October 15, 2025

Earthquake Data Analysis & Visualization

Project Report

Your Name: P.Poorna Sai Sriharsha
Student ID: Y23CS137

Your Name P.Gopi Krishna
Student ID Y23CS136

---

1 Content of the Project

This report presents a comprehensive exploratory data analysis (EDA) of a global earthquake dataset spanning from 1965 to 2016. The primary objective is to uncover patterns, trends, and fundamental principles of seismology through data visualization and statistical analysis. The analysis investigates several key areas: the temporal frequency of recorded earthquakes, the distribution of seismic energy release, the relationship between earthquake magnitude and depth, and the validation of the Gutenberg-Richter law. Finally, a web-based dashboard is created using Python's Flask framework to present all visualizations in a single, interactive interface.

---

2 Formulas Used

2.1 Seismic Energy Release Formula

The relationship between magnitude (M) and the relative energy (E) it releases is approximated by the formula:

$$E \propto 10^{1.5 \times M}$$

Significance: This formula highlights that a small increase in magnitude corresponds to a massive increase in released energy. We use this to calculate the cumulative energy released over time.

---

3 Dataset

- **Source:** The project utilizes the `earthquakes.csv` dataset, a curated collection of significant global earthquake events, typically aggregated from sources like the USGS.
- **Structure:** The dataset contains columns for Date, Latitude/Longitude, Type, Depth, Magnitude, and Source.
- **Preparation:** The raw data was preprocessed to convert the 'Date' column to a datetime format. Additional columns such as 'Year', 'Decade', and 'Depth Category' were engineered to facilitate the analysis.

4 Analysis and Code Explanation

4.1 Analysis 1: Earthquake Frequency by Decade

Goal: To observe if the number of recorded earthquakes per decade is increasing, decreasing, or remaining constant over time.

Method: We count the total number of earthquakes for each decade (1960s,1970s, etc.) and plot these counts on a line graph.

```
dc=df['Decade'].value_counts().sort_index()
plt.figure(figsize=(12,6))
sns.lineplot(x=dc.index,y=dc.values,marker='o',color='royalblue')
plt.title('Earthquake Frequency by Decade (1965-2016)',fontsize=16)
plt.xlabel('Decade')
plt.ylabel('Number of Earthquakes')
plt.grid(True)
plt.xticks(dc.index)
plt.show()
```

Code Snippet 1: Counting and plotting earthquakes by decade.

Insight: The graph shows a significant increase in recorded earthquakes, likely reflecting advancements in detection technology rather than an increase in actual seismic activity.

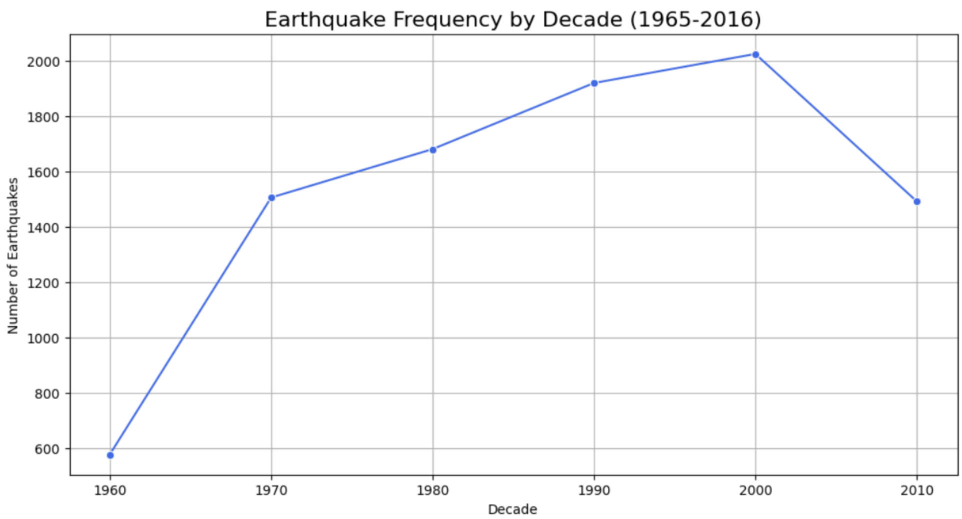

Figure 1: Trend of recorded earthquakes from the 1960s to the 2010s.

4.2 Analysis 2: Cumulative Seismic Energy Release

Goal: To track the total seismic energy released over time and identify which specific events were the most significant contributors.

Method: We convert earthquake magnitude to a relative energy value using the formula $E \propto 10^{(1.5M)}$. Then, we calculate a running total ('cumsum') of this energy over the years and plot it.

```
df['Energy']= 10**(1.5 * df['Magnitude'])
df.sort_values('Date', inplace=True)
df['Cumulative Energy']= df['Energy'].cumsum()
top_energy_events= df.sort_values(by='Energy', ascending=False).head
    (4)
plt.figure(figsize=(12, 7))
plt.plot(df['Date'], df['Cumulative Energy'], color='red')
plt.title('Cumulative Seismic Energy Release (1965-2016)', fontsize
    =16)
plt.xlabel('Year')
plt.ylabel('Cumulative Energy Released (Relative Scale)')
plt.grid(True)
for idx, row in top_energy_events.iterrows():
    plt.axvline(row['Date'], color='gray', linestyle='--', lw=0.5)
    plt.text(row['Date'], row['Cumulative Energy'], f" {row['Date'].
        year}\nM {row['Magnitude']} ", va='top')
plt.show()
```

Code Snippet 2: Calculating and plotting cumulative energy.

Insight: The plot is characterized by sharp, nearly vertical jumps, powerfully illustrating that a very small number of high-magnitude earthquakes are responsible for almost all the seismic energy released.

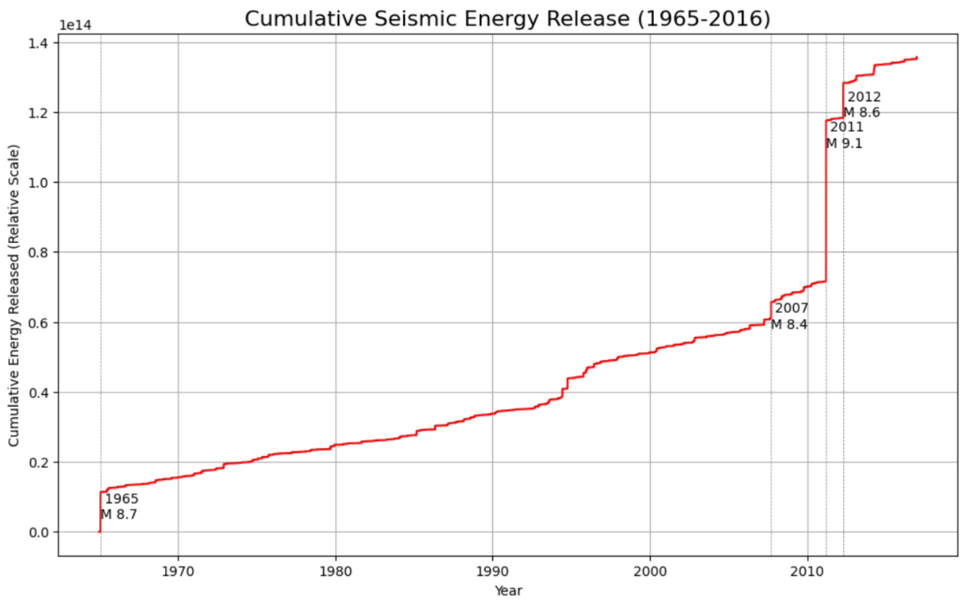

Figure 2: Cumulative energy plot showing the dominance of mega-quakes.

4.3 Analysis 3: Magnitude Distribution by Depth Category

Goal: To investigate the relationship between an earthquake's depth and its magnitude.
Method: We define a function to categorize each earthquake as 'Shallow', 'Intermediate', or 'Deep'. Then we use a boxplot to compare the range of magnitudes for each of these three categories.

```python
def assign_depth_category(depth):
    if depth < 70: return 'Shallow'
    elif 70 <= depth <= 300: return 'Intermediate'
    else: return 'Deep'
df['Depth Category'] = df['Depth'].apply(assign_depth_category)
plt.figure(figsize=(10, 6))
sns.boxplot(x='Depth Category', y='Magnitude', data=df, order=['
    Shallow', 'Intermediate', 'Deep'])
plt.title('Magnitude Distribution by Depth Category', fontsize=16)
plt.xlabel('Depth Category')
plt.ylabel('Magnitude')
plt.grid(True, axis='y')
plt.show()
```

Code Snippet 3: Categorizing depth and creating a boxplot.

Insight: The plot reveals that the most powerful events (those with the highest magnitudes) tend to be shallow-focus earthquakes. Deep earthquakes are common but generally have lower maximum magnitudes.

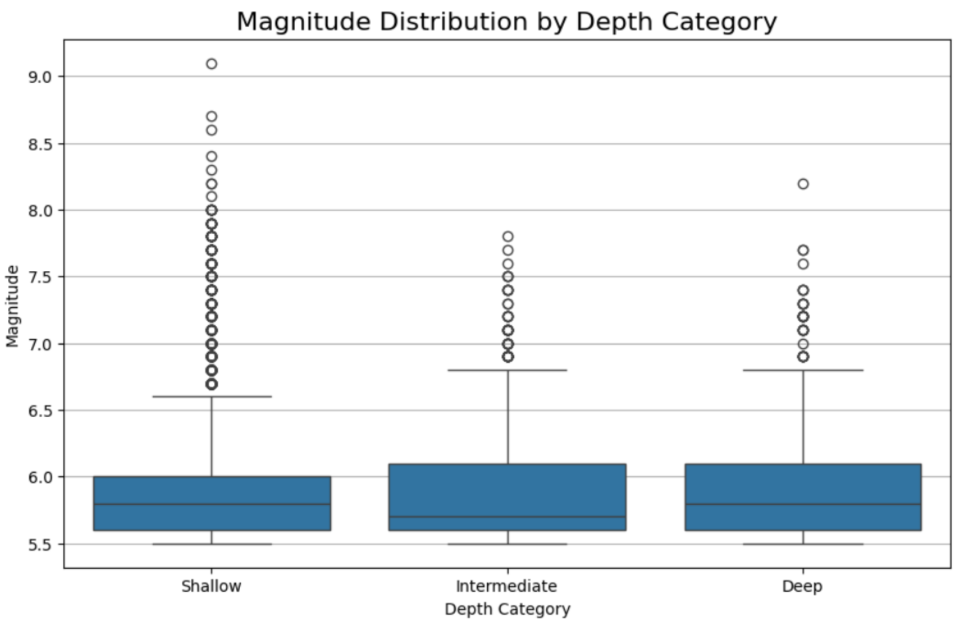

Figure 3: Comparison of magnitude ranges for different depth categories.

4.4 Analysis 4: Test of Gutenberg-Richter Seismology Law

Goal: To visually verify the Gutenberg-Richter law, which states that there are many more small earthquakes than large ones.

Method: We count how many earthquakes occurred at each magnitude level. We then plot this on a graph where the vertical axis (y-axis) is set to a logarithmic scale.

```
mag_counts = df['Magnitude'].round(1).value_counts().sort_index()
plt.figure(figsize=(10, 6))
plt.plot(mag_counts.index, mag_counts.values, 'o-', color='crimson')
plt.title('Gutenberg-Richter Law: Frequency vs. Magnitude', fontsize
    =16)
plt.xlabel('Magnitude')
plt.ylabel('Number of Earthquakes (Log Scale)')
plt.yscale('log')
plt.grid(True, which="both", ls="--")
plt.show()
```

Code Snippet 4: Plotting earthquake frequency vs. magnitude on a log scale.

Insight: The resulting plot shows a nearly straight, downward-sloping line, a classic visual confirmation of the Gutenberg-Richter law, where frequency decreases exponentially as magnitude increases.

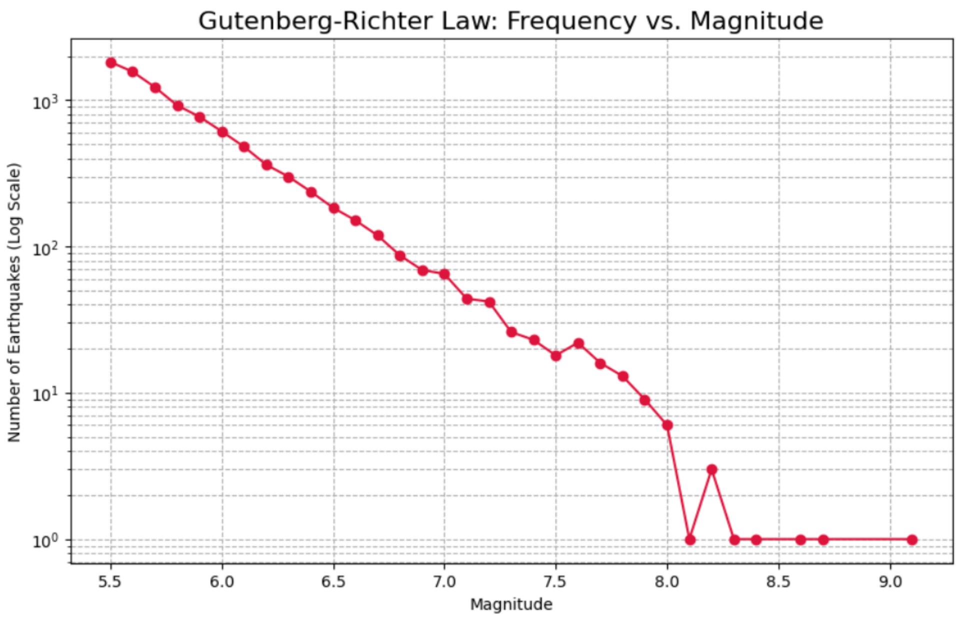

Figure 4: Log-linear plot confirming the Gutenberg-Richter law.

4.5 Analysis 5: Case Study of the 2011 Japan Aftershock Sequence

Goal: To visualize the aftershock sequence following a major earthquake, illustrating how seismic activity decays over time.

Method: We select a famous major event: the Magnitude 9.1 Tōhoku, Japan earthquake of 2011. We then filter the entire dataset to isolate all smaller earthquakes that occurred in the same geographic region within the 30 days *after* the mainshock. We then plot the magnitude of these aftershocks over time.

```
mainshock_date = pd.to_datetime('2011-03-11')
mainshock_lat = 38.322 mainshock_lon = 142.369
start_date = mainshock_date
end_date = start_date + pd.Timedelta(days=30)
lat_range=[mainshock_lat - 5, mainshock_lat + 5]
lon_range = [mainshock_lon - 5, mainshock_lon + 5]
aftershocks = df[(df['Date'] >= start_date) &(df['Date'] < end_date)
    &(df['Latitude'] >= lat_range[0]) & (df['Latitude'] <= lat_range
    [1])&(df['Longitude'] >= lon_range[0])&(df['Longitude'] <=
    lon_range[1])&(df['Magnitude'] < 9.0)].copy()
plt.figure(figsize=(15, 7))
plt.scatter(aftershocks['Date'],aftershocks['Magnitude'],alpha=0.5,
    color='royalblue',label='Aftershock')
plt.axvline(x=mainshock_date,color='red',linestyle='--',linewidth=2,
    label='Mainshock (M 9.1)')
plt.ylim(4.5, 8.0)
plt.legend()
plt.show()
```

Code Snippet 5: Filtering and plotting aftershocks for the 2011 Tohoku event.

Insight: The plot vividly shows an intense "storm" of aftershocks immediately following the main event. Both the frequency and magnitude of these aftershocks clearly decrease over the 30-day period, visualizing the principle of aftershock decay.

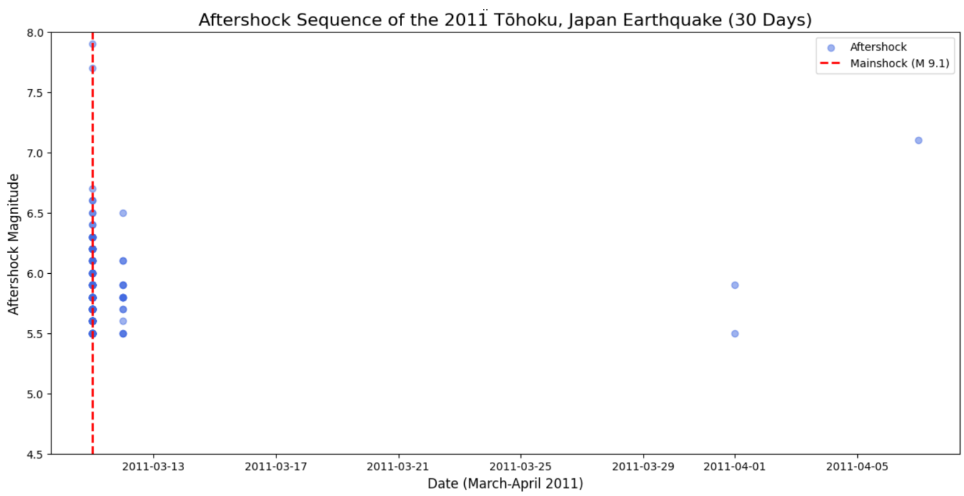

Figure 5: Magnitude of aftershocks over a 30-day period post-mainshock.

4.6 Analysis 6: Magnitude vs. Depth Bubble Chart

Goal: To find the "hotspots" where the most common combinations of magnitude and depth occur, in a simple and clear way.

Method: We use a 'Bubble Chart'. First, we group both Depth and Magnitude into simple categories (e.g., Depth: 'Shallow', 'Intermediate', 'Deep'; Magnitude: '5.5-6.0', '6.0-6.5', etc.).

```
magnitude_bins = [5.5, 6.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.5]
magnitude_labels=[f'{magnitude_bins[i]}-{magnitude_bins[i+1]}'for i in range(len(magnitude_bins)-1)]
df['Magnitude Bin']=pd.cut(df['Magnitude'], bins=magnitude_bins, labels=magnitude_labels, right=False)
bubble_data = df.groupby(['Depth Category', 'MagnitudeBin']).size().reset_index(name='Count')
bubble_data=bubble_data[bubble_data['Count'] > 0] plt.figure(figsize=(14, 8))
bubble_plot = sns.scatterplot(data=bubble_data,x='Magnitude Bin',y='Depth Category',size='Count',hue='Count',palette='plasma',sizes=(50, 2000),legend='auto')
plt.gca().invert_yaxis() plt.xticks(rotation=45) plt.grid(True, linestyle='--', alpha=0.5)
h,l=bubble_plot.get_legend_handles_labels()
plt.legend(h[0:7],l[0:7],title="Number of Earthquakes",bbox_to_anchor=(1.05, 1),loc=2,borderaxespad=0.)
plt.tight_layout() plt.show()
```

Code Snippet 6: Creating a bubble chart to show frequency hotspots.

Insight: The chart's largest bubble corresponds to shallow, 5.5-6.0 magnitude events, clearly indicating this is the most frequently recorded type of significant earthquake.

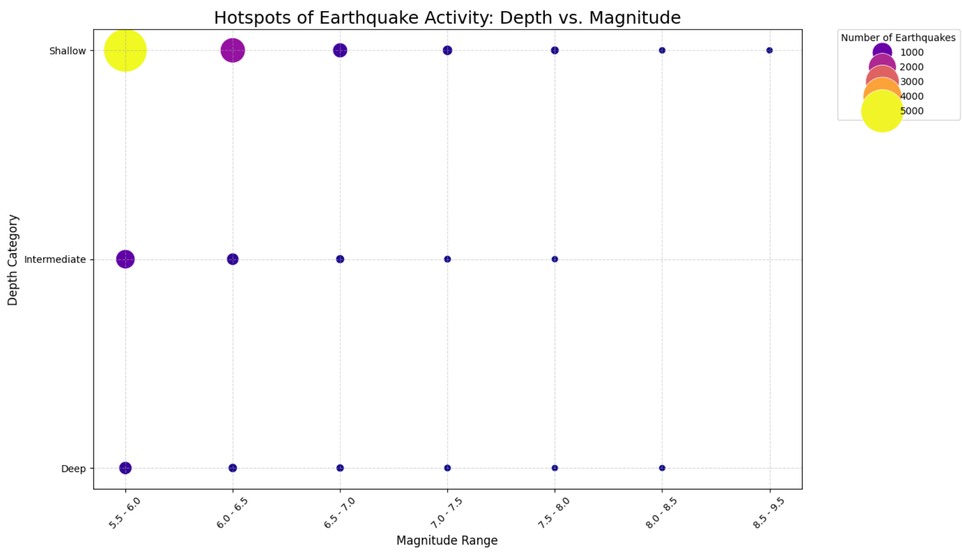

Figure 6: Bubble chart showing the prevalence of shallow, moderate-magnitude events.

4.7 Analysis 7: Magnitude by Reporting Agency

Goal: To compare magnitude measurements reported by different international agencies for consistency.

Method: We use a violin plot. The shape of the "violin" shows the distribution of magnitudes; where the violin is widest is where most of that agency's measurements lie.

```
top_sources = df['Source'].value_counts().head(5).index
df_top_sources = df[df['Source'].isin(top_sources)]
plt.figure(figsize=(12, 7))
sns.violinplot(x='Source', y='Magnitude', data=df_top_sources,
    palette='plasma')
plt.title('Magnitude Distribution for Top 5 Reporting Agencies',
    fontsize=16)
plt.xlabel('Reporting Agency (Source)')
plt.ylabel('Magnitude')
plt.show()
```

Code Snippet 7: Comparing magnitude distributions with a violin plot.

Insight: The similar shapes of the violins show a high degree of consistency in magnitude reporting among major global agencies, giving confidence in the data's reliability.

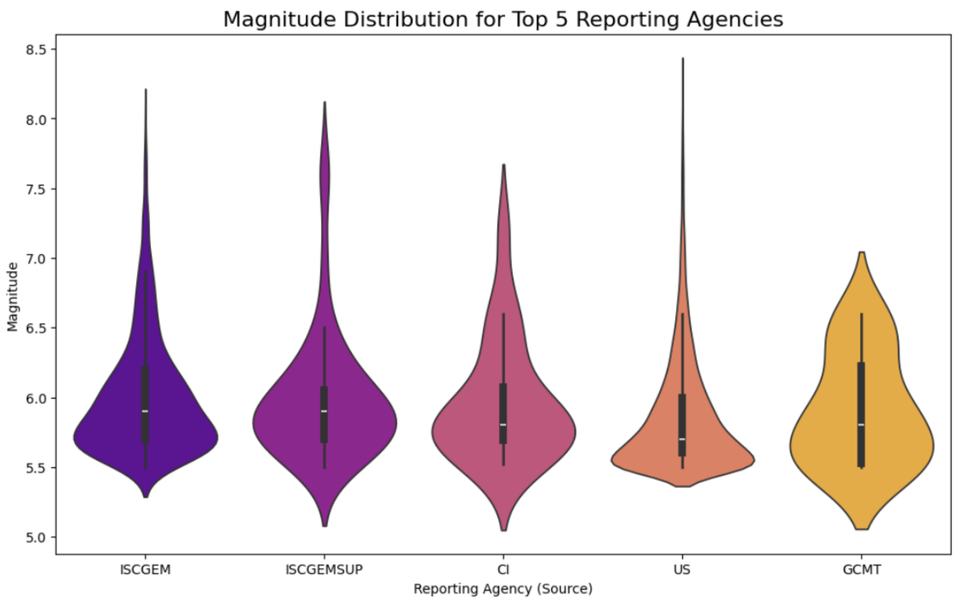

Figure 7: Violin plot showing consistent magnitude reporting among top agencies.

4.8 Analysis 8: Mapping Nuclear Explosions

Goal: To identify and map all events classified as nuclear explosions to reveal their geographical distribution.

Method: We filter the data to select only "Nuclear Explosion" events. Then, we use Folium to plot these locations on a dark-themed map, making more powerful explosions appear as larger circles.

```python
nuclear_df = df[df['Type'] == 'Nuclear Explosion'].copy()
nuclear_map = folium.Map(location=[49.8, 78.5], zoom_start=3, tiles="CartoDB dark_matter")
for idx, row in nuclear_df.iterrows():
    popup_text = f"<b>Event Type:</b> {row['Type']}<br><b>Date:</b> {row['Date'].strftime('%Y-%m-%d')}<br><b>Magnitude:</b> {row['Magnitude']}"
    folium.CircleMarker(
        location=[row['Latitude'], row['Longitude']], radius=row['Magnitude'] * 1.5, color='#ff3300',
        fill=True, fill_opacity=0.6, popup=folium.Popup(popup_text, max_width=300)
    ).add_to(nuclear_map)
nuclear_map
```

Code Snippet 8: Filtering and mapping nuclear explosion events using Folium.

Insight: The map shows that man-made seismic events are highly clustered in locations corresponding to known historical nuclear test sites, demonstrating seismic data's utility in monitoring human activities.

Figure 8: Global distribution of recorded nuclear explosion events.

5 Earthquake Analysis Dashboard

To consolidate all visualizations into a single, accessible format, a web-based dashboard was developed using the Flask micro-framework for Python. This dashboard runs all the analysis scripts on a backend server and renders the resulting plots and maps on a clean, modern webpage.

5.1 How to Run the Dashboard

The following steps outline the process for setting up the environment and launching the web application.

Step 1: Prepare the Project Environment

1. **Create the File Structure:** Organize your files in a single project directory as shown below. The `app.py` script and the `earthquakes.csv` dataset must be in the root folder. The HTML file must be placed inside a subfolder named `templates`.

```
/your-dashboard-project/
|-- app.py
|-- earthquakes.csv
+-- /templates/
    |-- index.html
```

2. **Install Required Libraries:** Open a terminal or command prompt and install all the necessary Python packages using pip.

```
pip install pandas matplotlib seaborn folium Flask
```

Step 2: Launch the Application

1. **Navigate to the Directory:** In your terminal, change the current directory to your project folder (`cd path/to/your-dashboard-project`).
2. **Run the Flask App:** Execute the Python script to start the local web server (`python app.py`).

Step 3: View the Dashboard

After running the script, the terminal will indicate that a server is running (e.g., at `http://127.0.0.1:5000`). Open a web browser and navigate to this address to view the complete dashboard.

5.2 Backend Code (app.py)

```
import pandas as pd
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import seaborn as sns
import folium
from folium.plugins import HeatMap
from flask import Flask, render_template
import io
import base64
app = Flask(__name__)
def generate_plots():
    try:
        df = pd.read_csv('earthquakes.csv')
    except FileNotFoundError:
        return {"error": "The 'earthquakes.csv' file was not found. Please make sure it's in the same directory as app.py."}
    # --- Data Preparation ---
    df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
    df = df.dropna(subset=['Date'])
    df['Year'] = df['Date'].dt.year
    df['Decade'] = (df['Year'] // 10) * 10
    df['Hour'] = df['Date'].dt.hour
    plots = {}
    # --- Plot 1: Earthquake Frequency by Decade ---
    plt.figure(figsize=(12, 6))
    dc = df['Decade'].value_counts().sort_index()
    sns.lineplot(x=dc.index, y=dc.values, marker='o', color='royalblue')
    plt.title('Earthquake Frequency by Decade (1965-2016)', fontsize=16)
    plt.xlabel('Decade')
    plt.ylabel('Number of Earthquakes')
    plt.grid(True)
    plt.xticks(dc.index)
    plots['plot1'] = save_plot_to_base64()
    # --- Plot 2: Cumulative Seismic Energy Release ---
    df['Energy'] = 10**(1.5 * df['Magnitude'])
    df.sort_values('Date', inplace=True)
    df['Cumulative Energy'] = df['Energy'].cumsum()
    top_energy_events = df.sort_values(by='Energy', ascending=False).head(4)
    plt.figure(figsize=(12, 7))
    plt.plot(df['Date'], df['Cumulative Energy'], color='red')
    plt.title('Cumulative Seismic Energy Release (1965-2016)', fontsize=16)
    plt.xlabel('Year')
```

```python
    plt.ylabel('Cumulative Energy Released (Relative Scale)')
    plt.grid(True)
    for _, row in top_energy_events.iterrows():
        plt.axvline(row['Date'], color='gray', linestyle='--', lw=0.5)
        plt.text(row['Date'], row['Cumulative Energy'], f"{row['Date'].year}\nM {row['Magnitude']}", va='top')
    plots['plot2'] = save_plot_to_base64()
    # --- Plot 3: Magnitude by Depth Category ---
    def assign_depth_category(depth):
        if depth < 70: return 'Shallow'
        elif 70 <= depth <= 300: return 'Intermediate'
        else: return 'Deep'
    df['Depth Category'] = df['Depth'].apply(assign_depth_category)
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='Depth Category', y='Magnitude', data=df, order=['Shallow', 'Intermediate', 'Deep'], palette='viridis')
    plt.title('Magnitude Distribution by Depth Category', fontsize=16)
    plt.xlabel('Depth Category')
    plt.ylabel('Magnitude')
    plt.grid(True, axis='y')
    plots['plot3'] = save_plot_to_base64()
    # --- Plot 4: Gutenberg-Richter Law ---
    mag_counts = df['Magnitude'].round(1).value_counts().sort_index()
    plt.figure(figsize=(10, 6))
    plt.plot(mag_counts.index, mag_counts.values, 'o-', color='crimson')
    plt.title('Gutenberg-Richter Law: Frequency vs. Magnitude', fontsize=16)
    plt.xlabel('Magnitude')
    plt.ylabel('Number of Earthquakes (Log Scale)')
    plt.yscale('log')
    plt.grid(True, which="both", ls="--")
    plots['plot4'] = save_plot_to_base64()
    # --- Plot 5: Japan Aftershock Sequence ---
    mainshock_date = pd.to_datetime('2011-03-11')
    aftershocks = df[
        (df['Date'] >= mainshock_date) &
        (df['Date'] < mainshock_date + pd.Timedelta(days=30)) &
        (df['Latitude'] >= 33.322) & (df['Latitude'] <= 43.322) &
        (df['Longitude'] >= 137.369) & (df['Longitude'] <= 147.369) &
        (df['Magnitude'] < 9.0)
    ].copy()
    plt.figure(figsize=(15, 7))
    plt.scatter(aftershocks['Date'], aftershocks['Magnitude'], alpha=0.5, color='royalblue', label='Aftershock')
```

```python
    plt.axvline(x=mainshock_date, color='red', linestyle='--', linewidth=2, label='Mainshock (M 9.1)')
    plt.title('Aftershock Sequence of the 2011 T hoku, Japan Earthquake (30 Days)', fontsize=16)
    plt.xlabel('Date (March-April 2011)', fontsize=12)
    plt.ylabel('Aftershock Magnitude', fontsize=12)
    plt.ylim(4.5, 8.0)
    plt.legend()
    plots['plot5'] = save_plot_to_base64()
    # --- Plot 7: Magnitude vs. Depth Bubble Chart ---
    magnitude_bins = [5.5, 6.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.5]
    magnitude_labels = [f'{magnitude_bins[i]} - {magnitude_bins[i+1]}' for i in range(len(magnitude_bins)-1)]
    df['Magnitude Bin'] = pd.cut(df['Magnitude'], bins=magnitude_bins, labels=magnitude_labels, right=False)
    bubble_data = df.groupby(['Depth Category', 'Magnitude Bin']).size().reset_index(name='Count')
    bubble_data = bubble_data[bubble_data['Count'] > 0]
    plt.figure(figsize=(14, 8))
    bubble_plot = sns.scatterplot(
        data=bubble_data, x='Magnitude Bin', y='Depth Category', size='Count',
        hue='Count', palette='plasma', sizes=(50, 2000), legend='auto'
    )
    plt.title('Hotspots of Earthquake Activity: Depth vs. Magnitude', fontsize=18)
    plt.xlabel('Magnitude Range', fontsize=12)
    plt.ylabel('Depth Category', fontsize=12)
    plt.xticks(rotation=45)
    plt.grid(True, linestyle='--', alpha=0.5)
    h, l = bubble_plot.get_legend_handles_labels()
    plt.legend(h[0:7], l[0:7], title="Number of Earthquakes", bbox_to_anchor=(1.05, 1), loc=2)
    plt.tight_layout()
    plots['plot7'] = save_plot_to_base64()
    # --- Plot 8: Magnitude by Reporting Agency ---
    top_sources = df['Source'].value_counts().head(5).index
    df_top_sources = df[df['Source'].isin(top_sources)]
    plt.figure(figsize=(12, 7))
    sns.violinplot(x='Source', y='Magnitude', data=df_top_sources, palette='plasma')
    plt.title('Magnitude Distribution for Top 5 Reporting Agencies', fontsize=16)
    plt.xlabel('Reporting Agency (Source)')
    plt.ylabel('Magnitude')
    plots['plot8'] = save_plot_to_base64()
    nuclear_df = df[df['Type'] == 'Nuclear Explosion'].copy()
```

```python
    nuclear_map = folium.Map(location=[49.8, 78.5], zoom_start=3, tiles="CartoDB dark_matter")
    for _, row in nuclear_df.iterrows():
        popup_text = f"<b>Event Type:</b> {row['Type']}<br><b>Date:</b> {row['Date'].strftime('%Y-%m-%d')}<br><b>Magnitude:</b> {row['Magnitude']}"
        folium.CircleMarker(
            location=[row['Latitude'], row['Longitude']], radius=row['Magnitude'] * 1.5, color='#ff3300',
            fill=True, fill_opacity=0.6, popup=folium.Popup(popup_text, max_width=300)
        ).add_to(nuclear_map)
    plots['map10'] = nuclear_map._repr_html_()
    return plots
def save_plot_to_base64():
    """Saves the current matplotlib plot to a base64 encoded string."""
    img = io.BytesIO()
    plt.savefig(img, format='png', bbox_inches='tight')
    img.seek(0)
    plot_url = base64.b64encode(img.getvalue()).decode('utf8')
    plt.close() # Close the plot to free up memory
    return plot_url
@app.route('/')
def dashboard():
    """Renders the dashboard page with all the plots."""
    plot_data = generate_plots()
    if "error" in plot_data:
        # A simple error page if the CSV is missing
        return f"<p style='font-family: sans-serif; color: red;'>Error: {plot_data['error']}</p>"
    return render_template('index.html', plots=plot_data)
if __name__ == '__main__':
    app.run(debug=True)
```

Code Snippet 9: Flask application to generate plots and serve the dashboard.

5.3 Frontend Code (index.html)

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Earthquake Analysis Dashboard</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <style>
        .folium-map {
            width: 100%;
            height: 600px;
        }
        .folium-map iframe {
            width: 100%;
            height: 100%;
            border-radius: 0.5rem;
        }
    </style>
</head>
<body class="bg-gray-100 font-sans">
    <div class="container mx-auto p-4 md:p-8">
        <header class="text-center mb-10">
            <h1 class="text-4xl font-bold text-gray-800">Earthquake & Seismic Events Dashboard</h1>
            <p class="text-lg text-gray-600 mt-2">An analysis of global seismic data from 1965-2016</p>
        </header>
        <div class="grid grid-cols-1 lg:grid-cols-2 gap-8">
            <!-- Card for Plot 1 -->
            <div class="bg-white p-6 rounded-lg shadow-lg">
                <h2 class="text-xl font-semibold text-gray-700 mb-4">Plot 1: Earthquake Frequency by Decade</h2>
                <img src="data:image/png;base64,{{ plots.plot1 }}" alt="Earthquake Frequency by Decade" class="w-full h-auto rounded">
            </div>
            <!-- Card for Plot 2 -->
            <div class="bg-white p-6 rounded-lg shadow-lg">
                <h2 class="text-xl font-semibold text-gray-700 mb-4">Plot 2: Cumulative Seismic Energy Release</h2>
                <img src="data:image/png;base64,{{ plots.plot2 }}" alt="Cumulative Seismic Energy Release" class="w-full h-auto rounded">
            </div>
            <!-- Card for Plot 3 -->
            <div class="bg-white p-6 rounded-lg shadow-lg">
```

```html
                <h2 class="text-xl font-semibold text-gray-700 mb-4">Plot 3: Magnitude by Depth Category</h2>
                <img src="data:image/png;base64,{{ plots.plot3 }}" alt="Magnitude Distribution by Depth Category" class="w-full h-auto rounded">
            </div>
            <!-- Card for Plot 4 -->
            <div class="bg-white p-6 rounded-lg shadow-lg">
                <h2 class="text-xl font-semibold text-gray-700 mb-4">Plot 4: Gutenberg-Richter Law Analysis</h2>
                <img src="data:image/png;base64,{{ plots.plot4 }}" alt="Gutenberg-Richter Law" class="w-full h-auto rounded">
            </div>
            <!-- Full-width card for Japan Aftershocks -->
            <div class="bg-white p-6 rounded-lg shadow-lg lg:col-span-2">
                <h2 class="text-xl font-semibold text-gray-700 mb-4">Plot 5: Case Study of the 2011 Japan Aftershock Sequence</h2>
                <img src="data:image/png;base64,{{ plots.plot5 }}" alt="Japan Aftershock Sequence" class="w-full h-auto rounded">
            </div>
            <!-- Full-width card for Bubble Chart -->
            <div class="bg-white p-6 rounded-lg shadow-lg lg:col-span-2">
                <h2 class="text-xl font-semibold text-gray-700 mb-4">Plot 6: Magnitude vs. Depth Bubble Chart</h2>
                <img src="data:image/png;base64,{{ plots.plot7 }}" alt="Magnitude vs. Depth" class="w-full h-auto rounded">
            </div>
            <!-- Full-width card for Reporting Agencies -->
            <div class="bg-white p-6 rounded-lg shadow-lg lg:col-span-2">
                <h2 class="text-xl font-semibold text-gray-700 mb-4">Plot 7: Magnitude Distribution by Reporting Agency</h2>
                <img src="data:image/png;base64,{{ plots.plot8 }}" alt="Magnitude by Reporting Agency" class="w-full h-auto rounded">
            </div>
            <!-- Full-width card for the Map -->
            <div class="bg-white p-6 rounded-lg shadow-lg lg:col-span-2">
                <h2 class="text-xl font-semibold text-gray-700 mb-4">Plot 8 Global Nuclear Explosion Events</h2>
```

```
                <div class="folium-map">
                    {{ plots.map10 | safe }}
                </div>
            </div>
        </div>
        <footer class="text-center mt-10 text-gray-500">
            <p>Dashboard generated using Python, Flask, and
               Matplotlib.</p>
        </footer>
    </div>
</body>
</html>
```

Code Snippet 10: HTML template for rendering the dashboard page.