Estimating Damage from Natural Disasters in the USA

Visualization Link: http://noaa-storms-viz.herokuapp.com

Github Link: https://github.com/ss-github-code/noaa_storm_analysis

Motivation

The National Oceanic and Atmospheric Administration (NOAA) in the USA releases the Storm Events Database that records the occurrence of storms and other significant weather phenomena having sufficient intensity to cause damage to property and/or crops. Our primary objective was to explore three major types of storms that cause widespread economic damage in the USA: tropical cyclones and floods (including hurricanes), severe local storms (including tornadoes), and wildfires and droughts. Our project focused on exploring varied independent variables like population, economic activity, and associated weather statistics that affect the total damage caused by these natural disasters in the USA.

While it is not possible to predict where and when the next natural disaster will strike, it is possible to develop models that can predict the amount of damage from such disasters. These predictive models can be useful to plan for better emergency management as the nation's increasing population grapples with the increase in both the frequency as well as intensity of storms caused by global warming. In addition, local governments can use the models to plan for disruptions by using these models for what-if analysis.

This project aimed to answer the following questions:

- What are the various factors that contribute to the amount of damage from natural disasters in counties in the USA?
- How to collect, clean, and visualize the input features related to natural disasters for the period 2000-2021. We collected data including population, economic activity, and weather statistics such as precipitation and temperature in an affected county.
- Is there a correlation between the input features and the damage caused by the storms? Can we apply estimators to estimate the economic damage of a storm?

Data Sources

1. NOAA Storm Events Database

This database includes details on the occurrence of storms that cause loss of life, significant property damage, and disruption to commerce in the counties and forest zones in the USA.

I. **Name**: Storms Events Database
II. **Data location**: https://www.ncdc.noaa.gov/stormevents/ftp.jsp
III. **Format**: CSV files (one per year)
IV. **Important variables**: State Name, County Name, State Federal Information Processing

Standards (FIPS) ID and County FIPS ID, Event Type, CZ_Type, Damage Property, Damage Crops, Begin Date, and End Date.

V. **Time period used**: 2000-2021
VI. **Size**: 1.5 GB (approximately before pre-processing)
VII. **Total records**: 13,274 records, 3.3 MB (post pre-processing)
VIII. **Access method**: Files were downloaded from the FTP site, then records were preprocessed and stored in a relational database (AWS RDS).

2. US Census.gov

To explore the socio-economic factors that impact the damage from natural disasters, we needed to access several secondary datasets. The Census Bureau's Population Estimates Program dataset provided the population estimate in the county affected by the storm for the year of the event (between 2009-2019). The Decennial Census provided the population for 2000 and 2010 and we interpolated the data for the years 2001-2008. The Census Bureau's County Business Patterns dataset provided the number of businesses, their number of employees, and their total payroll in the county. Its Non-Employers dataset provided the number of small businesses and the total revenue generated by them. Finally, its Economic Census provided the economic activity for the top 3 industries in the county.

I. **Name**: Population Estimates Program, County Business Patterns, Economic Census
II. **Data location**: https://api.census.gov/data
III. **Format**: Python string processed into CSV
IV. **Important variables**: Population estimate, number of establishments and their total number of employees and their total payroll, number of non-employer establishments, and their total revenue
V. **Time period used**: 2000-2021
VI. **Total records**: one per event in the storms database
VII. **Access method**: Data was downloaded using the API, preprocessed, and saved into the database (Amazon RDS)

3. US Drought Monitor (USDM)

The US Drought Monitor publishes a map showing the severity and location of drought throughout the country. We use the drought severity data for the county where the wildfire event occurred as input to the model for predicting the damage from wildfires.

I. **Name**: US Drought Monitor
II. **Data location**: https://droughtmonitor.unl.edu/
III. **Format**: Python string processed into CSV
IV. **Important variables**: Drought intensity level
V. **Time period used**: 2000-2021
VI. **Total records**: One map per wildfire event in the storms database
VII. **Access method**: Data was downloaded from the website and saved as CSV files in Github.

4. National Weather Service (NWS)

The primary dataset is missing the county FIPS, latitude, and longitude of storm events that occur in a forest zone (a National Weather Service forest zone covers more than one county). We used the zone county correlation dataset to get the FIPS, latitude, and longitude of a given zone FIPS ID for an event in the storm events dataset.

I. **Name**: Zone County Correlation File
II. **Data location**: https://www.weather.gov/gis/ZoneCounty
III. **Format**: Pipe delimited text
IV. **Important variables**: list of County FIPS in a zone, their latitude, and longitude
V. **Access method**: File downloaded from the website and saved in Github.

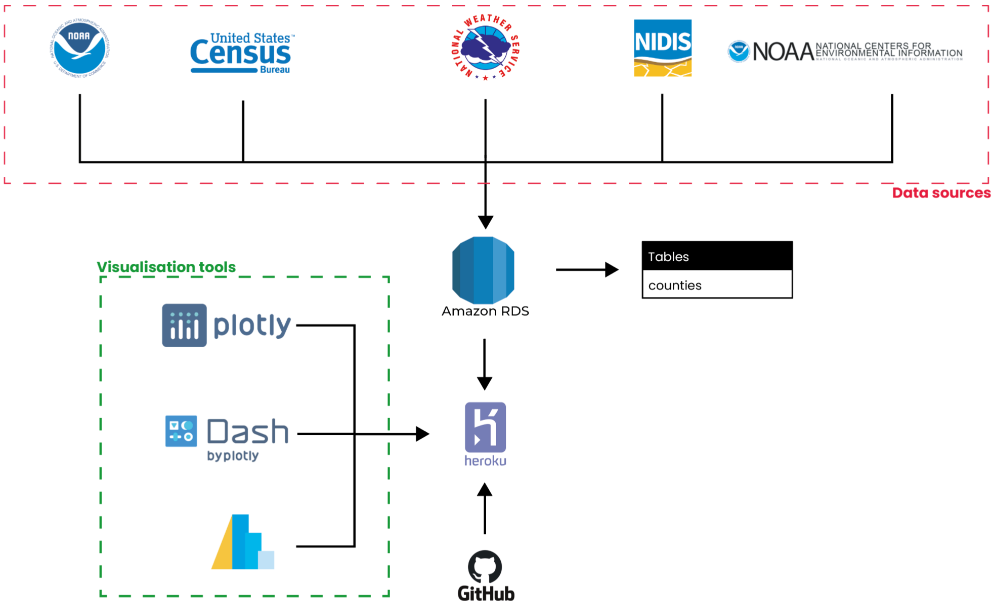
5. National Center for Environmental Information (NCEI)

To get the weather records for a county, we needed the list of weather stations in and around an affected county. From the list, we find the nearest weather station that has daily weather summaries for the past 10 years from the date of the event. We obtain the precipitation, snowfall, and temperature data for the affected county.

I. **Name**: Climate Data Online
II. **Data location**: https://www.ncdc.noaa.gov/cdo-web/
https://www.ncei.noaa.gov/access
III. **Format**: JSON
IV. **Important variables**: Date, precipitation, snow, snow depth, temperature max, and min
V. **Time period used**: 2000-2021
VI. **Total records**: approximately 3653 records for each of the 407 wildfire events
VII. **Access method**: Data was downloaded using the NCDC and NCEI API and saved as CSV files in Github.

Application Architecture

Below is the architecture of our live dashboard application:



Data Manipulation Methods

How specifically did you need to manipulate the data?

- **NOAA Storm Events Database**

The property and crop damage fields in the dataset were strings and used literals like M to represent millions. We transformed these fields into quantitative data. Additionally, we look at events where the total damage (sum of property and crop damage) is more than $1 Million.

The Plotly library requires that the county Federal Information Processing Standards ID (FIPS) be 5 characters long string instead of numbers; we transformed the FIPS field to match the requirements.

Some natural disasters (especially wildfires) take place in forest zones that cover multiple counties. In such cases, we exploded the list of counties that cover a forest zone using the zone county correlation file from the NWS. This allowed us to show the damage on the Plotly map using county FIPS. For this study, we split the total damage from such events equally among the affected counties.

We had to assimilate the information about the 3 types of natural disasters into the Pandas data frame in a specific way to ensure that the data could be visualized using both Plotly and Altair libraries. We put in significant effort to put together the tooltip text for the visualizations. These included gathering event dates and event types for each county - some of the counties had multiple events in a year and some had multiple reports of the same event.

We used the timestamp fields in the database to calculate the duration of events. Finally, to adjust for inflation, we downloaded the consumer price index table for the years 2000-2021 from the US Bureau of Labor Statistics. We adjust the total damage to 2020 $ and use the adjusted values to show the total damage on the dashboard as well as for analysis.

- US Census.gov

The US Census API for gathering population estimates, county business patterns, nonemployer statistics, and the county economic data all return strings that need to be manipulated to extract the relevant information into Pandas data frames. The API for gathering county business patterns changed for years before 2015. Similarly, the API for collecting county economic data from the 5-year economic census changed ever so slightly and these special cases had to be considered diligently.

The economic activity collected from the API was then processed to collect information about the top-3 industries by revenue in the affected county.

- US Drought Monitor

The US Drought Monitor publishes a country-wide drought report every Tuesday. We used Pandas datetime library to get the date for the Tuesday before a wildfire event. To show helpful tooltips on the Plotly map plotting the drought severity information on the map, the name of the county was required and it came from merging the drought report with the zone county file.

- NCEI

We used two APIs to gather weather data for an affected county. The first API returned the list of weather stations in and around a county. We used Haversine distance calculation to find the 3 nearest weather stations based on the latitude and longitude of the affected county. The second API was used to query the nearest weather station to retrieve the previous 10 years of weather data for the county. Pandas datetime library was used extensively to achieve the desired results.

How did you handle missing, incomplete, or incorrect data?

- **NOAA Storm Events Database**

Some records did not have the state and the county FIPS ID fields filled. We removed such records as we did not have the ID fields to automate collecting data for such events. Any event that covered multiple counties did not have a county FIPS associated with it. We used the zone county correlation file from the NWS to insert rows for the counties affected.

- US Census.gov

The US Census API provided population estimates for an affected county. The estimates were available for the years 2009-2019. For the years between 2000-2008, the population was interpolated from the decennial census records for the county from 2000 and 2010. The API for collecting county business patterns and county economic data would sometimes return no data for a county (the counties from the state of Alaska were more often in this category). We had no choice but to report NaN whenever such a missing record was found.

- NCEI

We were interested in the list of weather stations that were functional in the 10 years before a wildfire event. In addition, we wanted the list to only include weather stations that had recorded temperature and precipitation for this period. There were a few instances where the API was unable to find any weather stations that met the criteria. In a few other instances, the API returned weather stations that turned out to be non-functional (all rows of weather data were NaN). To recover from such an error, we get a list of the 3 nearest weather stations. Even with the additional error handling, we had a handful (less than 10) cases where we did not find weather data for an affected county.

How did you perform conversion or processing steps?

We took the help of several Python packages including Numpy, Pandas, and Regex. The processing steps have been coded using several helper functions. Among them, the `convertStrToNum` to convert the damage from string to quantitative floating-point values, `update_fips` to find county FIPS for those events that occurred in a forest zone, and the `calculate_total_damage` to compute total damage in a year for a county and compute event duration are significant in their use of Pandas split, apply, combine API. Extensive use of Python data structures such as dictionary and set have been used to massage the data into a form that allows Plotly and Altair libraries to easily visualize the collected information.

In addition, functions such as `haversine` use latitude and longitude to perform distance calculations to find the nearest weather stations to an affected county.

Finally, we have used Python libraries requests, json, psycopg2 to assist us with tasks such as downloading data, loading it to data frames, and saving data to a Postgres database.

What variables and steps did you use to join the data resources to perform your data analysis?

For this project, the county and state FIPS were the most useful columns. They identified the location of the natural disaster and were used to join the primary dataset to the secondary datasets. The APIs used to gather economic and weather related information exclusively used the state and county FIPS. Similarly, Plotly uses the county FIPS to plot information on the US map.

Briefly describe the workflow of your source code and what the main parts do.

There are two major workflows in the source code. They are: (a) collect and save county information (population, economic statistics, weather) to either Postgres database or CSV files checked into Github and (b) use the collected information to display on the dashboards.

The data collection workflows have been coded in Jupyter notebooks found in the notebooks folder: *Storm_Database.ipynb*, *WildFireWeather.ipynb*.

The *Storm_Database.ipynb* performs the job of collecting population and economic information for every county in the NOAA storm events database for the years 2000 to 2021. The main loop for getting the above data for every county affected by a storm in a year is as follows:

- `get_storm_data`
- `get_county_pop`
- `get_county_business_patterns`
- `get_county_non_emp_stats`
- `get_county_economy`

  Finally, the data is saved to the Postgres database.

The *WildFireWeather.ipynb* performs the job of collecting the drought and weather data for every county that suffered from a wildfire in these years (2000-2021). The main loop for getting the above data for every county affected by a wildfire in a year is as follows:

- `get_wildfire_events`
- `download_climate_info`

  Finally, the data is saved in CSV files and manually uploaded to Github.

The dashboards have been coded using the Plotly Dash framework. Besides the setup required for a Dash application, the main Python modules are *apps/app1.py* and *apps/app2.py*.
The first dashboard displays the storm events from a year (selected by the user) from the NOAA database on a Plotly choropleth map and an Altair scatter plot of storm damage and date. The interactivity of Altair allows the user to click on individual counties and that allows us to show the socio-economic information of the affected county.
The second dashboard similarly allows the user to explore the wildfire events from a selected year. It displays the drought information on a Plotly choropleth map highlighting the affected county and an Altair set of line and bar plots display the weather stats for the affected county.
Note that the code used in the two dashboards has also been tested to run in a Jupyter notebook: *siads_591_report.ipynb*. This notebook can be run on Google Colab.

What challenges did you encounter and how did you solve them?

The biggest challenge was to ensure that the Pandas data frame required for both visualization and analysis contained the information in exactly the shape required by the libraries used for the tasks. The process of extract-transform-load had to be refined and cleaned several times as we found unexpected hurdles while dealing with APIs that would fail in unexpected ways especially when dealing with older data (before 2010). The Pandas split-apply-combine, Python's data structures, and Python's exception handling patterns were extensively used to work around such issues.

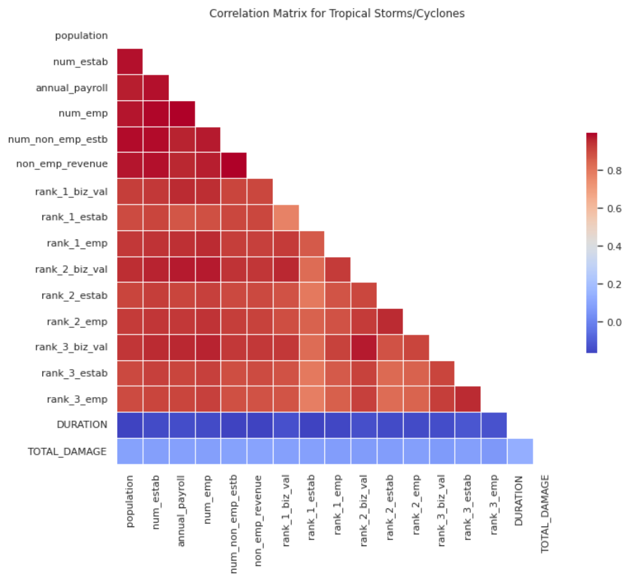
Another challenge was to understand the requirements of creating the dashboards using the Plotly Dash framework. We wanted to showcase our learnings from our coursework on visualization and we chose to explore the challenges involved in setting up a dashboard application online. We reviewed tutorials on Plotly Dash and studied how to incorporate both a Plotly as well as an Altair visualization in a dashboard. We found a very helpful resource on the web that allowed us to use the Altair library in Dash: https://github.com/plotly/dash-alternative-viz.

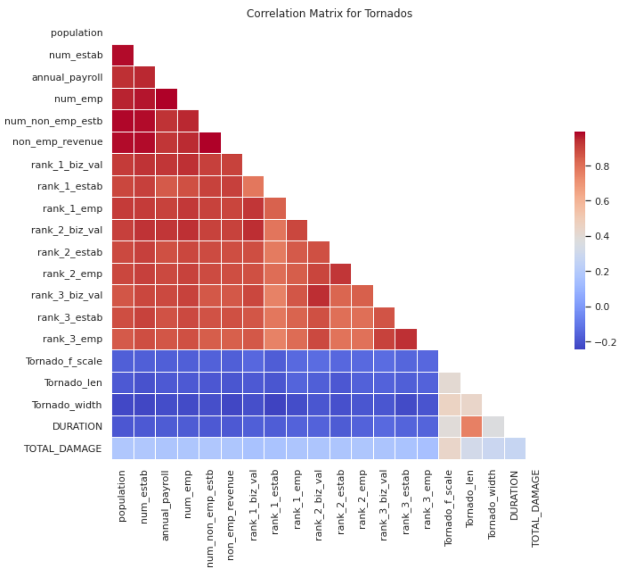
There was learning involved in setting up a live application using Amazon RDS (free tier), Heroku (free tier), and Github. We have also enabled the use of Redis on Heroku as an in-memory data store to cache visualizations plotted using Plotly. This is required as the Plotly library takes a long time to complete a visualization. We planned to have the Redis cache the visualization once created. However, the free tier from Heroku only offers 25 MB of memory for the data store which is insufficient to handle the many visualizations that our dashboard can generate. As a result, our dashboard is slow in loading every Plotly map. We haven't found a workaround to this problem.

Finally, we wanted to ensure that all the steps performed during our study are reproducible. The notebooks to gather all of the information have been tested several times and can be run in their entirety to recreate the database as well as the weather data CSV files. The generation of plots shown in the two dashboards has also been coded in a Jupyter notebook (*siads_591_report.ipynb*) to allow for cross-checking of the plots.

Analysis and Visualization

The **analysis** steps have been coded and documented in the Jupyter notebook: *noaa_storm_analysis.ipynb*. The workflow for the analysis included reading data saved in the database and adjusting the total damage for inflation (using consumer price index values for the event year and month), and considering a log-log model.
The first step in our analysis was to study the **correlation** between the various input features that we had collected. We used Pandas `corr` function to compute the pairwise correlation between the columns for each of the three major categories of natural disasters.



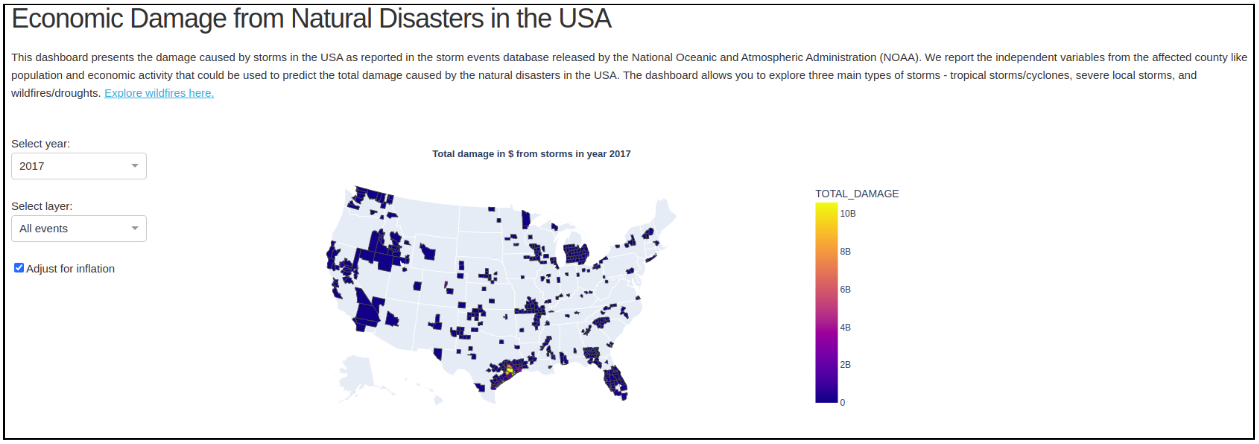


We found that several fields were highly correlated with each other. Not surprisingly, the number of establishments, their payroll, and their employees were correlated with each other and with the population of a county. Surprisingly, there was very little to no correlation between the total damage and duration fields. The correlation matrix for the input features of tornadoes (shown in the right figure above) shows a correlation between the tornado length and duration. Using the insights from the above correlation matrices, we limited the number of variables for the regression model (ordinary least squares regression from statsmodels). We have shown the result from the regression model for the data on tornadoes. We see that the coefficients for duration, tornado_f_scale, tornado_width, and rank_3_estab are significant at the 95% confidence level as shown below.

```
                              OLS Regression Results
==============================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept       4.137e-16      0.021   1.96e-14      1.000      -0.041       0.041
 DURATION          0.1209      0.024      5.100      0.000       0.074       0.167
 Tornado_f_scale   0.3722      0.025     15.077      0.000       0.324       0.421
 Tornado_width     0.1298      0.025      5.234      0.000       0.081       0.178
rank_3_biz_val     0.0506      0.065      0.775      0.438      -0.077       0.179
rank_1_biz_val     0.0623      0.066      0.941      0.347      -0.068       0.192
rank_2_biz_val     0.0457      0.087      0.526      0.599      -0.125       0.216
 rank_3_estab      0.1238      0.046      2.668      0.008       0.033       0.215
==============================================================================
```

We next split the 3 datasets into **train and test splits** and applied **one hot encoding** to the categorical features. We applied **principal component analysis (PCA)** to reduce the dimensionality of the 3 datasets to dimensions that explained 90% of the variance. We apply linear regression models using the sklearn and xgboost libraries and do a grid search for the best parameters for several models. We find that xgboost (gradient boosted decision tree) provides the best model with the lowest mean squared error on the test data and does not suffer from overfitting.

The **visualizations** presented in the two **dashboards** have used **Plotly** and **Altair** libraries. The dashboards use the Plotly **Dash** framework. The workflow for the visualizations is available both as a Jupyter notebook *siads_591_report.ipynb* and the set of Python files (the visualization code is mainly dealt with in *app_df.py*, *apps/app1.py,* and *apps/app2.py*). It consists of reading the data from the database and CSV files, assembling the data in the form that is acceptable in Plotly and Altair (including tooltips), and presenting the visualizations using the Dash framework. The choropleth map shows the economic damage for a user-selected year and allows the user to explore the three main storm categories as shown below.



Economic Damage from Natural Disasters in the USA

This dashboard presents the damage caused by storms in the USA as reported in the storm events database released by the National Oceanic and Atmospheric Administration (NOAA). We report the independent variables from the affected county like population and economic activity that could be used to predict the total damage caused by the natural disasters in the USA. The dashboard allows you to explore three main types of storms - tropical storms/cyclones, severe local storms, and wildfires/droughts. Explore wildfires here.

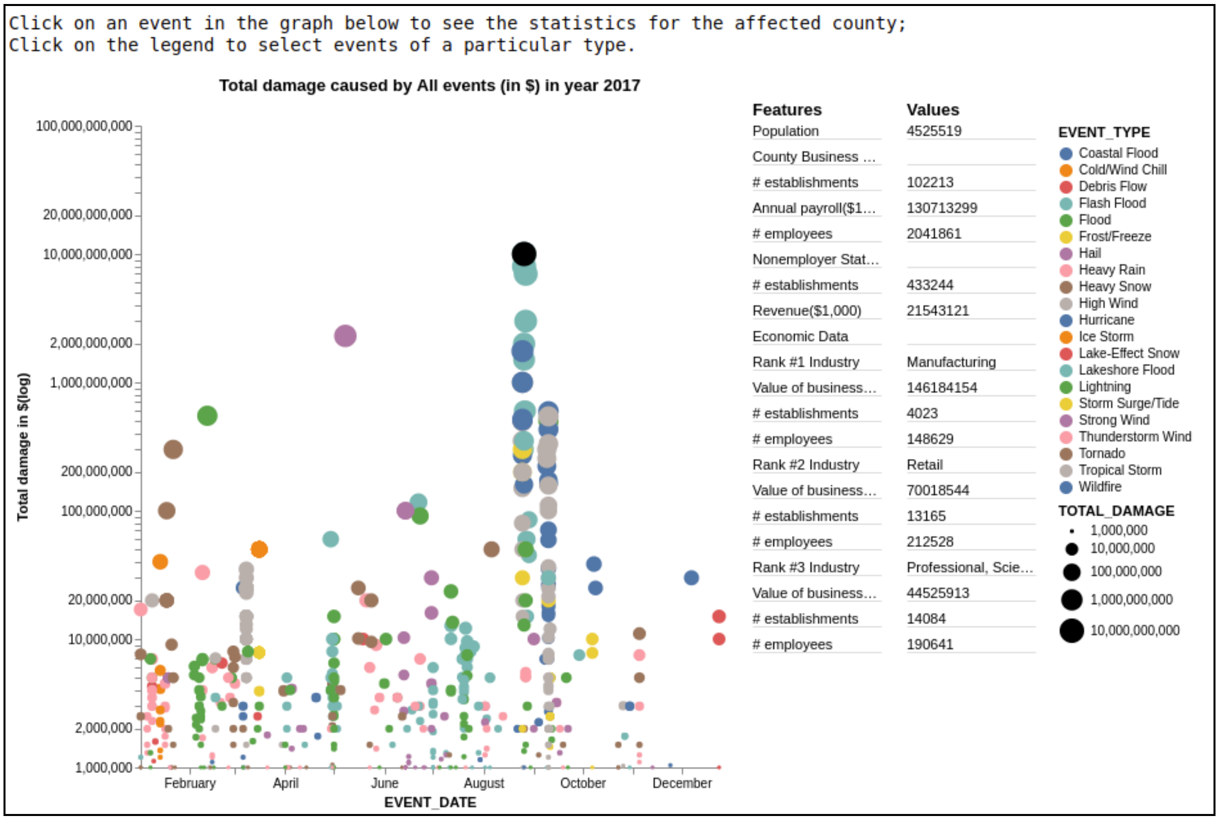
Select year: 2017

Select layer: All events

☑ Adjust for inflation

Total damage in $ from storms in year 2017

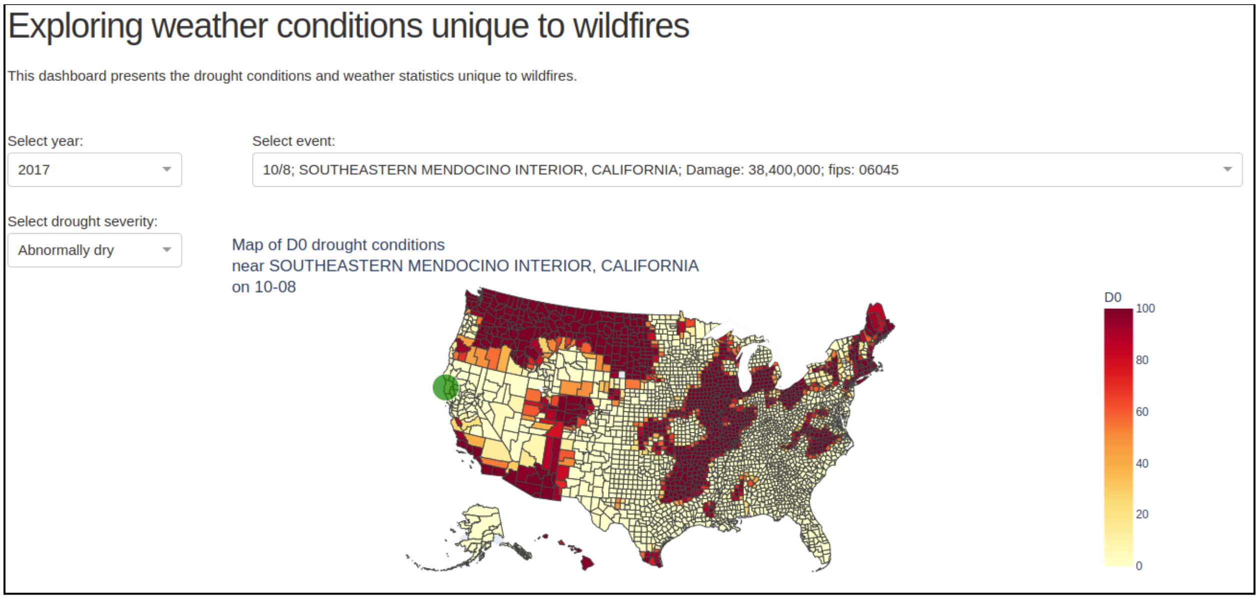
Below the interactive choropleth map, we show an interactive scatter plot to show the same set of events mapped by date on the x-axis. Interactivity allows the user to see the socio-economic data on a county affected by the natural disaster in that year as shown in the figure below.



```
Click on an event in the graph below to see the statistics for the affected county;
Click on the legend to select events of a particular type.
```

Features	Values
Population	4525519
County Business ...	
# establishments	102213
Annual payroll($1...	130713299
# employees	2041861
Nonemployer Stat...	
# establishments	433244
Revenue($1,000)	21543121
Economic Data	
Rank #1 Industry	Manufacturing
Value of business...	146184154
# establishments	4023
# employees	148629
Rank #2 Industry	Retail
Value of business...	70018544
# establishments	13165
# employees	212528
Rank #3 Industry	Professional, Scie...
Value of business...	44525913
# establishments	14084
# employees	190641

In a second dashboard, we have presented a choropleth map to show the drought severity map on the Tuesday before the wildfire event. Below the map, we also show line plots of the weather data gathered from the nearest weather station to the event as shown below.



Exploring weather conditions unique to wildfires

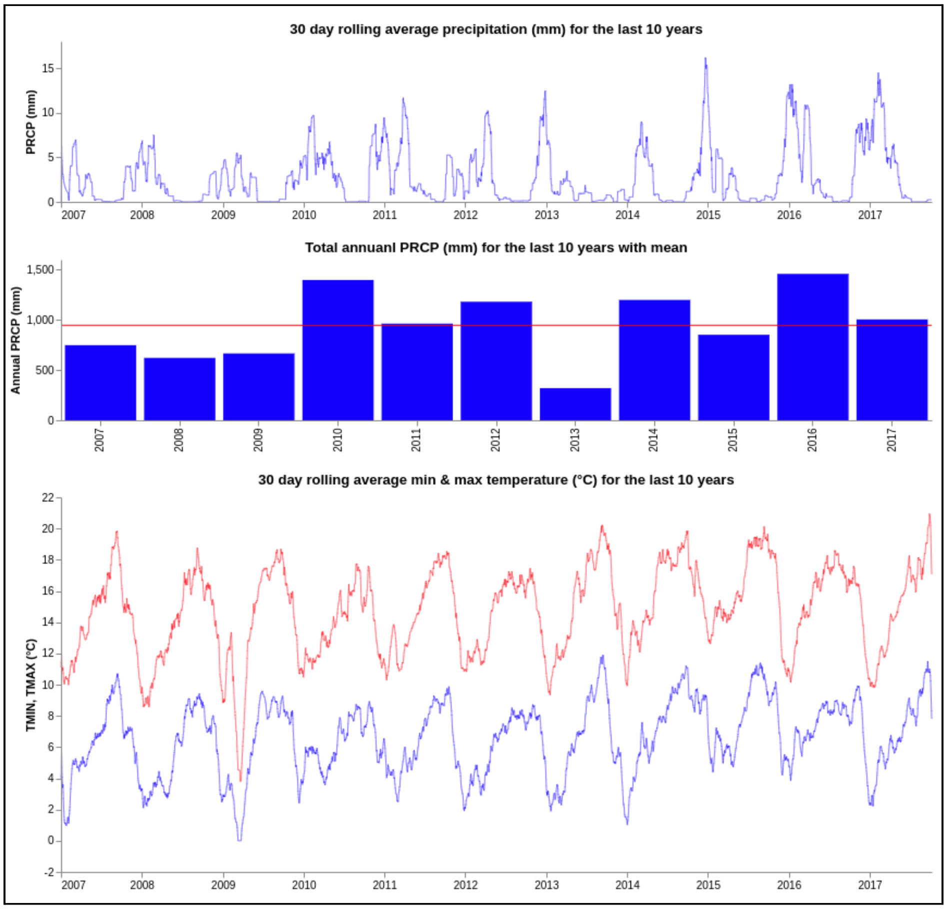
This dashboard presents the drought conditions and weather statistics unique to wildfires.

Select year: 2017

Select event: 10/8; SOUTHEASTERN MENDOCINO INTERIOR, CALIFORNIA; Damage: 38,400,000; fips: 06045

Select drought severity: Abnormally dry

Map of D0 drought conditions
near SOUTHEASTERN MENDOCINO INTERIOR, CALIFORNIA
on 10-08



We were successful in connecting several data sources to mine for relevant data from these sources in a consistent and reproducible manner. We learned several new Python libraries and were able to effectively use lessons learned from our coursework. We also struggled with the quirks of the Dash framework and the differences in Altair visualizations using the framework and in Jupyter notebooks. We were disappointed with how slow the Plotly library loads maps with the county data. We were disappointed that several of the linear regression models were prone to overfitting in spite of using standard scaling and PCA.

Statement of Work

All of us were involved in the initial exploration of the data sources and the consideration of several proposals. Shiv was primarily involved in collecting and saving census data using Census.gov API, coding the Dash visualizations, and in setting up the online dashboards using Heroku and Github. Sashaank and Shiv were responsible for the integration of the Amazon RDS. Sashaank was responsible for exploring data manipulation routines such as inflation adjustment and NCEI API for collecting weather data. Divya contributed to the project proposal and data analysis. The team explored principal component analysis, linear regression, and applied lessons learned from our coursework on data manipulation, supervised learning, visualization, and efficient data processing.