Chenyu Li (cl2337) Curtis Josey (cvj3) Wentao Li (wl553)

03/07/2016

Data Visualization: Unidentified Flying Objects (UFO) Sightings

Overview

The goal of this data visualization is to better understand the reported UFO sightings in New York State, without having to look at boring statistical results. In order to gain some perspective, a brief comparison of New York State to all other states in the United States will be presented. After viewing the completed project, the viewer will understand the best day of the week, month of the year, county to watch for UFOs. Due to the potential for UFOs to be of terrestrial origins, a relationship between high traffic airports and sightings will be made available. Further, the viewer will understand the most commonly observed shapes and the general classification of all UFO shapes.

Data Description

Data Sources

For this project, data was retrieved from four different sources. First, the *National UFO Reporting Center* website was used to obtain reported UFO sightings for New York State, including: location (City), duration, shape of UFO, date / time and a brief text summary of the incident. Second, a list of primary commercial airport locations in New York State and their respective enplanements (passenger boardings), was obtained from *Wikipedia*. Lastly, the data correlating Cities to County (plus Latitude and Longitude) was drawn from a mixture of Mapquest and Google Geo services and then cached in a local file.

Data Acquisition, Verification and Integration

The UFO report data was extracted from the raw HTML file, via JavaScript (Appendix A). Unfortunately, the reports contained several incidences of incomplete data and/or typos, presumably from insufficient data validation during the data entry process. Some records were outright discarded due to inapplicable data. Due to the sheer volume of typos, they could not be discarded and most of these records were either manually or programmatically corrected (e.g., JavaScript replace statements).

Additionally, a record lacking usable data in only one field was generally kept as-is and simply ignored during the affected visualization for concern that insufficient data would be left for examining trends and patterns otherwise (e.g., if no duration was given, the record was ignored for the duration graph; if that record did have a location, it would be included in the geographic heat map). An example of the code written for cleaning up duration data is available in

Appendix C, which also leverages a utility called *Juration* that includes some simple Natural Language Processing capabilities to convert text to time (e.g., “5 min” = 300 [seconds]).

For the purpose of rapid prototyping, most datasets were ‘cleaned on the fly’, to avoid wasting excessive time tuning data for a graph that might get discarded (and several were). Only once are design stabilized, did we take some of the raw data and aggregate into convenient .tsv files that would greatly improve run-time. This also proved helpful when developing the duration chart, as it was easier to manually insert the missing ‘zero-value’ minute intervals. The full set of code that produced these aggregations is in our GitHub repository, and can be made available upon request.

The data used for displaying NYS map is achieved following the instructions from stackoverflow. The output .json file contains information about transform, type, coordinate data for each county in New York State. For further marking of each county, the “id” property was added by hand to each county, totally 62 counties, with the formal county name. Same operation is conducted for generating US map.

Data Mapping

Choropleth Maps
Two choropleth maps are used in our project, one for United States in state unit and the other for New York State in county unit, showing the UFO sightings variation among US and inside NYS. The US map paths are plotted from geo data in us.json, with Albers projection. Colors are assigned to each state based on the number of UFO reports of each area (read from ufoUS_edit.json). The boundary for New York State is highlighted after the whole map is loaded, for zooming into detailed UFO reports distribution in counties. The whole map is scaled down to 0.4 and placed to top left for page layout. The NYS choropleth map is plotted with data from nys.json with Mercator project. It is scaled by 5000 for clear view. The raw data for UFO reports are formed in city unit, then these are transformed by creating an dictionary object that maps cities to counties based on geoLoc.json, and used for color assignment. Data of primary commercial airports in NYS is collected and displayed on NYS map in form of non-filled circles, with radius difference indicating the enplanements and center located at the exact geological position of each airport. To provide audience a better and easier comparison, the popularity of airports (i.e. radius for circles) are grouped into three scales. We selected the color set from ColorBrewer, with darker colors indicating more sightings. The map shows great covariance between airport enplanements and UFO sightings. Much more sightings are claimed at the place that having popular airport.

Bar Chart of UFO Shapes
This graph is mainly composed of a bar chart, referring the number of report (from the 1st report ever recorded by this website to present) for each shape of UFO. I used linear scale to vertically

arrange the bars (from the lowest number to the highest one), and there is a reasonable padding between each bar. In each bar, it is consisted of two part (shown by two color). The blue part suggests the portion which appear during the night, and the yellow portion represent the UFO which appear during the daytime. There are two transformation used: one is using translate to make the position of those bars to right place relative to the whole svg. The other one is rotating the title of each shape a little bit, which looks better and save space.

Heat Map of Observations Date/Time

The heat map correlates the reported sightings by month on the x-axis, and day of the week on the y-axis. This facilitates an intuitive visualization that highlights the peak reported periods: Saturday nights in the summer. A rectangular representation was used for each cell, in homage to the "lightboard" seen in the movie *Close Encounters of the Third Kind* from Steven Spielberg.

Graph of Observation Duration

In this graph, actual data points were grouped into one minute interval "bins" in order to create a naturally intuitive trend line. A warm color was used for the warmer (Summer / Fall) months, and a cooler color was used for the colder (Winter / Spring) months. This enabled comparison of outdoor conditions with the reported duration of sightings, and no appreciable difference in trend was noticed. Additionally, the individual data points were then rendered via a simple scatter plot with a .20 opacity, allowing for an easy comparison to the "raw data".

Data Visualization / Story

The overall visualization illustrates a positive covariance between airport enplanements and UFO sightings. The sightings are more frequent in the Summer and Fall, presumably when more people are outside. However, the general trend of duration is persistent, regardless of weather. Lastly, there was a surprising amount of human bias in duration reporting, that resulted in what appears to be artificial cluster/spikes in five minute intervals. Unsurprisingly, there was a stronger covariance with enplanements than population, so we opted not to visualize that data. One of the most surprising discoveries, was the creative misspellings observed for duration attributes (e.g., typos in "hours, "minutes", "seconds", etc) and location names, as well as imprecise locations (e.g., "route 90").

References

Close Encounters of the Third Kind. Dir. Steven Spielberg. Columbia Pictures Corporation,1977. Film.

Christie, Dom. "Juration." *GitHub*. Website. Accessed on February 28, 2016.
URL: https://github.com/domchristie/juration

COLORBREWER 2.0. (n.d.). URL: http://colorbrewer2.org/

Displaying NY state with counties map via .shp and TopoJSON. (n.d.).
URL:http://stackoverflow.com/questions/28116230/displaying-ny-state-with-counties-map-via-shp-and-topojson

National UFO Reporting Center. National UFO Reporting Center. Website. Accessed on February 28, 2016.
URL: http://www.nuforc.org/webreports/ndxloc.html.

Wikipedia contributors. "List of airports in New York." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 26 Dec. 2015. Web. 7 Mar. 2016.
URL: https://en.wikipedia.org/wiki/List_of_airports_in_New_York

Appendix A

JavaScript used to extract HTML table to JSON (copy from Console, paste into new file). The technique used was leveraged from a solution described on StackOverflow.com (http://stackoverflow.com/questions/9927126/how-to-convert-the-following-table-to-json-with-javascript).

```
 var myRows = [];
  var headersText = [];
  var $headers = $("th");

  // Loop through grabbing everything
  var $rows = $("tbody tr").each(function(index) {
    $cells = $(this).find("td");
    myRows[index] = {};

    $cells.each(function(cellIndex) {
          // Set the header text
          if(headersText[cellIndex] === undefined) {
          headersText[cellIndex] = $($headers[cellIndex]).text();
          }
          // Update the row object with the header/cell combo
          myRows[index][headersText[cellIndex]] = $(this).text();
    });
  });

  // Let's put this in the object like you want and convert to JSON (Note: jQuery will also do this for you on
the Ajax request)
  var myObj = {
          "myrows": myRows
  };
  console.log(JSON.stringify(myObj));
```

Appendix B

Initial code used to pull county name and latitude/longitude from MapQuest. Due to poor data quality, a manual intervention was performed to amend any incorrect results.

```
var geoData = {geocache: []};
            var citySet = new Set();

            ufoData.nysData.forEach(function (entry) {
                        citySet.add(entry.City.trim());
                        // console.log(entry.City);
            });

            var cityArray = Array.from(citySet).sort();
            var rxCnt = 0;
            var rxLen = cityArray.length;
for (var i=0; i < rxLen; i++) {

var key = "INSERT_YOUR_MAPQUEST_KEY_HERE";
$.ajax({
           url: "http://open.mapquestapi.com/geocoding/v1/address",
           dataType: 'json',
           crossDomain: true,
 //   type: 'POST',
           contentType:'json',
           data: {
           key: decodeURIComponent(key),
           json : JSON.stringify(
           {
           location: { "street": cityArray[i]}, options: { thumbMaps: false}
           })
           },
           // data: {location: { street: "Lancaster, PA"}, options: { thumbMaps: "false"}, maxresults: 1 },
           success: function(data) {
            rxCnt++;
            geoData.geocache.push ({ city: data.results[0].providedLocation.street,
                                                 state: 'NY',
                                                 county: data.results[0].locations[0].adminArea4,
                                                 lat: data.results[0].locations[0].latLng.lat,
                                                 lng: data.results[0].locations[0].latLng.lng });
            // var url = 'data:text/json;charset=utf8,' + encodeURIComponent(geoData);
            // window.open(url, '_blank');
            // window.focus();
            var json = JSON.stringify(geoData);
            var blob = new Blob([json], {type: "application/json"});
```

```
               // var url  = URL.createObjectURL(blob);

               //var a = document.createElement('a');
               //a.download     = "backup.json";
               //a.href         = url;
               //a.textContent = "Download backup.json";
               console.log(rxCnt);
               if (rxCnt >= rxLen) {//rxLen
                              var link=window.URL.createObjectURL(blob);
                              window.location=link;
               }
              },
              error: function(data) {
               rxCnt++;
               console.log( 'error occurred - ' + data );
              }
});

}
```

Appendix C

Code for preprocessing duration data to remove ‘noise’ (unusable data points).

```
ufoData.nysData.forEach(function (entry) {
                        var time = entry.Time;
                        // clean up unusable prose (due to time constraints, didn't bother with more elegant forms
of regex)
                        var duration = entry.Duration.toLowerCase().replace("+", "").replace("~", "").replace(",", " ")
                                                                        .replace("a ", "")
                                                                        .replace("roughly ", "").replace("around ",
"").replace("so far", "")
                                                                        .replace(" each", "").replace(" plus", "").replace("
for ", " ").replace(" maximum", "").replace(" max", "")
                                                                        .replace("approximately", "").replace("approx.",
"").replace("approx", "").replace("aprox.", "")
                                                                        .replace("aprox", "").replace("apx.",
"").replace("app. ", "").replace("aprx", "")
                                                                        .replace("over ", "").replace("under ",
"").replace("less than ", "").replace("less then ", "").replace("less ", "").replace(" less", "")
                                                                        .replace("more than ", "").replace("more then ",
"")
                                                                        .replace("up to", "").replace("and more",
"").replace("or more", "").replace("or so", "").replace("about ", "").replace("almost ", "")
                                                                        .replace("less than", "").replace("less then",
"").replace(" more", "").replace(" (more)", "").replace(" still going", "")
                                                                        .replace("or ", " ").replace(" or", " ")
                                                                        .replace(" and counting", "").replace(" still
happening", "")
                                                                        .replace("1 to ", "")
                                                                        .replace("split ", "1 ")
                                                                        .replace("couple ", "2 ")
                                                                        .replace("a few ", "3 ")
                                                                        .replace("few ", "3 ")
                                                                        .replace("several ", "5 ")
                                                                        .replace("half ", "0.5 ").replace("half-", "0.5 ")
                                                                        .replace("one ", "1 ")
                                                                        .replace("two ", "2 ").replace(" two", " 2")
                                                                        .replace("three ", "3 ")
                                                                        .replace("four ", "4 ")
                                                                        .replace("five ", "5 ")
                                                                        .replace("six ", "6 ")
                                                                        .replace("seven ", "7 ")
                                                                        .replace("eight ", "8 ")
                                                                        .replace("nine ", "9 ")
```

```
                                                    .replace("ten ", "10 ")
                                                    .replace("fifteen ", "15 ")
                                                    .replace("twenty ", "20 ")
                                                    .replace("thirty ", "30 ")
                                                    .replace("an hour", "1 hour")
                                                    .replace(".", "")
                                                    .trim().replace("  ", " ");
                // fix common typos
                duration = duration.replace("mnutes", "minutes").replace("miniutes",
"minutes").replace("mintues", "minutes")
                                                    .replace("ninutes", "minutes").replace("mins",
"minutes").replace("minutue", "minute")
                                                    .replace("minites", "minutes").replace("minuts",
"minutes").replace("minustes", "minutes")
                                                    .replace("minutesutes",
"minutes").replace("miinutes", "minutes").replace("minuites", "minutes")
                                                    .replace("mintes", "minutes").replace("min's",
"minutes").replace("min:s", "minutes")
                                                    .replace("monutes",
"minutes").replace("minutess", "minutes")
                                                    .replace("hous", "hours").replace("hr's", "hours")
                                                    .replace("secods", "seconds").replace("secons",
"seconds")
                                                    .replace("sceonds",
"seconds").replace("secounds", "seconds")
                                                    .replace("seconda", "seconds");

                // replace any text left in parens
                duration = duration.replace(/\(.*?\)/g, "");

                // infer a unit of 1, where only a unit of measure was given
                if (duration == "minute")
                        duration = "1 minute";
                if (duration == "hour")
                        duration = "1 hour";

                // look for "n to n" patterns (e.g., 3 to 5 minutes), average the numbers and replace (e.g., 4
minutes)
                if ( /^((\d+.*.to.*.\d+))/.test(duration) ) {
                        var x = duration.match(/(\d+)/);
                        var y = duration.match(/(\d+)(?=\D+$)/, '2');
                        duration = duration.replace(
                                                            x[0] + " to " + y[0],
                                                            ((parseInt(x[0]) +
parseInt(y[0]))/2).toString() + " ").replace("  ", " ");
                }
```

```
		try {
			var val = juration.parse(duration);
			histoArray.push(val);
		} catch (ex) {
			// ignore anything left, data quality is to low, averaging less than 10% drop-off
		}
	});
```