Project Report

(To access interactive plots in this report, please see *Assessment.ipynb*)

https://github.com/marcohoucheng/UCDPA_MarcoLam

Abstract

In this report, we look at the prices of Apple Inc. and Bitcoin in Euro for 2021. We use the `yfinance` API in Python for these prices in US dollars. By using a historical exchange rate dataset from Kaggle, we convert these prices into Euro. We apply statistical analysis on the data available to us, along with visualisation of the data. Finally, we discuss the outlook of this report and the potential ideas of further research/studies.

Introduction

The financial market is vibrant and unpredictable. The technology industry and cryptocurrencies have attracted a great amount of attention and have encouraged people to invest whom had never done before. In this report, we will quickly study one of the biggest technology cooperation and the biggest cryptocurrency, Apple Inc. and Bitcoin. We will focus on their performances in 2021, as the world moves away and recovers from the COVID-19 pandemic.

It is well known that the financial market is very difficult to model and predict. Through statistical analysis, I hope to draw some simple statistics, characteristics and patterns of, and between, the two investment options.

Dataset

There are 2 data sources used in this report. The first is a historical exchange rates dataset downloaded directly from Kaggle as a `.csv` file. According to the `About` session on the website, these rates are usually updated around 16:00 CET on every working day.

The second data source is an API package `yfinance`, through this package I will be able to download historical prices for investment of interest. The API typically returns the low, high, open and close prices. In this report, I will use this tool to get historical prices for Apple Inc. (AAPL) for the year 2021.

Although historical exchange rates can be captured through packages such as `yfinance`, I have chosen Kaggle in this report to show my knowledge in importing data through various methods.

Implementation Process

The AAPL data downloaded from `yfinance` are in US dollars, we will need to use the Kaggle dataset to convert these prices to Euro. As the exchange rates are usually updated around 16:00 CET on every working day, we will mainly focus on the opening prices for AAPL as it is consistently closest to 16:00 CET on each working day.

To convert currencies, we must first ensure that our data from different sources have the same data type and structure. The `yfinance` output is `DataFrame` from `Pandas`, so it would make sense to use the `read_csv()` function from `Pandas` to load in the `.csv` file. After inspecting this dataset, we need to change the date format to `DateTime` and make it the index of the dataset. We can also take a subset of the dataset to only include the USD/EUR conversion rates and filter the data to include the 2021 data only. Finally, we can rename the column headers to make them more consistent with each other.

The data sources are now ready to be merged into a single DataFrame, there are a number of ways to do this using the `Pandas` package. We look at `concat()` and `merge()`, `concat()` works as appending tables whereas `merge()` works similar to joining in SQL. As we need to have data for both prices and exchange rates, inner join is the appropriate choice to merge the data with the `merge()` function. Looking at the joined DataFrame, we can see that there are still missing records in the data. By running `dropna()`, we now have a DataFrame that has no missing data. Finally, the prices for AAPL and BTC in EUR can be found by dividing the USD-EUR conversion rate to the US dollars prices.

Before data analysis, we can sort our data with `sort_values()` and `sort_index()`, we will sort the data by `Ticker` and then `Date` (our index). This table will be use in our plots and data analysis later in order to confirm the data is in the order we want. We will also create a new column `Month` to allow us to group the data. If required, the DataFrame can be extracted into Lists, which can be joined into a Dict, or be extracted into a 2D `Numpy` array. As you will see in the code, you can do this by using a for loop with `iterrows()`. The 2D `Numpy` array will allow us to apply mathematical functions to the data such as `round()` to round the prices, or to find the correlation between the 2 arrays by using `corrcoef()`. We can demonstrate the correlation between AAPL and BTC rounded to 5 decimal places in the code.

Alternatively, we can use `groupby()` and `agg()` to our DataFrame to see aggregations of the data separated by groups desired. For this report, we will get the minimum, mean , maximum, and standard deviation for the prices for each instrument per month. We will also add 2 new columns containing the name of the instrument and the month of the data to simplify our visualisation process.

Results

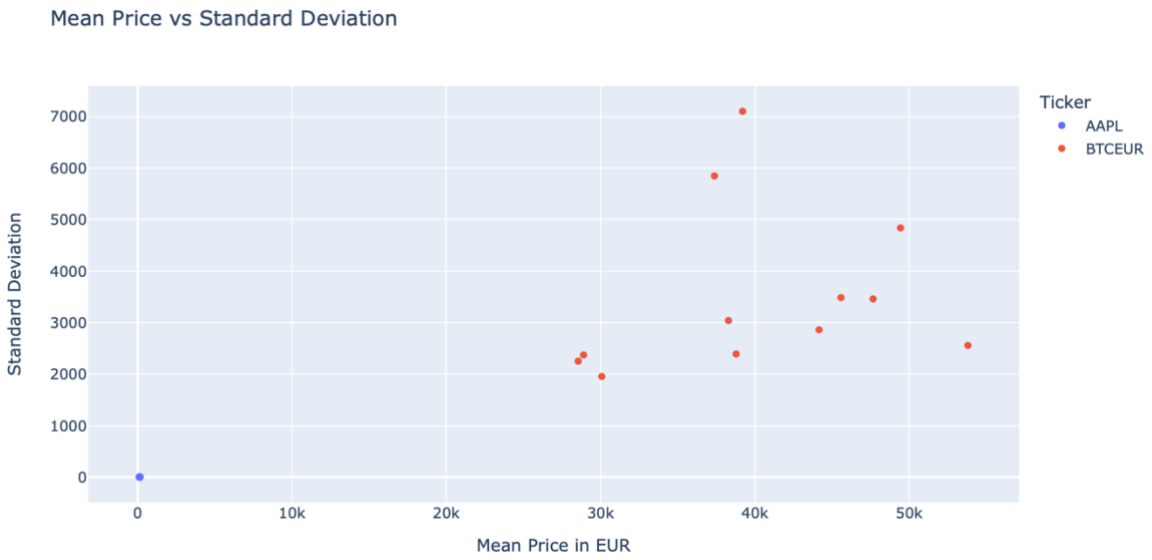
First, the correlation between the 2 prices is roughly 0.28264. A correlation close to 0 (and far away from -1 and 1) is deemed to be weak. So there is little correlation between prices of AAPL and BTC. In other words, we cannot simply predict/explain the price of AAPL or BTC given the other price.

Looking at the aggregated data shown below. It is clear that AAPL's prices are much lower than BTC generally. Scanning through the table, AAPL's highest price in 2021 was €160.16, whereas the lowest price for BTC was €25176.61.

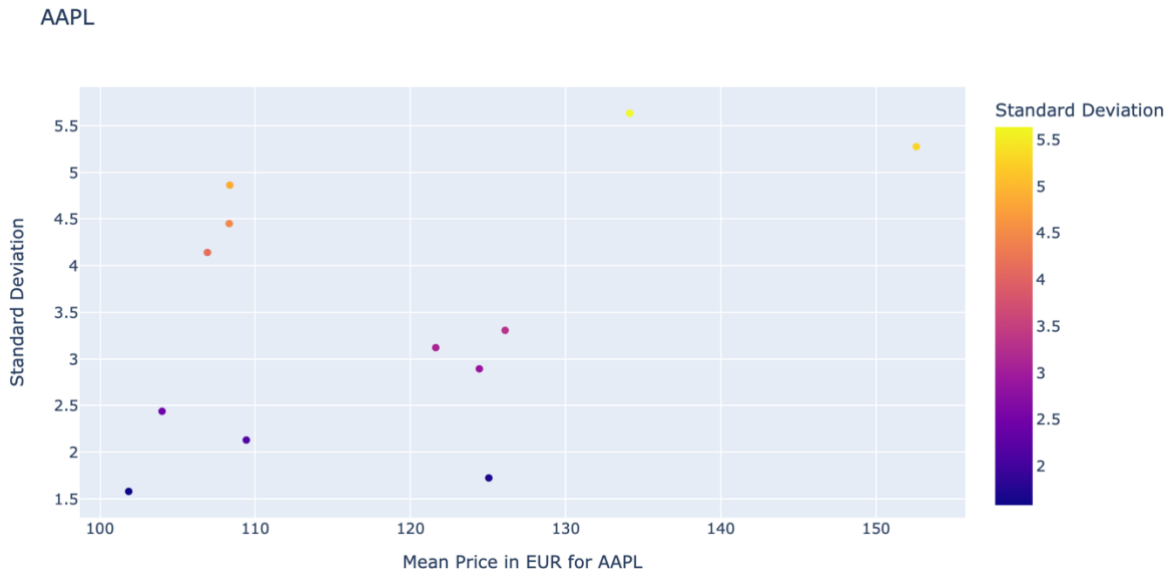
Standard deviation is often used as a measure of risk and volatility of a financial product. AAPL was the least volatile in March and the most volatile in November. Similarly, BTC was the least and most volatile in June and May, respectively. We can see more of this in our visualisations.

Ticker	Month	Min	Mean	Max	STD	Ticker	Month
AAPL	**1**	102.601673	108.355595	117.352616	4.863171	AAPL	January
	2	100.393439	108.318875	113.776323	4.450954	AAPL	February
	3	99.338438	101.845350	105.972755	1.580492	AAPL	March
	4	104.502828	109.419980	111.957489	2.131126	AAPL	April
	5	100.278715	103.994002	108.823715	2.439542	AAPL	May
	6	101.406522	106.914630	113.931707	4.141693	AAPL	June
	7	114.291494	121.623478	125.660270	3.121364	AAPL	July
	8	121.987118	125.040625	128.460540	1.724993	AAPL	August
	9	121.737522	126.091501	132.173914	3.306742	AAPL	September
	10	119.725401	124.432099	128.691549	2.894302	AAPL	October
	11	127.585077	134.118901	143.057227	5.633117	AAPL	November
	12	139.611456	152.579427	160.156181	5.275295	AAPL	December
BTCEUR	**1**	25176.611542	28541.751013	32299.519122	2252.214944	BTCEUR	January
	2	27403.656178	37369.969103	47418.394693	5846.374951	BTCEUR	February
	3	37467.438734	45570.316193	50260.364472	3483.738536	BTCEUR	March
	4	40610.503077	47649.499280	53095.748835	3459.233847	BTCEUR	April
	5	28414.971570	39193.787158	47868.248084	7100.530442	BTCEUR	May
	6	26586.831136	30067.227599	33388.807374	1955.472334	BTCEUR	June
	7	25311.149470	28887.244750	33686.055020	2372.364794	BTCEUR	July
	8	32217.630918	38277.689661	42216.650474	3040.407520	BTCEUR	August
	9	34681.518565	38775.922137	44401.754190	2391.195122	BTCEUR	September
	10	37773.053610	49427.881755	56717.568424	4837.984655	BTCEUR	October
	11	50008.486267	53796.751530	58348.220070	2556.523515	BTCEUR	November
	12	41018.709607	44150.304540	50460.685328	2860.146245	BTCEUR	December

Plotting the mean price against the corresponding standard deviation, we can once again see that both price and standard deviation for BTC is generally much larger than AAPL. In fact, we can't see the AAPL clearly as the scales for BTC's data is just much bigger. This can be viewed as a visualisation of "High risk, high reward." Hovering the curser will show you the month each point represents.

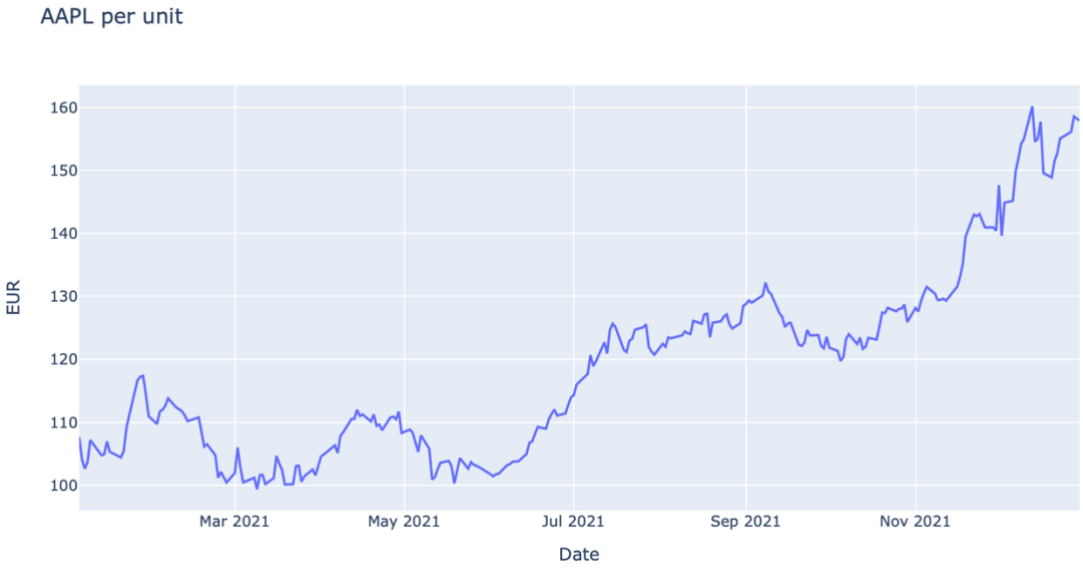


To look at the AAPL data, we can filter out the data by using `master_EUR_agg[master_EUR_agg['Ticker'] == "AAPL"]`.

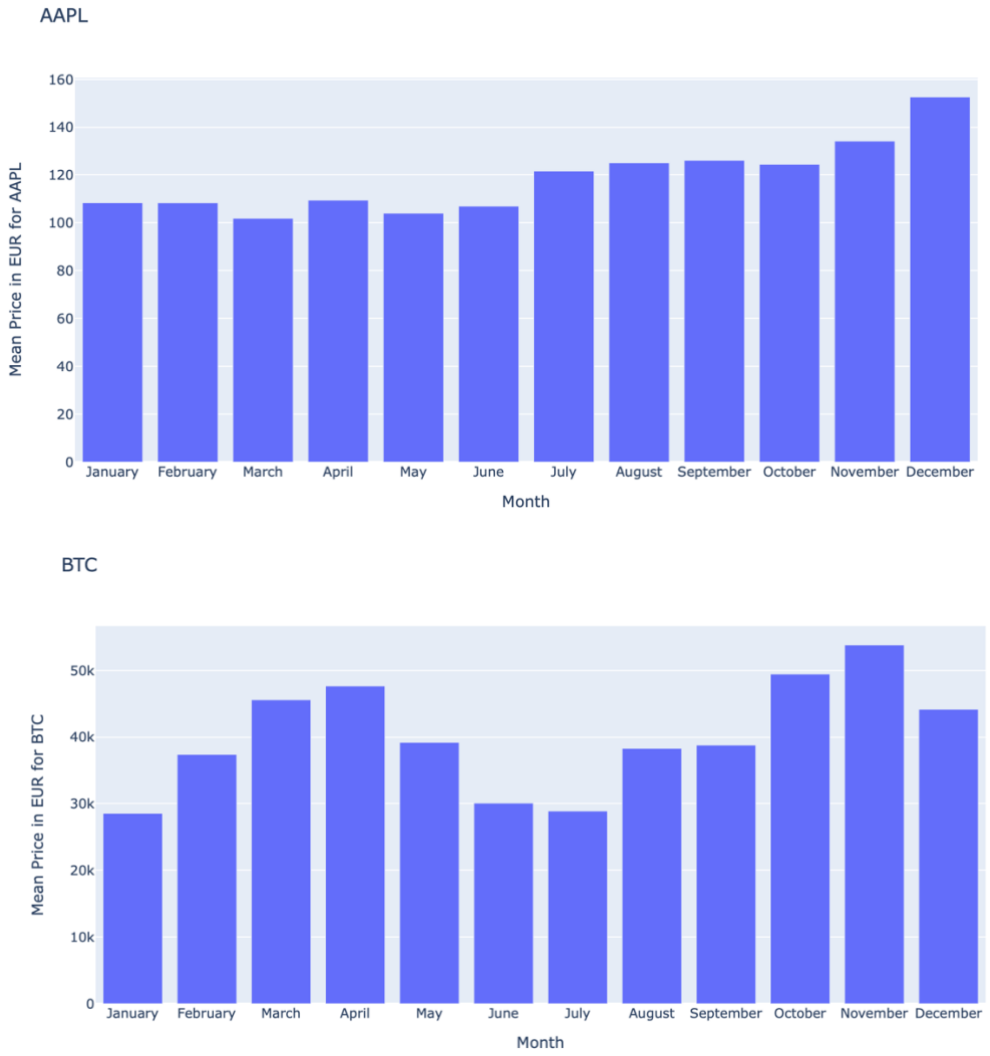


Now we can look at the line plot for the prices over the year. Again, the AAPL data is compressed to a line as the BTC prices are much larger and much more volatile. Hovering over the line will show you the price on a particular date. Recall that BTC was the most volatile in May and least in June, this can be seen as there is a large decrease in price for BTC in May and the price remained (relatively) stable afterwards in June.

We can see the AAPL's line chart, the huge spike in price in the last quarter of the year would explain the high standard deviation (volatility).



We can also look at the mean price per month for both AAPL and BTC.



Finally, the candlestick plot below shows the daily AAPL prices in 2021 in US dollars. This shows each day's opening, closing, minimum and maximum prices. You can drag the bar at the bottom to narrow down the period of time shown.

Insights

1. Conclusion

- There is little to no correlation of price movement between AAPL and BTC in 2021.
- A single unit of BTC is priced much more than AAPL, and it is also a lot more volatile (more risk).
- Statistically speaking, standard deviation is a measure of spread. Thus, a higher volatile of price implies a bigger standard deviation.
- `Numpy` is an excellent mathematical package, `Pandas` is great for data management and preparation for data analysis.

2. Further development and improvements

- I would convert all columns from `yfinance` output tables to EUR, so the candlestick plot can show the EUR data. Although it would be difficult to find the accurate exchange rates without knowing the time of each day when the highest and lowest prices occur.
- To plot AAPL and BTC together, it'd be a good idea to normalise the data (i.e. against their mean) or to use duo axes. Of course, this would simply make the plots look good, and not scientific whatsoever.
- I would like to look in to applying machine learning or statistical models to the data, although it is generally difficult to model financial data.
    1. Time Series Modelling: As financial data is real-time, an ARIMAX model might be a good candidate to predict prices in the future. An ARIMAX model is an extension of ARIMA model, where it can take in other time series data that are highly correlated to the data of interest. It would be ideal if there exists a group financial indices that are highly correlated to our subject.
    2. Deep Learning RNN/CNN models: Neural Networks are very powerful in prediction. Given the years of financial data in the world, theoretically we would have more than enough data to train such prediction model. RNN/CNN would have an edge over ANN as ANN is known not to excel in time series data prediction.
- Both of the models mentioned above are supervised regression methods. Given the amount of historical data in finance, it'd make sense to train and validate the prediction models with the available data. Choosing regression models over classifications is simply due to the fact that our interest is a numerical value, not member of categories.
- This project can be developed into a web application with `Dash`, an extension to the `Plotly` package. I would like to achieve this in the future and allow users to access this project and its interactive plots online.

References

Kaggle - Daily Exchange Rates per Euro 1999-2022

yfinance