Course: CAPP 30122
Team: Claire Boyd (ckboyd)
Benjamín Leiva (bleiva)
Raúl Castellanos (raulcastellanos)
Jack Gibson (jpgibson)

Project: *Covid Online:*
How were people interacting with COVID-19 government pages during the crisis?

Abstract:

Using data from analytics.usa.gov capturing website traffic for US government domains, we examine how trends in government website traffic changed over the course of the COVID-19 pandemic. In combination with daily COVID-19 cases to track the pandemic's course, we explore what government websites people accessed during the pandemic and how the traffic on these sites changed as the pandemic progressed. Additionally, we use auxiliary analytics.usa.gov data on traffic source and browser language to offer insights into the demographics of people using government websites and to showcase how people accessed government-published information (e.g., through search engines, social media, news, etc.) in times of crisis.

To showcase our findings, we built a web application to display core trends in traffic to government sites and offer analysis to contextualize these trends. Our hope is that this web application can be used by government agencies, journalists, or other researchers that might not have explored this web traffic data before. Our aim was to build the tool as flexibly as possible, so that others (or future us) could use our program as a basis to explore web traffic data in other government departments.

Interface:

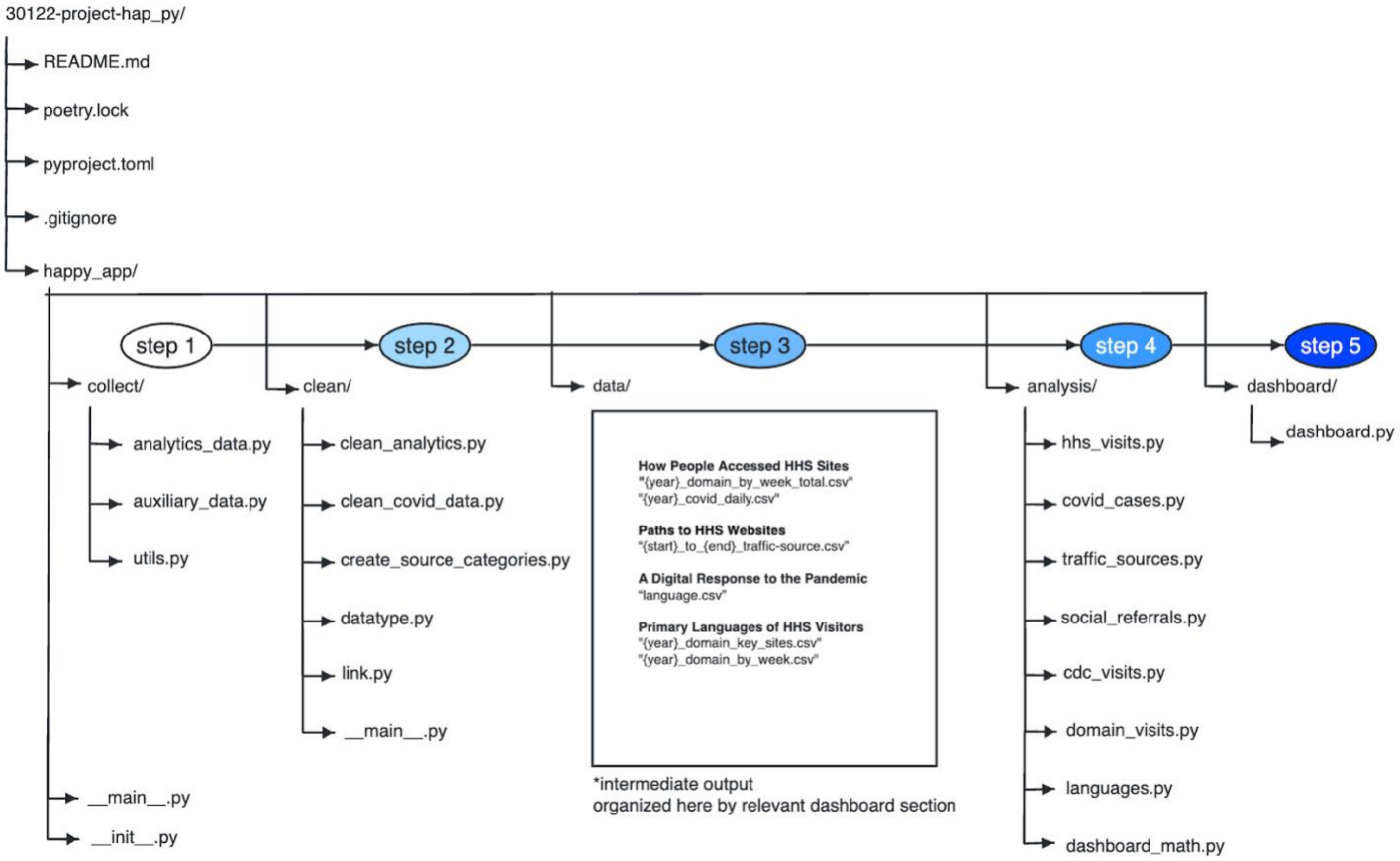
To get started, first download our repository locally. After downloading our repository locally and navigating to that folder, run **'poetry install'** and then **'poetry shell'** to ensure that your virtual environment is set up properly to run our program.

After setting up, there are two ways to interact with our program on the command line:

- **`python -m happy_app`:** Running this command will generate a URL (e.g. http://127.0.0.1:8051). To view our dashboard, copy and paste this URL into your preferred browser.
- **`python -m happy_app.clean`:** Running this command will re-create our data scraping and cleaning process, generating new data into our "data" folder. To view the dashboard with this newly generated data, run the `python -m happy_app` again.

Structure: Below is a diagram that outlines our program structure. We have 5 core folders within happy_app that each have a unique function:

- **Step 1:** collect/ contains crawlers that collect all necessary data
- **Step 2:** clean/ cleans all the data and merges with other datasets, when necessary
- **Step 3:** data/ contains all the intermediate outputs
- **Step 4:** analysis/ reshapes the data (if needed) and creates graph objects needed for visualization
- **Step 5:** dashboard/ formats all graphs and text into the dashboard



Roles: Below is a brief table that describes each team member's roles on the project, including an overview of each script within the repository they worked on.

Name	Module	Tasks
Jack	collect/	• **analytics_data.py:** This file contains a function that makes requests from the Analytics.gov API, aggregates the data, and returns a pandas dataframe for cleaning. The function can make requests based on desired government agency, report type, and time-frame which can inputted on a day-to-day range (e.g. March 12, 2021 – March 30, 2021) or from year-to-year (2019 – 2022).
Jack	clean/	• **clean_analytics.py:** This file contains the classes used to store and clean the Analytics.gov data throughout our data pipeline. AnalyticsData serves as the parent class and contains methods to split multi-year datasets into yearly ones, aggregate and sum data by different time ranges, export data, along with some helper methods needed to execute the other methods. We also included three child classes: TrafficData, LanguageData, and TrafficSourceData; which correspond to the analysis we showcase in our dashboard. Each of these child classes contain extra methods necessary for the specific analysis conducted on the type of data. For instance, TrafficData has function that selects key websites for further exploration, while TrafficSourceData includes a method to label traffic sources into broader analytic categories (e.g., search engine, social media, etc.) *(Claire collaborated on this as well, see specific comments in the code for who wrote each section)* • **datatype.py:** Contains abstract class for our data objects. Used to ensure we designed data storage and cleaning processing in parallel while working across types of data (e.g., Analytics.gov data vs COVID-19 data) • **link.py:** Main home for our data pipeline. Contains two functions that create the various data classes described above, execute the necessary methods to process and clean the data, then exports the files to the data folder to be used in our visualizations. Called in the clean/ __main__.py file to run our data collection process.
Claire	collect/	• **auxiliary_data.py:** This file contains functions that pull in all auxiliary data used for the dashboard, including: (1) covid data from the NYTimes GitHub repository and (2) language codes matched to language name, web scraped from a table online to help make sense of the language data pulled from analytics.gov. *This file also includes additional functions for data we considered using but ended up discarding (commented out).* • **utils.py:** This file contains constants used across clean/ and collect/ so we saved all of the hard coded content in one place in utils.
	clean/	• **clean_analytics:** See description above. • **clean_covid_data.py:** This file uses the abstract class DataType to create a new class CovidData() which has the following methods: fetch_data(), split_by_year(), sum_by(), and export(). This file also contains two helper functions used to clean the COVID data within the fetch_data() method.

		• **create_source_categories.py:** This script contains functions that categorize traffic sources from the analytics data, using hard coded dictionaries in utils.py to map traffic source categories to regex expressions to search for within the url fragments in the analytics data output. This function is called in clean_analytics, for a method in the TrafficSource() Class.
	analysis/	• **social_referrals.py:** This script calculates two functions that together calculate the average frequency of social referrals for a given source (e.g., Facebook, Twitter, Instagram).
Benja	analysis/	• **dashboard_math.py:** Contains helper functions used to create datasets or make calculations for visualizations in the dashboard. • **hhs_visits.py:** Contains a function that creates a Dash object of a line graph containing (1) the number of visits to HHS's website for a requested year during the pandemic (2020, 2021 or 2022); (2) the number of visits in 2019 (for comparison); and (3) the corresponding difference between both series. Also, shows relevant COVID-related government announcements that happened during the requested year. • **covid_cases.py**: Contains a function that creates a Dash object of a calendar heatmap of daily COVID cases for a requested year. • **traffic_sources.py:** Contains a function that creates a Dash object of a horizontal stacked bar chart that describes the source from which visits to HHS's website came from (search engine, direct link, .gov websites, social media, other), for three different time periods during the pandemic. • **cdc_visits.py:** Contains a function that creates a Dash object of a line graph containing the cumulative visits to the CDC website between 2019-2022, highlighting relevant government COVID-related announcements. • **domain_visits.py:** Contains a function that creates a Dash object of a line graph containing cumulative visits to specific COVID-related government websites (vaccines.gov, vacunas.gov, covid.cdc.gov, covid.gov, covidtests.gov) between 2020-2022, while highlighting relevant government COVID-related announcements. • **languages.py:** Contains a function that creates a Dash object of a bubble chart of the most used browser languages in HHS website's visits during the pandemic.
Raúl	dashboard/	• **dashboard.py:** This script features formulas to showcase our research on how people interacted with government web pages during the pandemic. Our main goal was to tell a compelling story using the data, highlighting who these people were, and what pages they visited. To achieve this, I used dash containers and created functions to streamline the process. These functions included creating titles, subtitles, graph containers, interactive graph containers, and number containers, each with unique characteristics. For instance, the interactive graphs have a dropdown component that I later used to callback the application. In the second part of the script, I integrated these functions into an application. To make the text more readable, I used Markdown format to emphasize certain parts of the text and include links. In the layout section, I structured the content in a way that best conveyed the story. Finally, in the callback section, I referred to our interactive component, which allowed users to make comparisons of years in the first section. Overall, this script effectively presents our research in a visually appealing and engaging way.

What the project tried to accomplish and what it actually accomplished. (1 page max).

Initially, we planned on comparing domestic real-time web traffic on the US Citizenship and Immigration Services (USIS) Case Status Portal to quarterly naturalization data, in order to answer the question of who is waiting in the US Citizenship queue. However, we had to discard this topic given a couple of obstacles: (1) the geographic data available was not well defined (i.e., it was aggregated by city name and not connected to state), and (2) the historic data available for the parameters we were most interested in were limited (only a few JSONs per year vs. more than one JSON a day of data for other topics).

Given those constraints, we shifted our scope to focus on how trends in government domain traffic changed over the course of the COVID-19 pandemic. For this approach, we used analytics.gov data to measure web traffic to the Department of Health and Human Services websites and contextualized this traffic with daily cases COVID data from the New York Times. At the beginning, we planned on using national unemployment data to see if the economic effects of the pandemic had a relationship with the visits to government websites but ended up discarding this complementary analysis because we wanted to choose a narrower narrative arch.

In our final product, we ended up analyzing the traffic data across four dimensions.

1. First, we visualized the weekly visits to the HHS website during the pandemic and its difference with a baseline year, while simultaneously plotting COVID data and COVID-related government announcements to see if there was a correlation between these two measures.
2. Second, we analyzed the sources from which traffic arrived at the HHS website, distinguishing between search engines, direct links, other government websites, social media, and others.
3. Third, we plotted the cumulative visits to the CDC website from 2019 onward and used this as a context for analyzing the cumulative visits to other COVID-related government websites like vaccines.gov or covid.gov among others, while also highlighting events that could be related to spikes in the website's visits.
4. Finally, we studied the browser language that visitors to the HHS website had configured to get a better sense of visitor's individual characteristics.

One part of the analysis that surprised us was how little traffic to HHS sites came from social media websites. If we were to do the project again, we might explore search engines a bit more and try to contextualize traffic from search engines with any search engine optimization that HHS sites may have invested in during the course of the pandemic to make their content rise to the top of users search results.