**Visualization of Air Crash Data**

**Abstract:**

Our project is based on air crashes from 1908 to 2015. There have been at least 5,000 crashes since then. These crashes have happened all across the globe. The crashes have caused varying rates of fatalities and damages.

What we aim to do in this project is to find out and visualize where majority of these crashes have happened. We can from that find, out which routes are the safest with respect to:

1. Casualty rate

2. Crash rate

We can also find out which airline was involved in most crashes and which airline has a higher crash rate (Helps if you’re travelling next time!)

We retrieved the dataset from <https://goo.gl/VGr5z9>. To be able to use it as input for our visualization, we put it in CSV format. The dataset has many variables, but we chose only a few of them that we’ll need for this visualization. We’ve chosen Location, Route, Flight, Causalities and Date.

It is an approach to applying statistical analysis and other analytic techniques to data which has a geographical or spatial aspect. Such analysis would typically employ software capable of rendering maps processing spatial data, and applying analytical methods to terrestrial or geographic datasets, including the use geographic information systems.

**Development:**

**Our Initial Approach:**

We used GeoJSON and TopoJSON to get the map plotted on the webpage. We used SVG for rendering the map. We used our dataset, converted it to CSV and used a Google Doc script to convert the places to Latitudes and Longitudes and then plotted this on the map.

We then wrote a “MouseOver” function for points on the map. A mouse over event would lead to tooltip/infobox kind of box on the screen with details about the crash.

At first we decided to render a flat map using SVG . That looked something like this:



To this map we later started adding more colors and we started writing code to color the map and use a hover or mouse over function. In parallel, we were also looking at Faux-3d Arc rendered globes and we were able to render it in the following way:



This globe rendered in Faux-3D arc code was very poorly rendered and often caused us transition issues along with making the entire visualization cluttered, slow and hard to use. Hence we decided to go only with the Flat SVG Map.

**Working with the SVG Map:**

We stopped working with Faux-3D arc maps for above said reasons and continued development on the SVG Maps. We used tutorials from various websites and were able to render an initial prototype that looked something like this:



This map had a few cities in it that was sourced/referenced from a CSV File with latitudes and longitudes of the cities.

**Issues in this approach:**

There were two main issues in this approach:

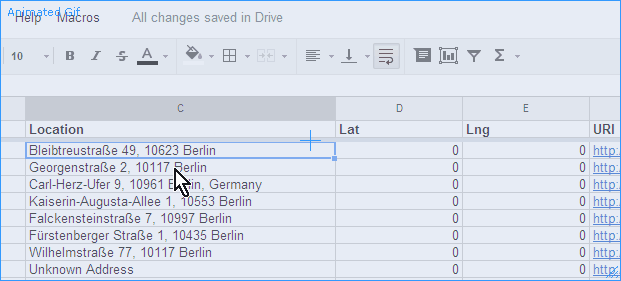
1. The map is mirrored – Meaning East is West and West is East

2. The map calls Latitudes and Longitudes for cities – variables that our dataset does NOT have.

Our Solution to the problem was to invert the SVG Map projection variable by 180 degrees. That seemed to work really well and the map as you can see below:



Our solution to the second problem was to import values in the Cities and Country column in our dataset onto a Google Docs Sheet and use a script to convert the city names to latitudes and longitudes. This is called GeoCoding and we did that using an open source script, referenced in the section below. A small GIF will show you how its actually done:



***This is a GIF image. To view it, visit this link:*** [***https://vilimpoc.org/blog/wp-content/uploads/2013/07/google-sheets-geocode-macro.gif***](https://vilimpoc.org/blog/wp-content/uploads/2013/07/google-sheets-geocode-macro.gif)

We got our cities converted into latitudes and longitudes using this script for Google Docs and next all we had to do was tell JavaScript the source of the information and it would start plotting points all over the globe for us.

**Project End Result:**

****

1908 - 1928



1929 - 1948



1949 - 1968



Infobox with crash details

**References/Bibliography:**

1. <http://uncompiled.github.io/datavis-militarization/military-spending-gdp.html>

2. <http://cartographicperspectives.org/index.php/journal/article/view/cp78-sack-et-al/1359>

3. <https://gist.github.com/dwtkns/4973620>

4. <http://www.d3noob.org/2013/03/a-simple-d3js-map-explained.html>

5. <http://bl.ocks.org/d3noob/401237468c9e38cea8c7>

6 . <http://bl.ocks.org/d3noob/2e224baa7472c5e9e0e8>  
7. <http://bl.ocks.org/d3noob/5189284>

8. <http://bl.ocks.org/d3noob/5189284>

9. <http://bl.ocks.org/d3noob/5189184>

10. <http://bl.ocks.org/d3noob/5193723>

11. <http://bl.ocks.org/phil-pedruco/7745589>

12. <http://www.mapsofworld.com/thematic-maps/world-air-disasters.html>

13. <https://opendata.socrata.com/Government/Airplane-Crashes-and-Fatalities-Since-1908/q2te-8cvq>