Assignment Report: Advanced Interactive Dashboards via D3.js

Course: Data Visualization & Analytics

Submission Date: November 30, 2025

Introduction

This report documents the step-by-step implementation of the performed tasks. It details the technical approach ("How") and the design rationale ("Why") for each component of the two major tasks: **Temporal & Spatial Simulation (Task 1)** and **Temporal & Hierarchical Simulation (Task 2)**. The goal was to move beyond static charts to create production-grade, performant, and interactive dashboards using pure D3.js (v7).

Task 1: Spatial & Network Analysis on Global Power Plants

Step 1: Data Loading & Centralized State Management

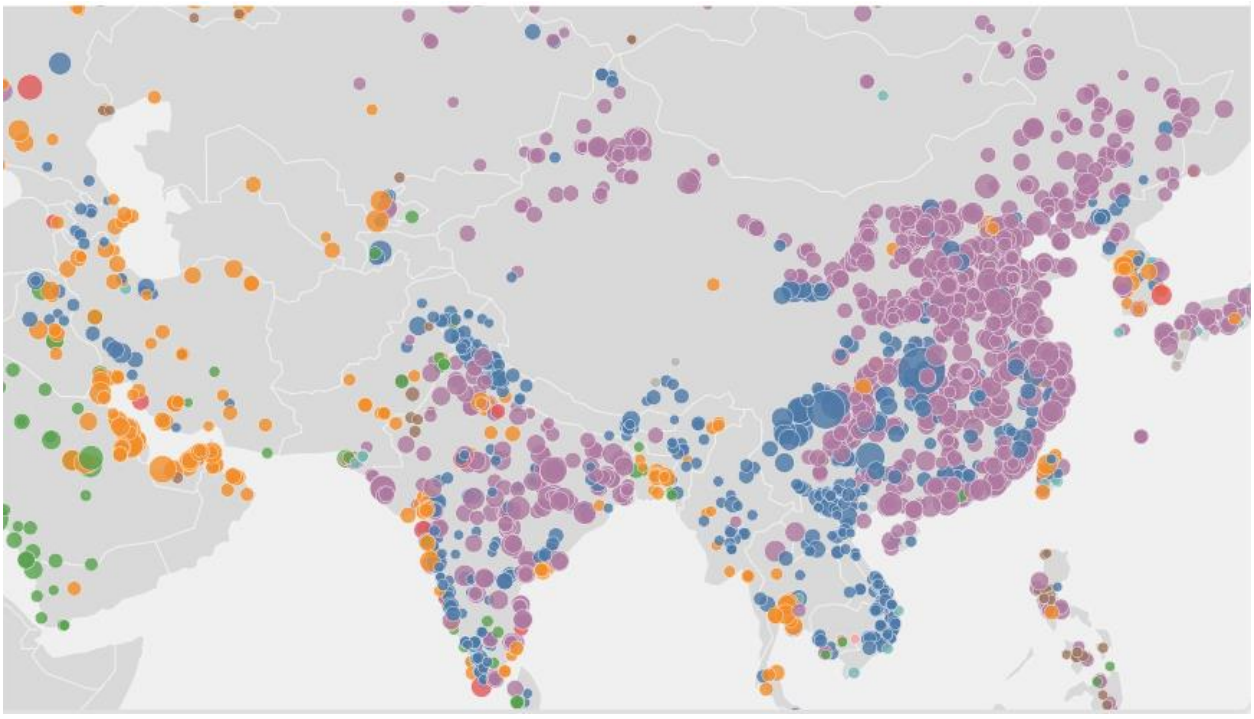
- **How:**
  - Implemented a central *main.js* script that uses *Promise.all* to load the *global_power_plant_database.json* and *world.geojson* files in parallel.
  - Created a reactive *state* object to store the raw data and current filter parameters.
  - Designed an *updateGlobalState()* function that acts as the "brain" of the dashboard. It filters the raw data based on current parameters and propagates the filtered dataset to the Map, Fuel Chart, and Timeline simultaneously.
- **Why:**
  - **Performance:** Asynchronous loading prevents the browser from freezing while fetching large datasets.
  - **Linked Views:** A centralized state is the backbone of interactive dashboards. It ensures that when a user interacts with one chart, all other charts update simultaneously to reflect that specific slice of data.

Step 2: Spatial Visualization (The Map & Semantic Zoom)

- **How:**
  - Used *d3.geoMercator* and *d3.geoNaturalEarth1* projection to render a base world map.
  - **Semantic Zooming:** Implemented logic to change the visualization based on zoom level using *event.transform.k*:
    - **Zoom Out (k < 2.5):** Shows aggregated "Country Bubbles" using *d3.rollup* to sum capacity per country. We used *pathGenerator.centroid(feature)* to calculate the mathematical center of each country shape for placement.
    - **Zoom In (k > 2.5):** Switches to individual power plant points colored by fuel type.

- **The "USA" Data Fix:** Addressed a data discrepancy where the GeoJSON used "United States of America" but the CSV used "USA". A normalization step was added to ensure US data aggregated correctly.

- **Why:**
  - Semantic zooming prevents visual clutter. Showing 35,000 points on a zoomed-out map results in over plotting i.e. too many points cluttered making no sense. Aggregating by country provides a high-level summary, while zooming in reveals the necessary granular detail.

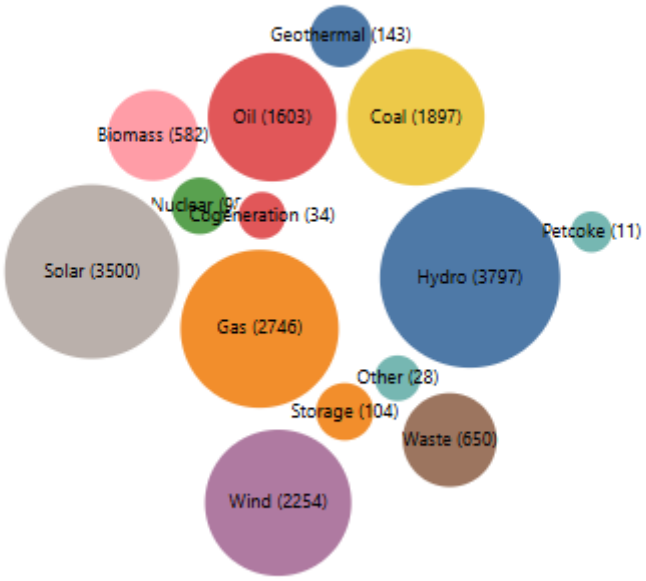


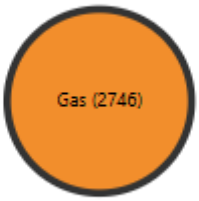
Step 3: The Challenge of Performance Optimization

- **How:**
  - **Pre-calculation**: Instead of calculating the projection logic (Latitude/Longitude) 35,000 times per frame during a zoom event, we calculate *d.x* and *d.y* once during initialization and store them in the data object.
  - **Viewport Filtering:** We implemented a scraping algorithm inside the render loop. Before drawing, the code calculates the visible boundaries of the screen (based on the current zoom transform). It filters the dataset to render *only* the points currently visible in the viewport.
- **Why:**
  - **Preventing DOM Thrashing:** Rendering 35,000 SVG elements simultaneously causes severe browser lag due to low frame rates. By only drawing the 500 points actually visible on the screen, we achieved a smooth 60 fps zooming and panning.

Step 4: Attribute Visualization (Fuel Type Chart)

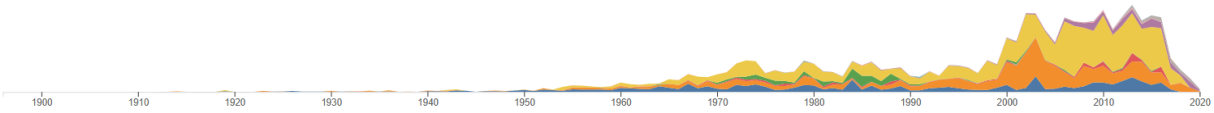
- **How:**
  - Created a Force-Directed Bubble Chart using *d3.forceSimulation*.
  - Forces applied: *charge* (repulsion), *center* (gravity), and *collide* to prevent overlap.
  - Implemented an *.update(data)* method that re-calculates node counts whenever the filtered dataset changes.
  - Added interaction: Clicking a bubble updates *state.selectedFuel*, triggering a global filter.
- **Why:**
  - **Distribution Analysis:** Allows users to instantly see the mix of energy sources.
  - **Filtering Interface:** It acts as a control panel. Clicking "Solar" filters the complex map to show only solar plants, making spatial patterns of specific energy types easier to identify.



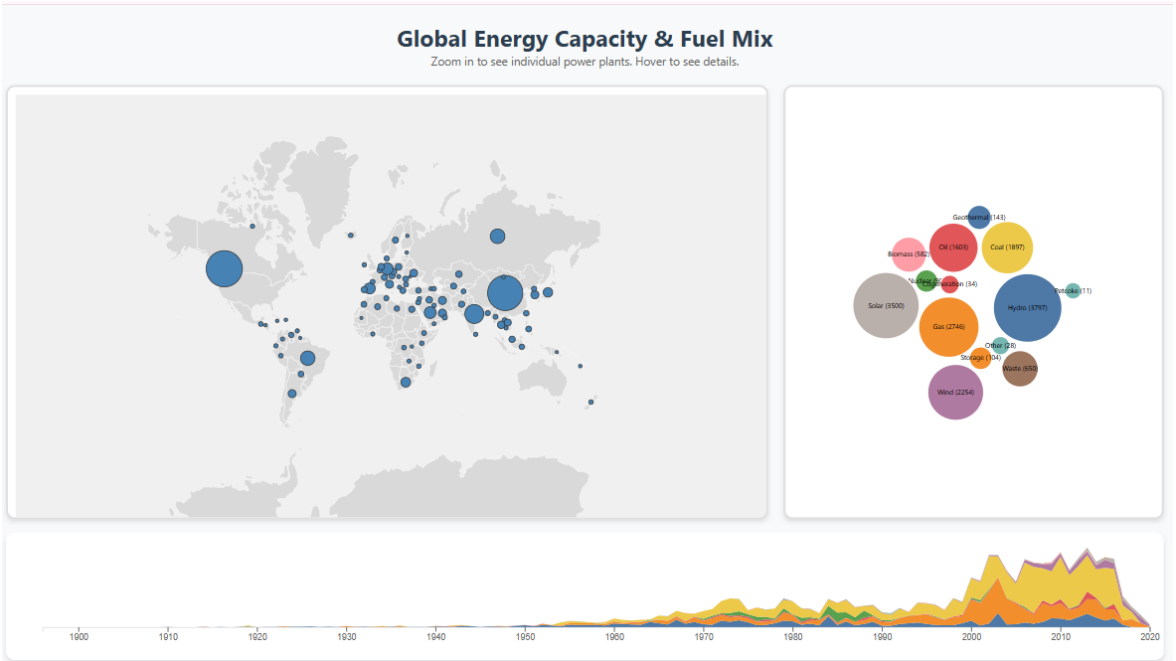


Step 5: Temporal Visualization (Stacked Area Timeline)

- **How:**
  - Built a Stacked Area Chart showing cumulative capacity added per year.
  - Integrated *d3.brushX*. When the brush is dragged, it calculates the year range and sends it to *main.js.*
- **Why:**
  - **Temporal Context:** A static map shows *where* plants are, but not *when* they were built.
  - **Drill-Down:** The brushing tool allows users to isolate specific historical periods, creating a fully linked spatial-temporal exploration tool.



Combined Dashboard Output:



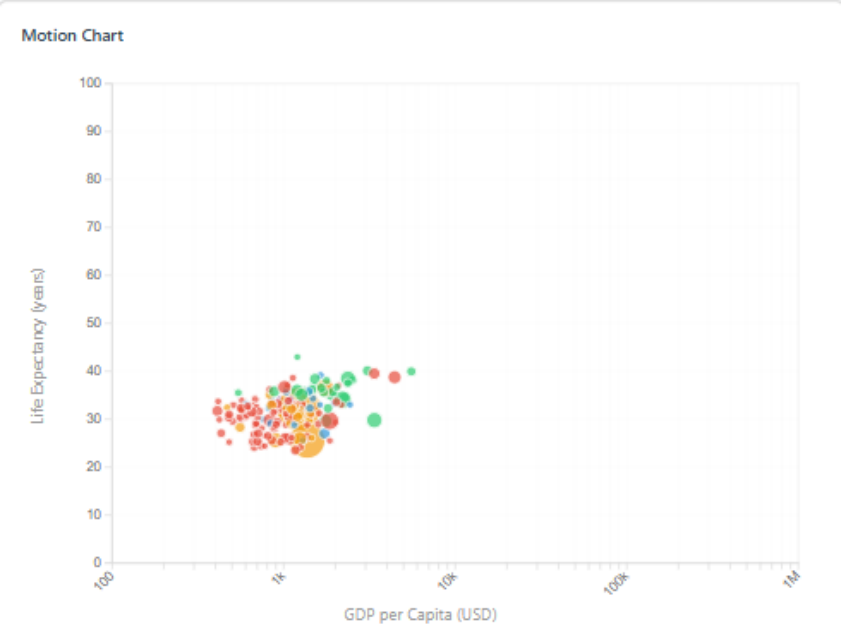
Task 2: Temporal & Hierarchical Simulation on World Data

Overview

This dashboard visualization shows us the correlation between GDP per capita and life expectancy over time.

Component 1: The Motion Chart (Animated Scatter Plot)

- **Implementation:**
  - **X-Axis:** Logarithmic Scale *(d3.scaleLog)* for GDP per Capita.
  - **Y-Axis:** Linear Scale for Life Expectancy.
  - **Radius:** Square Root Scale *(d3.scaleSqrt)* for Population.
  - **Animation:** Used *d3.interval* to create a game loop that increments the year every 500ms. Used *d3.transition* with linear interpolation to ensure bubbles move smoothly between years without jumping.
- **Design Rationale:**
  - **Log Scale:** Essential for economic data. A linear scale would compress developing nations into an unreadable cluster.
  - **Sqrt Scale:** Mapping population directly to radius would make large countries appear exponentially larger than they are.

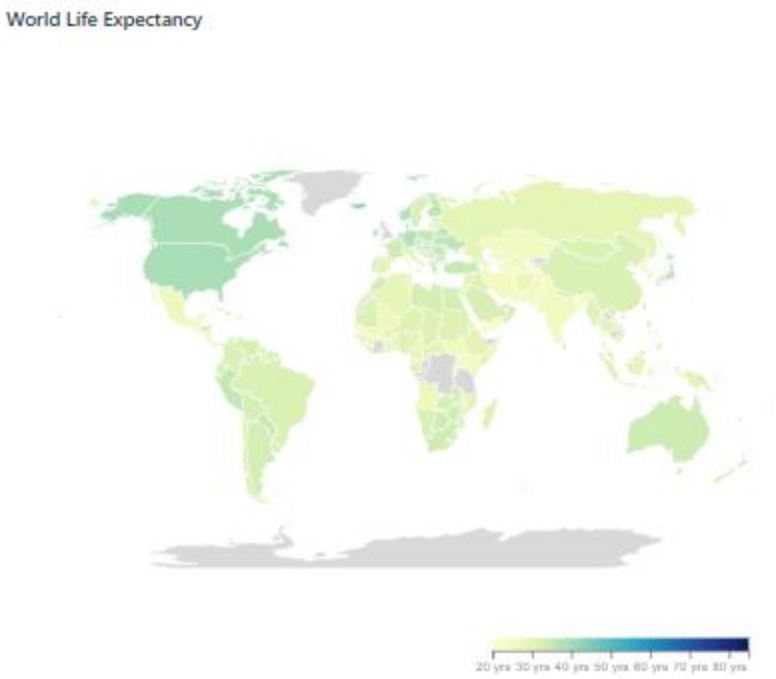


Component 2: Synchronized Choropleth Map

- **Implementation:**
  - Rendered a world map using *d3.geoNaturalEarth1*.
  - Used *d3.scaleSequential* with the *d3.interpolateYlGnBu* color scheme to encode Life Expectancy.

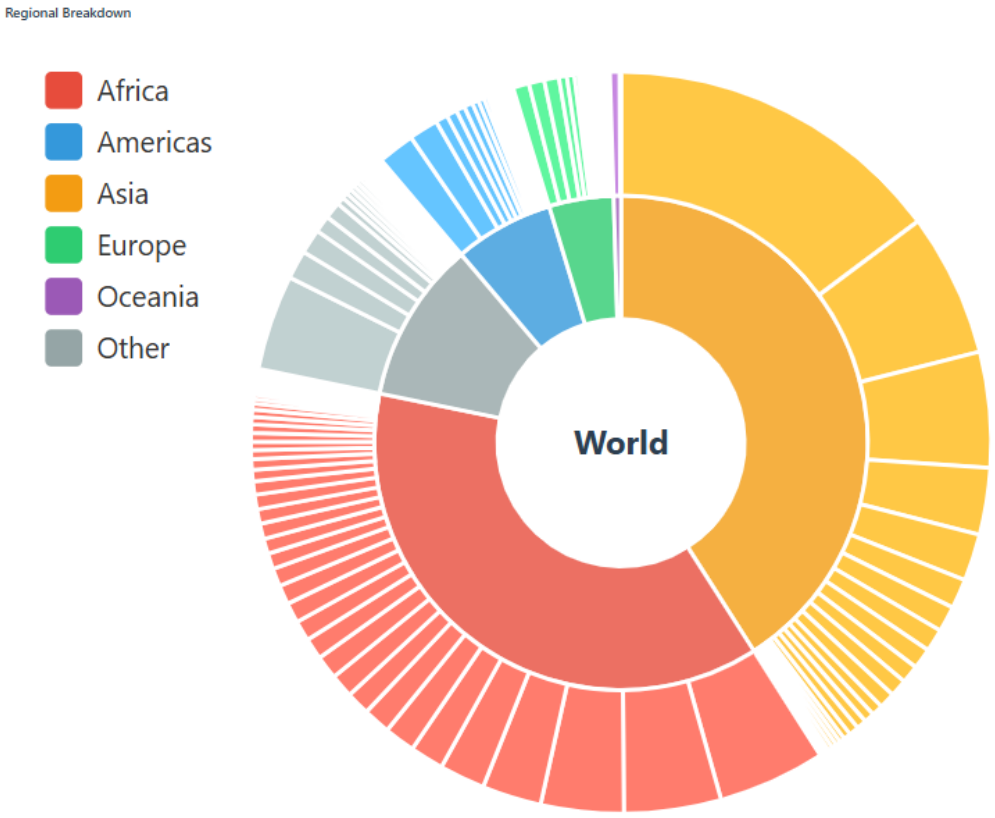
- **Synchronization:** The map exposes an *.update(year)* method called by the Motion Chart's animation loop, ensuring the map colors evolve in real-time with the scatter plot.

- **Design Rationale:**
  - **Spatial Context:** The scatter plot is abstract. The map reveals regional clusters of health trends e.g., the visible difference between Western Europe and Sub-Saharan Africa that might be missed in the scatter plot alone.



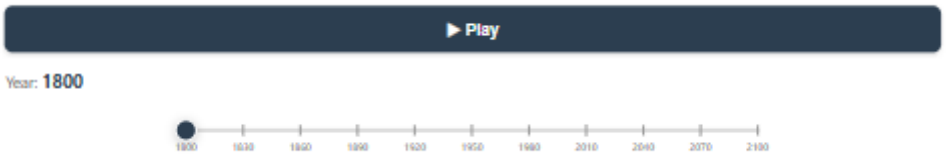
Component 3: Hierarchical Drill-Down (Sunburst Chart)

- **Implementation:**
  - Used *d3.hierarchy* and *d3.partition* to create a Sunburst chart displaying the heirarchy: World, Continent, Country.
  - **Interaction:** Hovering over a continent slice highlights corresponding bubbles in the Motion Chart by lowering the opacity of non-matching continents.
- **Design Rationale:**
  - **Summary & Filtering:** The Motion Chart contains hundreds of bubbles. The Sunburst provides a high-level summary and acts as a "Drill-Down" mechanism to focus on specific regions.

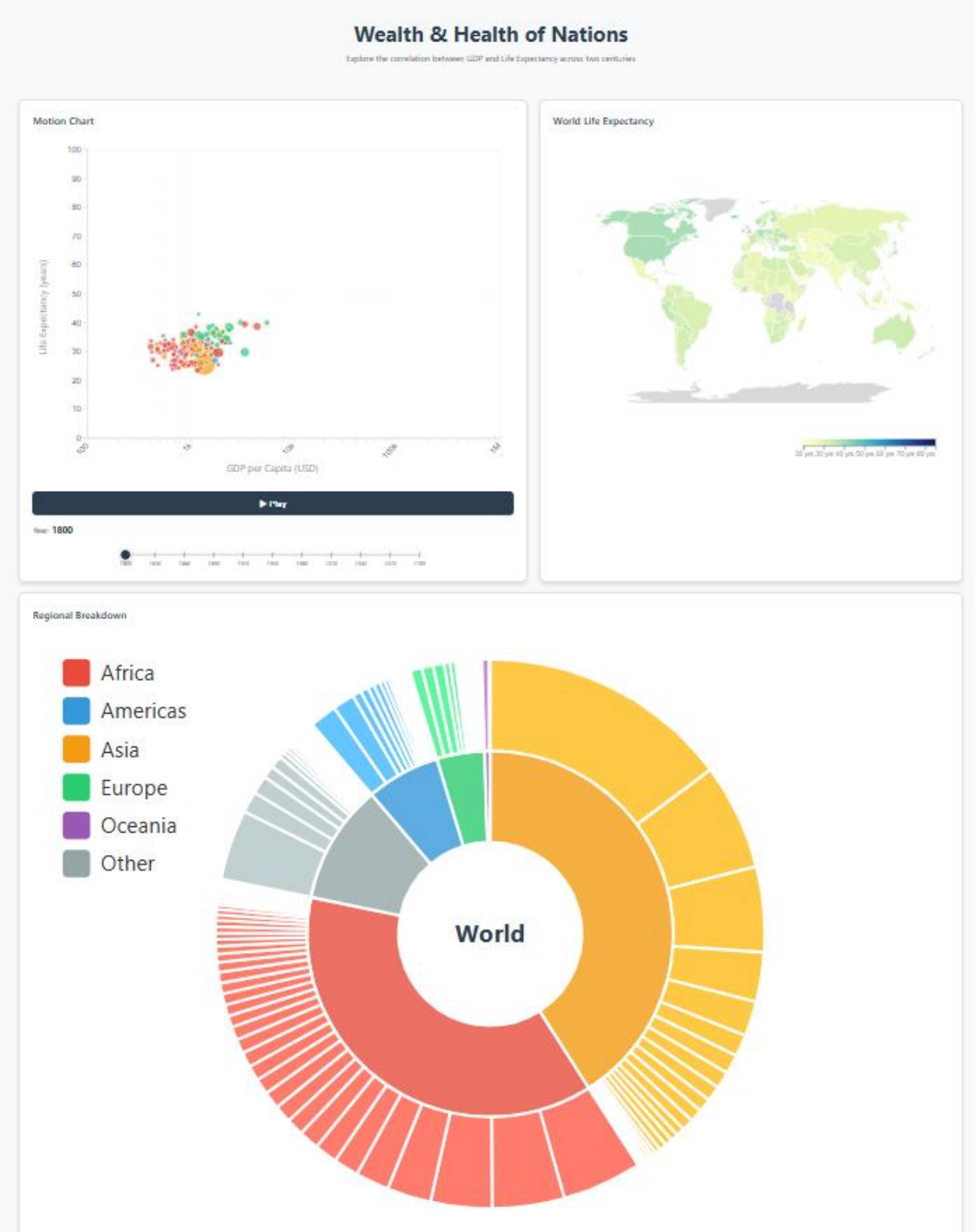


Component 4: Interactive State Controls (Slider)

- **Implementation:**
  - Implemented a draggable slider using *d3.drag*.
  - **Two-way Binding:** Dragging the slider pauses the animation and updates the year. Conversely, playing the animation automatically moves the slider handle.
- **Design Rationale:**
  - **User Experience:** Consistency is key. The user shouldn't feel like they are interacting with separate widgets. The synchronization ensures a cohesive storytelling experience.



Combined Dashboard Output:


Wealth & Health of Nations
Explore the correlation between GDP and Life Expectancy across two centuries
Motion Chart
Life Expectancy (years)
GDP per Capita (USD)
▶ Play
Year: 1800
World Life Expectancy
Regional Breakdown
Africa
Americas
Asia
Europe
Oceania
Other
World