CMSC 636 – Data Visualization

Assignment 1 – Part 2 Tutorial

Faisal Rasheed Khan

VB02734

vb02734@umbc.edu

Dataset:

Columns: 'aiddata_id', 'year', 'donor', 'recipient', 'commitment_amount_usd_constant', 'coalesced_purpose_code', 'coalesced_purpose_name'

Tutorial Steps:

Step 0: Install all the required packages.

- Use the command: pip install #package_name
- Install pandas, matplotlib, numpy, squarify, scikitlearn

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import squarify
```

Step 1: Loading dataset into Python

- Download the dataset
- Load the dataset in python using pandas library by specifying the path of the dataset

```
# Specify the path to your Excel file
excel_file_path = 'C:/Users/juver/Downloads/Aid_Data_Example.xlsx'

# Read the Excel file into a pandas DataFrame
df = pd.read_excel(excel_file_path, skiprows=1)
```

- Look at the dataset, how it is

In [3]:
```
print(df.head)
```

```
<bound method NDFrame.head of      aiddata_id  year          donor  count of donor              recipient  \
0      12191891  1998      Australia               1              Indonesia
1      27339565  2003      Australia               1              Singapore
2      34378730  2007      Australia               1               Colombia
3      13308187  1999      Australia               1            Timor-Leste
4      38835178  2008      Australia               1  Bilateral, unspecified
..          ...   ...            ...             ...                    ...
494    34085698  2007  United States               1  Bilateral, unspecified
495    26038958  2005  United States               1                 Brazil
496    50081282  2009  United States               1               Pakistan
497    50064569  2009  United States               1                Ukraine
498    29738512  2006  United States               1                Somalia

     count of receiver  commitment_amount_usd_constant  \
0                    1                    1.033230e+05
1                    1                    2.979690e+02
2                    1                    1.395560e+05
3                    1                    2.271530e+04
4                    1                    4.468550e+04
..                 ...                             ...
494                  1                    1.422430e+07
495                  1                    3.255560e+05
496                  1                    1.761800e+05
497                  1                    5.495000e+04
498                  1                    4.501060e+05

     coalesced_purpose_code                          coalesced_purpose_name
0                     24040              Informal/semi-formal fin. intermed.
1                     15105  Government and civil society, purpose unspecif...
2                     15130                    Legal and judicial development
3                     12220                               Basic health care
4                     15110         Public sector policy and adm. management
..                      ...                                              ...
494                   13020                         Reproductive health care
```

Step 2: Data Preprocessing

- Processing is the main step for a good visualization which appears good for a user to have information insights.
- First, we need to clean the data, as our data is clean and free of outliers, we proceed for the processing step.

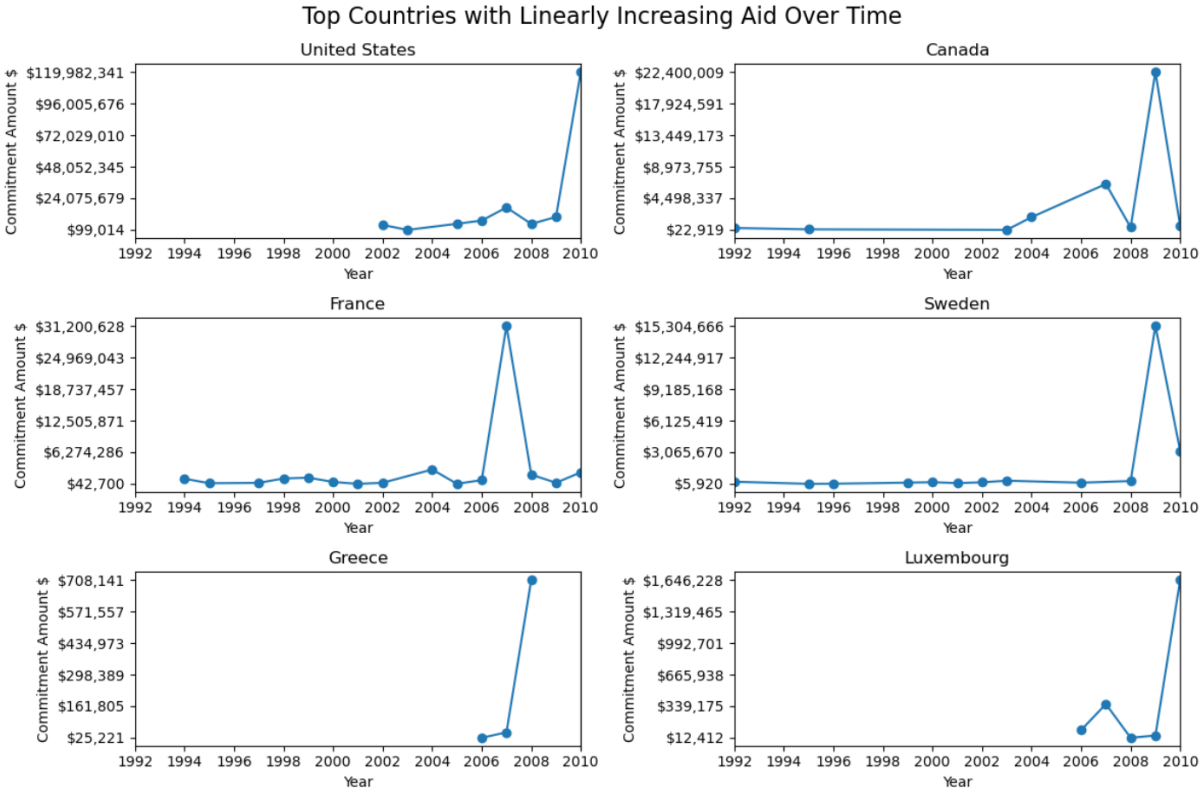
In [6]:
```
# Group by donor country and year by summing the commitment amounts for each donor in that particular year
grouped_df = df.groupby(['donor', 'year'])['commitment_amount_usd_constant'].sum().reset_index()

# Filter the DataFrame for top countries
filtered_df = grouped_df[grouped_df['donor'].isin(top_country_names)]

#linear regression for each country
country_slopes = {}
for country, data in grouped_df.groupby('donor'):
    x = data['year'].values.reshape(-1, 1)
    y = data['commitment_amount_usd_constant'].values
    model = LinearRegression().fit(x, y)
    slope = model.coef_[0]
    country_slopes[country] = slope
```

In [28]:
```
N = 10

# group by donor and amount sum
total_aid_by_donor = df.groupby('donor')['commitment_amount_usd_constant'].sum().nlargest(N)

# Normalize the data for color mapping
norm = mcolors.Normalize(vmin=min(total_aid_by_donor), vmax=max(total_aid_by_donor))
colors = [plt.cm.Blues(norm(value)) for value in total_aid_by_donor]

# donors
labels_donor = [f"{index}\n${value:,.0f}" for index, value in total_aid_by_donor.items()]

total_aid_by_recipient = df.groupby('recipient')['commitment_amount_usd_constant'].sum().nlargest(N)

norm_recipient = mcolors.Normalize(vmin=min(total_aid_by_donor), vmax=max(total_aid_by_donor))
colors_recipient = [plt.cm.Blues(norm_recipient(value)) for value in total_aid_by_recipient]

# recipients
labels_recipient = [f"{index}\n${value:,.0f}" for index, value in total_aid_by_recipient.items()]
```

Step 3: Visualization for Vis/Task 3

- We will be plotting here the countries whose aid has been increased over the years.
- To know this information we use linear regression to plot the line for the data.
- With this plot we will be knowing which countries has increased aid over the years.

```
# subplots
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(12, 8), sharey=False)  # Updated for independent y-axis
fig.suptitle('Top Countries with Linearly Increasing Aid Over Time', fontsize=16)

x_axis_min = 0
x_axis_max = 1

# Normalize
scaler = MinMaxScaler(feature_range=(x_axis_min, x_axis_max))

for i, ax in enumerate(axes.flatten()):
    if i < len(top_country_names):
        country = top_country_names[i]
        country_data = filtered_df[filtered_df['donor'] == country].sort_values('year')

        # Scale the y-axis values to the range 0 to 1
        scaled_values = scaler.fit_transform(country_data['commitment_amount_usd_constant'].values.reshape(-1, 1)).flatten()

        # Plot the data
        ax.plot(country_data['year'], scaled_values, label=country, marker='o')
        ax.set_title(country)
        ax.set_xlabel('Year')
        ax.set_ylabel('Commitment Amount $')

        # same range of years for all subplots
        ax.set_xlim(global_min_year, global_max_year)

        # Normalization to USD values
        inv_transformed_ticks = scaler.inverse_transform(ax.get_yticks().reshape(-1, 1)).flatten()
        ax.set_yticklabels(["${:,.0f}".format(tick) for tick in inv_transformed_ticks])


plt.tight_layout()
plt.show()
```

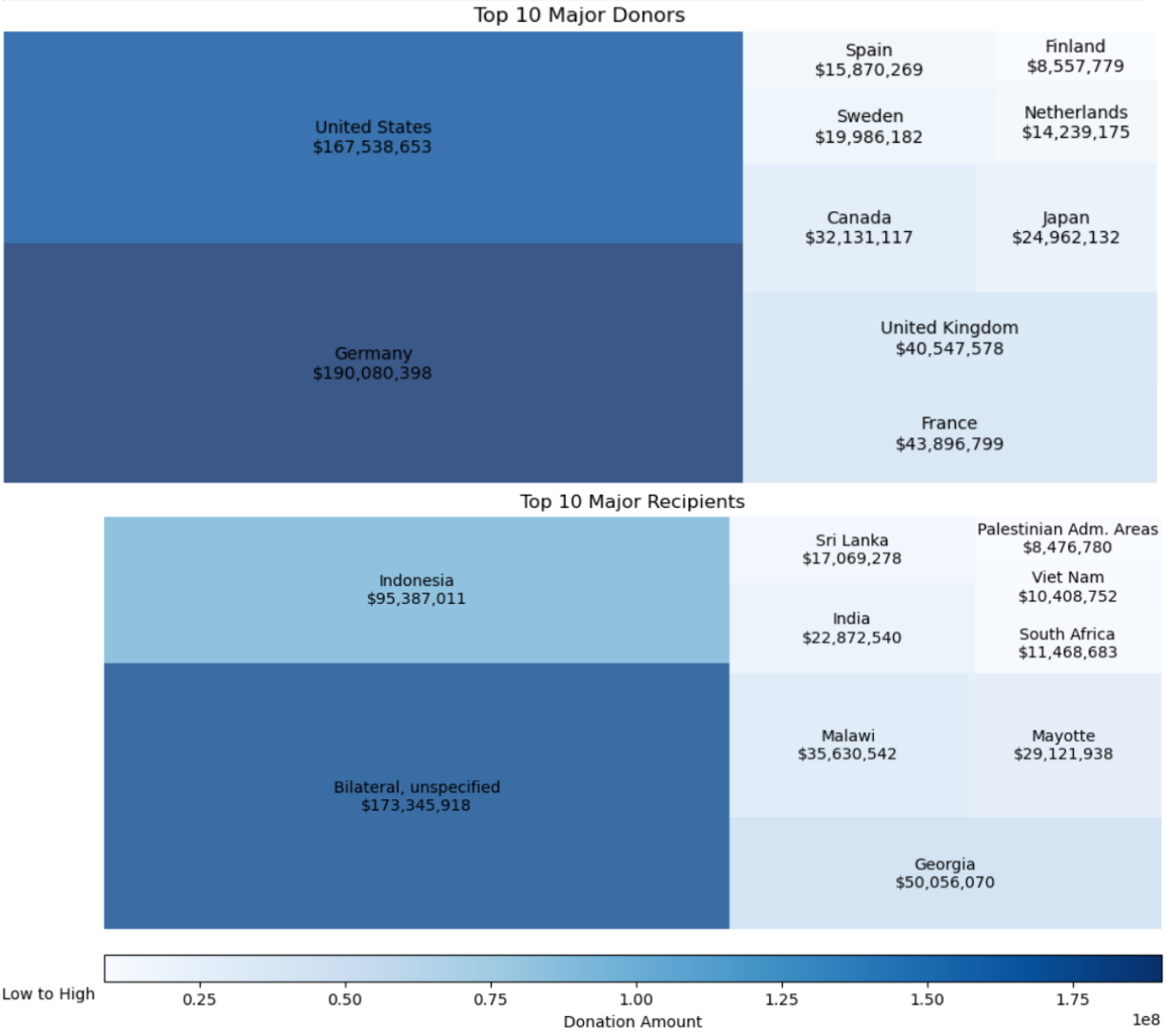


Step 4: Visualization for Vis/Task 4

- We will be visualizing the plot using matplotlib and will be plotting Treemap.
- The Squarify package provides the Treemap in python.

- Here, created the custom legend to give the description regarding the colors

```
plt.figure(figsize=(12, 6))
squarify.plot(sizes=total_aid_by_donor, label=labels_donor, alpha=0.8, color=colors)
plt.title(f'Top {N} Major Donors')
plt.axis('off')

plt.figure(figsize=(12, 6))
squarify.plot(sizes=total_aid_by_recipient, label=labels_recipient, alpha=0.8, color=colors_recipient)
plt.title(f'Top {N} Major Recipients')
plt.axis('off')

# Create a custom legend
sm = plt.cm.ScalarMappable(cmap=plt.cm.Blues, norm=norm)
sm.set_array([])  # You have to set_array for ScalarMappable.
cbar = plt.colorbar(sm, orientation='horizontal', pad=0.05, aspect=40)
cbar.set_label('Donation Amount')
cbar.ax.text(0, -0.5, ' High', va='center', ha='left')
cbar.ax.text(1, -0.5, 'Low to', va='center', ha='right')

plt.show()
```



Step 5: Adding Caption

- Matplotlib provides the caption option, we need to provide customized text and the matplotlib displays that.

```
caption_text = "a. [Donor] TreeMap representation of major aid provided by Donor countries to the recipient countries for a commi
plt.figtext(1,1,caption_text, wrap=True,  fontsize=10, color='gray')
```

References:

[1] Tierney, Michael J., Daniel L. Nielson, Darren G. Hawkins, J. Timmons Roberts, Michael G. Findley, Ryan M. Powers, Bradley Parks, Sven E. Wilson, and Robert L. Hicks. 2011. More Dollars than Sense: Refining Our Knowledge of Development Finance Using AidData. World Development 39 (11): 1891-1906

[2] Matplotlib

Tutorials — Matplotlib 3.8.3 documentation

[3] Pandas

pandas.DataFrame — pandas 2.2.1 documentation (pydata.org)

[4] Scikit Learn

scikit-learn: machine learning in Python — scikit-learn 1.4.1 documentation

[5] Squarify

Treemaps in matplotlib with squarify | PYTHON CHARTS (python-charts.com)

[6] Numpy

NumPy -