# Can machine learning help build a better stock portfolio?

Analysis of stock price charts to quantify correlations for portfolio optimisation (with Python code & data).

JP Hwang May 7 · 8 min read ★

Making sense of stock price charts

Machine learning has conquered many fronts in the last decade, from Netflix recommendations, medical diagnoses, or even developing pharmaceuticals.

Naturally, many looked to apply this new tool to financial markets, which ironically showcased clear limitations of machine learning. Although many have attempted to predict future prices of stocks, that has proven to be quite elusive for most as it is tantamount to fortune telling.

The most common ingredient in successful machine learning applications is extremely clever *pattern recognition*.

They predict the next words in your email *based o*n the first few words, approximate housing prices *based on* others in the neighbourhood, or provide medical diagnosis *based o*n an X-ray image. All enabled from analysing a litany of prior, similar or related observations.

How would we apply this superpower to the markets?

## What about portfolio construction?

Portfolios are built based on desired characteristics or outcomes. For example, you might build a portfolio designed to track the entire market or specific sectors. Or, your goal might be to build a diversified portfolio that would be resistant to fortunes in particular industries, by finding unrelated stocks or even those that move in opposite directions.

> We can use patterns hidden in the past to build better stock portfolios.

In other words, you might look for stocks that are well-correlated, not correlated, of inversely correlated.

This is seemingly a job perfectly suited for machine learning. In this article, I explore the task of comparing stock behaviours, identifying correlations and grouping, and using this information to build portfolios.

## Before we get started

`Packages`

I assume you're familiar with python. Even if you're relatively new, this tutorial shouldn't be too tricky, though.

To follow along, you will need `pandas` , `scipy` , `sklearn` and `plotly` . Install each (in your virtual environment) with a simple `pip install [PACKAGE_NAME]` .

### Data

I include the code and data in my GitLab repo here, so you should be able to easily follow along by downloading/cloning the repo if you wish. The directory name is `stock_correlations` .

## Building A Stock Portfolio

### What are we looking for?

For this project, we would like to be able to determine correlation of one stock to another. While there are numerous indicators of correlation, and even some factors such as primary industries which might be indicative of causality, let's choose the simplest measure for now — *price*.

We can get stock market data using an API (I use Tiingo, but you can use any others, such as those I discussed in another article here). Here's a sample of the code that I use:

```
# ========== FUNCTION TO GET STOCK DATA ==========
def get_stock_data(tkn, sym='amzn', start_date='2020-01-01'):
    headers = {'Content-Type': 'application/json'}

    requestResponse = requests.get("https://api.tiingo.com/tiingo/daily/" + sym + "/prices?start
    if requestResponse.status_code == 200:
        logger.info(f'Success fetching {sym} data from {start_date} to today')
    else:
        logger.warning(f'Something looks wrong - status code {requestResponse.status_code}')

    return requestResponse

# ========== GET STOCK DATA ==========
symbols_ser = ['AAPL', 'MSFT', 'JNJ', 'WMT', 'TSM', 'XOM']  # For testing

ticker_datas = []
for sym in symbols_ser:
    temp_data = get_stock_data(tiingo_token, sym=sym, start_date='2015-01-01').json()
    temp_df = pd.DataFrame(temp_data)
    temp_df['sym'] = sym
    ticker_datas.append(temp_df)
```

```
# Concatenate stock data
total_ticker_df = pd.concat(ticker_datas)
```

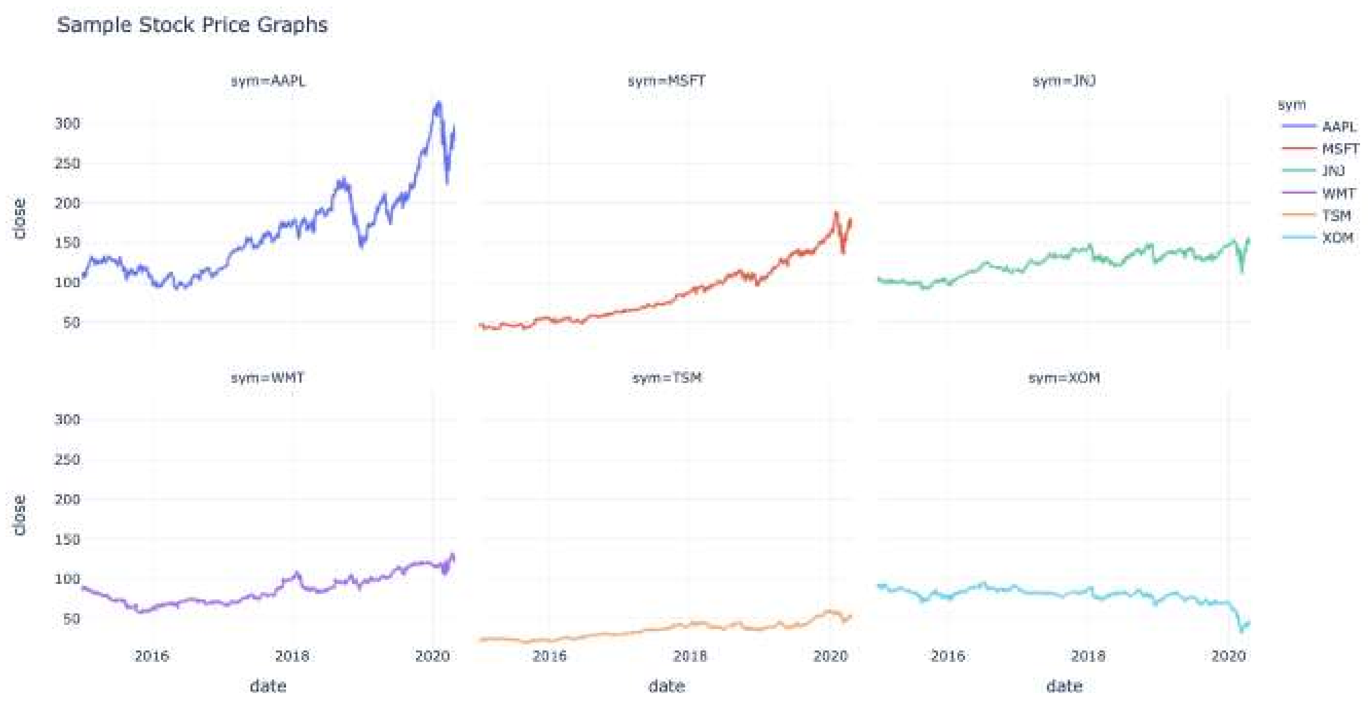
get_stock_data_function.py hosted with ♡ by GitHub view raw

Now that we have the raw data, what we are aiming to do is to explore correlation of each stock price data with another. Intuitively speaking, the exercise can be described as determining 'similarity' of shapes of each stock price chart.

Let's plot some of these shapes to visualise what we mean:

```
fig = px.scatter(total_ticker_df,
                 x='date', y='close', color='sym', template='plotly_white', title='Sample Stock P
                 facet_col='sym', facet_col_wrap=3
                 )
fig.update_traces(mode='lines')
fig.show()
```

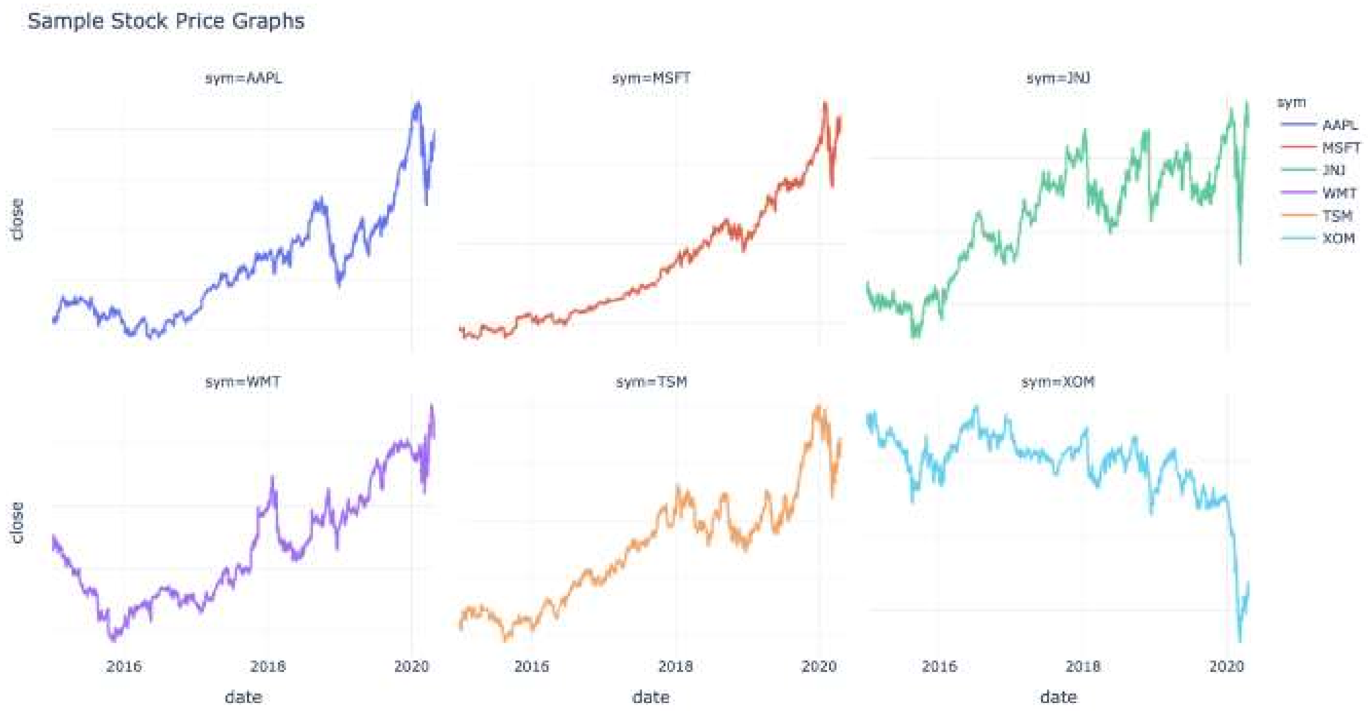
get_stock_data_plot1.py hosted with ♡ by GitHub view raw



Selection of stock price charts — do some of these look correlated?

Some of these look more correlated (i.e. 'similar') than others. Actually, for this kind of correlation analysis, the absolute scaling of the y-axis doesn't matter so much. So we can update the graph to ignore the y values to be limited to individual subplots, essentially normalising them (at least, visually).

```
fig.update_yaxes(matches=None, showticklabels=False)
```

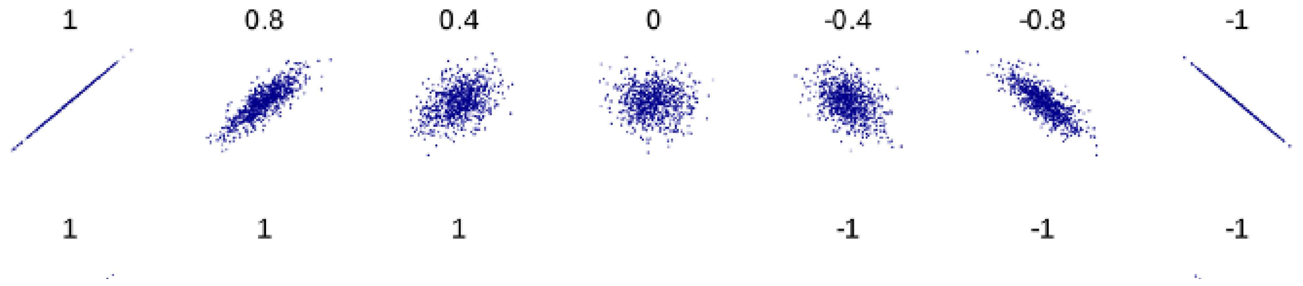


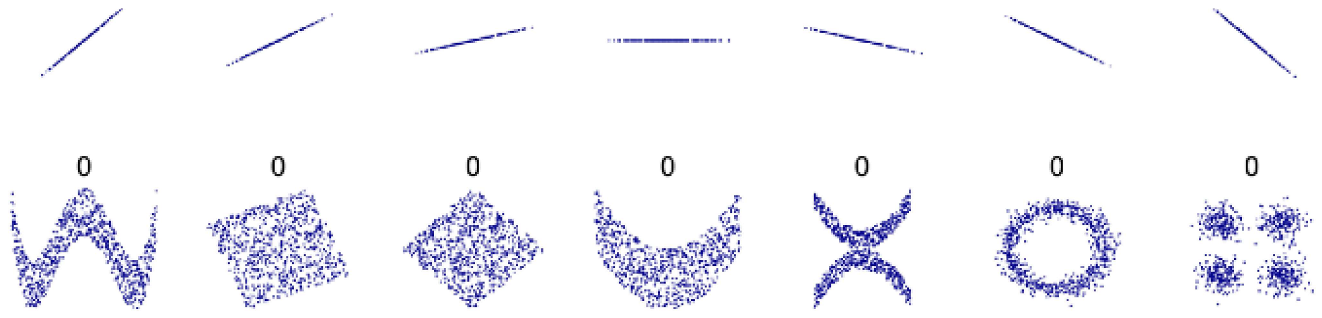
Selection of stock price charts — easier to see rough 'similarity'

Great! We now have something to use as a benchmark. But, how would we *quantify* correlation between each price chart?

## Correlation calculation

For a spatial distribution, we know that the distribution shape vs correlation value look like this.





Correlation examples (https://en.wikipedia.org/wiki/Correlation_and_dependence)

Determining correlation in stock prices is a different problem.

In a spatial distribution, the correlation is analysed between data in one dimension (X) against another (Y). On the other hand, each stock price data is already two-dimensional, as it charts price (Y) against time (t). In addition to that, we want to develop a method that will be used to compare potentially hundreds or thousands of stocks, all with different value ranges.

For now, let's keep it simple and use Pearson correlation coefficients as our measure of similarity. It is a relatively crude measure, but it will get the job done for what we are after, which is to demonstrate the concept.

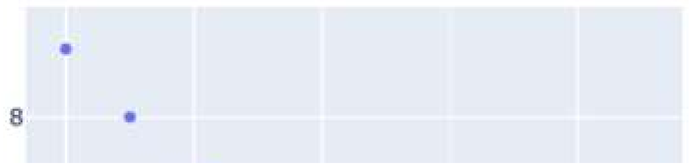
How do we determine these values? Scipy has a built-in Pearson correlation function `scipy.stats.pearsonr()` to call upon.

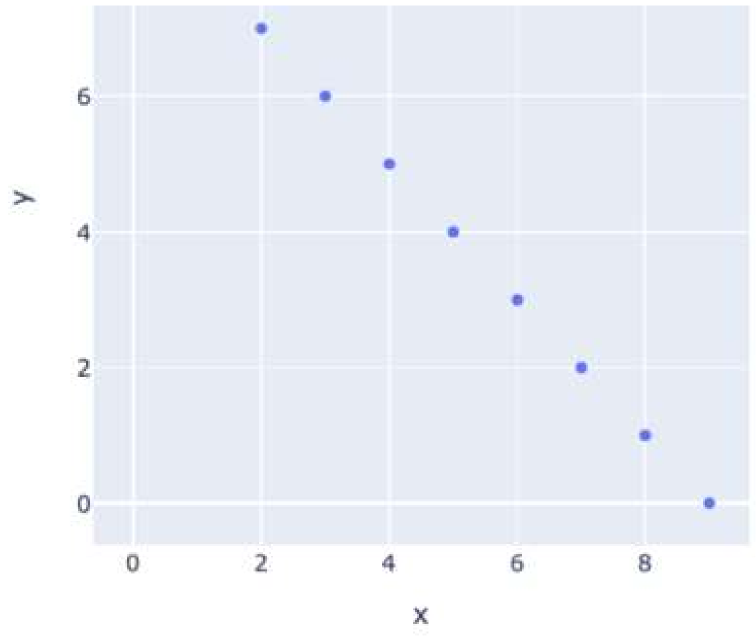
The syntax to use this function is to simply call it with two inputs, like:

```
scipy.stats.pearsonr(x, y)
```

Where *x and y* can be scalars *or* vectors.

For example, we can compare a list with its exact inverse — something that would look like this:





Perfect Inverse Correlation

```
tmplist = list(range(10))
scipy.stats.pearsonr(tmplist, tmplist[::-1])
```

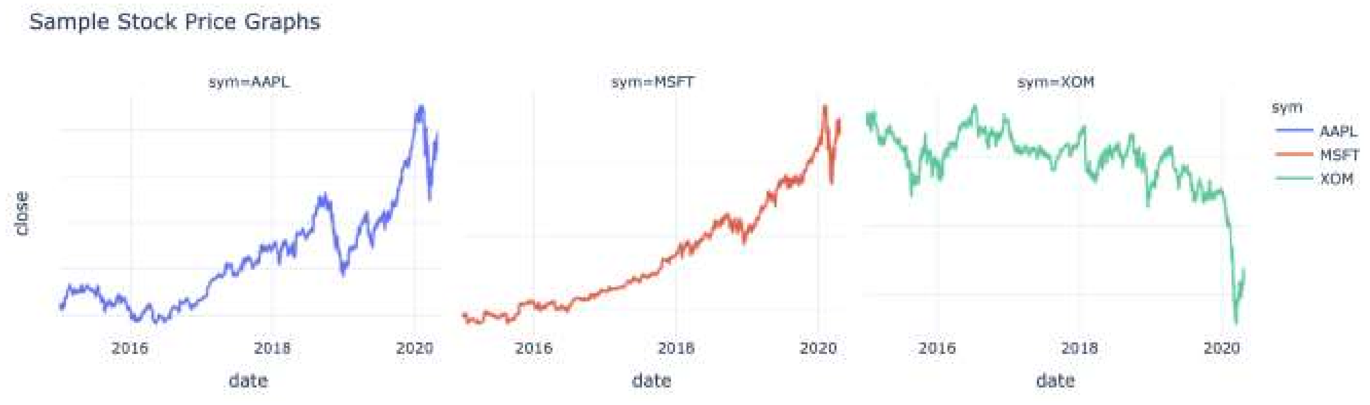
Which, as expected, produces outputs (-1, 0) where -1 is the correlation (perfectly inverse), and 0 is the p-value.

Our stock data from the dataframe `total_ticker_df` can be compared using similar syntax. For given `i` and `j` values:

```
ser_i = total_ticker_df[total_ticker_df.sym == symbols_ser[i]]
['close'].values
ser_j = total_ticker_df[total_ticker_df.sym == symbols_ser[j]]
['close'].values
r_ij, p_ij = scipy.stats.pearsonr(ser_i, ser_j)
```

Comparing the data for Apple and Microsoft produces an extremely high correlation of 0.947, indicating their similar fortunes for the last 5 years as a tech conglomerate. On the other hand, comparing Apple and Exxon Mobile data produces a high negative correlation of -0.687 showing that their fortunes were quite the opposite to each other.

An inspection of the stock price charts visually bear this out:



Similar / Reversals of fortunes

These figures can be put together into a matrix of correlations, as such:

```python
# Build a correlation (and p-value) matrix
r_array = np.zeros([len(symbols_ser), len(symbols_ser)])
p_array = np.zeros([len(symbols_ser), len(symbols_ser)])
for i in range(len(symbols_ser)):
    for j in range(len(symbols_ser)):
        ser_i = total_ticker_df[total_ticker_df.sym == symbols_ser[i]]['close'].values
        ser_j = total_ticker_df[total_ticker_df.sym == symbols_ser[j]]['close'].values
        r_ij, p_ij = scipy.stats.pearsonr(ser_i, ser_j)
        r_array[i, j] = r_ij
        p_array[i, j] = p_ij
```

get_stock_data_corr_matrix_simple.py hosted with ♡ by GitHub view raw

Producing:

```
>>> print(r_array)
[[ 1.          0.94783247  0.75459535  0.89186946  0.93008958 -0.68727618]
 [ 0.94783247  1.          0.79397608  0.91511258  0.92275757 -0.71430629]
 [ 0.75459535  0.79397608  1.          0.74710565  0.87785573 -0.3759863 ]
 [ 0.89186946  0.91511258  0.74710565  1.          0.86919002 -0.61902468]
 [ 0.93008958  0.92275757  0.87785573  0.86919002  1.         -0.5900107 ]
 [-0.68727618 -0.71430629 -0.3759863  -0.61902468 -0.5900107   1.        ]]
```

get_stock_data_r_array.py hosted with ♡ by GitHub view raw

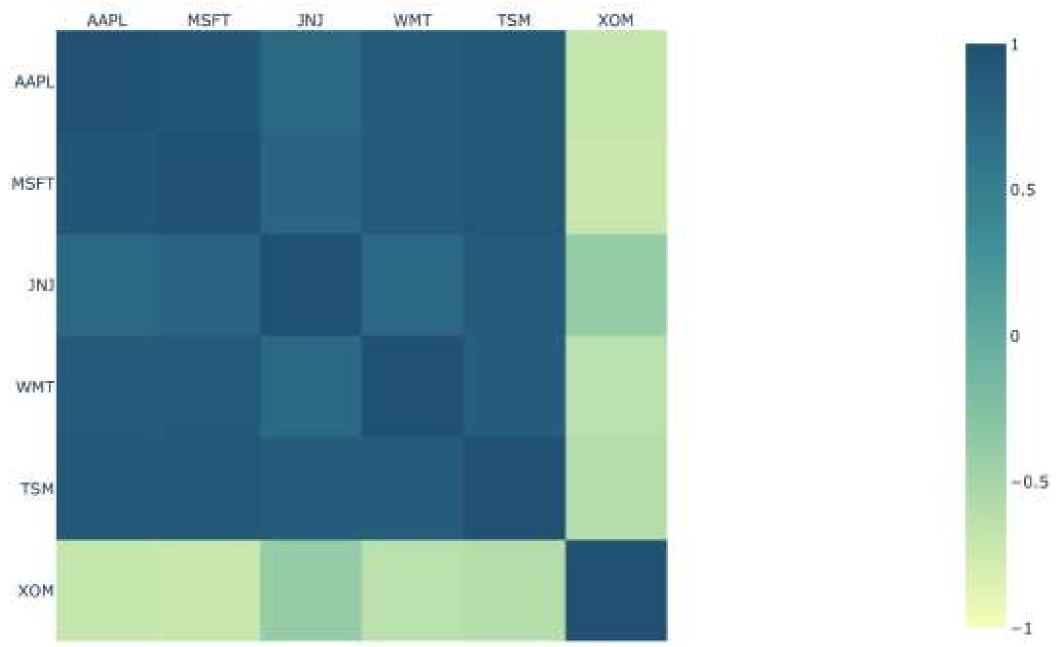
This can be visualised as a heatmap with Plotly, with this snippet:

```python
fig = px.imshow(r_array, x=symbols_ser, y=symbols_ser,
                color_continuous_scale=px.colors.sequential.Bluyl, color_continuous_midpoint=0
```

```
                    )
fig.update_xaxes(side="top")
fig.show()
```

get_stock_data_heatmap_simple.py hosted with ♡ by GitHub view raw

To produce:

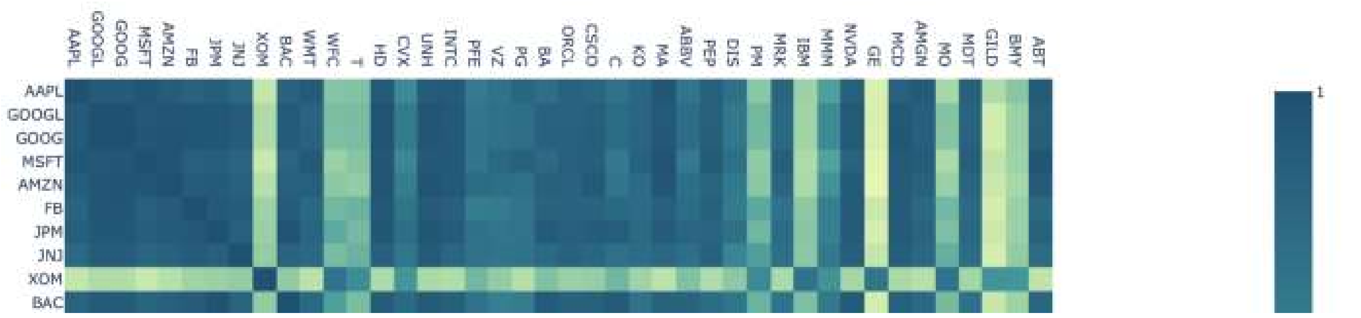


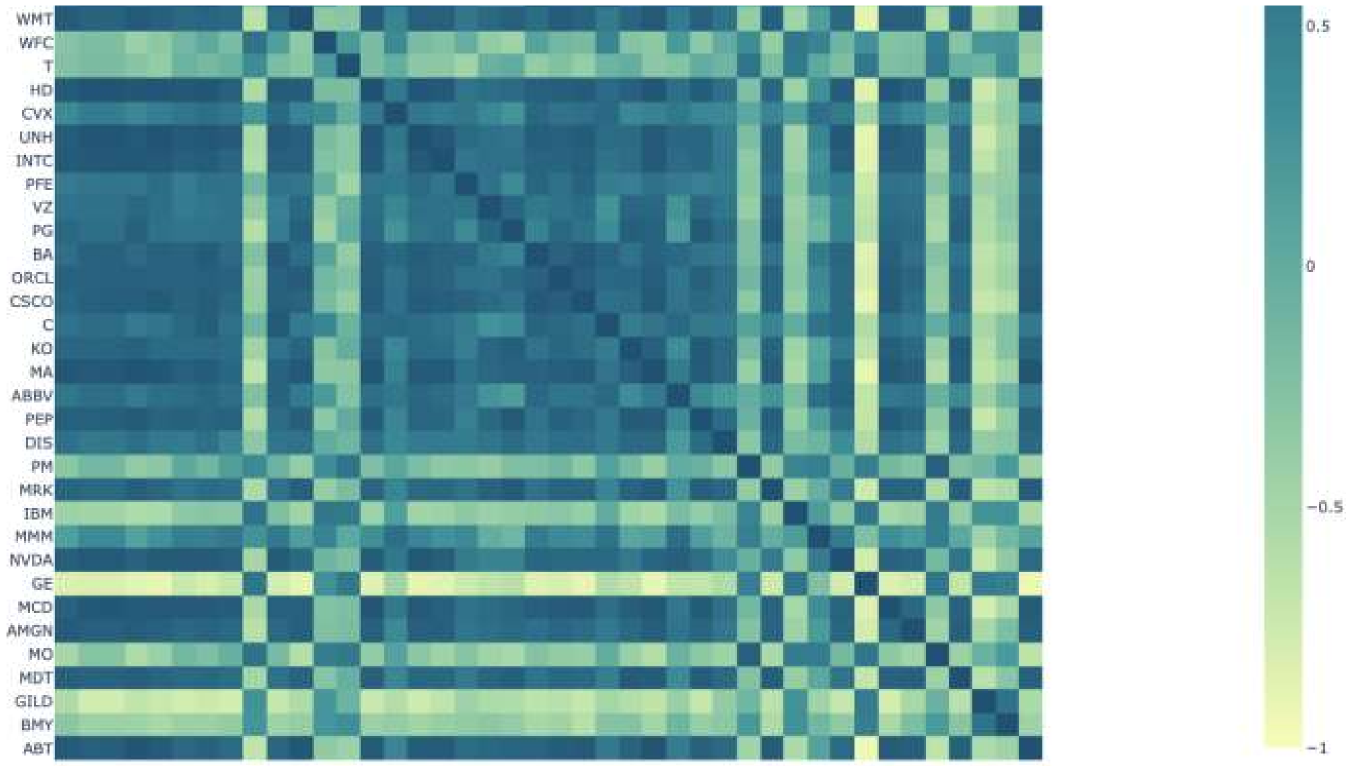
A simple stock price correlation heatmap

That’s fantastic. To recap, we’ve so far managed to download stock price data, calculate their correlations, and see which stocks might be correlated better than others.

So, let’s take a look at this correlation data for some of the largest companies in the U.S. (by market cap).

Upon inspection, the data still looks sensible, but it’s hard to make see very clear groups of data. The reason is that the ordering of companies here is by size, not by correlation.





Correlation matrix heatmap for ~top 50 U.S. companies

What we can do, instead, is to cluster related data here, using our similarity matrix. For that I will use scikit-learn's spectral biclustering algorithm.

Which clusters shuffled data into a 'checkerboard' pattern — in other words, there are underlying groups of data that are more similar than others (further detailed explanation). This is my implementation of it:

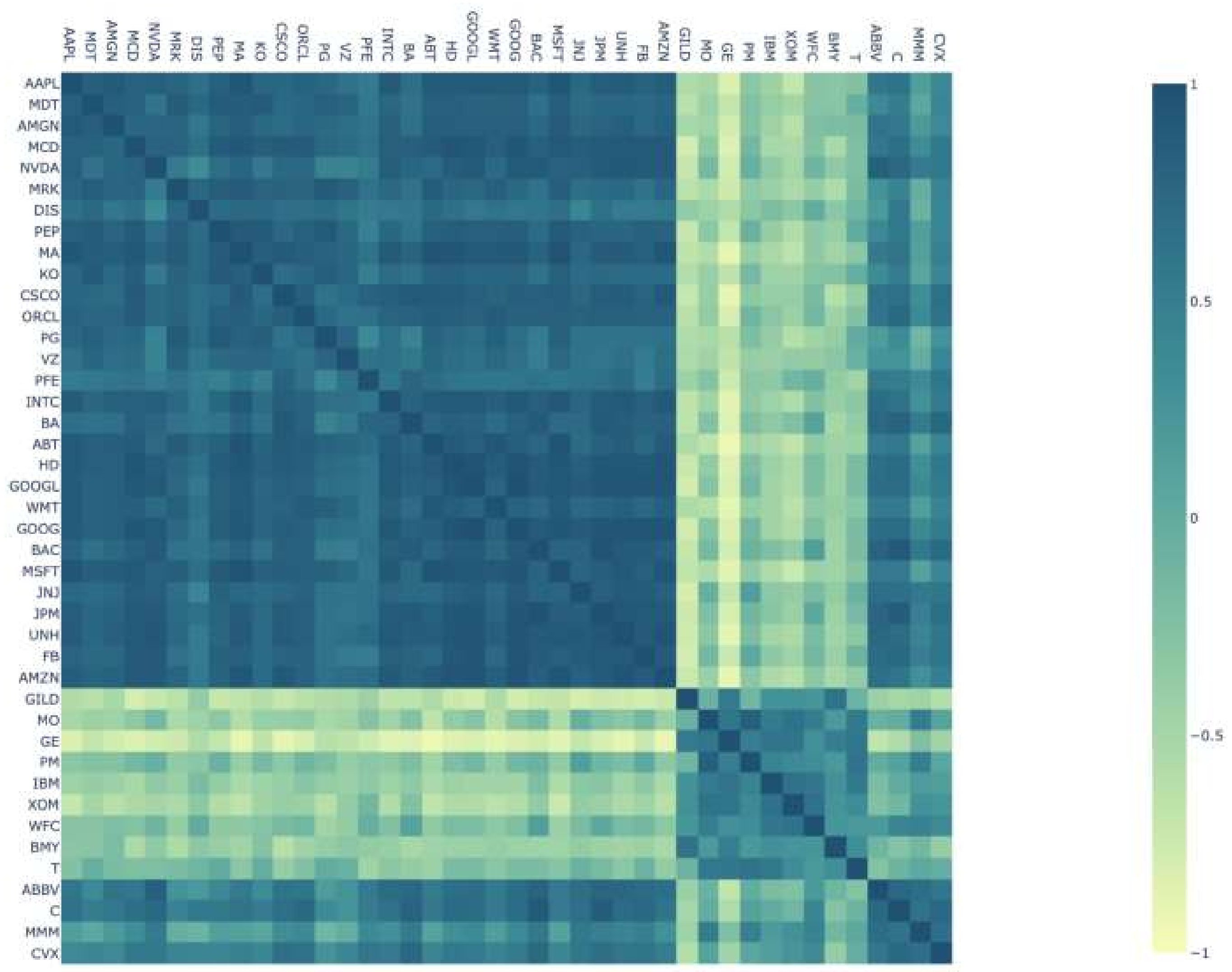
```python
# Cluster data
model = sklearn.cluster.SpectralBiclustering(n_clusters=3, method='log', random_state=0)
model.fit(r_array)

r_array_srt = r_array[np.argsort(model.row_labels_)]
r_array_srt = r_array_srt[:, np.argsort(model.column_labels_)]
syms_srt = filt_df.sym.unique()[np.argsort(model.row_labels_)]

fig = px.imshow(r_array_srt, x=syms_srt, y=syms_srt,
                color_continuous_scale=px.colors.sequential.Bluyl, color_continuous_midpoint=0,
                )
fig.update_xaxes(side="top")
fig.show()
```

get_stock_data_cluster_charts.py hosted with ♡ by GitHub view raw

And the resulting heatmap is as thus:



Stock groups

So there it is — we've successfully (although very crudely) grouped stocks based on their performance of the last five years. Obviously, this can be expanded to larger numbers of stocks and into finer subgroups.

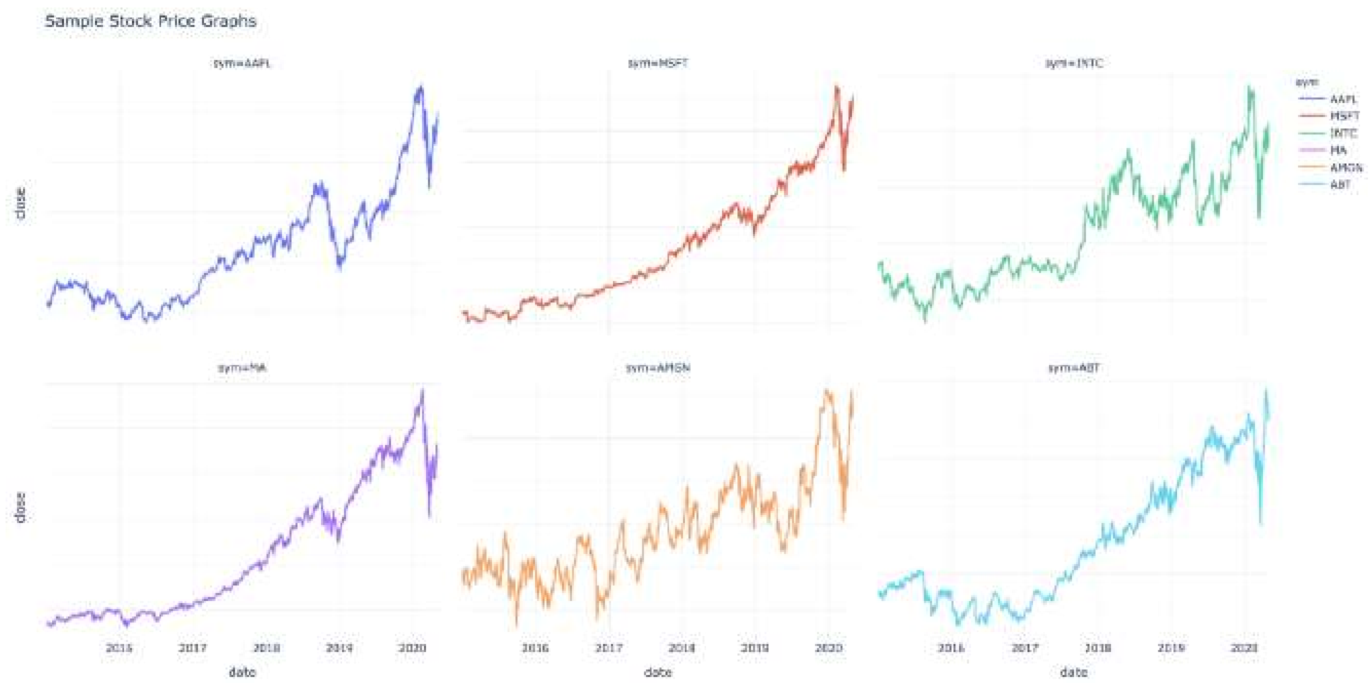
How did we do? Let's take a look at some groups of data. Here's how I identified and plotted them:

```
# See sample charts of correlated stocks
sim_charts = np.argsort(r_array[0])[-6:]
sim_syms = [filt_df.sym.unique()[i] for i in sim_charts]
sim_df = filt_df[filt_df.sym.isin(sim_syms)]
fig = px.scatter(sim_df,
                 x='date', y='close', color='sym', template='plotly_white', title='Sample Stock
```

```
                  facet_col='sym', facet_col_wrap=3
                  )
fig.update_yaxes(matches=None, showticklabels=False)  # If we want to simply view curve 'shapes'
fig.update_traces(mode='lines')
fig.show()
```

get_stock_data_sim_graphs.py hosted with ♡ by GitHub view raw

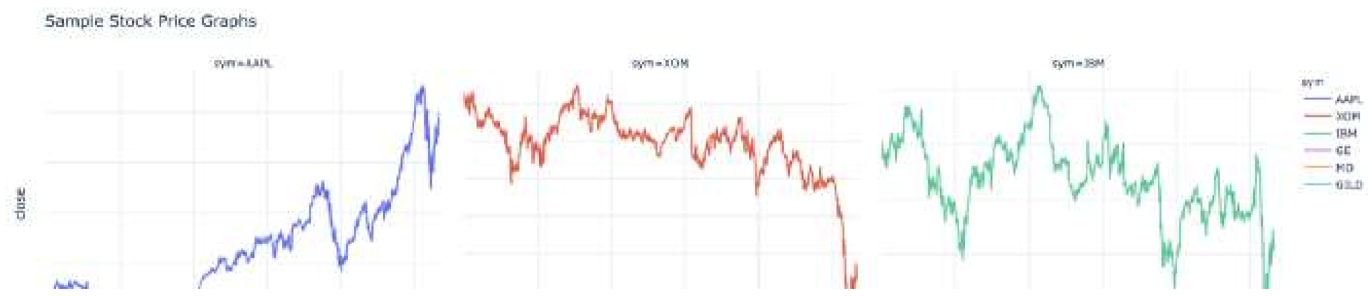
And the resulting charts:

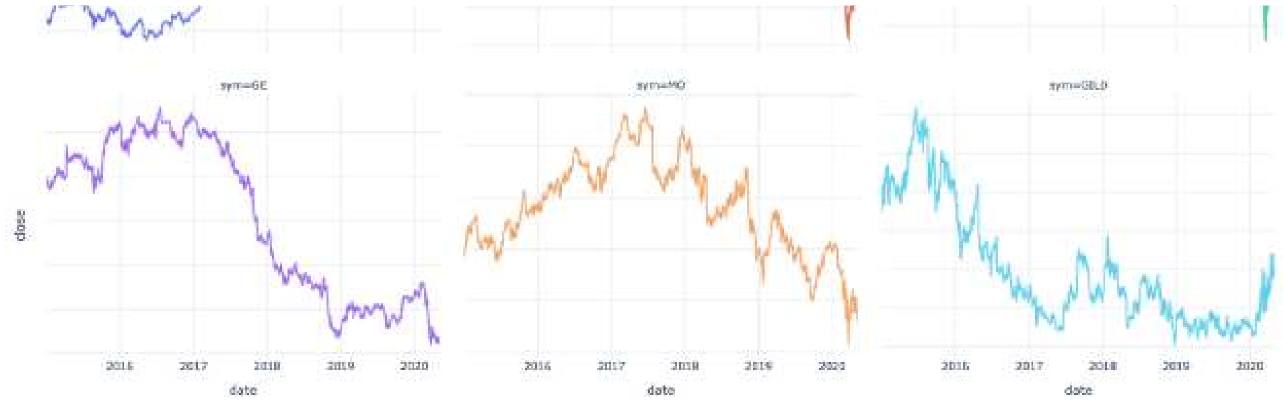


AAPL and correlated stocks

All of these look pretty well related to each other in their overall shape! The results include tech giants like Microsoft and Intel that you'd expect to see well correlated with Apple performance, but also others such as MasterCard, Amgen and Abbott.

What about the other end? Well, here are the top 5 most inversely correlated stocks — take a look:





AAPL and uncorrelated stocks

These numbers paint a very different picture. Here, the remaining charts are clearly, and significantly inversely correlated to the first (AAPL) chart.

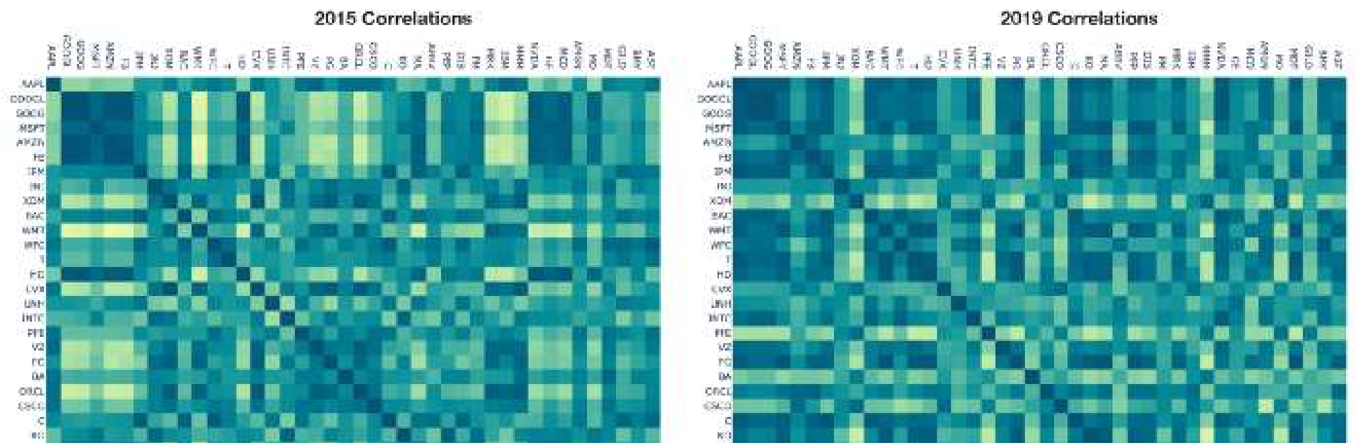
At a larger scale, this data can be used to collect groups of stocks that move together, or not, and to construct a portfolio.

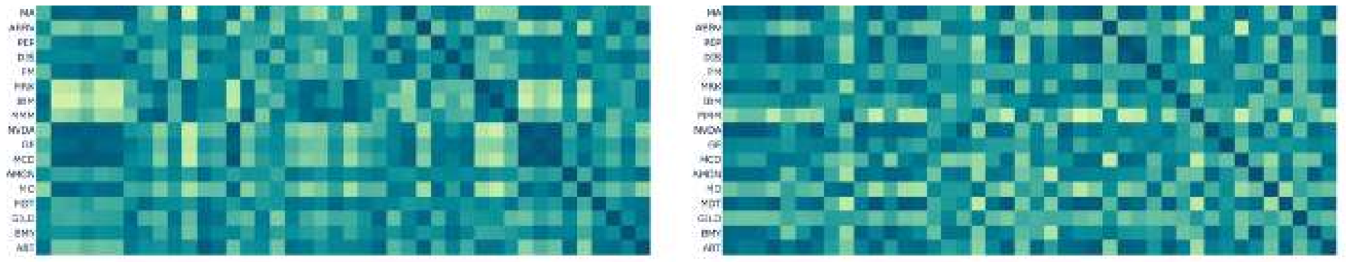
## A little bit extra (on our modelling assumptions)

What we've put together above is a pretty simple model, and a way to group stocks together by their historical movement.

But here's one note — is it reasonable to group stocks based on a period that long? Actually, probably not so. First of all, our graph is going to be dominated by the last few months' data where COVID-19's impact and the market response is going to disproportionately impact graph shapes. Also, companies don't stay stagnant! A company in 2015 often looks very different to that of 2020.

Take a look at this example comparing the same sets of stock data at different times. The chart on the left is for 2015, whereas the chart on the right is in 2019.





Correlation charts — 2015 vs 2019 — Very different!

To another, more fundamental point, is a Pearson Correlation the best algorithm to assess stock correlation?

If this basic introduction to identifying time-series similarities left you with more questions than answers, you're not alone. In my next articles, I will tackle some of these points, comparing resulting groups of stocks against their manual classifications, assessing stocks' movements between groups across time, and testing different algorithms.

Until then, stay safe!

If you liked this, say 👏 / follow on twitter, or follow for updates. ICYMI, I also wrote this article about financial data APIs:

**Comparing the best free financial market data APIs**

Start here if you are looking to analyse financial market data — for stock markets or crypto, as a data scientist…

towardsdatascience.com

and this about building web apps:

**Build a web data dashboard in just minutes with Python**

Exponentially increase power & accessibility by converting your data visualizations into a web-based dashboard with…

towardsdatascience.com

😁👍♡, JP

Data Science Finance Technology Programming Stock Market