object "LatLng". I'll discuss this later in client side code section.

This leads to our first step of data preprocessing i.e. data acquisition.

5.2 Data Acquisition

The data was acquired in csv files from UK open government sources(data.gov.uk [2018];ONS Geography [2018]) keeping in mind that final data must be quantifiable, geospatial, should be from disparate sources but integrable. Being integrable means that they should have some common properties on which we can base our integration process, in this case it is the "lsoa11" column found in both the datasets.

For this project 2 datasets that can be integrated were **discovered** online and **accessed** online from UK government website `data.gov.uk`. They fulfilled a reasonable degree of standards and conditions of falling under open government data category.

The 2 datasets are summarized in the table 5.1

5.3 Data Preprocessing

For this part a python script, which can be found online here AbdurRehman [2018], was written which performs the following tasks.

1. **Include 3rd party libraries** Data preprocessing was done by the most commonly used "python pandas" library. It can read raw data and perform every data manipulation process within less than a minute. There is a plethora of online tutorials and help forums so it is very easy to understand, effective and efficient for manipulating huge data.

   The "pymongo" library is used to make a connection to mongo db, read from, write to and update collections inside a database.

   As mongo db is a document store it is necessary that we convert data into Binary JSON. To achieve this "json" library is used.

```
1 import pandas as pd
2 import pymongo
3 import json
```

Listing 5.1: Importing 3rd party libraries

2. **Perform data extraction**: Data is read from csv files by "python panda" library's read_csv() function. This is achieved by the following part of script:

```
1 #read csv raw data
2 dfNspl=pd.read_csv("NSPL_FEB_2018_UK.csv")
3 dfImd=pd.read_csv("imd.csv")
```

Listing 5.2: Extract data

   The csv files are read into "data frames". Data frame is a data structure to hold 2 dimensional data which can be of any type. It automatically reads data types from raw data. At this point we have both the datasets i.e. IMD in data frame named "dfImd" and the NSPL dataset now resides in "dfNspl" with their data types intact just as google's "weighted location" requires.

3. **Perform data reduction**: After reading data above in listing:5.2 it is now time to get rid of unnecessary data. First of all we will keep the "live" post codes only, in other words the post codes that are current and not terminated i.e. where "doterm" column is empty.

   Secondly we get rid of rows where country denoted by "ctry" is England recognized by country code "E92000001" in the NSPL dataset.

Thirdly we get rid of any unwanted columns. We just mention the columns we need as shown on lines 3-8 for IMD dataset and line 10 for NSPL which is materialized on lines 11-12 in listing:5.3.

```
1  dfNspl=dfNspl.loc[dfNspl['doterm'].isnull()] #livepostcodes only
2  dfNspl=dfNspl.loc[dfNspl['ctry']=='E92000001']#filtering out all
       other countries in UK except England
3  columns_Imd=[
4          "lsoa11",
5          "EmploymentScore",
6          "EducationSkillsandTrainingScore",
7          "HealthDeprivationandDisabilityScore",
8          "CrimeScore"]
9
10 columns_Nspl=["lsoa11","lat","long"]
11 dfNspl=dfNspl[columns_Nspl]
12 dfImd=dfImd[columns_Imd]
```

Listing 5.3: Data Reduction

4. **Perform data mashup**: As already mentioned for combination of two disparate datasets we have to find a common column. This is "lsoa11", the local authority code. This column is unique in IMD dataset but not in NSPL data set. NSPL dataset has unique post codes and each post code has a unique <lat,long> location. We can notice that lsoa11 is a larger boundary and it is made up of multiple <lat,long> locations which makes it a one to many relationship. So we took outer join of IMD and NSPL on "lsoa11" column.

```
1  df_merge=pd.merge(dfImd, dfNspl, on='lsoa11', how='outer') #left
       outer join, to ignore any areas that are not included in IMD
       dataset
```

Listing 5.4: Data Mashup

The resulting number of rows are 1452285, same as the final NSPL data frame.

5. **Perform data transformation**: In our effort to make the data as lean as possible I performed two things:

   - changed the column names to shorter names for example "HealthDeprivationandDisabilityScore" was shortened to "HealthScore".
   - Second is a naive effort to spatially aggregate the data i.e. we round off <lat,long> because the number of decimal places represent the accuracy

of the location points. For example a 4th decimal place is worth up to 11 m (whuber- gis.stackexchange.com [2017]) and recording any more accuracy is futile. So, we round off the original 9 decimal place <lat,long> to 4 decimal places. This way many <lat,long> will be pointing to the same location on the map. Taking <latitude,longitude> as a composite index we group those <lat,long> pointing at the same location and take an average of the other value columns as seen below in listing: 5.5.

```
1  pd.options.display.float_format = '{:.4f}'.format
2
3  rounded=df_merge.round(4)
4
5  grouped_rounded = rounded.groupby(['lat','long']).mean()
6
7  grouped_rounded = grouped_rounded.reset_index()
8
9  final_columns=["lat","long",
10                "EmploymentScore",
11                "EducationSkillsandTrainingScore",
12                "HealthDeprivationandDisabilityScore",
13                "CrimeScore"]
14
15 #rename columns
16
17 grouped_rounded.rename(columns = {'
       EducationSkillsandTrainingScore':'EducationScore'
18                                     ,'
       HealthDeprivationandDisabilityScore':'HealthScore'}, inplace =
        True)
```

Listing 5.5: Data Transformation

6. **Data Entry into Mongo DB**:This is a straight forward task. A connection is made to the mongo db active at localhost:27017. The final transformed data frame "grouped_rounded" is converted to json on line:5 and written to the select collection on line:8 in listing 5.6.

```
1  mng_client = pymongo.MongoClient('localhost', 27017);
2  mng_db = mng_client['myNewdb2'] ;
3  collection_name = 'shortAndSelected4'
4
```

```
data_json = json.loads(grouped_rounded.to_json(orient='records'))
db_cm = mng_db[collection_name]
db_cm.remove()
db_cm.insert(data_json)

#imd District

dfImdDistrict=pd.read_csv("imd.csv")

#group
grouped = dfImdDistrict.groupby(['LocalAuthorityDistrictname'],
    as_index=False).mean()

grouped = grouped.reset_index()
# write to mongo
data_json = json.loads(grouped.to_json(orient='records'))
mng_client = pymongo.MongoClient('localhost', 27017);
mng_db = mng_client['myNewdb2'] ;
collection_name = 'imd2'
db_cm = mng_db[collection_name]
db_cm.remove()
db_cm.insert(data_json)
```

Listing 5.6: Data entry to mongodb

The "shortAndSelected4" collection is used to keep the merged data which is used to create heatmap layer. One other collection is imd 2 which is used to view district information in the info windows that are loaded on map click.

5.4 Application

The application is a "express js" application which is a web application framework for "Node.js". "Node.js runs single-threaded, non-blocking, asynchronously programming, which is very memory efficient" (W3 Schools [2018]). This means that system does not wait for an instruction to complete in order. It goes on to run the next instruction, while previous instruction completes and responds. Express JS only provides wrappers to control the native functions of node js which can otherwise get complicated.

This is a server client application with data selection made in server while that data selection is rendered on to google map as heatmap layer at the client side. This could have been a client only application but huge data selection at run time through client only applications can be a problem as browsers are not as powerful as servers.

It is important in a server-client application to keep track of the control flow of the application from beginning to the end. As I explain the server and client side implementation of this application the text below will keep its focus on routing of the control flow, which is handled by the express js API.

5.4.1 Server Side

The server side code is written in server-side javascript file. This is the file used to run this application using "node server.js" command in the application root folder. The server code is receiving/"serving" http requests and giving http responses. The tasks at hand for the server are:

1. **Importing libraries** The require() function imports the libraries whose objects are later used to control the application behavior. The libraries we need are:

   (a) **Express Module**: As the application is a express js application it has to be imported using require().

```
1 const express = require('express');//importing express module
2 const app = express();//creating express application object
```

   Listing 5.7: Importing express js

   The "app" contains methods for "routing HTTP requests, configuring middleware, rendering views, registering a template engine, and modifying application settings that control how the application behaves" (MDN Web Docs [2018]). This means that all fundamental application behaviors can be controlled by calling Express "app"lication object functions.

   (b) **Mongodb**: Mongo db client object will later be used to select data by column.

```
1 var MongoClient = require('mongodb').MongoClient;// importing
      mongodb client library
2 var url = "mongodb+srv://abdurrehman545:<password>@cluster0-u
     5onq.gcp.mongodb.net/test?retryWrites=true";
```

Listing 5.8: Importing mongodb client

The url on line: 2 in the listing above points to the Mongo db replica set hosted in a cluster on mongo db server. This will be accessed several times to select a different column of data later in the server side code according to the received post request.

2. **Setting application environment**: app.use() function makes the public folder accessible to the application running on "localhost:3000".

   app.set('view engine', 'ejs') sets the template engine required to render the client side. By default it is 'html' in node js but to use express js facilities such as accessing rendered response objects we have to use the "ejs" template engine. This also means the system will look for '.ejs' files to render the view.

```
1 app.use(express.static(__dirname + '/public')); //the "public"
     folder becomes available to the application running at
     localhost:3000
2
3 app.set('view engine', 'ejs');//registering a template engine
```

Listing 5.9: setting up application

3. **Server listening (for requests) at a certain address**: This is port number "3000" of the host machine it runs on, which is "localhost" in my case. This is achieved by calling app.listen() function. As a result the application can be accessed at "localhost:3000" web address.

```
app.listen(3000, function () //app is listening for http (routing) requests
{
  //any code here
});
```

Listing 5.10: Usual parts of an express.js

4. **Enabling routing:**The routing functions are app.get(a,b) and app.post(a,b). The parameter a is the (html form)address from where this function is expecting the get or post request. The parameter b is a callback function which is called in case this function receives a request. As can be seen here: 5.12 any <input type="submit"> can generate a request but as specified by "method" and "action" parameter. On the server side app.post(action) will identify "method" type "post" and the action will be identified as '/watch' as can be seen on line 1 of listing: 5.13. On line 2 the if statement checks which IMD tab was clicked.

5. **Data selection:** The data: 3 columns i.e. lat,long, IMD score are selected from the "shortAndSelected4" mongo db collection each time a IMD tab is clicked. The IMD score column changes based on the IMD tab clicked. On the server side the if statement on line 1 in listing: 5.11 checks if incoming request is posted from "Health" tab. If it is so, the data selection will look like this:

```
1  if (req.body.selectionImd == 'Health')
2    {
3     MongoClient.connect(url, function (err, db) {
4        if (err) throw err;
5        var dbo = db.db("myNewdb2");
6        var data;
7            var e = dbo.collection("shortAndSelected4")
8          .find()
9          .project({
10           _id: 0,
11           lat: 1,
12           long: 1,
13           health: 1
14         }).toArray(function (err, result) {
15           var bundle = [];
16           var min = 0;
17           var max = 0;
18           result.forEach(function (value, index, ar)
              {
19             if (value.health < min) {
20               min = value.health;
21             }
22             if (value.health > max) {
23               max = value.health;
24             }
25             bundle.push({
26               lat: parseFloat(value.lat),
27               long: parseFloat(value.long),
28               weight: (  parseFloat(value.health)),
29             });
30
31
```

```
32          });
33
34          dbo.collection("imd2").find().toArray(function(err2,
    resultImd){
35
36            res.render('watch', {
37              data: bundle,
38              imdData:resultImd,
39              imd: 'Health',
40              error: "Nothing out of mongo",
41              min: min,
42              max: max
43            });
44          });
45
46        });
47     });
48   }
```

Listing 5.11: Data selection

First thing to notice in the above code is that mongodb client is connected to the server identified by "url" on line 2 in listing: 5.8. Then on line 5 connection is made to the data base "myNewdb2". Lines 7-14 data is queried using find(). The "find()" function with empty parenthesis returns all rows in the collection. The project() function selects the columns. 0 means to hide the column while 1 means to show it in the returned array. We can notice the three columns <lat,long,health> are asked to be returned. The toArray(callback) on line 14 catches the returned array in the "result" parameter of the callback function. It is then used to form "bundle" array by formating data as required on the client side to be rendered on google map.

6. **Rendering a response:** In javascript "A callback is a function that is to be executed after another function (inside callback function's scope)has finished executing"(Brandon [2017]). So in this case we had request and response objects in the call back function which helped form the response that was rendered to the client side. Response.render() is the last function that is executed in each get or post request.

Res.render() inside app.get() or app.post(), is used to "render" data to the client side. This could have alternately been achieved by res.send(). The difference between the two is that we need multiple data sets while res.send() can only send a single file or data to the client side. Render keeps the original data format automatically while before res.send() we have to specify the format such as res.accepts=’json’. We can also route to a different address(page) with res.render(). Such as a app.get(’home’) originates from home page and res.render(’watch’...) leads to the watch page.

5.4.2 Client Side

The client side is a .ejs file and it’s role is making requests and receiving responses and rendering views based on those responses. The tasks handled at client side are:

1. **Generating requests**: When a user interacts with an element inside "html form" such as a button, it generates a request and routes control flow to server. At the server side there are get() or post() functions to handle those requests and generate responses. The following lines of code depict an input such as Health(IMD) tab generating a post request caught by a post() function.

   First thing to notice is the method value is "post" in the form tag, this differentiates between get and post server methods.

   Secondly, one can notice action value is same as the post function’s first parameter. We can also notice the if statement in server code below: line 2, is testing which input was interacted(or which button was clicked) with, inside the form.

```
1  <form action="/watch" method="post">
2    <div class="tab" align="right">
3      <input name="selectionImd" type="submit" class="tablinks"
   value="Health" id="Health">
4      <input name="selectionImd" type="submit" class="tablinks"
   value="Crime" id="Crime">
5 .
6 .
7 .
8 </form>
```

Listing 5.12: Client Side: Input inside HTML form

```
1  app.post('/watch', function (req, res)  {
2  if (req.body.selectionImd == 'Health') {
3    .
4    .
5    .
6    }
7    .
8    .
9    .
10   });
```

Listing 5.13: post function that will catch the request

2. **Receiving data**: client side uses multiple datasets so it is immediately checked for 'null' datasets. This also differentiates if the page is loaded for the first time i.e. if null data is found, or if a data selection request is already made. If a data selection was made and data is rendered from the server it is now time to prepare the heatmap layer array. The code between <%-%> tags can access the data that is rendered to the client side by the server by res.render(), as shown below on line 2 in listing: 5.14.

3. **Creating and rendering heat map data array on google map**:Please refer to listing: 5.14 below.

   First thing to consider is we need each row to contain "google weighted location" object. The "bundle" array gets the data in line 2 that we selected in the server code and rendered to the client side in a response object. The received data is of the form:<lat,long,weight>. In the heat map data array(heatStack[] in the code below) we need <lat,long> to be inside the google location object which is created at run time. The above is repeated for each row found in the response data(now the bundle array). This part of client side can be seen on line 5-14.

   New map is created inside the html element named 'map1', which is a html div on line 15.

   "Heatmap" variable declared on line 1 is given a new heatmap layer to load. On line 15-28 we can see the preperation of the base map. It includes setting up map with zoom declared on line 16, map center which is given by latitude and longitude on lines 17-21. We can notice the min and max zoom restriction on line 24-28 which is discussed in literature review and evaluation.

```
1        var map, heatmap;
2        var bundle = <%-JSON.stringify(data)%>;
3        var heatStack = [];
4
5        bundle.forEach(function (value, index, ar) {
6          var weight = parseFloat(value.weight);
7          var loc = new google.maps.LatLng(parseFloat(value.lat),
     parseFloat(value.long));
8
9          heatStack.push({
10           location: loc,
11           weight: weight,
12
13         });
14       });
15       map = new google.maps.Map(document.getElementById('map1')
    , {
16         zoom: 6,
17         center: {
18
19           lat: 52.5645,
20           lng: 1.4669
21         },
22         mapTypeId: 'terrain',
23       });
24       var opt = {
25         minZoom: 6,
26         maxZoom: 10
27       };
28       map.setOptions(opt);
29       heatmap = new google.maps.visualization.HeatmapLayer({
30         data: heatStack,
31         map: map,
32       });
```

Listing 5.14: creating map and rendering heat map data

4. **Extending interactivity functionalities**: There are a few interactions that client side code has overridden or extended from the default behavior.

(a) **Places search box**: Google's places search box is discussed in literature review in Chapter: 2 and it's usability benefits in this application are discussed in Chapter: 6. This section describes it's usage in my application. The sample code from this application is given in Appendix:A.2, this application only slightly modifies the available code by adding a region bias in the places autocomplete results. For example while user starts typing, addresses belonging to UK should appear first in the list. This can be achieved by simply setting the "options" variable in the following manner. This will modify the default search box behavior. For UK addresses to appear first we set country as 'gb' in the "componentRestrictions" property. We can also notice types 'regions'. This will make administrative boundaries like city, district, town to appear before individual building addresses.

```
function searchPlace() {
var options = {
types: ['(regions)'],
componentRestrictions: {
country: "gb"
}
};
```

(b) **Interactivity changing data representation**: When the application user clicks one of the IMD tabs, a request is generated. The server checks which tab was clicked and responds with corresponding data set.

i. **Restricting Zoom**: As can be seen on line 24-27 in listing 5.14

ii. **Changing radius** is a data representation change. This means each data point is represented by 10 px of space on the base map. When a "change radius" interaction is made the radius gets changed to 20px. This changes the physical representation of the data. This is achieved by the following lines of code:

```
1  function changeRadius()
2  {
3  var radiInUse = heatmap.get('radius');
4  if (radiInUse == 10) {
5  heatmap.set('radius', 20)
6  } else {
7  heatmap.set('radius', 10)
8  }
9
10 }
```

Listing 5.15: Changing Radius

The function above changeRadius() is called when "change radius" button is clicked from the application interface. As you can notice from the above code it is just toggling between 20px and 10px radii. The heatmap.get(propertyname) function is in use at line 3, it is used here to get the value of radius property. In the following line it checks for the current value and changes it to the other value using heatmap.set('property',value).

Changing opacity: The default opacity value can be overridden as shown below. The acceptable values are real number values from 0 to 1.

```
1 function changeOpacity()
2 {
3 heatmap.set('opacity', heatmap.get('opacity') ? null : 0.2);
4 }
```

Listing 5.16: Changing opacity

The functions to get and set the heatmap property(heatmap.set()) on line 3 , this time property name is 'opacity'. The tertiary statement is used here which means it toggles between the default set by google maps and the value given by us, i.e. if the property value is unset or null change it to 0.2 and vice versa. This will reduce the opacity of the heat map layer over the base map or alternately increase it.

(c) **Map Click:** To extend this functionality I had to write an event listener function. Google maps api provides various events that can be used to trigger further actions. In my application this event loads/updates the news section.

This function does the following:

i. **Gets postal town through reverse geocoding**: The click event object contains the location (latitude,longitude) on map which was clicked by a user. This location(lat,long) pair is passed to the google geocoder api which reverse geocodes it to an address array named "results". We need 2 addresses from this results array. To know more about address types please refer to Chapter 2

```
1 var newsString;
2 results.forEach(function(value){
3 if (value.types[0] == "postal_town")
```

```
{
newsString = value.address_components[0].long_name;
}
});
```

Listing 5.17: Searching results array for postal town

ii. **Gets district through reverse geocoding**: The "administrative_level_3" address type is most probably the name of the district. If this address type does not exist in the result array then "administrative_level_2" address type will match a district authority name in the district IMD dataset. So they are checked in that order. The function is made to break as soon as it finds the district.

```
for (var i = 0; i < results.length; i++)
{
  if (results[i].types[0] == "administrative_area_level_3
    ")
  {
    console.log("District", results[i])
    strToFind = results[i].address_components[0].long_name
    ;
    rowNumber = searchStringInArray(strToFind, imdData);
    console.log(rowNumber)
    break;
  }
  else if (results[i].types[0] == "
    administrative_area_level_2")
  {
    strToFind = results[i].address_components[0].long_name
    ;
    rowNumber = searchStringInArray(strToFind, imdData);
    console.log(rowNumber)
    console.log("city", results[i])
    break;
  }
}
```

Listing 5.18: Searching results array for districts

iii. **Load Google Infowindows**: At each map click google map's "info window" pops up. The district information we got from reverse geocoding is used to match the district authority name. Its corresponding average IMD decile along with the town and district name is displayed in this info window.

```
1 var infowindow = new google.maps.InfoWindow;
2 infowindow.setContent("<b>Town: </b>" + newsString + "<br
     ><b>District: </b>" + strToFind +
3 "<br><b>Mean " + imdThis + " decile:</b> " +
4 imdDecile +
5 "<br><b>Mean Population:</b> " + population);
6 infowindow.open(map, marker);
```

Listing 5.19: load info windows

As can be seen above, info windows act as html pages. Any text can be displayed inside info windows by using infowindow.setContent(text). One can also notice that first an info window is created seperately(line 1 to 5) and then opened on a map with a marker as seen on line 6 in the code above.

iv. **Load news**:News articles mentioning IMD theme and the town name are requested via HTTP request from the newsapi.org which responds with a JSON data array which contains news articles. The following code, listing: 5.20, is making a request given a URL encoded string:

```
1 var xhr = new XMLHttpRequest();
2 var place_name_url_encoded = encodeURI('("' + postTown +
     '"AND"' +
3 imdThis + '")');
4
5 xhr.open('GET',
6 "https://newsapi.org/v2/everything?sources=bbc-news,the-
     guardian-uk&q=" +
7 place_name_url_encoded +
8 "&sortBy=popularity&apiKey=4eee6a0fd1ac4b97b84b00815012e40
     6",
9 true);
10 xhr.send();
11 xhr.onreadystatechange = processRequest;
```

Listing 5.20: requesting news articles from newsapi.org

As can be seen in the code above on line:2 "newsString" variable is carrying the postal town value we obtained in listing 5.17 on line 5. For e.g. if a "Health" heat map is loaded and map is clicked at <lat=51.511994,long=-0.090195> the request address will look like the one below:

```
1 https://newsapi.org/v2/everything?
2  sources=bbc-news,the-guardian-uk
3  &q=%28"London"AND"Health"%29
4  &sortBy=popularity
5  &apiKey=x
```

Listing 5.21: newsapi.org http request URL

The http request address above has a number of parameters in it. First thing to note here is the specific library is used:"everything". This library gives access to all latest news stories unlike others such as "top headlines". Next, one can notice the source of news specified:"bbc-news" and "the guardian". Lastly one can see the query to select news requesting articles that mention "London" and "Health" together. This produces a JSON array like the one here: 5.22.

```
1 {
2     "status": "ok",
3     "totalResults": 70210,
4     "articles":
5     [{
6         "source": {
7             "id": "bbc-news",
8             "name": "BBC News"
9         },
10        "author": "https://www.facebook.com/bbcnews",
11        "title": "London sexual health clinics ’
    oversubscribed’",
12        "description": "Several clinics have closed in
    London, with at least one NHS trust turning patients
    away.",
13        "url": "http://www.bbc.co.uk/news/uk-england-
    london-42771665",
14        "urlToImage": "https://ichef.bbci.co.uk/news/1024/
    branded_news/E1D1/production/_99690875_hi015118584.jpg"
    ,
```

```
        "publishedAt": "2018-01-22T17:41:32Z"
    },
    .
    .
    .
    {
        "source":
        {
            "id": "daily-mail",
            "name": "Daily Mail"
        },
        "author": "http://www.dailymail.co.uk/home/search.html?s=&authornamef=Associated+Press, By Associated Press",
        "title": "Thousands march to demand more money for UK health service",
        "description": "LONDON (AP) - Thousands of people have marched through London demanding more government money for Britain’s overburdened National Health Service.Trade unions...",
        "url": "http://www.dailymail.co.uk/wires/ap/article-5347891/Thousands-march-demand-money-UK-health-service.html",
        "urlToImage": "http://i.dailymail.co.uk/1/2018/02/03/14/wire-2218154-1517669278-463_636x382.jpg",
        "publishedAt": "2018-02-03T14:48:08Z"
    }]
}
```


Listing 5.22: newsapi.org result json array

One can see the total results returned are 70,200. Each result entity has different documents like author, title, url etc which are used to create a display for news section.

Chapter 6

Evaluation And Results

The usability of GIS has been under debate since early 1960s and since then HCI has been an integral part of GIS. We design our map application in a way that makes the usage effective efficient and enjoyable (Haklay and Zafiri [2008]). Haklay and Zafiri further notes that the size of the map area as a proportion of the total interface can have significant implications on the effectiveness of the application. Google maps interface provides a full screen option that naturally puts rest of the application interface behind. Usability engineering studies recommend that the proportion of screen used for mapping must allow rest of application interface to create a context for a user who is not very efficient.

The application is made out of 2 things:

1. The map
2. The full application (interface) including the map

The map should take most of the screen asset, which it does in our case where 80% of the screen width and 88% of screen height is dedicated to map use. The 20% of remaining screen width is dedicated for the tabs that select data that will directly change the map look or load a heat map layer for the first time. This is an interactivity which changes the data.

The earlier iteration of my application made a mistake: when a tab was clicked and the heat map layer was loaded it did not mention which theme loaded the heat map layer. This was pointed out in testing period by the supervisor so corrected in later iterations.

One other usability mistake that still remains is that the map legend is not on the map face while other map interactions like changing radius and gradient sit on top

of the map. The point here is that any interactions that change the map must be strategically located on the map just like English IMD Map [2015] or Parallel.co.uk [2015]. This is better than my map in a way that it gives 100% of screen width and height to the map. While loading map key or map legend, a bar for switching IMD themes like our tabs on the left,grid of results on clicking the map are also loaded on the map face. However they do not consider rest of application interface as important, rather there is no application beyond map.

6.1 Experiments

The application may allow a user to ask the system some analytical questions related to a switchable IMD theme such as Health, Crime etc in a region. As we have learnt in our Literature review heat maps can be used to test hypothesis. These questions below are an example of actions that a person will do to test that hypothesis.

6.1.1 What is the Country Level Health Situation?

The user is greeted with first Screen with the data selecting tabs on the left with no map loaded as can be seen here 6.1. At this point system is waiting for user to interact with it. Data will be selected and heat map will be loaded for that data, with a single (IMD)tab click as shown here 6.2.

As one can see the map is loaded at zoom level 6 and it summarizes the country level situation. This can be described as mostly green with some chunks of red around major cities like "Manchester", "London" etc and some regions in the North Eastern and Southern part of England.

As explained in literature review that the colors are formed due to point density interpolation. Each data point has a radius of influence and this creates a buffer zone around a data point. Each data point has a weight attached to it which gives this buffer a color. These buffers when overlap with buffers of other data points it creates a resultant color. By looking at the visualization it shows large green areas and this just means that most of data points in England have a weight close to Health Score of 3.458. When we zoom in the data points get distant from each other and out of each other's radius of influence. As will be seen later on zooming in, this visual representation changes.

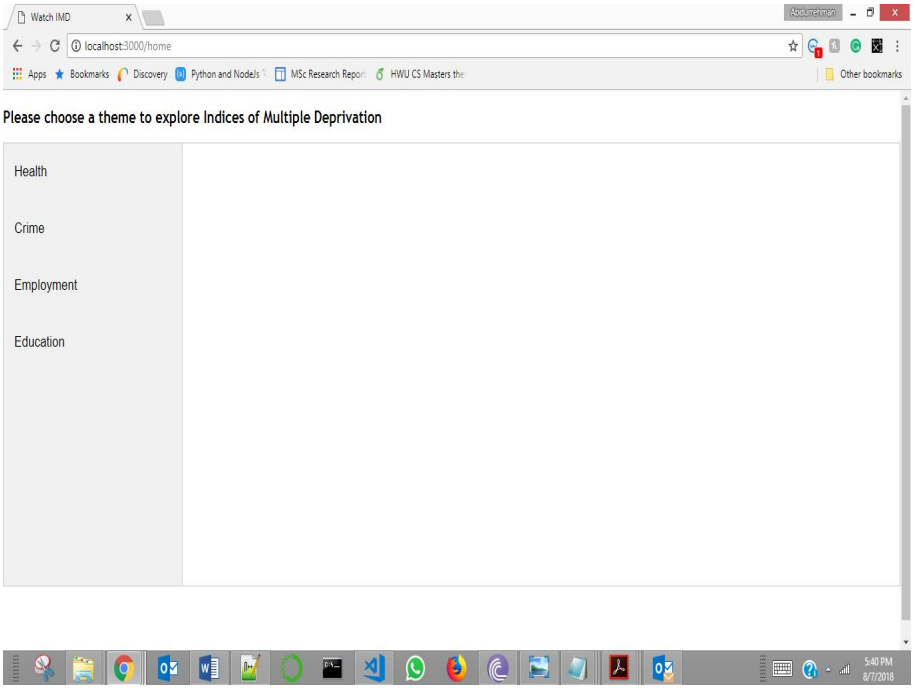

Figure 6.1: First page

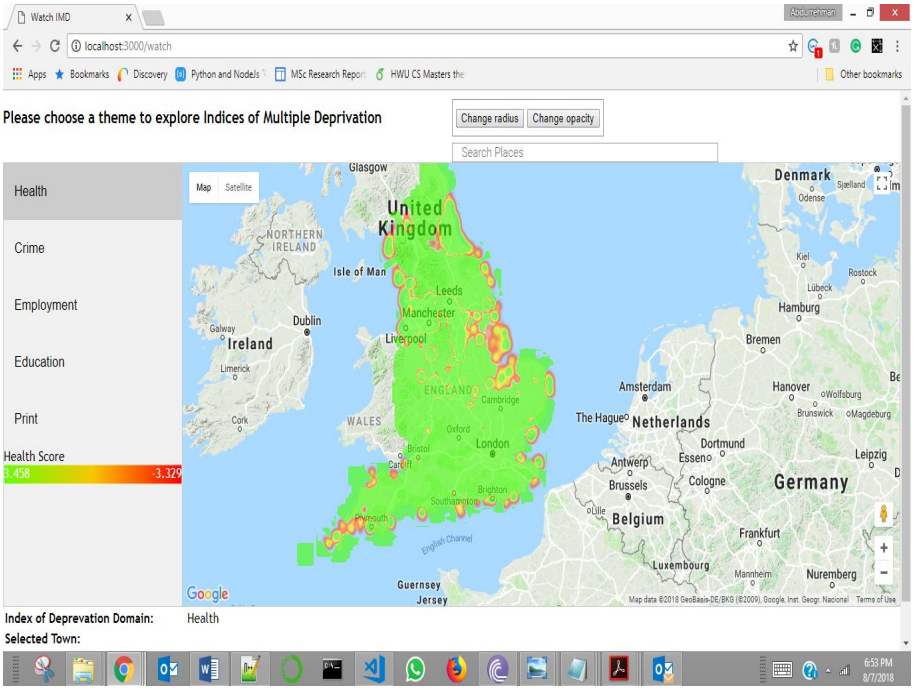

Figure 6.2: Health heatmap layer

6.1.2 What is the crime situation in "City of London"?

This is a district level Information as can be seen in Figure 6.3, London can be seen at zoom level 6. A user has two choices to reach "City of London" district. He can either search it, which is also the easy way. Or he can zoom and pan towards it till he reaches it on the map. Searching for a place will require less interaction that is how it adds to the ease of use and the ease to find a place just like HCI puts emphasis on providing ease to complete a task efficiently. A user can then click the place he wants to view the detail of. In any way either by searching or by zooming and panning the resulting map is shown here 6.3 and 6.4.

6.1.3 what is the Employment situation in or around an address?

The address can only be pointed out by the help of search box because I have restricted the maximum zoom level to 10 it's difficult to find exact address or to reach streets at this zoom level. A place search will help. To load Employment theme heat map layer we click the tab on the left named "Employment". The resulting map can be seen here 6.5. We can see this is a more differentiating map than Health map. The reason is that the range is not very wide even(0 to 0.58) close lying regions are not on same deprivation value. One other thing we can note that we can see green region around big cities like Manchester, London etc and red regions around more rural areas, this can be an indication that rural areas have less jobs.

Post Code: "TW3 3AP" As can be seen in 6.6 the place is reached easily by searching for a for it. This way we can learn if a place is situated in a well of neighborhood or not.

6.1.4 compare the Education situation between neighboring districts "Hounslow" and "City of West Minster"

Just like before a user loads the heat map for Education theme and the resulting map can be seen here 6.7. A similar trend can be seen here i.e. big cities like Manchester and London are more green in color while rural areas are more red.

The user can find "Hounslow" and "City Of Westminster" by zooming and panning or if he/she cannot see them at max zoom he can search for them one after other. We can also load info windows side by side to compare the details of these 2 regions. Figure 6.8 show this phenomenon. This confirms the country level trend too, i.e.

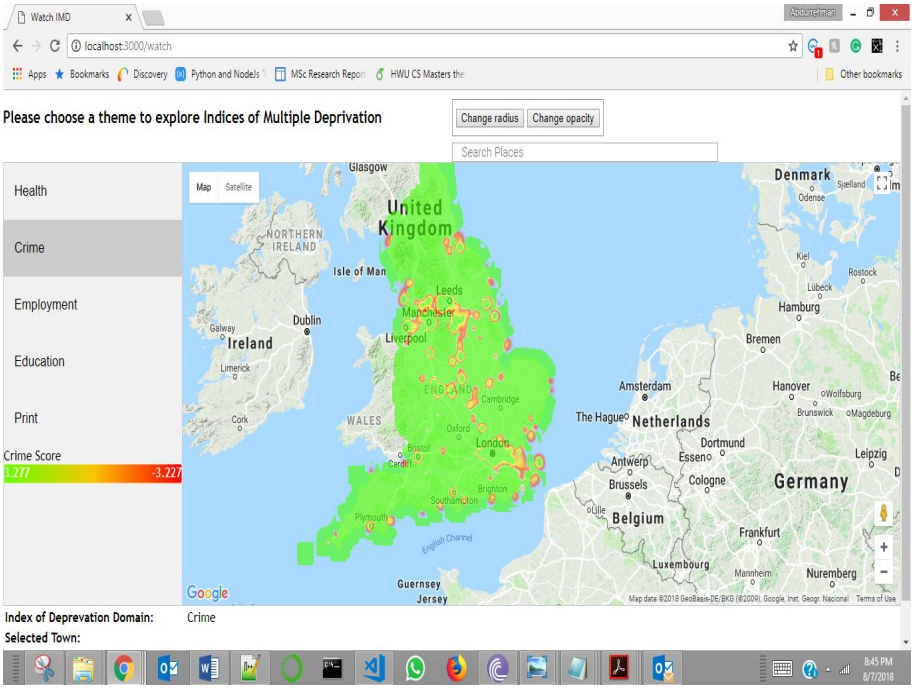

Figure 6.3: Crime heatmap layer

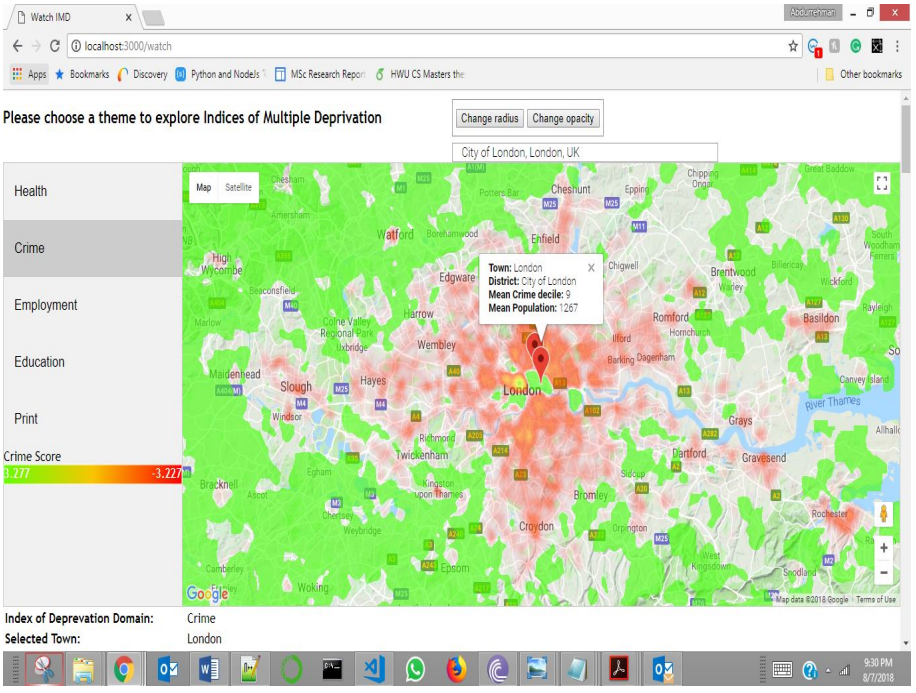

Figure 6.4: Crime in City of London

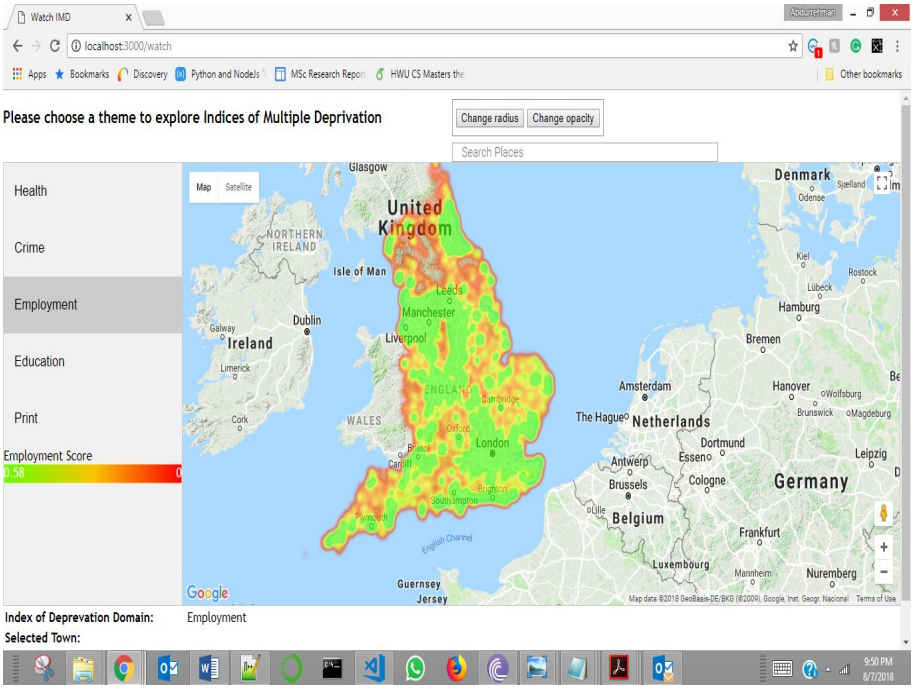

Figure 6.5: 3a.

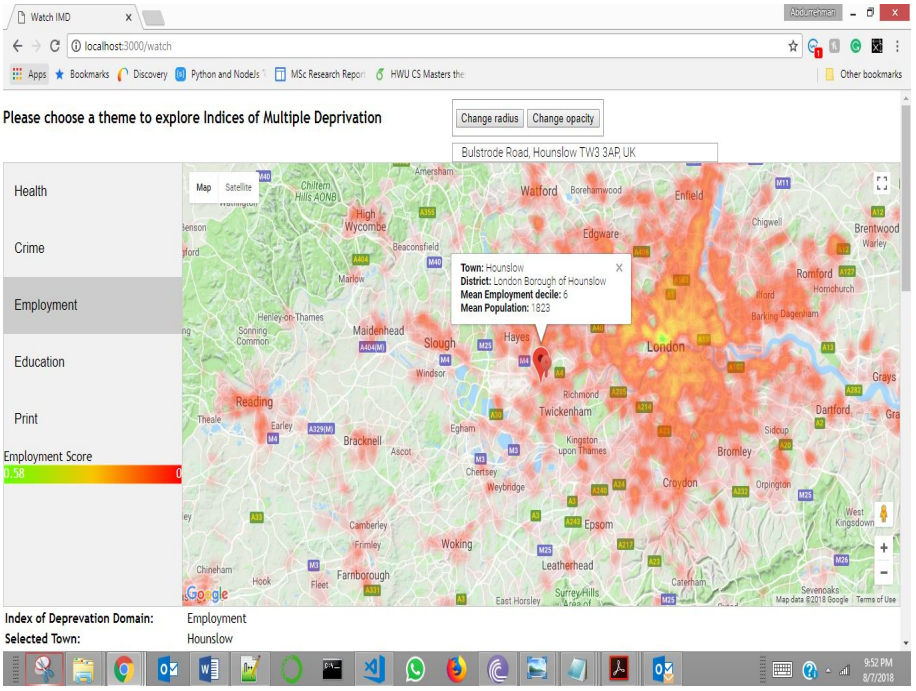

Figure 6.6: 3b.

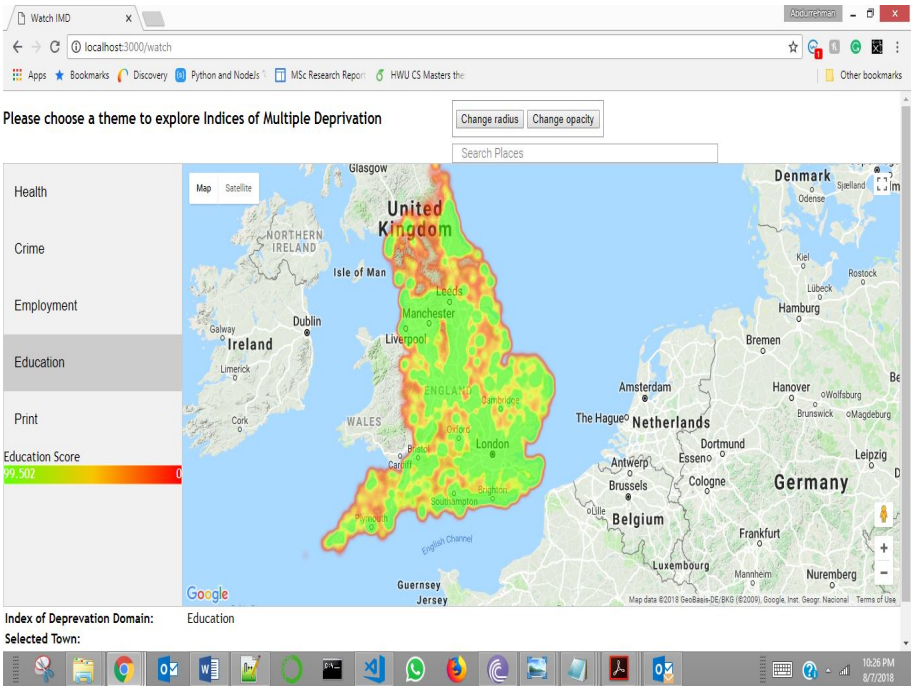

Figure 6.7: 4a.

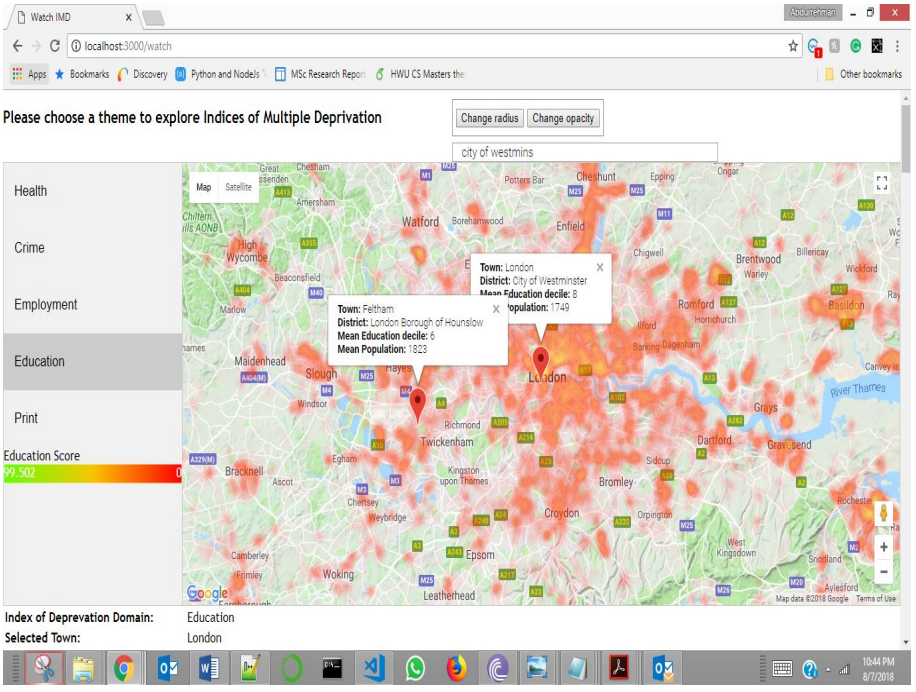

Figure 6.8: 4b

we see a drop in living condition related to Education as we move away from city centers.

6.1.5 User wants to read "Crime" news related to Bradford

Given "crime" heatmap is already loaded. A user who is interested in knowing crime situation in random parts of country, can "search" for random places of interest like "Bradford" or browse to it by zooming and panning. Upon finding "Bradford" the user clicks a map region around Bradford and he can then see the info window with some more IMD details and news loaded in the news section. This can be a perfect application for a user who requires news by region for e.g. news by town or one particular town. As mentioned in literature review this functionality is an example of response from map to application interface. This use case can be seen here: 6.9.

There is a chance that application user might need a prior understanding of what the data means. For example the map heading reads "Please choose a theme to explore Indices of Multiple Deprivation". While this might be enough for someone who knows what is the data about, on the contrary, a user, who has no idea what Indices of Multiple deprivation data is, might not develop an interest to use this application any further. To mitigate this, the application interface should:

1. guide a user what this data and its values mean: This can be simply done by providing a link to the source of data and it's providing authority, in our case the (data.gov.uk [2018];ONS Geography [2018]). It can also have map tours or story maps ESRI [2018]. One disadvantage of using story maps is that the map interface cover over application interface will decrease even further.

2. what each theme means: Tab names are self explanatory but they should have been backed up by an excerpt from the documentation clarifying what they mean.

3. what are the various range of values for each theme: This is partially done by legend provided at the end of the tab menu, it should have been placed on the map interface.

4. how does the color on map change for each range of values: This again is partially covered by the legend although it is not rightly positioned.

One problem with jumping to places with search box is that we might miss the changing data representation on each zoom level. There can be a freelancer journalist user who wants to know the problems faced by various regions in the country in various fields like Health Crime etc. Such a user will most probably chase the red color and might want to zoom in to pick a region facing worse situation, so that he can find a base for his research.

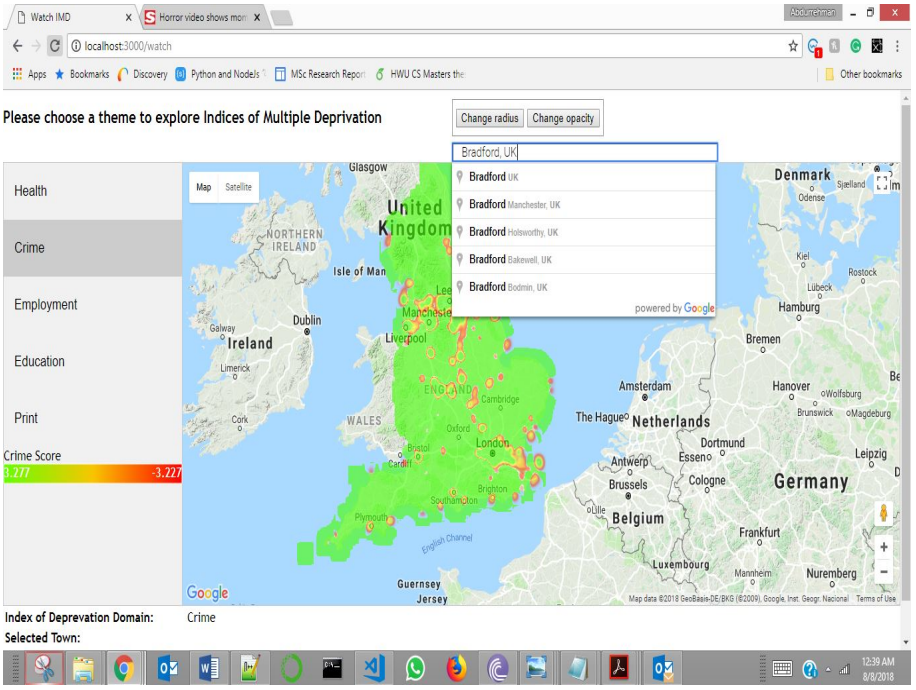

Figure 6.9: 5a.

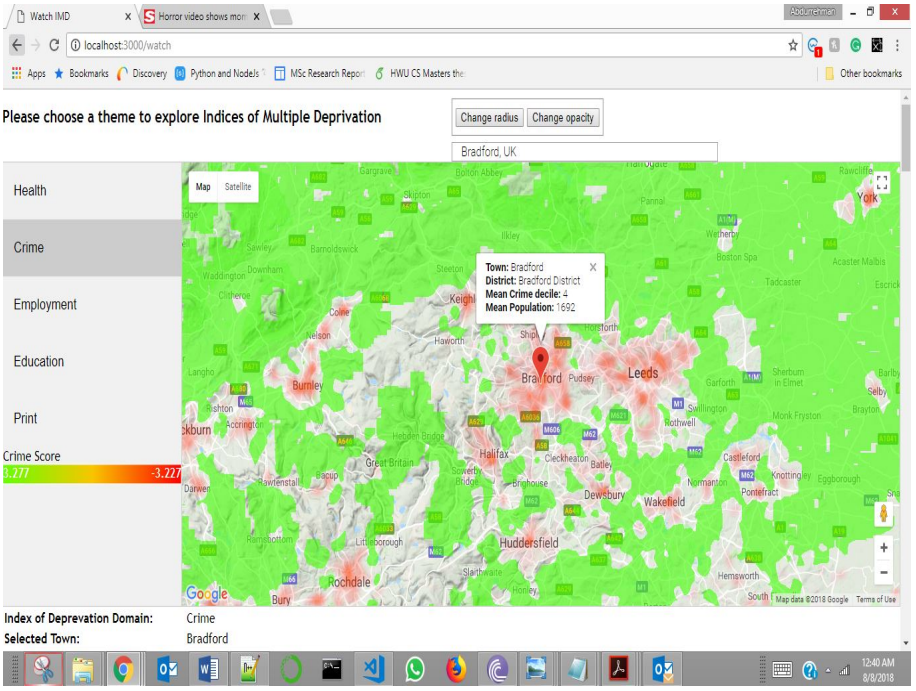

Figure 6.10: 5b.

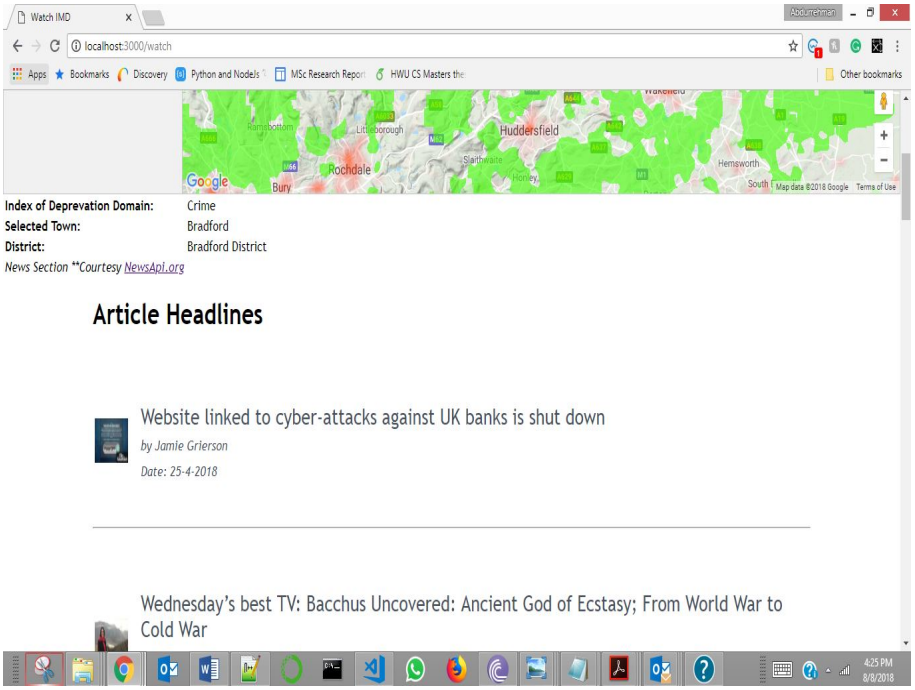

Figure 6.11: 5c. Crime News for "Bradford"

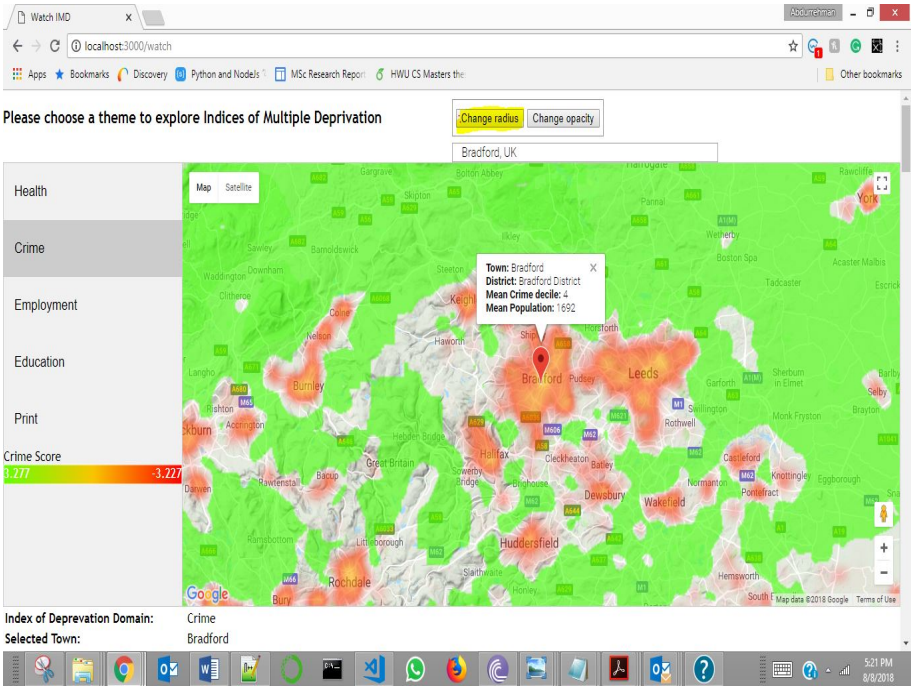

Figure 6.12: 5c. Changed Radius

Sometimes while zooming in to max level, i.e. 10, the heat map starts to disappear which is a common property of heat maps. This happens as mentioned earlier, due to individual data points getting away from neighboring points on zooming in. In these cases it is useful to increase the radius so that the visualization still works as seen in Figure 6.12.

The user might be novice in data science but a user of our application can be assumed to have a reasonable understanding and experience of using the google map interface. As we know there are millions of people who have access to a smart phone or a PC and have used google maps interface regularly if not on a daily basis.

So, understanding and using interactions might not be a problem in this application. Problem might appear if a user does not understand the value of data and might not be satisfied with our visualization effort. For such a user reasonable amount of information is needed about how this application should let a user test a hypothesis. To facilitate such a user we can introduce user manuals or tutorial videos allowing a user to learn how to use this application before he goes on to use it.

Chapter 7

Conclusion and Future Work

On the basis of literature review and the practical applications I have explored on the internet and the processes I have gone through myself, we can conclude that cartography or map making is not new, what's new is a map's ability to display exploratory data using interactions. This task is made very simple by open mapping technologies that are reliable in most cases 99.9% accurate(google, arcgis etc) (Roongpiboonsopit and Karimi [2010]).

Secondly, we can conclude that the modern online map visualization applications are capable of keeping a user interested, courtesy eye catching map interfaces like google map's interface is universally accepted and understood. So, any application which uses these mapping providers inherit the visual acceptability. For it's better market, google maps is apparently the best choice for mapping technology, they are a tech giant and they spend considerable amounts of funds and work into designing their features it is best to just use those services. As a developer their is a plethora of documentation and help forums on google maps javascript API that help you getting unstuck.

Thirdly, the data. The open government movement which is a part of the big data revolution itself has sparked a whole new debate within mapping communities, i.e. to ask for the availability of geospatial open government data for example census data was the first to be disseminated within UK. Where exploratory visualizations unearth analytical trends of a region it is better for the citizens of that region to know the quality of life in geographic detail. So they can move to a better location or take a picture and report their concerns to someone responsible.

Fourthly the state of my application is not final there are a lot of interesting things that can be achieved in the future. A few proposals at this stage can be:

1. Mongo db provides Geo Spatial data management and functions. It will be interesting to work with a specialized database system that can hold geographic information.

2. The data can be updated quarterly for example NSPL dataset. It is a greater problem with big data or huge data with heavy in flow of data. In that case manual or supervised data preprocessing has to be a constant head ache. We need to find a way to acquire data automatically by a piece of code written in server side. This can be achieved by using newer sources of data such as the linked data repository(https://www.ukdataservice.ac.uk/). As most of data found at uk government sites is consistent because their disseminating authority is credible and highly funded, the data integration and cleaning is not a huge problem. However data selection and loading into a nosql database system can still consume a few minutes which will be very rare, such as quarterly or whenever data will be updated.

3. As a part of future work I will also like to compare various top mapping technologies. For example Google Maps Versus ArcGIS versus leaflet. For example prototyping simultaneous similar maps using different mapping technologies.

4. Currently when we select a IMD theme that is just supposed to switch the heat map layer, it goes back to server to get new data selection. This should be avoided in any future work because toggling visualization layers should be handled at client side. So the different layers should just be toggled not the full map. As pointed out in (Smith [2016];Jeremy [2002]) interactivity includes toggling different layers of visualizations. So that a user can visualizes trends and generates a hypothesis in a better way. Simply this means that the base map should not be reloaded rather only the heat map layer should switch.

5. To keep a user interested, the application interface should be simple but have a little more information on the datasets, their source of origin and their credibility.

In a nutshell this field of socio economic exploratory interactive and analytical map making is evolving very quickly due to a combined effect of big data, open Government data and open source mapping. The situation is superficial as the phenomenon is mostly understood by researchers, academia and developer enthusiasts and so their utilization at this time is some what scientific and their topics of discussion are largely uninteresting for general public. General public in my opinion is mostly interested in their own life. For example a direction map application that can choose a path based on safety and security. This science can make an impact if it has the

ability to trickle down to people's every day lives regardless of their careers. For example this application can be useful for house searching, planning a business or political hypothesis testing. One advantage of this not being a final application is that it can be molded in any way. This can also add up to some CEO's dashboard at work, if the visualizations are changing by minutes or hours, which means we should have that much open government data **velocity**. Even if this data set is not used we can use big data at the back end which has way more **velocity** that can change a visualization as things happen.

Appendix A

Reverse Geo Code Result

A.1 Google Heat Map Layer Weighted Locations

```
/* Data points defined as a mixture of WeightedLocation and LatLng objects */
var heatMapData = [
  {location: new google.maps.LatLng(37.782, -122.447), weight: 0.5},
  new google.maps.LatLng(37.782, -122.445),
  {location: new google.maps.LatLng(37.782, -122.443), weight: 2},
  {location: new google.maps.LatLng(37.782, -122.441), weight: 3},
  {location: new google.maps.LatLng(37.782, -122.439), weight: 2},
  new google.maps.LatLng(37.782, -122.437),
  {location: new google.maps.LatLng(37.782, -122.435), weight: 0.5},

  {location: new google.maps.LatLng(37.785, -122.447), weight: 3},
  {location: new google.maps.LatLng(37.785, -122.445), weight: 2},
  new google.maps.LatLng(37.785, -122.443),
  {location: new google.maps.LatLng(37.785, -122.441), weight: 0.5},
  new google.maps.LatLng(37.785, -122.439),
  {location: new google.maps.LatLng(37.785, -122.437), weight: 2},
  {location: new google.maps.LatLng(37.785, -122.435), weight: 3}
];

var sanFrancisco = new google.maps.LatLng(37.774546, -122.433523);

map = new google.maps.Map(document.getElementById('map'), {
  center: sanFrancisco,
  zoom: 13,
  mapTypeId: 'satellite'
});

var heatmap = new google.maps.visualization.HeatmapLayer({
  data: heatMapData
});
heatmap.setMap(map);
```

A.2 Google Search place Sample

```
<input id="pac-input" class="controls" type="text" placeholder="Search Places" style="margin-left: 50%;">
    <script>
            searchPlace();

    function searchPlace() {
      var options = {
        types: ['(regions)'],
        componentRestrictions: {
          country: "gb"
        }
      };



      var input = document.getElementById('pac-input');
      var searchBox = new google.maps.places.SearchBox(input);
      var autocomplete = new google.maps.places.Autocomplete(input, options);


      // Bias the SearchBox results towards current map's viewport.
      map.addListener('bounds_changed', function () {
        searchBox.setBounds(map.getBounds());
      });

      var markers = [];
      // Listen for the event fired when the user selects a prediction and retrieve
      // more details for that place.
      searchBox.addListener('places_changed', function () {
        var places = searchBox.getPlaces();

        if (places.length == 0) {
          return;
        }
```

```
34
35          // Clear out the old markers.
36          markers.forEach(function (marker) {
37            marker.setMap(null);
38          });
39          markers = [];
40
41
42          // For each place, get the icon, name and location.
43          var bounds = new google.maps.LatLngBounds();
44          console.log(places)
45          for (var i = 0; i < places.length; i++) {
46            place = places[i];
47            if (!place.geometry) {
48                      //Error
49              return;
50            }
51            var icon = {
52              ...};
53
54            // Create a marker for each place.
55            var marker2 = new google.maps.Marker({
56              map: map,
57              icon: icon,
58              title: place.name,
59              position: place.geometry.location
60            })
61            markers.push(marker2);
62            ...
63        });
64      }
65    </script>
```

Listing A.1: Google places search box

Bibliography

AbdurRehman, S. (2018). map_vis. https://drive.google.com/open?id=1UpZYyLr9zgasqX8nKOet86Tg6dK8wV-M. Accessed: 2018-08-08.

AlAnazi, J. M. and Chatfield, A. (2012). Sharing government-owned data with the public: a cross-country analysis of open data practice in the middle east.

Alazmi, A. R. and Alazmi, A. R. (2012). Data mining and visualization of large databases. *International Journal of Computer Science and Security*, 6(5):295–314.

Ayala, C., Øyvind Hauge, Conradi, R., Franch, X., and Li, J. (2011). Selection of third party software in off-the-shelf-based software development—an interview study with industrial practitioners. *Journal of Systems and Software*, 84(4):620 – 637. The Ninth International Conference on Quality Software.

Ballatore, A., Tahir, A., McArdle, G., and Bertolotto, M. (2011). A comparison of open source geospatial technologies for web mapping. *International Journal of Web Engineering and Technology*, 6(4):354–374.

Bradner, S. (1997). Best current practice.

Brandon, M. (2017). Javascript: What the heck is a callback? https://codeburst.io/javascript-what-the-heck-is-a-callback-aba4da2deced. Accessed: 2018-08-08.

Castanedo, F. (2015). *Data Preparation in the Big Data Era*. O'Reilly Media, Inc., 1 edition.

CHIKKALA, S. S. (2017). Evaluation criteria for selection of api products.

data.gov.uk (2018). English imd. http://dclgapps.communities.gov.uk/imd/idmap.html. Accessed: 2018-07-17.

DeBoer, M. (2015). Understanding the heat map. *Cartographic Perspectives*, 0(80).

Diehl, J. (2004). Preprocessing and visualization. In *Seminar Data Mining in WS*, pages 1–21.

Dodsworth, E. and Nicholson, A. (2013). Visualizing our futures: Using google earth and google maps in an academic library setting.

Dupin-Bryant, P. A. and Olsen, D. H. (2014). Business intelligence, analytics and data visualization: A heat map project tutorial. *International Journal of Management & Information Systems (Online)*, 18(3):185.

English IMD Map (2015). English imd map. http://dclgapps.communities.gov.uk/imd/idmap.html. Accessed: 2018-08-08.

ESRI (2018). Esri story maps. https://storymaps.esri.com/stories/maptour-sp-palmsprings/. Accessed: 2018-08-08.

ESRI (2018). Heat Map vs. Hot Spot Map. https://yuhuinnovation.maps.arcgis.com/apps/MapJournal/index.html?appid=b89b59ef0ad14534aae4e5fc85fdb2eb. Accessed: 2018-07-17.

Fernandes, A. I., Goulão, M., and Rodrigues, A. (2013). A comparison of maps application programming interfaces. *arXiv preprint arXiv:1305.3485*.

Fung, A. (2013). Infotopia: Unleashing the democratic power of transparency. *Politics & Society*, 41(2):183–212.

Gandomi, A. and Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, 35(2):137–144.

Gilbert, S. and Lynch, N. A. (2012). Perspectives on the cap theorem. *Computer*, 45(2):30–36.

Ginner, N. M. and Henderson, P. S. (2018). Finding Hot Spots using ArcGIS Online - Minimizing the Subjectivity of Visual Analysis . http://proceedings.esri.com/library/userconf/fed17/papers/fed_72.pdf. Accessed: 2018-07-17.

Glaser, B. G. and Strauss, A. L. (2017). *Discovery of grounded theory: Strategies for qualitative research*. Routledge.

Google Developer Console (2018). Google heat map layer. https://developers.google.com/maps/documentation/javascript/heatmaplayer?authuser=1. Accessed: 2018-07-17.

Google Developer Guide (2018a). Geocoding api. https://developers.google.com/maps/documentation/geocoding/intro. Accessed: 2018-07-17.

Google Developer Guide (2018b). Google api key. https://developers.google.com/maps/documentation/geocoding/get-api-key. Accessed: 2018-07-17.

Google Developer Guide Places Search (2018). Google place search. https://developers.google.com/places/web-service/search. Accessed: 2018-07-17.

Google Developer Guide Places Search Box (2018). Google Place Search Box. https://developers.google.com/maps/documentation/javascript/places-autocomplete#places_searchbox. Accessed: 2018-07-17.

Google My Maps (2018). https://www.google.co.uk/maps/about/mymaps/.

Graves, A. and Hendler, J. (2013). Visualization tools for open government data. pages 136–145.

Gurin, J. (2014). Big data and open data: How open will the future be. *ISJLP*, 10:691.

Hadjigeorgiou, C. et al. (2013). Rdbms vs nosql: Performance and scaling comparison. *EPCC, The University of Edinburgh*.

Haklay, M. M. and Zafiri, A. (2008). Usability engineering for gis: Learning from a screenshot. *The Cartographic Journal*, 45(2):87–97.

Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

International, S. (2015). Open data in big data world. *Science International*, 1.

Jeremy, C. (2002). Interactivity types in geographic visualization. *Cartography and Geographic Information Science*, 29(2):85–98.

Joel, G. (2014). Big data and open data: what's what and why does it matter? https://www.theguardian.com/public-leaders-network/2014/apr/15/big-data-open-data-transform-government. Accessed: 2018-07-17.

Koua, E. L., Maceachren, A., and Kraak, M. J. (2006). Evaluating the usability of visualization methods in an exploratory geovisualization environment. *International Journal of Geographical Information Science*, 20(4):425–448.

Lee, H.-Y., Ong, H.-L., and Quek, L.-H. (1995). Exploiting visualization in knowledge discovery. In *KDD*, pages 198–203.

Manikandan, S. (2010). Data transformation. *Journal of Pharmacology and Pharmacotherapeutics*, 1(2):126–127.

Matheus, R., Ribeiro, M. M., and Vaz, J. C. (2015). *Brazil Towards Government 2.0: Strategies for Adopting Open Government Data in National and Subnational Governments*, pages 121–138. Springer International Publishing, Cham.

MDN Web Docs (2018). Express/node introduction. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction. Accessed: 2018-08-08.

Michael, W. (2015). A common business question - why buy GIS when Google Maps is free. https://www.linkedin.com/pulse/common-business-question-why-buy-gis-when-google-maps-wallace. Accessed: 2018-07-17.

Ministry of Housing, Communities and Local Government (2015). English index of multiple deprivation. https://www.gov.uk/government/statistics/english-indices-of-deprivation-2015. Accessed: 2018-04-03.

newsapi.org (2018). https://newsapi.org/.

ONS Geography (2018). Postcode products. https://ons.maps.arcgis.com/home/item.html?id=0c31adaeb0444d119b336ca00cb54efe. Accessed: 2018-04-03.

ordnancesurvey.co.uk (2018). Gis in use. https://www.ordnancesurvey.co.uk/support/understanding-gis/gis-in-use.html.

Parallel.co.uk (2015). Parallel uk maps imd. https://parallel.co.uk/imd/. Accessed: 2018-08-08.

Parasie, S. and Dagiral, E. (2013). Data-driven journalism and the public good:"computer-assisted-reporters" and "programmer-journalists" in chicago. *New media & society*, 15(6):853–871.

Paul, J. (2015). Quantification of heat map data displays for high-throughput analysis. *Journal of Pharmacogenomics and Pharmacoproteomics*, 6(2):1–7.

Plantin, J.-C. (2018). Digital traces in context| google maps as cartographic infrastructure: From participatory mapmaking to database maintenance. *International Journal of Communication*, 12(0).

Powney, G. D. and Isaac, N. J. B. (2015). Beyond maps: a review of the applications of biological records. *Biological Journal of the Linnean Society*, 115(3):532–542.

Roongpiboonsopit, D. and Karimi, H. A. (2010). Comparative evaluation and analysis of online geocoding services. *International Journal of Geographical Information Science*, 24(7):1081–1100.

Roth, R. E. (2013). Interactive maps: What we know and what we need to know. *Journal of Spatial Information Science*, 2013(6):59–115.

Roth, R. E. (2015). Interactivity and cartography: A contemporary perspective on user interface and user experience design from geospatial professionals. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 50(2):94–115.

Roth, R. E., Çöltekin, A., Delazari, L., Filho, H. F., Griffin, A., Hall, A., Korpi, J., Lokka, I., Mendonça, A., Ooms, K., and van Elzakker, C. P. (2017). User studies in cartography: opportunities for empirical research on interactive maps and visualizations. *International Journal of Cartography*, 3(sup1):61–89.

Rozhkovsky, R. and Bilchenko, A. (2018). Live ua map. https://liveuamap.com. Accessed: 2018-07-17.

Safarov, I., Meijer, A., and Grimmelikhuijsen, S. (2017). Utilization of open government data: A systematic literature review of types, conditions, effects and users. 22:1–24.

Sen, S., Swoap, A. B., Li, Q., Boatman, B., Dippenaar, I., Gold, R., Ngo, M., Pujol, S., Jackson, B., and Hecht, B. (2017). Cartograph: Unlocking spatial visualization through semantic enhancement. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, pages 179–190. ACM.

Smith, D. A. (2016). Online interactive thematic mapping: Applications and techniques for socio-economic research. *Computers, Environment and Urban Systems*, 57:106–117.

Sui, D. Z. (2004). Tobler's first law of geography: A big idea for a small world? *Annals of the Association of American Geographers*, 94(2):269–277.

The National Archives UK (2015). Open government license for public sector information. http://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/.

The New Inquiry (2018). https://whitecollar.thenewinquiry.com/, author=Whitecollar.thenewinquiry.com. Accessed: 2018-04-03.

theguardian (2011). Population breakdown by ethnicity interactive 2011. https://www.theguardian.com/news/datablog/interactive/2011/may/19/ethnic-breakdown-england-wales. Accessed: 2018-07-17.

theguardian (2016). eu-referendum. https://www.theguardian.com/politics/ng-interactive/2016/jun/23/eu-referendum-live-results-and-analysis. Accessed: 2018-07-17.

Vieira, V., Webster, T., Weinberg, J., and Aschengrau, A. (2008). Spatial-temporal analysis of breast cancer in upper cape cod, massachusetts. 7:46.

W3 Schools (2018). Node.js introduction. https://www.w3schools.com/nodejs/nodejs_intro.asp. Accessed: 2018-08-08.

whuber- gis.stackexchange.com (2017). Measuring accuracy of latitude and longitude? https://gis.stackexchange.com/questions/8650/measuring-accuracy-of-latitude-and-longitude. Accessed: 2018-08-08.

Ziegler, P. and Dittrich, K. R. (2007). Data integration—problems, approaches, and perspectives. In *Conceptual modelling in information systems engineering*, pages 39–58. Springer.