Interactive Data Visualization with Matplotlib and Pandas

Your Name

August 26, 2025

1 Introduction

Data visualization is a crucial aspect of data analysis. While static plots are useful for presenting insights, interactive visualizations allow users to dynamically explore datasets. This report demonstrates how to create an interactive visualization project using **Matplotlib** for plotting and **Pandas** for data manipulation.

2 Methodology

2.1 Data Generation

Since no real-world dataset was provided, we generated synthetic stock price data for three companies (`AAPL`, `MSFT`, and `AMZN`) using a random walk process. This was achieved with NumPy's random number generator, then wrapped into a Pandas DataFrame with dates as the index.

2.2 Visualization Setup

An initial line plot was created to display stock prices over time. The `subplots_adjust` function was used to leave extra space at the bottom of the figure for interactive widgets.

2.3 Interactivity

Two types of interactivity were added:

- **Slider:** A slider was created to control the end date of the time series. Dragging the slider dynamically updates the plot to show data only up to the chosen date.
- **Buttons:** Three buttons allow switching between `AAPL`, `MSFT`, and `AMZN`. When a button is clicked, the plot updates to display the corresponding stock's price history.

3 Implementation

The following Python script implements the interactive visualization:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button

# Generate synthetic data
np.random.seed(42)
dates = pd.date_range(start="2023-01-01", periods=200)
df = pd.DataFrame({
    "AAPL": np.cumsum(np.random.randn(200)) + 150,
    "MSFT": np.cumsum(np.random.randn(200)) + 250,
    "AMZN": np.cumsum(np.random.randn(200)) + 100
}, index=dates)

# Initial plot
fig, ax = plt.subplots()
plt.subplots_adjust(bottom=0.3)
stock = "AAPL"
line, = ax.plot(df.index, df[stock], label=stock)
ax.set_title(f"{stock} Stock Price Over Time")
ax.legend()

# Slider
ax_slider = plt.axes([0.25, 0.15, 0.65, 0.03])
date_slider = Slider(ax_slider, "End Date", 0, len(df)-1,
                     valinit=len(df)-1, valstep=1)
def update(val):
    end = int(date_slider.val)
    line.set_ydata(df[stock].iloc[:end])
    line.set_xdata(df.index[:end])
    ax.relim()
    ax.autoscale_view()
    fig.canvas.draw_idle()
date_slider.on_changed(update)

# Buttons
ax_button1 = plt.axes([0.05, 0.7, 0.1, 0.05])
ax_button2 = plt.axes([0.05, 0.6, 0.1, 0.05])
ax_button3 = plt.axes([0.05, 0.5, 0.1, 0.05])
btn_aapl = Button(ax_button1, "AAPL")
btn_msft = Button(ax_button2, "MSFT")
btn_amzn = Button(ax_button3, "AMZN")

def plot_stock(event, ticker):
    global stock
    stock = ticker
    line.set_ydata(df[stock])
    line.set_xdata(df.index)
    ax.set_title(f"{stock} Stock Price Over Time")
    ax.legend([stock])
    fig.canvas.draw_idle()

btn_aapl.on_clicked(lambda event: plot_stock(event, "AAPL"))
btn_msft.on_clicked(lambda event: plot_stock(event, "MSFT"))
btn_amzn.on_clicked(lambda event: plot_stock(event, "AMZN"))

plt.show()
```

4 Results

The result is an interactive plot where:

- The slider dynamically changes the visible time range.
- The buttons switch between different stocks.

This approach makes exploratory data analysis more flexible and engaging.

5 Conclusion

This project illustrates how to combine Pandas and Matplotlib with widgets to create interactive data visualizations. Such tools are particularly useful in exploratory data analysis where users need to test different perspectives of the same dataset.