Data Visualization

Mapping Storm event: NOAA Dataset

Index

About Dataset

Background:
Storm Data, an official publication of the National Oceanic and Atmospheric Administration (NOAA), meticulously documents meteorological events. These events are characterized by their intensity, causing loss of life, injuries, significant property damage, and disruptions to commerce.

Scope of Data:
Our dataset spans from 1950, to 2023, providing a pseudo real-time record of meteorological occurrences in the United States. It encompasses a wide array of events, from typical storms to rare phenomena that attract media attention, such as unexpected snow in usually temperate regions. Additionally, the dataset captures significant meteorological events like record temperatures and precipitation linked with other occurrences.

Purpose:
The primary goal of compiling this dataset is to offer a comprehensive understanding of the impact and characteristics of storms and significant weather events. It serves as a valuable resource for analysis and research, facilitating the development of more effective preparedness and response strategies. The dataset's broad temporal coverage and diverse event types make it an invaluable asset for advancing meteorological studies and improving our ability to comprehend and respond to various weather phenomena.

Psuedo-Realtime Data Refresh Mechanism

The NOAA Storm Event Database consists of compressed .csv files, specifically in the gzip format, with individual files corresponding to each year within the time span from 1950 to 2023.

a. ***Previous Refresh Method:***
   Our initial data refresh method used BeautifulSoup, a web scraping library, to extract the latest storm data for a specific year (e.g., 2023) and all event types. This method involved unzipping the folder containing the desired data. However, it was constrained to fetching data from a latest year and did not consider information from other years, limiting the comprehensiveness of our dataset.

b. ***New Refresh Method:***
   To address the limitations of the previous method and enhance the completeness of our dataset, we have introduced an improved refresh method. The new approach is encapsulated in the following functions:

- download_and_combine_csvs(urls): This function efficiently downloads, decompresses, and combines data from multiple URLs. It ensures a comprehensive dataset by encompassing storm data from various years.
- get_urls_with_keyword(url, keyword): This function utilizes web scraping techniques to extract URLs containing the specified keyword from the NOAA Storm Event Database page. In our case, the keyword is set to 'details' to obtain relevant URLs.

  This innovative method ensures a comprehensive and up-to-date dataset by dynamically retrieving and combining data from various years. The enhanced functionality addresses the limitations of the previous method, offering a more robust and inclusive approach to data refreshing. The resulting StormEvents_df DataFrame is then further processed, converting the 'STATE' column to uppercase for consistency and ease of analysis.

Exploring Storm Reports from 1950-2023: Time, Type, Distribution

To analyze following topics matplotlib is used:

1. 15 Most Reported Types of Storm Reports
2. Top Months for Storm Reports
3. Hail Reports by State/Territory
4. Wind Reports by State/Territory
5. Tornado Reports by State/Territory
6. Tornado Frequency by F/EF Category
7. Violent Tornadoes by State/Territory

About Matplotlib:

Matplotlib serves as the primary tool for visualizing data from the NOAA Storm Event Database, chosen for its extensive functionality and seamless integration with Python. As we delve into the analysis of a diverse range of meteorological data, Matplotlib's flexibility allows us to create a variety of visualizations, including bar charts, line plots, and histograms, effectively communicating patterns and trends within the dataset. Additionally, Matplotlib's integration with Notebooks aligns seamlessly with our interactive and exploratory data analysis approach, facilitating the dynamic generation and refinement of visualizations within the collaborative notebook environment.

Challenges

- *Mapping Tornado Ratings Complexity:* The mapping process of tornado ratings into distinct categories demanded careful handling due to the diverse rating formats. The potential for errors or oversights in the mapping function introduced a layer of complexity, necessitating thorough scrutiny to ensure the precision of tornado-related visualizations.
- *Visual Representation Strategy:* The choice of appropriate chart types for visualizing different aspects of storm reports, such as opting for pie charts for event types or bar charts for monthly and state-wise distributions, demanded careful consideration. Ensuring the selected charts effectively conveyed the intended information without compromising clarity was crucial.
- *Handling Unknown or Undefined Data:* Dealing with the presence of unknown or undefined values, particularly in tornado ratings and other categorical data, presented challenges. Addressing these uncertainties became pivotal to accurately represent and interpret the data, safeguarding the overall integrity of the visualizations.

Exploring Storm Reports through States, Months, and Total Injuries

To analyze following topics vegalite is used:

1. Number of Storm Reports in Each State
2. Monthly Trends in Storm count in 2023
3. Injury and Death Trends Over Time

About Vega-lite

The visualizations presented leverages the capabilities of Vega-Lite, a declarative visualization grammar designed for crafting expressive and interactive data visualizations. Utilizing a JSON-based declarative syntax, Vega-Lite simplifies the process of defining intricate visualizations, enabling users to create complex representations without delving into the intricacies of low-level details. This choice is driven by the accessibility it offers, catering to users with varying levels of expertise in data visualization.

Visualizations:

1. Total number of storm events by states
2. Holistic Examination of Storm Occurrences per State and Month
3. Temporal Patterns in Injuries Arising from Storm Onset

Challenges

- *Resolving Connectivity Challenge*: Overcoming the obstacle of connecting the bar chart with the US state map involved identifying the necessity of utilizing selected_yearmonth.BEGIN_YEARMONTH to obtain the selected month. This crucial detail was not explicitly elucidated in the official documentation.
- *Accurate Cumulative Sum for Line Chart*: In constructing the line chart, the requirement for cumulative sums of injuries emerged. To address this, we implemented a window transformation, ensuring precise depiction of the total injuries over time, incorporating both direct and indirect injuries.
- *Seamless Vega-Lite Integration via IPython*: The integration of Vega-Lite code into the notebook was smoothly achieved by leveraging the IPython package. This approach facilitated the comprehensive display of the entire HTML file, contributing to a cohesive and effective visualization experience.

Insights into Weather Events: Analyzing Temporal Patterns and Impact analysis

To analyze the following questions altair and matplotlib is used.

1. Temporal Patterns in Event Reporting by Top Sources
2. Understanding the Impact: Total Property Damage by Magnitude Range
3. Analysis of Fatalities and Injuries Across Various Weather Events
4. Differences in Indirect and Direct Casualties Across Various Weather Events

About Altair:

A declarative statistical visualization package for Python called Altair makes it easier to create engaging and eye-catching representations. It is based on Vega and Vega-Lite and offers a high-level interface with expressive and succinct syntax for building sophisticated visualizations. When it comes to making interactive graphs and charts for data visualization and exploration, Altair is a great choice.

Stumbling block: In the initial stages of our analysis using Altair, a significant challenge arose in the form of 'max row errors.' This limitation restricted our ability to fully utilize the entire dataset, compelling us to work with only a subset of the data. In response to this constraint, we implemented a strategic solution by incorporating the following line of code: `alt.data_transformers.disable_max_rows()`.

Visualizations

1. Temporal Patterns in Event Reporting by Top Sources
   The presented chart offers a dynamic exploration of temporal patterns in event reporting, specifically focusing on the monthly distribution of total events reported by top sources. With the ability to interactively select a specific year, this visualization enables a nuanced understanding of how reporting patterns evolve over time.

The decision to employ a area chart is particularly justified by:

- *Temporal Patterns:* Area charts excel in presenting the evolving patterns and trends over time, particularly when showcasing the distribution of events throughout months. The x-axis in this scenario represents the months, providing a lucid representation of the fluctuations in event counts throughout the year.
- *Aggregated Overview:* The filled areas within the lines of an area chart serve to visually summarize cumulative values, enhancing the ability to discern the collective impact of events reported by top sources during each month. This feature proves particularly advantageous when scrutinizing the overall influence of different sources on the total event count.

Challenges

- *Data Filtering:* Initially, addressing challenges arose in filtering the data for the selected year (selected_year). Ensuring that only relevant entries were considered required meticulous handling of the 'YEAR' and 'SOURCE' columns to exclude null or NA values and refine the dataset for accurate analysis.
- *Aggregation for Monthly Totals:* Aggregating the data to calculate monthly totals for the top sources introduced complexities. Ensuring accurate grouping by both 'SOURCE' and 'MONTH_NAME' while calculating event counts posed an initial challenge in structuring the dataset for meaningful insights.
- *Data Reliability Enhancement:* Initially, we focused on the year 2023 for analysis. However, to ensure accuracy, we expanded our dataset from 1950. Data integrity issues before 2000, including missing values and manual inputs, led us to refine the dataset, excluding years prior to 2000. This decision ensures our report relies on credible data, bolstering the analysis's credibility.

2. Understanding the Impact: Total Property Damage by Magnitude Range

   This visual analysis explores the connection between storm magnitudes and the ensuing property damage. To overcome data challenges, a tailored conversion process was implemented, resulting in the 'DAMAGE_PROPERTY_inM' column. By binning the

magnitude data, the chart provides a concise overview of how varying storm intensities contribute to the total property damage.

The decision to employ a bar chart is particularly justified by:

- *Magnitude as Categorical Variable:* Binning the continuous magnitude data transforms it into discrete categories, resembling a categorical variable. A bar chart is well-suited for presenting the aggregated total property damage within each of these magnitude ranges.
- *Comparative Analysis:* Bar charts are effective for comparisons between magnitude bins and their respective total property damage values. The distinct bars provide a clear visual framework, facilitating the identification of patterns, trends, or disparities among the different magnitude ranges.

Challenges:

- *Transformation Challenges with Data:* Initially, dealing with the 'DAMAGE_PROPERTY' column posed difficulties due to its diverse formats, which included values with 'K,' 'M,' or 'B' suffixes indicating thousands, millions, or billions, respectively. A detailed conversion process was imperative to ensure accurate numerical representation. The solution implemented involved the generation of a new column, 'DAMAGE_PROPERTY_inM,' through the application of a customized conversion function tailored to handle the varying magnitude scales.
- *Complexities in Binning and Aggregation*: The process of creating meaningful bins for the 'MAGNITUDE' column and subsequently calculating the cumulative damage within each bin introduced intricacies. Determining suitable bin boundaries required careful consideration, and aggregating the total damage for each magnitude range demanded precise data processing for the derivation of meaningful and accurate outcomes.

3. Analysis of Fatalities and Injuries Across Various Weather Events

Challenges

- *Bar Label Overlapping:* One of the initial challenges was dealing with bar label overlapping, especially when there was limited space between bars. The default placement of labels resulted in poor readability.
- *Customizing X-axis Tick Labels:* Incorporating custom x-axis tick labels for the different event types posed a challenge. Aligning them appropriately and ensuring they were easily interpretable required careful consideration.
- *Optimizing Layout Constraints:* Adjusting the layout constraints for the subplots to achieve the desired visualization format required experimentation. Ensuring an aesthetically pleasing and informative layout proved to be a crucial aspect.

Execution Plan

Before actualizing the visualization portraying injuries and deaths across various weather events, we meticulously devised the subsequent steps:

- *Selection of Weather Event Types:* Identification of specific weather event types, including 'Tornado,' 'Winter Weather,' 'Excessive Heat,' 'Thunderstorm Wind,' and 'Winter Storm,' was conducted to ensure a comprehensive analysis.
- *Structured Data Compilation:* Relevant data encompassing the number of deaths, injuries, and the overall impact for each weather event type was gathered. This structured dataset was instrumental in presenting information in a clear and organized manner.
- *Presentation of Visualization:* The matplotlib library was employed to exhibit the final visualization. The resulting plot effectively conveyed the total number of deaths, injuries, and overall impact for each chosen weather event type.

4. Differences in Indirect and Direct Casualties Across Various Weather Events

   This visualization is a multiple line graph where it shows the difference between direct and indirect casualties through interactive drop down bqplot line plot. This graph aims to deliver the characteristics of each of the deathly events chosen for this specific graph through its numbers in indirect and indirect deaths and injuries.

Challenges

- *Widget Employment:* The line label is initially empty, requiring the selection of a different variable using the dropdown to populate the label in the line label widget. The widget lacks visual appeal, characterized by small font size and a very light color.
- *Visual:* After integrating two widgets, the line graph became horizontally compressed, making it more challenging to recognize the information presented. Additionally, the x-axis label, characterized by a small size and a light color, further downgraded the visibility of the graph.

Visualization Explanation

- *Selection of Weather Event Types*: The most dangerous events were chosen by selecting those with the highest number of casualties from the dataset. The top six events, namely 'Tornado,' 'Winter Weather,' 'Excessive Heat,' 'Thunderstorm Wind,' 'Winter Storm,' and 'Heat,' were employed to represent the visualization.
- *Line Labels*: Solid and dashed lines were used in each graph to distinguish between them. Solid lines represent direct casualties, while dotted lines are used to depict indirect casualties.
- *Dropdown Widget*: A dropdown widget has been included in the top left corner, allowing users to switch between the Injuries and Deaths line graph.

- *Line Labels Widget*: Positioned in the top right corner, a line label widget provides information about the labels of each line. The content of this widget changes when different variables are selected using the dropdown widget.

References:

- https://www.ncei.noaa.gov/pub/data/swdi/stormevents/csvfiles/
- https://vega.github.io/vega-lite/
- https://matplotlib.org/stable/users/index.html
- https://altair-viz.github.io