Amulya Parmar, Ben Kozlowski, and Elizabeth Ho
Github repository: https://github.com/amulyaparmar/amazing-206-project

SI 206 Final Report: Political Delta

Setting and Achieving Our Goals

Our aim was to combine data from different platforms to create a comprehensive measure of a candidate's political power. This was important to us because we are a group of politically active individuals, and this project is timely due to the on-going nature of campaigns for the 2020 presidential election.

ORIGINAL GOALS
Our goals were to gather the candidates' twitter stats, site traffic info via their campaign websites, and assess their popularity via Google Trends.

ACHIEVED GOALS
We ended up achieving all of our project goals. Because The Google Trends API is not currently available on the RapidAPI marketplace, we decided to use the pytrends API, an unofficial API for Google Trends.

Problems Faced

Dealing with pandas-
Pandas is a commonly used open-source data analysis library. However, it was completely new to Eliza and Ben. Working with a new library is always challenging, and learning to use pandas was no different. We were forced to learn pandas because pytrends (the API that we utilized to access Google Trends) returns a pandas.DataFrame object.

Git control-
Because we're working in a group of three, and each of us has different schedules, it was necessary to utilize version control. We used GitHub. Throughout the project we had a number of merge conflicts, which were difficult to navigate.

Site traffic API limits-
The SiteTraffic API displayed the site traffic to each presidential candidate's site. Our problem was that you have to pay for API queries using this platform, and our group did not want to do this. We solved this challenge by registering for 50 free API queries.

Calculation Files

General_averages.csv
This file contains *calculated* averages across the DEM_Primary database table which contains the average polling statistics against Trump.

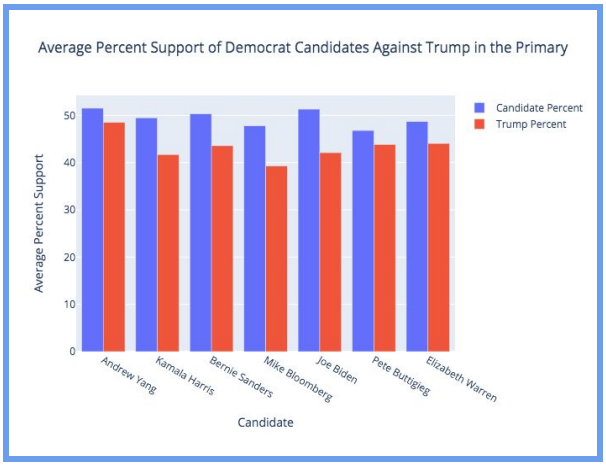
Primary_averages.csv
This file contains *calculated* averages across the DEM_Primary database table which contains the average polling statistics against other democrats.

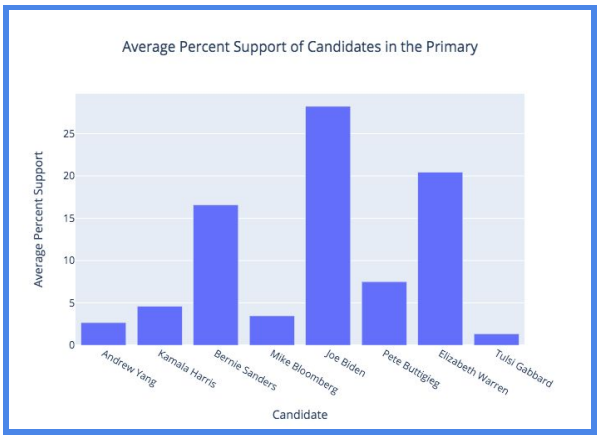
Correlations.csv:
This file contains the Pearson correlation coefficients calculated between different statistical digital metrics and polling statistics.

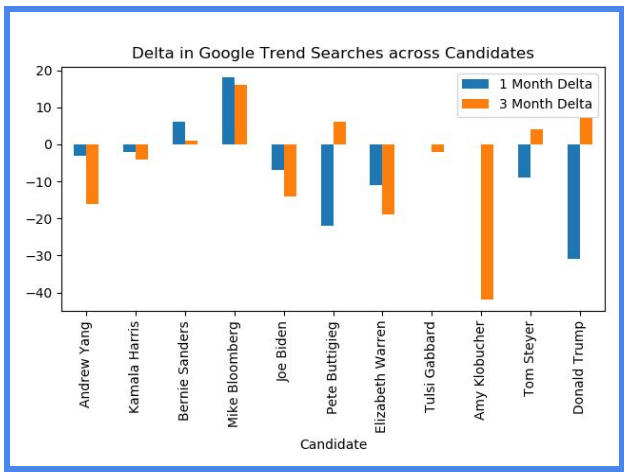
The goal is to easily demonstrate whether there is a statistically significant correlation between these statistics to each other and polling statistics from a correlation matrix.

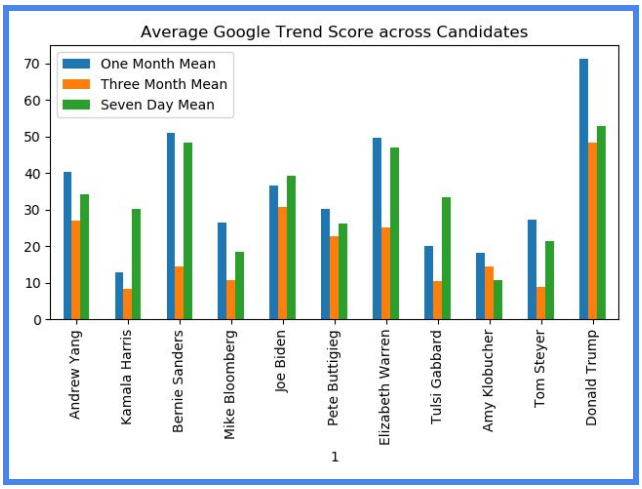
	Pageviews	Google Tre	Three Mon	Average Pc	Twitter Followers_x
Pageviews	1	0.590232	0.511776	-0.10304	-0.01139
Google Tre	0.590232	1	0.555992	0.588417	0.613052
Three Mon	0.511776	0.555992	1	0.58253	-0.09424
Average Pc	-0.10304	0.588417	0.58253	1	0.528028
Twitter Fol	-0.01139	0.613052	-0.09424	0.528028	1

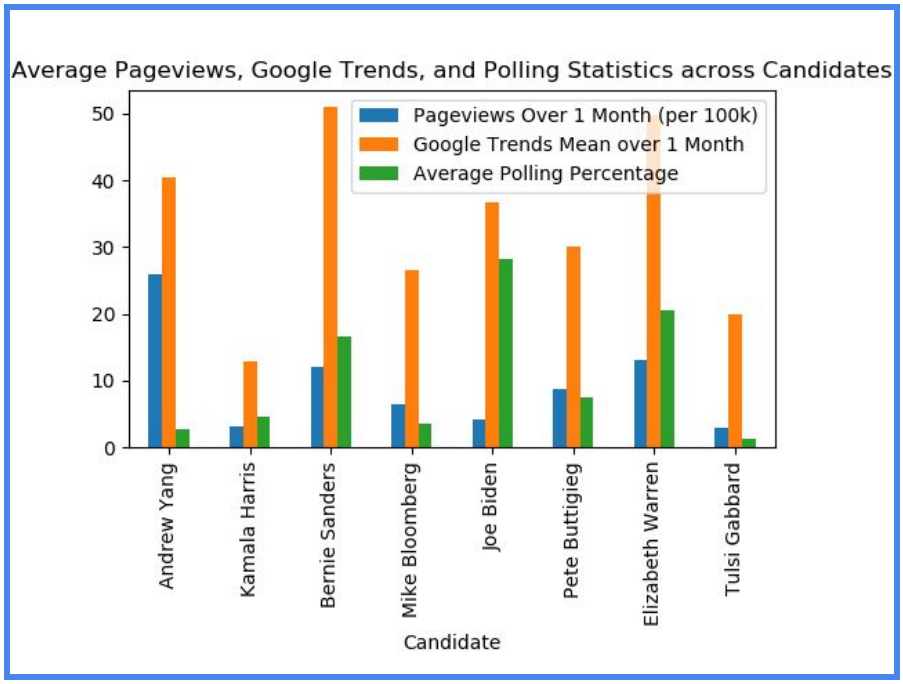
Visualizations

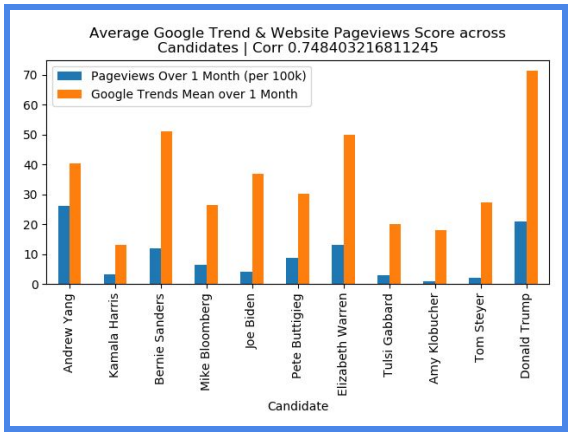


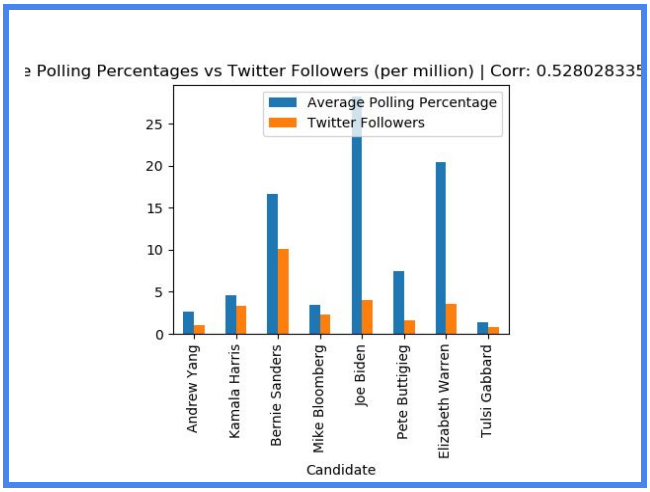







Average Pageviews, Google Trends, and Polling Statistics across Candidates
Pageviews Over 1 Month (per 100k)
Google Trends Mean over 1 Month
Average Polling Percentage
50
40
30
20
10
0
Andrew Yang
Kamala Harris
Bernie Sanders
Mike Bloomberg
Joe Biden
Pete Buttigieg
Elizabeth Warren
Tulsi Gabbard
Candidate


Average Google Trend & Website Pageviews Score across Candidates | Corr 0.748403216811245
Pageviews Over 1 Month (per 100k)
Google Trends Mean over 1 Month
70
60
50
40
30
20
10
0
Andrew Yang
Kamala Harris
Bernie Sanders
Mike Bloomberg
Joe Biden
Pete Buttigieg
Elizabeth Warren
Tulsi Gabbard
Amy Klobucher
Tom Steyer
Donald Trump
Candidate


e Polling Percentages vs Twitter Followers (per million) | Corr: 0.528028335
Average Polling Percentage
Twitter Followers
25
20
15
10
5
0
Andrew Yang
Kamala Harris
Bernie Sanders
Mike Bloomberg
Joe Biden
Pete Buttigieg
Elizabeth Warren
Tulsi Gabbard
Candidate

Code Documentation

We have two files. One file, called get_data.py creates our database with several tables. This should only be run once on a computer because we were instructed to not use any SQL drops in our program. The other file is called create_visualizations.py, and is used to generate our data visualizations based on the populated data. This can be ran many times.

get_real_clear_politics(candidates, cur, conn):
Requires: List of candidates and their websites, and cursor and connection for the SQL database that is being edited.
Modifies: Populates the DemGeneral and DemPrimary tables with polling results for Democrat candidates in a general election against Trump and in a primary election against the other Democrat candidates, respectively.
Effects: Does not return anything.

get_gtrends(candidates):
Requires: List of candidates and their websites.
Modifies: Populates the Gtrend_DELTA and GTrend_MEAN tables with the delta and mean of each candidate's score for search results.
Effects: Does not return anything.

get_site_traffic(candidates):
Requires: List of candidates and their websites.
Modifies: Populates the WebsiteData and WebsiteDelta tables with pageview and visitor statistics for each candidate's website.
Effects: Does not return anything.

get_twitter(candidates):
Requires: List of candidates, and their twitter usernames
Modifies: Gets the number of twitter followers per candidate, divides that number by a million and saves the information to the twitter_table db
Effects: Does not return anything.

visualize_gtrends(candidates):
Requires: List of candidates and their websites.
Modifies: Does not modify anything.
Effects: this creates graphs of google trends information

visualize_real_clear_politics(candidates):
Requires: List of candidates and their websites
Modifies: Writes a table of candidates and the average polling statistics to CSV files.

Effects: Calculates the average polling statistics for each candidate in the general and primary elections.

visualize_twitter():
Requires: List of candidates, and their twitter usernames
Modifies: Does not modify anything.
Effects: Plot the # of twitter followers (divided by 1,000,000) against against the number of polling statistics (%)

visualize_site_traffic():
Requires: List of candidates and their websites
Modifies: Writes a table of candidates and the site statistics to CSV files.
Effects: Calculates the average polling statistics for each candidate in the general and primary elections.

setUpDatabase(db_name):
Requires: Name of database file.
Modifies: Does not modify anything.
Effects: this populates a given database

Resources Used

Date	Issue Description	Location of Resource	Result
12/11/2019	We needed a way to access the Google Trends data as there was no official “API” to access the data	https://pypi.org/project/pytrends/1.1.3/	Use this resource to learn what resources necessary to install the Google Trends “unofficial” api. We installed it successfully what parameters were needed for the
12/3/2019-12/19/2019	We referred to this github page to learn about how to perform pytrends API calls, common issues when trying to access interest	https://github.com/GeneralMills/pytrends	Without this page we would not have been able to realize that accessing dates the interest score over the last number of

	scores from dates that were not showing up		years was not possible in certain formats. We were able to learn work-arounds from this site.
12/12/2019	Problem: SQLite3 Query to list all tables in database only shows one table	https://stackoverflow.com/questions/17997722/sqlite3-query-to-list-all-tables-in-database-only-shows-one-table	We could now make sure the all db tables were properly displayed.
12/15/2019	Problem:my xlabel cut off in my matplotlib plot	https://stackoverflow.com/questions/6774086/why-is-my-xlabel-cut-off-in-my-matplotlib-plot	Solution we could now present most of our graphs without having text being cut off from them.
12/16/2019	Problem: not able return amatplotlib.figure.Figure object from Pandas plot function	https://stackoverflow.com/questions/14936646/how-to-return-a-matplotlib-figure-figure-object-from-pandas-plot-function	We could now return a matplot figure and be able to pass it between functions