UCL

Final Coursework

CASA0003: Digital Visualisation
Centre for Advanced Spatial Analysis
Term 3, 2019

Haien TAN
Shuke LEI
Kristian LUNOW NIELSEN
John HOOPES IV

5591 words

Project Output Description Table:

Project Output	Output Description
Project Output Files	Zip File on Moodle
Presentation slides	Powerpoint included in Output Files / Available at: https://docs.google.com/presentation/d/1N4t2aggPv9KiGh8QDM_EbhpaFxiMtaP_MIu4j-z0aOk/edit#slide=id.p
Project Website	Available at: https://robisoniv.github.io/casa-digital-visualisation/ui/

Individual Contributions Outline Table

Task Name	Major Contributors	Additional Contributors	Relevant Chapters
Concept Development	All team members		
Methodology	John Hoopes IV		Chapter 2
UI/UX	John Hoopes IV	Kristian Lunow Nielsen	Chapter 2
Data Preparation and integration	Kristian Lunow Nielsen	Shuke Lei	Chapter 3
Design	Shuke Lei	Haien Tan	Chapter 4,5,6

Global Visualisation	Haien Tan	Kristian Lunow Nielsen, John Hoopes IV	Chapter 4
National Visualisation	Shuke Lei, Haien Tan		Chapter 5
Local Visualisation	Haien Tan, Shuke Lei		Chapter 6
Development	Kristian Lunow Nielsen John Hoopes IV	John Hoopes IV, Shuke Lei Kristian Lunow Nielsen, Shuke Lei	Chapter 7.1/7.2 Chapter 7.3
Future Work	John Hoopes IV	Kristian Lunow Nielsen	Chapter 8
Conclusion	All team member		Chapter 9

1. Introduction

Maps represent objects as they relate to each other in space. By providing humans with a visual abstraction of some territory, maps enable people to understand the topological relationships between features in their environment, thereby empowering map readers to more knowledgeably interact with the world.

The digital era has resulted in an explosion in both the availability of mappable data and the sophistication of maps in their ability to convey depth of meaning. Designing and developing these interactive maps, however, requires specialized, technical skill sets, constraining our collective opportunity to distil the insight latent in unexamined spatial data into actionable knowledge. Even graphical user interfaces like Tableau (*Tableau Software 2019*) and Mapbox Studio (*Mapbox 2019*) - which are designed for non-developers - are complex and daunting for users unused to working with technical software.

Cognizant of this problem, we chose to approach our Digital Visualisation module group assignment with goals on multiple levels. To investigate the topic - Invisible Cities - our imagination was captured by the invisible traces behind each physical product we encounter in an urban system. We decided to investigate the trade, transport and logistics of physical goods at the global, national (United Kingdom) and local (London) levels. Our aim was to illustrate the interconnectivity of our globalizing world and, perhaps, to encourage visualization users to think critically about the products they use, where they come from, and the environmental and social impacts of their consumption.

Our broader goal was to develop a reusable framework for visualizing spatial and non-spatial data on the web. To achieve this, we needed to identify an intuitive, versatile interface and write code to handle a diversity of input data, including vector (points, lines and polygons) and raster spatial data, as well as rich HTML content (text, links, photos, videos, HTML embeddable content, interactive charts, etc). Further, configuration of each snapshot should be simple and intuitive, with an eye towards developing a graphical editor enabling non-technical people to add and manipulate content.

Design Values: Scalability, usability, simplicity, depth, performance, extensibility, privacy.

More detailed requirements as defined in the early stages of application design can be found at github.com/invisible-origins/architecture. This includes work related to front-end and back-end functionality, though databases and server-side logic was not developed for this stage of the project.

2. Methodology

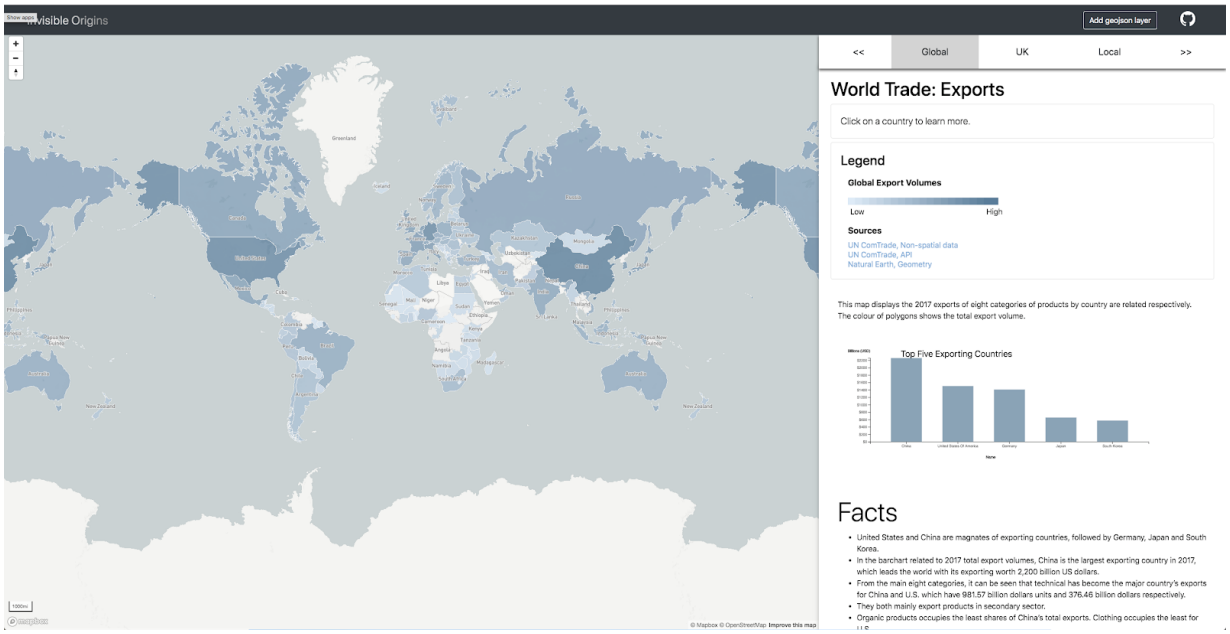
2.1. Processing

"Simplicity, carried to an an extreme, becomes elegance." ~ Jon Franklin
(A-Z Quotes 2019)

Because we were designing our interface with unforeseen data in mind, we emphasized simplicity and tidiness. To this end, there are three top-level elements: the navigational bar, the map - for visualizing spatial data - and the content division - for visualizing HTML content and non-spatial data. In this way, sophisticated geographic visualizations could be displayed alongside divs (which we termed “cards”) providing context, explanation and additional content to the map view (see *Figure 1*).

Figure 1: Lofi App Wireframe - early iteration

Figure 2: Final App Design



To provide the most value to the user in deriving insights from visualized data, interactions and animations were carefully designed. Providing visual feedback between

associated elements - especially between the card elements and map (see World Trade: Exports - country highlight on relevant bar hover, and vice versa, or the map highlight on relevant legend entry hover) - allows the user to quickly see locations and spatial distributions of grouped data, and examine data distributions along different dimensions. The possibilities here are myriad, building on visualization ideas developed by Bostock (*2019*).

Animated scrolling and map fly-to functionality provided the user feedback upon action, reinforcing the link between a card and its corresponding map view (i.e. visible layers and extent) in their mind. Further, animations provide the user with the sense of connectedness in the visual design. For example, a pane slides down to reveal an area to drop files for visualization. Some principles of material design were employed in creating the application (*Material Design 2019*).

2.2. Privacy

Some of our target users may, for reasons including legality, security, ethics or business considerations, require the ability to visualize spatial data without transmitting the data to an externally-controller server. Our experimental “Load geojson layer” feature enables users to visualize point, linestring and polygon features from geojson files dragged and dropped into the file upload area, without that data leaving their local machine.

3. Data

3.1. Managing the data

This visualisation utilises a variety of different data sources, ranging from public to private actors and from global to local coverage. The data sources and their individual data categories can be seen from *Table 1*.

Table 1 – Data sources

Name	*Contribution*	*Data categories*	*Links:*
Government of United Kingdom	National ports freight AIS derived track lines	Non-Spatial Spatial	Here (1) Here (2)
Civil Aviation Authority	National airport freight	Non-Spatial	Here (3)
Natural Earth	Countries geometries Global airports locations	Spatial Spatial	Here (4) Here (5)
US NGA	Global ports locations	Spatial	Here (6)
Airports Council International	Global freight, busiest airports	Non-Spatial	Here (7)
Wikipedia	Global freight, busiest ports	Non-Spatial	Here (8)
UN ComTrade	Trade flows	Non-Spatial	Here (9)
World Bank	Trading Partners	Non-Spatial	Here (10)
Schmidt, D. J.	Wrap for UN ComTrade API	Non-Spatial	Here (11)
OpenStreetMap	Local locations	Spatial	Here (12)

Using a variety of data sources requires managing a variety of data formats, including single files or directories of files of various formats.. To manage and prepare all the

data, a variety of Python scripts were developed, which are found in *Data/Data Processing* on the project-related GitHub. *Table 2* presents an overview of the different scripts and their usages, while Table 3 details the libraries used and their versions. The scripts have three main purposes, namely support (0), production (1) and exploration (2), also indicated in *Table 2*. The support scripts contain functions which support the production process, and the use of support scripts results in a cleaner and more elegant setup. The production scripts are the scripts that generate the final output files, in json, geojson and csv format, to be used in the visualisation. The production scripts take input generated from the support functions together with the variety of data sources found in *Data/Data Sources*. Finally, the explore scripts acts as interfaces for some of the support functions, for the user to better get a sense of what is happening in the background.

Table 2 – Python Scripts for Data Management and Processing

Name	*Usage*	*Credit*	*Purpose*
Airports.ipynb	Preparation of global and national airports json files.	The team	1
functions.py	1. Extract national trade flows. 2. Get an overview of a collection of files within multiple folders. 3. Get trading partners of a given country.	The team	0
Data-Extraction-Using-API.ipynb	Interface for getData.py	The team	2

DealingWithTheData.ipynb	Sensing the data and construction of *top_five_...* data.	The team	1
GlobalAttributes.ipynb	Preparation of global data json file.	The team	1
Ports.ipynb	Preparation of global and national ports json files.	The team	1
UN_Comtrade.py	API wrapper	Schmidt, D. J.	0
Understanding_content_ of_folder_data.ipynb	Interface for extractDescription.py	The team	2

The driver of the data underlying the layers of the visualisation, is the availability of informative dimensions in the data. This visualisation focus on the trade flows in 2017, which restricts the number of countries with available trade flow data to 143 out of a total of 261 available in (4, table 1.). Spatial and Non-Spatial data are mainly joined on the ISO3 codes of the countries, with ISO3 codes being unique three-letter country identifiers.

Table 3 – Libraries used for data management and processing

Name	*Version*	*Source*	*Purpose*
Numpy	1.16.3	Numpy 2019	
Pandas	0.20.3	Pandas 2019	
Matplotlib	3.0.3	Matplotlib 2019	

Os	Native	Os 2019	
Time	Native	Time 2019	
Geopandas	0.4.1	Geopandas 2019	
Basemap	1.2.0	Matplotlib Basemap Toolkit 2019, Stack Overflow 2019	Visualising Spatial data
Ezodf	0.3.1	ezodf 2019	Process ods files
Re	Native	Re 2019	

The airports/ports do not initially come with a country classification. However, the country classification can be attached by performing a spatial join with the country shapefile from (4), implying that two output files contains 187 and 180 countries respectively. The next step is to determine the dimensions of the data to visualise, which is mainly determined by the availability and informativeness of the dimensions in the data. For the airports, there are only two available dimensions: size and type. For the ports there are more than 50 dimensions, some more useful than others. An example is shown in *Figure 3*, to illustrate how the informativeness varies over dimensions. The *Information/Observation* ratio is calculated as the share of observations which do not belong to the group *others* or *none.*

Figure 3 – Information/Observations ratios of three different dimensions

```
Distribution of ports in each size category:
 {'Minor': 2153, 'Small': 990, 'Medium': 361, 'Major': 160, 'Other': 5}
For "Sizes": Information/Observations Ratio = 0.999
------------------------------------------------------------------------
Distribution of ports in each type category:
 {'Coastal': 2112, 'Other': 670, 'River': 814, 'Lake': 73}
For "Types": Information/Observations Ratio = 0.817
------------------------------------------------------------------------
Distribution of ports with Railway access:
 {'None': 2159, 'M': 377, 'L': 146, 'S': 987}
For "Railway Access": Information/Observations Ratio = 0.412
```

Figure 3 shows that the information in the dimension *railway access* is half the size of information in the dimension *type*, and the *railway access* dimension is not considered going forward. The lack of information is in general due to lack of reporting from the individual ports, especially for the smaller ports or those located in developing countries. Besides size and type, which are considered dimensions, for airports and ports, it is of interest to indicate the busiest locations. Including the busiest locations present a minor challenge, as the busiest-location-information do not come with port/airport unique identifier, except for the name of the location. The name of a location is not the most reliable identifier, as the name can be spelt in a variety of ways, which is why some manual identification was needed in this case, which can be seen in both *Airports.ipynb* and *Ports.ipynb.* The final considered dimensions of the airports/ports are size, type, whether or not the locations is one of the top-20th busiest locations on Earth, its ranking within top 20 and the amount of freight passed through as of 2017.

The considered dimensions are relevant because they sustain the aim of visualising invisible cities, and they add great value for the user, in terms of visualising different

spatial dimensions of the data. But most importantly, they aid to illustrate the potential of this reusable framework, in the sense that they, mainly, are categorical variables on which the map design is based.
The data on ports/airports in United Kingdom inherits the dimensions from the global data, just discussed. However, additional dimensions are available as location-specific freight data is easily available. For the ports, additional 7 freight-specific dimensions are added, and 13 for the airports. These additional dimensions are the basis for the pie charts in the layer *The UK in the world*, and they enable the visualisation of the busiest locations within United Kingdom. Furthermore, they present some interesting future extensions.

Finally, as a cherry on the pie and to support the visualisation of ports in United Kingdom, a layer of anonymised AIS line strings from 2015 is explored. The AIS data shed light on the shipping patterns in and around the United Kingdom, with one of the world's busiest shipping lanes passing through the English Channel (*Marine Insight 2019*). The data set is huge and contains millions of observations. To handle the data, the vertices if the line strings are extracted as points in a json file. In order to include the data, efficiently, in the visualisation, the json file was converted into mbtiles using Tippecanoe (*Tippecanoe 2019*), in order to being able to upload the data to MapBox, to rely on their back-end to support the handling and serving of data.
The local layers focus on the biggest airport and port respectively, implying that these layers inherit data from the previous layers, seen in the pie charts. Furthermore, these layers intend to visualise the everyday life around these two highly central and important areas, by utilising OpenStreetMap data embedded in MapBox.

3.2. Challenges

The biggest challenges in terms of the data is obviously the variety of different data formats from the sources, along with the scaling of the data size, as more countries are included in the national layer or from considering more dimensions. An increasing

amount of data formats used, implies an increasing importance of the data preparation. Increasing use of different data types and data formats requires an increasing understanding of the data, to develop creative yet valid solutions, to support the relevance and importance of the visualisation in question. Scaling of the data size, no matter the reason, puts a higher demand on the architecture of the visualisation and an increasing need for a strong back-end, to sustain performance and maintain aesthetics of the architecture and data management.

4. Global illustration

Based on the scale of the globe, there are four sections we will introduce: Export, Import, sea transport and air transports.

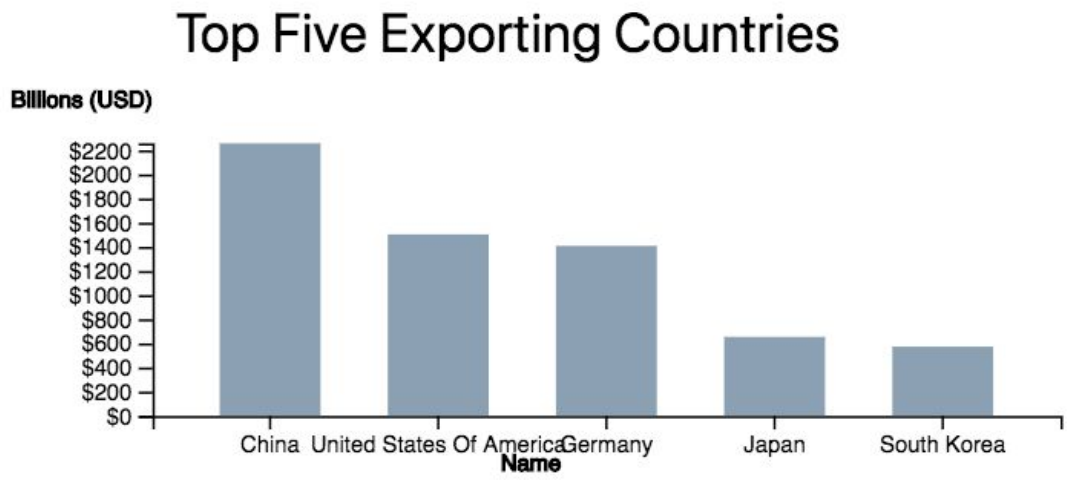
For the first two sections, the referred data are available in the United Nations Database and . Territories of countries are represented by coloured polygons. Exports are visualised on a blue colour gradient and imports on an orange gradient, with deeper hues indicating larger trade values.

For the last two sections, the referred data are available in the US NGA, Natural Earth, Civil Aviation Authority, Civil Council International and Wikipedia. Airports and ports are presented by circles of differing colours and sizes in map.

4.1. Global Export

In this section, it shows the total exporting volume in 2017 and exports of eight categories of products by country are related respectively.

Figure 4. The ranking with top 5 exporting countries

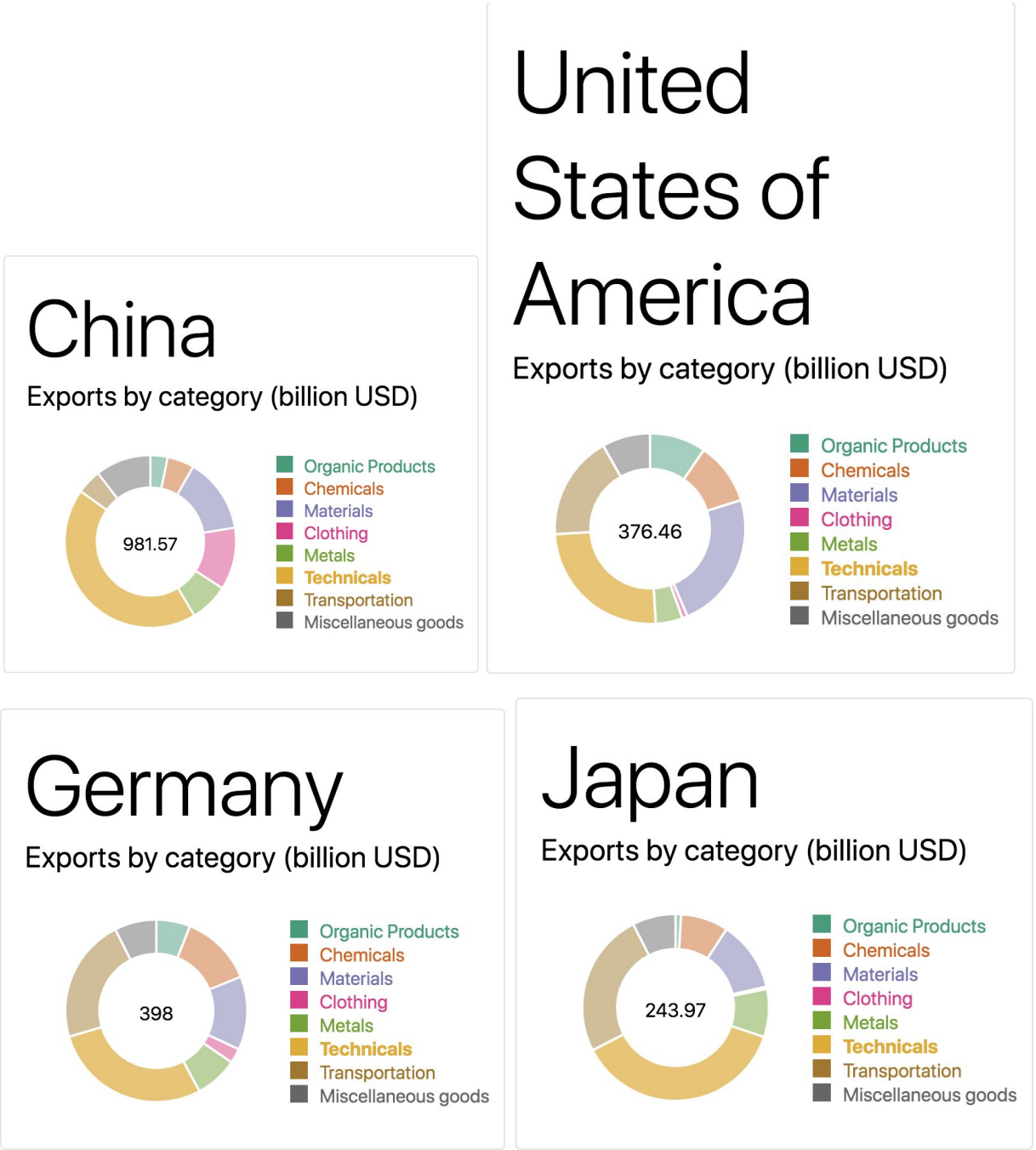


The bar chart, in *Figure 4*, shows the volumes of Top 5 exporting countries in 2017. China is the largest exporting country in 2017, which leads the world with its exporting worth 2,258 billion US dollars. But it is worth mentioning that three out of Top 5 exporting countries are smaller territories. A country's exporting value and its territory will be highlighted via hovering on the corresponding bar. Although Germany claims third with its exporting worth 1,408 billion US dollars, the difference with the second place, the large territory the United States, is quite small.

Furthermore, the type of main export products might affect the total exporting volume of a country. The relevant pie chart can be show in the feature content area by clicking on a polygon. The pie chart presents how much eight main categories of exports share for a country. In the *Figure 5*, It is not difficult to find that Technical exports comprise the largest share in the top five exporting countries. And the rest of countries in general

focus on the export of Materials and Transportations, like Russia, Canada and Australia, etc.

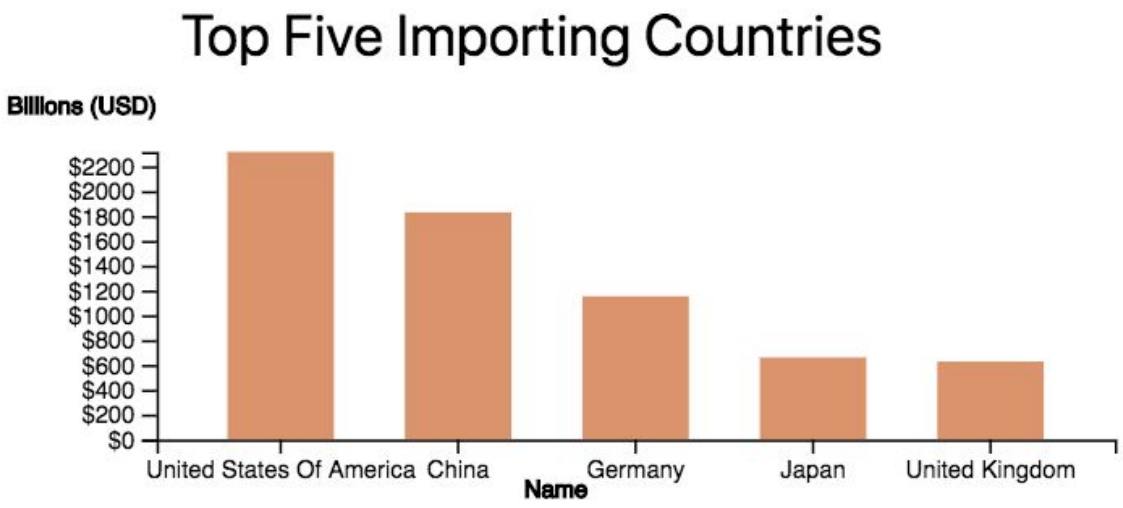
Figure 5. the pie charts for top five exporting countries



4.2. Global Import

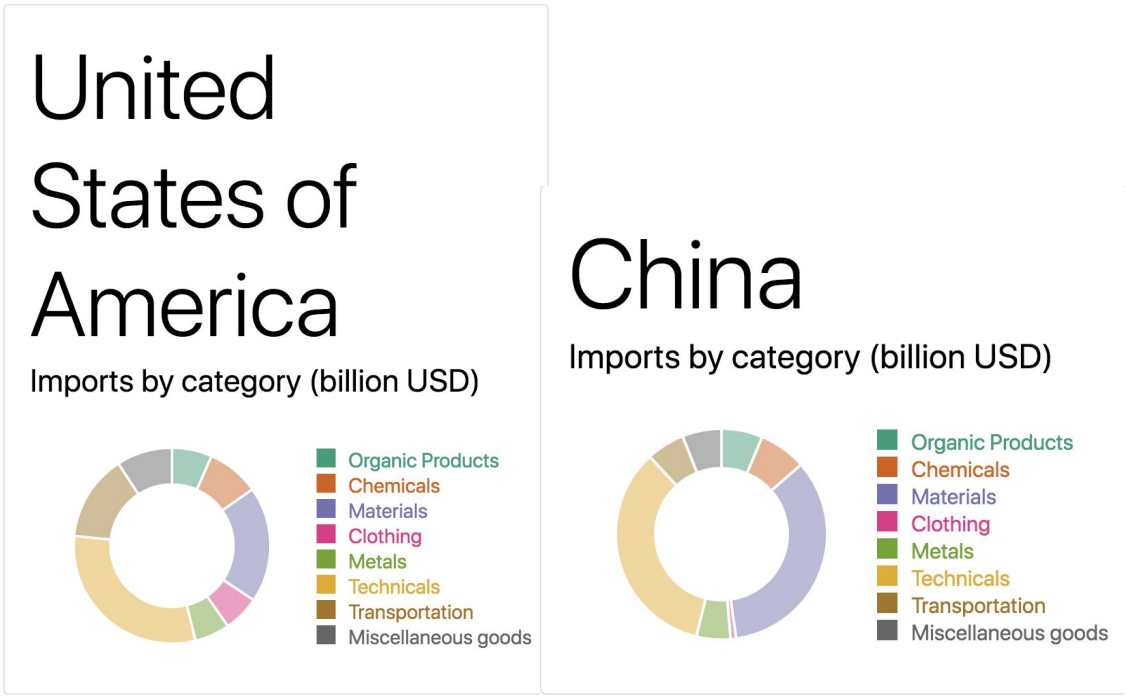
This section shows us the total import volume in 2017 and the shares of eight main import products in each country. Similar to export, the deeper coloured polygon, the more total import in the country. The United States and China are magnates of importing countries. From the map, we can see that the differences of import among Asian countries are quite big. From the *Figure 6*, it shows China imports 1,829 billion US dollars, is more than twice the imports for the Japan.

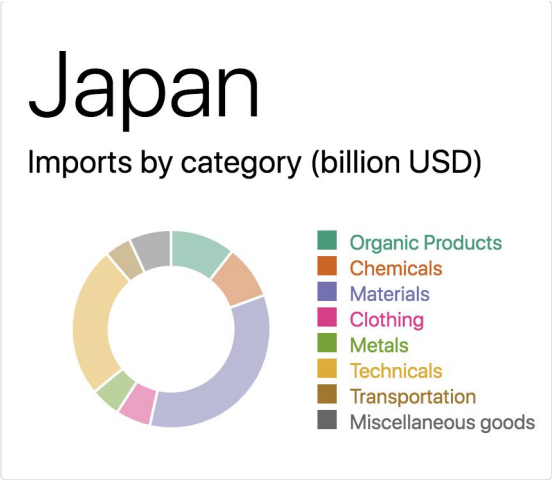
Figure 6. The Top 5 importing countries in 2017



Pie charts indicate that for both China and the USA technical and material products are both major imports.

Figure 7. the pie charts for top five importing countries





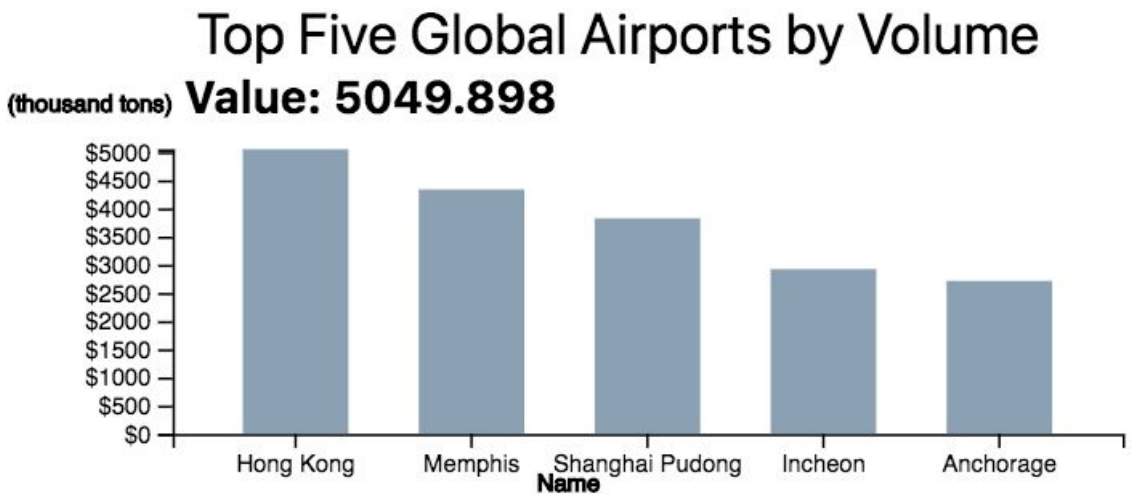
4.3. Global Air Transport

From now, we will focus on the global air transports and global sea transport.
In this section, it displays the distribution of global airports. Each airport will be presented as a specific circle according to its usage and size (*Colorbrewer2, 2019*). There are five categories of airports, which is civilian, military/civilian, military, spaceport and other. For our data, most airports belong to civilian. It is obvious that airports are generally distributed around the southeast and south Asia, Mediterranean and Latin America. And, we can find that airports do not build in some regions with extreme

conditions and low populations. There are few airports appear in very high latitudes. For example, northeast Russia with extremely severe winters has few airports.

Based on the passed-through amounts in 2017, the top five global airports are ranked in the bar chart, *Figure 8*. We can see that three out of top five global airports are located in the Asia, the rest are in North America. With the passed-through volume of approximately 5,049 thousand tonnes, Hong Kong International Airport was the busiest airport in the world in 2017.

Figure 8. Top five Global Airports by passed-through amounts



4.4. Global Sea Transport

Ports are divided into four types according to their locations - coastal, lake, river and other - and by their sizes - minor, small, medium, major and other.

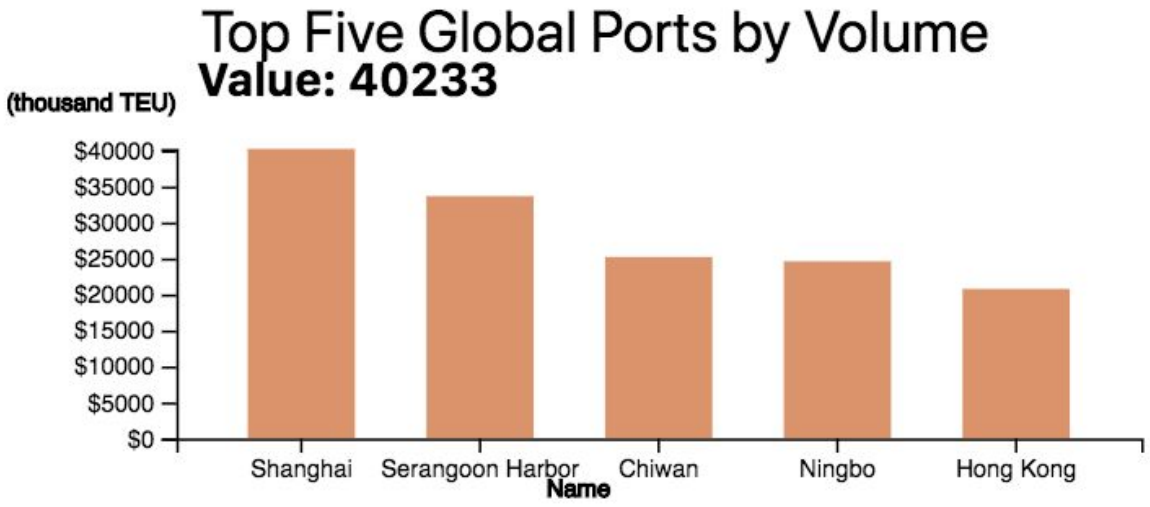
We can find a trend that the number of ports decreases with increasing latitude. Obviously, there is fewer ports appear in the latitude above 72 degree.

In addition, we still find that many ports are mainly distributed near the equator. Many of ports appear around the territory of Indonesia. Also, there are many ports built in the seaward boundaries of the equatorial countries, like Brazil, Cameroon.

Moreover, the distribution of ports is related to country's topography. It is obvious that few ports are built in the territory of inland countries. The distribution of ports in island countries is much denser than in non-island countries.

As shown in the bar chart in *Figure 9,* the top 5 ports are all located in Asia. Shanghai is the world's busiest container port.

Figure 9. Top five Global Airports by passed-through amounts



5. UK illustration

The UK part shows air transportation and sea transportation in the United Kingdom in 2017. The first page shows both airport (*Civil Aviation Authority 2019* and *Natural Earth 2019*) and port locations (*UK GOV 2019* and *US National Geospatial Intelligence Agency 2019*) in the United Kingdom. The colours of airports represent the usage and the colours of ports represent the harbour type. The sizes of the points represent the sizes of airports and ports.

Two pie charts show the destination and provenance of the trades by both air and sea in 2017 (*Figure 10* and *Figure 11*). It can be clearly shown in two charts that for the transportation between the UK and Europe, the airport traffic and the port traffic show an opposite trend. As for air transportation, more than half of the goods transport to non-Europe foreign countries but only a few goods are transported to Europe. However, the pie chart for sea transportation shows that almost half of the goods are transported

to Europe. Therefore, freight transport between the United Kingdom and Europe are usually by sea rather than by air.

Figure 10. UK Airport Traffic

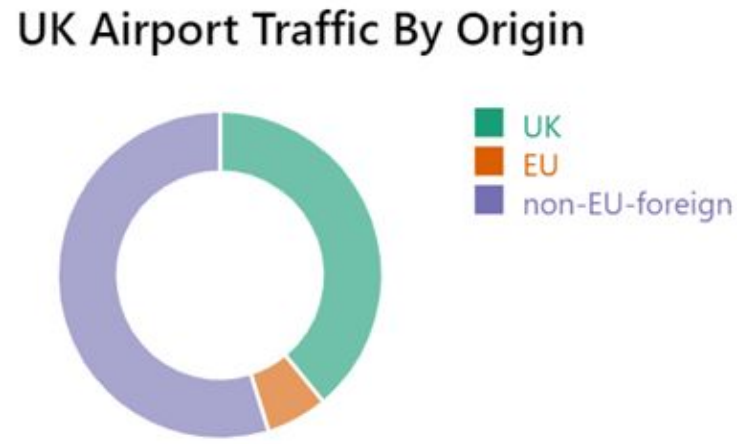
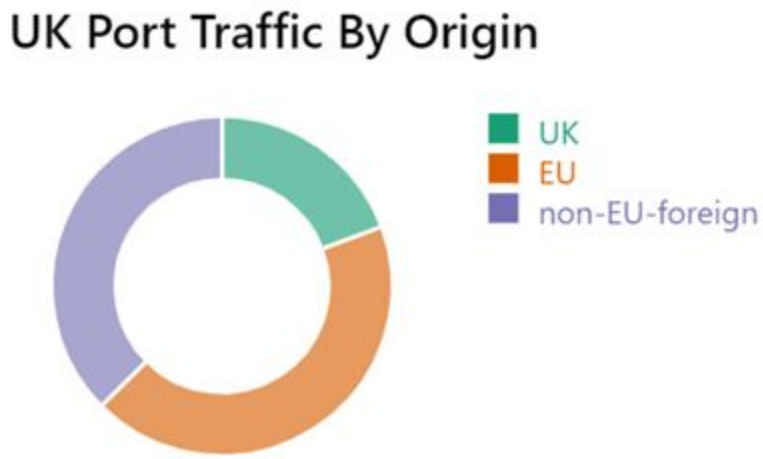


Figure 11. UK Port Traffic



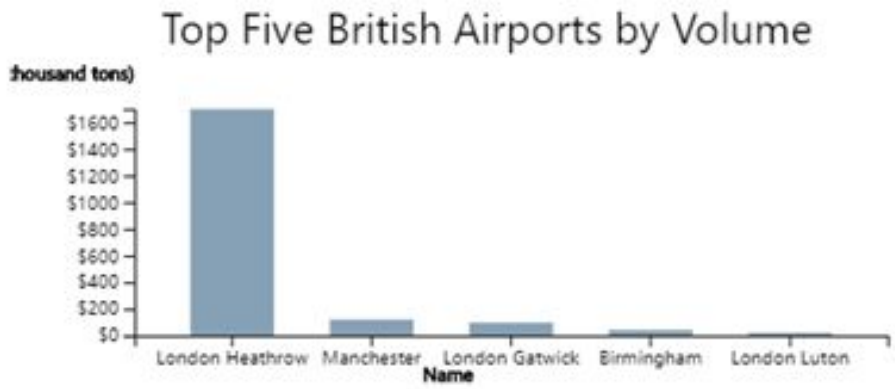
5.1. UK Air Transport

For the UK airports, the colour of the points shows that all the airports in the United Kingdom are civilian, which means that most of the airports in the United Kingdom are not only for cargo transportation but also passengers transportation. Therefore, most airports in the United Kingdom are located in big cities. In addition, the map shows that the airports in England are larger and also more than those in Scotland, Northern Ireland and Wales. These may because that the airports always located in the big cities,

and there are more big cities such as London, Birmingham and Manchester are located in England.

The bar chart in *Figure 12* shows the top five busiest UK airports by freight volume. From the bar chart, three of the top five British airports are located in London, which is the biggest city in the United Kingdom. It is clear that the volume of the goods transported by Heathrow airport is much higher than any other airports (around 1.7 million tons), which is more than ten times the number in second place (around 120 thousand tons). Heathrow Airport originated in 1929, and now is the busiest UK airport by both passenger traffic and cargo traffic. Heathrow Airport is used by over 80 airlines flying to 185 destinations in 84 countries. To the runway and four terminal buildings, it is one of the busiest airports in the UK and the world. The airport is the primary hub of British Airways and is a base for Virgin Atlantic. It has four passenger terminals (numbered 2 to 5) and a cargo terminal (*Heathrow 2019*).

Figure 12. Top Five British Airport by Volume

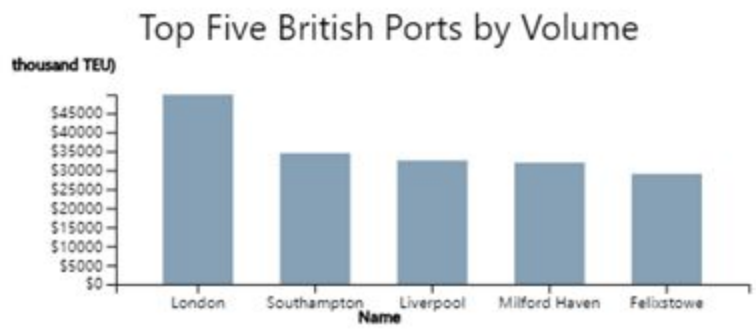


5.2. UK Sea Transport

As for UK ports, the type of the ports shows a different trend compared to the ports on the global level. In global pages above, most of the ports are coastal. However, it can be shown on the map that most of the ports in the United Kingdom are river ports. This may because Britain is an island country and there are many rivers leading to the sea, it will be more convenient for the British to build the ports by the river rather than on the sea.

Figure 13 shows the top five busiest ports in the United Kingdom. All of the five ports are located in the south of Britain, and the busiest port is Port of London, which transported 50 million teus cargos in 2017.

Figure 13. Top Five British Ports by Volume



6. Local illustration

The local part shows the busiest airport and the busiest port of the United Kingdom in 2017. It can be shown from national data analysis above that the London Heathrow Airport is the busiest airport and the Port of London is the busiest port for goods transportation in the UK. For both the airport and port, the buffers were used to show the main facilities in five kilometre's distance. The facilities data are from OpenStreetMap.

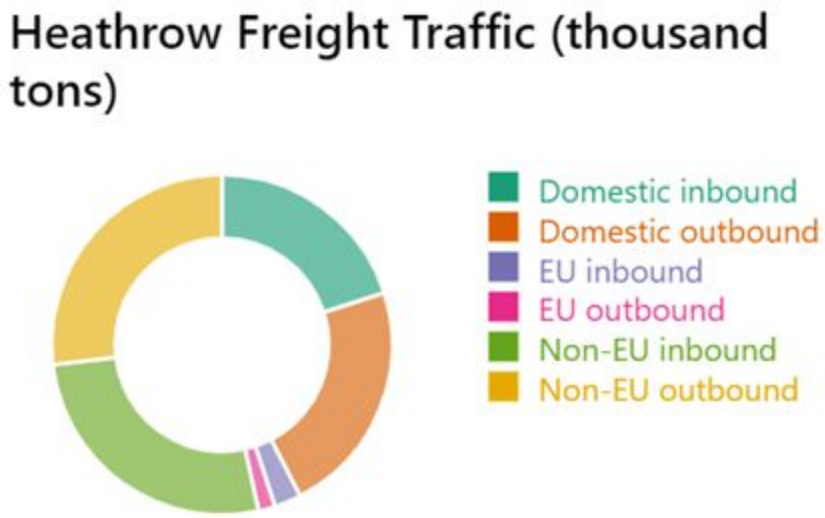
6.1. London Heathrow Airport

As for London Heathrow airport, three facilities were chosen to show the environment around the airport. Transit station locations show the convenience of transportation, hotels and restaurants show locations of amenities for travelers. It can be shown in the map that as the busiest UK airport by both passenger traffic and cargo traffic, Heathrow has many facilities inside the airport and also around, which brings convenience to people's travel. Within five kilometre's distance around the airport, there are 33 hotels and 25 restaurants. Besides, 14 stations are enough to prove that this airport is really busy.

The card on the right side of the page shows some information about Heathrow. By clicking the buttons under ‘A tour of Heathrow’, the map will fly to the specific terminal and will show the 3D buildings and the facilities of that terminal. Besides, the video below also shows the introduction of Heathrow airport. This part refers to two examples of Mapbox: ‘Fly to a location based on scroll position’ and ‘Display buildings in 3D’.

The pie chart (*Figure 14*) shows the volume of freight picked up and set down for both domestic and international transportation in 2017. The data came from the Civil Aviation Authority. For both inbound freight and outbound freight, the largest proportions are for non-Europe foreign countries (more than a quarter for each). Besides, the proportion of goods transported between the UK and Europe is really small. This pie chart shows almost the same trend as UK airports, and one of the reasons is that freight of Heathrow airport has a large proportion of total UK air transportation.

Figure 14. Heathrow Freight Traffic



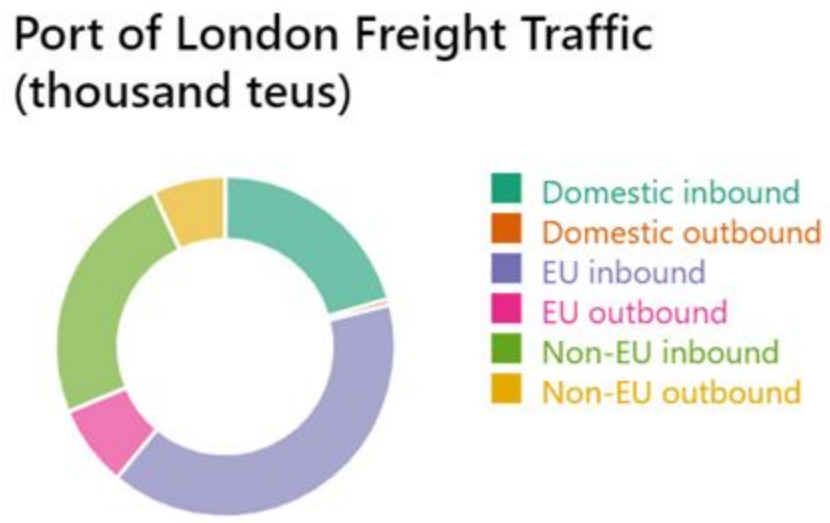
6.2. Port of London

Port of London was once the largest port in the world (*PLA 2019*). The port facilities are located along River Thames and stretch from the capital to the North Sea. There are more than 70 ports located in Port of London, and this project chose the location of Tilbury Port, which is one of the main and largest ports in Port of London, as a representation. Same as Heathrow, a buffer was added to show the facilities around Port of London in 5 kilometre's radius and the ‘source’ button will link to an introduction video.

The stations and restaurants were chosen to represent the convenience around the port. The stations and restaurants are much less than those around Heathrow because this port is only for cargo transportation rather than passengers. Besides, instead of hotels, there are more schools in this place. Therefore, although this the busiest port in Britain, this port is only for goods transportation, but not for people's travel.

The pie chart in Figure 15. shows the inbound and outbound freight for domestic and international transportation. For all of the three transportation, the volume of inbound freight are always more than outbound. This may be because London is a big and international city and has a large population, which can lead to a high demand for goods. Besides, as for total inbound and outbound between different areas, Europe has the largest proportion that almost half of the total transportation, which is almost the same as the UK's situation.

Figure 15. Port of London Freight Traffic



6.3. Challenge

For the Heathrow Airport and Port of London, we only have their own information such as the volume of the goods and whether the goods are domestic transportation or international transportation. But we do not have the data to show which specific airport or the areas they came from or going to. So we cannot have the flows showing on the map.

7. Development

7.1. SVG – Charts and Legends

All charts and legends were built in Scalable Vector Graphics (SVG) format using D3.js (*D3 2019*), which is a JavaScript library build for powerful visualisation in any web browser. Metadata on the functions are listed in Table 4 and they are found in the script *functions.js*, located in *docs/ui/scripts* on GitHub. Their purpose follows the outline described in the data section.

Table 4 – Functions related to charts and legends

Name	*Usage*	*Purpose*
createBarChart	Create bar chart	1
createPieChart	Create pie chart	1
createLegends	Create legends	1
structureData	Structure data for legends	0
interpolateColors	Set colours for legends	0

All the functions were built with the intention of being generic, supporting the intention of building a reusable framework. The bar and pie charts are built in a similar fashion, which is that a *svg* element is embedded within the relevant *div*, on which multiple *g*'s are appended containing the information of interest. Interaction, between information contained in two *g*'s or between any *g* and the main map, is enabled by applying the same class to related elements within an SVG. More specifically, to link countries in the main map with individual bars in the bar charts, thereby enabling highlight on hover, we class the bars by the relevant country ISO3 code, which is used for filtering the highlight

layer on the main map. The global pie charts are built on click, where the country-specific-trade-flow-data is extracted from the layer and passed into *createPieChart*. Within the function, each arc and legend is classed with the category name, which enables interaction on hover.

The legends are built dynamically based on the layers contained in a given card. The legends are built on one single SVG element in two stages. First, the needed size of the SVG is determined by examining the data type and the number of sources of each layer. Secondly, for each layer, the data and relevant colours are extracted and structured using *interpolateColors* and *structureData*. Titles, colour referencing, and sources are appended on separate *g*'s. The sources have a matching link embedded, opening in new tabs when clicked. Interaction on point layer legend entries is enabled in the way as previously described, by classing legend text's by categories in the layer, used for filtering.

The interaction between the spatial element, in terms of the main map, and the non-spatial information on the cards was of great interest from the beginning. The interaction between the main map and information on the cards increases the informativeness of the visualisation, and aids education of the viewer, with the aim of hopefully improving decision making.

7.2. Challenges

Two challenges occurred with different basis, namely interactivity on classes and dealing with different data types.

Firstly, interactivity on classes, whether it is between the main map and elements in the cards or graphical elements within a chart and its corresponding legend text, is based on the class names. This presented a challenge because d3 doesn't support selection if a name contains certain symbols ("/" as an example) or if the name has spaces, because the last will result in multi-class-labelling. To deal with this challenge, one simple solution is to replace conflicting

elements with something that is allowed. An example is to replace spaces with underbars, in the class labelling.
Secondly, dealing with different data types presents a challenge, when it comes to create legends for the data visible for the user. Different data types are a challenge because the legends need to be formed differently, depending on the data type. In order to use the available space most efficiently and create as aesthetic and appealing a design as possible, gradient-fill-legends are best visualised horizontally and multiple-category-point-based legends are best visualised vertically, adding a size representation next to it if needed. This is what is considered in the function *createLegends*, while at the same time considering the sources of each data type. The output is elegant and interpretable, but the code could have been structured better, with support functions for each data type – which is a task for future development.

7.3. Architecture

Since we were designing with an eye towards extensibility, we placed emphasis on writing intuitively-structured code, enabling developers to continue to build on our efforts. We have many plans to improve and extend the app architecture (see Future Work), but currently the execution occurs as follows as the browser renders the page:

7.3.1. `./index.html`

Our entry point is an HTML page. This page was primarily built using bootstrap, d3 and mapbox - relevant stylesheets and javascript libraries are loaded from supported content delivery networks. (Minor performance gains could likely be achieved here by importing only necessary modules, and bundling code into a single HTTP GET request.) In the document's body, site and map navigational elements are included, as well as empty holder elements for mapboxgl.js and d3.js to create the map and graphics. Just before the closing </body> tag, four custom javascript files are loaded and executed.

7.3.2. `./scripts/functions.js`

In this script we declare all the functions used to load and interact with the visualization. No functions are invoked in this file.

7.3.3. `./data/layers.js`

Here an array of object literals assigned to a globally-declared variable, `layerData`. Each object contains information about the type, source, and styling of each layer.

7.3.4. `./data/cards.js`

In `cards.js` an array of objects containing attributes and methods for each card included in the visualization is assigned to global variable, `cardData`.

Note: Both layer and card objects represent an opportunity to define classes to contain the data, to be instantiated on page load. Originally these were designed as JSON files, but the lack of native support to store methods in JSON (*StackOverflow 2010*), along with the advantage of having card- and layer-specific methods written within the object describing the card or layer (as opposed to in another data structure, referenced by some key) meant that we ended up storing our definition of cardData and layerData in javascript files.

7.3.5. `./scripts/load-map.js`

In `load-map.js`, code is executed that loads data, populates interface content (including the map and cards), and sets event listeners that provide the interactive functionality of the app.

8. Future Work

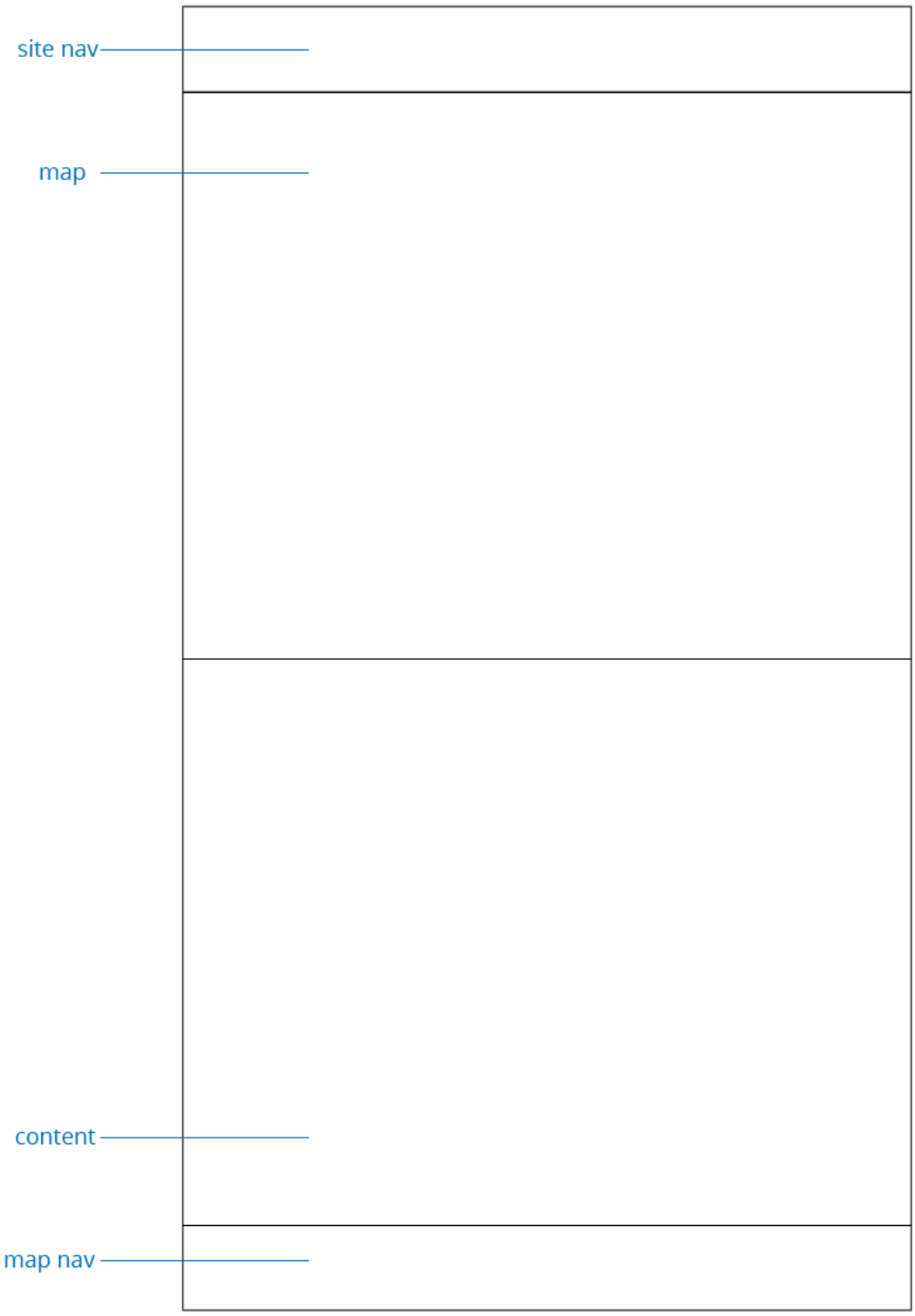
This is a good start, but extending this project into a framework will require substantial input of tim and development resources. Here we outline several opportunities for improvement. Substantial work is described here in some detail; smaller feature upgrades are listed below.

8.1. Mobile

Visualizing interactive maps on mobile devices is difficult due to screen size constraints and the lack of a hover state; in many way the layout and interactivity of the app needs to be redesigned. We envision a modified interface to provide mobile users a robust interactive experience of the app. Map div height would be fixed, and content would

scroll ‘under’ the map area. While untested, application logic and visualizations should work without significant adaptation for mobile. Wireframes below.

Figure 16. Visualizing interactive maps on mobile



8.2. Back end

This application has a relatively lightweight data load - we rely on Mapbox server architecture to host larger datasets, and load smaller ones in from JSON and CSV files hosted on our web server. Implementing a more sophisticated back end server and database management system would allow client instances to request data as needed, using AJAX requests or a websockets connection. In this way users could interact with much more complex visualizations based on larger datasets, with much of the data and computational load outsourced to the server. We have not designed this architecture thoroughly, but would look closely at implementing it in NodeJS (*NodeJS Foundation 2019*) or GeoDjango (*GeoDjango 2019*), and MongoDB (*MongoDB 2019*) or PostgreSQL (*PostgreSQL 2019*).

8.3. Graphical Editor Interface

Given that our app is designed to enable less technical people to build interactive visualizations, providing an intuitive graphical interface enabling users to create, update or delete cards and layers is a logical next step. This feature would require a full stack architecture including login functionality, databases, etc. We are considering building atop existing content management system frameworks such as wordpress to improve the likelihood of adoption.

Maintaining a simple and intuitive design will be a challenge as we build interfaces intended to provide users graphical ways to visualize complex spatial and non-spatial data. An extensive research and testing process will improve the outcome of the effort. Interim solutions - such as a Google Form, plus a parser script to build protocol-compliant layer and card objects - could enable rapid progress.

8.4. Feature Upgrades

- Animate map layer transitions on card change (*Stack Overflow 2019*).

- Refactor createLegend, createPieChart and createBarChart functions to utilize the same SVG elements, rather than redrawing them, for computational efficiency, and to incorporate D3 animated transitions.
- Fix Mapbox popups reliability and mousemove bugs.
- Add GeoJSON validity checks and multi-file / zip / shapefile upload capability to local file layer load feature.
- Implement VR-responsive interface
- Write tests and ensure cross-browser compatibility, including fallbacks for legacy browsers
- Ensure accessibility features are included for users with disabilities including hearing, vision, speech and motor impairments.

9. Conclusion

Few topics illustrate the interconnectivity of our world as well as the origins of the products we find in cities. They each have a story, invisible traces back to their origins, weaving through the immensely complex supply and logistics chains we employ in the 21st century. This visualization sought to convey that scale and complexity to people in an elegant way at every scale.

Concurrently, we aimed to build a reusable visualization framework to provide people with a way to better understand the world. Much remains to be done to complete this task, but the outlines are in place, and a functional prototype is built.

10. Bibliography

A-Z Quotes. 2019. TOP 25 SIMPLICITY IN DESIGN QUOTES (of 54) | A-Z Quotes. [ONLINE] Available at: https://www.azquotes.com/quotes/topics/simplicity-in-design.html. [Accessed 28 May 2019].

Bostock, Mike. Scatterplot Matrix Brushing. (2019). https://bl.ocks.org/mbostock/4063663

Colorbrewer2. (2019). *ColorBrewer: Color Advice for Maps*. [online] Available at: http://colorbrewer2.org/#type=qualitative&scheme=Set2&n=7 [Accessed 26 May 2019].

D3 (2018). Data-Driven Documents. [online] Available at: https://d3js.org/ [Accessed 13 May 2019]

ezodf. 2019. ezodf · PyPI. [ONLINE] Available at: https://pypi.org/project/ezodf/. [Accessed 28 May 2019].

GeoPandas 0.5.0 — GeoPandas 0.5.0 documentation. 2019. GeoPandas 0.5.0 — GeoPandas 0.5.0 documentation. [ONLINE] Available at: http://geopandas.org/. [Accessed 28 May 2019].

GeoDjango | Django documentation | Django. 2019. GeoDjango | Django documentation | Django. [ONLINE] Available at: https://docs.djangoproject.com/en/2.2/ref/contrib/gis/. [Accessed 28 May 2019].

Heathrow (2019). Facts and figures | Heathrow. [online] Available at: https://www.heathrow.com/company/company-news-and-information/company-information/facts-and-figures [Accessed 25 May 2019].

Mapbox. (2019). Display buildings in 3D. [online] Available at: https://docs.mapbox.com/mapbox-gl-js/example/3d-buildings/ [Accessed 25 May 2019].

Mapbox. (2019). Fly to a location based on scroll position. [online] Available at: https://docs.mapbox.com/mapbox-gl-js/example/scroll-fly-to/ [Accessed 25 May 2019].

Mapbox. (2019). Mapbox Studio. [ONLINE] Available at: https://studio.mapbox.com/. [Accessed 27 May 2019].

Material Design. (2019). Homepage - Material Design. [ONLINE] Available at: https://material.io/. [Accessed 27 May 2019].

Matplotlib. (2019). Matplotlib: Python plotting — Matplotlib 3.1.0 documentation. [ONLINE] Available at: https://matplotlib.org/. [Accessed 27 May 2019].

Matplotlib Basemap Toolkit. (2019). Welcome to the Matplotlib Basemap Toolkit documentation — Basemap Matplotlib Toolkit 1.1.0 documentation. [ONLINE] Available at: https://matplotlib.org/basemap/. [Accessed 28 May 2019].

Marine Insight. (2019). The Strait Of Dover- The Busiest Shipping Route In The World. [ONLINE] Available at: https://www.marineinsight.com/marine-navigation/the-strait-of-dover-the-busiest-shipping-route-in-the-world/. [Accessed 28 May 2019].

MongoDB. (2019). The most popular database for modern apps | MongoDB. [ONLINE] Available at: https://www.mongodb.com/. [Accessed 28 May 2019].

NumPy — NumPy. (2019). NumPy — NumPy. [ONLINE] Available at: https://www.numpy.org/. [Accessed 27 May 2019].

Node.js Foundation. (2019). Node.js. [ONLINE] Available at: https://nodejs.org/en/. [Accessed 28 May 2019].

os. (2019). os — Miscellaneous operating system interfaces — Python 3.7.3 documentation. [ONLINE] Available at: https://docs.python.org/3/library/os.html. [Accessed 28 May 2019].

Pandas. 2019. Python Data Analysis Library — pandas: Python Data Analysis Library. [ONLINE] Available at: https://pandas.pydata.org/. [Accessed 27 May 2019].

PLA. (2019). *Port of London Authority*. [online] POLA2012. Available at: http://www.pla.co.uk/ [Accessed 28 May 2019].

PostgreSQL. 2019. PostgreSQL: The world's most advanced open source database. [ONLINE] Available at: https://www.postgresql.org/. [Accessed 28 May 2019].

re. 2019. re — Regular expression operations — Python 3.7.3 documentation. [ONLINE] Available at: https://docs.python.org/3/library/re.html. [Accessed 28 May 2019].

StackOverflow 2010. https://stackoverflow.com/questions/2001449/is-it-valid-to-define-functions-in-json-results

Stack Overflow. 2019. python - How to install Matplotlib's basemap? - Stack Overflow. [ONLINE] Available at: https://stackoverflow.com/questions/33020202/how-to-install-matplotlibs-basemap. [Accessed 28 May 2019].

Tippecanoe (2019). Builds vector tilesets from large (or small) collections of GeoJSON, Geobuf, or CSV features. [online] Available at: https://github.com/mapbox/tippecanoe [Accessed 21 May 2019].

Tableau Software. 2019. Business intelligence and analytics software. [ONLINE] Available at: https://www.tableau.com/en-gb. [Accessed 27 May 2019].

time. 2019. time — Time access and conversions — Python 3.7.3 documentation. [ONLINE] Available at: https://docs.python.org/3/library/time.html. [Accessed 28 May 2019].