**Data Analysis and Visualization**

**Semester Project**

**Submitted By :**

22L-7482 (Yousha Saibi)

22i-1859 (Omer Hafeez)

22i-1966 (Hafsa Shakeel)

**Submitted to :** Ma’am Daniya Jadoon

**Force Directed Layout**

 **Page Layout**:

* We chose a black background to create better contrast and highlight the content. Everything is centered for a clean and professional look.

 **Header and Summary**:

* The header (<h1>) displays the title "Force Directed" in white, placed at the top and centered.
* We added a summary section (<p class="summary">) to briefly explain the content, using a light gray font to make it visually softer.

 **Bootstrap Integration**:

* We integrated Bootstrap 5 for a responsive and consistent design. Classes like container-fluid, row, and col-lg-12 ensure that the layout adapts well to different screen sizes.

 **Chart Section**:

* The main feature is a card layout with an SVG placeholder (<svg id="fdl-svg">) where the force-directed chart will be rendered.
* We set the card dimensions to 1500px by 700px, providing ample space for the visualization.

 **Tooltip**:

* We included a tooltip (<div id="tooltip">) that remains hidden by default. It will dynamically appear during interactions with the chart for better user engagement.

 **External Libraries**:

* We’ve linked external libraries like D3.js, d3-scale-chromatic, and d3-geo-projection to handle data visualization tasks.
* The JavaScript file forcedirected.js will take care of rendering the force-directed chart and managing its interactivity.

**Key Features**

1. **SVG Setup**
   * The SVG container is initialized with dimensions (fwidth and fheight) and a viewBox for scaling. Styling ensures a responsive and centered layout.
2. **Data Processing**
   * Data is read from data\_with\_coordinates.csv using d3.csv. The script:
     + Groups rows by parentName.
     + Creates nodes for parents and links them to a central Earth node.
     + Adds child nodes for the top 10 entries per parent based on value\_latest\_year.
3. **Node and Link Structure**
   * Nodes:
     + Central node (Earth) with group: 0.
     + Parent nodes (group: 1).
     + Child nodes (group: 2).
   * Links:
     + **Parent links**: Between Earth and parent nodes.
     + **Child links**: Between parent nodes and child nodes.
4. **Force Simulation**
   * d3.forceSimulation governs the layout:
     + **forceLink**: Links nodes with varying distances (shorter for child links).
     + **forceManyBody**: Applies a repulsive force.
     + **forceCenter**: Centers the graph within the SVG.
     + **forceX and forceY**: Provides directional constraints.
5. **Node and Link Rendering**
   * Nodes:
     + Represented by circles.
     + Colors indicate hierarchy level:
       - Red for Earth.
       - Blue for parents.
       - Green for children.
   * Links:
     + Colored lines (#1f77b4 for parent links, #ff7f0e for child links).
6. **Tooltip**
   * A tooltip (FDLtooltip) appears on hovering over nodes, showing the id (location name).
   * Positioned dynamically relative to the mouse pointer.
7. **Drag and Zoom**
   * **Drag**:
     + Nodes can be dragged interactively.
     + Functions (dragstarted, dragged, dragended) ensure smooth motion and physics recalculations.
   * **Zoom**:
     + The graph can be zoomed and panned using d3.zoom.
8. **Interactivity**
   * Hovering nodes shows tooltips with details.
   * Dragging nodes adjusts their position dynamically.
   * Panning and zooming allow exploration of large datasets.

**Treemap**

This HTML snippet is an extension of the **Tuberculosis Data Analysis** webpage. We’ve structured it to include various interactive visualizations and tools to analyze data effectively. Here’s a detailed breakdown:

**1. Page Overview**

* The title, **Tuberculosis Q**, is displayed prominently at the top of the page, emphasizing the focus on tuberculosis data analysis.
* A summary paragraph describes the purpose of the dashboard, highlighting its interactive features like world maps, timeline animations, swarm plots, and sunburst diagrams.

**2. Visualizations and Features**

**Timeline Animation**

* A card contains a timeline visualization with controls:
  + A **play/pause button** allows users to animate through the timeline.
  + A **slider** (year-slider) lets users manually explore data from 1990 to 2017.

**World Map**

* A small card with a placeholder (map-svg) for a world map visualization. This will likely display data geographically using D3.js.

**Force-Directed Layout**

* Another card features a placeholder (fdl-svg) for a force-directed graph, enabling users to analyze relationships and connections in the data.

**Treemap Visualization**

* A larger card contains a treemap chart placeholder (treemap-svg) for hierarchical data representation.

**Sunburst Diagram**

* The final visualization is a **sunburst diagram** (sunburst-svg), styled with a bordered and light background for clarity.

**3. Responsive Design**

* We utilized **Bootstrap 5** for a flexible grid layout and responsive cards, ensuring the dashboard works well across different screen sizes.

**4. Styling and UX**

* The page has a black background with white text for contrast.
* Light gray is used for non-primary text, providing a clean and modern aesthetic.
* Tooltips and SVG elements enhance interactivity.

**5. External Libraries**

* The page integrates essential D3.js libraries (d3.v7.min.js, d3-scale-chromatic, d3-geo-projection) to handle data-driven visualizations.
* A custom script, worldmap.js, is included for the map rendering.

**Final Notes**

This dashboard combines multiple data visualization techniques into one cohesive design, providing users with various tools to explore tuberculosis data efficiently.

**Key Features**

1. **Loading Data**
   * The d3.json function loads data from a file called usa\_data.json.
2. **Tiling Function**
   * The custom tile function uses the d3.treemapBinary algorithm to partition the data into rectangular regions within the treemap.
3. **Hierarchy Creation**
   * The data is structured into a hierarchical format using d3.hierarchy. The hierarchy:
     + Summarizes values with .sum(d => d.Value).
     + Sorts nodes by descending values for better visual hierarchy.
4. **Color Scales**
   * **stateColors**: Assigns colors to top-level states using d3.schemeCategory10.
   * **shadeScale**: Adjusts color brightness for child nodes based on their value relative to their parent's value.
5. **SVG Container**
   * An SVG element (#treemap-svg) is initialized and styled to host the visualization. The viewBox ensures scalability.
6. **Rendering**
   * The render function creates the visualization:
     + Adds rect elements for each node and colors them based on hierarchy depth.
     + Appends text for node labels.
     + Adds interactivity: clicking on nodes zooms in or out.
7. **Zooming**
   * **Zoom In (zoomin)**: Focuses on a specific subtree by updating the scales (x, y) and transitioning the group.
   * **Zoom Out (zoomout)**: Returns to the parent node with smooth transitions.
8. **Positioning**
   * The position function calculates the dimensions and placement of each rectangle based on the hierarchy.
9. **Interactivity**
   * Clicking a node dynamically adjusts the view to zoom in or out, providing an engaging user experience.
   * Tooltips display hierarchical paths and values on hover.
10. **Error Handling**
    * Errors during data loading or processing are logged to the console.

**SunBurst**

**Core Functionalities**

1. **SVG Setup**
   * The viewBox centers the chart and ensures responsive scaling.
   * Radius is proportional to the width of the chart.
2. **Data Processing**
   * The JSON data is organized into a hierarchy using d3.hierarchy and processed to calculate values and sort nodes.
3. **Color Mapping**
   * A rainbow palette (d3.interpolateRainbow) differentiates categories, creating a visually appealing chart.
4. **Segment Rendering**
   * Arc paths are calculated based on hierarchical layout, ensuring proportional representation.
5. **Interactivity**
   * **Zoom:** Clicking a segment zooms into its children while transitioning arcs and labels.
   * **Labels:** Adjust visibility and position dynamically based on segment size and zoom level.
6. **Tooltips**
   * Static titles show hierarchical paths and values when hovering over segments.
7. **Parent Circle**
   * A clickable central circle zooms out to the root node, simplifying navigation.

**Design Choices**

* **Compact Layout:** Small gaps between arcs minimize visual clutter.
* **Dynamic Labels:** Visible only for sufficiently large segments, ensuring readability.
* **Smooth Transitions:** Enhances user experience during zoom-in/out actions.

**Visualization Dashboard using Bootstrap and D3.js**

**Key Components**

1. **Bootstrap Integration**
   * Provides responsive design with container-fluid, row, and col classes.
   * Cards are used for organizing content into distinct sections.
   * Styling includes margins, paddings, and headers to enhance appearance.
2. **Dashboard Layout**
   * **Row 1:**
     + **World Map**: Displays a geographic visualization using an SVG with ID map-svg.
     + **Timeline Animation**: Includes a play-pause button and a range slider for dynamic visualization over years.
   * **Row 2:**
     + **Swarm Plot**: Reserved space for a detailed swarm plot with fdl-svg.
     + **Treemap Visualization**: Provides hierarchical data exploration with a breadcrumb navigation and level filter.
3. **Custom Styles**
   * Full-screen background with optional image or color (black by default).
   * Font styling for tooltips, headers, and chart elements.
   * Enhanced visuals with rounded corners (border-radius), padding, and card layouts.
4. **Interactivity**
   * **Tooltips**: Appears dynamically on hover, positioned absolutely, and styled for visibility.
   * **Treemap Navigation**: Breadcrumbs and a dropdown filter allow users to jump between hierarchy levels.
5. **Scripts and Libraries**
   * **D3.js**: Used for rendering visualizations (worldmap.js, timeline-viz.js, etc.).
   * **Bootstrap JS**: Adds interactivity to buttons and elements.
   * Additional libraries like d3-scale-chromatic and d3-geo-projection extend visualization capabilities.

**Timeline Visualizations**

**Key Features:**

1. **Scales and Axes:**
   * The X-axis represents the timeline from 1990 to 2017.
   * The Y-axis represents the death rates of different age groups.
2. **Multiple Age Groups:**
   * The code visualizes age groups such as < 5, 5-14, 15-49, 50-69, and > 70 with distinct line colors.
3. **Legend:**
   * A clickable legend is implemented, allowing users to filter the lines by age group.
4. **Tooltip and Hover Line:**
   * When hovering over the chart, a tooltip shows the year and the death rate for each age group.
   * A vertical dashed line highlights the current year.
5. **Play-Pause Animation:**
   * The timeline is animated to show the data changing over time. Users can also manually explore the timeline using a slider.
   * The play button controls the animation, and the slider lets you jump to specific years.
6. **Right Axis (Relative Rates):**
   * A right-side Y-axis shows the percentage change in death rates between the starting and ending years.