Final Project Documentation
CS 465: Information Visualization

Code enclosed and available at https://github.com/danasilver/flighty.

- Who worked on the project
Dana Silver and Maya Najarian

- How to deploy the project
The page is entirely static and can be run as is with a simple web server. It does, however, depend on two RESTful APIs: Flightstats (https://developer.flightstats.com/api-docs/) and Hotwire (http://developer.hotwire.com/docs/). API keys are hardcoded into the client-side application (not best practices, but it makes deployment easier right now).

Neither of the APIs allow CORS requests, so we wrote CORS enabled wrappers around each API. They are deployed to flightstats-api.herokuapp.com and hotwire.herokuapp.com, respectively. These are always online and should be available with little downtime. If you want to run the servers yourself, the code is available on GitHub at github.com/danasilver/flightstats and github.com/danasilver/hotwire. To get set up (example for flightstats):

Install Node.js and npm dependencies, then

```
$ git clone git@github.com:danasilver/flightstats.git # SSH
$ cd flightstats
$ npm install
$ node app.js

Listening on 3000. # Available at localhost:3000
```

- Where the data came from
We used three sources of data to create our final project. The first source was Mike Bostock's airports slide from a presentation on d3 (http://mbostock.github.io/d3/talk/20111116/airports.html). From this presentation we extracted two datasets: one containing popular domestic airports with IATA codes and coordinates; one containing flights with origins, destinations, and number of flights. This dataset was used to plot the locations of frequently used airports onto our projection of the United States and size the circles used to represent the airports based on the number of flights originating from that airport. The second source of data was the FlightStats API, documented at https://developer.flightstats.com/api-docs/. This was used to find real-time flights from the origin airport chosen by the user. Finally, we used the Hotwire API (documented at http://developer.hotwire.com/docs/) to display the historic price data for the specific route selected. The two APIs require keys to use, which are hardcoded into the application.

- **What you did to transform the data into a usable state**
We originally thought we would pull the historic flight data, including prices, from many disorganized government Excel sheets. We spent a fair amount of time cleaning up (by hand and programmatically) quarterly historic flight data from the United States Department of Transportation's Research and Innovative Technology Administration, despite not ultimately using it. You can see this data in our Git history at https://github.com/danasilver/flighty/tree/d74bf07bf4b5b6c7799aef7103dfaafa7070971b/data/flights/csv.

We transformed the two datasets from Mike Bostock's slides into one JSON file using a Node.js script. We did not need the flight paths as we were plotting live data, but wanted the number of flights originating at each airport to size the circles that represent each airport.

We did not need to heavily transform the data used in our final project as the airport dataset was already in usable form, and the API's required only constructing an appropriate request for the data.

- **A description of who the visualization is for and what questions it is designed to answer**
Our visualization is designed for a user looking to take a spontaneous vacation, or anyone interested in flight trends based on route, price, or time of day. It is designed to answer the following questions: What flights are departing in the next 6 hours? Are there flight path trends for certain airports at certain times of the day? What are the previous price trends of a specific route? Should I buy now or wait a few months, based on historic data?

- **A description of the visualization including reasons for the encodings and interactions you chose. A description of any approaches you contemplated or tried and then rejects would be appropriate as well.**
This visualization allows the user to interactively see real-time flight data originating from a single airport. The user can dynamically select an origin airport from the select menu in the top right-hand corner of the visualization. Once an airport is selected, flights originating from that airport in the next 6 hours are displayed by destination airport in the Flights panel to the right of the map. The possible destination airports are highlighted in red and the user can mouse over the destinations on the map or scroll through the listing to see the name of the airport. The user can then select a destination airport either by clicking directly on the map or alternatively on the listing for that airport. At that point, the specifics of the flights along this route are displayed and the historic price data of this route (if available) is shown in a line plot displayed in the bottom right-hand side of the visualization. The user can then easily navigate to another destination by clicking directly on the map or using the flight listing to see the possible airport codes once again. It is also very easy to choose another origin airport, if that is desired.

We chose to encode the number of outgoing flights for each airport in the size of the airport circle using a square root scale so more popular airports would be distinguishable and it would be easier for the user to differentiate between airports in the same area based on relative size. We chose green and red for the respective encodings of origin and destination as a corollary to go and stop, with all other airports a neutral blue. Though it is not perceivable for many of the flights, the paths connecting the airports are great arcs which match the curvature of the earth.

We primarily use select, filter, and dynamic queries as our methods of interaction as the sheer number of flights in a six hour period is too much for a user to grasp all at once. Filtering flights first by origin, then by potential destinations, we have a logical way to only show users data that is pertinent only to the current step of planning a trip. First you choose an origin, then a destination, then a flight. Only showing a limited amount of data at once allows us to provide the details off the chart in the Flights panel that is linked to the map. Hovering over a potential destination in the Flights panel highlights that destination on the map. We chose not to incorporate the reverse mapping as the destination on the map may not be visible in the panel and it would annoy a user to have the choices scrolling up and down if they had already scrolled to a specific one. It's also unnecessary, as we show the hovered or clicked destination above the panel.

- As assessment of the success of your final project. What works, what doesn't work, what changes you would make if you were to do this again.
This visualization effectively allows the user to explore and interact with real-time flight data. It is very easy to use and provides useful information for the user. One possible change would be to allow the user to choose an origin airport by clicking directly on the map. Another improvement would be to find a more complete dataset corresponding to the historic price data for specific flight routes. Currently the Hotwire API only has this data for a limited amount of routes. Ideally, we would like to have data for all possible flight paths.