CSU11013 Submission Report - Group 46

Trust-Pilot*

We hereby declare that Trust-Pilot has no legal affiliation with Trustpilot.

Design Outline

The story of Trust-Pilot* is one of hardwork, dedication and collaboration. At the beginning of the Hillary term, emails for our future copilots ;) were made available through BlackBoard. Emails were sent and soon a Whatsapp group chat was formed. We first arranged to meet to get to know each other and exchange ideas. Conceptual heatmaps, barcharts and histograms dominated discussions aimlessly while we found a sense of direction for the project. It wasn't until week 2, when we were expected to start putting pen to paper, hand to keyboard, that we came up with our idea. Amidst a hectic Trinity Central lab, exploding with ideas, Trust-Pilot* was born.

Features Implemented

1. Interactive buttons to compare airlines

For our home page, we decided to create ten interactive buttons which would display a rating for each airline depending on which button was pressed.
The first step in doing so was data analysis. Given the CSV file, we began to figure out how best to analyse the dataset for the purpose of rating the airlines. To calculate a rating for each airline, we decided to use three data points; delays, cancellations and Co2 emissions. We devised a plan to firstly calculate the mean and standard deviation for the three data points, and then calculate a z score for each airline, assuming a Gaussian distribution.
To calculate the metrics above, we used the standard formulae from the log tables. Inheritance was used to combine the methods of the distributionMetrics class with the added method of the z score. Once a z score had been calculated for each airline and data point, we used a tree map to store the data. After doing some research on the most efficient ways to store data, we decided that the key-value pair data structure a tree map provides best suited our needs.
With the appropriate calculations done, we could now calculate a rating for each airline based on Co2 emissions (computed using distance * average emissions per mile), delays and cancellations. We then decided to multiply each factor by a weight, depending on which factors were more important (see fig.1). Since the ratings were all quite close to each other, we placed the values into an array and used the sort() method to order the ratings. Multiplying the index of the sorted array by 0.5 gave us an even spread of ratings for each airline.
The button's graphics were first conceptualised after reading a master's paper on button size and design. The study found that large rounded buttons were most user friendly for older people. With this in mind widgets were created and given vibrant colours and labels. Hit boxes were implemented based on their x and y coordinates to give them functionality. A title button to reset the screen by setting all the trustRating values back to 0, and triangles to change screens were also added, each changing the screen integer being displayed. In front

of the airline widgets lies another “shaded” widget of 0 length. When the mouse hovers over the widget, this “shaded” widget is given a length calculated by using (Emma's weighted ratings) * (the length of the original widgets). This makes it look as though the widget is shaded in to reflect the data being displayed. Users may also change the data being presented using the buttons on the top left of the screen, this changes the trustRatingType of data being displayed and shades the corresponding type of data.

2. Maps to display Co2 emissions, cancellations and delays

For the heat map, we needed to access the frequencies for each data point, depending on the state pressed. To do so the heatMapMetrics class was created, which required a column name and an array of states as constructor parameters. To obtain an array of states, we used the already created widget array and accessed the ‘origin’ attribute on each widget object. The method calculateFrequency() returns a sum of the given data point for a particular state using a for each loop. For every data point, the method insertFrequencies() places the frequencies for every state into the hash map with state as the key.

To implement the calculate distance feature on the Co2 map, we used a variety of mouse methods. In the mouseDragged() function, we set a boolean variable to true such that the state to state distance would be displayed rather than just the Co2 emissions for the state in question. This variable was also used so that a line would be drawn in draw(), using the coordinate of the current mouse position. To calculate the distance, a method in main was created which searched through the table looking for a row that had both states in it.

Dividing up the Work

Finn:

After deciding on our basic direction i got started on Trust-Pilots* homescreen. I knew I wanted to make it user friendly and easily accessible to all, so I decided on a basic widget format that takes inspiration from a commodity we are all familiar with, a phone keypad.

From there I coloured the buttons and implemented a feature when you hover over an airline, represented as a widget, the programme reads the data from the table and shades the widget to a representative area based on the data. And for overall “trustworthiness” a very cool algorithm cooked up by Emma. To reset the widgets I made the title widget act as a home button to set all the widgets back to normal. Arrows were then added on either side of the screen to allow users to switch between the title page and the other members' features. From there 3 buttons were added to the top left of the screen to allow users to change to data that was being displayed by the airline widgets. A very neat animated plane was also added by Paddy to give the background some much needed razzmatazz.
Finally I put in the full names for each airline, revamped the home button and got rid of the lines that were appearing at the left hand side of the widgets. I also added text to the top right in a 70’s style display to let the user know which data is being displayed. There is now also a disclaimer at the bottom to explain what “trustworthiness” is!

Emma:

Firstly, I analysed the data using statistical methods. As outlined above, I carried out different calculations, like z scores to accurately rate the airlines. By this time, Finn had completed the homepage for trust-pilot*, meaning I could combine my data analysis with his visual representation. I also helped Paddy with aligning the data for each state on the heat maps. When a particular state was pressed, the data for that state appeared. (see Fig.2.1)
I also implemented the distance calculator feature on the Co2 map. This allows the user to drag a line between two states to calculate the distance. While hovering over a state, a pop up appears prompting the user to draw a line and when done, the distance appears.(see Fig.2.2)
Finally, once the first heat map had been made, I completed the other two using Paddy’s map class. I created two new instances of both Map and Screen and also had to create two new widget arrays. Paddy and I then worked on ensuring that our program was acting predictably everytime a state was pressed. This was done by checking that every state had a different colour.

YuChen / Paddy:

Before week 1 of our project, I began researching loading data out, from the Processing website itself and Youtube. The objective of the first week was just to display some sort of

message on a screen using the data that was given to us on Blackboard. I based much of my code from the first week on the code from https://processing.org/reference/Table.html, Using a for each loop, I was able to load some significant data out and using the text() method in Processing, combined with a display class I made, I was able to create a temporary screen that goes through each column of the csv file.

The Map:

The next team objective was to create different types of heatmap. Instead of attempting to draw the actual map out, I used a svg file from https://simplemaps.com/resources/svg-us, to get the whole of the USA, along with the individual state. I was able to turn the svg file into a PImage using the loadShape() function in processing. For each state, I used the getChild() function with the states' initials as parameters to assign them to a PImage variable.

To change the colour of the PImage, I used the setFill() function instead of the regular fill() function (it didn't work when I just used fill).

The next step to an interactive map, was to determine when the mouse is over a specific

'state. I encountered a major challenge while doing this, as each states' shape is unique and using the method I used in the method from previous weeks' assignment was going to take too long and is extremely inaccurate. I would like to credit one of my mates, Justin Cunningham for coming up with a solution to this problem. Since each state is a widget,(using the widget class) I presetted the colours of all the states to roughly the same colour, but all the colours would differ by 1 pixel. Then, using the get() function I was able to get the colour of each state. Using a switch statement (now in a for each loop: thanks to Emma Murphy), I am able to detect 'different colours' with the mouse and this leads to the detection of the mouse over specific states. The Map class has since created 3 different heatmaps.

Bhudhav:

From the first lab we had together, I began brainstorming with the team about the ways in which we could visualise the large dataset we had been given.

Problems Encountered

The first issue we encountered early on in the project was merging our work using Github. Merge conflicts and failures to pull new versions were happening every two minutes. However, we decided to overcome this problem by familiarising ourselves with Github and routinely pushing and pulling our versions. This prevented any huge merge conflicts in the future.

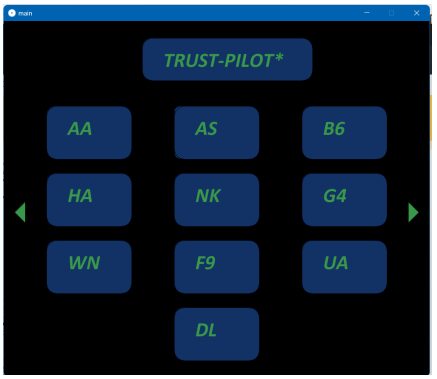
Furthermore, in combining our work, initially it was difficult to comprehend other people's code since we've never had to do such a thing in our course thus far. For example, we found aligning Emma's data analysis with Finn's visual representation of the data difficult since we had to figure out where to merge them.

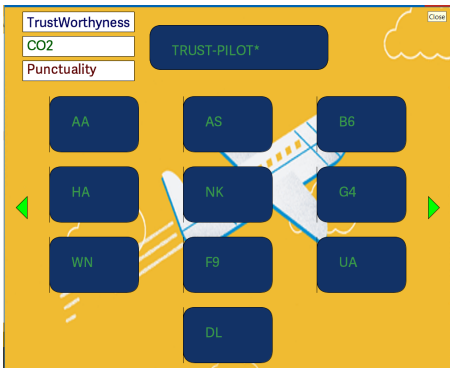
Another issue we encountered was the colours on the maps. Particularly for the orange and red maps, the program was not acting predictably when a certain state was pressed. In the

end we figured out that in using the colour of where the mouse was, we had to be careful not to have two states the same colour. This is what was causing the program to act unpredictably.

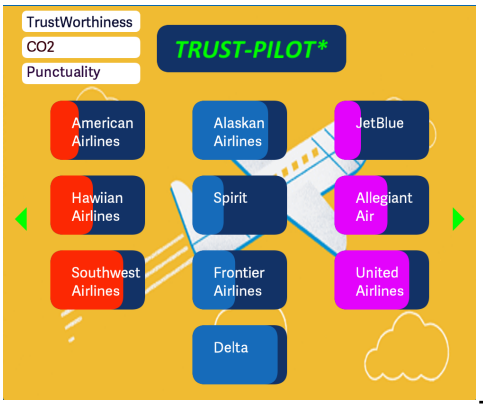
Finally, in our Co2 heat map, implementing the distance feature was an issue since we were only using the 2k CSV file. This meant that we didn't have data for some of the state to state distances. We considered using a larger data file, but decided that an easier fix would be to use a pixel count for each pixel drawn by the line. If no data was found in the CSV file, we used this count multiplied by a constant to estimate the distance.

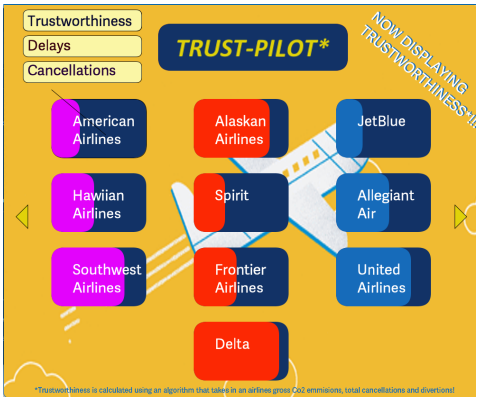
Figuratives Page

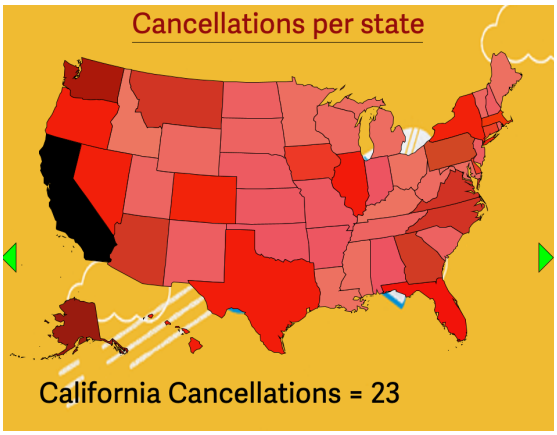




Home pages progression from version 0 to current iteration!







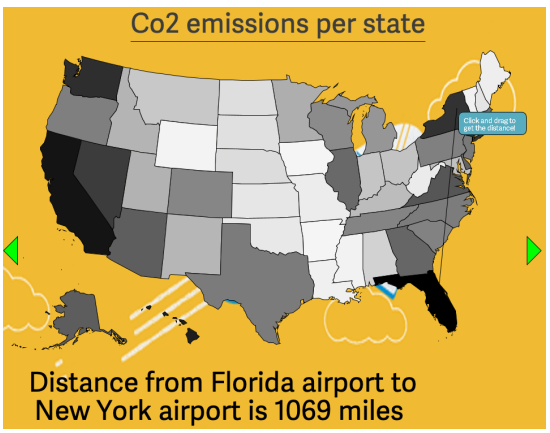


Fig2.1

Fig2.2