Changhyun Kim

Summary

The website content provides a comprehensive guide on using the

Visualizing Stock Market Data with YFinance and Python

The stock market generates a massive amount of data daily, and making sense of this data requires effective tools for analysis and visualization. The `yfinance` library in Python simplifies access to historical stock market data, while visualization libraries like Matplotlib, Plotly, and Seaborn allow you to create insightful charts to analyze trends and patterns. This blog will walk you through using `yfinance` to fetch stock market data and various techniques to visualize it effectively.

AI-generated image (author)

Yahoo Finance (YFinance)

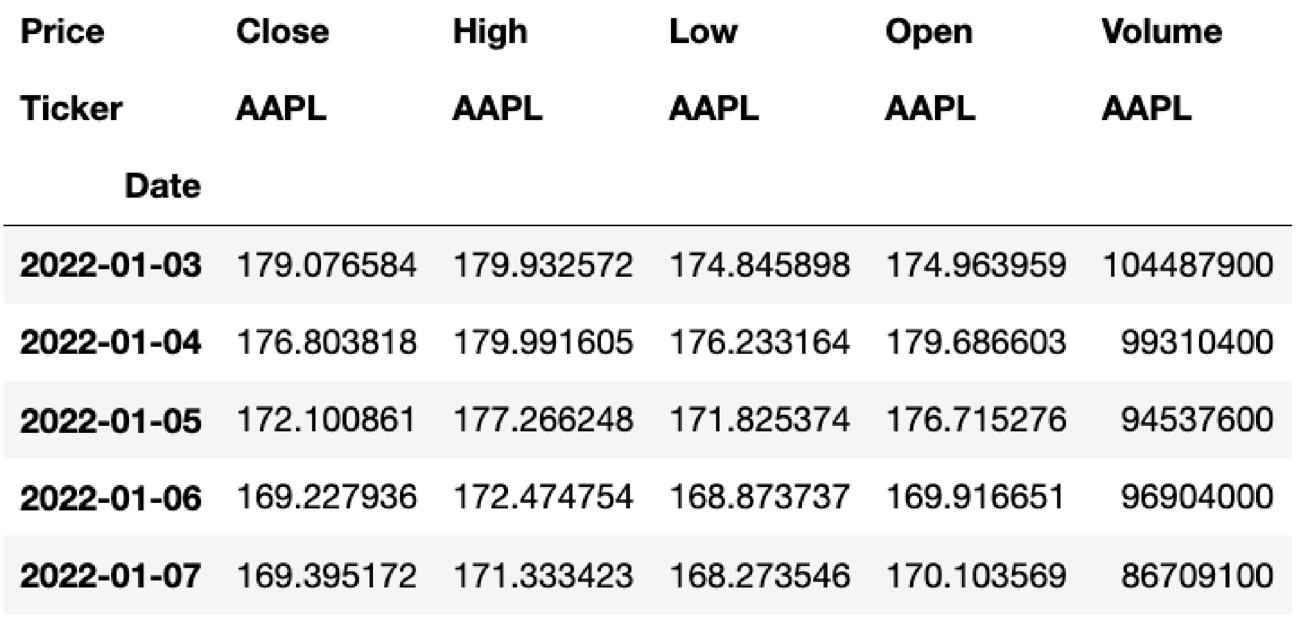
`yfinance` is a python library that simplifies the process of downloading historical stock market data from Yahoo Finance. It provides detailed data for stocks including historical data, adjusted close prices, volume and other key metrics.

If you don't have `yfinance` installed in your device, make sure to run the code below to install the library.

Once you have the library installed, you can retrive stock data only in a few lines of code. Let's fetch Apple's (ticker: AAPL) stock data for the year 2022, including open , high, low, close and volume metrics.

```
import yfinance as yf

# Download historical data for Apple
stock_data = yf.download("AAPL", start="2022-01-01", end="2023-01-01")
stock_data.head()
```

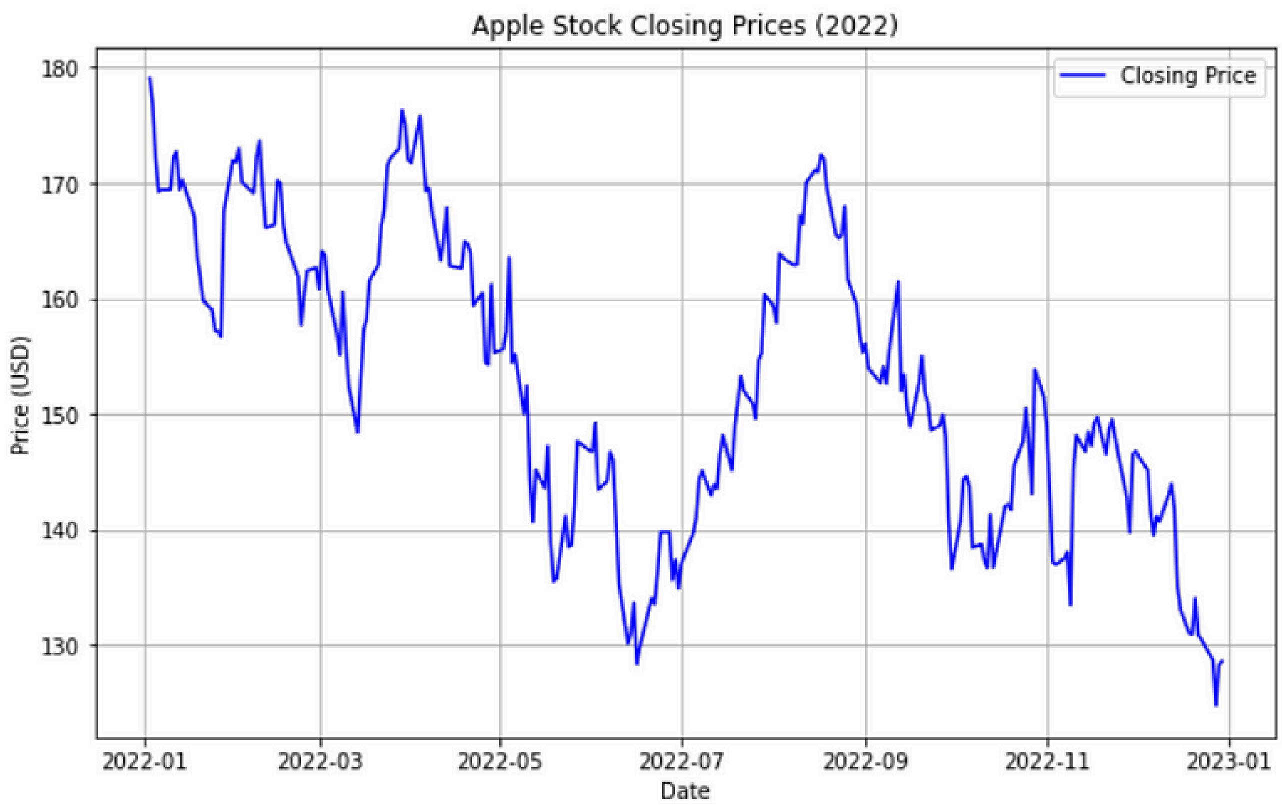


Price	Close	High	Low	Open	Volume
Ticker	AAPL	AAPL	AAPL	AAPL	AAPL
Date					
2022-01-03	179.076584	179.932572	174.845898	174.963959	104487900
2022-01-04	176.803818	179.991605	176.233164	179.686603	99310400
2022-01-05	172.100861	177.266248	171.825374	176.715276	94537600
2022-01-06	169.227936	172.474754	168.873737	169.916651	96904000
2022-01-07	169.395172	171.333423	168.273546	170.103569	86709100

Visualizing the Stock Data

- **Line plot for stock prices**

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plt.plot(stock_data['Close'], label='Closing Price', color='blue')
plt.title("Apple Stock Closing Prices (2022)")
plt.xlabel("Date")
plt.ylabel("Price (USD)")
plt.legend()
plt.grid()
plt.show()
```



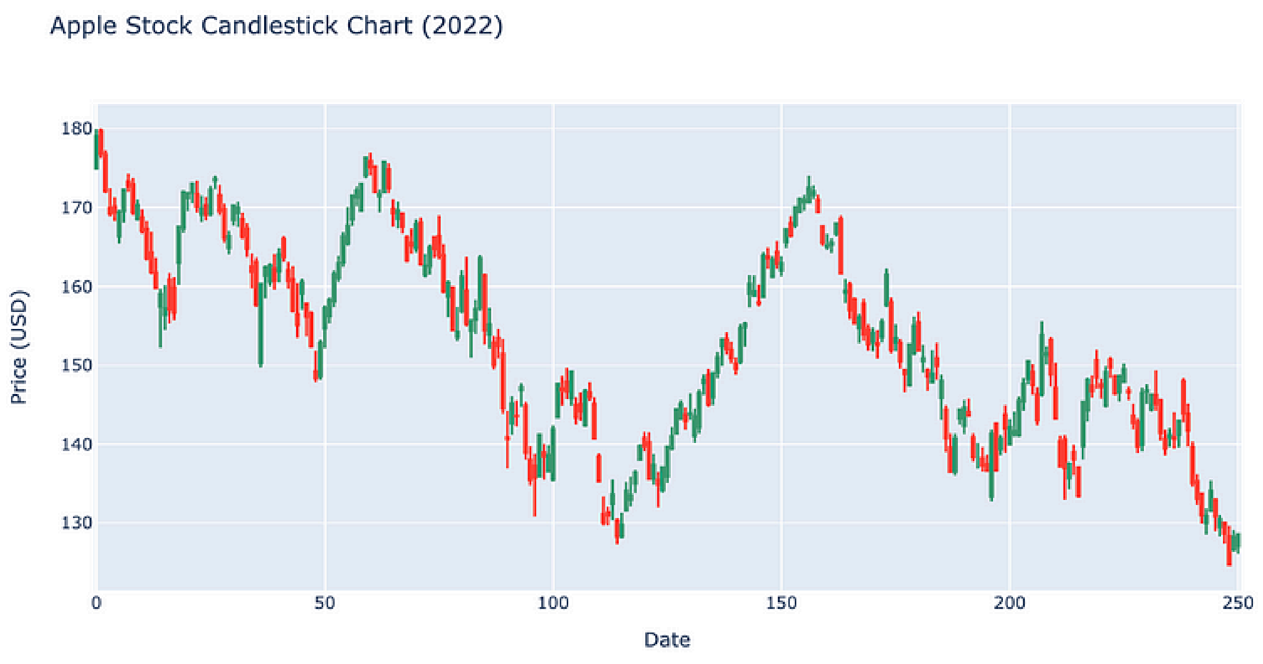
2. Candlestick chart with Plotly

Candlestick charts are popular for technical analysis and provide detailed insights into price movements.

```
stock_data = yf.download("AAPL", start="2022-01-01", end="2023-01-01")
stock_data = stock_data.stack().reset_index()

fig = go.Figure(
    data=[
        go.Candlestick(
            x=stock_data.index,
            open=stock_data['Open'],
            high=stock_data['High'],
            low=stock_data['Low'],
            close=stock_data['Close']
        )
    ]
)

fig.update_layout(
    title="Apple Stock Candlestick Chart (2022)",
    xaxis_title="Date",
    yaxis_title="Price (USD)",
    xaxis_rangeslider_visible=False
)
fig.show()
```



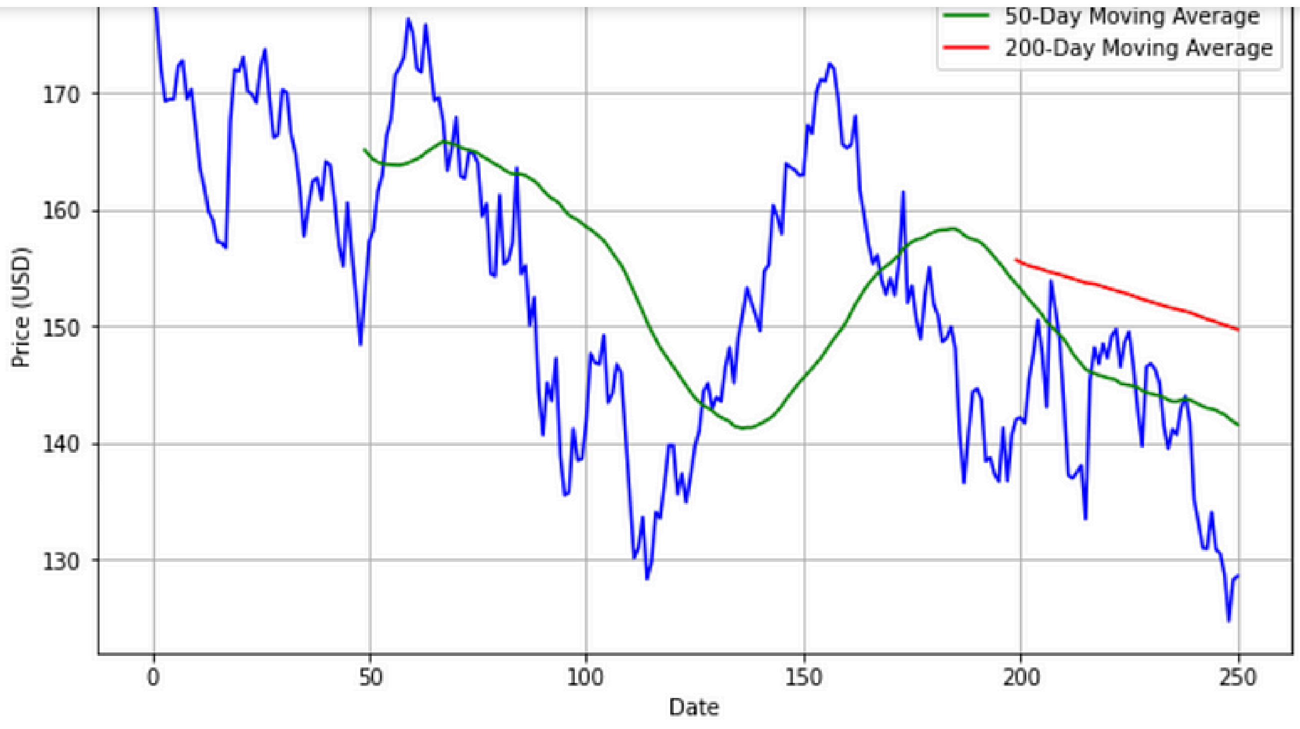
3. Moving averages (MA)

Moving averages smooth out price fluctions, making trends easier to be observed. You can you `rolling()` function in Python to calculate the moving averages depending on the number of days you would like to base the calculation on.

The example below shows both the 50-day and 200-day moving averages.

```
stock_data['50_MA'] = stock_data['Close'].rolling(window=50).mean()
stock_data['200_MA'] = stock_data['Close'].rolling(window=200).mean()

plt.figure(figsize=(10, 6))
plt.plot(stock_data['Close'], label='Closing Price', color='blue')
plt.plot(stock_data['50_MA'], label='50-Day Moving Average', color='green')
plt.plot(stock_data['200_MA'], label='200-Day Moving Average', color='red')
plt.title("Apple Stock Moving Averages (2022)")
plt.xlabel("Date")
plt.ylabel("Price (USD)")
plt.legend()
plt.grid()
plt.show()
```



Visualizing stock data with `yfinance` is an effective way to gain insights into market trends and trading activity. By combining yFinance with visualization libraries like Matplotlib and Plotly, you can create powerful tools for analyzing and presenting stock market data. Whether you're an investor, a trader, or a data enthusiast, mastering these techniques will enhance your ability to make informed decisions based on market data.

In addition to the data we explained here, you can also obtain financials data on a quarterly and annual basis, earnings data, stock splits and dividends-related data, sustainability scores, analysts' recommendations and so on. Obtaining these data using from `yfinance` and analyzing them through visualization will help in various dimensions. I will make sure to upload additional posts on `yfinance` library and how to use their data for analyzing stocks in the future!

Recommended from ReadMedium

Unicorn Day

Why Google Sheets Might Be Your Best Friend for Stock Data Analysis

Ever found yourself frustrated with unstable financial APIs? You're not alone… 🤔

5 min read

Kridtapon P.

Improve Your Trading Strategy with These 5 Rarely-Known Indicators

Learn how the Aroon Oscillator, Price Volume Trend, and others can guide your trading decisions

7 min read

Kevin Meneses González

This Is How I Scrape 99% of Websites (Step-by-Step Guide)

"Without data, you're just another person with an opinion." — W. Edwards Deming

3 min read

John Loewen, PhD

Which Python Dashboard Is Better? Dash, Panel And Streamlit Showdown

Prompting GPT-4 for multi-visual interactive dashboard creation

Ari Joury, PhD

Stop Copy-Pasting. Turn PDFs into Data in Seconds

Automate PDF extraction and get structured data instantly with Python's best tools

11 min read

PURRFECT SOFTWARE LIMITED

Top 12 Python GUI Libraries to Bring Your Interfaces to Life

Python GUI Libraries Made Easy: Find the Perfect Tool for Your Next Project

6 min read