What are the basics to automating and visualizing your company's data with Python?

Author: Vincent Zhang, *Senior Partner of IT & Data Analytics*

Advisor: Sooraj Ruparelia, *Managing Partner*
Advisor: Safwan Patel, *Managing Partner*

November, 2022

Abstract

Data analysis becomes increasingly important for companies in today's economy as more and more societal advances become deeply intertwined with data, such as machine learning, targeted advertising, and financial analysis. Producing hindsight, insight, and foresight based on big data introduces tedious challenges when analyzed manually. Python equips data analysts with the power to analyze big data with its gamut of libraries and automate any analytics processes in PowerBI and Tableau through its automation libraries. This report introduces the basics of fully automating stock chart visualization in Python.

Introduction

Two popular software that provides convenient and powerful insight generation through a Graphical User Interface (GUI) include PowerBI and Tableau. On the other hand, Python gives its users tremendous flexibility in customizing how analytics are generated. This report works through a beginner example of automated data analysis of stock data.

Libraries

We leverage the powerful libraries developed in Python to automate the analysis.

```
import yfinance as yf
import pandas
import matplotlib.pyplot as plt
from datetime import datetime
```

The first line imports "yfinance", a third-party library developed to extract stock data from Yahoo finance. The "pandas" library features a DataFrame object that stores data like an excel sheet. "matplotlib" helps us graph the data. "datetime" includes many convenient tools to manipulate date and time.

Industry Insight: your supervisor may task you with analyzing financial data at work which involves a multi-step process that includes retrieving data, cleaning data, and analyzing data; while the core task may be the analysis, collecting the correct data could become an arduous odyssey. Knowledge of existing libraries, such as "yfinance" for stocks, could improve efficiency at work by maximizing the time spent on the core steps of an assignment.

General Logic

Data needs to be processed before it can be graphed.

```
def main():
    data = init(['AMZN', 'TSLA', 'GOOG'], start='2020-01-01')
    plot(data, 'Open', "Opening Prices Stock Chart")
    plot(data, 'Volume', "Volume")
```

"init()" is a function we defined to prepare the data. The last two lines are simply plots of the selected data.

Industry Insight: when tasked with a Python project at work, structuring code in clear steps makes it easy for others to understand the overall procedure even without programming experience. From the "main()" function above, it is clear that in our stock visualizer, we perform three steps: initialize the data, plot the opening prices, and plot the volume.

Preparing the Data

```
def init(stocks, start='2021-01-01', end=datetime.today().strftime('%Y-%m-%d')):
    return yf.download(" ".join(stocks), start, end)
```

The “init” function takes in a “stocks” parameter, which is a list of tickers the user wants to find the stock data for. “start” and “end” are dates for which we want the stock data. They are defined in such a way that if “start” or “end” are not provided at the function call, the default values defined in the function will be used instead. The “end” parameter has a default value set to today’s date to ensure the data displayed is up-to-date.

“init()” returns a “pandas” DataFrame whose columns consist of statistics like high, low, open, and close, each of which contains another layer of headings for stock tickers. This DataFrame uses the date as its row index.

Industry Insight: your employer may provide generic instructions and leave implementation details at your discretion. Setting default parameter values enables flexibility in implementation. We can easily change start and end dates without having to change the function implementation, and at the same time, we have the option to leave the dates unspecified.

Graphing the Data

```
def plot(data, stat, title=""):
    for ticker in data[stat]:
        data[stat][ticker].plot(label=ticker, figsize=(16,7))
    plt.title(title)
    plt.legend()
    plt.show()
```

The “plot()” function generalizes the graphing process for data within the “pandas” DataFrame “data” that holds the statistics we are looking for, e.g., daily low, as well as the title of the graph. We plot every ticker within the “stat” category to the graph and show it.

Graphed Results

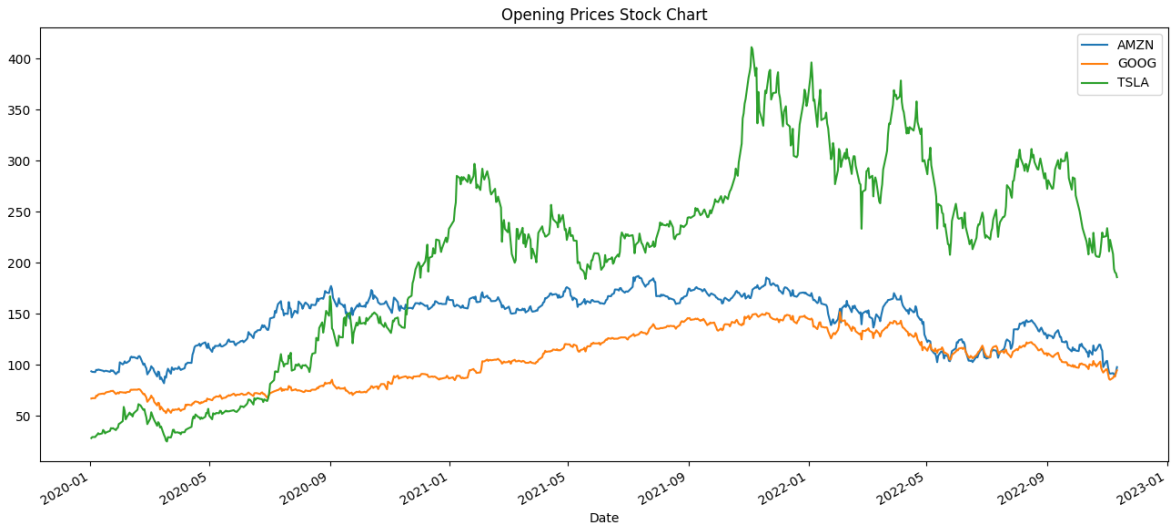

Figure 1: graph of the opening prices of AMZN, GOOG, and TSLA stocks from 2020.

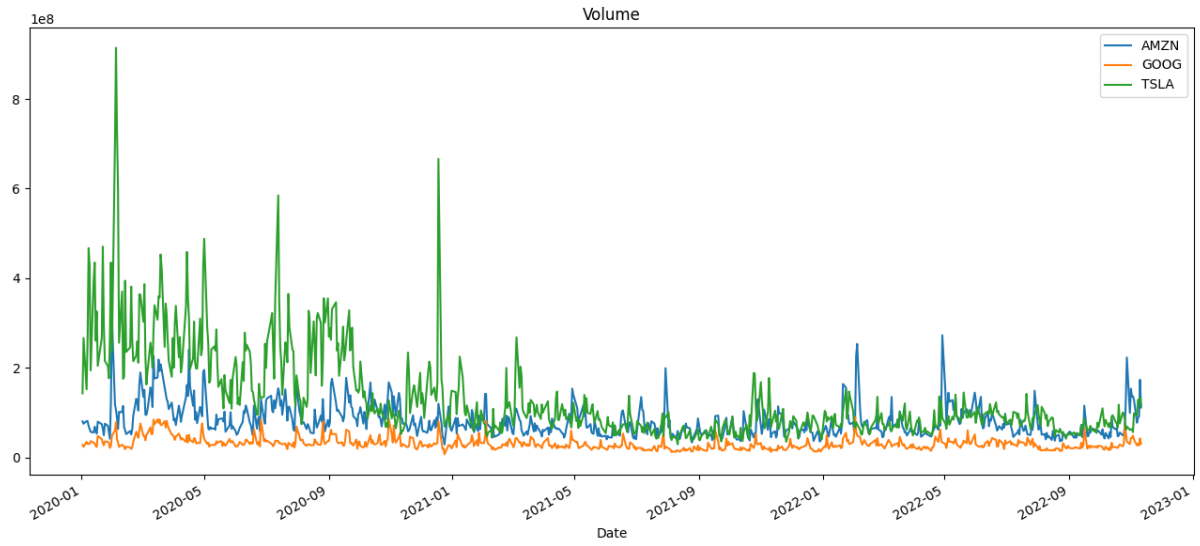

Figure 2: graph of the volume of AMZN, GOOG, and TSLA stocks from 2020.

Industry Insight: your supervisor may task you with creating graphs or charts of data. Finding the right balance between information and clarity ensures that the salient aspects are complemented and highlighted while clutter is avoided. In our stocks chart, the most important details are the title, legend, and axis labels.

Benefits of this Python Implementation

With generalized functions, it is easy to view stock data for other companies by simply updating the call to "init()" in the "main()" function. Moreover, the clarity of this implementation allows for the code to be easily extended for enhanced functionality and abstraction.

Benefits of Analyzing Data with Python

Python's wealth of libraries and its simplicity in programming facilitate monumental customizations to a company's specific analytics needs. Even if some analytics cannot be fulfilled by Python, its automation libraries allow for automated analysis on GUI analytics software like PowerBI and Tableau.

Conclusion

This short introduction to automated analysis of data with Python shows the power and value Python data analysis can bring to your company.

(See Appendix A for the full program and installation instructions.)

Appendix A – Full Code & installation Guide

Full Code

```
import yfinance as yf
import pandas
import matplotlib.pyplot as plt
from datetime import datetime

def init(stocks, start='2021-01-01', end=datetime.today().strftime('%Y-%m-%d')):
    return yf.download(" ".join(stocks), start, end)

def plot(data, stat, title=""):
    for ticker in data[stat]:
        data[stat][ticker].plot(label=ticker, figsize=(16,7))
    plt.title(title)
    plt.legend()
    plt.show()

def main():
    data = init(['AMZN', 'TSLA', 'GOOG'], start='2020-01-01')
    plot(data, 'Open', "Opening Prices Stock Chart")
    plot(data, 'Volume', "Volume")

if __name__=="__main__":main()
```

Installing Python

Visit the following links according to your operating system and download a suitable installer for your machine. For Windows user, ensure to select “add to PATH” at end of the installation.
Windows: https://www.python.org/downloads/windows/
Linux/Unix: https://www.python.org/downloads/source/
macOS: https://www.python.org/downloads/macos/

Package Installations

For any package installations, run the following command in a terminal or command prompt.
`python -m pip install <package name>`
For example, to install “matplotlib” used in the code above, run the following command.
`python -m pip install matplotlib`

Running the Program

After saving the full code in a “.py” file, open a terminal and use the “cd” command to navigate to the directory where the file is saved and run the command, `python filename.py`

Installation Alternative

To run the program with the installations done for you, visit an online python complier such as “Replit” which will automatically install packages when you run a python program after you sign up for an account.