Project Write-Up: Graph Analysis and Visualization

Table of Contents

Overview

This project leverages graph theory and visualization techniques to analyze a dataset of companies. The goal is to examine centrality measures, market cap distributions, global representation, and outliers. Two distinct visualizations—centrality with and without labels—highlight insights, particularly focusing on United States companies.

The code is implemented in Rust, utilizing the plotters library for visualizations and a modular design for clarity and scalability. Below, we delve into the methods used, their implementation, and the outcomes observed.

Dataset Description

The dataset contains:

- Market Capitalization: Represents the size of each company.
- Stock Prices: Numerical values indicating the trading price of each company's stock.
- Countries: Locations where the companies are based.

The data is processed from a CSV file, and relationships between companies are built based on shared attributes, such as countries.

Methodology

Data Loading and Processing

The project begins by reading the dataset using csv::Reader. Three vectors are initialized:

- market cap: Stores the market capitalization of each company.
- prices: Stores stock prices.
- countries: Stores the country of each company.

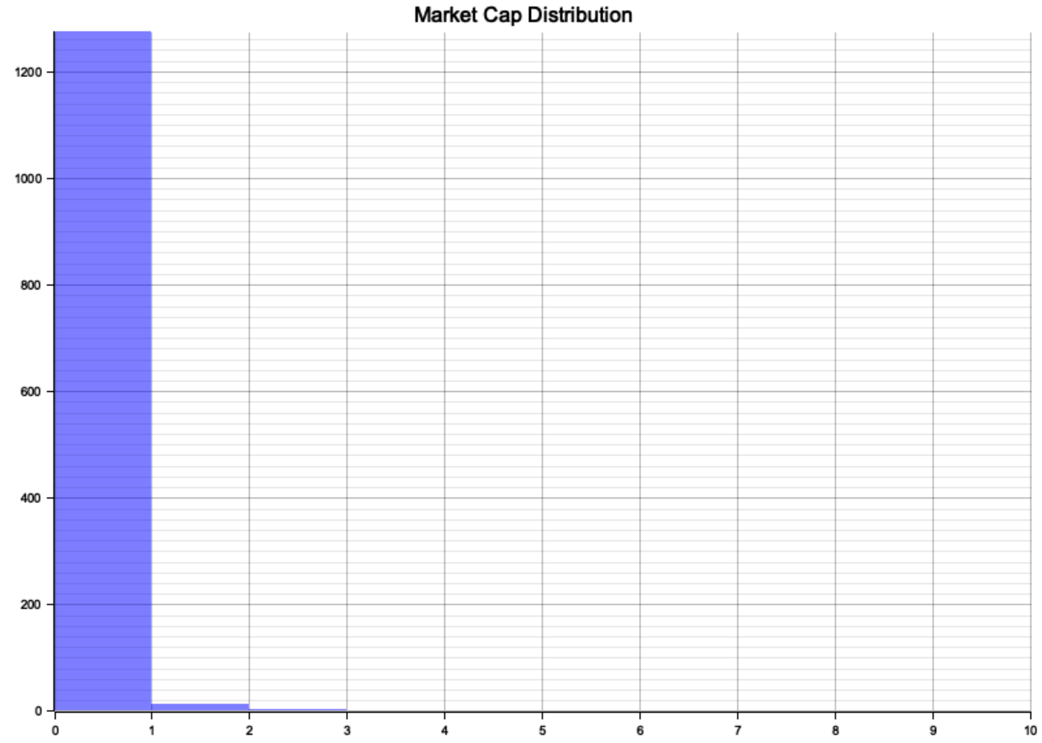
```
let mut marketcap = Vec::new();
let mut prices = Vec::new();
let mut countries = Vec::new();

for record in reader.records() {
    let record = record?;
    if let (Ok(mc), Ok(price), Ok(country)) = (
        record[3].parse::<f64>(),
        record[4].parse::<f64>(),
        record[5].parse::<String>(),
    ) {
        marketcap.push(mc);
        prices.push(price);
        countries.push(country);
    }
}
```

Market Cap Distribution

The dataset's market cap is divided into bins to observe its distribution.

- A histogram is generated using plotters, showing the frequency of market cap values.
- The distribution reveals a strong clustering of companies with smaller market caps.

Visualization: Market Cap Distribution

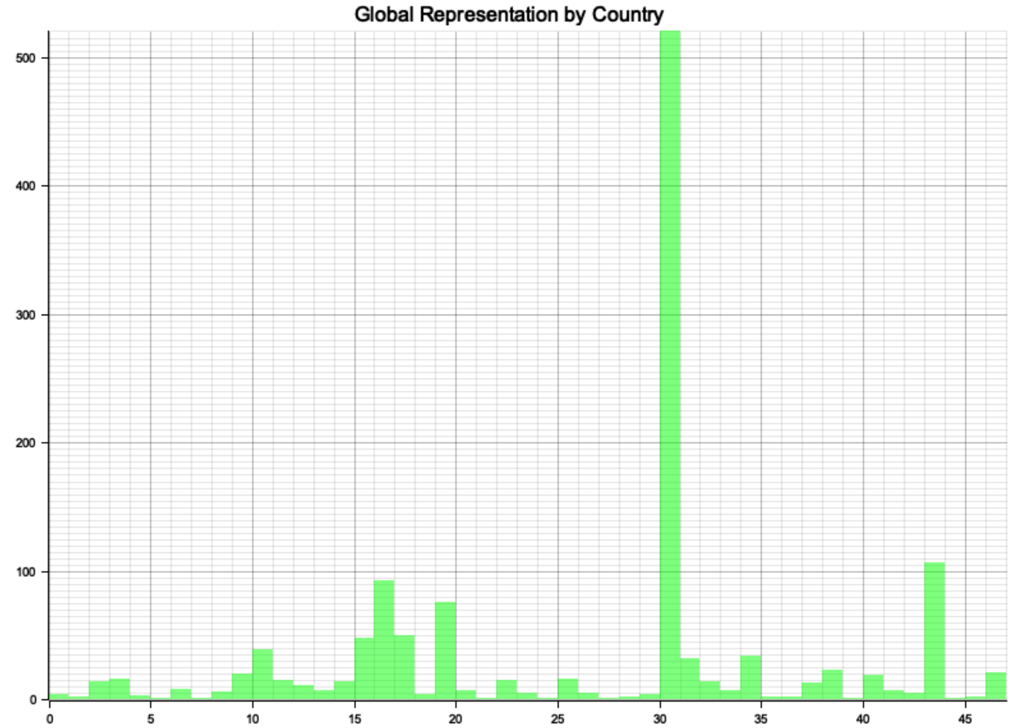


Global Representation

Using a HashMap, the number of companies per country is computed. A bar chart is generated to visualize the global representation, highlighting the dominance of certain countries (e.g., the United States).

```
let mut country_map = HashMap::new();
for country in &countries {
    country_map.entry(country.clone()).or_default() += 1;
}
```

Visualization: Global Representation by Country



Outlier Detection

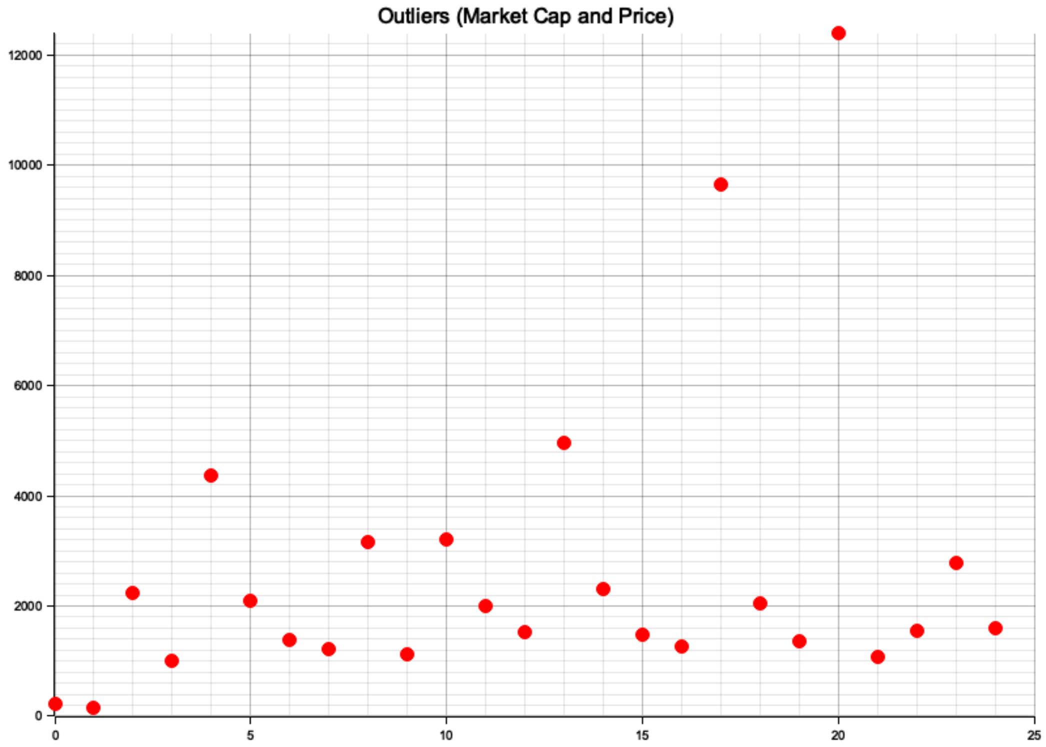
Outliers are identified based on high market caps or stock prices. The detection criteria:

- Market cap exceeding the 90th percentile.
- Stock price exceeding a threshold (e.g., $1000).

Outliers are visualized as red points, representing anomalies in the dataset.

```
for (i, &cap) in marketcap.iter().enumerate() {
    if cap > cap_threshold || prices[i] > 1000.0 {
        outliers.push((cap, prices[i], countries[i].clone()));
    }
}
```

Visualization: Outliers (Market Cap and Price)



Graph Construction

A graph is constructed using the Graph struct. Nodes represent companies, and edges are created between nodes if companies share the same country.

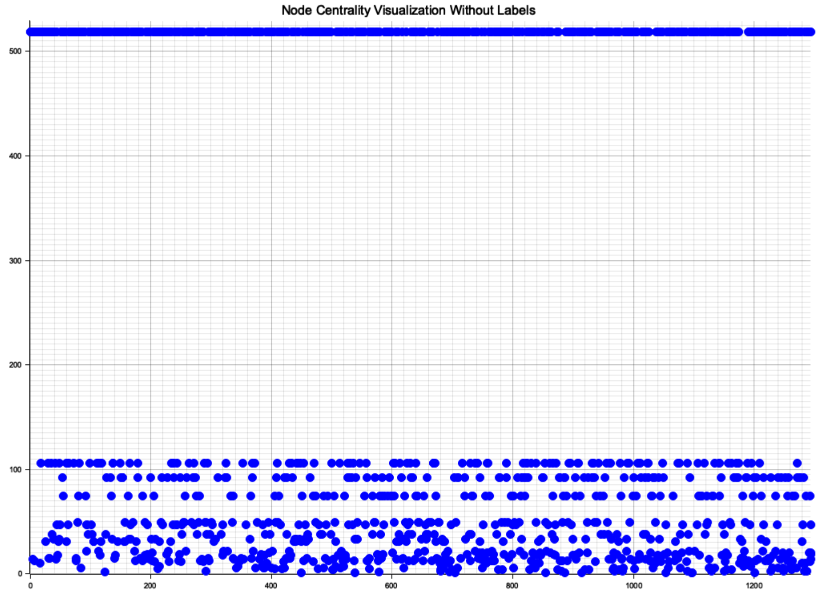
```
let mut graph = Graph::new();
for (i, country) in countries.iter().enumerate() {
    for (j, other_country) in countries.iter().enumerate() {
        if i != j && country == other_country {
            graph.add_edge(i, j);
        }
    }
}
```

Degree Centrality

The centrality of each node is calculated by counting its neighbors. This provides insights into the influence or connectedness of nodes within the graph.

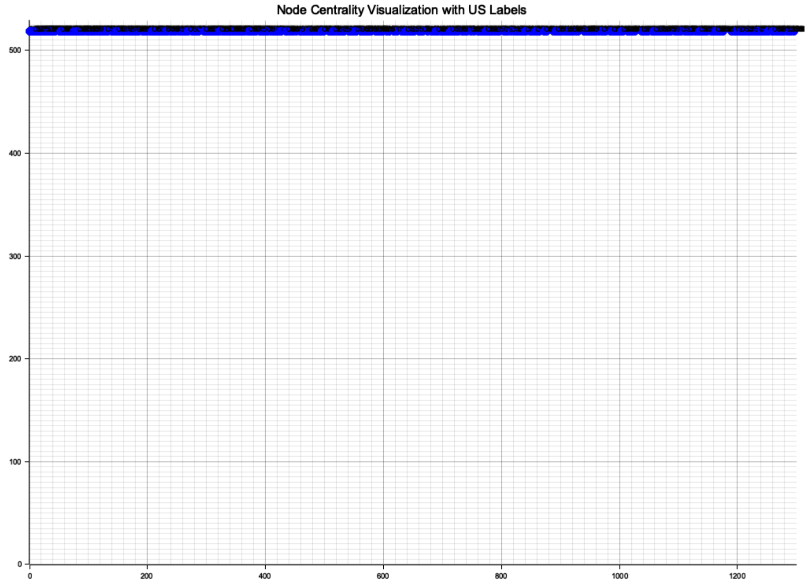
```
pub fn compute_degrees(&self) -> HashMap<usize, usize> {
    let mut degrees = HashMap::new();
    for (node, neighbors) in &self.adjacency_list {
        degrees.insert(node, neighbors.len());
    }
    degrees
}
```

Visualization: Centrality Without Labels



This visualization plots the centrality scores of all nodes without distinguishing labels. It highlights the overall structure of the graph, showing many low-centrality nodes and a few high-centrality outliers.

Visualization: Centrality with US Labels



This visualization focuses on nodes representing U.S. companies. Labels are added for nodes with high centrality (e.g., greater than 30).

```
for (&node, &score) in centrality {
    if let Some(label) = labels.get(&node) {
        if label == "United States" {
            chart.draw_series(std::iter::once(Circle::new((node, score), 5, BLUE.filled())))?;
            chart.draw_series(std::iter::once(
                Text::new(
                    format!("US: {}", score),
                    (node, score + 5),
                    ("sans-serif", 12).into_font(),
                ),
            ))?;
        }
    }
}
```

Outcomes and Insights

Market Cap Distribution

Insight: The majority of companies have smaller market caps, with a steep decline in frequency for larger caps. This reflects the reality that most firms are small to medium-sized, with a few major players dominating.

Global Representation

Insight: The bar chart highlights the dominance of companies based in the United States, which far surpasses representation from other countries. This emphasizes the economic influence of the U.S.

Outliers

Insight: The outlier chart pinpoints companies with extremely high market caps or stock prices. These outliers likely correspond to major corporations with global impact, such as tech giants or industry leaders.

Graph Centrality

Insight: Centrality analysis reveals that most nodes (companies) have low connectivity, representing limited shared attributes. However, U.S. companies stand out with higher centrality, indicating strong interconnectedness and influence within the dataset.
US-Focused Centrality: Nodes labeled as "United States" with significant centrality demonstrate the central role of the U.S.-based companies in the datasets network structure.

Conclusion

This project demonstrates how graph theory and data visualization can uncover patterns and anomalies in a dataset. By combining centrality measures with focused visualizations, the analysis highlights the structural relationships and dominance of specific entities within the dataset. The modular codebase ensures extensibility, allowing further exploration of other graph metrics or datasets.