A Deep Dive into the Global Terrorism EDA Project

2025-10-03

Contents

1 Setting Up Our Workshop (The Imports)

Every great artist or detective needs their tools. In Python, we start by "importing" our toolkits, which are called libraries.

The Goal

To bring all the necessary Python libraries into our notebook so we can use their special functions.

The Code

```
# Import our standard tools
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Import our new visualization tool, Seaborn, with the nickname 'sns'
import seaborn as sns

# Set a nice style for our plots
sns.set_style('darkgrid')
```

The Explanation

- `import pandas as pd`: Pandas is our magical spreadsheet tool. It helps us organize, clean, and work with data in tables (which we call DataFrames). We

give it the nickname pd to save typing.

- `import numpy as np`: NumPy is the master of math, especially for large lists of numbers. Pandas uses it in the background, but it's good practice to import it. We give it the nickname np.
- `import matplotlib.pyplot as plt`: This is our main artist for drawing charts and graphs. We only need the pyplot part of it, and we give it the nickname `plt`.
- `import seaborn as sns`: Seaborn is like an expert assistant to our artist, Matplotlib. It helps us create more beautiful and complex charts with less code. It has great default styles and color schemes. We give it the nickname `sns`.
- `sns.set_style('darkgrid')`: This is a command from Seaborn. It's like choosing a background for our drawings. "darkgrid" adds a gray background with white grid lines, which makes our charts look professional and easy to read.

2 Loading the Evidence (Reading the CSV File)

Now we need to load the data file (`gtd.csv`) into our Pandas DataFrame.

The Goal

To transfer the data from the CSV file on our computer into a variable in our Python notebook so we can start working on it.

The Code

```
df = pd.read_csv('gtd.csv', encoding='latin1')
```

The Explanation

- `df = ...`: We are creating a variable named `df` (short for DataFrame) to hold all our data.
- `pd.read_csv(...)`: This is the main command from Pandas to read a Comma-Separated Values (CSV) file.
- `'gtd.csv'`: This is the name of the file we want to read. It must be in the same folder as our notebook or uploaded to our Colab session.
- `encoding='latin1'`: This is an important real-world step. Think of "encoding" as the language the file was written in. Most files today use a "language" called UTF-8. However, this dataset is older and contains special characters, so it was saved in a different "language" called latin1. By telling Pandas this, we prevent errors when it tries to read the file.

3 First Look & Focusing the Investigation

This dataset is huge (180,000+ rows, 135 columns). Trying to analyze everything at once is impossible. We must first understand its size and then intelligently select the most important columns.

The Goal

To get a quick overview of the data and then create a smaller, more manageable DataFrame with only the columns we need for our analysis.

The Code

```
# 1. Check the shape of the data
print(f"The dataset has {df.shape[0]} rows and {df.shape[1]} columns.")

# 2. Select a subset of columns we care about
df = df[['iyear', 'imonth', 'iday', 'country_txt', 'region_txt',
         'provstate', 'city', 'attacktype1_txt', 'targtype1_txt',
         'gname', 'weaptype1_txt', 'nkill', 'nwound']]

# 3. Rename the columns to be cleaner and easier to use
df.rename(columns={
    'iyear': 'Year', 'imonth': 'Month', 'iday': 'Day', 'country_txt': 'Count
    'region_txt': 'Region', 'provstate': 'State', 'city': 'City',
    'attacktype1_txt': 'Attack_Type', 'targtype1_txt': 'Target_Type',
    'gname': 'Group', 'weaptype1_txt': 'Weapon_Type',
    'nkill': 'Killed', 'nwound': 'Wounded'
}, inplace=True)
```

The Explanation

Checking the Shape:

- `df.shape` gives us the dimensions of our DataFrame as (rows, columns). We use this to understand the scale of our data right away.

Column Selection:

- `df = df[...]`: We are overwriting our original `df` with a new version that contains only a selection of columns.
- `[['iyear', 'imonth', ...]]`: The double square brackets `[[...]]` are used to provide a list of column names that we want to keep. We chose these columns because they seem most relevant to our mission (when, where, what, who, how many casualties). This is data reduction.

Renaming Columns:

- `df.rename(...)`: This is the Pandas command to rename columns or rows.
- `columns={…}`: We tell it we want to rename columns. We provide a dictionary where the key is the old name (`'iyear'`) and the value is the new name (`'Year'`).
- `inplace=True`: This is a very important command. If `inplace=False` (the default), Pandas would just show us what the renamed DataFrame looks like but wouldn't actually save the changes to `df`. By setting `inplace=True`, we are telling Pandas: "Make these changes directly to my DataFrame `df` and save them." It's like editing a document and hitting the save button.

4 Cleaning the Clues (Handling Missing Data & Feature Engineering)

Real-world data is messy. It often has missing values (empty cells). We need to clean these up before we can do any analysis.

The Goal

To find and handle missing data in our key columns (`Killed`, `Wounded`) and then create a new, more useful column called `Casualties`.

The Code

```
# 1. Check for missing values
print(df.isnull().sum())

# 2. Fill missing 'Killed' and 'Wounded' values with 0
df['Killed'] = df['Killed'].fillna(0)
df['Wounded'] = df['Wounded'].fillna(0)

# 3. Create a new 'Casualties' column (Feature Engineering)
df['Casualties'] = df['Killed'] + df['Wounded']
```

The Explanation

Finding Missing Values:

- `df.isnull()`: This command goes through the entire DataFrame and replaces every cell with either `True` (if it's empty) or `False` (if it has a value).
- `.sum()`: When used after `.isnull()`, this command counts all the `True` values in each column. The result is a list of columns and how many empty cells each one has.

Filling Missing Values:

- `df['Killed'].fillna(0)`: We select the `Killed` column and use the `.fillna(0)` command. This finds all the empty cells (known as NA or NaN) in that column and fills them with the number 0. We do this because it's a reasonable assumption that if the number of killed wasn't reported, it was likely zero. We do the same for the `Wounded` column.

Feature Engineering:

- This is the process of creating new data features from existing ones. It's like being a clever detective who combines two separate clues to make a new, more powerful one.
- `df['Casualties'] = ...`: We are creating a brand new column named `Casualties`.
- `df['Killed'] + df['Wounded']`: The value for each row in our new column will be the sum of the values in the `Killed` and `Wounded` columns for that same row. This gives us a single number to represent the total human cost of an attack.

5 Visualizing the Story (The Core Analysis)

Now for the most exciting part: creating charts to see the patterns in our data! For each chart, we'll explain the code and why we chose that specific type of chart.

Question 1: How has terrorism changed over time?

[width=0.8]example-image

The Code

```
plt.figure(figsize=(15, 7))
sns.countplot(x='Year', data=df, palette='viridis')
plt.title('Number of Terrorist Attacks Per Year', fontsize=20)
plt.ylabel('Number of Attacks')
plt.xticks(rotation=90)
plt.show()
```

The Explanation

- `plt.figure(figsize=(15, 7))`: We're telling Matplotlib, "Prepare a canvas for drawing that is 15 inches wide and 7 inches tall." This gives our plot plenty of space.
- `sns.countplot(...)`: This is a Seaborn command that automatically counts the occurrences of each item in a column and draws a bar for it.
- `x='Year'`: We want the bars to be arranged along the x-axis (the bottom axis) based on the `Year` column.

- `data=df`: We specify that the data should come from our `df` DataFrame.
- `palette='viridis'`: This sets a nice color scheme for the bars.
- `plt.title(...), plt.ylabel(...)`: These commands add a title to the top of our chart and a label to the y-axis (the vertical axis).
- `plt.xticks(rotation=90)`: This takes the labels on the x-axis (the years) and rotates them 90 degrees so they are vertical. This is essential to prevent them from overlapping and becoming unreadable.
- `plt.show()`: This command displays the final chart.

Why this chart? A bar chart (or count plot) is perfect for showing the exact count of events for different categories (in this case, years). A line chart would also have been a great choice here to emphasize the trend over time.

Question 2: Which regions are the most affected?

[width=0.8]example-image

The Code

```
top_regions = df['Region'].value_counts().nlargest(10)

plt.figure(figsize=(12, 6))
sns.barplot(y=top_regions.index, x=top_regions.values, palette='mako')
plt.title('Top 10 Most Affected Regions', fontsize=20)
plt.xlabel('Number of Attacks')
plt.ylabel('Region')
```

```
plt.show()
```

The Explanation

- `df['Region'].value_counts()`: This is a super useful command. It counts how many times each unique region appears in the `Region` column and sorts it.
- `.nlargest(10)`: After counting, we use this command to keep only the top 10 results.
- `sns.barplot(...)`: This Seaborn command draws a bar plot.
- `y=top_regions.index`: We put the region names on the y-axis (vertical). The `.index` contains the names of the regions.
- `x=top_regions.values`: We put the attack counts on the x-axis (horizontal). The `.values` contains the counts for each region.

Why this chart? We chose a horizontal bar chart. This is a pro move! Why? Because the names of the regions are long (e.g., "Middle East & North Africa"). If we made a vertical bar chart, the names on the x-axis would be crowded and unreadable. By putting them on the y-axis, they have plenty of space. Rule of thumb: If your category labels are long, use a horizontal bar chart.

Questions 3 & 4: Most Common Attack Methods and Most Active Groups

The logic for these charts is exactly the same as for the regions chart. We use `.value_counts()` to count the items, `.nlargest()` to select the top ones, and a

horizontal bar chart for clear, readable labels.

For the “Most Active Groups” chart, we added one extra step:

```
df[df['Group'] != 'Unknown']
```

This is a filtering step. It tells Pandas to create a temporary DataFrame that includes only the rows where the `Group` column is not equal to (`!=`) `'Unknown'`. We do this because the `'Unknown'` category is so large it would hide the details of the other groups, and we are specifically interested in the known attackers.

This entire process — from loading and cleaning data to asking questions and creating insightful visualizations — is the complete Exploratory Data Analysis (EDA) pipeline. You have successfully taken a massive, messy, real-world dataset and turned it into a clear story with actionable insights.

6 The Intelligence Briefing (Our Overall Findings)

After conducting our detailed investigation and looking at all the charts, we can now step back and see the complete story. If we were presenting this to a global security organization, this would be our main summary.

The Big Picture: A Story in Four Key Points

1. A Drastic Shift in Modern History: For decades (1970-2000), terrorist attacks occurred at a relatively stable, lower rate. However, the 21st century, particularly after 2004, saw a dramatic and terrifying increase in the frequency of attacks, peaking in 2014. Our analysis shows a clear "before" and "after," marking a new era of global conflict.
2. Conflict is Highly Concentrated: Terrorism is not a uniformly spread global problem. Our data clearly identifies specific hotspots. The overwhelming majority of attacks and casualties are concentrated in the Middle East & North Africa and South Asia. This tells us where security and humanitarian efforts are most critically needed.
3. The Dominant Tactics are Clear: While there are many ways attacks are carried out, Bombings and Explosions are the most common tactic by a huge margin. This suggests that the primary capability of most major groups revolves around acquiring and using explosive materials. This is followed by armed assaults.
4. A Few Groups Drive the Majority of Attacks: While thousands of groups are listed, a small number of highly active and deadly organizations are responsible for a large percentage of the attacks. Our analysis identified the Taliban and the Islamic State of Iraq and the Levant (ISIL) as the most prominent actors in this dataset. The fact that "Unknown" is the top category also tells us that in many cases, no group claims responsibility, which is an important finding in itself.

In short, the story of modern terrorism from 1970-2017 is one of a dramatic escalation in the 21st century, heavily focused in specific regions, and driven by a

few key groups using well-established tactics.

7 Why This Project Matters (The Need and Impact)

This is the most important part of your project. In an interview, you need to explain why you chose to do this analysis and what its value is.

Why did we need to do this project?

Analyzing data like this is crucial for making sense of complex global problems. Without data analysis, our understanding of terrorism would be based only on news headlines and anecdotes. Data allows us to see the real, quantifiable patterns and answer critical questions.

For Governments and Security Agencies:

- **Resource Allocation**: This analysis helps leaders decide where to focus their counter-terrorism efforts. Seeing that the Middle East and South Asia are hotspots tells them where to direct intelligence, security, and diplomatic resources.
- **Threat Assessment**: By understanding that bombings are the most common attack type, agencies can focus on disrupting supply chains for explosive materials and develop strategies to mitigate such attacks.
- **Policy Making**: The trend over time can inform international policy. The dramatic rise after 2004 could be studied to understand what global events might have caused it, helping to prevent future escalations.

For Researchers and Journalists:

- **Contextualizing Events**: This data provides a historical backdrop for news stories. When an attack happens, journalists can use this data to understand if it's part of a larger trend or an isolated incident.
- **Data-Driven Storytelling**: It allows for the creation of powerful, evidence-based reports on the human cost of conflict, moving beyond opinions to factual analysis.

For You, The Data Analyst (Why it's a great portfolio project):

- **Shows Technical Skill on "Real" Data**: You successfully handled a large (180,000+ rows), messy (135 columns, missing values, weird encoding) dataset. This is far more impressive than a small, clean "toy" dataset.
- **Demonstrates Maturity and Social Awareness**: Tackling a serious, impactful topic shows that you are interested in using your data skills to understand important real-world issues. It sets you apart from candidates who only analyze cartoon characters or video game sales.
- **Proves Your Ability to Tell a Story**: You didn't just make charts. You started with a huge, confusing file and ended with a clear, concise intelligence briefing. This ability to find the story in the data is the most valuable skill a data analyst can have.

With this project, you have not only mined a vast dataset but also crafted a compelling narrative that transforms numbers and charts into meaningful insights about global terrorism.