Visualize your Data

Data Visualization Libraries

- MatPlotLib - General-purpose plotting
- Seaborn - Statistical graphics and beautiful themes
- Plotly - Interactive and web-based plots
- Geopandas - Geospatial data visualization

In [ ]:
```
#load pandas

import pandas as pd
```

MatPlotLib

In [ ]:
```
# import the library

import matplotlib.pyplot as plt

# load dataframe

scotus = pd.read_csv("scotus_approval.csv")

# set datatime

scotus["date"] = pd.to_datetime(scotus["date"])

scotus = scotus[(scotus["date"] >= '2023-09-29')]
```

In [ ]:
```
# filter pollster to YouGov

scotus = scotus[scotus["pollster"] == "YouGov"]
scotus
```

In [ ]:
```
# create a line graph of scotus approval rating

#use plot to call the line plot shape

scotus.plot()
```

In [ ]:
```
# set date and yes appoval

scotus.plot(x="date", y="yes")


# Show the plot
plt.show()
```

In [ ]:
```
# Multiple Variables: You can also plot multiple y-variables by passing a list to t

scotus.plot(x="date", y=["yes", "no"])

plt.show()
```

In [ ]:
```
# Stylize the graph
scotus.plot(x="date", y="yes",
            color="red",       # color
            linewidth=0.75,    # line size
            linestyle='--',    # Dotted line style
            marker='o',        # Circle markers for each data point
            markersize=5       # Size of the markers
           )
```

In [ ]:
```
scotus.plot(x="date", y="yes",
            color="red",       # Color
            linewidth=0.75,    # Line size
            linestyle='--',    # Dotted line style
            marker='o',        # Circle markers for each data point
            markersize=5       # Size of the markers
           )


# Adding labels and title
plt.title("Scotus Approval Ratings Over Time")
plt.xlabel("Date")
plt.ylabel("Approval Ratings (%)")

# plt.xlabel("")
```

Adjust Font Size

In [ ]:
```
# Plot the 'yes' approval ratings
scotus.plot(x="date", y="yes",
            color="red",       # Color of the line
            linewidth=0.75,    # Width of the line
            linestyle='--',    # Dotted line style
            marker='o',        # Circle markers for each data point
            markersize=5       # Size of the markers
           )

# Adding title and axis labels with custom font sizes
plt.title("Scotus Approval Ratings Over Time".upper(), fontsize=20, color="red", fo
plt.xlabel("")  # No label for the x-axis
plt.ylabel("Approval Ratings (%)", fontweight="bold")  # Bold y-axis label


# Remove Legend
plt.legend().set_visible(False)

# Adjusting tick parameters for better readability
plt.tick_params(axis='both', labelsize=8)              # Set the font size for t
```

```python
# Display the plot
plt.show()
```

Adjust Date Format and Breaks

Date Formats

In [ ]:
```python
import matplotlib.dates as mdates

scotus.plot(x="date", y="yes",
            color="red",      # Color of the line
            linewidth=0.75,   # Width of the line
            linestyle='--',   # Dotted line style
            marker='o',       # Circle markers for each data point
            markersize=5      # Size of the markers
           )

# Adding title and axis labels with custom font sizes
plt.title("Scotus Approval Ratings Over Time".upper(), fontsize=20, color="red", fo
plt.xlabel("")  # No label for the x-axis
plt.ylabel("Approval Ratings (%)", fontweight="bold")  # Bold y-axis label


# Remove legend
plt.legend().set_visible(False)

# Adjusting tick parameters for better readability
plt.tick_params(axis='both', labelsize=8)                 # Set the font size for t


# Add reference line for July 1, 2024
plt.axvline(pd.Timestamp("2024-02-29"), color='green', linestyle=':', linewidth=1.5


# Setting a continuous scale on the y-axis
plt.ylim(25, 55)  # Set limits for y-axis for better visualization

# Customizing date format on the x-axis
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))  # Format for da
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=2))   # Set major ti

# Display the plot
plt.show()
```

Add Addtional Variable

In [ ]:
```python
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import pandas as pd

# Assume scotus DataFrame is already loaded and 'date' is in datetime format
# Plotting approval ratings
plt.plot(scotus["date"], scotus["yes"],
         color="coral",
         linewidth=1.5,
         linestyle="--",
         marker="o",
         markersize=6,
         alpha=0.7,
         label="Approval")  # Label for legend

# Plotting disapproval ratings
plt.plot(scotus["date"], scotus["no"],
         color="skyblue",
         linewidth=1,
         linestyle="-",
         marker="^",
         markersize=4,
         alpha=0.7,
         label="Disapproval")  # Label for legend

# Adding title and labels
plt.title("SCOTUS Ratings Over Time".upper(), fontsize=20)
plt.xlabel("")  # No label for the x-axis
plt.ylabel("Ratings (%)", fontweight="bold")  # Bold y-axis label

# Customize legend
plt.legend(loc="upper left", fontsize=10)

# Adding grid for better readability
plt.grid(True)

# Adjusting tick parameters for better readability
plt.tick_params(axis='both', labelsize=8)  # Set the font size for ticks

# Setting a continuous scale on the y-axis
plt.ylim(25, 70)  # Set limits for y-axis for better visualization

# Add reference line for February 29, 2024
plt.axvline(pd.Timestamp("2024-02-29"), color='green', linestyle=':', linewidth=1.5

# Customizing date format on the x-axis
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))  # Format for da
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=2))   # Set major ti

# Display the plot
plt.show()
```

Export Plot

In [ ]:
```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import pandas as pd

# Assume scotus DataFrame is already loaded and 'date' is in datetime format
# Plotting approval ratings
plt.plot(scotus["date"], scotus["yes"],
         color="coral",
         linewidth=1.5,
         linestyle="--",
         marker="o",
         markersize=6,
         alpha=0.7,
         label="Approval")  # Label for legend

# Plotting disapproval ratings
plt.plot(scotus["date"], scotus["no"],
         color="skyblue",
         linewidth=1,
         linestyle="-",
         marker="^",
         markersize=4,
         alpha=0.7,
         label="Disapproval")  # Label for legend

# Adding title and labels
plt.title("SCOTUS Ratings Over Time".upper(), fontsize=20)
plt.xlabel("")  # No label for the x-axis
plt.ylabel("Ratings (%)", fontweight="bold")  # Bold y-axis label

# Customize legend
plt.legend(loc="upper left", fontsize=10)

# Adding grid for better readability
plt.grid(True)

# Adjusting tick parameters for better readability
plt.tick_params(axis='both', labelsize=8)  # Set the font size for ticks

# Setting a continuous scale on the y-axis
plt.ylim(25, 70)  # Set limits for y-axis for better visualization

# Add reference line for February 29, 2024
plt.axvline(pd.Timestamp("2024-02-29"), color='green', linestyle=':', linewidth=1.5

# Customizing date format on the x-axis
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))  # Format for da
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=2))   # Set major ti

# Save the plot as an image file
plt.savefig("scotus_ratings_over_time.png", dpi=300, bbox_inches='tight')  # Save a

# Display the plot
plt.show()
```

Seaborn

https://seaborn.pydata.org/

In [ ]:
```python
# load the libraries

import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd


reviews = pd.read_csv("customer_reviews.csv")
reviews.dtypes
```

In [ ]:
```python
# create the bar plot

sns.countplot(data = reviews, x = "Department_Name")
```

In [ ]:
```python
# create the bar plot

sns.countplot(data = reviews, x = "Department_Name")

# Add title and labels
plt.title('Count of Reviews by Department')
plt.xlabel('Department Name')
plt.ylabel('')

# Show plot
plt.show()
```

In [ ]:
```python
# Set Seaborn style
sns.set_style("whitegrid")

sns.countplot(data = reviews, x = "Department_Name")

# Add title and labels
plt.title('Count of Reviews by Department')
plt.xlabel('Department Name')
plt.ylabel('')

# Show plot
plt.show()
```

Color Brewer Palettes

https://colorbrewer2.org/

In [ ]:
```python
# Set Seaborn style
sns.set_style("whitegrid")

# Define Color Brewer palette
brewer_palette = sns.color_palette("Pastel1")

# Create countplot with Color Brewer palette
sns.countplot(data=reviews, x="Department_Name", palette=brewer_palette)

# Add title and labels
```

```
plt.title('Count of Reviews by Department')
plt.xlabel('Department Name')
plt.ylabel('Number of Reviews')  # Added ylabel for clarity

# Show plot
plt.show()
```

In [ ]:
```
# Calculate counts and sort by values
department_counts = reviews['Department_Name'].value_counts()
sorted_departments = department_counts.index.tolist()  # Get the sorted order of de

# Create countplot with the specified order
sns.countplot(data=reviews, x="Department_Name", order=sorted_departments)
```

Plotly

https://plotly.com/python/

In [ ]:
```
import plotly.graph_objects as go

# Assuming reviews is a DataFrame containing data
# Create a Plotly histogram figure
fig = go.Figure(data=[go.Histogram(x=reviews["Age"])])

# Display the histogram
fig.show()
```

Add Bins

In [ ]:
```
import plotly.graph_objects as go

# Create a Plotly histogram figure with additional options
fig = go.Figure(data=[go.Histogram(x=reviews["Age"],
                                   # Set number of bins
                                   nbinsx=20,
                                   )])

# Update layout for better appearance
fig.update_layout(title="Histogram of Age",
                  xaxis_title="Age",
                  yaxis_title="Frequency",

                 )

# Display the histogram
fig.show()
```

Change Theme

Plotly provides several built-in themes (also called templates) that you can use to customize the appearance of your visualizations. Below are some of the available themes:

- **plotly** - The default Plotly theme with a classic look.
- **ggplot2** - Inspired by the ggplot2 library, this theme provides a clean and modern aesthetic.
- **seaborn** - Inspired by the Seaborn library, this theme emphasizes visual appeal with muted colors and a grid background.
- **simple_white** - A minimalist theme with a white background, suitable for clean presentations.
- **presentation** - Designed for creating presentation-ready plots, with a focus on visibility and clarity.
- **xgridoff** - A theme with grid lines removed, providing a cleaner look for visualizations.
- **ygridoff** - Similar to xgridoff but removes vertical grid lines.
- **plotly_white** - A theme with a white background and light grid lines, combining elements from Plotly and simple white.
- **plotly_dark** - A dark theme that provides high contrast for visualizations, making them suitable for dark backgrounds.
- **dark** - A simple dark theme, offering high contrast for better visibility.

In [ ]:
```python
import plotly.io as pio

# Set the default theme
pio.templates.default = "ggplot2"  # Change to any available theme like 'plotly', '


# Create a Plotly histogram figure with additional options
fig = go.Figure(data=[go.Histogram(
    x=reviews["Age"],
    # Set number of bins
    nbinsx=20,
    opacity=0.7,
    # Set fill and line colors
    marker=dict(
        color='#ffbf00',  # Fill color
        line=dict(color='#f08080', width=3)  # Line color and width
    )
)])

# Update layout for better appearance
fig.update_layout(
    title="Histogram of Age",
    xaxis_title="Age",
    yaxis_title="Frequency",
    bargap=0.1,  # Set gap between bars
)

# Display the histogram
fig.show()
```

Change Size

In [ ]:

```
# Set the default theme
pio.templates.default = "ggplot2"  # Change to any available theme like 'plotly', '

# Create a Plotly histogram figure with additional options
fig = go.Figure(data=[go.Histogram(
    x=reviews["Age"],
    # Set number of bins
    nbinsx=20,
    opacity=0.7,
    # Set fill and line colors
    marker=dict(
        color='#ffbf00',  # Fill color
        line=dict(color='#f08080', width=3)  # Line color and width
    )
)])

# Update layout for better appearance, including figure size
fig.update_layout(
    title="Histogram of Age",
    xaxis_title="Age",
    yaxis_title="Frequency",
    bargap=0.1,  # Set gap between bars
    width=800,   # Set the width of the figure
    height=600   # Set the height of the figure
)

# Display the histogram
fig.show()
```