19Fall CS496 Assignment 3

Jingyi Lu & Lingzhi Xi

Visualization for Worldwide Terrorist Attacks during 2016-2017

Storyboard

The aim of our project is to explore the terrorist events around the world; we chose The Global Terrorism Database (GTD) that includes information on worldwide terrorist attacks from 1970 through 2017 (Kaggle link) at this time. Since GTD has a mass of data, we chose to **specifically look at data from 2016 through 2017**; first, we tried to use data from 2012 through 2017 but such a mass of data almost froze our laptops when processing, so we decided to just process data from 2016 through 2017 (data during 2012~2015 is stored in our dataset although we don't use it for visualization).

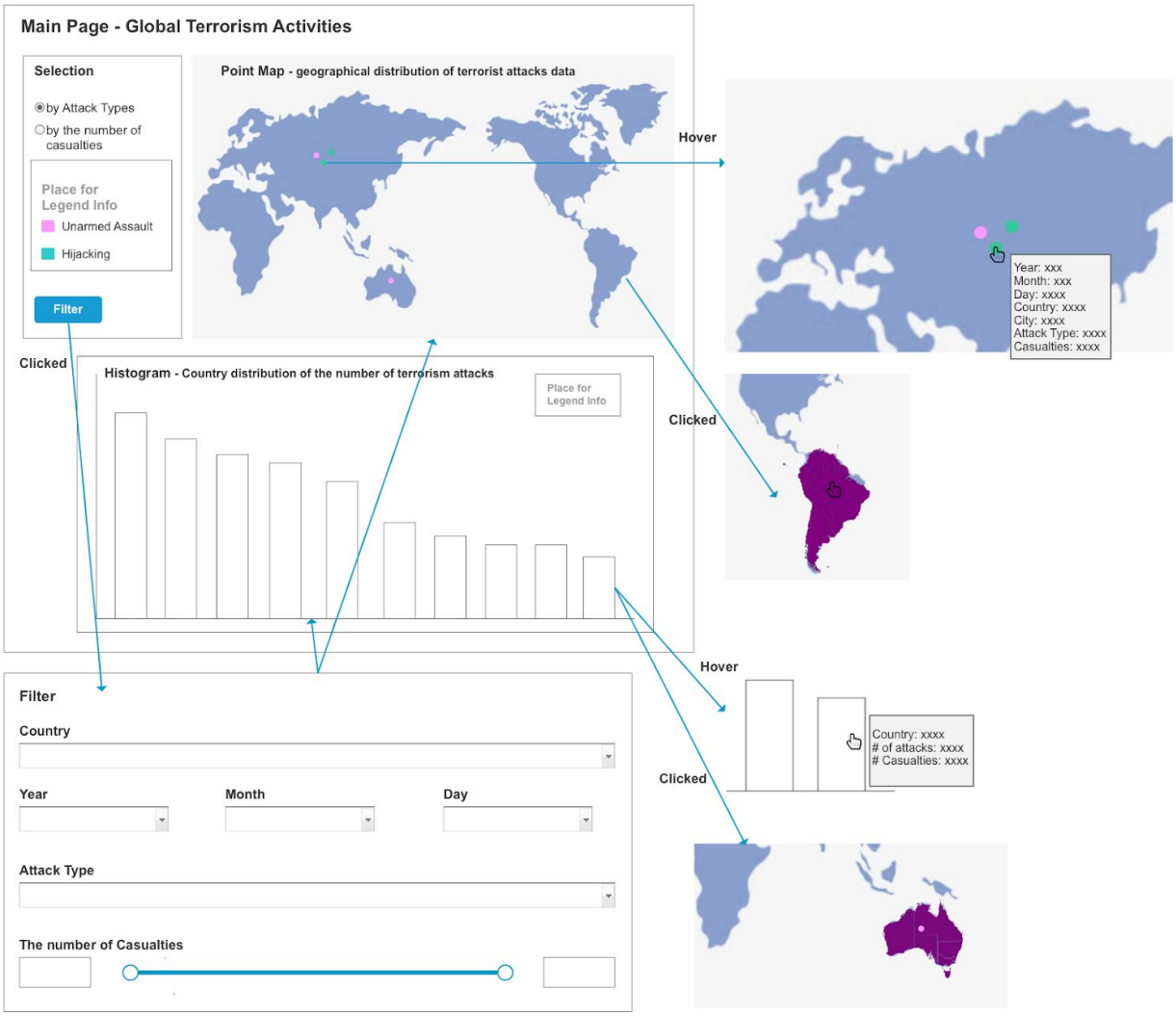
For this project, we planned to build an interactive visualization application that users could explore different aspects (e.g. the number of casualties, attack types) of the information on worldwide terrorist attacks during 2016~2017 within various data columns provided. Below is the list showing the different data columns we used at this time and their descriptions:

Column	eventid	iyear	imonth	iday
Description	Unique ID of each terrorist event	The year which the event occurs	The month that the event occurs	The numeric day that the event occurs

Column	country	country_txt	city
Description	the country code where the event occurs	the name of the country where the event occurs	the name of the city where the event occurs

Column	nkills
Description	The number of total confirmed fatalities for the incident (a.k.a. the number of casualties)

Before discussing the features we planned to add as well as the interaction techniques we implemented for our application, the storyboard shown below is helpful to gain a basic idea about how our application would behave like:



From the storyboard above, we could conclude features we planned to have in our application: a **point map** showing the geographical distribution of terrorist attacks data,

a **selection menu bar** of the point map that enables users to change the data displayed by points in the point map (either representing attack type or casualties), a **bar chart** showing the distribution of the number of terrorist attacks by country (in descending order), a **filter model** making users able to explore data based on personal preferences (effects would apply on both point map and histogram) that could be evoked by clicking the filter button in the selection menu bar.

Now, the featured interaction techniques we would like to implement should be clarified: First, obviously, **dynamic query** is implemented; every time, after editing the selection menu bar or filter model, corresponding graphs (either two or just point map) would continuously update based on the customized filtering. We believed dynamic query is effective here because it enables users to only look at data that is necessary for their personal needs, avoiding useless data and thus, improving the efficiency. For instance, a U.S citizen may only want to look at terrorist attacks in the U.S and its neighboring countries; dynamic query can easily achieve this goal.

Secondly, **selection-tooltip** is indispensably implemented. When the mouse hovers over a point in the point map, a tooltip showing information on the attack represented by this point would automatically come up; also, when the mouse hovers over a bar in the bar chart, a tooltip showing information on the attacks in the country represented by this bar would automatically come up. We considered tooltip to be effective here because there are various kinds of information available for every terrorist attack event. Therefore, a tooltip could quickly provide users with a glance of all these information.

Third, we also implemented **highlighting**. When mouse clicks on a certain region (not on a point) of the point map, the country that is located on this region would be selected/highlighted using a bright color and the country's name would appear; also, the map will automatically zoom in, navigating users to the selected country. We believed

highlighting is effective here because there are many countries available to explore and highlighting can help users specifically look at countries they want.

Fourthly, we implemented **brushing and linking**. When clicking to select a bar in the bar chart, the region in the point map that represents the country represented by that bar would be automatically selected/highlighted using a bright color and the country's name would appear; also, the map will automatically zoom in, navigating users to the selected country. We felt brushing and linking is effective and necessary at this time because it could help users quickly locate the information they want and also connect two interactive graphs together to simplify user operations.

Finally, we also implemented **navigating**. Users can use the mouse to zoom in/zoom out the point map and also drag the mouse to move directions. We thought navigating is effective because the world map is large and there are a large number of countries to explore. Navigating helps users quickly locate the country they want to look at.

In terms of programming, we used Python for data preprocessing. Then, we completed data visualization in React using D3; used AntD as the UI framework and imported a number of libraries, such as styled-components, lodash and so on.

Final Interactive Visualization Application

Our final interactive visualization application has the following features (features different from part 1 will be discussed in the next section): a **point map** showing the geographical distribution of terrorist attacks data on the top right, **a selection menu bar of this point map** on the top left, a **histogram** showing the distribution of the number of terrorist attacks by date (default) on the bottom right, **a selection menu bar of this histogram** on the bottom left, a **filter model** customizing the data based on personal preferences (effects will apply on both graphs) evoked by clicking the filter button in the top left selection menu bar.

Specifically, by default, points (each refers to one attack) on the point map represent the different attack types using different colors; the selection menu bar of the point map enables users to change the data displayed by points in the point map (either representing attack type or casualties). Also, the selection menu bar of the histogram enables users to change the distribution type of the number of terrorist attacks displayed (either by country or by date). If displaying the distribution type of the number of terrorist attacks by country (descending order by default), the histogram therefore becomes a **bar chart**; also at this time, there is a **scrollable bar** locating closely under the bar chart. Due to unreadability for displaying all countries (more than 100), we utilized a scrollable bar that helps users go through all available countries (10 at a time).

Interactive techniques discussed in section 1 are all implemented in our final application; some changes made related to interactive techniques will be discussed in the next section.

Changes between Storyboard and Final Implementation

Our final application does look a little different from our storyboard in the beginning. Some changes are explained below:

First, we **updated the bar chart**; In addition to showing the distribution of the number of terrorist attacks by country, we also made **a histogram showing the distribution of the number of terrorist attacks by date** because we realized this is also a crucial aspect users may tend to look from. **Brushing and linking** is also applied to this new histogram, when users drag mouse to select several bars at the same time which actually results in selecting a time range. First, these selected bars are automatically highlighted with all other non-selected bars marked grey; also, the point map will automatically update, only points representing attacks happening during this selected time range remain appearing on the point map.

Also, **a new selection menu bar of the histogram/bar chart** is added on the left of the histogram/bar chart so that users could change the distribution type of the number of terrorist attacks displayed, either by country as a bar chart or by date as a histogram. Lastly, a **scrollable bar** will appear closely under the bar chart when the bar chart is showing the distribution of the number of terrorist attacks by country. Not all available countries, which exceed the number of 100, could be clearly shown in the bar chart at the same time due to the limited space; with the help of the scrollable bar, users could go through the number of terrorist activities of all countries within clearly seeing information in the bar chart. The bar chart only shows 10 countries at a time and users just adjust the scrollbar bar to see information on all available countries.

Source Code for our Application

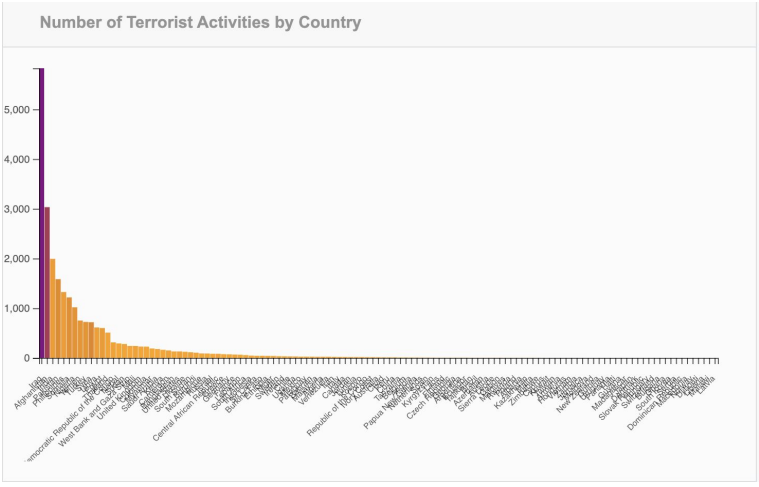
Link to the webpage to explore our visualization online:
https://serinajingyilu.github.io/EECS496Assignment3/
Github Link (set-up instructions included):
https://github.com/SerinaJingyiLu/EECS496Assignment3

Breakdown of the Work

	Design	Programming	Report
Jingyi Lu	finding suitable dataset resources / making decision about which interactive techniques to implement	Implementing point map / filter model / histogram	Cooperatively writing the whole report
Lingzhi Xi	Prototyping the design of our application / handling detailed design like the use of color	Implementing bar chart / selection menu bar	Cooperatively writing the whole report

Comments on Development Process

We spent three days making the initial design and around a whole week developing our visualization application. The most difficult part from our perspective is to integrate D3 and React. The motivations for us to use React are the following: the reusable code blocks (known as components) created by it, the use of the virtual DOM that brings a better render performance for our application, and a debugging system that quickly tells us if we encounter any bugs and which component they are in. However, we met one problem: it is difficult to integrate React and D3 seamlessly since they both want to paint the DOM. React paints the DOM through its virtual DOM and D3 manipulates the DOM directly. In other words, when D3 paints the DOM directly, meanwhile, React would try to use a virtual representation of that DOM. As a result, React will lose track of it and throw strange errors. To solve this problem, we spent a lot of time trying different integration approaches. We first used D3 only for data, and let React do the render part. This approach works with minimal interaction. Then, we tried to let React hand over portions of DOM to D3, which finally created highly interactive and customized graphics. Besides the React part, we also encountered trouble when trying to create a legible bar chart within x-axis including so much countries. For example, we decided to create a bar chart to show the distribution of the number of terrorist attacks by country. Since we have 123 countries in our dataset, when we put all of them in a bounded container, it will look like the following:



Obviously, chart above does not look very good; labels of each bar do not show clearly and bars are very small and too close to each other. After some research, we finally found a way to address this issue: a scrollable bar was added under the bar chart to facilitate users going through the number of terrorist activities of all countries within clearly seeing information in the bar chart. As a result, the bar chart only shows 10 countries at a time and users just adjust the scrollbar bar to see all available countries.