7CCSMDPJ

Individual Project Report 2017/18

Student Name: Luke William Ronald Whittaker

Student Number: 1763234

Project Title: London Traffic Matters

Degree Programme: Data Science

Date of Submission: 23rd August 2018 (Change to actual date, this is a guide)

Supervisor Name(s): Dr Rita Borgo

Word Count: XXXX

This dissertation is submitted for the degree of MSc in Data Science.

RELEASE OF PROJECT

Following the submission of your project, the Department would like to make it publicly available via the library electronic resources. You will retain copyright of the project.

I **agree** to the release of my project

I **do not** agree to the release of my project

Signature: Date:

Abstract

* The Abstract is a short, executive summary of your project. This should include a brief description of the project objectives and research question(s) addressed, followed by a brief description of the main contribution(s) of the project, including a summary of the results achieved and the primary conclusions drawn from the work. This should appear on a single page by itself and should be the second page of the report.

Acknowledgments

I wish to thank my project supervisor Dr Rita Borgo whose help and guidance was beneficial to the completion of this project.

Table of Contents

This is a list that includes and identifies the main sections, sub-sections and appendices (if any) of the report.  Page numbers must be included.

Nomenclature

1. Introduction 6

2. Background 7

2.1 What is Data Visualization 7

2.2 Visualizing Data on Maps 8

2.2.1 Mercator Projection ?

2.2.2 Azimuthal Projection ?

2.2.3 Conic Projection ?

2.3 Different Software Technologies ?

2.3.1 Python ?

2.3.2 JavaScript ?

2.3.3 WebStorm ?

2.3.4 XAMPP Server ?

2.3.5 Leaflet JavaScript ?

2.4 Bertin’s Visual Variables ?

2.5 Glyph Visualizations ?

3. Related Work 8

4. Approach 10

4.1 Data Cleaning ?

4.2 Question 1 ?

4.3 Question 2 ?

4.4 Question 3 ?

4.5 Question 4 ?

4.6 Question 5 ?

5. Results 11

6. Conclusion 12

7. References 13

8. Appendices 14

8.1. Appendix: Hints for Success 14

8.2. Appendix: Submission Details 16

8.3. Appendix: Plagiarism Warning 17

Glossary

This is a glossary defining all abbreviations and symbols used in the report. The contents should be listed in alphabetical order. Note that placing entries here is NOT a substitute for properly and completely defining terms in the body of the report, accompanying the first time a term is used.

CSS Cascading Style Sheets ?

CSV Comma Separated Value ?

DfT Department for Transport 6

DOM Document Object Model ?

GB Gigabyte 8 these will need to be changed

GIS Geographic Information System 8

HTML Hypertext Markup Language ?

IDE Integrated Development Environment ?

JSON JavaScript Object Notation ?

TB Terabyte 8

PC Pedal Cycles ?

2WMV 2 Wheeled Motor Vehicle ?

List of Figures and Tables

Background:

Figure X: Example of the Mercator Projection

Figure X: Example of the Azimuthal projection

Figure X: Example of the Conic projection

Related Work:

Approach:

Figure X: A snippet of the raw data for major roads

Figure X: Pseudo code for creating a dataset of London data from 2010 – 2016

Figure X: Shows a snippet of major road data for London boroughs

Figure X: Pseudo code for finding the yearly counts of vehicles

Figure X: Pseudo code for splitting major roads by borough and year

Figure X: Pseudo code for converting CSV file in to Geo CSV

Figure X: Pseudo code for converting Geo CSV to a JSON file

Figure X: Pseudo JavaScript code for displaying JSON data

Figure X: Pseudo code for finding the distribution of traffic

Figure X: Pseudo code for finding the most common vehicle per year

Results:

# Introduction

The Introduction is the first content section of your report. You should describe the general area (e.g., application domain) in which your project research is conducted, the motivation for conducting the research and the overall aims of the research. Be sure to outline your research questions and give a brief summary of the conclusions drawn, though the conclusions will be detailed later in the report. With the Introduction, you want to interest your reader and tell them why they should care about your research and why they should read the rest of the report. The report will be read (marked) by examiners with a technical Computer Science background, but not necessarily any knowledge of your domain, so make sure that you provide enough information for a naive reader.

# Background

Visualizing data is a vital process to undertake when exploring and aiming to bring insight to any format of data. With this in mind there are some aspects that should be taken in to consideration when exploring the data. As mentioned previously the data to be used for this project is traffic data provided by DfT. Thus, we must ensure that the data visualizations used to explore and analyse this data are appropriate for the tasks performed. The aspects that should be consider are: choosing the appropriate graph or visual encoding to bring meaning; this can take the form of the most appropriate graph to show the relationship between two variables, one having a numeric data type and the other taking a nominal data type; in this instance a Bar chart would be appropriate to show this relationship. Moreover, when it comes to deciding what map graphic to use when handling geo location data; we must look at the type of projection of the geology (this is be explored further in Visualizing Data on Maps) and the aesthetics of the map, this could be similar to Google’s satellite view, or the default view in Google Earth and Google Maps respectively.

In addition, the visual encoding used is of importance as this allows the reader to understand what the data means. Thus, if we were to have visual encodings that do not appropriately display the results then incorrect or misleading conclusions may be formed. Thus, this section of the report will look to explore in detail the theory behind data visualization, how we can visualise data through maps, technologies available to aid the creation of such visualizations, and advanced ways to enhance visual encodings that provide a greater level of insight to the data.

## What is Data Visualization

Within data analytics there are many sub-domains that exist to provide standardized ways in which to extract, transform, process, and display any format of data available. Data visualization is one sub-domain of data analytics; however, its main focus is how to best display data. With this in mind, we can deem it a vital domain of analytics due to its role for presenting data in such a way to bring instant understanding upon first sight. Moreover, with this visual representation of data, we are able to draw critical insight in to specific patterns unique to the data. Additionally, we are able to convey complex concepts in ways that enable readers of all levels to be able to gain insight to the data.

The concept of using visualizations to understand data has been used by humans to good effect for many centuries, whether this was the Romans using maps to navigate throughout the Roman Empire, to John Venn who inventing the Venn diagram that visualises symbolic logic. However, modern day visualizations still draw from past techniques, such as bar charts for visualising data with features having numeric and nominal data types. Further, bar charts can be enhanced through the use of colour to distinguish between the different nominal features. Additionally, line graphs are still well suited to visualizing temporal data – data recorded over a certain period of time – and many more visualizations. Moreover, with the technology age it brought with it the ability to take these visualizations to greater heights; this can be seen in the amount of data that can be processed, with modern computing we are able to process high volume of data, data which can range from 1-10s of GBs to many TBs of data.

Having this ability to visualize large amounts of data, for example stock market trend data visualized as a line graph, we are able to make quicker judgments on whether to sell, buy, or hold on to stocks. In contrast, if we were to view this data in a Microsoft Excel spread sheet, we might not be able to intuitively see what the trend might be at first glance. Moreover, such data representing the stock market would be obtained on a constant basis, which would result in the data held within this spreadsheet being time intensive task to scroll through to try and find patterns. In addition, handling data regarding Geographic Information Systems, more commonly known as GIS data, has been greatly enhanced with modern computing. This level of computing allows for processing of 10,000+ data regarding geographical locations, which leads on to the discussion regarding visualizing data on maps.

## Visualizing Data on Maps

Before we can discuss how we would go about visualizing data on to a map, we must first become familiar with what a map is and how they can differ from map to map. According to French cartographer Jacques Bertin a graphic can be deemed a geo-graphical map when all of the elements of the geo-graphical graphic are arranged upon a plane in such a way as they are observed on the surface of the earth [?].

Now that we understand what a map is, we can now explore the different types of projections a map can take. We need to use projections with regard to maps, this is due to the earth being represented as a flattened sphere. Thus, with the aid of projections we are able to unfold the whole of this sphere to allow it fit on to either a computer screen or paper. As such some features it looks to take in to account are the maps area, distance, bearing, scale, shape, and direction [?]. There are three main types of projections used these are Mercator, Azimuthal, and Conic projections.

### Mercator Projection

The Mercator Projection was created by Gerardus Mercator in 1959 and comprised a set of eighteen sheets, when put together creates a mosaic about 48inches tall and 80inches wide. With this way of projecting the earth’s surface sailors were able to greatly benefit, this was due to the earth’s bearings being represented as straight lines [?]. However, there is arguments to suggest that the Mercator projection should not be the universal projection taught; this is because the Mercator projection distorts the area distance each country has from the equator. For example, countries south of the equator look smaller than those north of the equator; such that Europe looks much larger compared to Africa despite Europe having a smaller population than Africa. With this level of distortion, we are not able to preserve angles and areas based upon their longitude and latitude values [?]. Below is a figure that looks to show this distortion, along with a representation of the bearing lines.

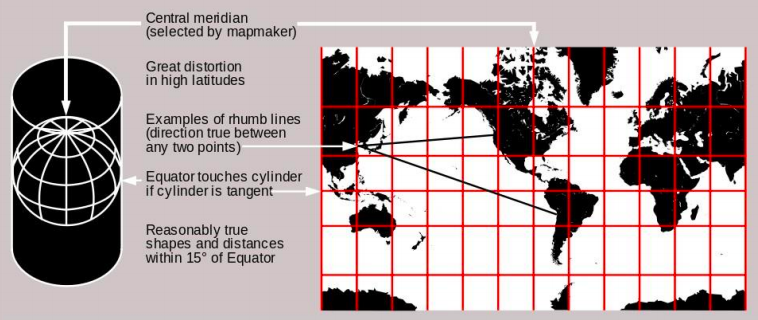


Figure X: Example of the Mercator Projection [?]

### Azimuthal Projection

The Azimuthal projection looks to unfold a map on to a plane tangent on the earth and place the whole map on to a single page, or computer screen. In contrast to the Mercator projection, the Azimuthal projection looks to accurately represent the area of all regions of the map. This results in angles close to the centre point of the map being correct; moreover, the distances increase linearly outward from the centre which shows the true distances of regions. The below figure shows a visual representation of the Azimuthal Projections [?].

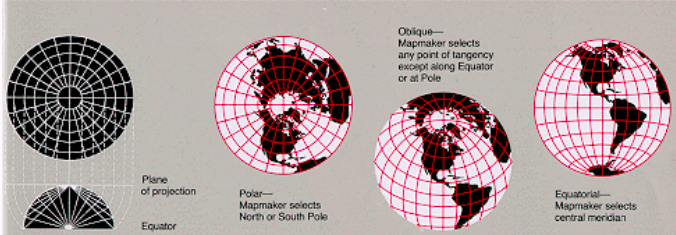


Figure X: Example of the Azimuthal projection [?]

### Conic Projection

A Conic projection looks to unfold a map upon cone that looks to separate the earth via its Latitude. This projection distorts the distances and areas of the earth. However, the Conic projection is best suited for viewing regions of the earth from East to West compared to viewing it from North to South [?]. The figure below shows an example of a Conic projection.

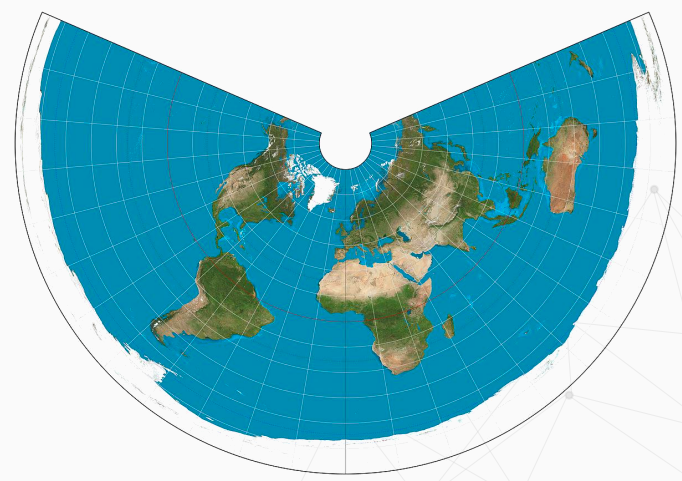


Figure X: Example of the Conic projection [?]

## Different Software Technologies

During the initial stages of this project, decisions needed to be made regarding the process and the technologies to be used throughout the creation of this project. Within this section discussions will be had exploring the pros and cons to each of the technologies used and explored throughout the completion of the project.

### Python

Before we can discuss what aspects of the project would be aided through the addition of Python, we must first look to understand the motivation and characteristics of the Python programming language. Python is one of many human-readable programming languages, more specifically called higher-level languages. In addition, Python programmes are processed through the Python interpreter; this way of program execution varies from many over languages, such as a language called C#, pronounced C-Sharp. C# is what known as a compiled because it is compiled in to machine language code before it is executed; we will speak more about this later.

Python was created by Dutch Computer Scientist Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands [3?]. Python takes a lot of its functionality from many languages, more specifically C and C++; we can see this through the Cython interpreter, which allows for C extensions to be built with the ease of writing Python programmes but with the speed of C.

Now that we know who created Python we can explore the fundamental philosophy behind its creation and see how it differs from other languages. Firstly, Python aims to ensure that function, class, or loop can be easily read by any reader; it is this level of readability that makes Python one of the best programming languages for beginners to learn. With this in mind we can take a look at the five philosophies of Python, which include:

* Beautiful code is better than ugly code
* Being explicit if much better than being implicit
* Simple code is better than complex code
* Complex code is better than complicated code
* Readability is important

Python looks to apply all of these philosophies by using English instead of symbols for logical operators [3?]. For example, in C like languages we would use “&&” for the logical operand AND; whereas in Python we use the English word “and”.

When comparing Python to other high-level programming languages we can see there are some distinct differences. These differences come in the form Python being a strongly typed and dynamically typed language. Python differs from a language such as C, which is a weakly typed language [3?]. For example, in C we can have the following legal code:

byte foo = 50;

int f = foo;

The reason the above is valid, is because the C compiler will make the type conversion from a byte to an integer for us. However, if we were to perform the following in Python it would result in an error message appearing:

foo = 50

f = “40”

print foo + f

This would not be accepted by the Python interpreter because we are looking to add a number and a string together, in order to successfully execute this, we would need to convert the string in to a number.

As mentioned above, Python is a dynamically typed language, this means that the Python interpreter does not know what data type each variable has before the whole program is executed. This allows for some level of freedom with variables; however, it does mean that all variables must be instantiated when created. In contrast, C# is a compiled language, this means that C# programmes must be translated in to machine code, this is a low-level programming language, before it is executed. One benefit compiled languages have over dynamic languages is that the compiler knows the data type of each variable before execution, thus allows for optimizations to be made to improve performance [4?].

Now that we have a good understanding of what Python is as a programming language and discussed its characteristics. We can now mention how Python will be used throughout this project. Python will be used extensively throughout the data pre-processing and data curation process. The reason being that Python has libraries that aid these tasks, these include: csv, and json which aid in handling such data, and pandas which will aid in manipulating the curated data. Python was also chosen for this project due to its ease of use which will enhance the progression of the project, along with it being cross platform which meant that curation of the data would not be limited to a certain hardware.

### JavaScript

JavaScript is another example of a high-level programming, unlike Python its main functionality is to enable a web browser to host interactive web pages. As such, search boxes, interactive sections of a webpage, and having the ability to watch a video within a web browser; it is accurate to assume that they were created with JavaScript. JavaScript was created by Brendan Eich whilst working at Netscape Communications Corporation [1?]. Although, JavaScript is considered a high-level programming language, it is different to many other programming languages in one profound way. As we have mentioned before programming languages can be either compiled, dynamically typed, weakly or strongly typed. JavaScript is considered to be weakly typed because it does not worry too much about a variables type. However, it is considered to be statically type, such that we can declare a variable without having to instantiate it.

JavaScript, unlike many other programming languages, is known as a text-based programming language; JavaScript is also considered to be one of the three pillows of web development alongside HTML and CSS [2?]. Moreover, one of the key features to JavaScript is the ability to manipulate the Document Object Model, more commonly known as the DOM. The DOM allows for JavaScript to manipulate HTML tags such as <p> (paragraph tag) and many more; it also enables JavaScript to handle events such as when a user clicks on a certain button on the web page [5?].

JavaScript was chosen as one of the programming languages for this project; due to its ease of use, in the sense of its syntax (prior experience with programming languages like JavaScript), and that JavaScript is well suited to work on the web browser; which will aid in the creation of the web page that will look to visualise the traffic data provided by DfT. In addition, there are many JavaScript libraries that are well suited for data visualization, one of which we will discuss later in this section.

### WebStorm

WebStorm is a fully functioning Integrated Development Environment, or IDE, that allows programmers to develop modern programmes for any web browser, whilst utilising the JavaScript ecosystem. In the context of this project there were both advantages and disadvantages to using WebStorm. The advantages of using WebStorm were: we were able to enjoy its functionality of code completion, this aids development by ensuring variable names, classes, and functions are properly spelt – this prevents case sensitive errors occurring – and allows us to see what functions are available within a class with needed to locate the class to find out what functions are available. WebStorm also provides up-to-date error detection, this allows us to see where possible error might occur when typing code [?].

Moreover, WebStorm provides an integrated debugger; a debugger allows us to place break points – locations within our code where we wish the program to stop when executed – within our files. When these break points have been reached, we are able to take incremental steps throughout the program to see what logic is truly happening. As such, we are able to properly investigate errors when they occur and have the knowledge to fix them. Finally, WebStorm allows us to host programs on a server, which in turn prevents modern browsers breaking their security protocols [?].

However, regarding this project when using WebStorm the server capability was unable to be used for an unknown reason. As a result, an alternative way to host the project through the web browser was found, this server is called XAMPP which is discussed below.

### XAMPP Server

As a result of the server associated with the JavaScript IDE WebStorm not being able to successfully work with this project. After exploring other alternatives, a free open source server was found, this server is called XAMPP. The acronym XAMPP has the following meaning [?]:

X – Means the server is cross-platform, i.e. works for Windows, Linux, and Mac OSX

A – Means the server is associated with the Apache distribution, i.e. an Apache server

M – Stands for MariaDB, the database used to be a MySQL database

P – Stands for PHP

P – Stands for Perl

Due to XAMPP being open source, this means that there is not fee associated with running either the server for a database or the Apache server [?]. Moreover, this project will look to utilise the Apache server when running the program through a modern web browser – the modern web browser used was Google Chrome – which will enable the program to access local files; a functionality that web browsers prevent due to security protocols. Further, XAMPP is easy to install due to its open source and cross-platform capabilities.

### Leaflet JavaScript

As mentioned above, there are many JavaScript libraries that are available when it comes to visualizing data for web browsers. Further, in order to find the most appropriate library to use for displaying our London traffic data, the library must be able to handle GIS data, along with utilising map visuals – this can be similar to what Google Earth looks like. As such, Leaflet JavaScript, otherwise known simply as Leaflet, was chosen because it provides all of this functionality among others [?].

We may ask, what is Leaflet? Leaflet is an open source JavaScript library that can create interactive maps for the web browser, whilst being mobile friendly. Leaflet is also a very light weight library, with the file being 38KB of JavaScript code, which allows for web browsers to loads this in no time. Further, Leaflet has a combination of simplicity, performance, and usability that makes development with it rapid; it can also be extended with plugins that enhance its features whether that be in regard to how data is displayed, or the map image used [?].

## Bertin’s Visual Variables

In 1967 a French cartographer called Jacques Bertin published a research paper called Semiology of Graphics. This research laid down the foundations of how we as humans construct and interpret visual information [?]. As such, Bertin found that humans interpret the following six features when looking a visual, whether this be on a page or computer screen. These six features are more commonly known as Bertin’s visual variables and they include:

* Shape
* Size
* Value
* Texture
* Colour
* Orientation
* Two Planar Dimensions

By combining these visual variables, we are able to make visuals capable of delivering insight to any format of data [?]. These variables help to bring insight to a visual because they look to bring different dimensions to them. For example, shape can be used to distinguish between different population sizes, such that the bigger the visual the greater the population. Shape can be used to distinguish between multiple classes; colour can also perform this function too.

The orientation of a visual could be how it is positioned, for instance one visual could be placed at a 90-degree angle and another one might be placed horizontally. Value can show the level of a visual; for example, we could use value to show the difference between employees and the amount they sell, with the darker the visual the more they have sold. Whereas texture could be used to distinguish between visuals that have the potential to overlap if they were displayed in the same way.

## Glyph Visualizations

In modern times a glyph is used to visualise a multivariate visualization, aiming to bring information regarding multiple features in to a single visual that provides meaningful insights [?]. Moreover, according to Borgo [?] glyphs can be categorised in to two interpretation, these being narrow and broad. Narrow interpretation states that a glyph should be small in size and represent features within a data record [?]. Moreover, these glyphs should be placed in the most appropriate location within the display space. Further, the design of the glyph should ensure they do not resemble visuals such as icons or symbols [?].

In contrast, a broad interpretation of a glyph is a small visual that represents features of a data record, or a group of features from a data record. Moreover, the glyphs can be places on to the display space independently, or they can hold a relationship with other glyphs from this space. An example of this could be a glyph representation of a network graph. However, similar to the narrow interpretation, these glyphs should not resemble visuals such as icons or symbols.

# Related Work

# Approach

The approach to this project was to explore the public dataset provided by DfT [?], and devise questions that would act as a guide to produce insightful meanings about the level of traffic throughout London. With this in mind this project had the aim of answering the following research questions:

1. Has the number of vehicles travelling through London increased or decreased between 2010 to 2016?
2. How is London traffic affected by sporting events?
3. Is there a uniform distribution of traffic, or do some London boroughs experience greater levels of traffic?
4. What is the most common type of vehicle that travels through a London borough, and does this change from year to year?
5. What type of visual encoding works best for exploring traffic data?

Before we can delve in to how each question was approached and their outcomes, we must first spend some time understand the initial data provided by DfT, and how this data was transformed in to datasets that allow for these questions to be answered.

The data provided by DfT came in the form of multiple comma separated value file, more commonly known as CSV files. Of said files the two files that were of importance to this project were Raw-count-data-major-roads.csv and Raw-count-data-minor-roads.csv. It is worth noting that both CSV files are similar, thus by describing one file that explanation refers to the other. With is in mind we will explore the raw data for major roads. Below is a example of the raw data for major roads.

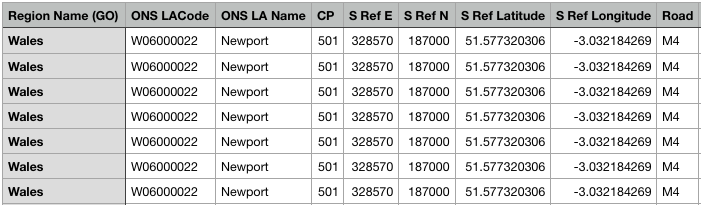


Figure X: A snippet of the raw data for major roads.

As we can see from the figure X we are provided with a lot of data about each record within the dataset. For example, we can see that the data is currently showing entries from Newport in Wales and gives details about their latitudes and longitudes among others. Further, this shows that the current state of the data is not ideal to start answering our research questions.

This leads on to the stage of initial exploration of the data and data cleaning. For this step, Python was used, more specifically Pandas which allows us to read the data about major roads in to a table like structure and provides functionality for manipulating data with ease. On first inspection it was noted that major roads file was 347.5MBs in size; this equates to 1,752,356 data records and 33 features. The reason for such a large data file is due to many records being associated with regions of the United Kingdom that are not located within London; for instance: Wales, Scotland, and South West England to name a few. Moreover, we can see that the data also holds metadata regarding latitude and longitude values, this will help with plotting the data on to the map that will feature in the web page. In addition, we also have data regarding all of the types of vehicles ranging from cars, buses, Light Goods Vehicles (LGV), and many more.

## Data Cleaning

The data cleaning aspect to this project is vital to obtaining meaning and accurate results. With this in mind we can now discuss the process taken to create the three data formats which include, cleaned CSV file containing records only associated with London, a CSV file separating all data for each borough in to separate years, a Geo CSV file that contains summaries of all vehicles along with their latitude and longitude values; and finally, a JavaScript Object Notation, or JSON, file that contains all of the information within the Geo CSV file. The reason for the creation of the JSON file is due the ease Leaflet, and in turn the projects web page, works with data structured in JSON.

First, we must discuss how we obtained a dataset containing all records associated only with London. Again, it is worth noting that this section and subsequence sections to follow, will discuss how this was achieved using the major roads data; reason being that identical steps executed on to the minor roads data. Below is a figure showing pseudo code to explain the steps taken.

Read in CSV file “Raw-count-data-major-roads.csv”

Take a look at the head of the data

Look at the unique Regions that exist in the data

Remove regions that are not in London, using Pandas groupby function

Create a list of the values that return true

Create a new Pandas DataFrame with the list of true values

# Now we can remove records with a year 2000 – 2009

Group records within the London DataFrame by years > 2009

Create a list of values that returned true

Create a new Pandas DataFrame with the new data

Check that we only have data from 2010 – 2016

Save major roads data frame to a CSV file

Figure X: Pseudo code for creating a dataset of London data from 2010 - 2016

After performing this initial cleaning of the data, we arrive at a data set that has records for all of the London boroughs between the years of 2010 to 2016. The reason for choosing the period of 2010 to 2016 was due to the 2012 Olympics that was staged in London. With this period, we are hopeful to find an insight in to the level of traffic through the boroughs that staged aspects of the sporting event. Moreover, we are able to retain the vital metadata held for each borough, which will be used for further analysis. Below is a figure showing a section of the refactored London borough dataset for major roads.

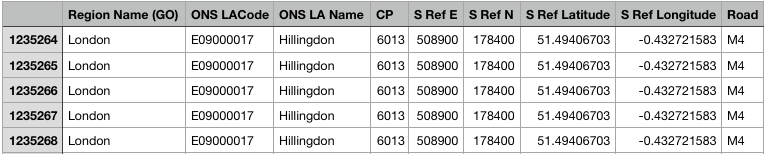


Figure X: Shows a snippet of major road data for London boroughs

Now that we have access to all of the data regarding London boroughs, discussions can now turn to the research questions addressed above, along with how the London borough data set was transformed further to aid in the research.

## Question 1

The first research question this project looked to explore was: Has the number of vehicles travelling through London increased or decreased between 2010 to 2016? With this question we are interested in looking at the total number of vehicles that travel through London each year from 2010 to 2016. By examining each year, we are able determine whether the is a positive or negative trend towards the level of vehicles.

For this research question we are able to use the newly created major roads file discussed above. The reason we can use this file is because we have access to the metadata for the feature of interest, these features include: PC (pedal cycles), 2WMV (motor cycles), cars, buses, LGV, HGV*.* Moreover, we can use this file because we are not interested in values for individual boroughs, but as counts of vehicles for all boroughs. The following figure shows pseudo code for finding the yearly counts of vehicles from 2010 to 2016.

Read in Major\_Roads CSV file

Look at the head of the file

Create a list of years and an empty list yr\_cnts to hold the values

Def yearlyCounts(data frame t\_df, list of years):

For y in years:

Create a data frame using the groupby function for y

Create a list of values that returned true

Create a new data frame with the true values

Find counts for PC, 2WMV, car, bus, LGV, and HGV

# for debugging

Print the counts

Save to a variable the total counts

Add total counts to the yr\_cnts list

Plot the data using Matplotlib

Figure X: Pseudo code for finding the yearly counts of vehicles

In order to find the yearly counts for each vehicle, the following steps were taken. First, Pandas was used to read the major roads CSV file, which in turns stores the data in table like structure. Then, two lists were instantiated. The first to be created is a list of years we wish to find counts for, this list will also act as the main object for the loop within the programs yearlyCounts function. The second list which holds the total counts for each year is set to an empty list, this is due to no data being available at the time of creation.

The next step is to create a function that takes a data frame and the list of years as parameters. Then, we can loop through each year of the list of years and creating a data frame for each year. Once, this data frame has been created we can count all of the number of vehicles for PC, 2WMV, cars, buses, LGV, and HGV. After these values have been obtained we can simply add them together and save the result in to a variable and then store this value in to the yr\_cnts list. When the loop has successfully looped through each year and added all the totals to our list, we can look to display this data on to a graph. The graphs for this project were created with the aid of a Python package called Matplotlib. Please see the Results section for an analysis of the results for this question and the following questions.

## Question 2

The second research question for this project is: How is London traffic affected by sporting events? To answer this question a web page was created that looks to display the GIS points for each record within the dataset. Moreover, the data points to be displayed on the web page will include a text box that will appear once clicked; the aim of this text box is to provide tailored information about each point on the web page.

The first step to complete for this question is to split the major roads data in to multiple files that contain data for a single year for each borough. For example, for the borough Barnet there are CSV files for 2010 to 2016 and so on for all of the boroughs. The following pseudo shows how the separation of these files was done.

Read in the Major\_Roads.csv file

Create a DataFrame and include only features we require

# Start with Hillingdon

# 2010

Create a DataFrame using groupby for Hillingdon

Create a list of the values that returned true

Then place those values in to a Pandas DataFrame

Then create a DataFrame for 2010 from the Hillingdon DataFrame

Create a list of true values

Then place them in to a DataFrame

Save data for Hillingdon in 2010 to a CSV file in the /Major\_Roads/Hillingdon as Year2010.csv

# Repeat for years 2011 to 2016, then 2010 – 2016 for each of the other boroughs

Figure X: Pseudo code for splitting major roads by borough and year

The first task is to use Pandas to read in the major roads data. Once we have access to this data, we are able to create a data frame that performs an initial level of feature selections. The feature selection aspect was to remove data that would not enhance the readers understanding of the data, along with multiple features that can be consider under one feature. For example, within the data there existed multiple features regarding HGV vehicles, the difference was to specify the number of axels difference HGV vehicles have. Moreover, after looking through the data description report provided by DfT [?], we were able to see that the feature HGV was a total count of all HGV vehicles, as such the features regarding difference axels could be removed.

After the creation of the data frame, which excludes all of the unrequired features, we looked to create a data frame for the borough Hillingdon. Then, we looked to group all of the data regarding 2010 from Hillingdon in to another data frame. When this was successful, we simply saved the data frame as a CSV file in the directory /Major\_Roads/Borough\_Name/Year.csv. This was then repeated for all years and boroughs.

However, the data is still not ready to be used for the web page in its current state. Further, transformation to the data was performed. The next step was to convert the CSV data we have just created in to a Geo CSV file. A Geo CSV is like any normal CSV file, with the only difference being that a Geo CSV file include latitude and longitude values. The data held within these Geo CSV files initially were to be what was displayed on to the web page, but we will discuss this soon. The Geo CSV files will be structure in a different way to previous CSV files. The reason for this difference is due to the original data holding multiple records for one pair of latitude and longitude values. Thus, we looked to produce summary values for all of the vehicles associated to one pair of latitude and longitude values. Below shows pseudo code for how the Geo CSV files were created.

# For debugging purposes

Check to see if we have different longitude values for one latitude values

# Create a function for when we only one value for latitude and longitude

def averagesForOneGPSValue(borough, year, road\_type):

read in data of borough

find the first latitude value of the list of latitudes

define a list of titles for the CSV file

with open(file path to save data) :

write titles to CSV file

find average values and values for

borough, pc, 2wmv, car, bus, lgv, hgv, road, road\_category, direction,

year, day count, and longitude

add all of the data to a list, then write it to the CSV file

def vehicleAverages(borough, year, road\_type):

read in data for borough

get a list of the latitudes

define a list of titles

with open(file path to save data):

write titles to CSV file

for each lat in list of latitudes:

create a data frame for lat

find average values and values for

borough, pc, 2wmv, car, bus, lgv, hgv, road, road\_category, direction,

year, day count, and longitude

add all the data to a list and then write it to the CSV file

# We would call the functions like so, E.g.

vehicleAverages(“Barnet”, “2010”, “Major\_Roads/”)

Figure X: Pseudo code for converting CSV file in to Geo CSV

The reason for creating two functions for the purpose of converting the data in to a Geo CSV file, was due to some of the data files only containing multiple records for one GIS location. Moreover, after close inspection of the latitude and longitude values, it was noted that one latitude value is associated to only one longitude value. This meant that if we had more than one latitude value we could simply loop through a list of latitude values. Which in turn allows for creating a data frame to calculate the average values for all of the vehicles, along with finding the values for other metadata that would be of use. Once all of the Geo CSV files had been created, the next step was to try and display this data on to the web page. However, when this step was tried errors were occurring with the MarkerCluster plugin that extends the functionality of Leaflet. The MarkerCluster plugin looks to group nearby markers into a single visual marker, but then expands once clicked to show the markers within that cluster. However, in light is this issue and its inability to be solved, the Geo CSV file was later converted in to a JSON file. Below shows the pseudo code for converting the Geo CSV file in to a JSON file.

# Function for reading in CSV files

def read\_csv(borough, year, roadType):

instantiate an empty list to hold the rows within the CSV file

with open(file path to CSVdata):

read the titles for the CSV file

for each row in the CSV file:

add data as a dictionary format like: title: value

send data to write\_json function

# Function for writing data to json file

def write\_json(data, roadType, borough, year):

with open(file path to save data):

write data with json.dumps

# For debugging read in a JSON file and view the contents

# We can run the functions like so

read\_csv(“Barnet”, “2010”, “Major\_Roads/”)

Figure X: Pseudo code for converting Geo CSV to a JSON file

When running the functions in figure X, we are able to successfully convert the Geo CSV files in to JSON. The way it works is by reading each row of the CSV file and converting it in to a list of dictionary objects. For example, is we have the following titles [“Hello”, “World”] and a list of value values [“1”, “2”]; when converting these to dictionary object they would become: [{“Hello”: “1”}, {“World”: “2”}]. Moreover, once the data had been converted to JSON we looked to display this data on to the web page, this time with success. The following pseudo code shows how this was completed.

Create a map variable to store details of the map, ensure its centre is pointed to London

Add a tile layer to the map

// Function to display data on to the map

Function getUserRequestedData()

{

Get values from the borough, year, and road type drop boxes the user selects

Perform an AJAX call to the local file system to read the JSON data

$.ajax({

type: “GET”,

url: “the data we want to display”,

dataType: “json”,

error: function for when an error is caused {}

success: function(our data)

{

Send data to displayData function with our data as an argument

}

});

function displayData(our json data)

{

// debugging print data to console to see if we read it in correctly

for each item in our JSON data

{

// Create a Leaflet marker

L.marker([latitude and longitude values for each item

using bindPopup use CSS to construct the information that will

be displayed to the use.

]).addTo(map)

}

}

}

Figure X: Pseudo JavaScript code for displaying JSON data

The above figure shows the pseudo JavaScript for displaying our JSON data. The first step is to create a variable that contains all of the map details, this include where its centre should be located, the minimum zoom allowed, and the initial zoom of the map. Then, we can add a tile layer which will show the map tile the data will be placed upon. Once, the map tile has been successfully loaded on to the web page, we can now begin to handle the users request to display the data.

Located on the web page are three drop down boxes for the London borough, years, and the road type. When the user selects from the three drop boxes and presses the submit button, we instantly process an AJAX call to the systems local files to search in the project file space for a file contain the information the user has requested. From here the program can perform one of two tasks; if there is no data then the program will send an alert to the user prompting them that they have requested data that does not exist. However, if the data does exist, then all of the JSON data is passed to the displayData function. The role of displayData is to loop though each JSON object and plot them on to the map tile, along with a popup box constructed using CSS that will display all of the metadata about that GIS location. The results to this research question can be seen with the Results section.

## Question 3

The third question this project looks to answer is: Is there a uniform distribution of traffic, or do some London boroughs experience greater levels of traffic? With this research question, we aim to discover what areas of London, if any, experience a greater level of traffic flow compared to other boroughs. By comparing the distribution of traffic, we can identify the borough, or boroughs that may need to reduce the number of traffic flow. Moreover, as we know vehicles, excluding bicycles, exert a large volume of toxic gases through the combustion of petrol or diesel. The results of this analysis along with those composed for research question 4; we can identify the London borough that experience the greatest level of traffic and compare to see if there is a potential for this borough to exert a large volume of these toxic gases.

To answer this question, we are able to utilize the major roads CSV file; we are able to work with this file because we are not concerned with separating the data by individual years. We simply want to find the boroughs with the highest volume of traffic. With this in mind we can group the data by borough and calculate the total number of vehicles recorded from 2010 to 2016. The following pseudo code shows how the distribution of traffic was calculated.

Figure X: Pseudo code for finding the distribution of traffic

First create an empty list to hold the total counts

Create two lists of boroughs, one for the function, the other to be used for the graph

# Create a function to calculate total vehicle counts for each borough

def vehicleDistribution(list of boroughs b, road\_type):

read in the data for major roads

for each b in boroughs:

create a data frame of b

find the sum value for pc, 2wmv, car, bus, lgv, and hgv

add all of the sums to obtain a total

add the total to the total counts list

Use matplotlib to create a bar chart of the results

The pseudo code above shows the steps taken to find the overall distribution of traffic throughout London. As such, the first steps taken were to create three list structures that will be used throughout the calculations. The first list, total counts is used to store the value calculated regarding the vehicles for each borough. The next two lists are similar, in that they are lists of borough names, however their roles differ. One list will contain abbreviations of all the borough, this allows the graph to easier to read, the other list contains the full name of each borough and will be used as the main object that the loop inside the functions accesses. Next, we look to access the major and by looping the borough list, we can create a temporary data frame to store all of the metadata regarding that borough; this will allow us to calculate the number of vehicles travelled through a borough between 2010 and 2016. Once, the values have been obtained and added to the total counts list, we can then look to create an appropriate graph to display the data. For this data a bar chart was chosen, the reason being that we are dealing with one numerical value (vehicle counts) and one nominal data (name of boroughs). Further analysis and results can be seen in the results section.

## Question 4

The fourth questions this project looked to answer is: What is the most common type of vehicle that travels through a London borough, and does this change from year to year? Following on from the brief mention about the toxic fumes exerted by motor vehicles from Question 3. Research in to this question is very important as it would provide a vital insight in to the areas of London that both experience a high level of traffic, along with a record of the types of vehicles travelling through them. For example, if it was noted that the London borough of Camden experienced the greatest level of traffic, we could explore what type of vehicles are contributing to this level of traffic. Moreover, if after exploration it was found that there were a large volume of cars travelling through Camden, we could devise strategies in order to reduce this level. As we are analysing the records over the period between 2010 to 2016, we can see whether or not the level has increased or decreased; which will show whether or not more effects need to be made to reduce the level further from 2017 onwards. Below is pseudo code for calculating the most common vehicles for each year.

First create a list of London boroughs

Create a list to hold vehicle counts

Create a list of vehicles for the x axis of the bar chart

Create a list of years

# Create a function to find most common vehicle per year for each borough

def mostCommonVehicle(road\_type, list of boroughs b, list of years):

for each b in boroughs:

for each y in years:

Read in data

Find counts for pc, 2wmv, car, bus, lgv, hgv

Add them to vehicle counts list

Using matplotlib create a bar chart, change the colour of each vehicle

Save image

Reset the vehicle counts list to be empty ready for the next borough

# We call the function like so

mostCommonVehicles(“Major\_Roads/”)

mostCommonVehicles(“Minor\_Roads/”)

Figure X: Pseudo code for finding the most common vehicle per year

The first steps to take in order to answer this question, are to create a few lists that will contain borough names (this list tells the loop which borough to work with), a structure to hold the vehicle counts, a list of headings for the graph, and a list of years. Next, we need to read in the data regarding the current borough we are working with. After the data has been successfully imported we can find the vehicle counts and add them to the vehicle counts list. Once this has been completed we can create a bar chart for the current data. It is important to remember to empty the vehicle list once we have finished, if we do not empty the list we will obtain a result that is incorrect because we will have data regarding the current data and the previous data. Results and further analysis can be seen in the Results section.

## Question 5

What type of visual encoding works best for exploring traffic data?

# Results

The Results section of your report basically contains the answers to your research questions. This section should present the results of your evaluation, provided quantitatively, qualitatively and/or visually, as appropriate, followed by an analysis of the results. If you have performed experiments and/or analysis, then these should be presented here. Use figures and drawings to explain the significance of your results.

Discuss with your project supervisor(s) and/or domain advisor(s) how best to present your results. The main point is to make sure that it is clear to the reader what the answers to your research questions are and how you arrived at these answers.

# Conclusion

The Conclusion is the last section of your report (other than Appendixes). In this section, you can revisit the research questions and summarise your answers. Clearly explain how your investigation and your answers are a contribution---why your work is worthy of a passing mark. Also in the Conclusion section, it is good to have subsections that highlight (a) Future Work, in case you were going to keep working on the same line of research or you wanted to recommend follow-up investigation for another student to pursue next year; and (b) Lessons Learned, where you can explain how you might do things differently if you started over, because you've learned valuable things along the way (these could be technical, but they could also be personal, such as organising your time better or listening to the project coordinator who told you to BACK UP your work frequently).

# References

This is your bibliography. It is a list of source materials that you have mentioned in your report, including articles, books, web sites, software tools and libraries, data sets, videos and other items you have drawn upon to conduct your research. Any material mentioned in the report that is not your own work must be given a reference.  All references must contain the following details: authors, title of the work, sources (journal title, conference title, publisher (for books)), date of publication, and page numbers (except for online sources). General guidelines for reference formatting can be found in the **Resources** section of this KEATS page.

1. Agafonkin, V. (2017) Leaflet. [Online] <https://leafletjs.com/index.html>. [Accessed 9/08/2018].
2. Apache Friends. 2018. What is XAMPP?. [Online] <https://www.apachefriends.org/index.html>. [Accessed 9/08/2018].
3. Bertin, J. Semiology of Graphics: translated by William J. Berg. Esri Press 2011
4. Borgo, R*.* 2018 Data Visualization: Maps [Presentation] <https://keats.kcl.ac.uk/pluginfile.php/2517528/mod_resource/content/2/7CCSMSDV_Vis_Lec07_Maps.pdf>. [Accessed 9/08/2018].
5. Borgo, R., Kehrer, J., Chung, D.H.S, Maguire, E., Laramee, R.S., Hauser, H., Ward, M., & Chen, M. 2013 Glyph-based Visualization: Foundations, Design Guidelines, Techniques and Applications.
6. Chapman, S. (2017) More facts about JavaScript. [Online] <https://www.thoughtco.com/a-brief-history-of-javascript-2037675>. [Accessed 8/08/2018].
7. Chapman, S. (2018) Introduction to JavaScript. [Online] <https://www.thoughtco.com/what-is-javascript-2037921>. [Accessed 8/08/2018].
8. Depart for Transport: Road Traffic Statistics [PDF] <http://data.dft.gov.uk/gb-traffic-matrix/traffic-majorroads-miles-metadata.pdf>. [Accessed 15/07/2018]*.*
9. JetBrains. (2018) WebStorm. [Online] <https://www.jetbrains.com/webstorm/>. [Accessed 9/08/2018].
10. Lafore, R. Microsoft C Programming for IBM. Howard W. Sams & Company. 1988
11. McGrath, M. Python in Easy Steps. In Easy Steps. 2015
12. Monmonier, M. Rhumb lines and map wars: a social history of the Mercator projection. The University of Chicago Press. 2004. [Online] <https://books.google.co.uk/books?hl=en&lr=&id=nvwu4Ba_Qp0C&oi=fnd&pg=PR7&dq=mercator+projection&ots=f2jiVVNFY5&sig=JxGgJUCfDOeXUnscfLYqFaxsWJ0#v=onepage&q=mercator%20projection&f=false>.
13. W3schools. (2018) JavaScript HTML DOM. [Online] <https://www.w3schools.com/js/js_htmldom.asp>. [Accessed 8/08/2018].

# Appendices

Supplementary materials may be included, such as additional tables and figures that would detract from the narrative if you included them in the main sections, above. Each appendix must be labelled (for example, Appendix A, Appendix B).  All Appendices must be referred to somewhere in the text.

## Appendix: Hints for Success

* Start by creating an outline of the report, which gives you an overall structure of the report.
* Think of the text of your report as a sandwich: the "bread" is the introduction and the conclusion. Start writing the "meat" first--the inside of your sandwich. Write about your results. Then write about the methodology you used to achieve these results, the software you wrote, the libraries you integrated and the data set(s) that you explored. Write your introduction and conclusion last!
* Show understanding of the topic and demonstrate the contribution of the work. At least 70% of the content of the report must be your own contributions and achievements.
* Always use your own words.
* The main report and any appendices must constitute one PDF document.
* Pages must be numbered consecutively.
* Captions must be provided for all figures and tables.
* For graphs, all axes and units must be labeled (in a font large enough to be read--a good guideline is that no label in a figure should be smaller than the font in the body of the report, even when the figure is included in the report; sometimes you have to generate the PDF in order to make sure this is the case).
* Equations (or important equations), figures and tables must be numbered.
* All figures and tables must be referred to in the text.
* Units of all variables must be provided.
* Numerical values (floating-point numbers) should be displayed with appropriate precision (e.g., 2 decimal places for currency, more or less as appropriate).
* Contractions ~~shouldn't~~ should not be used.
* Check punctuation of sentences. In particular, those sentences with equations. For example, if an equation is at the end of a sentence, a full stop should be used. If sentences are comprised of multiple clauses, use commas (,) and semi-colons (;) as appropriate, in order to help the reader understand what you are trying to say.
* All variables must be defined.
* Font face of variables throughout the report (in the text, equation, figures and table) must be consistent.
* Use proper headings for chapters, sections, subsections.
* Chapters, sections, subsections should be numbered, and the same numbering system should be used throughout the report.
* It is suggested that vector and matrix variables should be in **bold** and scalar variables should be in italics.
* Terms and abbreviations should be written in italics and defined the first time they are used.
* References must be used for text quoted in the report that is not yours, as well as software and other materials (e.g., images) that you did not generate yourself from scratch.
* A standard reference format must be adopted and be consistently applied throughout the report.  General guidelines for reference format can be found here.
* Always back up your files!!

## Appendix: Submission Details

Report:

* TheFinal Report (Dissertation) must be submitted electronically via KEATS by the 4pm deadline on the due date.
* You should make sure in advance that you can upload your report so that there are no last-minute glitches. You can upload multiple times. The final version uploaded will be the one marked (all uploads over-write any previous uploads).
* Submit the report as a PDF file. There are various ways to convert .doc/.docx files into PDF. For LaTeX users, pdflatex automatically produces PDF files of good quality.
* Do not send your final report to your supervisor directly.

Source Code:

* All work on source code must stop once the code is submitted.
* Keep a working version of your source code that you can demonstrate during the Oral Presentation.
* Your examiners may ask to see the last-modified dates of your program files, and may ask you to show that the program files used in the project examination are identical to the program files submitted with your project.
* Any attempt to demonstrate code that was not included in your submitted source listings is an attempt to cheat.Any such attempt will be reported to the KCL Misconduct Committee.

## Appendix: Plagiarism Warning

IMPORTANT NOTICE:

Given the importance of the project in the degree programme, the penalties for plagiarising project work are especially severe (and include the possibility of permanent exclusion from the College with no possibility of receiving a KCL degree). Our Department has considerable expertise in detecting plagiarised work. As a student at King's, you will have read the College's statement on plagiarism and you will have already signed a declaration to state that you understand the term and agree to abide by the statement on plagiarism. If you require further explanation of the College's policy on plagiarism, then please ask your supervisor for guidance.

**Plagiarism** is when you use someone else's work without acknowledgement, which may include concepts, design, ideas, a piece of program code, a section of text, diagrams, figures, approaches, methods, results, techniques, etc.  All materials, works or contributions that are not your own must be acknowledged, using correct citation procedures (see the **Skills Training** section of the KEATS page for more information).