Project Title: Analyzing Trends In The Current Job Market

Group Members:

Apoorva Gerasa Shankarappa Gowda : 34510679

Amizthan Madan : 34426804

Ahrthi Suresh : 33987082

GitHub Link: https://github.com/ahrthisuresh/Analaysis_Of_Data_Science_Jobs_2024

Background and Motivation

The global job market has seen significant shifts in recent years, with opportunities expanding across various sectors, regions, and roles. In this evolving landscape, it has become important to analyze how jobs are distributed geographically, what kinds of roles are most in demand, and how experience levels and industries factor into hiring trends. Our dashboard aims to provide a comprehensive view of the data science job landscape using interactive and visual tools to support career planning and workforce analysis.

With the growing interest in remote and hybrid work environments, as well as emerging roles within data science and adjacent domains like AI, analytics, and machine learning, understanding geographic and domain-level patterns is crucial for aligning career strategies. Additionally, this work helps highlight disparities in job distribution and opens discussions about access to opportunity based on region and experience level.

Domain Goals and Audience

The main goal of this project is to explore and visualize global trends in data science job postings across roles, domains, and experience levels. The intended audience includes:

- Students seeking data science roles across companies and locations
- Recruiters and HR professionals
- Academic advisors and career services
- Policymakers and educators tracking data science job market trends and sector-specific shifts

The visualization aims to help these users identify where jobs are concentrated, how different regions and industries are evolving, and what levels of experience are most in demand. By combining geographic, industry, and experience-level filters, the tool can support highly specific career exploration.

Tools and Frameworks

Our preprocessing was done using **Python** with the Pandas library. The front-end visualizations were built entirely using **D3.js v7**, and the visualizations were served using a local server (`python3 -m http.server`) to allow CSV/JSON loading.

The project folder was organized with:

- `/data`: Cleaned and exported datasets (CSV/JSON)
- `/js`: JavaScript logic for each dashboard
- `/styles`: CSS
- Root HTML files for each dashboard
- Python for data preprocessing

Dashboard Design & Navigation

Like in the example, highlight that:

- Each dashboard (Skills, Industry, Timeline, Map) has its **own dedicated page**.
- The homepage acts as a **navigation hub**, guiding users to different insights.
- Color palettes, fonts, and structure were kept consistent for a unified experience.

Design Evolution

Structure this like:

1. Industry Dashboard

- Started with a horizontal bar chart
- Transitioned to a pie chart, then a **donut chart** to improve layout and make space for a percentage-based legend.
- Used `d3.pie`, `d3.arc`, and `d3.scaleOrdinal` for rendering and styling.

2. Timeline Dashboard

- Initial approach: weekly bar chart
- Realized the dataset only covered ~2 weeks → switched to **daily aggregation**
- Final form: **line chart** showing posting trends over time using `d3.line`, `d3.scaleTime`, and tooltips

3. Map Dashboard

- Started as a bar chart grouped by location
- Finalized as a **choropleth map** using TopoJSON + GeoJSON and `d3.geoPath()` with a sequential color scale
- Country names were matched between data and world map objects

---

Data

Our dataset, `dashboard_cleaned_data.csv`, was sourced from Kaggle. It contains the following relevant columns:

- `date`: The posting date of the job
- `location`: The geographic location of the job listing (country-level)
- `industry`: The domain or sector (e.g., Technology, Healthcare, Education)
- `company`: Name of the hiring company
- `skill`: A comma-separated string listing required skills
- `level`: The seniority or experience requirement of the role (e.g., Entry, Mid)

This raw dataset included a mix of structured and semi-structured fields (especially the `skill` column), as well as inconsistencies in text capitalization, whitespace, and missing values.

Data Preprocessing

To ensure the data was usable for interactive visualization in D3.js, We authored a Python preprocessing script, data_preprocessing.py. This script serves as the backbone of all dashboards by producing clean, transformed, and aggregated data in visualization-ready formats.

The cleaning steps include:

Type Enforcement and Validation

- **Date Parsing**: Ensured all values in the date column were converted to datetime objects using pandas.to_datetime() with error handling.
- **Numeric Coercion**: Converted count-based aggregations to integers where necessary.

Standardization

- **Whitespace and Formatting**:
  - Trimmed extra spaces in the skill column using .str.strip() and split by comma for counting.
  - Standardized location and industry values to reduce duplicates from casing or formatting variations.
- **Missing Data**:
  - Dropped rows missing critical fields like location, industry, or date.

Aggregation and Export

- **Industry Aggregation** → export_industry_trends()
  Groups job postings by industry and exports a summary CSV used in the donut chart.
- **Location Aggregation** → export_location_data()
  Filters and formats location-level data for the geographic visualization (map).
- **Job Trends Over Time** → export_job_trends_over_time()
  Groups jobs by posting date and exports as a CSV to power the timeline line chart.
- **Skills Parsing** → export_skill_data()
  Explodes comma-separated skills into a long format and counts occurrences.

Each function outputs cleaned data into either .csv or .json, ready to be loaded directly into a D3 dashboard.

The preprocessing was modularized to allow clean scalability across different visualization needs.

The script outputs the following files:

- industry_trends_data.csv → [industry, counts]

- location_data.csv → [location, industry, level]
- job_trend_over_time.csv → [date, count]
- skill_demand_data.json → [{ skill, count }]

These files are placed in the /data folder and consumed via d3.csv() or d3.json() in the corresponding dashboard scripts.

In future stages, additional datasets may be merged for skill-level analysis, company-wise breakdowns, or job trend timelines using fields such as date and title.

Exploratory Data Analysis

At the outset of our project, we engaged in exploratory data analysis (EDA) to uncover patterns, validate data quality, and identify key variables to visualize. This step played a critical role in shaping our dashboard design.

Visualizations Used

Bar Chart (Preliminary – Industry & Skills Frequencies)

We used bar charts to get an initial sense of distribution across industry sectors and top-mentioned skills. This helped us detect whether a few categories were dominant or whether the data was more evenly spread.

Line Chart (Preliminary – Job Postings Over Time)

We plotted job posting dates on a time-series line chart. Our initial grouping by week yielded too few data points, prompting a shift to daily aggregation for finer resolution. This early view revealed fluctuations in demand over short time spans.

Frequency Tables (Skills & Industries)

We generated raw frequency tables for categorical fields like industry, level, and location. These tables helped us determine thresholds (e.g., top 10 skills) for final visualizations and revealed messy/inconsistent entries that needed standardization.

Geographic Heat Sketch (Pre-map Check)

Before implementing the choropleth map, we generated a simple bar chart of job counts per country to confirm geographic coverage. This helped us assess whether we had enough non-US data to justify a global map, and also surfaced mismatches in country names (e.g., “USA” vs “United States”).

Insights Gained

Skill Dominance

Certain skills like Python**,** SQL**, and** Data Analysis were mentioned significantly more than others. This prompted us to focus our top skills visualization on the top 10, while also cleaning up inconsistencies (e.g., removing redundant whitespace, unifying capitalization).

IT Sector Leads

During the early analysis, the IT / Data Analytics industry appeared as a clear leader in job postings. This influenced our decision to use a donut chart for industry breakdown — highlighting dominance while maintaining readability for smaller sectors.

Short-Term Time Range

We discovered that the dataset spanned only ~2 weeks of job posting activity. This insight led to a switch from weekly to daily aggregation in the timeline dashboard to extract as much detail as possible from the limited time frame.

Country Name Inconsistencies

Before building the map dashboard, EDA revealed that several country names in the dataset didn’t align with TopoJSON country identifiers (e.g., “USA” vs “United States of America”). This required a normalization step to ensure proper data binding during map rendering.

Analysis Tasks

This milestone focused on:

- Aggregation: Counting jobs by country using a custom parser
- Geographic analysis: Displaying the global spread using choropleth shading
- Tooltip interactivity: Allowing users to hover and view job counts per country

Skill Demand Comparison: We analyze the frequency of various technical and soft skills mentioned in job postings to identify the most in-demand competencies in the data science field. This helps users—especially students and job seekers—prioritize which skills to learn based on current market trends.

Industry-Wise Breakdown: We explore which industries are hiring the most data science professionals. Initially visualized with a bar chart, we transitioned to a donut chart to better communicate proportional representation across a small number of categories while keeping the visualization clean and focused.

Temporal Trends in Job Postings: We perform a time-series analysis to track how job postings fluctuate over time. By plotting daily posting counts, we gain insights into hiring patterns and temporal spikes in demand, helping to identify active hiring periods.

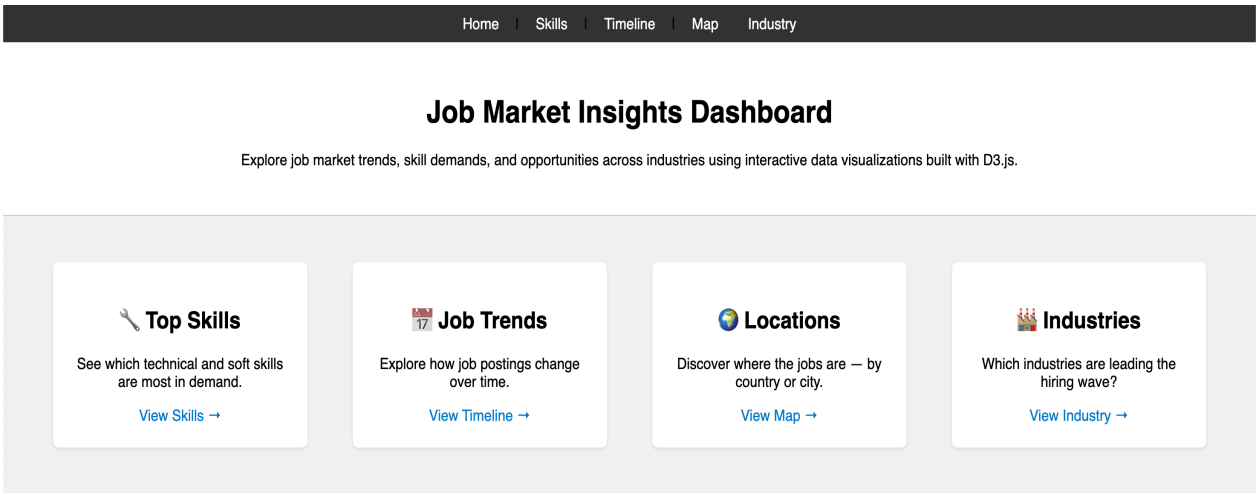
Geographic Job Distribution: We use a choropleth map to visualize the number of job postings across countries. This spatial mapping highlights global hubs for data science opportunities and allows for easy comparison of regional hiring volumes.

Planned future analysis:

- Filtering by experience level (Entry, Associate, Mid-senior, Senior)
- Industry segmentation: Visual breakdown of job types and sectors across regions
- Timeline-based visualizations to show how job openings have evolved over time.

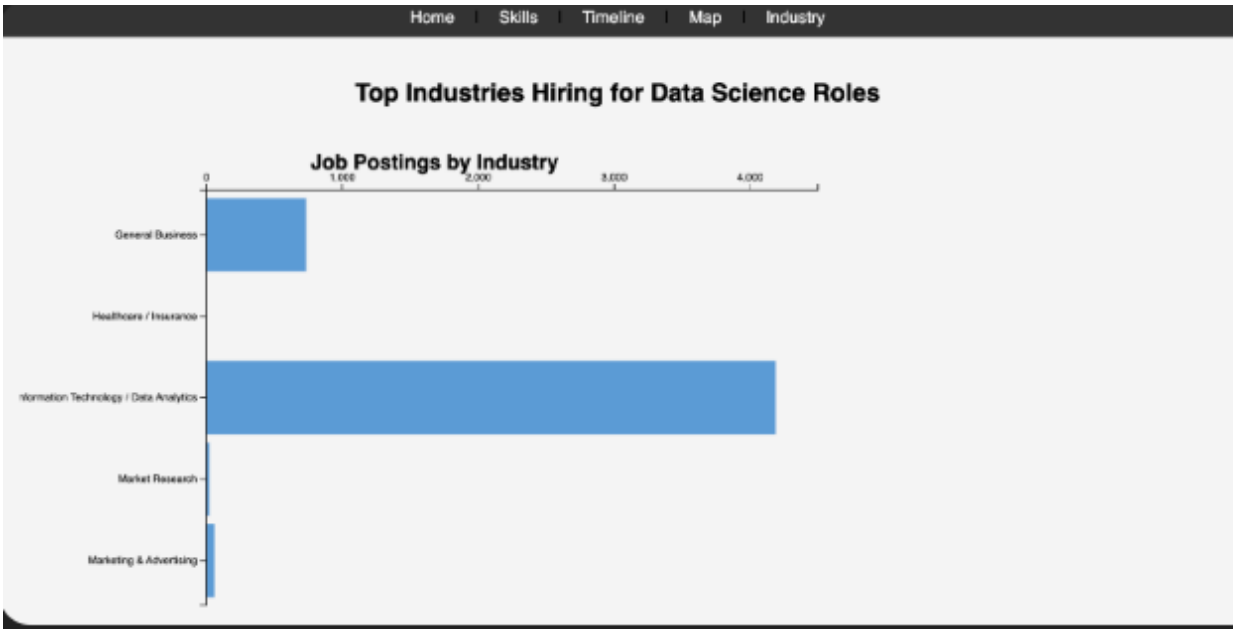
Design Overview and Implementation

To provide users with a cohesive entry point into the project, we designed a centralized home dashboard that introduces the core dimensions of our analysis: Skills, Job Trends, Locations, and Industries. Each category is represented through a card-based layout that is both visually appealing and functionally intuitive. The landing page offers brief contextual descriptions and direct navigation links, allowing users to explore individual dashboards seamlessly. This modular design ensures that users can quickly grasp the scope of our project while encouraging deeper exploration based on their interest.

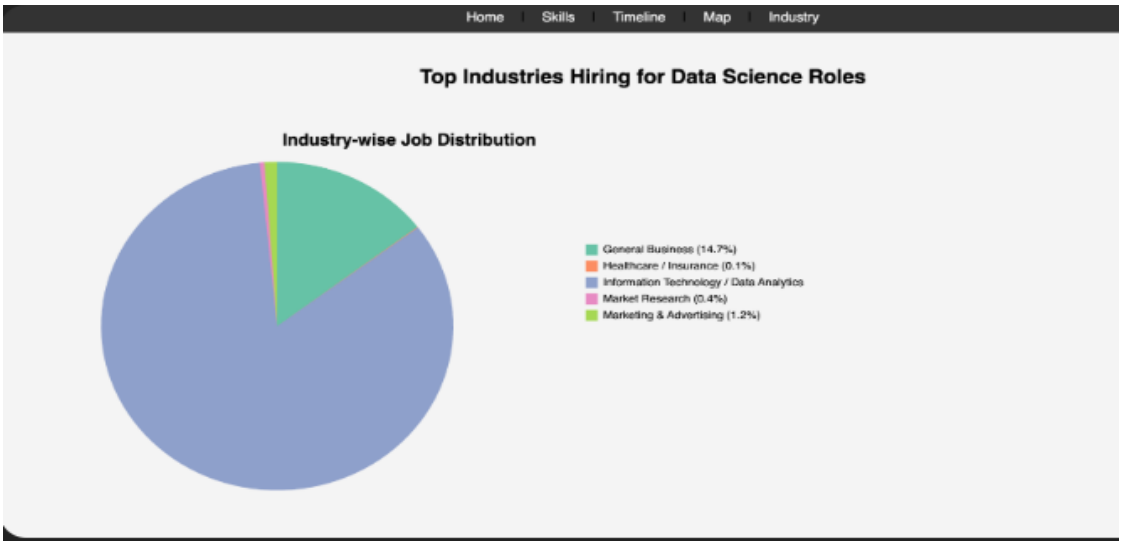


Industry-wise Job Distribution:

Initial Attempt: we proposed a horizontal bar chart to display industry-level job counts. However, as the number of industries grew, bar chart readability became a challenge — especially for categories with similar frequencies.



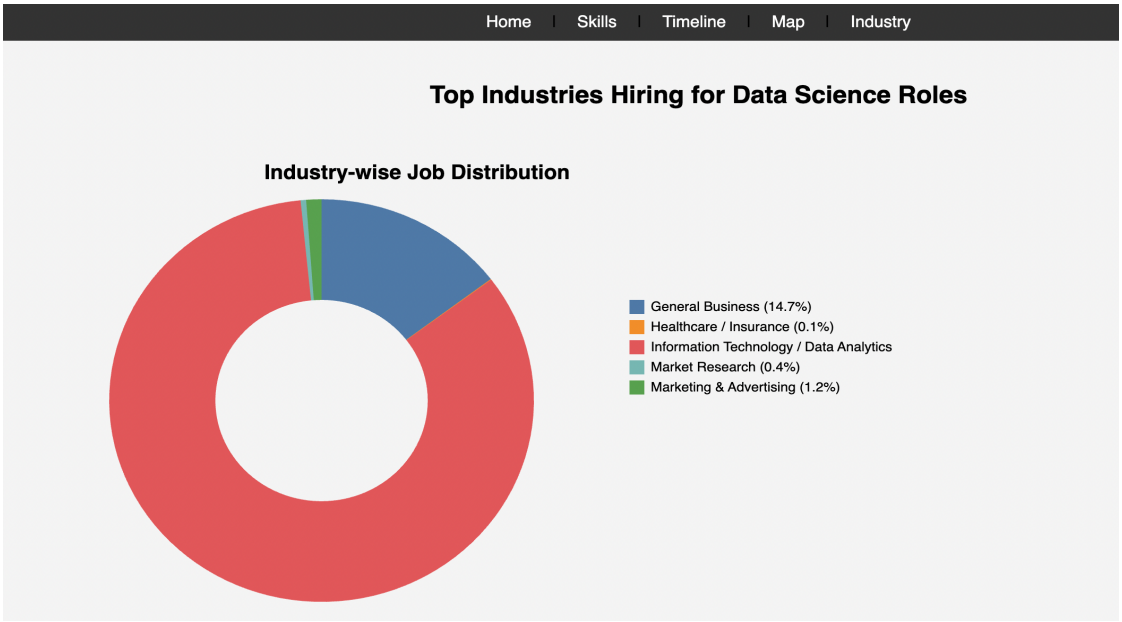
Second Attempt: We shifted from a bar chart to a pie chart because the bar chart became visually overwhelming with numerous industry categories and long text labels. The pie chart allowed us to represent proportions more intuitively, making it easier to compare industry share at a glance.



Final output: Finally we decided on a **donut chart**, which:

- Saves space
- Is visually intuitive for comparison of proportions
- Easily accommodates a **percentage-based legend**

We also explored label placement *inside* slices, but this proved ineffective for smaller segments. Instead, we positioned a **clean, right-aligned legend** that includes both the industry name and its relative percentage share.



We implemented an industry-wise job distribution dashboard using a **D3.js donut chart**. The goal was to provide a compact yet informative way to visualize which industries dominate the data science job market.

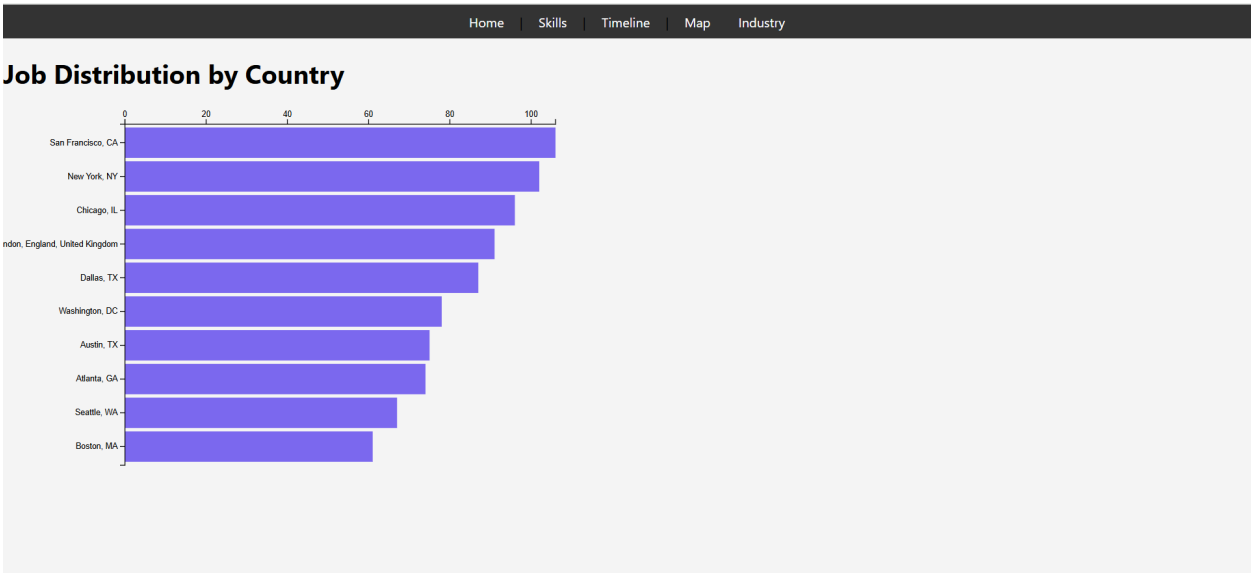
To construct the donut chart:

- We first grouped job posting data by industry using a Python preprocessing script.
- This grouped dataset was exported into industry_trends_data.csv, which was later consumed by the D3 visualization.
- A color scale from the d3.schemeCategory10 palette was used to differentiate slices.
- To make the chart more readable, we designed a legend placed beside the chart, showing both the industry name and the exact percentage share of total jobs.

Job distribution by country:

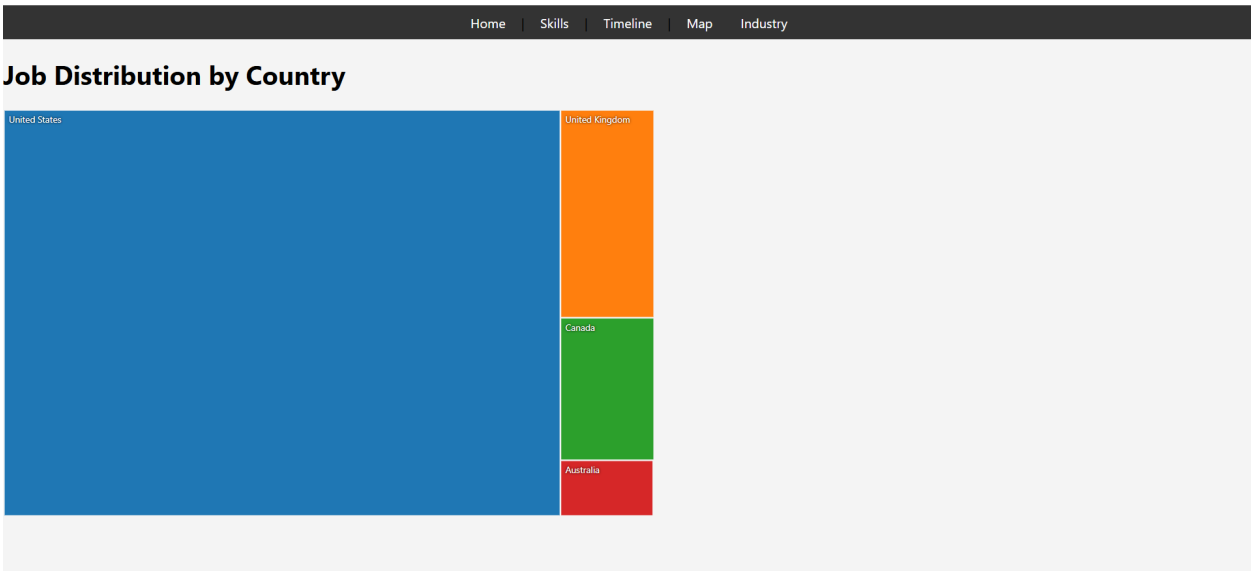
Initial Attempt: Bar Chart

The first visualization attempt was a bar chart showing the number of jobs per country. This layout allowed a quick comparison of job counts but lacked spatial context. Without a map-based approach, users couldn't easily interpret where the demand was located geographically. While useful for raw counts, it didn't intuitively communicate patterns of job distribution.



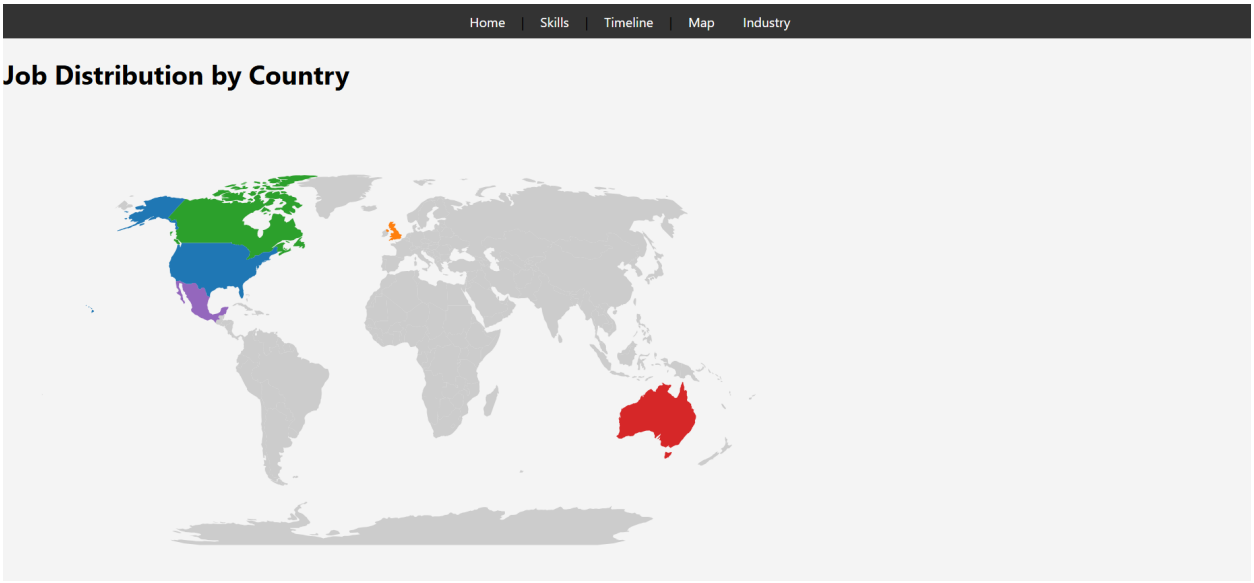
Second Attempt: Treemap

To improve the use of space and emphasize the relative proportions of job listings by country, a treemap was implemented. Although visually compact, this layout proved difficult to interpret geographically. It became visually overwhelming when more countries were included, and users struggled to associate rectangle sizes with real-world locations.



Final Visualization: World Map

The current and most effective design is a world map rendered using D3.js. Countries are colored based on job counts using a categorical color scale. Hovering over a country displays the exact number of jobs available. This version offers both intuitive spatial context and interactivity, making it easier to spot regional patterns and disparities in job availability.



Key considerations during this design phase included:

- Ensuring country name normalization (e.g., USA vs. United States)
- Avoiding color collisions across similar value categories

- Balancing interactivity without cluttering the view

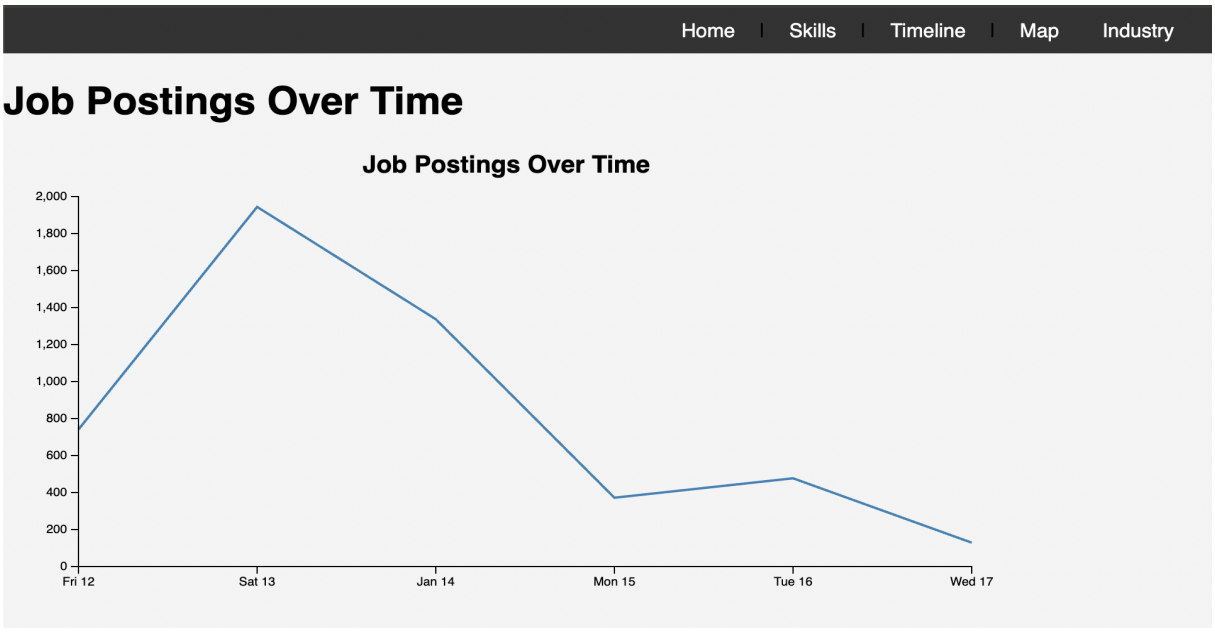
Further improvements will include filtering capabilities and better legends.

Job Postings over Time:

We wanted to visualize how job postings fluctuate over time to identify patterns such as spikes in hiring or dormant periods. This would help answer questions about industry behavior and temporal trends.

- Initially, we attempted a grouped bar chart by week, but this didn't provide smooth insight into short-term fluctuations.
- We transitioned to a line chart which allowed better visual tracking of ups and downs in job posting volumes. This design clearly shows trends and potential cycles in hiring. Time (x-axis) vs. job count (y-axis). A line chart was used to visually connect sequential data points, emphasizing changes.
- We preprocessed the data to aggregate daily job counts from timestamped entries, ensuring a clean and analyzable structure.

The dashboard highlights a noticeable peak around a specific date and a gradual drop-off after that. It helps stakeholders infer recruitment surges or dips.



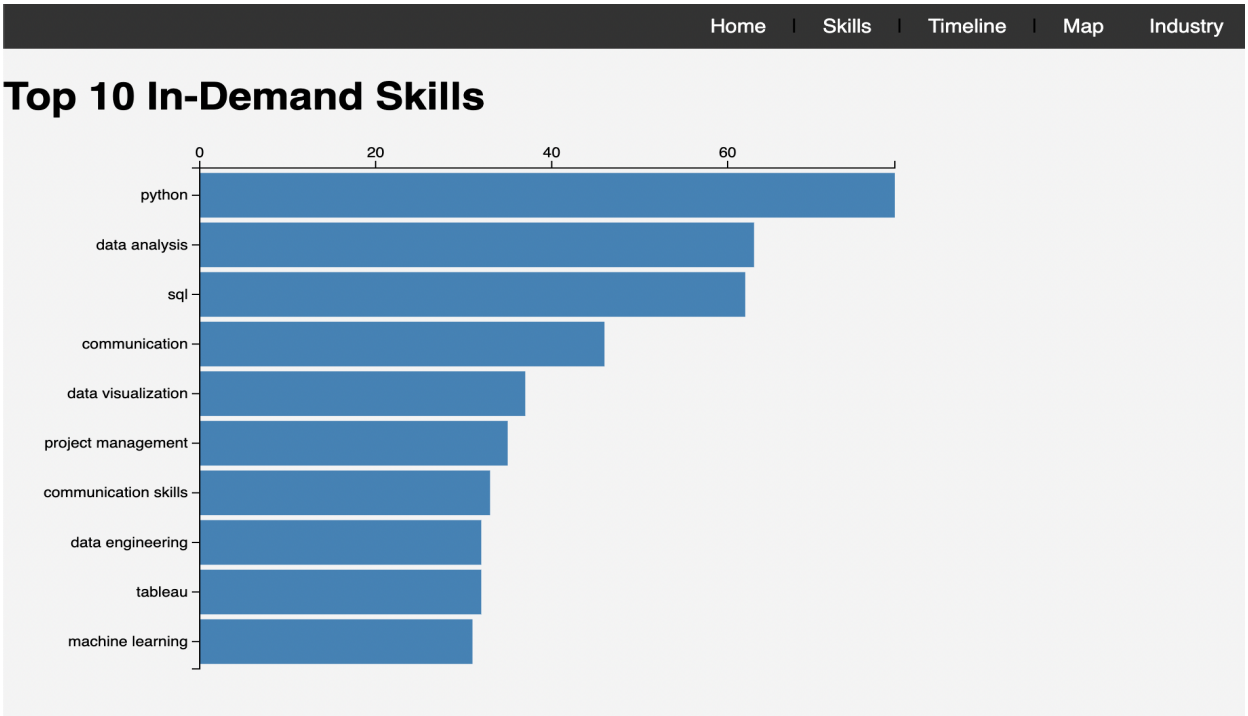
Due to the dataset being limited to a specific time period, seasonal or long-term trends couldn't be captured. In the future, expanding the data to cover more weeks or months could provide a more comprehensive view.

Skills Dashboard:

We initially considered a simple frequency count of technical skills mentioned across job postings. The idea was to highlight what skills are most in demand using a straightforward bar chart to ensure interpretability and quick comparisons.

A horizontal bar chart was implemented to represent the top 10 in-demand skills. We chose this format over alternatives like pie charts because bar charts are better at showing fine-grained comparisons in quantity. This format aligns with the principle of expressiveness, as it makes exact value comparison much easier for the user.

The dashboard reveals that **Python, Data Analysis, and SQL** are among the most in-demand skills across data science-related roles. The visualization helps stakeholders quickly identify which skills to prioritize for learning or hiring.



Although effective, this chart only shows frequency, not skill co-occurrence or trends over time. In the future, we could explore a skill network or timeline-based skill trend chart.

Justifying Our Design Decisions

Visualization Evolution Based on Data Structure: Each dashboard's design evolved through iteration, influenced by what we learned from the data:

- The industry dashboard began as a bar chart but quickly proved visually inefficient for comparing a small number of broad categories. We transitioned to a pie chart, and

eventually a donut chart, which preserved proportional comparisons while allowing a dedicated legend with percentage labels. This made the visualization both space-efficient and informative.
- The job timeline chart started with weekly aggregation but, due to the limited date range, we shifted to daily granularity. A line chart was chosen for its ability to clearly show fluctuations and trends in short time spans.

Encoding Decisions Driven by Data Type: Each chart type was chosen to match the type of data it represents:

- **Line chart** for temporal data — allows users to see patterns and rates of change over time.
- **Bar chart** for ranked categorical data — makes it easy to compare individual values (skills).
- **Donut chart** for industry breakdown — simplifies category comparison while minimizing clutter.
- **Choropleth map** for geographic distribution — visually intuitive for showing country-wise job volume.

This alignment between data type and chart design ensured that viewers could interpret each dashboard with minimal effort.

Focus on Readability and Clean Layout: We avoided visual overload by placing emphasis on whitespace, spacing, and color contrast. We ensured that:

- Labels are placed where they won't overlap with chart elements.
- Side legends are used where label space is limited (e.g., donut chart).
- Color schemes are perceptually distinct (e.g., interpolateBlues for map, schemeCategory10 for donut chart).

All charts are centered and scaled to fit within a balanced layout, which also prepares the design for future integration into a unified dashboard view.

Tools and Frameworks

- **D3.js**: Core library used for creating the map and handling interactivity
- **TopoJSON**: Used to load and render country boundaries

- **HTML/CSS**: Structuring and styling the page
- **JavaScript**: Core logic for data parsing and visualization
- **Git & GitHub**: Version control and collaboration

Potential future tools:

- **Plotly.js** or **Chart.js** for additional chart types
- **CSV parser libraries** for more robust preprocessing

Difficulties Encountered

Developing the visualization came with a number of challenges, particularly around data normalization and rendering accuracy:

- **Data Preprocessing:** We initially considered loading the full cleaned dataset directly into D3 and performing filtering and grouping in-browser. However, this resulted in slower load times and more complex frontend logic. By preprocessing the data server-side using Python, we achieved:
    - Faster load times and leaner JS code
    - Easier debugging and better control over data formats
    - Reusability of aggregation logic for multiple views

  This decision was especially helpful in syncing visual design across dashboards and maintaining a consistent structure throughout the project.

- **Inconsistent Country Names**: One of the first difficulties was resolving mismatched country names between the CSV data (e.g., "USA", "England") and the GeoJSON map file (e.g., "United States of America", "United Kingdom"). This led to some countries incorrectly showing 0 jobs or not highlighting at all. The solution involved writing a normalization block to alias all variants to standardized country names.

- **Incorrect Tooltip Counts**: Initially, some tooltips displayed 0 jobs even when countries had job postings. This was caused by a mismatch between the normalized names in our dataset and the raw names coming from the GeoJSON d.properties.name. We had to consistently apply normalization logic in both the fill and mouseover sections of the code.

- **Data Granularity and US Bias**: A significant portion of the data was US-centric, which resulted in an imbalanced map. To maintain visual balance, we considered downsampling or visually softening US regions, but ultimately chose to present the data transparently while noting this limitation.

- **Location Parsing Complexity**: The location field had inconsistent formats. Some rows had only city and state, others had city, state, and country. We had to write logic to handle these variations and default to "United States" when the country part was missing.

- **Color Scaling Limitations**: Using a categorical color scale with many countries caused repeated or very similar colors for different regions. While this was acceptable for the milestone, it will be improved using a continuous scale or legend-based gradient in the final version.

- **Initial GeoJSON Load Errors**: At first, the world.geojson file failed to load due to file extension mismatches (.txt instead of .geojson) and server path issues. This required rechecking MIME types and properly placing the file in the data/ folder.

- **Choosing Aggregation Granularity :** We had to decide **how much to aggregate**:
    - Weekly vs. daily trends for the timeline chart
    - Grouping by exact location or broader regions (e.g., city vs country)
    - Skill frequency with vs without experience-level filters

- **Folder Structure & Relative Paths** As the folder structure deepened (e.g., data/, js/, styles/), handling **relative paths** correctly became tricky. Some charts wouldn't load because JSON or CSV files weren't being found due to incorrect relative paths. It took several iterations and local server testing to consistently resolve assets across pages.

- **Debugging Blank Dashboards** In the early stages, some dashboards loaded without errors in the console, yet nothing appeared on screen. This was often due to:
    - SVG container dimensions being too small or misaligned
    - Improper data parsing (e.g., counts being treated as strings)
    - Charts attempting to render before the data finished loading

  These issues were resolved over time through debugging, logging, and gradually isolating problematic code.

Planned Schedule and Milestone

Tasks completed by milestone:

- Data cleaning and reduction
- Map visualization using D3.js
- Country normalization for consistent tooltips
- Initial GitHub repo and commits
