# Data Visualization with Python

Matt Steele

## Data Visualization Libraries

- MatPlotLib
- Seaborn
- Plotly
- GeoPandas
- O’Reilly Learning Platform

# Recap

- Functions and Arguments
- Variables
- Loading Libraries and Aliases
- Run Code

```python
import pandas as pd
```

# MatPlotLib

https://matplotlib.org/

**Overview**: A comprehensive library for creating static, animated, and interactive visualizations in

- Versatile Plotting: Supports various plot types, including line, bar, scatter, histogram, and more
- Customization: Offers extensive options for customizing colors, labels, titles, and styles.
- Integration: Works seamlessly with NumPy and Pandas for data manipulation and analysis.
- Interactivity: Capable of generating interactive plots for use in Jupyter notebooks and web app

## Load MatPlotLib Library

we are going to call the MatPlotLib package. Then we will load the data we want to plot and do a bit of cleanup

```python
# 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

# filter pollster to YouGov

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

## MatPlotLib: plot function

The plot() function in Matplotlib is a versatile method used to create a wide range of pl

- Data Input: Accepts data in various formats, including DataFrames, lists, and NumPy
- Automatic Axis Handling: Automatically assigns the x-axis and y-axis based on the D
- Default Plot Type: By default, it creates a line plot for numeric data. Different plot typ
- Customization Options: Allows customization of various plot elements, including titl

```
scotus.plot()
```

## Axis/Variable Handling

Custom Axis Assignment: You can explicitly define which columns to use for the x-axis and y-axis using the x and y parameters. This allows for greater flexibility in visualizing specific relationships within your data.

```
# set date and yes appoval

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



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

## Stylize the Graph

The plot() function provides several parameters that allow for customization of various plot elements, including color, line width, and more.

```
# 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
           )
```

## Add Labels

the plt object is part of the Matplotlib library’s pyplot module, which provides a collection of functions for creating static, interactive, and animated visualizations in Python.

```python
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("YouGov Approval Ratings Over Time")
plt.xlabel("Date")
plt.ylabel("Approval Ratings (%)")

## remove labels

plt.xlabel("")
```

## Adjusting Plot Elements with plt

you can fine-tune the plot’s appearance using various plt entries. These adjustments help improve readability and ensure the visual presentation aligns with the intended message.

## Adjusting Plot Elements with plt

```python
# 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("YouGov Approval Ratings Over Time", fontsize=26, color = "coral"
plt.xlabel("")
plt.ylabel("Approval Ratings (%)", fontweight = "bold")

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

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

## Adjusting Scales

In Matplotlib, you can adjust the scales of the axes to improve the clarity of your data visualization. Below is an example of how to set a continuous scale on the y-axis:

# Adjusting Scales

```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="
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)
```

# Add additional variables

```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 forma

# 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"
```

# Seaborn

https://seaborn.pydata.org/

**Overview:** A Python data visualization library based on Matplotlib, designed for statistical graphics and enhanc

**Key Features**: - **Built-in Themes**: Offers aesthetic themes to enhance the visual appeal of plots. - **Statistical Fur

# Load Seaborn

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


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

## Create a Bar Plot

```python
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()
```

# Set a Style

Seaborn offers several built-in themes to enhance the aesthetics of your visu

## Style Options

- `"darkgrid"`: Light background with gridlines, great for visibility.
- `"whitegrid"`: Similar to darkgrid but with a white background.
- `"dark"`: Dark background without gridlines.
- `"white"`: Clean white background, minimal distractions.
- `"ticks"`: Adds ticks to the axes for a more refined look.

## Set a Style

```python
# Set Seaborn style
sns.set_style("darkgrid")

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()
```

## Set a Palette

A well-chosen color palette can enhance the readability and aesthetic appeal of your visualizations. Seaborn supports various palettes, including those from Color Brewer, which are specifically designed for effective data visualization.

# Set a Palette

```python
# Set Seaborn style
sns.set_style("darkgrid")

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

# 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()
```

# Plotly

https://plotly.com/python/

**Overview:** Plotly is a powerful library for creating interactive visualizations in

## Key Features:

- **Interactive Visualizations**: Easily create plots that allow for zooming, pann
- **Versatile Plotting**: Supports various chart types, including line, bar, scatter
- **Customization**: Offers extensive options for customizing colors, labels, title
- **Integration**: Works seamlessly with Pandas and NumPy, allowing for smoo
- **Web Integration**: Built for the web, making it easy to embed visualizations

## Create a Histogram

```python
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

Adding bins to a histogram is crucial for visualizing the distribution of data. In Plotly, you can specify the number of bins to better understand the frequency of data points within specified ranges.

# Add Bins

```python
# 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

You can change the theme in Plotly by using the pl

1. **plotly**
   - The default Plotly theme with a classic look.
2. **ggplot2**
   - Inspired by the ggplot2 library, this theme pro
3. **seaborn**
   - Inspired by the Seaborn library, this theme em
4. **simple_white**

- A minimalist theme with a white background,

5. **presentation**

   - Designed for creating presentation-ready plot

6. **xgridoff**

   - A theme with grid lines removed, providing a

7. **ygridoff**

   - Similar to xgridoff but removes vertical grid lin

8. **plotly_white**

   - A theme with a white background and light gr

## Change theme

```python
import plotly.io as pio

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


# 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
    )
)])
```

# Size of the Graph

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

# 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"
```