Name: OM M PATEL
Roll No.: 21BCP094
Group: G3
Division: D2

LAB ASSIGNMENT-1

Title: Discuss and Analyze Different Data Visualization Tools

Objective:

This lab assignment aims to explore and analyze various data visualization tools used for representing and understanding complex datasets. This assignment will give you insights into the strengths, weaknesses, and practical applications of different visualization tools.

1. Introduction to Data Visualization

Data visualization describes any effort to help people understand the significance of data by placing it in a visual context. Patterns, trends and correlations that might go undetected in text-based data can be exposed and recognized easier with data visualization software. If done right, they offer key insights into complicated data-sets in ways that are meaningful and intuitive, further helping in decision making. In other words, data visualizations turn large and small data-sets into visuals that are easier for the human brain to understand and process

2. Selected Data Visualization Tools

A. Matplotlib

B. Plotly

C. Seaborn

D. ggplot2 (R)

3. Capabilities and Features:

A. Matplotlib:

Matplotlib makes scientific-level plotting easy and it's Python library.Pyplot is a module in it which provides a MATLAB-like interface.

- It provides a wide variety of 2D plots (e.g., line, bar, scatter, histogram).
- A large variety of customizable options like line styles, markers, and color types.
- Basic interactive features like zooming and support subplotting.
- Good compatibility with pandas, numpy, and data science tools.

B. Plotly:

It paves ease for building interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.

- Provides easily embedded plots for web applications.
- Provides a wide range of plots, from 3D to world map to etc.
- More customizable than Matplotlib, has more options for improving aesthetics.
- A most unique feature is, that it provides options to host plots online.

C. Seaborn:

It's built on top of matplotlib, mostly used for mathematical and statistical graphs, and has a high-level interface for drawing attractive and informative statistical graphics.

- The foremost MOAT of Seaborn is, that it can build statistical graphs with lesser complexity
- It's Function based, has most of functions to create complex graphs like heatmaps and etc.
- It also integrates well with pandas and numpy.
- Easier to create graphs for aggregated data.

D. ggplot2(R):

It helps to creating plots and figures, based on grammar of the graphics(language). You provide the data, tell ggplot2 how to map variables to its behaviour, and what graphical tools to use, and it takes care of the details.It's applied using R language.

- Enables the addition of multiple layers of data and aesthetic mappings.
- Provides higher customization options for themes, colors, and plot elements.
- built around the grammar of graphics, providing a consistent and logical framework.
- Integrates easily with data frames and the tidyverse suite of packages.

4. Practical Scenario:

A. **Matplotlib** Can be used by the weather department for plotting time series forecasting of weather.

B. **Plotly** can be used by TV/OTT broadcasters to show interactive plots in a dashboard format with live updating capabilities.

C. **Seaborn** can be used by researchers or students while building their models or training their datasets to create correlation maps and other kinds of statistical mapping.

D. **Ggplot2** can be used for large datasets where relationships also have to be mapped, allowing for easy comparison across different groups.

5. Strengths And Weaknesses

A) Matplotlib

Strengths:

- Good for basic plots and subplots. Especially with static plots.
- Easy to understand and can be used in scientific research by researchers.
- It acts as a base for newly formed data visualization libraries.

Weaknesses:

- As it provides a basic level of visualization it requires complex coding for certain plots.
- Static by default, requiring additional libraries (e.g., Plotly, mpld3) for interactivity.

B) Plotly

Strengths:

- Higher Detailing and Aesthetics options.
- Supports a wide range of complex plots.

Weaknesses:

- Steeper learning curve compared to Matplotlib and Seaborn.
- Can be more resource-intensive, especially while hosting plots on the web.

C) Seaborn

Strengths:

- Eases Statistical Graph Creation.
- As it has functions, it Supports a wide range of complex plots.

Weaknesses:

- Least customizable when compared to Plotly or even Matplotlib.
- As it's more focused on statistical plots, for non-statistical plots it may be inefficient.

D) ggplot2

Strengths:

- A consistent and logical framework for building plots..
- Supports a wide range of multi-layered plots.

Weaknesses:

- Steeper learning curve compared to Matplotlib and Seaborn.
- Can be less intuitive for simple plots compared to other libraries.

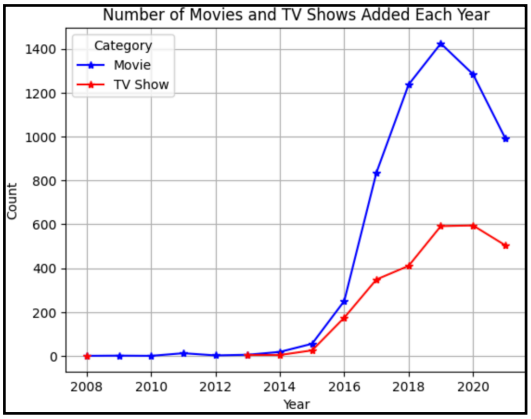
6. Code Example

Output

A) Matplotlib

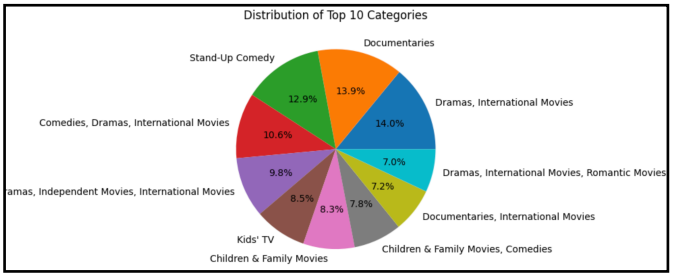
1.

```
import matplotlib.pyplot as plt #1
plt.figure(figsize=(10, 6))
tp.plot(kind='line', marker='*',colormap='bwr')
plt.title('Number of Movies and TV Shows Added
 Each Year')
plt.xlabel('Year')
plt.ylabel('Count')
plt.legend(title='Category')
plt.grid()
plt.show()
```



2.

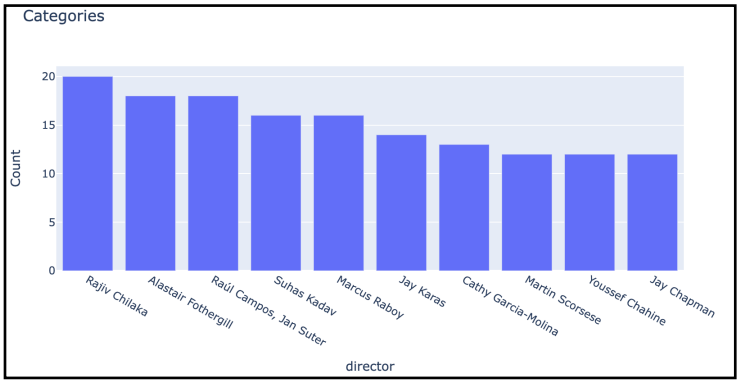
```
category = df['listed_in'].value_counts()
category
cat = category[:10]
plt.pie(cat, labels=cat.index, autopct='%1.1f%%')
plt.title('Distribution of Top 10 Categories')
plt.show()
```



B) Plotly

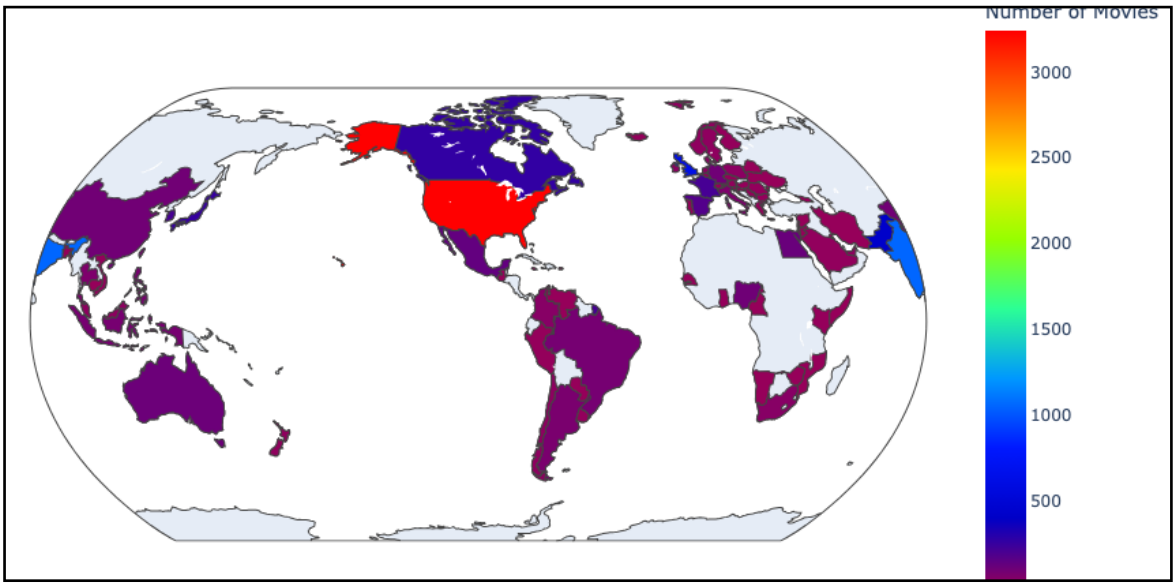
1.

```
dr = df['director'].value_counts()
dr = dr[dr.index != 'Not Given']
dr
fig = px.bar(dr10, dr10.index, y=dr10.values,
             labels={'x': 'listed_in', 'y': 'Count'},
             title='Categories')
fig.show()
```



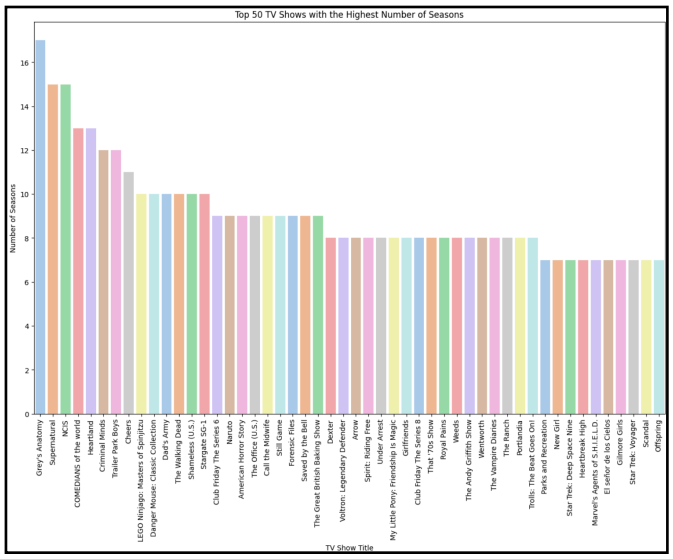
2.

```
country_movie_counts = df1['country'].value_counts().reset_index()
country_movie_counts = country_movie_counts[country_movie_counts['country'] != 'Not Given']
country_movie_counts.columns = ['country', 'count']
country_movie_counts

import pycountry
def get_iso_alpha_3(country_name):
    try:
        return pycountry.countries.lookup(country_name).alpha_3
    except LookupError:
        return None
country_movie_counts['iso_alpha_3'] = country_movie_counts['country'].apply(get_iso_alpha_3)
country_movie_counts = country_movie_counts.dropna(subset=['iso_alpha_3'])

fig = px.choropleth(country_movie_counts, locations='iso_alpha_3', color='count',
                    color_continuous_scale="rainbow",
                    hover_name='country',
                    labels={'count':'Number of Movies'},
                    title="Number of Movies Produced by Country")
fig.update_geos(projection_type="natural earth")
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



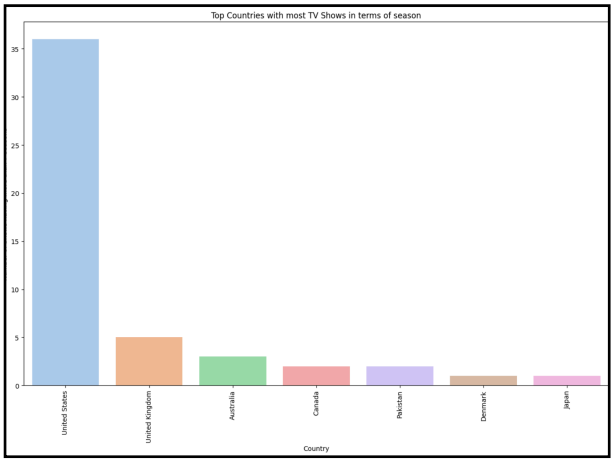
C) Seaborn

```
tv = df[df['type'] == 'TV Show']
tv['no_of_seasons'] = tv['duration'].str.extract(r'(\d+)').astype(int)

top_tv = tv.sort_values(by='no_of_seasons', ascending=False).head(50)
country_wise_top_tv = top_tv[['title', 'no_of_seasons','country']]
top_tv[['title', ‘no_of_seasons']]

plt.figure(figsize=(16, 10))
sns.barplot(data=top_tv, x='title', y='no_of_seasons', palette='pastel')
plt.title('Top 50 TV Shows with the Highest Number of Seasons')
plt.xlabel('TV Show Title')
plt.ylabel('Number of Seasons')
plt.xticks(rotation=90)  # Rotate x-axis labels to fit long titles
plt.show()
```



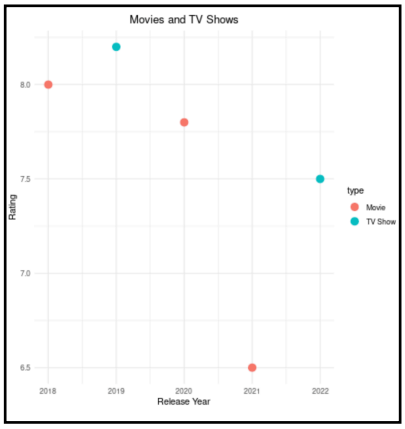
2.

```
country_counts = country_wise_top_tv['country'].value_counts().reset_index()
country_counts.columns = ['country', 'number_of_shows']
country_counts

plt.figure(figsize=(16, 10))
sns.barplot(data=country_counts, x='country', y='number_of_shows', palette='pastel')
plt.title('Top Countries with most TV Shows in terms of season')
plt.xlabel('Country')
plt.ylabel('Number of Shows having more than 7 seasons')
plt.xticks(rotation=90)  # Rotate x-axis labels to fit long titles
plt.show()
```

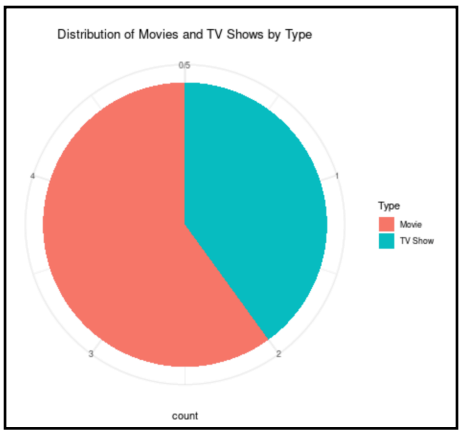


D) ggplot2

```
library(ggplot2)

movies <- data.frame(
    title = c("The Adventure", "Mystery Island", "Romantic Journey", "Future Wars", "The Great Heist"),
  release_year = c(2020, 2019, 2021, 2018, 2022),
  rating = c(7.8, 8.2, 6.5, 8.0, 7.5),
  type = c("Movie", "TV Show", "Movie", "Movie", "TV Show")
)
print(movies)

# Scatter plot: Release Year vs Rating
ggplot(data = movies, aes(x = release_year, y = rating, color = type)) +
  geom_point(size = 4) +
  labs(title = "Movies and TV Shows",
```



```
      x = "Release Year",
      y = "Rating") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))


# Plotting: Pie chart of Type distribution
ggplot(data = movies, aes(x = "", fill = type)) +
  geom_bar(width = 1) +
  coord_polar("y") +
  labs(title = "Distribution of Movies and TV Shows by Type",
      fill = "Type") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



7. Suitability for Different Scenarios

- **Matplotlib**: Ideal for static plots, publication-quality plots where fine-grained control over the plot appearance is required. Best suitable for research or student projects, where customization is also required along with not-so-complex plotting.
- **Seaborn**: Ideal for statistical data visualization and exploratory data analysis. It simplifies the process of creating complex statistical plots and is best used when working with Pandas data frames. Best suitable for research projects and statistical-related areas.
- **Plotly**: Ideal for creating interactive plots 3D-2D and dashboards. It is the best choice for web applications, live data visualizations, and scenarios where user interaction with the plot is necessary. Best suitable for broadcasters who have to present live plots/graphs.
- **ggplot2(R):** Ideal for comparing multiple subsets of large data in a single visualization, as it supports multi-layered plots. Can be used to develop solutions around large health or public datasets.