Visualization of Domestic Flight Performance in the United States Process Book

Danyang Chen, Wenwan Yang, Wenshuai Ye

May 4, 2015

1 Overview and Motivation

With the significant increase in the number of flights each year, more and more passengers get stuck at the airport due to delay. According to the investigation conducted in 2014, the aggregate minutes of delay in reaching final destinations amount to over 80 millions. Inspired by this situation, we desire to identify airports and airlines of best and worst performance so that people can choose airlines based on their needs. In this process book, we will cover detailed project objectives, description of our data, exploratory analysis we have done, our design evolution, implementation, and improvement and future works.

2 Project Objectives

At first, we aimed to answer the following questions through data visualization:

- How much time will the flights be delayed at a specific airport?
- How much time will the flights be delayed for a specific airline?
- Whats the most on-time airport in the United States?
- Whats the most on-time airline in the United States?

As we started to dig into the data and implement our designs to working prototypes, we found that our visualization can address more questions and gain insights on:

- How are airports distributed in U.S.?
- The comparison of the performance among airports and airlines.
- Will airports with long departure delay always be associated with long arrival delay?
- Are there any airports or airlines that always depart or arrive earlier than scheduled? (Surprisingly, the answer is yes. The details are introduced in the Evaluation section)

3 Data

There are three types of data used in the visualization: topological json data loaded to construct a US map, airport location data used to map the cities, and detailed flight data for exploration and main visualization.

a **topological json data**
us.json: This is a json file containing the us map info. The original file was downloaded and extracted from United States Census Bureau[1] as an shp file. With topojson installed, we converted the file to a json file, which could then be directly imported to javascript.

b **airport location data**
airport.csv: This is a csv file containing the latitude and longitude of each airport uniquely identified by IATA / FAA code, which could be used as a key to merge with flight data files. We downloaded the file from a third party website called Open Flights[2].

c **detailed flight data**
*ontime_*2014*xx.csv*: These are csv files extracted and downloaded from Bureau of Transportation Statistics, United States Department of Transportation[3]. Each file contains information on year, month, flight date, carrier, airline id, flight number, original airport id, original airport, original city, original state, destination airport id, destination airport, destination city, destination state, departure time, departure delay time, arrival time, arrival delay time, carrier delay time, weather delay time, security delay time, late aircraft delay time, and national aviation system (NAS) delay time of flights. Based on efficiency, we only investigate flights operated in 2014 for this project. *ontime_avg_*2014.*csv*: Because we are only interested in the average performance of flights given a certain year, month, carrier, departure airport, and arrival airport, the *ontime_*2014*xx.csv* files are read in to calculate the averages of delay times in python. We merged the 12 averaged output files into one and write out the *ontime_avg_*2014.*csv* file.

4 Exploratory Data Analysis

Map Visualization

We made a draft map with airports scattered on it as nodes initially to observe how the map fits to the screen and how the nodes fit to the map. It turns are that there are too many airports (21490), as shown in figure 1. Since we are only interested in the airports with available flight data and airports that could be located in the United States map we generated, we filter out the majority of them, which in turn reduces airports to 318.

We then calculated the total number of routes and found that this number equals 73544, confirming that we do need to remove airports with empty routes on the map. The routes are represented as links (edges) that connect two nodes on the map. For each airport, as a result, there are multiple outbound links that point to their corresponding arrival airports. An adjacency list proves to be an efficient data structure to accomplish this. Namely, given an airport, we would create a list that stores all the airports with direct flights from this airport.

[1] www.census.gov/geo/maps-data/data/cbf/cbf_nation.html
[2] openflights.org/data.html
[3] www.transtats.bts.gov

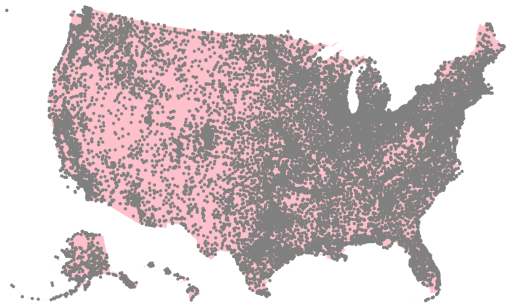

Figure 1: US Map

Bar Chart Visualization

The bar chart shows the average delay times by airports, and the user can interactively filter the airports by states and routes. We use horizontal bar chart, which creates better visualization given the space and location in our framework. Due to the shear volumes of airports in the United States, the height of the bar chart might exceed the height of svg. To prevent this, we add a slider so that user can scroll up and down to view the bars.

Line Chart Visualization

While the bar chart aims to display information on the delays by airports, the line chart allows users to dynamically visualize the delay times by airlines and delay reasons. Since the data are based on the current observed airports on the map, we need to aggregate and wrangle the data each time we switch observed airports. Because the number of airports and routes are non-trivial, it might be inefficient if we use a list to look up airports. Instead, we can use a set, which reduces the complexity from linear time to constant time.

With exploratory analysis, we envisioned a framework as displayed in figure 2

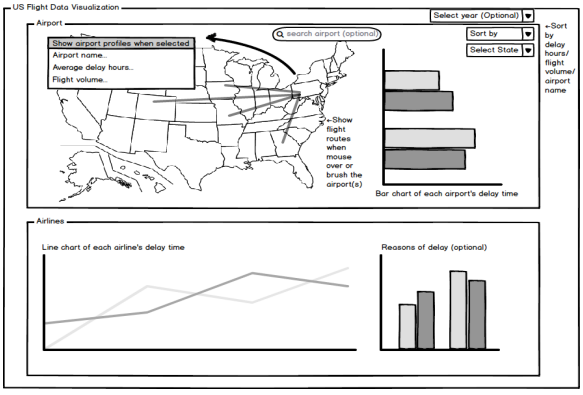

Figure 2: Mockup

5 Design Evolution

As we started our implementation, we came up with more and more ideas on designing the layout and displaying the information of the visualization. The section illustrates our thoughts, some of which have been implemented, whereas the rest are unimplemented alternatives due to various reasons such as complexity and compatibility.

5.1 Implemented Features

Show the routes connecting to the airport when the mouse is over that particular node Inspired by our second homework on various design layouts, we can add interactions dynamically displaying information after scattering nodes on the map. When the user moves the mouse over a particular node, the information regarding this airport will be displayed. In addition, all the links connecting to this node will appear on the screen.

Update the bar chart and line chart when clicking on a particular node We hope to figure out a way to update the bar chart and line chart accordingly. Initially, we implemented this update in a way that when the user moves the mouse over a node, the update would occur. This proves inefficient because with such considerable number of nodes on the map, a user might accidentally put the mouse over several nodes. In this circumstances, the program would perform calculations to update both bar charts and line charts futilely. As a result, we design the interaction in a way such that the line chart and the bar chart will be updated only when the user clicks on a particular node (airport). The program then recalculates the delay data based on the current observed airports and all airports connecting to it. Consequently, The bar chart will only display airports of interest.

Search for a route Suppose a user is only interested in a particular route. To satisfy these users requirements, we provide them with an interface to search for a route of interest by entering the departure airport and arrival airport. The line chart and the bar chart will be updated accordingly.

Filter the bar chart by states Due to the considerable number of airports gathered, the length of the bar chart might result in a less user-friendly interface. To address this issue, we provide a dropdown menu so that the bar chart itself can be filtered by states. after selecting the state of interest, the bar chart will only show the airports located in the states selected and filter out all others.

Sort the bar chart To better answer the questions on the performance of airports, we add the sorting functionality. with this dropdown menu, users can sort the bar chart by airport names, departure delay times, arrival delay times, and flight volumes of the airports.

Change the line chart by the cause of delay As we all know, the delay of a flight can result from various issues such as weather, traffic control, maintenance, etc. To inform users of delay reasons, we add radio buttons above line charts allowing them to change the delay categories to the ones they are interested in. The total delay is defined as the sum of departure delay time and the arrival delay time.

5.2 Alternative Features (Not Implemented)

Add the data of the economic status of each airline and visualize the relationship between the economic status and the average delay time on the right corner. This is the feature followed by visualizing data by different states. After we compared the average delay time by states, we discovered that some states always have longer or shorter time than other states. The economic status of the states might be a factor for that because the richer the state, the busier that the airport would be because traffic controllers might face more challenges. Alternatively, we could interpret it as the poorer the state, the poorer facility the airports are equipped, which in turn results in more delay. We later decided not to implement this feature since statistical models might answer these questions more effectively than visualizations.

Inform users to decide what / when / where to fly With the data and functionality we have gathered and created, we can make some general recommendations on the best airline, best travel seasons, and the most likely delay reasons by popping up a box that displays airlines that meet these standards. However, users could actually figure out this information based on the interactions we have already implemented. As a result, this feature was discarded as well.

6 Implementation

6.1 Implementation before Milestone 1

Till this point we had finished data cleaning and preparation, and implemented a working prototype of our design based on the real data. As shown in the following screenshot, on the left corner of the page lies a map of the United States, where each gray dot represents an airport. The bar chart on the right of the map shows the average departure (in blue) and arrival (in pink) delay time of each airport in minutes. The user can sort the airports by name, departure delay time, and arrival delay time. In the design studio of the peer review, we received the feedback that there are too many airports and the bar chart is too long to navigate, as stated in the exploratory analysis. We adopted their advice and added the select state filter menu to improve the user experience by allowing users to select a specific state and clearly see the performance of the airports in which he or she shows interest. The line chart below the map shows the average delay time in minutes of each airline. Each line represents an airline. The gray block on the right is an optional feature to be implemented: to show bar chart informing the cause of delay.

6.2 Improvements and Feature Work before Milestone 1

We will continue to implement our design, complete the line chart of airlines to add label denoting the objects, and enable interaction and animation between the multiple charts. For example, when the user clicks an airport, the corresponding point on the map will be highlighted, the flight routes connecting to that airport will be shown, and the airlines line chart will also be updated to only show the per airline information in the selected airport, etc. We have received several feedbacks from both our TF and the peer review session. We will also work on the suggestions and improve our design.

- Find a way to make the visualization clearer. There are too many airports both on the map and in the bar chart. Instead of plotting all of the airports, we are thinking to only show selected airports with significant traffics.

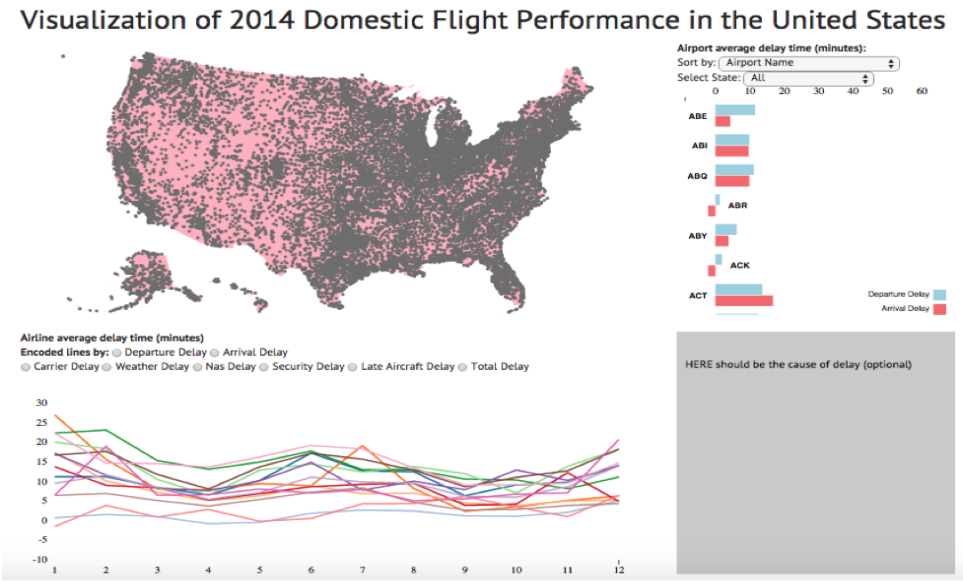

Figure 3: Visualisation

- Add comparison to the mean. We can add the overall average delay time as reference in both the bar chart and the line chart, so that when the user looks into the details, he or she will not lost the general picture and compare the airport or airline to others average performance.

6.3 Implementation after Milestone 1

After implementing static structure, we desired to create interactions that allow users to dynamically visualize information, as illustrated in the exploratory analysis section.

Before tapping into any dynamic interactions, we filtered out the majority of airports by subsetting to those with available flight data. This procedure cleans the map by dramatically reducing airports. In addition, instead of implementing the optional recommendation feature introduced in the exploratory analysis, we extend the bar chart to the bottom right corner.

With framework established, we started building interactions within each module and between various modules. To construct within-module interactions, we used an *updateVis* prototype function in each javascript module so that the map, the line chart, and the bar chart can be updated based on the user behavior by executing these functions with newly wrangled data.

Within-module Interactions

Map: When moving the mouse over a node, links are added to represent routes from this airport to every possible arrival airport. Additionally, because users can be overwhelmed by the shear volume of the nodes, we added a search function for users to search for the airports they have interest.

Bar Chart: The feedback from peers points out that our bar charts might be too long to extract any useful information. To tackle this issue, we attached drop down menus that allow users to filter bars by states and sort them by different values. With these, we can answer questions such

as What is the most on-time airports?. Moreover, average delay dash lines are added for baseline comparisons.

Line Chart: Line chart displays the average delay times by airlines and months. If a user switches delay time categories, the *wrangleData* function will recompute the average delay times based on the selected categories. When the user moves the mouse over a line, the line will be highlighted and the associated airline name will pop up.

Between-module Interactions

With framework established, we could create event handlers that bind various modules together. For example, the user might only be interested in the data of some airports. To accomplish this, we add an event handler to bind all three modules together. When the user clicks on a node (This works only when the mouse has changed to a hand-like pointer.), the bar chart and line chart will display that airport and all airports with links connecting them only. Likewise, when the user move the mouse over a bar, it triggers the map to update links that originates from that airport. Last but not least, the search function will trigger such updates as well by filtering out the information on line charts and bar charts irrelevant to the two airports of interest.

7 Evaluation

7.1 Evaluation before Milestone 1

The first data visualisation is a US map with each dot representing an airport. As we can see from the US map, the eastern part of the United States has more airports than the western part. and the costal city has more airports than the inland city.

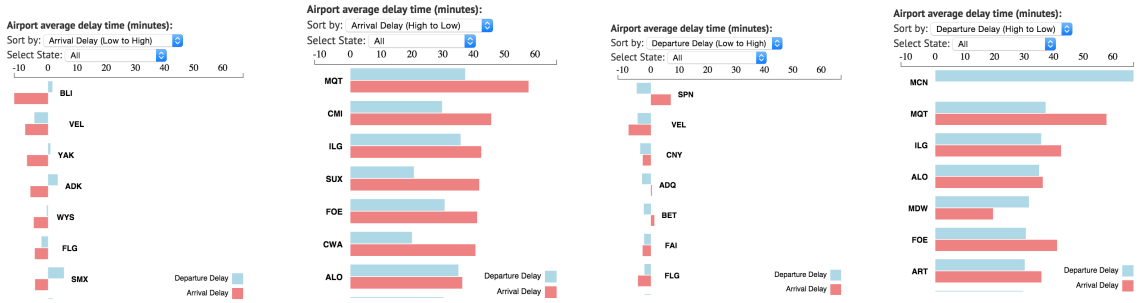

Figure 4: From left to right, shortest arrival delay, longest arrival delay, shortest departure delay, longest departure delay

The second data visualization is a barchart comparing the airport average delay time. *MQT, CMI, ILG, SUX, FOE* are the five airports that have the longest arrival delay.

Likewise, from the bar chart, we find that *BLI, VEL, YAK, ADK, WYS* are the five airports that have the shortest arrival delay.

From the other two figures, we also notice that *MCN, MQT, ILG, ALO, MDW* have the longest departure delay and *SPN, VEL, CNY, ADQ, BET* have the shortest departure delay.

The line chart compare the total delay time of different airlines. *AS* has the shortest delay time. *B6, F9, MQ, OO* usually have longer delay time than other airlines.

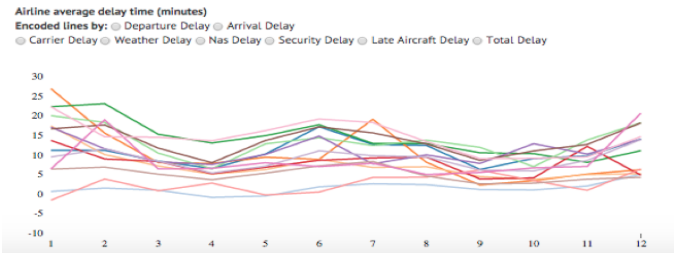

Figure 5: line chart

7.2 Evaluation after Milestone 1

With interaction features added, we discover a lot of informative results that can answer most of the questions we raised at the beginning. When opening the webpage, the first impression we get is how the airports are distributed in the United States. From observation, we notice that there are more airports in the coastal cities than the inland cities in general, as shown below. This is not surprising for various reasons. For example, costal cities might have larger demand on flights because they generally attract international business more than inland cities do. Below shows a more detailed description on what / how our visualization product can tell us more useful information.

This map filters out irrelevant airports with colors changed to more user-friendly ones. Then, looking on the bar chart on the right side enables us to answer airport related questions such as which airport is the busiest, which one is the most on-time, and which one has the largest average delay time. As shown in the snapshot, *ATL* is the busiest airport in the United States. It is not surprising as *ATL* is one of the biggest airports in the world.

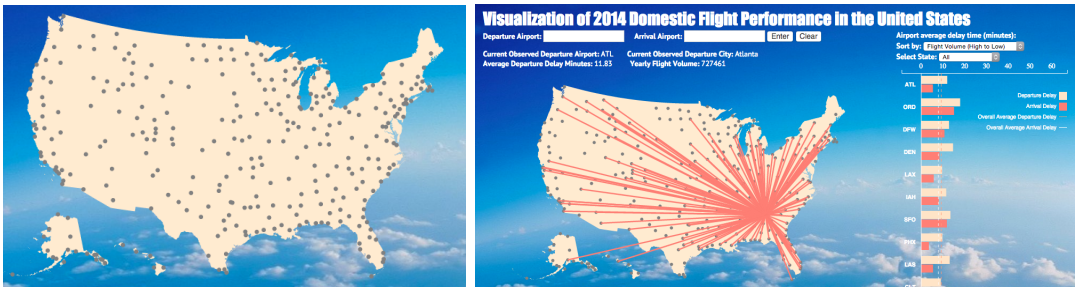

Figure 6: Bar Chart Examples

VEL and *ADK* are the most on-time airports in departure and arrival time respectively. It is also impressive that their average departure/arrival time are actually earlier than scheduled.

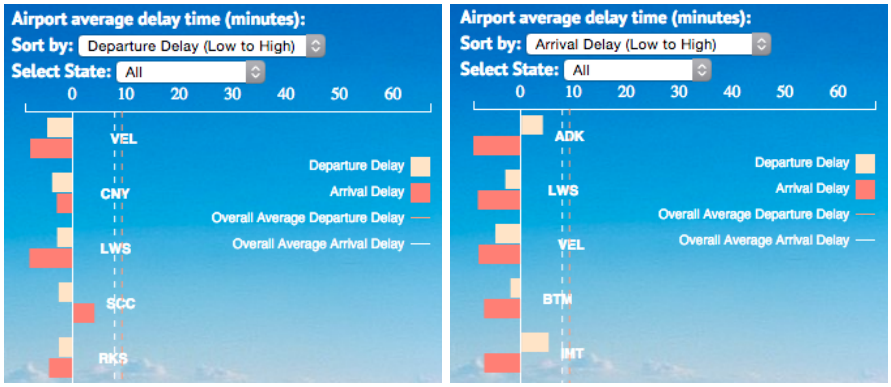

Figure 7: Bar Chart Examples

In contrast, *MCN* and *EGE* have the largest departure and arrival delay times. However, they dont have many flights.

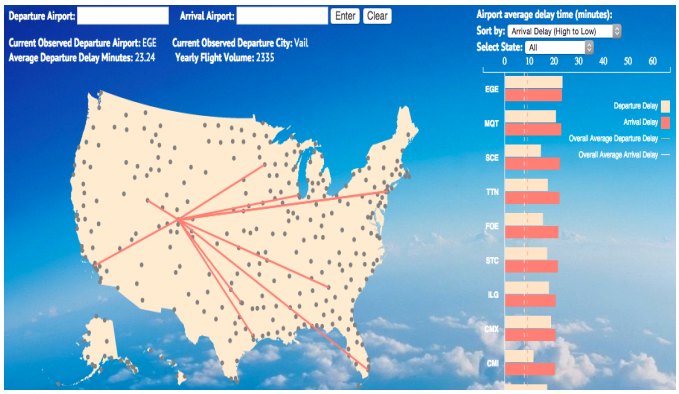

Figure 8: Bar Chart Demonstrations

The line chart shows the performance of the airlines. In general, Southwest Airlines is one of the airlines that have the highest average departure delay. And Frontier Airlines has the highest average arrival delay.

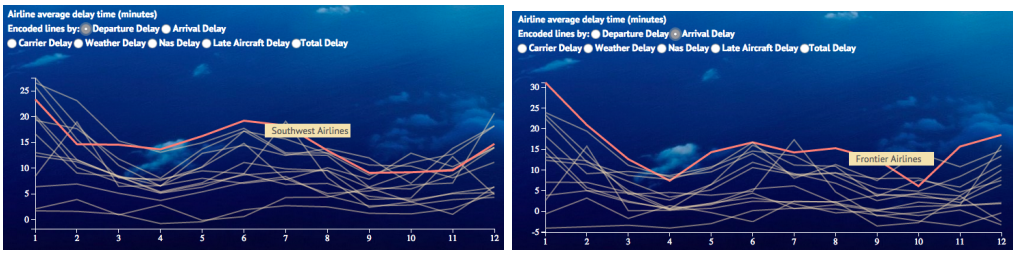

Figure 9: Line Chart Examples

On the other hand, Alaska Airlines shows the best performance as it has the lowest average time for both departure delay and arrival delay.

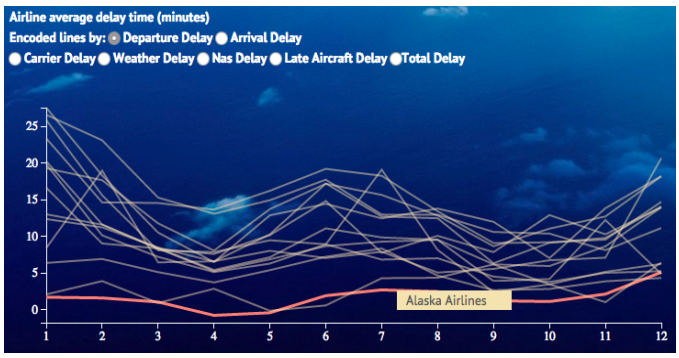

Figure 10: Line Chart Examples

Besides, for both departure and arrival delay, the delay time of most the airlines are much higher in January. It makes sense because the weather in January tends to be much severer than others. This phenomenon is more obvious when we look at the delay time caused by weather. We can also click on one airport in either the map or the bar chart, to see airline performance for that specific airport. Moreover, if we are interested in taking a flight from some place to another, we can input the airports in the searching box, and the specific flight line will be shown. For example, search *BOS* as the departure airport and *SFO* as the destination.

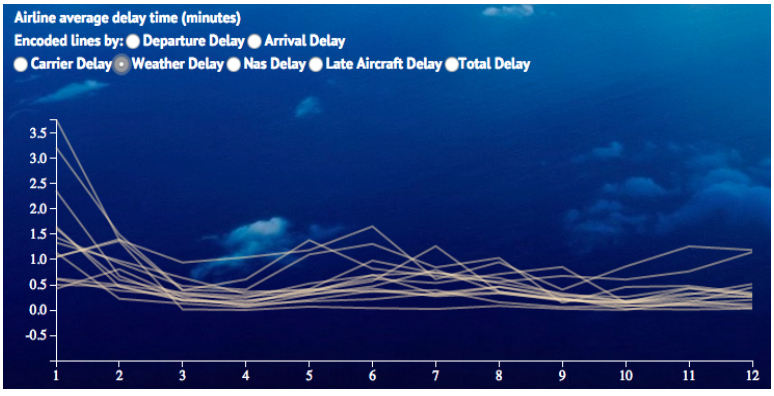

Figure 11: Line Chart Examples

The visualization shows that we should expect a departure delay from *BOS* for about 10 minutes, and an arrival delay at *SFO* for approximately 13 minutes.

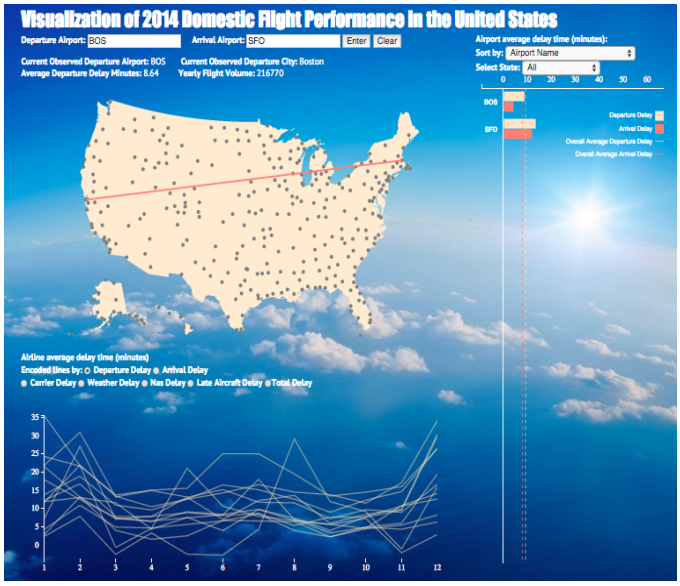

Figure 12: More Examples

We can also check the line chart to look for a suitable airline with better performance. If we are travelling in January, then we should avoid booking the tickets from Delta Airlines as it has the highest delay time caused by the severe weather.

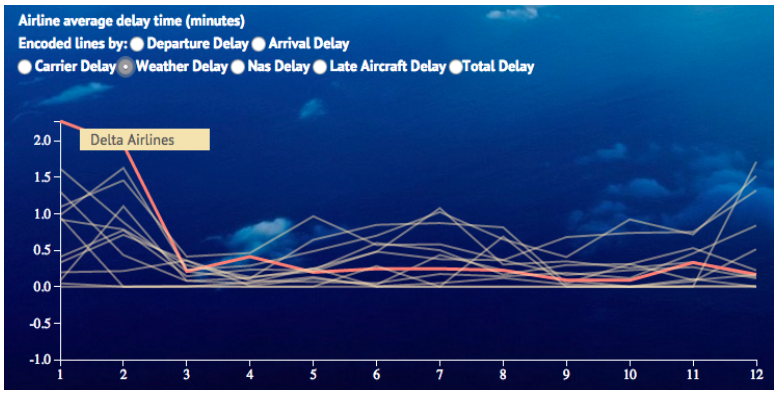

Figure 13: More Examples