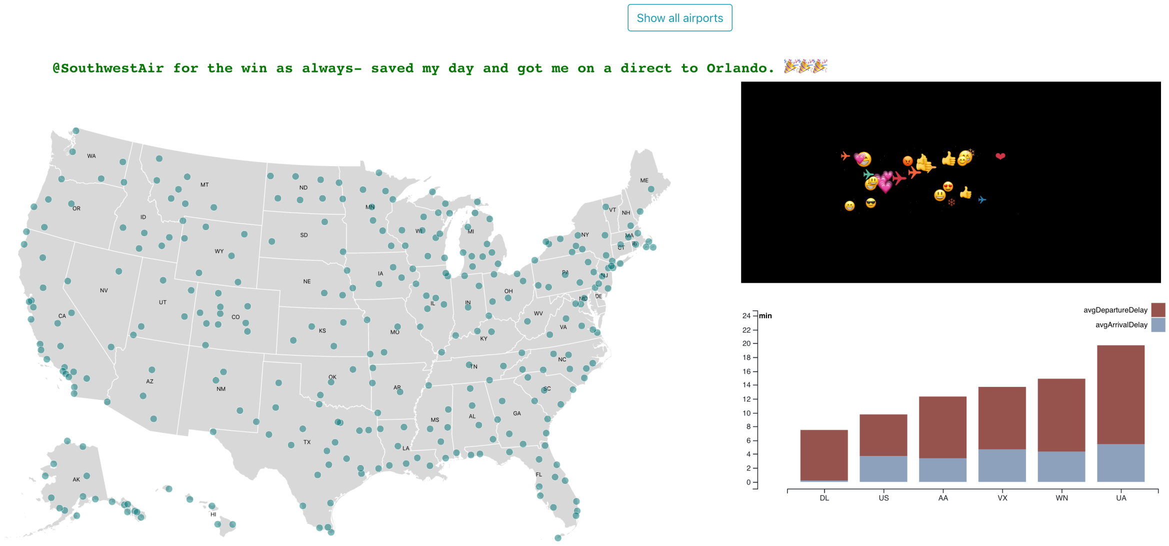
# The Map-based Airlines' Visualization with Twitter Sentiment Analysis

Yuncong Ma, Weixi Liu



@SouthwestAir for the win as always- saved my day and got me on a direct to Orlando. 🎉🎉🎉

Fig. 1: Project Overview

**Abstract**— The goal of this web-based consulting interactive visualization is mainly to display statistic inform for the past airlines by combining three kinds of visualization patterns, and it provides a chance for audiences to explore the potential useful inform throughout the vis patterns. At the meantime, we combined the advantage of the React.js with D3 to creative interactive visualizations and using the Amazon Website Service to deal with big data problem (instead of uploading the whole large dataset along with website, we store the data into MongoDB and transport information via AWS). This vis is consisted of a US national map, of which circles represent different airports that are distributed across the entire country. The position of each airport is located according to the longitude and latitude data coming from us.json file. While the users move the mouse over the map, the chosen state or airport will be highlight and the more corresponding detail inform will pop up. As a extend practice of assignment 4, there are another two coordinated views sited beside the main map view—a stack bar chart and a word cloud. The stack bar chart is used to display the statistical data of the degree of flight delays among six different airline companies. Since the passengers' feedback is the second half of the vis topic, the word cloud will present those key words of which the appearance has highest frequency.

**Index Terms**—Sentiment Analysis, Interactive Visualization, React, Amazon Website Service, MongoDB, Word Cloud.

## 1 Introduction

As a major public transportation, airline market has been ushering a booming development. On a global scale, a continuous world-wide growth of air traffic could be observed, and according to several market researches, the growth is expected to maintain positive rates up to 2030.

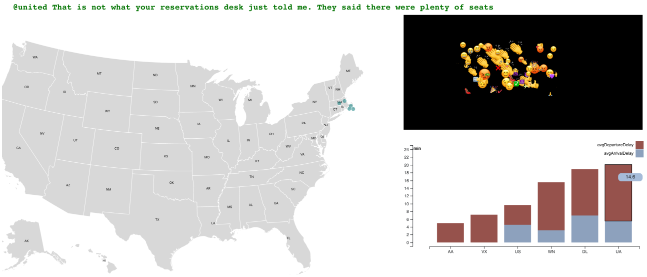
However, there are many factors that affect the performance of the commercial aviation system, which can lead to annoying results to their passengers sometimes. Given the uncertain factors of the whole aviation system, passengers usually have to plan their travel many days or even months before the departure date. Meanwhile, in order to decrease the trip costs, avoid the rush traffic hours, and then obtain a relaxed travel experience, travelers also hope to gain as more detailed information as possible.

Converting the traditional numeric information into a more vivid visualization form, could help the viewers gain their desired information efficiently and easily. So we intend to build a map based interactive consulting visualization, which combines two date sets coming from the US Department of Transportation Bureau of Transportation Statistics. We hope this application can reveal some potential patterns under the flight records and display them to the viewers.

We apply D3.js library to build the whole data visualization including one map view which is used to depict the airports and the airlines. While users move the mouse over an area belonged to a specific state, the area will highlight of which the color change and the name of that state will pop up. Once users click the state, it will filter out other states, that only displays the airports sited inside the chosen state. For sure, we also provide a button placed at the top middle of the web page, to reset the map view to its original status. Correspondingly, the right-side bar chart will change once users click a specific state. The

- *Yuncong Ma is with Worcester Polytechnic Institute. E-mail: yma10@wpi.edu.*
- *Weixi Liu is with Worcester Polytechnic Institute. E-mail: wliu6@wpi.edu.*

*Manuscript received 31 March 2014; accepted 1 August 2014; posted online 13 October 2014; mailed on 4 October 2014.*
*For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.*



@united That is not what your reservations desk just told me. They said there were plenty of seats

Fig. 2: A snapshot of an interaction.

bar chart we use here is to display a comparison of fight delay period among at most six airline companies. This bar chart is a variety of the plain bar, which is divided horizontally into upper and lower part, each of them present the degree of departure and arrival delay respectively. Initially, this stack bar chart will display average delay time across all the flight data. Of course, to make the bar chart more practical, we also append the numeric value of the flight delay time while users move the mouse over a specific part of the bar chart. Meanwhile, the word cloud view will change as a reaction of the click action on the stack bar chart, the key words it shows every time will change correspond with the content of passengers' tweets. Most of the words are the reflection of the feel of the passengers, either is positive or negative.

## 2 Background

As the time goes by, the use of coordinated multiple views has been changing and expanding a lot, in addition it also becomes part of larger sense making environments where the techniques are being used to analyze large datasets, integrate alternate viewpoints, and generate nuggets of information.[5] Nowadays, D3.js library is one of the most popular tools to implement the coordinated multiple views and then analysis large data set. It is worth and quite practical to apply these ideas and tools while building the final project. Meanwhile, data visualization is not just a way that simply transforms the data into several tables or charts, instead, it also involves pre-processing on data such as clearing, filtering, mapping or other aggregation operation. The process that chose appropriate visualization pattern with the dataset is challenging, but on the other hand, a good vis always provides its audiences an intuitive and logical experience. Utilizing all the handy techniques we have to develop an extension of the previous project, is a good study path for our future work.

There are three important parts: designing the whole visualization, fixing the big data problem (our original dataset contains 300M+ rows), achieving the interactions between visualizations with the help of d3 and react instead of using dispatch.

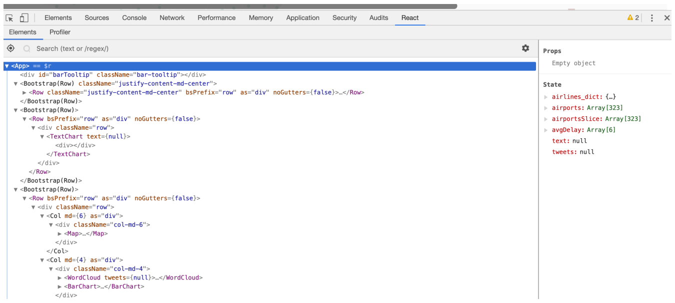
## 3 Method

Basically we used D3.js to build each data visualization including US Map, Stacked Bar Chart, Word Cloud.

### 3.1 Interaction

People usually used dispatch which is a tool to create interaction between data visualization views to create interactions, we decide to use React.js to do the same thing instead of dispatch.

React is a JavaScript library for building user interfaces, the core idea behind it is integrating separate parts into a whole thing (class) and controlling each part's state. For example, we can create a simple spinning button by combining a button element, a spinning figure, and a state variable to control spinning or not. We embed the three things into a class SpinningButton, we set the spinning state with false as default, when user click the button, setting the state with true, showing the spinning figure and starting up a timer function as the same time, when time running out, set the state with false back, hidden the spinning figure.

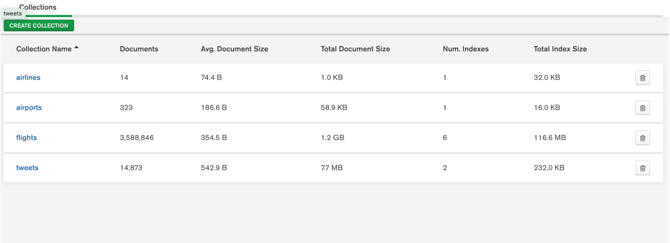


```
▼ <App> == $r
    <div id="barTooltip" className="bar-tooltip"></div>
  ▼ <Bootstrap(Row) className="justify-content-md-center">
    ▶ <Row className="justify-content-md-center" bsPrefix="row" as="div" noGutters={false}>…</Row>
    </Bootstrap(Row)>
  ▼ <Bootstrap(Row)>
    ▼ <Row bsPrefix="row" as="div" noGutters={false}>
      ▼ <div className="row">
        ▼ <TextChart text={null}>
            <div></div>
          </TextChart>
        </div>
      </Row>
    </Bootstrap(Row)>
  ▼ <Bootstrap(Row)>
    ▼ <Row bsPrefix="row" as="div" noGutters={false}>
      ▼ <div className="row">
        ▼ <Col md={6} as="div">
          ▼ <div className="col-md-6">
            ▶ <Map>…</Map>
            </div>
          </Col>
        ▼ <Col md={4} as="div">
          ▼ <div className="col-md-4">
            ▶ <WordCloud tweets={null}>…</WordCloud>
            ▶ <BarChart>…</BarChart>
            </div>
```

```
Props
  Empty object

State
▶ airlines_dict: {…}
▶ airports: Array[323]
▶ airportsSlice: Array[323]
▶ avgDelay: Array[6]
  text: null
  tweets: null
```

Fig. 3: React states example.



| Collection Name ▲ | Documents | Avg. Document Size | Total Document Size | Num. Indexes | Total Index Size | |
|---|---|---|---|---|---|---|
| airlines | 14 | 74.4 B | 1.0 KB | 1 | 32.0 KB | 🗑 |
| airports | 323 | 186.6 B | 58.9 KB | 1 | 16.0 KB | 🗑 |
| flights | 3,588,846 | 354.5 B | 1.2 GB | 6 | 116.6 MB | 🗑 |
| tweets | 14,873 | 542.9 B | 7.7 MB | 2 | 232.0 KB | 🗑 |

Fig. 4: Dataset in MongoDB.

As you can see, we can use this kind of idea to control our visualization information and transport the information by react states, the figure 3 shows our state.

### 3.2 Big Data Problem

As we mentioned before, our original dataset contains 300M+ rows, the data size is above 600MB, it is definitely not a good idea to store the data along with the website. Instead of uploading the dataset, we create a MongoDB (a NoSQL database) database to store our data. There are two reasons we choose MongoDB, the first reason is it is NoSQL data structure as JSON format, it's convenient to I/O data. It has a good compatible with Amazon Website Service which is also good for transporting data. Of course, MongoDB is also good for aggregation operation which help us to retrieve different kind of query combinations.

With the help of MongoDB and AWS, we don't need to uploading the whole large dataset which could make the website run slowly. On the other hand, we can keep the whole dataset without any pruning and modification which could lead information loss.

We can see in figure 4, our flights collection is up to 1.2G which is super large.

## 4 D3 Transition

When the data changes in a d3 views, what is the interesting thing we may notice? That is transition. It's boring if a data visualization just change their view by refreshing the whole view, that is deleting the whole dataset, and replacing with new dataset. D3 provides a really good way to deal with this problem, it's called, enter, update and exit as shown in figure 5. It's super useful and interesting part of D3.js and also it's hard to understand, as lease we spent a lot of time to understand these kind of concepts. Generally, the processes are, supposing our view has been implemented. Now, we need to update the view by the new dataset. First, the D3 collects the previous dataset and new coming dataset, and secondly, the D3 finds the common part and different part (we could or we better specify which data field we are going to compare ), and lastly, the D3 update the different part and remove the useless data. From these procedures, we could efficiently and perfectly control the transitions.

## 5 Results

With the help of D3.js, React.js, MongoDB and Amazon Website Service, we successfully create our whole interactive data visualization.

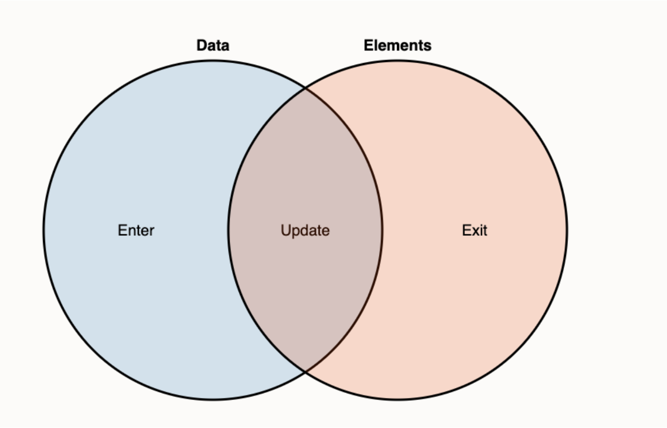


Fig. 5: D3 Transition example.

Our visualization shows the average airline departure delay time, the average airline arrival delay time and Tweeter user's sentiment. User could click any state to show the information corresponding the all airports in this specific state. User also could click the stack bar chart to show each specific airline's corresponding information.

## 6 Discussion

It gives an idea that how to depict a visualization of adjacency relations in hierarchical data, especially with a huge dataset. [4]

They presented a visual analysis of Twitter time-series, which combines sentiment and stream analysis with geoand time-based interactive visualizations for the exploration of real-world Twitter data streams. [1]

It introduced a novel visualization called NodeTrix.[3]

It restyles many useful and powerful d3 data visualizations.[2]

### 6.1 Conclusion

This paper is mainly trying to find a way to create a interactive visualization by combining React.js and D3.js. The visualization is also helpful for people who want to know what is the average delay of each airlines and what is the people's attitude by presenting Tweeter's sentiment information.

It also find a way to deal with big data problem when the dataset it's too large to upload to website.

Moreover, it gives a good example to use D3 transition for improving the visualization's quality.

At the end, we find that combining React and D3 is a good way to develop interactive visualization, it's efficient, easy to control and extensible. Although our current visualization's interaction is not fast, that is because the transporting problem between the AWS and MongoDB, and the dataset is too large to do complicate query operation like finding all given airline in some airports. That is nothing to do with the React and D3.

In the future, we could add more views to allow user to discover more information. For example, we could add a timeline to find the delay information according the time range. We also could add a functionality that user could click any two of airports to see the corresponding information between the two airports, or click any two of us states to show corresponding information.

## Acknowledgments

We would like to acknowledge Professor Lane Harrison for his insight in helping come up with ideas on how to build the visualization and the guidance provided throughout the project.

## References

[1] M. Hao, C. Rohrdantz, H. Janetzko, U. Dayal, D. A. Keim, L.-E. Haug, and M.-C. Hsu. Visual sentiment analysis on twitter data streams. In *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 277–278. IEEE, 2011.

[2] J. Harper and M. Agrawala. Deconstructing and restyling d3 visualizations. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 253–262. ACM, 2014.

[3] N. Henry, J.-D. Fekete, and M. J. McGuffin. Nodetrix: a hybrid visualization of social networks. *IEEE transactions on visualization and computer graphics*, 13(6):1302–1309, 2007.

[4] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on visualization and computer graphics*, 12(5):741–748, 2006.

[5] J. C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007)*, pages 61–71. IEEE, 2007.