Using data visualisations for airline route planning

Robert Joscelyne
Work completed on 2nd of November 2023
robjoscelyne@yahoo.co.uk

***Abstract* — A start-up airline wishes to operate a new short-haul route serving the San Francisco Bay Area. The management team have shortlisted three potential departure airports. Passengers rate airlines' punctuality. Delays and cancellations can cause travellers to avoid the airline or seek cheaper future tickets. For these reasons, the management team must know which airport shall provide the best on-time performance. A visualisation was developed using attention and distraction principles to help effectively communicate the airport with the lowest cancellations and delays. Data preprocessing was performed on a dataset downloaded from the United States Bureau of Transportation Statistics. Dataset queries were developed in Python to match the business objectives: short-haul flights departing in the morning from 1. San Francisco International (SFO), 2. Mineta San Jose International Airport (SJC), and 3. Metropolitan Oakland International (OAK). An interactive visualisation using D3.js was developed and showed that the airport with the best on-time performance was Metropolitan Oakland International.**

I. Dataset

The dataset was downloaded from the United States Bureau of Transportation Statistics [13]. The source data spans January 2018 to July 2022. Five datasets totalling 1.24GB, were downloaded from the URL listed below. The download took 20 minutes.

https://www.transtats.bts.gov/DL_SelectFields.aspx?gnoyr_VQ=FGK&QO_fu146_anzr=b0-gvzr

An attempt was made to merge the datasets into a single Microsoft Excel file; however, the program became unresponsive due to the quantity of data. In the author's opinion, this would satisfy the volume element of big data. Code was written in Python to handle the large quantity of information. The datasets were merged and then optimised using Apache Parquet, an open-source format for efficient data storage and retrieval. A single parquet file was uploaded to Google drive, which can then be accessed in the notebook. When the notebook was run, the dataset took approximately 5 seconds to download.

The dataset is comprised of 2,700,014 rows with 120 columns. The data can be divided into quantitative (numerical) and qualitative (categorical) types. NOIR (Nominal Ordinal Interval Ratio) levels were used to classify the features:

1. CarrierDelay (mins) and DepDelayMinutes (mins) – Quantitative and ratio.

2. Cancelled (label) – Qualitative and nominal.
3. Distance (miles) – Quantitative and ratio.
4. CRSDepTime (time) – Quantitative and interval.
5. Origin (label – airport name) - Qualitative and nominal.

II. Data Exploration, Processing and Cleaning

Feature selection

For scheduling, CRSDepTime was chosen and may explain if flights get delayed or cancelled during specific periods. Origin shall be needed to compare delays and cancellations for different airports. CarrierDelay (mins) and DepDelayMinutes (mins) were selected for delay analysis. Cancellations were represented by the feature Cancelled. For more information, see the feature selection header in the notebook.

Missing values

The dataset was checked for missing values. In total, 2,273,524 missing values were detected in the CarrierDelay feature. See figure 1. Removing records with missing values would mean that 84% of the dataset would be lost.

```
Year has                        0  missing values
CRSDepTime has                  0  missing values
Origin has                      0  missing values
OriginState has                      0  missing values
Distance has                    0  missing values
CarrierDelay has                     2273524  missing values
DepDelayMinutes has                  79451  missing values
Cancelled has                   0  missing values
```

Figure 1. Missing values present in the dataset.

Departure delay was an alternative feature that explained the same information. Departure delay only had 79,451 missing values equating to 3% of the total data. Value imputation was investigated for both features; however, it could create erroneous data relationships. With fewer missing records, eliminating 3% of the dataset seems preferable. DepDelayMinutes replaced carrier delay. Figure 2 shows feature selection and missing value elimination.

```
Year has                        0  missing values
CRSDepTime has                  0  missing values
Origin has                      0  missing values
OriginState has                      0  missing values
Distance has                    0  missing values
DepDelayMinutes has                  0  missing values
Cancelled has                   0  missing values
```

Figure 2. Updated feature selection and removal of missing values.

Generating data for visualisations

The dataset contained over 2 million aircraft movements dating from January 2018 to July 2022. Using Jupyter notebooks, the Python loc function from Pandas was used to pick specific rows and columns depending on business objectives: short-haul morning flights from 1. San Francisco International (SFO), 2. Mineta San Jose International Airport (SJC), and 3. Metropolitan Oakland International (OAK). Short-haul flights can be considered less than 3 hours of flight time [1], equating to a distance of no greater than 600 miles. Section I determined the data type for DepDelayMinutes was quantitative and ratio. This data type makes it possible to compute the mean. The average number of delays was computed for each airport in figure 3.

Bar chart - compute average delays for short haul morning flights departing SFO SJC and OAK

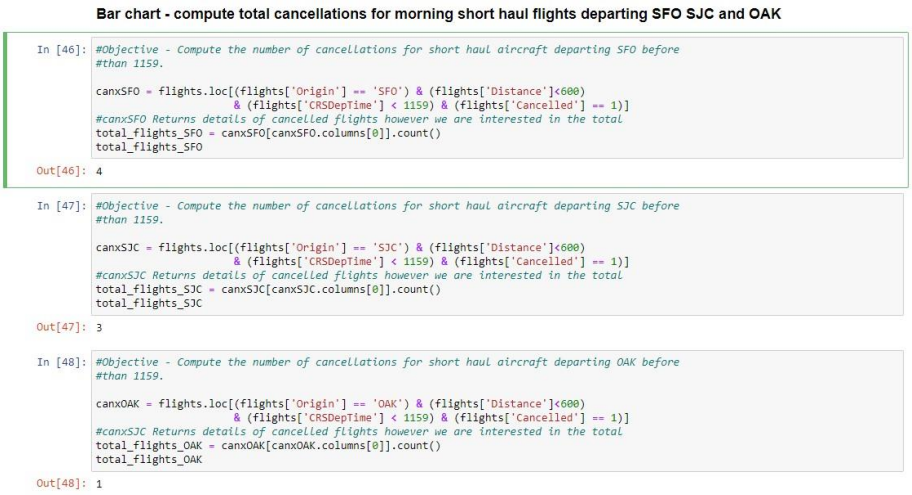
```
In [13]: #Objective - Compute the average delay for short haul aircraft departing before than 1159 for SFO.
         delSFO = flights.loc[(flights['Origin'] == 'SFO') & (flights['Distance']<600)
                              & (flights['CRSDepTime'] < 1159)]
         avDelSFO = delSFO['DepDelayMinutes'].mean()
         avDelSFO
Out[13]: 10.54010977411864

In [15]: #Objective - Compute the average delay for short haul aircraft departing before than 1159 for SJC.
         delSJC = flights.loc[(flights['Origin'] == 'SJC') & (flights['Distance']<600)
                              & (flights['CRSDepTime'] < 1159)]
         avDelSJC = delSJC['DepDelayMinutes'].mean()
         avDelSJC
Out[15]: 4.90282948622487

In [49]: #Objective - Compute the average delay for short haul aircraft departing before than 1159 for OAK.
         delOAK = flights.loc[(flights['Origin'] == 'OAK') & (flights['Distance']<600)
                              & (flights['CRSDepTime'] < 1159)]
         avDelOAK = delOAK['DepDelayMinutes'].mean()
         avDelOAK
Out[49]: 3.8696336255338277
```

Figure 3. Code used to compute mean delays for SFO SJC and OAK.

The data type for the feature Cancelled was identified as qualitative and nominal. The average could not be computed for this data type (without further preprocessing). However, it was possible to compute the mode (the most frequent number) providing a total value for cancellations. The total cancellations were calculated for each airport, as shown in figure 4.



Bar chart - compute total cancellations for morning short haul flights departing SFO SJC and OAK

```
In [46]: #Objective - Compute the number of cancellations for short haul aircraft departing SFO before
         #than 1159.

         canxSFO = flights.loc[(flights['Origin'] == 'SFO') & (flights['Distance']<600)
                              & (flights['CRSDepTime'] < 1159) & (flights['Cancelled'] == 1)]
         #canxSFO Returns details of cancelled flights however we are interested in the total
         total_flights_SFO = canxSFO[canxSFO.columns[0]].count()
         total_flights_SFO
Out[46]: 4

In [47]: #Objective - Compute the number of cancellations for short haul aircraft departing SJC before
         #than 1159.

         canxSJC = flights.loc[(flights['Origin'] == 'SJC') & (flights['Distance']<600)
                              & (flights['CRSDepTime'] < 1159) & (flights['Cancelled'] == 1)]
         #canxSJC Returns details of cancelled flights however we are interested in the total
         total_flights_SJC = canxSJC[canxSJC.columns[0]].count()
         total_flights_SJC
Out[47]: 3

In [48]: #Objective - Compute the number of cancellations for short haul aircraft departing OAK before
         #than 1159.

         canxOAK = flights.loc[(flights['Origin'] == 'OAK') & (flights['Distance']<600)
                              & (flights['CRSDepTime'] < 1159) & (flights['Cancelled'] == 1)]
         #canxSJC Returns details of cancelled flights however we are interested in the total
         total_flights_OAK = canxOAK[canxOAK.columns[0]].count()
         total_flights_OAK
Out[48]: 1
```

Figure 4. Code used to compute total cancellations for SFO SJC and OAK.

The map visualisation used the IATA code, e.g. SFO, to plot the position of the airports within the United States. See section 2 of the notebook for more info.

III. VISUALISATION

Choice of visualisations

A bar chart is used to compare the numerical values of categories and can quickly find the lowest or highest values in the data [2]. Airports were x-axis categories, and cancellations and delays were y-axis quantitative values. A multiple-unit or right-side y-axis was considered; however, this was avoided as it could clutter the visualisation. The message was to show the lowest value indicating which airport had the best on-time performance. A bar chart was considered a suitable medium to convey this idea. See figures 5 and 6. A logo was placed on the top left to indicate official company business. Llama Airways is a fictitious company created for this assignment, but the dataset is based on real-world aircraft movements. The chart could be considered exhibitory and utilitarian.

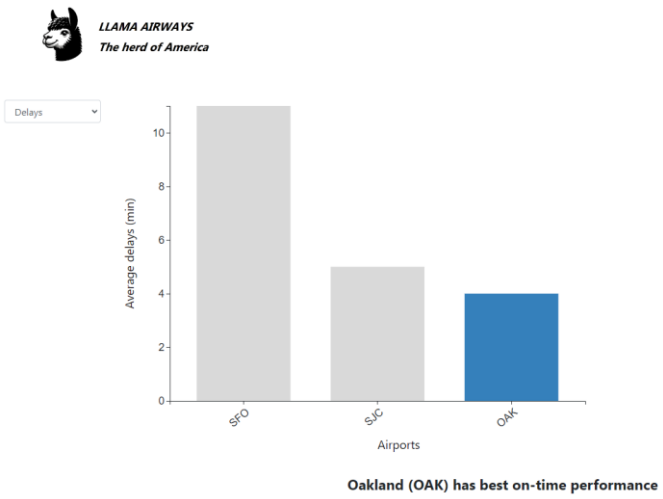


Figure 5. Bar chart created in D3.js showing average delays for SFO SJC and OAK.

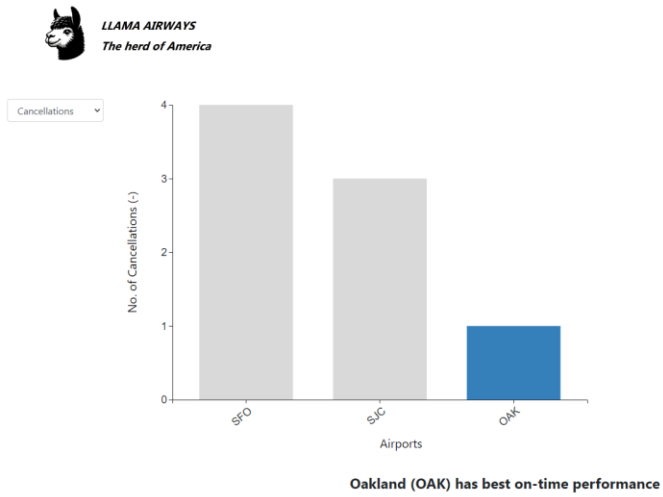


Figure 6. Bar chart created in D3.js showing total cancellations for SFO SJC and OAK.

As the analysis focused on airports and the bar charts didn't encode spatial information, it seemed a good idea to use a dot distribution map to show their relative positions [3]. See figure 7. The scaled dot plot map is focused on the San Francisco Bay Area. The audience is the United States-based Llama Airways management team; therefore, it may be presumed that they are familiar with the region. Excel 3D maps offer a feature to draw 3D charts. A decision was made to avoid using 3D visualisations as these could distort the information. The dot plot map could be described as exploratory with a big-picture focus.

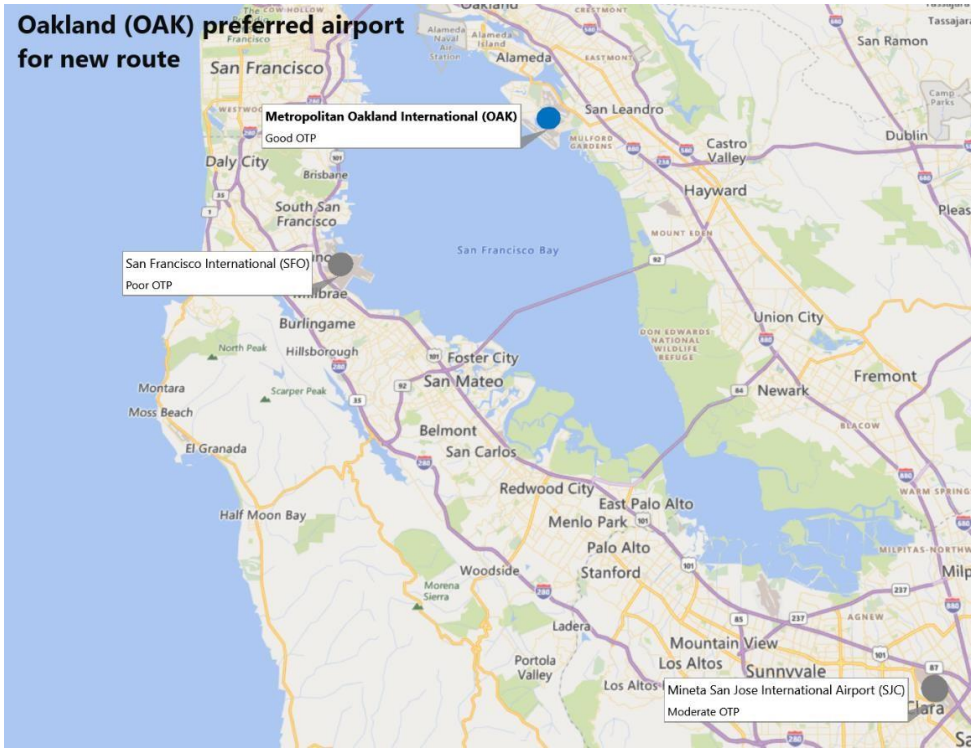


Figure 7. Dot distribution map showing the spatial relationships between the airports.

Design

Colour vision deficiency is a common condition that affects around 1 in 12 men and 1 in 200 women [4]. Most people who are colour-blind have difficulty distinguishing between red, yellow and green shades. Greyscale can be a good choice for individuals with colour blindness [5]. A blue and grey colour palette was used to improve the readability of both visualisations. See figures 5, 6 and 7. The surface area that is coloured affects how we perceive colour. Therefore the bars in the bar chart were widened. The circular markers on the map were enlarged beyond the default values. The idea is to help lessen the cognitive load for readers who are already struggling to differentiate the colours [5].

In addition to colour blindness, about 15% of the population is affected by long-sightedness (hyperopia). Long-sightedness reduces the ability to see nearby objects [6]. The Jaeger scale is a typical near-acuity test used by optometrists [7]. Optimal near-acuity is when a person can read the 10th line on the chart (20/20 vision). To factor in long-sightedness, the text in all visualisations was adjusted to be the same or greater in size than the 4th line on the Jaeger chart (20/100 vision), equivalent to 14 font in Times New Roman [7].

Based on these findings, a font size of 14 shall be the minimum size for text on the bar chart and 3D map. Figure 8 shows Times New Roman font at size 18 in use.

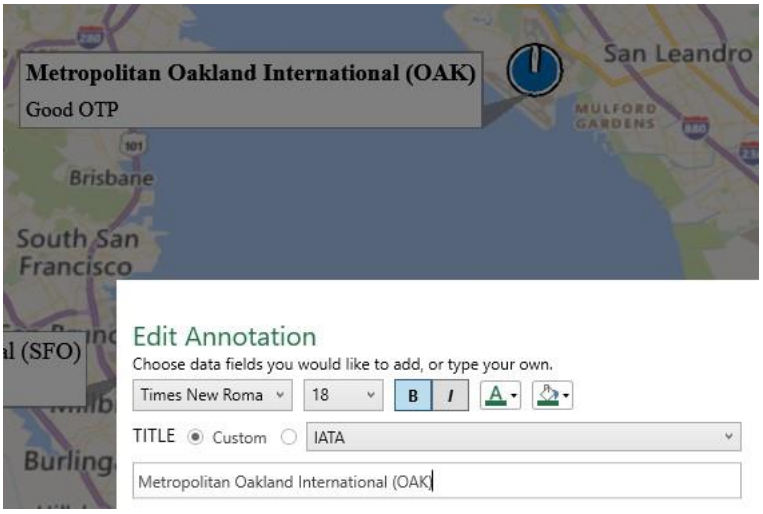


Figure 8. A font size of 18 was used for annotations in dot plot map.

Saliency identifies regions in a visualisation that catch the viewer's eye. Saliency can guide the reader's attention to convey a message. Guttenberg diagrams, F and Z patterns are document structuring theories. All these approaches use the top left as the starting position [8]. This approach was incorporated into the bar chart by placing the airline's logo in the top left corner. See figures 5 and 6. The same approach was applied to the dot plot map showing the conclusion of the analysis. See figure 7.

The similarity principle from Gestalt theory says that people tend to group similar elements, e.g. elements with the same colour [9]. Using a blue action colour, the notion of similarity (or lack thereof) was used to distinguish the OAK category from the other airports. A similar technique is utilised on the map, where a blue circle indicates Oakland's location. The bottom right of the bar graph and the top left of the map display an additional instance of bold text. Text intensity assists the message of the visualisation.

Another example from Gestalt theory is closure. The principle of closure states that we finish incomplete shapes [9]. The bar charts in figures 5 and 6 show that even without a border, it is still possible to imply the enclosure of the chart as a grouped object. Removing unnecessary graphics can help reduce clutter and improve readability. Based on recommendations from Knaflic [10], the following steps have been taken to reduce clutter from the visualisations. Gridlines have been removed from the bar charts. The quantitative values shown in the bar chart and dot plot map are whole numbers (see video). Instead of a legend or chart, the map labels airport locations with names and on-time performance rankings.

The preattentive attributes of form and spatial position [11] were used in the bar charts. The bar chart uses length to encode the values. Even without any colour or other preattentive attributes, you can easily see that OAK has the lowest delays and cancellations; conversely, it is evident that SFO has the most significant number of delays and cancellations. Sorting the values (high, medium and low) helps make it more evident

that OAK has the best on-time performance. The white background helps emphasise the categories' spatial position [11]. A dark background colour was not chosen as this could distract from the information presented on the chart. The bars are equally spaced, which helps emphasise that each category is distinct and does not belong to any sub-categories.

Interactivity

Instead of creating two separate bar charts for delays and cancellations, a drop-down list was incorporated to switch between the two metrics on the same chart. This method required using D3.js to load delay or cancellation data. The logic used to create this functionality is shown in figure 9. When the user switches between the two metrics, smooth transitions activate to blend in the new information. These animations were implemented for aesthetic reasons. Techniques presented by Janes [12] were used to develop the bar chart.

```
d3.select("#selected-dropdown").text("first");
d3.select("select")
.on("change",function(d){
var selected = d3.select("#otp-select").node().value;
console.log("Selected value is ",selected );
        if (selected == "delays")
        {
          flag = 1
        }

        if (selected == "cancellations")
        {
          flag = 0
        }
  d3.select("#selected-dropdown").text(selected);
  })

function update(data) {
  const value = flag ? "delays" : "cancellations"
  const t = d3.transition().duration(185)
```

Figure 9. D3.js code used to switch between delays and cancellations.

The dot plot map created with Microsoft 3D maps shows airport locations. On-time metrics were considered for the map. This information was already on the bar chart; thus, it was unneeded. The management team can access individual on-time performance indicators by hovering over the airport name. This information is hidden by default but can be retrieved. The bar charts developed for this project can be accessed from GitHub pages. https://robjoscelyne.github.io/data_visualisations/

Summary of tools used:

1. Jupyter notebooks using the Python library Pandas for data import, cleaning and exploration.
2. Visual Studio and npm server for D3.js code used for the bar chart.
3. Microsoft Excel 3D Maps for the dot plot map.

IV. Conclusions

Rotating the y-axis label from vertical to horizontal would enhance the readability of the bar graph. The same procedure might be applied to diagonally positioned x-axis labels to conserve space. Studies on saliency have demonstrated that a reader's attention is drawn to the upper left corner of the page. Currently, the company logo appears in the upper left corner, while the summary text appears in the lower right. From section III, it was established that a reader’s attention would start at the top left of a visualisation. Therefore the positions of the summary sentence and logo could be switched, so the reader views the concluding statement first. To decrease clutter even further, the y-axis may have been eliminated. The values for delays and cancellations would be placed on the bar columns. These values would be modified as the user toggles between delays and cancellations.

The dot plot map was created using Excel 3D maps. The advantage of this application was that a spatial relationship between airports could be rapidly created. However, the program's interactive usefulness was restricted. Future work may incorporate an animation of an arc emanating from the chosen airport and displaying OTP performance. The D3.js connection map method connects two points on a map. Transitions may have accentuated the arc's aesthetic appeal to users. Time constraints prevented the implementation of such a feature.

D3.js has more customisable options than Excel 3D maps, but the learning curve is steeper and needs an understanding of Javascript. A local server was required to run the bar chart in the browser. An application called npm was installed through Microsoft Visual Studio Code. These factors resulted in more time being spent developing the bar chart.

The scope of the research question could be expanded. Instead of focusing on departure airports, the analysis could have covered the arrival airports. A possible approach could have involved determining the best departure time for the return flight at three potential destination airports. The conclusion would have been to suggest using two airports instead of just one. As in figures 3 and 4, more Python code would be required to query the dataset and get additional on-time performance values.

REFERENCES

[1] Wilkerson, Jordan & Jacobson, Mark & A, Malwitz & Balasubramanian, Sathya & Wayson, Roger & G, Fleming & Naiman, Alexander & Lele, Sanjiva. (2010). Analysis of emission data from global commercial aviation: 2004 and 2006. ATMOSPHERIC CHEMISTRY AND PHYSICS. 2010. https://www.researchgate.net/figure/Flight-length-and-duration-forshort-haul-and-long-haul-flights_tbl7_267927654 [cited 02/11/22]

[2] P. Lundblad, "Third Pillar Of Mapping Data To Visualisations: Usage", QLIK blog. 2015. https://www.qlik.com/blog/thirdpillar-of-mapping-data-to-visualizations-usage [cited 02/11/22]

[3] S. Ribecca, "The Data Visualisation Catalogue", Dot map. 2017. https://datavizcatalogue.com/methods/dot_map.html [cited 02/11/22]

[4] NHS, "Colour vision deficiency (colour blindness)", NHS. 2019. https://www.nhs.uk/conditions/colour-visiondeficiency/#:~:text=It's%20a%20common%20problem%20that,and%201%20in%20200%20 [cited 02/11/22]

[5] A. Katsnelson, "Colour me better: fixing figures for colour blindness", Nature. 2021. https://www.nature.com/articles/d41586021-02696-z [cited 02/11/22]

[6] NHS, "Long-sightedness", NHS. 2019. https://www.nhs.uk/conditions/long-sightedness/ [cited 02/11/22]

[7] J. Schwiegerling, "Field Guide to Visual and Ophthalmic Optics", SPIE Press, Chapter 1: Ocular function, Visual Acuity p.19. https://spie.org/publications/fg04_p19-20_visual_acuity?SSO=1 [cited 02/11/22]

[8] S. Bradley, "3 Design Layouts: Gutenberg Diagram, Z-Pattern, And F-Pattern", Vanseo Design. 2011. https://vanseodesign.com/web-design/3-design-layouts/ [cited 02/11/22]

[9] User Testing, " 7 Gestalt principles of visual perception", Cognitive psychology for UX. 2022. https://www.usertesting.com/resources/topics/gestalt-principles [cited 02/11/22]

[10] C. Knaflic, "Declutter this graph", Storytelling data. 2017. https://www.storytellingwithdata.com/blog/2017/3/29/declutterthis-graph cited [02/11/22]

[11] C. Ware, "Preattentive Visual Properties and How to Use Them in Information Visualization", Information Visualisation. 2018. https://www.interaction-design.org/literature/article/preattentive-visual-properties-and-how-to-use-them-ininformation-visualization cited [02/11/22]

[12] A. Janes, "Mastering data visualisation in D3.js", GitHub, 2020. https://github.com/adamjanes/udemy-d3/tree/master/05/5.06 [Last accessed 02/11/22]

[13] Bureau of Transportation Statistics, " On-Time : Marketing Carrier On-Time Performance", United States Department of Transportation. 2022. https://www.transtats.bts.gov/DL_SelectFields.aspx?gnoyr_VQ=FGK&QO_fu146_anzr=b0-gvzr cited [13/11/22]