Data Analysis & Visualization Task

Problem Statement:

For Task 1:

The dataset contains tens of thousands of global power plants, making it difficult to analyze spatial patterns, fuel types, and historical trends using static or traditional visualizations. The challenge is to build an interactive dashboard that can efficiently handle large data while allowing users to explore locations, fuel categories, and commissioning years in a clear and responsive way.

For Task 2:

There is a need for an interactive and intuitive way to explore how global health and wealth indicators have changed over time. Traditional static charts cannot effectively show relationships, regional differences, or long-term trends. The challenge is to build a synchronized, multi-view dashboard that clearly visualizes these development patterns across countries and continents.

In order to solve these problems, followings steps and strategies have been taken

1. DATA ARCHITECTURE

Efficient Loading

Data is loaded in parallel using:

```
async function loadData() {
  const [csv, topo] = await Promise.all([
    d3.csv("global_power_plant_database.csv", d3.autoType),
    d3.json("https://cdn.jsdelivr.net/npm/world-atlas@2/countries-110m.json")
  ]);
```

CSV parsing is handled automatically by d3.autoType, reducing manual conversions. Same for 2nd dataset *country_data.json* also uses promise.all()

Grouping / Nesting (Efficient Restructuring)

How

Data is grouped in three critical ways:

1. **By Country**
   Used for semantic zooming and map aggregation.
2. **By Fuel Type**
   Used by the Force-Directed Fuel Clustering.
3. **By Year**
   Used by the Brushable Timeline.

This is implemented through Map() objects for O(1) lookup and minimal overhead. Grouping once and caching results dramatically improves performance compared to recomputing aggregates on every zoom or filter action.

2. ADVANCED MAPS

Projection & GeoPath

The dashboard uses:

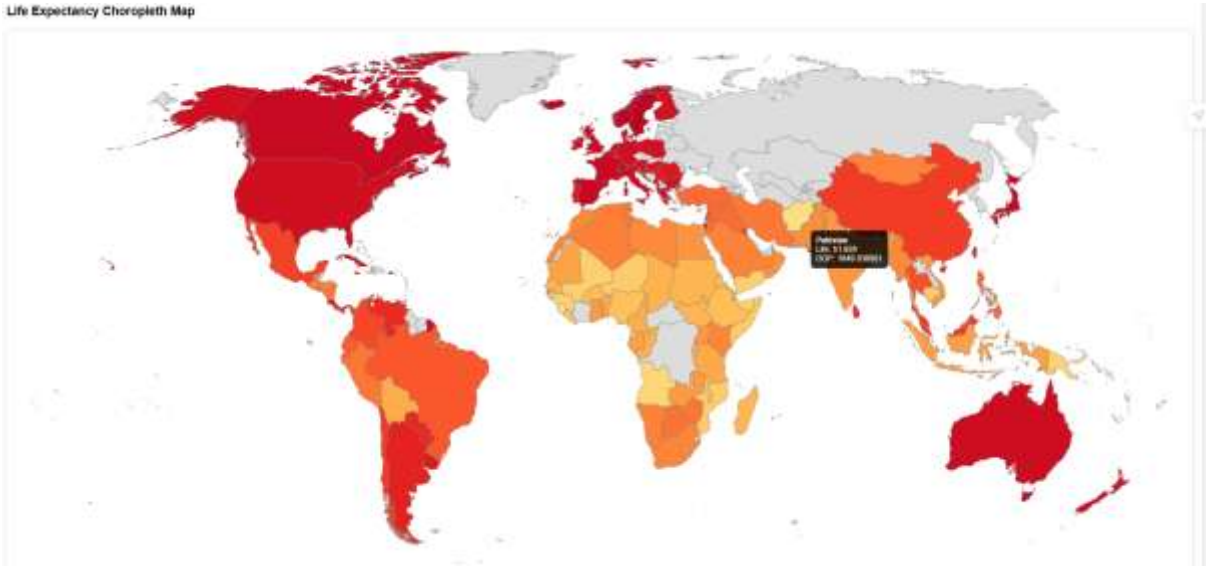
```
projection = d3.geoNaturalEarth1()
    .scale(CONFIG.MAP_WIDTH / 6.5)
    .translate([CONFIG.MAP_WIDTH/2, CONFIG.MAP_HEIGHT/2]);

path = d3.geoPath().projection(projection);
```

- **NaturalEarth1** provides a visually appealing balance of shape accuracy and readability.
- The projection centers the world without distortion at poles in a way suitable for global datasets.

Dynamic Color Updates (Choropleth Sync)-Task 2

We used a sequential color scheme **interpolateYlOrRd**.

.fill(d => colorScale(lifeExpValue));



- Bright yellow = low life expectancy
- Dark red = high life expectancy
- Color scale is intuitive → widely used in demographic studies

Zoom & Pan

A d3.zoom() behavior modifies the transform of both country shapes and map layers:

```
  zoomBehavior = d3.zoom()        Chat (CTRL + I) / Share (CTRL + L)
    .scaleExtent([1, 16])
    .on("zoom", (event) => { currentTransform = event.transform; mapLayer.attr("tra
  mapSvg.call(zoomBehavior);
}
```

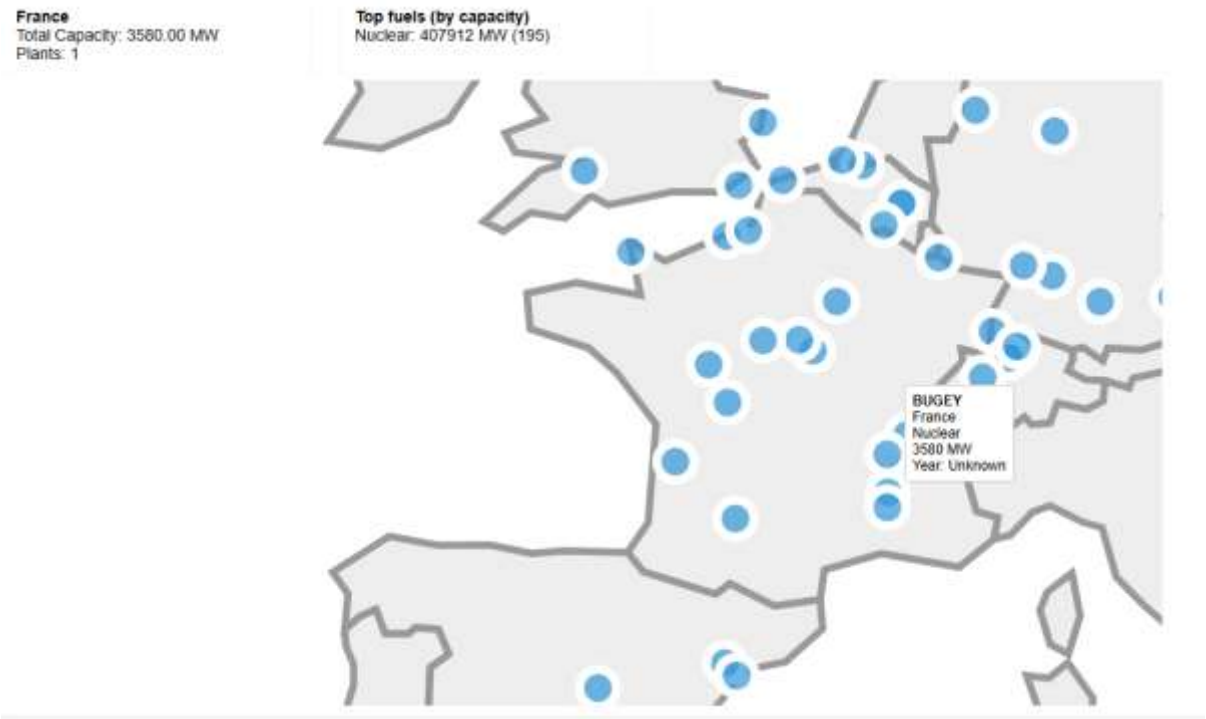
- Supports semantic zooming.
- Helps users explore plant-level details without clutter.

Semantic Zooming-Task 1

Based on zoom scale:

- Zoom < 4.5 → render **country aggregates**

- Zoom >= 4.5 → render **individual power plants**



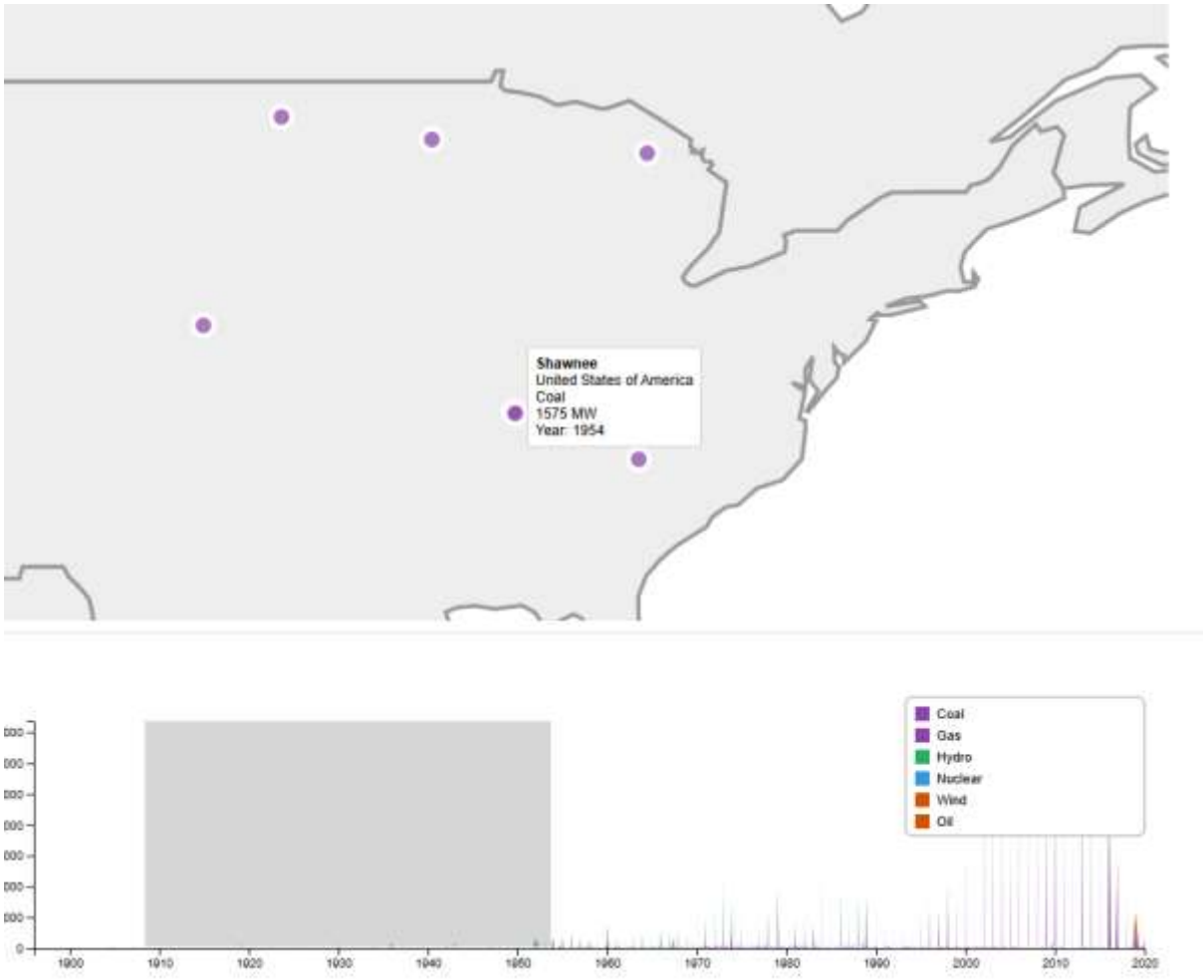
Rendering ~35,000 circles at once can crash browsers.
Semantic zooming ensures optimal performance and clear visual meaning.

3. INTERACTIVITY — BRUSHING

A timeline renders stacked area charts representing capacity added per year.
A d3.brushX() rectangle enables selection of year ranges:

The brush triggers immediate updates to:

- World Map (plant or country circles)
- Fuel Force Cluster
- Stats panels



Brushing is essential for temporal exploration—allowing users to:

- Identify growth periods for certain fuels.
- Understand historical development of countries.
- Filter high-density map data to manageable time slices.

Dragging & Slider (d3-drag)

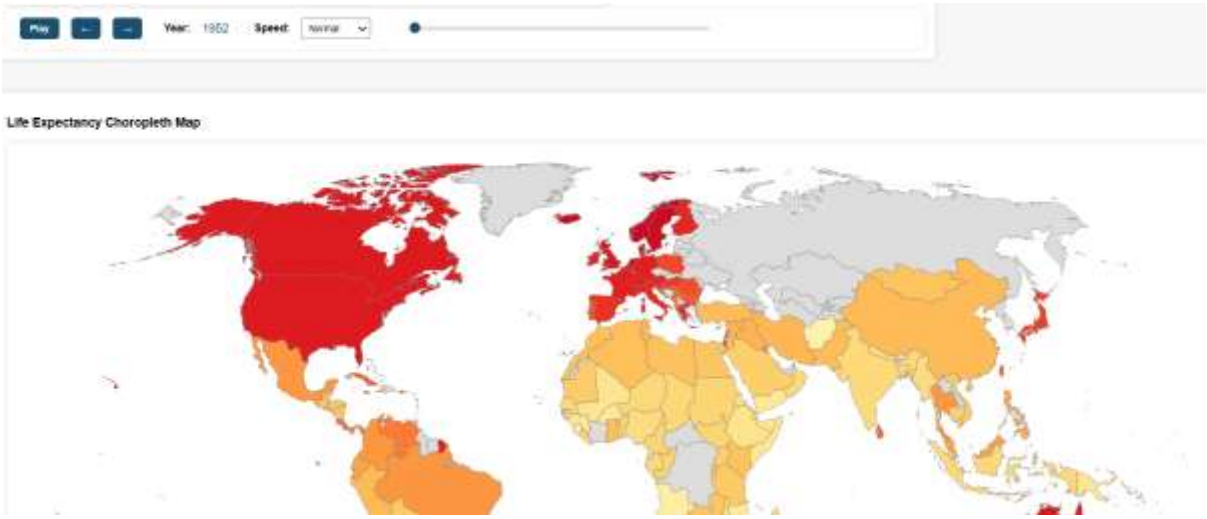
Custom Slider to Scrub Year Timeline

The slider uses **d3.drag()** to manually change year:

```
if (!sliderHandle.empty()) {
  const drag = d3.drag()
    .on("start drag", (event) => {
      stopPlaying(); // stop auto play
      const b = sliderBounds();
      const pct = Math.max(0, Math.min(1, (event.sourceEvent.clientX - b.left) / b.width));
      const yr = pctToYear(pct);
      // find index and set
      const idx = years.indexOf(yr);
      if (idx >= 0) { currentYearIndex = idx; updateToYearIndex(currentYearIndex, 0); }
    })
    .on("end", ()=> {});
  sliderHandle.call(drag);
```

How It Works

- Drag handle left → earlier years



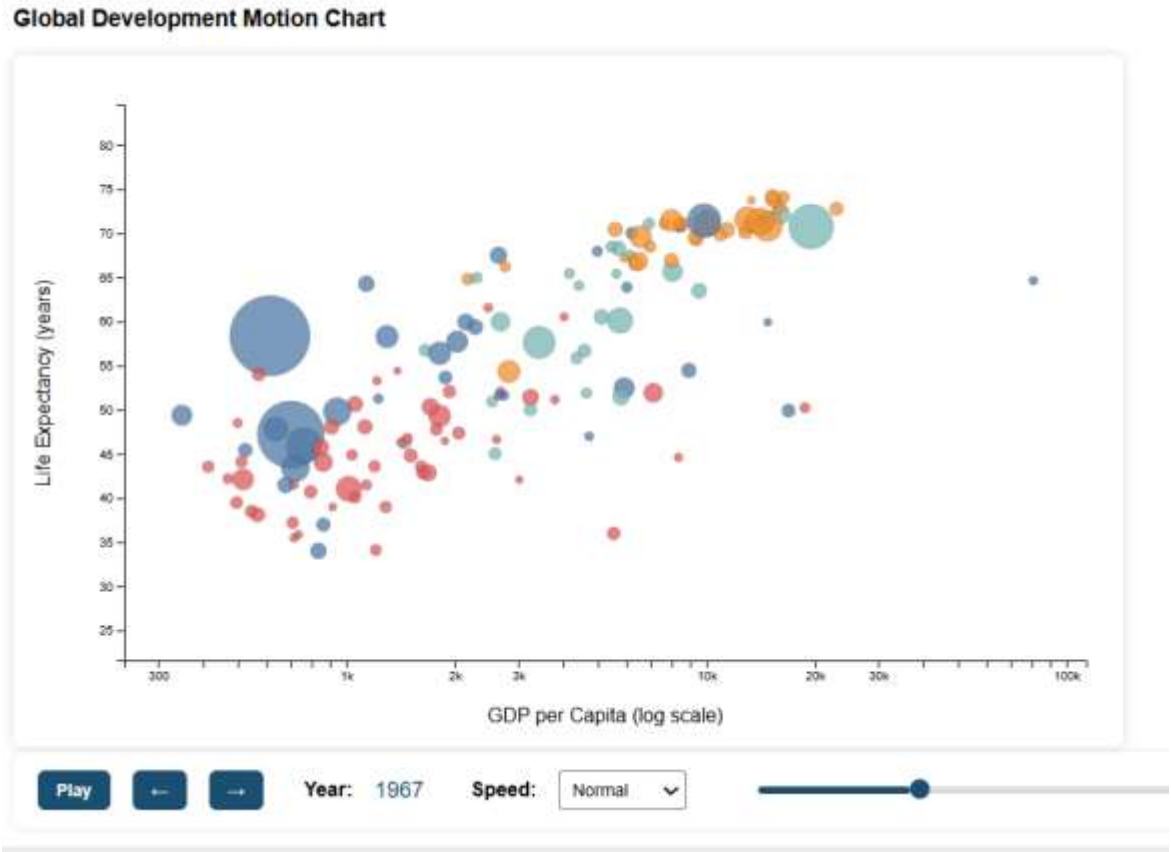
- Drag right → future years

4. ANIMATION LOGIC

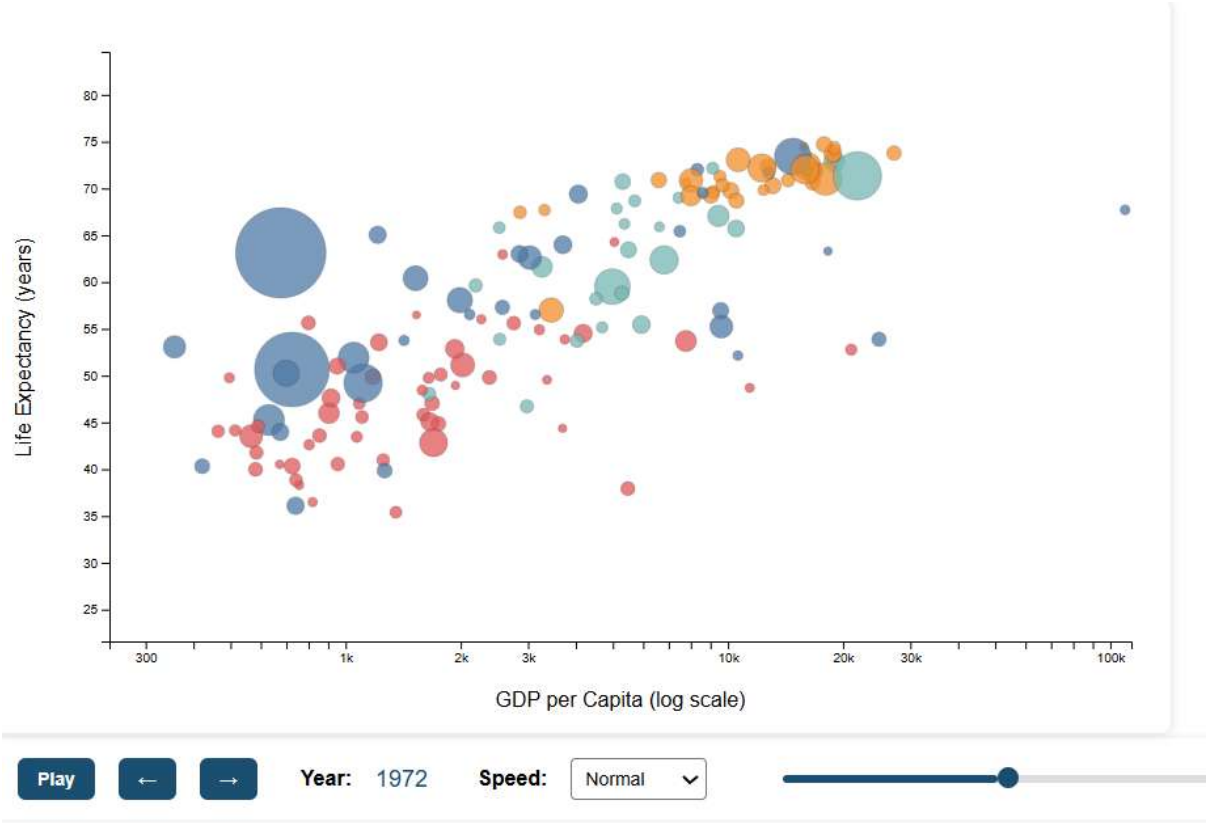
Using d3.interval for Play/Pause logic

```
playInterval = d3.interval(()=>{
  const nextIdx = (currentYearIndex + 1) % years.length;
  updateToYearIndex(nextIdx);
}, duration + 100);
```

Visual Representation:



Smooth transition from 1967 to 1972



Smooth Transitions

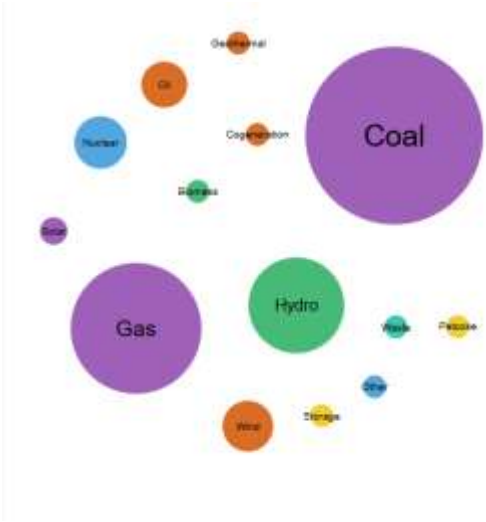
Transitions are used for circle radius, opacity, and force simulation movement:

```
enter.transition().attr("r", d => Math.max(3, Math.sqrt(d.capacity) / 50));
circles.merge(enter)
  .attr("cx", d => projection(d.centroid)[0])
  .attr("cy", d => projection(d.centroid)[1]);
```

The force layout uses .sim*ulation.alphaTarget(0.3).restart();*

- Avoids visual jumps.
- Helps highlight data changes intuitively.
- Shows dynamic clustering behavior without overwhelming the GPU.

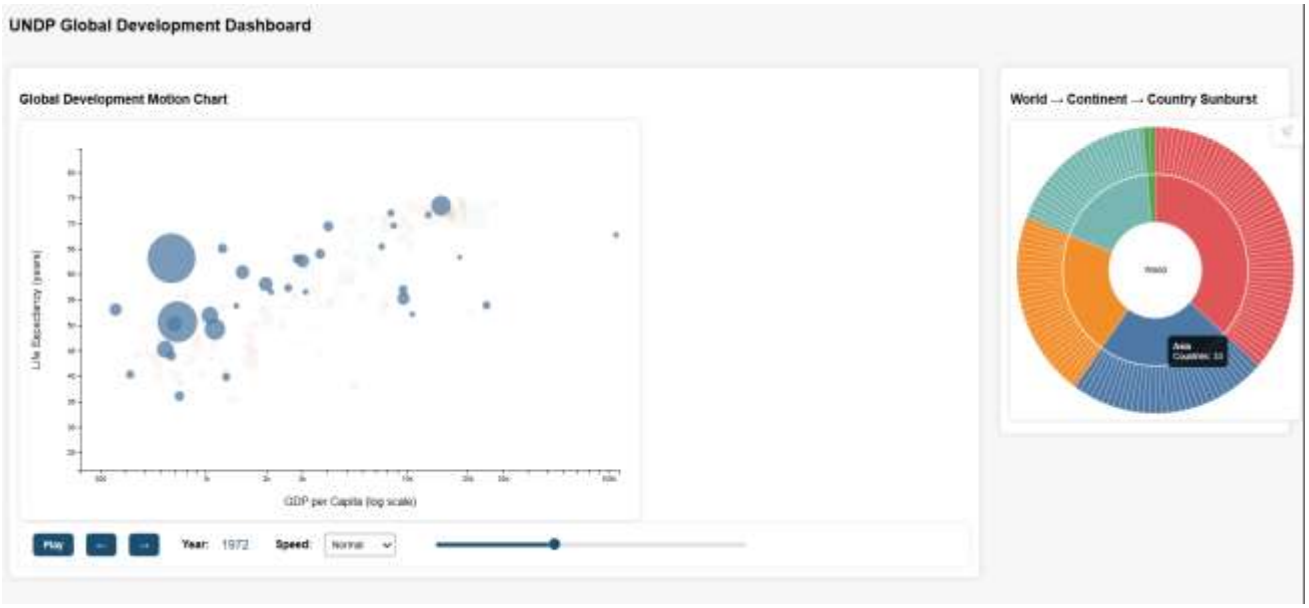
from 1900 to 1965



from 1965 to 2020

5. VISUAL ENCODING

For 1st task we used d3.scaleLog () for GDP per capita, d3.scalelinear (), For Life Expectancy, d3.scaleSrt () for Population (Bubble radius).Also the feature of continent selection was implemented and results are shown below:



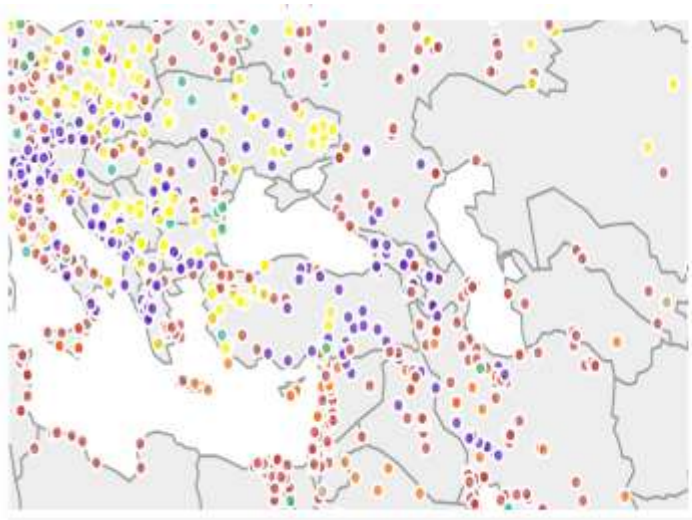
Scales

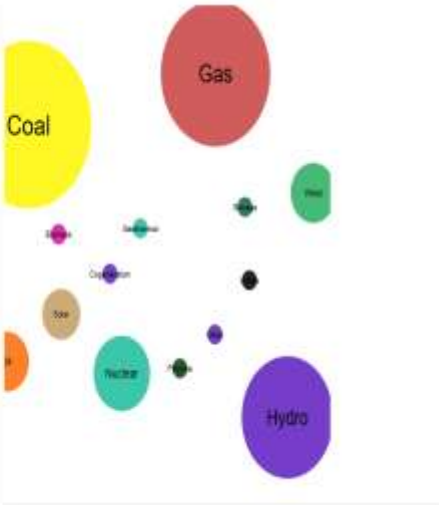
- Circle size uses sqrt(capacity) to maintain perceptual accuracy.
- Timeline uses d3.scaleLinear() for proper year spacing.
- Color scheme assigned deterministically to fuel types.

Correct scaling ensures:.

- Avoid misinterpretation of bubble sizes.
- Colors remain consistent across map, cluster, and timeline.

- Capacity differences appear proportional





6. UI/UX & POLISH

Tooltips

Small floating panels display contextual details:

- Plant name
- Capacity
- Fuel type
- Country
- Commissioning year

Labels & Titles

Each dashboard region includes:

- “World Map”
- “Fuel Cluster”
- “Timeline – Capacity Added Per Year”

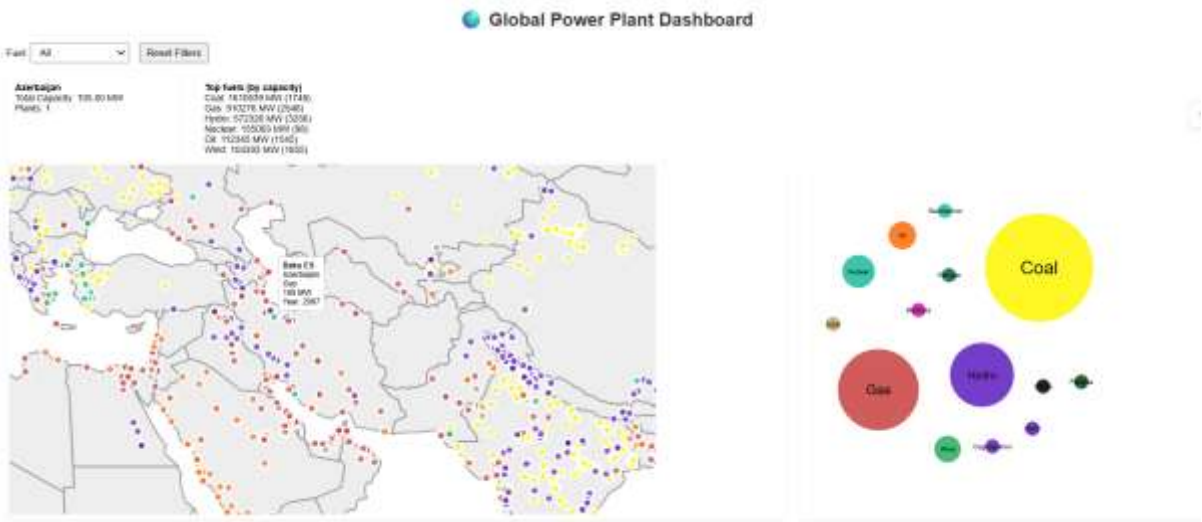
Layout

CSS Grid layout:

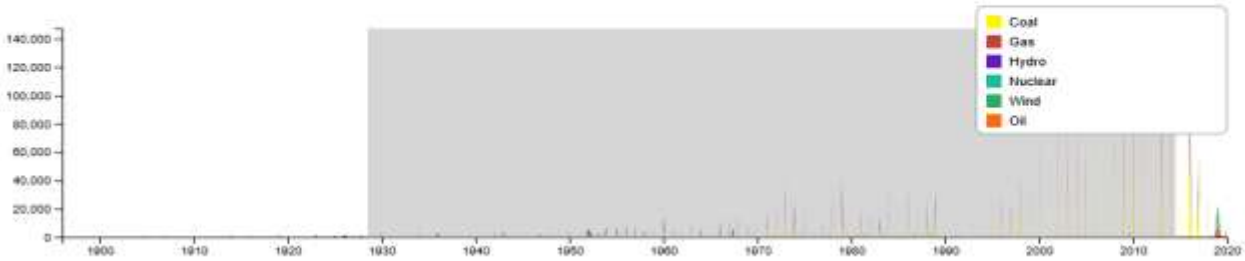
- 65% width map
- 35% width force cluster
- Full-width timeline

7. Final Dashboards

Global Power Plant Dashboard



Time Line – Capacity Added Per Year



Features:

1. Global Interactive Map

- Renders a world map using the NaturalEarth projection.
- Displays power plant data across ~35,000 points without performance lag.
- Uses semantic zooming to switch between country-level and plant-level visualization.
- Supports smooth zoom & pan interactions using mouse or touch.
- Displays country aggregates or individual plants depending on zoom level.
- Colors all plant markers based on fuel type.
- Shows real-time tooltips on hover containing:
    - Country name
    - Capacity
    - Number of plants
    - Fuel type

2. Semantic Zooming (Smart Detail Switching)

- **Zoomed Out (Scale < 4.5)**:
    - Shows **country-level circles** representing *total capacity*.

    - Bubble size proportional to country's aggregated MW.
- **Zoomed In (Scale ≥ 4.5)**:
    - Switches to **individual power plant circles**.
    - Up to 8,000 plants rendered safely without crashing.
    - Plant size scaled by capacity.

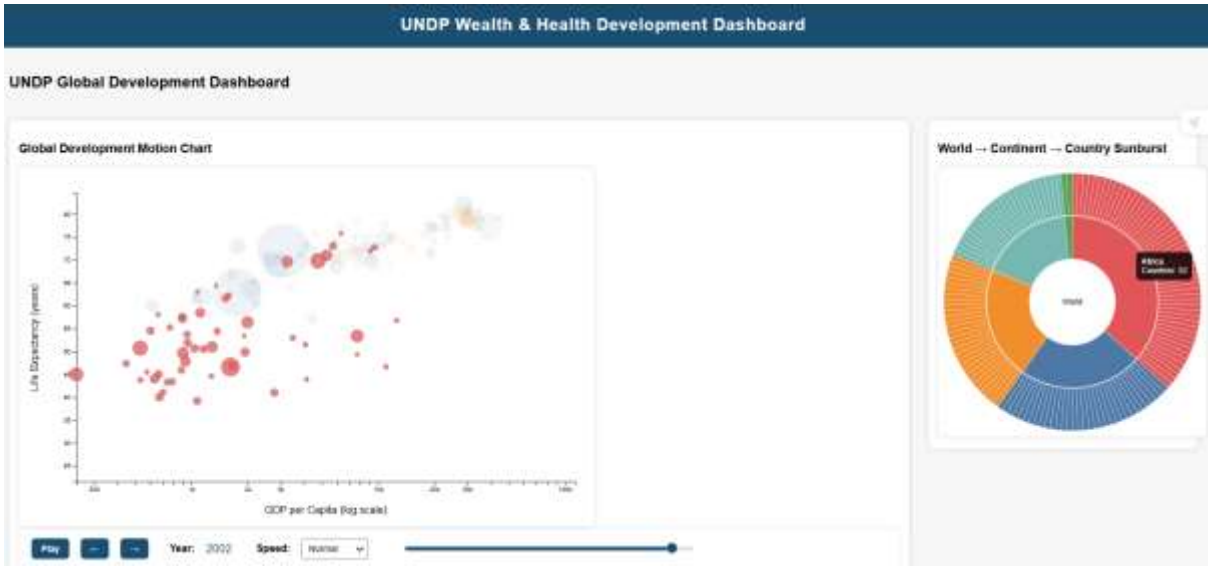
3. Force-Directed Fuel Cluster

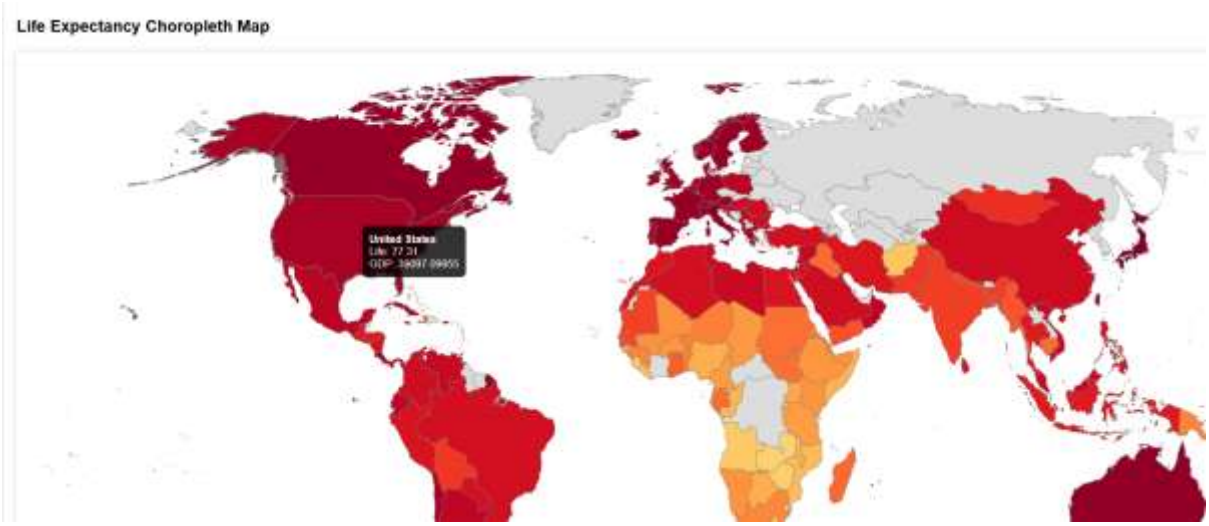
- Visualizes each **fuel type** as a bubble (Solar, Coal, Hydro, Nuclear, etc).
- Circle size represents total capacity of that fuel globally (or within filters).
- Automatically animates into position using force simulation:
    - Collision detection (avoid overlaps)
    - Gravity toward center
- Each bubble is clickable (acts as a **global filter**).
- Clicking a fuel bubble:
    - Filters map + timeline to **only show plants of that type**.
    - Clicking again toggles filter off.

4. Timeline

- **Click + drag** to brush a year range.
- **Release** to apply filter.
- **Click empty space** to clear brush.
-

2. **UNDP Global Development Dashboard**





Dashboard Features

- **Motion Chart (Scatter Plot)**
  Shows the relationship between **GDP per Capita (wealth)** and **Life Expectancy (health)** over time.
  Bubble **size = population**, bubble **color = continent**, and bubbles **move smoothly** as years change.
- **Choropleth Map**
  Displays global **Life Expectancy distribution** for the currently selected year.
  Countries are shaded using a **yellow→red gradient**, giving a quick visual of regional health levels.
  Updates **in real time** whenever the year changes in the motion chart or slider.
- **Sunburst Hierarchy (World → Continent → Country)**
  Shows how countries are grouped by continent.
  Hovering a continent **highlights only that region** on the motion chart by fading all other bubbles.
  Helps users drill down to regional development patterns.
- **Year Slider (Manual Scrubbing)**
  Allows users to **manually scrub** through years (1952–2007) by dragging a handle.
  Instantly updates the map and scatter plot, stopping any automatic animation.
- **Play/Pause Animation Controls**
  Users can animate historical progress, step forward/backward by year, and adjust animation speed.
  Automatically synchronizes scatter plot movement and map color transitions.
- **Tooltips & Hover Interactions**
  Hovering over any country (bubble or map) reveals detailed statistics:
  GDP, Life Expectancy, Population, Continent — without cluttering the screen.

Conclusion

The developed dashboard successfully visualizes complex power plant data through an interactive map, fuel clustering, and a brushable timeline. It enables fast filtering, smooth navigation, and clear insights across geography, fuel type, and time. The result is a simple, responsive, and effective tool for understanding global energy patterns.

The dashboard successfully visualizes global development trends through a synchronized motion chart, choropleth map, sunburst hierarchy, and interactive controls. It enables users to

easily compare countries, explore regions, and observe changes over time. Overall, it provides a clear, engaging, and accessible way to understand how wealth and health have evolved worldwide.