INFO 3300 Project 1 Writeup
Team: Terrence Horgan (TMH233), Kevin Hernandez (keh222), Gregory Clerie (gec73)

A description of the data. Report where you got the data. Describe the variables. If you had to reformat the data or filter it in any way, provide enough details that someone could repeat your results. If you combined multiple datasets, specify how you integrated them. Mention any additional data that you used, such as shape files for maps. Editing is important! You are not required to use every part of the dataset. Selectively choosing a subset can improve usability. Describe any criteria you used for data selection. (10 pts)

- Kevin Hernandez: The billboard API was deprecated a while back, so I was thinking of writing a scraper myself, but thankfully I found https://github.com/guoguo12/billboard-charts, a python script for scraping billboard data straight from the website. I put it in a while loop and scraped all available hot-100 charts (http://www.billboard.com/charts/hot-100). I then aggregated them using node.js scripts, mapping and reducing to the formats I needed.

The data, in its most general, original form, as spit out from billboard.py looks as follows:

```
{
  "date": "2016-03-05",          // Date of the billboard chart
  "entries": [                   // An array of "chart entries"
    ...
    {
      "spotifyLink": "...", // Link to the song on Spotify
      "peakPos": 1,         // Highest position ever held by the song
      "artist": "Adele",    // Bae.
      "lastPos": 6,         // Where it was last week
      "title": "Hello",     // "It's me..."
      "rank": 7,            // Current position in the chart
      "weeks": 17,          // How long it has been on charts
      "change": "-1"        // # diff from last week, or "New" if new
    }
    ...
  ]
  "name": "hot-100",             // Name of the chart this entry is in
  "previousDate": "2016-02-27"  // Date previous chart was published
  "latest": true,                // true only for this week's chart
                                    (at the time of scraping)
}
```

It's evident it has a lot of redundancies and unnecessary fields. In the "billboard_scripts" folder are the scripts I used to remove redundancies and mold the data into different forms I used at

some point or another. They have a brief and rough description of the output format on the header. The final format I used for the graph I kept is:

```
[ { year:              // An array of year objects, each with...
    songs: [           // An array of song objects, each with...
      { artist:        // The name of the artist
        title:         // The title of the song
        weeks:         // How long it was on the charts that year
      }
      ...
    ]
  }
  ...
]
```

- Terry Horgan:
    - The data I got from TheTopTens.com and decided to take the top 20 countries with the most poular music, and then display those countries on a color scale using a Json Map. I created an array that had the country's name, un_a3 country code, and rank, and used that to then color those specific countries, a specific color based on their rank, onto the map. Based on what position a certain country was in the Top 20, it was colored a certain color to represent its rank. I chose to use the Top 20, as anymore made the map seem crowded and too much information thrown at you in colors. I did go through and view different map projections, baker, mercator, mollweide, but I settled on kavrayskiy7 as I really liked how the map flowered naturally. To do the actual coding, I first found a map template from Mike Bostock, and modified that to what I needed it to do. After making my array with the data, and creating a world-map.json file that had everything I needed (un_a3 country code json map files were rare it seemed) and then just used for loops to loop through and grab the country code from my array and the country codes from the GeoJson file array, and when they matched, I would color them based on my color array I created. I did this by putting the property “color” as my value of the array rank for that country, and then would color it based on the rank. If the country was not found in my array and the GeoJson file, then I left it as is. I included the original template I used for citation purposes(its the zip called “d3-worldmap-boilerplate.zip”, so that you know the code that I created and the code that I used from Mike Bostock.. I also went ahead and modified Mike Bostock code to not only show the name when you hover over the country, but the rank as well, and if the country was not part of the Top 20, then I modified the code so that it did not include the rank, just the name of the country.

    -
- Gregory Clerie:
    - Since there's no database that contains yearly revenue for artists, I aggregated my revenue data from Forbes' Top Earning Artists list from the year 2015. Originally, I gathered the data for 2005-2015, but since each list contained many of the same prominent artists there were unnecessary overlaps, so I condensed the data into just the past year. Since no database had the exact numbers, I had to gather the data from Forbes and put the information into JSON format manually. The Spotify followers data was contributed by Kevin and he used a manipulation of his original scraper function to collect the followers information for me and it was already in a neatly formatted JSON file.

- **B. A description of the mapping from data to visual elements. Describe the scales you used, such as position, color, or shape. Mention any transformations you performed, such as log scales. (10 pts)**

- Kevin Hernandez:
    - My data is presented as a bar graph mapping mostly from the length of the "song" array in each "year" object and the length of a transformation that aggregated all song objects into an array of artist objects. No special scales were used. My x-axis maps 0 to 1000, my y-axis 1958 to 2015. I didn't include 2016 data, because it's not complete for the year yet.

- Terry Horgan:
    - First off when I got the Mike Bostock template, I really did not like the mercator projection he used nor did I like how he scaled the maps. This was when I then chose to use the kavrayskiy7 projection, and after messing with the scaling for awhile I settled on just taking the width and dividing it by 6. I figure this would work well even for widths that my monitor could not do, and all could view the map in a nice way. For my color scale I used quantize because due to how I created my array of 20 colors.

- Gregory Clerie:
    - To graph the data, I used d3.json and sorted the JSON data into a Javascript array. I then iterated through the data using a for-loop for different indexes to reach different levels of the array. Then I appended circles and text over the circles by using the logarithmic xScale and yScale of each artist's data as the loop iterated through. The xScale was set with the domain being the minimum and maximum of Spotify followers of all the artists and the range being set to accommodate the svg size. Similarly, the yScale was set with the domain being the minimum and maximum of revenue of all the artists and the range being set to accommodate the svg size. The radius was set with the domain being the

minimum and maximum of Spotify followers of all the artists and the range being set to appropriate values and the spectrum scale with the domain being the minimum and maximum of revenue of all the artists and the range being set from dark green to bright green to represent the range of revenue earned comparatively. The radius and spectrum scales were used to represent the popularity of the artist and how much money they made, respectively.

- **C. The story. What does your visualization tell us? What was surprising about it? (5 pts)**

- Kevin Hernandez:
    - I originally mapped every single song's trajectory through time, which resulted in a tremendously convoluted chart of triangles (songs tend to follow of triangular shape, up to their peak position and then back down out of the chart. Few songs bounce up and down). It was also super wide - it needed at least 20,000 pixels of width for the lines to be discernible for all those years (27k songs!)

      I noticed a bit of a pattern scrolling through this chart, and decided to aggregate my data in the way I finally did to make it obvious. Songs used to jump in and out of the chart way more often before, but towards the 90s songs stayed in the chart longer.

      It turns out - and this is evident in the new way that I presented the data - there was a trend in which the "top 1%" of artists (those who make it to the Billboard hot-100) shrank. Even more considerable was the diminution of the number of songs that made it to the chart year to year. A good example of this is in 1967 and 1994, the numbers decreased by 50%.

- Terry Horgan:
    - The idea that I really want the visualization to tell the audience is that music artists come from around the world, in all parts of the world. It would seem, and it is true, that the 2 most popular are US and UK, but it really surprised me to see countries such as South Korea, Jamaica, and even Ukraine in the top 20. I guess I never thought of these places that are big with music, but in reality they are.

- Gregory Clerie:
    - The data tells you that an artist doesn't necessarily have to be tremendously popular online to still generate revenue. Despite the gap a little to the right of the middle of the graph (most likely representative of the age gap of the artist fans with the exceptions of Jay-Z, Lady Gaga, and Justin Timberlake), distinguishing new, upcoming megastar artists like Calvin Harris and One Direction from older

artists like U2 and the Rolling Stones, the revenue generated by both sides is relatively equal. If anything, it shows that older generations still pay for their music more than newer generations who use services like Spotify.