**MANYA MALLIKARJUN**

**General Dataset Explanation:**

**1. Introduction:** The report aims to provide insights into the analysis and visualization of a dataset obtained from the HubMap project. The dataset comprises various releases containing information about different anatomical structures.

**2. Data Acquisition**: The dataset was obtained from the HubMap project repository on GitHub. It includes releases categorized into different versions, each containing CSV files about various anatomical structures such as brain, heart, kidney, etc. The data was accessed programmatically using JavaScript and processed for further analysis.

**3. Data Cleaning:**  Upon acquisition, the dataset underwent cleaning procedures to ensure data quality. This involved the removal of unwanted rows and checking for empty values within the dataset. The cleaning process aimed to prepare the data for accurate analysis and visualization.

**4. Analysis**: The analysis phase involved extracting relevant information from the dataset. This was achieved by parsing through the CSV files and aggregating counts based on specific attributes. Counts were recorded for different attributes such as CT/1, AS/1, BGene/1, and BProtein/1 across different releases of the dataset.

**5. Visualization**: Visualization is a crucial aspect of data analysis as it facilitates the interpretation of complex datasets. The analysis results were visualized using a line chart, with each release represented on the x-axis and the corresponding attribute counts displayed on the y-axis. Different attributes were differentiated using distinct colors for easy identification.

**6. Results**: The visualization revealed trends in attribute counts across different releases of the dataset. It provided insights into the distribution and variation of attributes within each anatomical structure. The analysis highlighted variations in attribute counts across releases, indicating potential areas of interest for further investigation.

**7. Conclusion** In conclusion, the analysis and visualization of the dataset from the HubMap project provided valuable insights into the distribution of attributes across different anatomical structures. The cleaned and analyzed dataset can serve as a foundation for further research and exploration in the field of anatomical mapping and understanding.

**Code Explanation:**

In this JavaScript code, we're tackling data processing and visualization tasks. We started by importing necessary modules like **d3** for data manipulation and **FileAttachment** for handling file attachments. Then, will initialize an empty **Map** named **finalCountsMap** to hold the final counts of various categories.

Next, we set up file attachments for multiple CSV files representing different releases of data. These files contain information about different body parts and their associated attributes.

Now comes the data processing part. We asynchronously process the data from these files, iterating over each release and dataset within that release. For each dataset, we read its CSV data and count occurrences of different categories like **CT/1**, **AS/1**, **BGene/1**, and **BProtein/1**. These counts are then aggregated based on the release they belong to and stored in the **finalCountsMap**.

Once all data has been processed and counts have been aggregated, we log the aggregated counts to the console for inspection.

Finally, we move on to visualization. We configure and draw a chart to visually represent the aggregated counts. This involves setting up the chart's layout, axes, and marks such as lines and dots representing the data points. The resulting chart is then rendered in the HTML document using the **drawChart** function. This entire process allows us to efficiently process multiple datasets, aggregate the results, and present them visually for analysis.

A graph on a black background

Description automatically generated

To describe the graph for inclusion in a report, you could use the following structured explanation:

**Data Description:**

**CT/1 (Red line):** Represents the cell types count. It shows a sharp increase from Release 1 to Release 4, followed by a plateau, indicating a rapid expansion in identified cell types that stabilizes in the latest release.

**AS/1 (Blue line):** Tracks the counts of alternative splicing events. The trajectory follows a similar pattern to the CT/1 line, highlighting significant discoveries up to Release 4 with a slight dip and recovery by Release 5.

**BP/1 (Yellow line):** Illustrates the number of identified biological processes. The line rises steadily until Release 4, with a slight decline thereafter, suggesting a saturation in the identification of new biological processes.

**BGene/1 (Green line):** This line shows the count of background genes involved, which remains relatively flat across all releases, indicating a consistent use of a standard set of genes for comparison or control purposes.

**Interactive Features:** The graph is enhanced with cursor interaction, allowing users to hover over each point to view specific data counts at each release. This feature aids in the precise analysis of trends over time.

**Conclusion of Visualization**: The visualization effectively highlights the evolution of key biological data across software releases. The most notable trends include the rapid growth in the identification of cell types and alternative splicing events, which slightly tapers off by the fifth release, suggesting a move towards refining existing data rather than identifying new types. The stability in background gene data suggests consistent experimental conditions. The interactive elements of the graph provide a user-friendly tool for exploring these trends in more detail.

**Important Point to run the code:**

When running the code with **npm run dev**, the visualization won't be displayed immediately. Instead, after pressing enter and saving the file, the graph will then be rendered and shown. This delay in displaying the graph is due to the asynchronous nature of the data processing and visualization setup. Once the necessary data is processed and the chart is configured, saving the file triggers the rendering of the graph, making it visible for analysis.