Assignment 3 : Novel Visualisation

Graham Hutchinson

April 2020

1 Introduction

I decided to create a novel visualisation of the the entire history of medals awarded at the Olympics. The visualisation I created was an explanatory visualisation, intended to inform a user of the history of the Olympics, and how many medals were won by each nation each year, how many medals each nation had won cumulatively, and how many medals were awarded each year. I used the Olympics medals dataset available on Kaggle [1] for the data for this visualisation. This dataset containing information on each athlete that competed in the Olympics, for each Olympic Games, what nation they were representing, and what medal they won, if any. I decided to use a line graph to represent the number of total medals awarded at each games, that live updated with each additional year that was visualised, e.g. all the medals awarded at the 1896 games, followed by the 1900 games and so on. Additionally, I used two chorpleth maps that were also animated to update for each games, one representing the number of medals won by each nation at those specific games, and another to represent the total number of medals won by each nation cumulatively.

2 Data

I first downloaded the Kaggle dataset [1] and began considering how best to parse the data for my use. For the live line graph plot, I needed the cumulative medals won at each games for each year. Olympic Games are held every four years, beginning in 1896, with a few notable exceptions e.g. 1940 and 1944 due to World War 2. So, I wrote a python script that, beginning in 1896, combed through all the data, and any tuple whose year was specified to be the same as the current year and whose medal was set to be either "Gold", "Silver" or "Bronze", iterated a counter. After all the data has been combed through, the counter and the year is saved to an array with two columns, years and total medals, and the process is repeated for each 4th year until the dataset ends, in 2016. Following this, I saved the array to a csv file. So, I had a csv file with all the total medal and year information that was ready to animate. Next, I needed to parse the data for choropleth use. I wrote a python program that first declared an empty list for country codes. Next it combed through the data and each time it found a tuple whose medal was either "Gold", "Silver" or "Bronze", it would check if that tuple's country code was in the list. If it wasn't, it would add the country code to the list. After this, the program would have a list with the country code for each nation that had ever won an Olympic medal. Next, the program declared an empty array, with 3 columns: Year, Country Code and Number of Medals and with $(len(country) * len(years))$ rows, i.e. For each year and each country, there was an entry for the number of medals won by that country at that Olympic Games. As before, I saved the array to a csv file, and I had data for the yearly choropleth map. Next I needed to obtain data for the cumulative choropleth map. To do this, I loaded the previous program's csv file, and created a duplicate with the same shape, except with all the Medal column set to zero. Next, I iterated through the the first array, and set the medal count in the second array to be equal to the corresponding medal count in the first array plus that country codes previous total e.g. if Spain had 20 consecutive medals before the 1904 Games, and then won 10 medals at the 1908 Games, then the tuple for 1904:Spain would be set to 20 and the tuple for 1908:Spain would be set to 30 and so on. Once this was completed, I had data that could be used for the cumulative choropleth map visualisation.

3 Visualisation

For the visualisation section, I first had to decide how I wanted to visualise this data. There was a continuous time element to the data, i.e. an Olympic Games every four years, so it seemed appropriate to incorporate this into the visualisation by using animation. I then had to decide which type of visualisations to animate. I decided to use a chroropleth map [2], to visualise both the early Country and Medal data as well as the cumulative Country and Medal data. To use a choropleth map, I needed to install geopandas, and download a shapefile of Earth. I then used geopandas to plot the map of the earth on my python figure. Once this was completed, I merged the shapefile data with the two choropleth map csv files, and then plotted the data for the year 1896 as a test. Once I had this working, I learnt how to animate these plots in python [3]. To animate them, I had to declare a animate function, which had a variable i that increased with each "frame". The function could also be delayed by a number of milliseconds. I set the year to be equal to $1896 + (4 * i)$), and then the program isolated in both of the two choropleth map datasets the tuples whose years equaled the current year. Then both of the datasets were plotted in choropleth form. So now I had a program with two choropleth maps that animated. Next I used the total medals per year dataset aquired in the first program, and declared two empty lists, xs and ys. Each "frame", I appended the i-th value in the first column to xs and the i-th value in the second value to ys. I then plotted xs and ys. This resulted in a live updating line graph, that was in sync with the two choropleth maps, and visualised the number of total medals issued for each Olympic Games. For for the last remaining quadrant of the figure, I added a simple counter that displayed the current year, as calculated from i, i.e. $1896 + (4 * i)$). Finally, for each frame I saved the current visualisation as a png file, then used the imagemagick library to compile these images into a gif.

4 Analysis

The approach of using a choropleth map to visualise data concerning countries is not novel, nor is it novel to use a line graph to plot two variables. However, I could not find any visualisations that had used animated choropleth maps to visualise this kind of data. I also believe that it was novel to sync up these three data sources so that they could give a more complete picture for each Olympic

Year. A negative with this visualisation is the colour bar, as I could only declare a colour bar once, it could not be updated in the animation. So the colour bars are only really relevant for the last few frames. Additionalluy, due to extended high performance from several countries, including USA, Russia, the UK, etc. the maps tended to only clearly show the colours of a few countries, i.e. the ones that routinely performed well.

References

[1] Rgriffin, “120 years of olympic history: athletes and results,” Jun 2018.

[2] B. Cooley, “Let’s make a map! using geopandas, pandas and matplotlib to make a choropleth map,” Jun 2018.

[3] P. Pandey, “Animations with matplotlib,” Mar 2020.