How you can Create Beautiful Cryptocurrency Graphs in Python

Kush Follow
Apr 20, 2021 · 6 min read

As cryptocurrencies are beginning to being brought into our society as a norm, more and more people are interested in them. People are trading with them, investing in them and even using them to create further products. In today's article I will show you how you can plot beautiful graphs using Plotly to display critical price data. We will be plotting two graphs: one of a simple candlestick chart and 2 simple moving averages and the other of 4 different cryptocurrencies to see how they correlate with each other:

1. Plotting a candlestick chart for Bitcoin
2. Plotting multiple cryptocurrency graphs

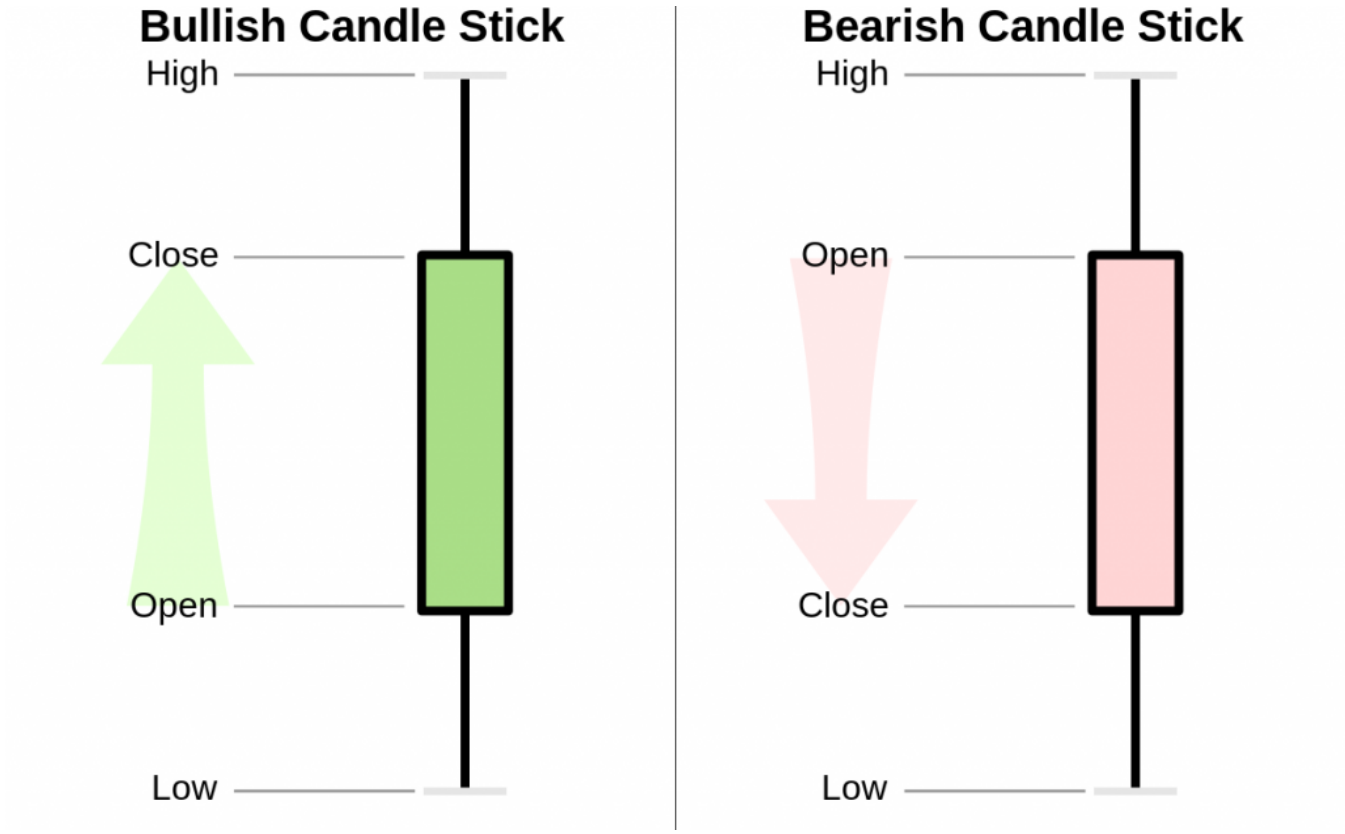
Before we begin, there are a few terms we need to get familiar with:

What is a Simple Moving Average?

Simple moving average is the value which can be used to show the current trend in volatile data and is calculated by dividing the sum of the recent prices by the number of time periods for which we calculated the sum. Short term averages, those with shorter time periods, react quicker to changes in the market. Long term averages react much slower, but show a more generalised direction of travel for the market.

What is a Candlestick chart?

A candlestick chart is a type of financial chart which can provide a lot of information about price movements, such as the opening, closing, highest, and lowest prices of a certain security in a specific time period.



https://www.newtraderu.com/2020/06/01/how-to-read-candlestick-charts/

Here is a very good article about understanding the candlesticks and how to read their patterns, if you want more information.

We also need to install the 3 different libraries we will be using today. This can be done in one simple command:

```
pip install pandas pandas-datareader plotly
```

Now let's get started, shall we!

Plotting a candlestick chart for Bitcoin:

To plot a candlestick chart, we can first import the required libraries into our file and establish the cryptocurrency and currency we are going to look at:

```
1  from datetime import datetime, timedelta
2
3  import pandas as pd
4  import pandas_datareader as pdr
5  import plotly.graph_objects as go
6
7  CRYPTO = 'BTC'
```

```
8   CURRENCY = 'GBP'
```

single_plot.py hosted with ❤ by GitHub view raw

You can obviously change the currency to your local currency if needed.

The first thing we need to do is to create a function which we can call to fetch the data. For this we will be using the *pandas_datareader* library to collect the data from Yahoo Finance and return it to us in a pandas dataframe object:

```
1   def getData(cryptocurrency):
2       now = datetime.now()
3       current_date = now.strftime("%Y-%m-%d")
4       last_year_date = (now - timedelta(days=365)).strftime("%Y-%m-%d")
5
6       start = pd.to_datetime(last_year_date)
7       end = pd.to_datetime(current_date)
8
9       data = pdr.get_data_yahoo(f'{cryptocurrency}-{CURRENCY}', start, end)
10
11      return data
```

single_plot.py hosted with ❤ by GitHub view raw

In the first few lines of the getData function, we calculate the string of the date for today and for exactly 1 year ago in the format `YYYY-MM-DD` (this is required for the get_data_yahoo method. We can then use the two dates to fetch the historical data for our cryptocurrency from Yahoo Finance via the pandas_datareader library and return it.

In our main code we will create the main chart and plot the data on it:

```
1   if __name__ == '__main__':
2       crypto_data = getData(CRYPTO)
3
4       # Candlestick
5       fig = go.Figure(
6           data = [
7               go.Candlestick(
8                   x = crypto_data.index,
9                   open = crypto_data.Open,
```

```
                high = crypto_data.High,
                low = crypto_data.Low,
                close = crypto_data.Close
            ),
            go.Scatter(
                x = crypto_data.index,
                y = crypto_data.Close.rolling(window=20).mean(),
                mode = 'lines',
                name = '20SMA',
                line = {'color': '#ff006a'}
            ),
            go.Scatter(
                x = crypto_data.index,
                y = crypto_data.Close.rolling(window=50).mean(),
                mode = 'lines',
                name = '50SMA',
                line = {'color': '#1900ff'}
            )
        ]
    )
```

single_plot.py hosted with ❤ by GitHub view raw

We can get the cryptocurrency data from the function we created just now and use it to create a candlestick chart. In the candlestick chart there are 5 required parameters which are all self-explanatory, apart from the *x* value which is linked to the date (indicated by the index). Notice here we also add 2 scatter graphs on the same figure. These will be our simple moving/rolling averages and are calculated on the closing prices of our data. I also chose two colours (pink and blue) to differentiate each line from the other, although there will be a legend with the graph so you can check which is which. To change the time period for the averages, you can alter the integer value of 'window' in the rolling method.

```
fig.update_layout(
    title = f'The Candlestick graph for {CRYPTO}',
    xaxis_title = 'Date',
    yaxis_title = f'Price ({CURRENCY})',
    xaxis_rangeslider_visible = False
)
fig.update_yaxes(tickprefix='£')
```

```
fig.update_yaxes(tickprefix="£")

fig.show()
```

single_plot.py hosted with ❤ by GitHub view raw

We can then add a title to the graph and axis and remove the range slider at the bottom of the graph as this is not very useful. Lastly, before we show the graph we add the £ symbol (change this to whatever currency symbol you want) as a prefix to the price data on the *y* axis.

When running the program, we should see our browser open, and the graph appear on screen as such:

The candlestick graph for Bitcoin

We can use the buttons on the top right of the graph to zoom in and out, pan around the graph and even save the graph as an image.

Plotting multiple cryptocurrency graphs:

We will plot multiple cryptocurrency graphs on the same chart as simple line graphs, this is because many candlestick graphs can get quite confusing at times. From this plot we can see how different cryptocurrencies correlate with each other.

To start off, we import the same libraries as before and define our constants again. This time we define them as a list of cryptocurrencies as we will be plotting these. We can

also copy over the same getData function as before:

```python
from datetime import datetime, timedelta

import pandas as pd
import pandas_datareader as pdr
import plotly.graph_objects as go

CRYPTOS = ['BTC', 'ETH', 'LTC', 'XRP']
CURRENCY = 'GBP'

def getData(cryptocurrency):
    now = datetime.now()
    current_date = now.strftime("%Y-%m-%d")
    last_year_date = (now - timedelta(days=365)).strftime("%Y-%m-%d")

    start = pd.to_datetime(last_year_date)
    end = pd.to_datetime(current_date)

    data = pdr.get_data_yahoo(f'{cryptocurrency}-{CURRENCY}', start, end)

    return data
```

multi_plot.py hosted with ❤ by GitHub view raw

In our main section of code, we can then get the data and create a dictionary with it, the key being the name of the cryptocurrency and the value being the pandas dataframe returned from the getData function:

```python
crypto_data = {crypto:getData(crypto) for crypto in CRYPTOS}

# crypto_data = dict()
# for crypto in CRYPTOS:
#     crypto_data[crypto] = getData(crypto)
```

multi_plot.py hosted with ❤ by GitHub view raw

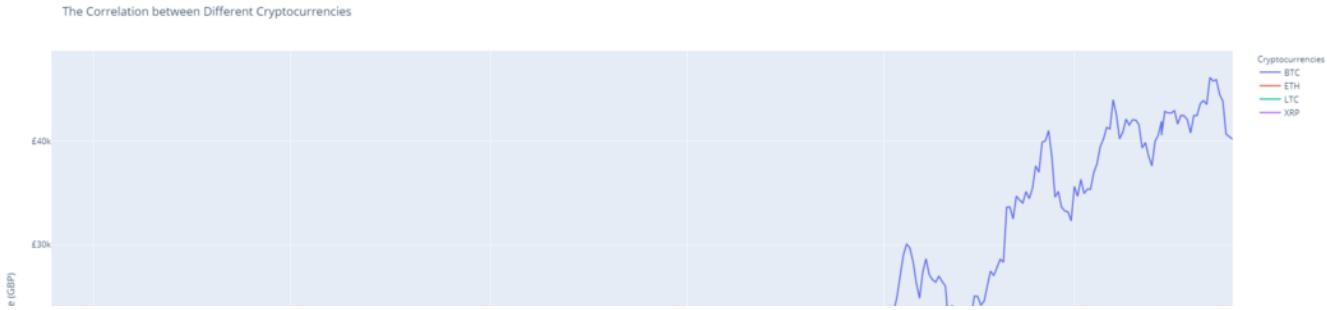
This syntax might look a bit strange to you, but all it is doing is executing the same code as the 3 commented lines below. The technique used here is called dictionary comprehension and if you want to learn more, you can head over here.

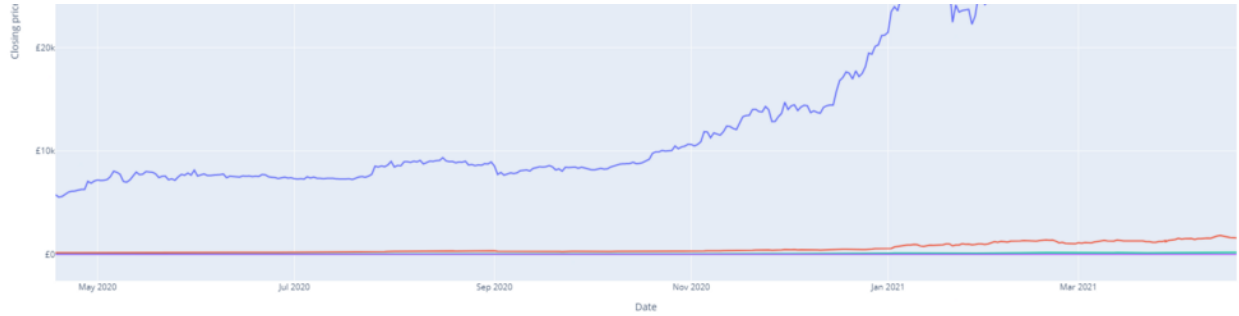
To plot our multiple graphs, we can make a new figure as we did before, but this time iterate through each item in our dictionary and add a 'trace' (essentially the data) to our chart. Here we will be using a scatter graph to plot the closing price of our data to simplify things (a lot of candlesticks on one chart can be confusing):

```
fig = go.Figure()

    # Scatter
    for idx, name in enumerate(crypto_data):
        fig = fig.add_trace(
            go.Scatter(
                x = crypto_data[name].index,
                y = crypto_data[name].Close,
                name = name,
            )
        )

    fig.update_layout(
        title = 'The Correlation between Different Cryptocurrencies',
        xaxis_title = 'Date',
        yaxis_title = f'Closing price ({CURRENCY})',
        legend_title = 'Cryptocurrencies'
    )
    fig.update_yaxes(type='log', tickprefix='£')

    fig.show()
```

multi_plot.py hosted with ❤ by GitHub view raw

We also can add titles to our axis, legend, and chart. Here we also update our *y* axis to make it a logarithmic scale. This is because the price of BTC is much higher compared to the rest, and if we have a linear scale then this will look very odd, as we would see the other cryptocurrency graphs as straight lines:

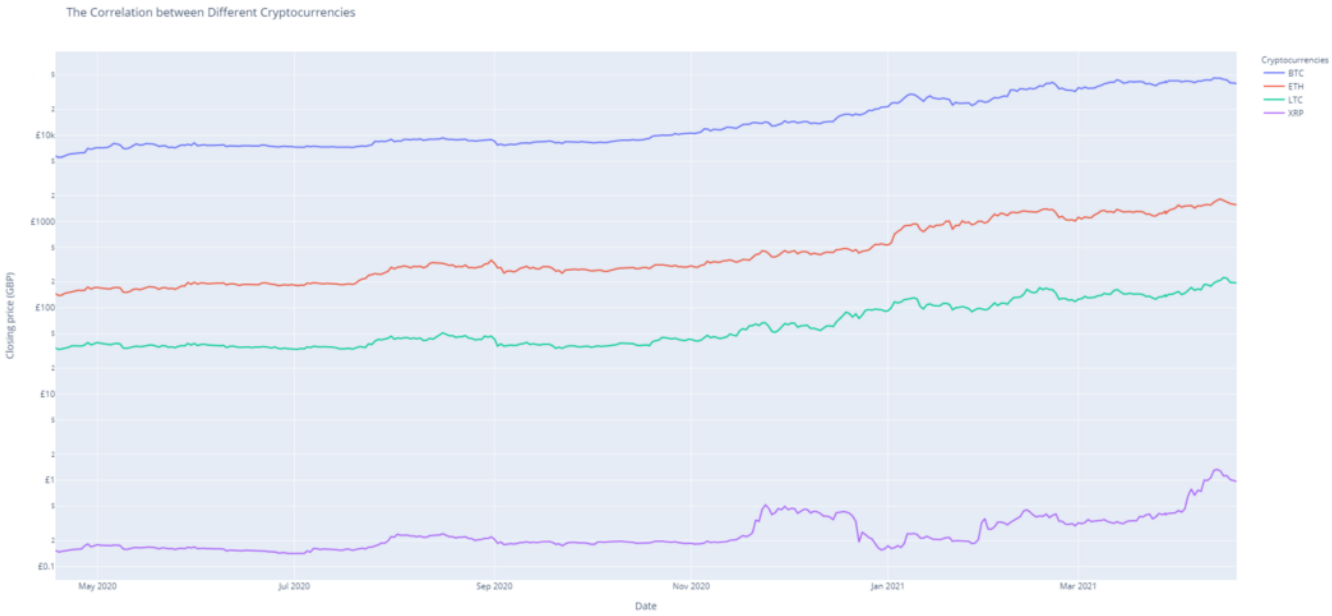




An example correlation graph with a linear y axis

Note that due to this change for the *y* axis, to check the price of a cryptocurrency you will have to pay closer attention to the graph.

After running the program with the *y* axis set to being logarithmic, you should hopefully get a graph looking like this:



Multiple cryptocurrency graphs shown next to each other to compare correlation

As you can see from the graph that BTC, ETH and LTC all have very similar trends. But ETH and LTC are much more strongly positively correlated than BTC and ETH, you can see this by looking at the patterns in the different graphs and see how they change in comparison to each other. On the other hand, you can see that BTC and XMR are negatively correlated as XMR spikes at around the 25th of November but BTC dropped slightly then. This can be quite useful when trading as you can see which cryptocurrencies are correlated with each other and you can choose them to invest in.

Final thoughts

When I first learned about Plotly, I never imagined it to be this simple and easy to plot graphs, but not just normal graphs, beautifully precise graphs with so many other features. Note: please don't take any of what I say as financial advice, I'm not an economist, just a programmer.

I hope I taught you something new. If you have any questions or feedback, please don't hesitate to comment below and I'll endeavour to answer them.

Thank you for reading! 💖

Python Programming Cryptocurrency