# Making sense of the stock market: Data mining financial time series for summarative patterns

Francesco Vaglienti

February 20, 2020

## 1 Introduction

Stock markets are one of the most important structures that dictate civilized life. Were it not for stock markets, insurance companies would not be able to offer vital services, leaving certain surgeries only to those rich enough to afford them. Stock markets are also the foundation upon which retirement plans rest, allowing freedom from work at the end of ones life. And last but not least, it is because of stock markets that long-standing industries can be forced to innovate.

However, when it comes to analyzing the stock market, it has always been one of the most elusive of time series. Many movies[1] and even more papers have dedicated themselves to explaining the seeming chaos that are financial markets. But what if you could describe the majority of stock movements with no more than a handful of patterns.

Being able to describe a stock through a limited set of patterns would allow the entirety of the stock to instantly be understood, as every movement could be explained through the pattern it is part of. To put it differently, once the dataset has been transformed, it is nothing more than sequence of patterns and all that is left is to find the reason as to why the pattern occurs.

This is far easier to find than to understand why a stock went up on a particular day, due to patterns inherently being repetitive. To understand the patterns, all that needs to be done is to create a timeline alongside the stock and check what event occurs systematically alongside the pattern. Patterns also have the added advantage of adding predictability as well as stability to data analysis.

Take, for example, the stock price of Coca-Cola and Pepsi, two competitors within the soda industry. By describing the stock price of both Pepsi and Coca-Cola (from hereon: PepCo) through patterns, it becomes easy to understand both companies and their markets as a whole. The patterns where PepCo move together show aspects regarding to the soda industry as a whole. While the patterns where PepCo diverge from one another, show where the companies have different solutions to a similar problem. Finally, the remaining parts that are not covered by patterns shows the unique properties of the company that are independent from both their competitors and the market as a whole. This illustrates that patterns can explain all the movements of a company's stock, from the biggest macro to the smallest micro level.

Within this paper, it is shown that through applying a DITTO-like [1] algorithm called PYCCOLI, a set as small as 13 patterns can describe over 85% of the movements of the Pepsi and Coca-Cola stock. It is also shown that this method can be applied to describe a single stock, an index stock, or the relations between an index and its component stock.

The results achieved are due to the application of the minimum description length principle as proposed within KRIMP [2] (from hereon: KRIMP-principle). The KRIMP-principle can be summarized as the idea that the more something shortens the description of the data, the better it describes the data.

It has been proven to be a powerful method to describe univariate patterns [3; 4] and has been extended to apply to multivariate time series through DITTO [1]. PYCCOLI is in essence a smaller, parallelized version of DITTO.

This paper is the first to utilize the KRIMP-principle to extract patterns from financial time series. The KRIMP-principle lends itself well to describing financial markets as it assumes the same random walk as the efficient market hypothesis [5].

To elaborate on this, first a brief explanation of the efficient market hypothesis will be given, followed by one of the KRIMP-principle. These will then be brought

[1]such as: Darren Aronofsky's $\pi$

together to show why the KRIMP-principle lends itself so well to financial analysis.

The financial random walk theory states that stock market prices changes are random and thus cannot be predicted, as proposed by [6]. The efficient market hypothesis (EMH) then seeks to explain this hypothesis.

According to the EMH, market prices represent a culmination of all information known regarding the stock. Additionally, due to new information regarding the stock being released in random order, the market then integrates the news in said order. This then gives price movements their random movement.

Resulting from all past information being priced in and past prices only indicating information regarding past information, the efficient market hypothesis states that no analysis of the historical stock price can reveal information regarding its future price. This is also known as the weak form of the EMH, which unintuitively is the form which has the strongest support.

As previously mentioned, the KRIMP-principle looks for a way to shorten the description of the dataset. It achieves this by seeking to shorten the statistical description of the dataset. In other words, it shortens the dataset by finding statistical relations within the dataset and uses said relations as descriptors. Therefore, if there are no statistical relations within the dataset, it cannot be shortened through the KRIMP-principle.

Putting both together, it means that if historical prices truly do not affect current price, the KRIMP-principle will not be able to compress the dataset. Given that the stock movements will be random and that therefore there will be no statistical relation within the dataset.

It, however, also means that any relation that is found, is a genuine statistical relation underlying the dataset. As such, the advantage of the KRIMP-principle is that it allows the extraction of true statistical relations from the dataset.

In brief, the goal of this paper is to demonstrate the merit of PYCCOLI as a method of analysis for financial time series. This is done by analysing real-market data and discussing the utility of the findings, while also demonstrating that they are but the tip of the iceberg.

The remainder of this paper is structured as follows: Within Section 2 the necessary definitions are given; Section 3 explains the method used and the theory behind it; Section 4 gives some context regarding the experiments; Section 5 presents the results from using the PYCCOLI algorithm on the selected datasets and finally Section 6 offers some conclusions and suggests future work.

## 2 Definitions

Before going into how all patterns are extracted from the dataset, a definitions of both a dataset and a pattern must be given. As the presented work draws mainly from DITTO [1], the definitions are lifted from said work. The reader is thus referred to DITTO for more in-depth definitions as well as theory.

As data type to analyse, a dataset $D$ is considered. $D$ contains $|D|$ multivariate event sequences $S \in D$, all over attributes $A$. Assume $A$ is indexed such that $A_i$ refers to the $i^{th}$ attribute of $D$.

A multivariate pattern or pattern $S$, is a bag of $|A|$ univariate patterns, $S = \{S_1, ..., S_{|A|}\}$. An univariate pattern $S_i \in \Omega_i^n$ simply is a sequence of $n$ events drawn over an alphabet of singletons $\Omega_i$.

The length of the patterns is denoted as $||S||$, which is the total number of events; the time length of the pattern is denoted as $t(S)$, which is its total number of time steps, finally the support of the pattern is denoted as $sup(S)$ representing the amount of times $S$ can be found without overlap within $D$.

## 3 Method and Theory

With the necessary notation out of the way, the method used for obtaining the results can be discussed. The methodology is divided into three parts, acquiring the data, transforming the data into usable input and finally constructing and using the PYCCOLI algorithm. If the reader is not interested in the details, they are advised to skip to Section 3.4.

### 3.1 Obtaining financial data

Markets, as well as the data pertaining to them, are both within the public domain. However, there is no easy solution to simply collect all of the data for free. To address this problem three different data miners were constructed to gather all the stock data that will be mentioned below.

The reason to write three different data miners is due to the uncertainty regarding the quality of the data, as some free data providers will at times purposefully put false datapoints within their data. This is why three different datasources were mined for financial data, i.e. Yahoo Finance, Finam [Russian] and AlphaVantage. When it was found that all three gave the same data regarding a stock, Finam was used due to its high amount of available data.

With the more practical side of how to acquire stock data out of the way, the question becomes what data is needed. To analyze a stock, merely the stock data pertaining to itself is needed (that is, its open, close, high, and low prices as well as the volume sold), however when analyzing a stock index, things slightly change.

A stock index represents a collection of stock that are bundled together, notable examples are the Standard and Poor 500, the NASDAQ, and the index that will be used for analysis within this paper, the Dow Jones Industry Average.

To obtain the data regarding what stocks are needed to represent the composition of the index, the reference website SlickCharts was used. This website shows the current components of all three aforementioned index stocks.

Once the component data and the appropriate stocks have been collected, a final component must be added, the stock data of the index itself. With this done, all data needed for analysis has been collected.

There is however one caveat, which is that components change over time. This means that the currently analyzed components might not represent the past components of an index stock. However, for sake of simplicity, current components were treated as if they were all-time components.

## 3.2 Transforming financial data

When analyzing financial data, the measure of interest lies within the price movements. Consequently, the measure of interest lies not within the absolute value of a stock, but rather within its difference relative to the previous day.

However, it must be taken into account that the longer a stock exists, the larger its price and thus the bigger its day-to-day absolute differences get. This is especially the case with the large (blue-chip) stocks that are used within this analysis. This causes the problem that the day-to-day absolute differences may get larger over time while the relative price movements remain the same.

The solution to this is to take the percentual difference between every day and its previous day. This assures that underlying change always stays relative to the size of the current whole. This is done for every sub-component of every stock used, allowing the transformation process to move onto discretization.

Discretization is necessary, as no patterns can be found within continuous variables. This is due to the fact that, in essence, a set of continuous variables can be considered as a set of unique variables and that no patterns can be found between solely unique variables.

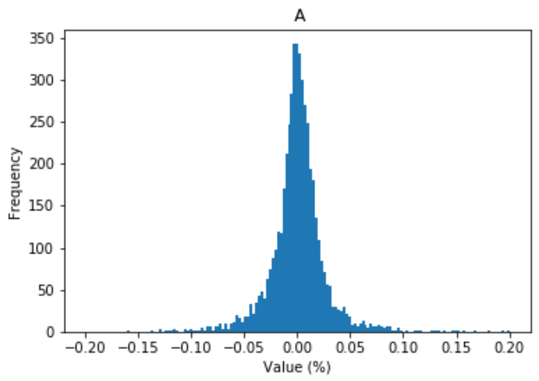

Figure 1: *Distribution of perceptual differences within Agilent Technologies Inc. stock. Click here or on the image for the distribution of all stocks.*

For discretization DITTO uses SAX [7], however SAX assumes normality which is not the case when observing financial data. This is caused by the tendency of stocks to either increase over time or disappear from the market altogether. The result of this survivorship bias is a positive skew within the dataset. Furthermore, many days are marked by minor adjustments of under 1%, while some days are marked by extreme adjustments of as much as 90%. Thus, the distribution is not only defined by a high peak but also by 'fat tails', as exemplified within figure 1.

To circumvent this problem while still constructing bins that are as distinct from one another as possible, k-means [8] was used. The reason for creating bins that are as distinct from one another is that it gives as much expressiveness to the data as possible, which will result in more structure being found.

Thus, for every sub-component (such as: closing price) from every stock, k-means was run. The best amount of bins used was determined to be 5 within a pre-study. A benefit of using an odd value means that the median value of 3 can be used as a no change value. In extension, values higher than 3 indicate an increase while values lower than 3 indicate a decrease, making the results easier to interpret. It should be noted that due to the aforementioned positive skew, the no change value sometimes is 2 rather than 3.

Taken all together, once stock and index data have been collected, the percentual difference between every current and previous day is taken and is individually classified though k-means into 5 distinct bins. As such, every part of every instrument is represented as a single attribute with an alphabet of 5. The data is then put together into a single dataset.

## 3.3 Analyzing financial data

Finally, the method through which the data is analyzed can be explained. As previously mentioned, PYCCOLI is a derivative of DITTO. As such, both are based upon the KRIMP [2] algorithm, which is displayed within figure 2.

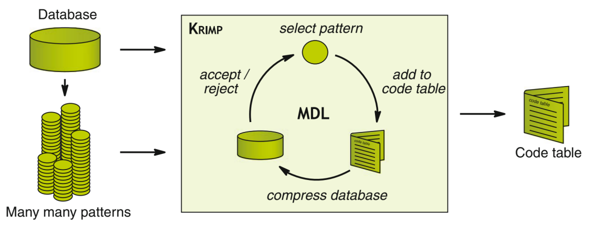

Figure 2: *The KRIMP algorithm, taken from [2]*

**KRIMP** KRIMP, and by extension any algorithm based on its philosophy, has a goal to find the **minimal pattern set**. That is to say, to find the set of patterns which results in the smallest minimum description length (MDL) of a dataset.

Calculating the MDL of a dataset is done through a codetable. The codetable tells a cover algorithm how to compress the database and this compression can be given a value. MDL represents this compression value where a smaller value signifies that the data is more compressed.

The more compressed the data it, the shorter its description. According to the KRIMP-principle, if two things describe an object and one uses less descriptors to arrive at the same result, then the one with the shorter description is better. This corresponds to the idea behind Occam's razor.

Bringing it back to codetables, this means that the more a codetable compresses a dataset, the better it describes it. It then follows that the KRIMP algorithm seeks to find the codetable that maximally compresses the dataset.

The KRIMP algorithm achieves this through a simple but effective method. Take a codetable existing solely out of singletons and plug it into the cover function, this provides the baseline MDL. Then add a pattern to the codetable, if this decreases the MDL keep it in the codetable, otherwise reject it. Next, go over the codetable to see if any pattern can be pruned. Repeat this process until every possible pattern has been generated, and the optimal codetable will have been found.

**PYCCOLI** PYCCOLI, as previously mentioned, is based upon KRIMP. As such it follows the same general process for patterns selection as described within the previous section. However, PYCCOLI has its own solution when it comes to the aspects of how candidates are generated and how to manage the flow of the procedure. The pseudocode for this can be seen in Algorithm 1.

**Algorithm 1:** The PYCCOLI algorithm

```
Input  : The dataset $D$ and singleton table $ST$
Output : An approximation to the minimal
         patterns set problem
1  $CT \leftarrow ST$
2  $Cand \leftarrow M^*(CT \times CT)$
3  while True do
4      $mdl \leftarrow L(D, CT)$
5      $CT, used\_patterns \leftarrow$
         $Addition(D, CT, Cand)$
6      $CT \leftarrow Removal(D, CT)$
7      if $L(D, CT) < mdl$ then
8          $mdl \leftarrow L(D, CT)$
9          $Cand \leftarrow M^*(CT \times used\_patterns)$
10     else
11         $Break$
12     end
13 end
14 return $CT$
```

The PYCCOLI algorithm takes the given singleton table and uses takes the Cartesian product of the singletons to create the candidates. Within this process it uses multiprocessing (denoted as: $M^*$) to speed up candidate generation. Once the candidates have been created, a while loop is initiated where patterns are added and removed.

For addition, as can be seen in Algorithm 2, PYCCOLI uses the same logic as KRIMP to check whether a pattern shortens the MDL. However, multiprocessing is used to check multiple patterns at the same time, halt-

**Algorithm 2:** The Addition algorithm

**Input :** The dataset $D$, a codetable table $CT$ and the candidates $Cand$
**Output :** A codetable $CT$ and the $used\_patterns$

```
1  $used\_patterns \leftarrow \emptyset$
2  while True do
3      for $M^*(x \in Cand)$ do
4          $Cand \leftarrow Cand \ominus x$
5          if $L(D, CT \oplus x) < L(D, CT)$ then
6              $CT \leftarrow CT \oplus x$
7              $used\_patterns \leftarrow used\_patterns \oplus x$
8              $Break$
9          end
10     end
11     if $Cand = \emptyset$ then
12         $Break$
13     end
14 end
15 return $CT, used\_patterns$
```

ing the process when a good pattern has been found. If a new pattern has been found which shortens the MDL, it is saved to an array which will later be used for new candidate generation and also added to the codetable. Both this array and the codetable are returned.

**Algorithm 3:** The Removal algorithm

**Input :** The dataset $D$ and a codetable table $CT$
**Output :** A shorter codetable $CT$

```
1  $P \leftarrow CT$
2  while True do
3      for $M^*(p \in P)$ do
4          $P \leftarrow P \ominus p$
5          if $L(D, CT \ominus p) < L(D, CT)$ then
6              $CT \leftarrow CT \ominus p$
7              $Break$
8          end
9      end
10 end
```

Then pruning or in PYCCOLI terms, removal, is applied as can be seen within Algorithm 3. The Removal algorithm goes through every non-singleton pattern within the codetable, and checks whether removing it shortens the MDL. If so, the pattern is stripped from the codetable. Once every pattern has been checked, the codetable is returned.

Finally, after Addition and Removal, the PYCCOLI algorithm checks to see whether the current round of searching has lowered the MDL. If not, the **Apriori principle** [9] guarantees that no further candidates will be found, thus the program exists and returns its optimal code table.

If the MDL has been lowered, the algorithm begins again from the top. But not before saving the lower MDL value as its current MDL and generating a new set of candidates. The new set of candidates are the Cartesian product of the current code table and the array returned by the Addition algorithm.

**Algorithm 4:** The DITTO algorithm

**Input :** The dataset $D$ and singleton table $ST$
**Output :** An approximation to the **minimal patterns set problem**

```
1  $CT \leftarrow ST$
2  $Cand \leftarrow CT \times CT$
3  for $X \in Cand$ in Candidate Order do
4      if $L(D, CT \oplus X) < L(D, CT)$ then
5          $CT \leftarrow Prune(D, CT, \oplus X)$
6          $CT \leftarrow Variations(D, X, CT)$
7          $Cand \leftarrow CT \times CT$
8      end
9  end
10 return $CT$
```

**In comparison to DITTO** Before presenting the results found, a couple of points must be addressed. It had been mentioned that PYCCOLI was based upon the DITTO algorithm (see: Algorithm 4). In effect, PYCCOLI is a smaller (in source code size) recreation of the DITTO algorithm written in Python. This is what gives it its name, as piccoli means small in Italian. However five differences exist that must be highlighted for the sake of completeness.

1. PYCCOLI unlike DITTO uses multiprocessing. Adding to this, PYCCOLI has a max memory usage of 3 GB, whereas there is no limit to DITTO's memory usage.

2. DITTO has a prune order while PYCCOLI has no prune order. As both algorithms still pass every candidate, the only notable difference while testing was within performance. However, for some

datasets having a prune order can hypothetically result in a lower MDL. It was, however, omitted from PYCCOLI to optimize runtime.

3. PYCCOLI uses the cover order[2]: $\downarrow t(X)$, $\downarrow ||X||$, $\downarrow support(X|D)$. Whereas DITTO uses $\downarrow ||X||$, $\downarrow support(X|D)$. In short, this means that PYCCOLI prioritizes patterns that last over multiple units of time, which in practise results in more interesting code tables.

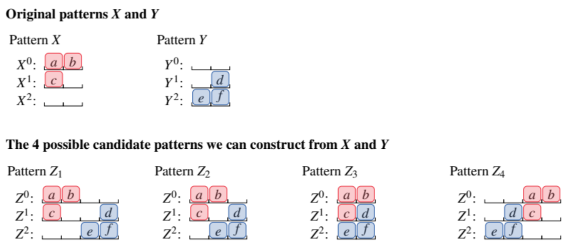
Original patterns X and Y
Pattern X
Pattern Y
The 4 possible candidate patterns we can construct from X and Y
Pattern Z1
Pattern Z2
Pattern Z3
Pattern Z4

Figure 3: *Pattern generation, taken from [1]*

4. The pattern generation is significantly different as DITTO allows gaps and creates the Cartesian product of its pattern whereas PYCOLLI's Cartesian product merely 'sticks it onto the end'. To serve as an example take figure 3. Whereas PYCOLLI only creates pattern $Z_3$, DITTO also reverses the order as can be seen in $Z_4$. Furthermore, DITTO also creates patterns $Z_1$ and $Z_2$, thus allowing a potentially better cover of the dataset.

5. PYCCOLI generates every complete generation of patterns before moving onto the next generation of patterns. As such it is an anytime algorithm, whereas DITTO is a depth first algorithm and does not have such a function built in. This means that after $n$ round(s) of computation, PYCOLLI will contain the optimal codetable for the dataset with maximal pattern length $2^n$. DITTO, on the other hand, finds a good pattern and keeps extending it until it stops lowering the MDL.

## 3.4 Method in short

As the methodology is quite lengthy, a final overview is given to serve as a brief summary to how exactly the data is acquired.

[2]See [1] Section 4.1 for a detailed explanation of covering and how cover order relates to it

In short, a collection of stock and index data is taken and the continuous values are transformed into discrete units, ranging from negative to positive change. The transformed financial data is then run through the PYCCOLI algorithm, which outputs a codetable. This codetable, which contains patterns that describe the dataset, is the best (approximate) description of the dataset and as a result maximally compresses the dataset.

## 4 Experiments

The experiments section is divided into four general sections, the first serves to illustrate the general conditions that every experiment ran under. The second section serves to give an example of a full experiment and the third shows how to read the codetable which results from an experiment. Finally, the last section give context as to why each dataset was chosen.

### 4.1 General conditions

PYCCOLI has been fully implemented in Python. Furthermore, even though PYCCOLI is an anytime algorithm, experiments were run till completion. The experiments were run using an AMD Ryzen 3950X processor with 32 GB of memory.

For analysis the last 1500 days of stock data were used. Within the dataset, this represents the range from the 10th of September 2013 up until the 23 of August 2019[3]. When analyzing individual stocks the high, low, open, close and volume of each stock were analysed. When an index was analysed or two different stocks were compared, then, only the closing prices were used for analysis. As data interval, daily data was chosen. This was done to keep computational time to a feasible level. It should be noted that even while only examining end-of-day data, the analysis of certain datasets still took three days worth of processing time.

### 4.2 Example experiment

The experimental procedure can be described within three parts, data gathering, data processing and data analysis. For the experimental procedure the Dow Jones index will be taken as an example, however the procedure remains the same for any other stock or index.

[3]With the exception of bitcoin which runs from 2014 till 2020

#### 4.2.1 Data gathering

As all data has already been collected[4], only the necessary stocks need to be put together. For this the component file[5] needs to be consulted to find what stocks are needed.

For the Dow-Jones index, this gives 30 component stocks and the 1 index stock price thus 31 stocks combined. The 31 stocks are imported from the stock data through the `import_csv_to_df` method within the `input_gen.py` file and joined together into one table with the `join_dataframes` method from the same file. As this means all stocks have been gathered into a single dataframe, data gathering has been completed.

#### 4.2.2 Data processing

As the example is index data, only the closing prices should be used for analysis, as such all other data should be filtered out. Furthermore, as mentioned within the method, the data must be turned into percentual differences and discretized. As all methods will be taken from the `input_gen.py` file, no further reference to the file shall be made.

First the data is cleaned from any missing values, this is done by pushing the dataframe through the `remove_NaN_rows` method, which removes rows which contain missing values. Following this, as the amount of days to be analysed is 1500, the last 1500 rows of the dataframe are taken and the remainder are removed.

To get the percentual row differences, the dataframe is put through the `row_diff` method. To then filter out any columns that do not include closing price information the `filter_columns` method is used with the argument `Close`. This ensures that the only columns remaining within the dataframe are those regarding closing prices of the stock.

Finally, as the rows need to be clustered into five different clusters, the `kmean_columns` is used with the argument `nm_clusters = 5`, this produces and adds the clustered rows to the dataframe. The clustered columns are filtered from the dataframe by using the filter method but this time with the argument `LB`[6] and then the filtered dataframe is exported to a PYCOLLI-friendly file through the `export_to_pyccoli` method. This will save a `.dat` file on the hard drive.

[4] `Link to all stock data`

[5] `Link to index reference file`

[6] Which stands for labelled

#### 4.2.3 Data analysis

To initialize data analysis, PYCCOLI must be run on the previously created `.dat` file. For this the `pyccoli` method from the `pyccoli.py` file must be taken and given the argument `filename = '[Location on drive].dat'`, the option for the amount of cpu's to be used (such as 4) can be controlled through the `cpu=4` argument. This argument is optional as by default PYCCOLI uses every core of the cpu.

The PYCCOLI algorithm will then export a codetable both in pickle[7] format and in text format. As such the experiment is complete and the file can be manually reviewed.

### 4.3 Understanding codetables

As the output generated by PYCCOLI is a codetable, it is necessary to be able to understand the given output. To accomplish this, two patterns will be taken from the Dow-Jones code table.

To read a pattern from the codetable, two files are needed, a codetable file[8] and a definition file[9].

Within the Dow Jones codetable, one of the non-singleton patterns is the following:

$\underline{\texttt{((21, (2,)), (22, (2,)))}}$ *(26, 2, 1)*

The underlined section is the actual pattern, whereas the part in italics is information regarding the pattern. First an explanation regarding the pattern will be given and then the part in italics will be explained.

The pattern `((21, (2,)), (22, (2,)))` is a combination of two singletons, namely `(21, (2,))` and `(22, (2,))`. To find to what stock they relate to the definition file[9] is used.

With the definition file, `(21, (2,))` can now be translated to `Close-VZ_-LB` going in the 2 direction. `Close-VZ_-LB` is short for closing price, VZ ticker, k-means labelled. Therefor the singleton pattern indicates that the closing price of the VZ (Verizon telecom) stock ticker is going down in comparison to yesterday.

For `(22, (2,))`, it means that the closing price of the CSCO (Cisco systems) stock ticker is going down. Adding one and one together, the pattern can now be

[7] `pickle` is an internal Python format through which a variable can be reloaded into a Python environment. Consult the `output_gen.py` file for functions regarding the codetable, an example is made available within the `ct_statistics.py` file.

[8] `Link to Dow Jones's codetable`

[9] `Link to Dow Jones's definition file`

translated as `((21, (2,)), (22, (2,)))` describing the closing price of Verizon and Cisco going down together.

Moving onto the *`(26, 2, 1)`* part of the pattern. The first element is the support of the pattern within the dataset, in this case 26. The second part is the total length of the pattern $||S||$, and the final part of the pattern indicates its length within time or $t(S)$.

Another example is given to ensure clarity.

`((3, (3, 2)), (16, (2, 2)))` *`(33, 4, 2)`*

This pattern again consists out of two stocks `(3, (3, 2))` and `(16, (2, 2))`, which are the closing prices of Home Depot and Procter & Gamble respectively. For Home Depot the stock indicates that at first that the closing price does not change and then drops. For Procter & Gable it drops two days in a row. As such taking the whole line together, PYCCOLI says: I have noticed 33 times within the dataset that if Procter & Gamble's price drops while the price of Home Depot remains the same, then the next day the price of both stocks will drop.

It should be noted that patterns can exist out of sub-patterns that cover a variety of days such as :

`((1, (2,)), (15, (3,)), (19, (3, 3)), (24, (3, 3, 4, 3)))`
*`(2, 8, 4)`*

Where two patterns that occur on the same day are put together with a pattern that lasts two days and one that lasts 4 days.

## 4.4 Context regarding datasets chosen

### 4.4.1 Apple

Apple provides an ample starting point for analysis as most financial papers use it as example. Consequently, due to its frequent usage when testing novel methods of financial analysis, no obvious patterns should remain within the historical stock data.

The expected lack of financial patterns is due to the fact that if there was an easy way to profit off the stock it would already have been taken advantage of. As such Apple should be the stock PYCCOLI has the most difficulty with, as it is both analyzed in many papers and incorporated within many private trading engines.

Analyzing Apple stock also gives insight into the viability of only analyzing a single stock, this then allows for further comparison with the larger datasets. In sum, Apple makes a good pick because it is tough to analyze and it is a good example of a small sample.

### 4.4.2 Dow-Jones index

The Dow-Jones index was selected to include the analysis of a complete index. Due to the Dow-Jones being the smallest index within the context of the American financial market, it also is the least intensive index to run.

Further, unlike other indexes where inclusion into an index is based on market capitalization, inclusion within the Dow-Jones is weighed based on the price of the stock. Although it should not make a major difference, companies within the Dow-Jones should be less likely to split their stock as this would reduce their position within the index. As stock splits can not correctly be classified, a dataset with less stock splits is preferable.

### 4.4.3 UnitedHealth and McDonald's

United Health and McDonald's were chosen due to them being a surprising pattern to appear together within the Dow-Jones codetable. UnitedHealth is a health insurance and healthcare provider within the United States. One could assume that McDonald's uses the services of UnitedHealth, but upon researching this does not seem to be the case. To investigate the differences between patterns generated by an index and patterns generated by stock, they were also run without the remaining index components.

### 4.4.4 UnitedHealth and Travellers

Although UnitedHealth specialized in health insurance and Travelers does not offer health insurance, both companies do offer insurance policies. This would make them a more obvious fit than McDonalds and UnitedHealth, and as such makes an interesting comparison with the results from UnitedHealth and McDonalds. It is for this reason that the stocks of both companies were run outside the Dow-Jones index. This way a comparison can be made between patterns found inside and outside the index and between intuitively unrelated and related stocks.

### 4.4.5 Bitcoin

Bitcoin was analysed due to it being a cryptocurrency and as such lacking market fundamentals. That is to say, Bitcoin can not be said to have any underlying value and is not used in day-to-day commerce nor accepted as official currency anywhere. As a result, its value lies within what people will pay for it. Given then that its values lies within perception, it is probably fair to assume that part of the perception is based upon historic value. Assuming then that historical value probably shapes current value, it is an interesting experiment to see if PYCCOLI can pick up signs of this being the case.

#### 4.4.6 Pepsi and Coca-Cola

As the final dataset, Coca-cola and Pepsi are run against each other in PYCCOLI's very own Pepsi Challenge. Of course, rather than trying to figure out which is which, the goal of PYCCOLI is to understand what ties them together and what makes them unique. Pepsi and Coca-Cola are especially well suited for this as they both are the household names within soda and both operate in the same beverage industry. It should though be noted that Pepsi also manufactures snacks and thus is more diversified than Coca-cola, who focuses solely on beverages. However, it remains perhaps the most ideal stock comparison and is used to show the performance of PYCCOLI when comparing two similar stocks.

## 5 Results

Within this section, first, a general overview of the results is given. This is then followed by a more in-depth analysis of each dataset.

| | **Support** | | **Length** | | **Time** | |
|---|---|---|---|---|---|---|
| **Dataset** | **Mean (*SD*)** | **Median** | **Mean(*SD*)** | **Median** | **Mean(*SD*)** | **Median** |
| Apple | 21(24) | 15 | 4(1) | 4 | 2(2) | 1 |
| Bitcoin | 20(24) | 11 | 4(2) | 4 | 3(2) | 2 |
| DOWJ | 14(11) | 10 | 3(1) | 3 | 1(1) | 1 |
| PEPCO | 98(90) | 63 | 2(1) | 2 | 1(0) | 1 |
| UNHTRV | 70(72) | 39 | 2(1) | 2 | 1(0) | 1 |
| UNHMCD | 62(57) | 36 | 2(1) | 2 | 1(0) | 1 |

*Table 1: Overview of the statistics of every dataset that was analysed through PYCCOLI.*

### 5.1 General overview

When looking at the overall results, a few general trends can be observed. First, as can be seen within table 3, every[10] analysed dataset returns patterns. Second, as can be seen within table 1, the support distribution of every dataset has a median that is lower than the mean. Indicating a positive skew within the support distribution. Third, the high standard deviation observed for every dataset hints towards certain patterns appearing less than ten times within the dataset, while others will appear in over 10% of the dataset. Taking all of these findings together, it appears as if a minority of patterns define the majority of the dataset for every dataset.

When looking at how long the patterns are, pattern length seems to be almost equal for every dataset. There is a slight exception however for Apple and Bitcoin who enjoy larger patterns. Apple and Bitcoin also seem to be the exception when looking at how much their patterns cover the dataset cover. They are the only two to be near 90% coverage, followed by the Dow-Jones index and Coca-Cola/Pepsi at 85%, with the remainder around 73%.

This pattern is reproduced when looking at encoding where the order remains the same. Apple and Bitcoin are the only ones with more than 30% compression, followed by the Dow-Jones index and Coca-Cola/Pepsi at around 25% and with the remainder being around 13%.

To conclude, Apple and Bitcoin benefit most from being compressed through PYCCOLI. However, every dataset could be compressed significantly, which shows that any type of stock analysis produces successful results through PYCCOLI.

[10]For simplified generalization the Apple stock run with only the closing price is excluded.

### 5.2 Apple

| **Pattern** | **Pattern Amount** |
|---|---|
| Open, Close | 9 |
| Volume | 29 |
| High, Low | 6 |
| Open, High, Low | 3 |
| High, Low, Close | 5 |
| Open, High, Close | 1 |
| Open, High, Low, Close | 44 |
| **Total** | 97 |

*Table 2: Pattern collections found within Apple.*

#### 5.2.1 Complete stock

97 patterns are found, combining many different elements of the stock together as can be seen in Table 2. Of note is that volume does not create patterns with any other sub-components of the stock, and that the most frequent type of pattern is one containing all price elements of the stock.

| **Analysis level** | **Dataset** | **Type of data** | **Non-singleton Pattern amount** | **Baseline MDL** | **Compressed MDL** | **Dataset compressed (%)** | **Length dataset** | **Dataset covered (%)** |
|---|---|---|---|---|---|---|---|---|
| Stock | Apple | Open, Close,High, Low, Volume | 97 | 31725.8 | 19149.28 | 39.64 | 7495 | 91.46 |
| Stock | Apple | Close | 0 | 31725.8 | 31725.8 | 0 | 1499 | 0 |
| Index (DOWJ) | DOWJ index | Close | 866 | 283712.14 | 205876.81 | 27.43 | 42000 | 83.23 |
| Stock | UNH, MCD | Close, Close | 17 | 9352.86 | 8236.97 | 11.93 | 3000 | 74.13 |
| Stock | UNH, TRV | Close, Close | 15 | 9098.41 | 7806.88 | 14.20 | 3000 | 72.33 |
| Stock | Bitcoin | Open, Close,High, Low, Volume | 102 | 30397.37 | 20602.32 | 32.22 | 7360 | 88.14 |
| Stock | KO, PEP | Close, Close | 13 | 9165.88 | 6950.37 | 24.17 | 3000 | 85.66 |

*Table 3: General results of every dataset that was analysed through PYCCOLI.*

Of further interest is that using only four patterns, 66% of the low and high price can be described. A simlair results is found within the volume results, as in that 30% of the volume can be described with only two patterns. Both of which can be seen within figure 4.

#### 5.2.2 Closing Price analysis

Finally, an analysis was run of the Apple stock using only the closing price, however no patterns were found and as such no compression occurred.

## 5.3 Dow Jones Index

The Dow-Jones data with its almost 900 patterns is much to vast to descriptively portray as a whole. As such, only subsections relating to stock comparisons from the codetable are reported.

#### 5.3.1 UnitedHealth and McDonald's

McDonald's and United Health appear together in ten patterns, however in only five of those do they appear without any other patterns. To better highlight the relationship between both stocks, only those five were used within figure 5. Together, they cover 26.7% of the dataset.

#### 5.3.2 UnitedHealth and Travelers

Displayed within figure 6 are the four patterns retrieved from the index analysis. Together they cover over 23% of their respective attributes.

Of interest is that the four patterns displayed within the figure, were the only four patterns where both UnitedHealth and Travelers appeared together. It should also be noted that these patterns were extracted from larger patterns. Within these larger patterns, the patterns contained up to five different stocks, the exact relations of which are shown within table 3.

| **Pattern type** | Pattern combinations | | | | |
|---|---|---|---|---|---|
| A | UnitedHealth | Travellers | United Technologies | IBM | |
| B | UnitedHealth | Travellers | Pfizer | | |
| C | UnitedHealth | Travellers | American Express | Verizon | |
| D | UnitedHealth | Travellers | McDonalds | Chevron | Merck & Co |

*Table 4: Patterns used for UnitedHealth and Travellers analysis.*

## 5.4 UnitedHealth and McDonalds stocks

Just like within section 5.3.1 only five patterns were taken, however in this case the patterns cover over 50% of the dataset. Consequently, it is a much higher pattern to cover ratio than when using the patterns found through the index. The patterns are displayed within figure 7.

## 5.5 UnitedHealth and Travelers stocks

The same experiment was run with UnitedHealth and Travelers to see if the outcome would be consistent with the results found for UnitedHealth and McDonald's. Within figure 8, just like in figure 6, only four patterns were used to cover the data. However when the stock patterns rather than the index patterns were used, the data ends up being covered for 48% rather than 23%. This is, again, a vast improvement in regards to pattern to cover ratio.

## 5.6 Bitcoin

Figure 9 shows the results from using the ten bitcoin patterns with the most support. The results show that already the top ten patterns cover over 55% of the total dataset.

| **Pattern** | **Pattern Amount** |
|---|---:|
| Open | 1 |
| Open, High | 2 |
| Open, High, Low | 17 |
| Open, High, Low, Close | 1 |
| Open, High, Close | 1 |
| Open, Close | 3 |
| High, Low | 13 |
| High, Low, Close | 13 |
| High, Low, Volume | 1 |
| High, Close | 2 |
| Low | 1 |
| Low, Close | 1 |
| Close | 17 |
| Close, Volume | 1 |
| Volume | 28 |
| **Total** | 102 |

Table 5: *Pattern collections found within Bitcoin.*

In examining how the patterns are distributed, as can be seen within table 5, a significant difference can be perceived from the patterns types observed within table 2.

For example, the Bitcoin distribution contains a pattern where volume is mixed with price, which does not occur within Apple's patterns. Also of note, is that the Bitcoin pattern tend to be more evenly distributed over all components of the Bitcoin price.

However, most Bitcoin patterns use at max 3 sub-components. This stands in contrast to Apple, where the usage of 4 sub-components seems to be the norm rather than the exception.

### 5.7 Coca-Cola and Pepsi

Finally before moving onto the discussion of the results, four patterns, where Pepsi and Coca-cola move in the exact same direction are used to give over 43% coverage of the total dataset. It seems for almost half of the time, PYCCOLI does truly believe Pepsi and Coke are the same thing. The reader can examine figure 10 to see whether they agree.

## 6 Discussion

The goal of this paper was to examine the effectiveness of PYCCOLI when used for financial data analysis. In this regard, the results section shows that the algorithm is highly successful in getting results. Patterns are found whether the data source is a single stock, a cryptocurrency, a comparison between several stocks, an index, or the comparison between an index and a stock. Within the current section, the meaning of these results will be expanded upon by giving explicit examples as to how they could be used.

The remainder of the discussion is divided into nine sections, the first six go in-depth on the results found within each type of data. The seventh section is a more global view at the general trend within the codetables, with the eighth section discussing what insights this generates regarding the PYCCOLI algorithm. The final section then concludes by suggesting future research directions.

### 6.1 Apple

There is perhaps no better starting than the Apple stock when only the closing price was used. The lack of patterns found indicates that PYCCOLI considers all the points within the dataset to be statistically independent from one another.

This is completely in line with the efficient market hypothesis, which predicts that all historic points of financial movements should be independent from one another. This results is therefor not surprising as it falls completely in line with prevailing financial analysis.

It would have been bizarre if PYCCOLI would have found patterns within the closing price, as such an obvious pattern would have already have been known. The results therefor provide a comforting sanity check of KRIMP-style algorithm's ability to ignore random patterns.

Of course this only highlights the exceptionally of the results found when using all sub-components of the stock. Bearing in mind that the efficient market hypothesis holds so well when only observing closing price, there is no intuitive reason as to why when other components of the stock are included, the financial movements stop being independent from one another.

Take, for example, the pattern:

$$\frac{\texttt{((0, (4, 4)), (1, (3, 2)), (2, (4, 3)), (3, (3, 3)))}}{\textit{(19, 8, 2)}}$$

The pattern can be translated as follows: The stock closing price will remain the same for two days, however each day the opening price will climb higher and higher. The first day will be marked by a higher lowest price and the second by a lower highest price.

On first glance this does not even seem to make sense, how can a stock keep the same price at the end of the day as the previous day but open with a higher price. The answer for this however, lies in that the stock has been selling well within the pre-market, which is how the stock price has managed to rise between the close and the open.

It is thus perhaps partly due to pre-markets that this pattern is observed, but what the reason is for pre-markets to buy while the high and low price converge is left to speculation. However, it is not hard to imagine how a market participant could gain a competitive advantage through knowledge of this pattern.

For example, a participant who normally would have bought the stock within the pre-market hour could rest assured with the knowledge that the stock will be at a lower price at the end of the day. Even better, the participant would know when the best moment to buy is. This is due to the lowest price between the first day and second day remaining constant. As such, the stock will be at its cheapest when the price on the second day equals the lowest price of the first day.

Although this specific example covers only 2.5% of the dataset, it still seems far too often to be deemed an exception. Furthermore, it is by no means obvious why the pattern moves the way it does. However, not all patterns found within Apple's codetable are this surprising.

A general look at the codetable[11] reveals that the majority of patterns can be explained through common sense reasoning. For these common sense patterns it is obvious they all follow an underlying trend together. In other words, they are all moving in the same direction. This is exemplified by patterns such as:

$\underline{\texttt{((0, (4,)), (1, (3,)), (2, (4,)), (3, (4,)))}}$ *(69, 4, 2)*

$\underline{\texttt{((0, (2,)), (1, (1,)), (2, (2,)), (3, (2,)))}}$ *(40, 4, 2)*

$\underline{\texttt{((0, (3,)), (1, (2,)), (2, (3,)), (3, (3,)))}}$ *(89, 4, 2)*

Which shows the Apple's stock, respectively, going up, going down and staying the same.

With regards to the distribution of the Apple patterns displayed within table 2, it is puzzling why the subcomponents of the stock integrate so well together. The price seems to contain a lot of structure regarding itself, but it is not evident why this should be the case. It is intuitive, however, that the high compression rate is most likely due to frequent usage of these large patterns that span across all components.

Finally, the example shown within figure 4 shows how only a few components are needed to almost fully describe the course of a stock. This is a huge benefit to any academic trying to model the stock behaviour, as only a small set of movements are needed to describe the entirety of the stock. Using a small set of descriptive patterns reduces model complexity, which makes the model easier to understand. Additionally, it also means there are fewer ways to recreate the entirety of the stock, this significantly reduces the amount of possible hypothesis and therefor less experiments are needed to produce a working model.

In conclusion, the results from the Apple stock show the usefulness of using patterns to describe stocks and highlights that interesting frequent patterns can be found within historical stock data.

## 6.2 Dow-Jones index

At first glance the codetable of the Dow-Jones[12] is different from all the others due to it mainly having singletons within the top 25 spots of the codetable, with the exceptions of:

$\underline{\texttt{((0, (3, 3)), (5, (3, 3)))}}$ *(92, 4, 2)*

$\underline{\texttt{((9, (3,)), (17, (3,)))}}$ *(88, 2, 1)*

Looking at the definition file[13] shows that the patterns relate to Boeing and 3M, and IBM and Walmart respectively. It is surprising to again find a pattern that has a time span that lasts two days, even if the patterns do not indicate any exceptional movement.

Another pattern that stands out within the codetable is:

$\underline{\texttt{((24, (2, 3)))}}$ *(60, 2, 2)*

Which represents the Coca-Cola stock lowering and then stabilizing, this could serve as a warning sign for an investment firm that the company's stock price is slowly getting weaker over time.

Interestingly, the Motley Fool reports just that for the Coca-Cola stock. Although this is but a single example, it was often the case that patterns found within the codetable would, when researched, have a corresponding analyst report dated after the patterns occured.

[11]Link to Apple codetable

[12]Link to Dow-Jones codetable

[13]Link to Dow-Jones definitions

As such, the patterns from the PYCCOLI algorithm could be used to not only to get objective financial news, but to get it before market analysts report it. Trading based on news information before analysts report on it would be a major benefit to any trading firm.

## 6.3 UnitedHealth and McDonald's

Within the analysis of UnitedHealth and McDonald's both the results of the index analysis and the stock comparison analysis are taken. This is to highlight how the patterns that both return complement each other.

When looking at the graph within figure 5 and figure 7 the first thing that stands out is that both figures have the following pattern in common:

```
((0, (4,)), (1, (4,))) (161, 2, 1)
```

Which is the red pattern in figure 5 and the blue pattern in figure 7.

This pattern on its own covers over 10% of the dataset, for which there is no obvious explanation. On first thought one could hypothesized that the underlying American economy is growing, however then other stocks would be expected to be present within the patterns originating from the index.

What further stands out is that when looking at both figures, the patterns complement each other. This can be seen within figure 11, where red symbolizes patterns that are from the index while blue are from the stock analysis, leading to 66% of the dataset being covered in comparison to the 50% covered by the stock patterns and 25% covered by the index patterns. In other words, when combining the patterns used within each graph a higher amount of the stock is covered.

This should not be expected to be the case by default, as the patterns could 'be in each others way' and actually prevent each other from forming meaningful patterns. Rather it seems that each set of patterns chosen describes a different part of the relation between McDonald's and UnitedHealth.

The meaning behind these patterns could be that the patterns found through the index show how the stocks relate towards the economy as a whole. In contrast, the patterns found when only analyzing the stock could highlight the micro economics between both stocks.

All in all, an interesting relation is found between McDonald's and United Health, the cause of which is unknown.

## 6.4 UnitedHealth and Travellers

The results from UnitedHealth and Travellers seem very similair to the ones found within the previous section. The figures 6 and 8 again complement each other as can be seen within figure 12, where again red symbolizes patterns that are from the index while blue are from the stock analysis.

This time the combination leads to 56% of the dataset being covered. This is again an improvement when compared to the 48% covered by the stock and the 23% covered by the index. However, the improvement is not as large as for UnitedHealth and McDonald's.

Interestingly enough McDonald's appears within pattern D in table 4. It seems that no matter how hard UnitedHealth tries, it cannot shake loose from its connection with McDonald's.

When further comparing it with the UnitedHealth/McDonald's results another unpredicted finding stands out, that is, although the encoding of UnitedHealth/Traveller is better than the one of UnitedHealth/McDonald's the amount of data covered is lower. This is an interesting finding and likely can be attributed to UnitedHealth/Traveller using less patterns to encode the dataset.

As such, it seems that the insurance companies have more shared structure than UnitedHealth/McDonald's. However, both datasets are significantly helped by using both index and stock patterns.

## 6.5 Bitcoin

Bitcoin was expected to be the best performer and ended up as second best. It seems that the sub-components allow for many parts of the price to be explained, as can be seen in figure 9.

Within the graph, what is perhaps more interesting than the patterns relating to stock price are the patterns relating to the volume. The patterns returned seem very adapt at covering the days when volume drops heavily. The natural extension would be to see if these heavy drop patterns link with any patterns within the price, this however does not seem to be the case.

When looking at the codetable[14], the top three patterns cover almost 30% of the dataset:

```
((1, (2,)), (2, (4,))) (141, 2, 1)
((1, (3,)), (2, (4,))) (132, 2, 1)
((1, (2,)), (2, (3,))) (132, 2, 1)
```

[14]Link to Bitcoin codetable.

The three patterns are also highly similar, as they could all be represented as the pattern `((1, (2,)), (2, (4,)))`, which when looking at the definitions[15] is a pattern between the high price and the low price. It is certain that with how frequent this pattern is, it would be interesting to use it as a base for a research into the cryptocurrency, either as a private or public entity.

Of further interest is that unlike the Apple stock where most patterns cover all 4 sub-components of price, table 5 shows that within the Bitcoin stock the norm seems to be 3 sub-components. Furthermore, these three sub-components do not only include pricing sub-components but in two cases mix volume and price. These are the following two patterns:

$$\frac{\texttt{((1, (2,)), (2, (4,)), (4, (1, 1, 2, 2)))}}{\textit{(11, 6, 4)}}$$

$$\frac{\texttt{((3, (3,)), (4, (1, 3, 2)))}}{\textit{(3, 4, 3)}}$$

Of which both seem to indicate the price stabilizing, and the volume correspondingly reducing. Possibly hinting towards a period of calm after erratic buying and selling.

It is interesting to speculate why only the Bitcoin patterns contain volume within their patterns. It could however be due to the lower market capitalization of the currency, and as such its volume being more sensitive to price changes.

In conclusion, the Bitcoin price returns many interesting patterns that seems to hint towards the currency's instability. This would make sense as it is a small and highly speculated product. However, it remains to be seen if that is the whole story as to why this is the only dataset to mix pattern with its volume.

## 6.6 Pepsi and Coca-Cola

The analysis of Pepsi and Coca-Cola is one of the most interesting results. When looking at the patterns within figure 10, the movement between the stocks is nearly identical. This is confirmed when looking at the patterns used (where Pepsi is 0, and Coca-Cola is 1):

| | |
|---|---|
| `((0, (1,)), (1, (1,)))` | *(72, 2, 1)* |
| `((0, (2,)), (1, (2,)))` | *(280, 2, 1)* |
| `((0, (3,)), (1, (3,)))` | *(210, 2, 1)* |
| `((0, (4,)), (1, (4,)))` | *(84, 2, 1)* |

As they all show the two stocks going in the same direction.

[15] `Link to Bitcoin definitions.`

When analyzing the codetable[16], the Pepsi side of the pattern tends to overall be in a more positive direction than Coca-Cola. This can be seen in patterns such as `((0, (3,)), (1, (2,)))` and `((0, (4,)), (1, (3,)))`, which occur within 10% en 15% of the dataset respectively. It is unsure whether this is due to Pepsi doing on average better than Coca-Cola or due to the real median value of Coca-Cola being 2. However, this can be ascertained with a quick glance at the stock price of both, from which it can be determined that Pepsi really has been outperforming Coca-Cola over the 2013-2019 period.

Even with this minor difference, the overall results strongly point towards the two soda manufacturers stock moving together in harmony. Intuitively this makes a lot of sense, as it would be expected for the soda market to move in trends. For example, it is not hard to imagine that soda as a whole suffered when the United Kingdom imposed a sugar tax. And with governmental health policies around the world becoming stricter every day, it is likely that this harmonic trends will not stop anytime soon.

It is easy to imagine how the results from this analysis could be used. For example, a market analyst might have a large position in Pepsi and want to get an idea of the unique risk Pepsi faces within the soda industry. For this, they would only need to analyze the parts not covered by patterns within figure 10.

Or from an academic perspective, a researcher might be interested in what policy changes 'hurt' a soda companies market capitalization the most. As such, they would only need to add legislative changes to the timeline of 10, to see where they effectively curbed the market value of Pepsi and Coca-Cola.

As a final note, although no metric for this is offered, the shapes of the patterns between Pepsi and Coca-Cola seem to be the most similar of any of the analyzed datasets. As such, it really appears as if PYCCOLI is picking up underlying information that dictates the price movements of both of these stocks.

## 6.7 An overview of the results

A point should be made as to how surprising it is that patterns are found in the first place. PYCCOLI only uses historical price and volume to construct its patterns. As a reminder, the weakest form of the efficient market hypothesis states that using historical price, no information

[16] `Link to Pepsi and Coca-Cola's codetable`

can be obtained regarding future price.

This stands in contradiction to the results found. The codetables on multiple occasions show that there are patterns that occur over multiple days. These observations have also been found within other financial papers and have been dubbed market anomalies. However, they are not anomalies within the reported codetables. Rather, they represent the underlying structure of the stock.

Being part of the underlying structure of the stock means that they could be seen as representing a 'memory' occurring within the stock price. That is to say that, in certain cases, the regular stock market, just as was seen within cryptocurrency, does look at the past to determine future price.

Also of interest, is that the codetables do not only find the 'normal' patterns but also finds the 'weird' patterns. This can be seen, for example, within Apple's codetable, where the expected patterns of the price moving in the same direction are found, but also abnormalities are found that lack obvious explanation.

This is further supported by codetables such as the one from McDonald's and UnitedHealth, where there is no obvious answer as to why McDonald's keeps cooccuring in the pattern even when looking at the index patterns of UnitedHealth and Travellers.

On a different note, it is surprising to see that Apple and Bitcoin compress better than Pepsi and Coca-Cola. Perhaps this is due to the patterns of Pepsi and Coca-Cola being shorter than those of their competitors, Bitcoin and Apple. Conceivably, it is also the reason as to why Apple is compressed more than Bitcoin. In other words, the fact that Apple's patterns are longer and always target all four of the sub-components of price offsets the advantage Bitcoin has of being able to mix volume and price together.

Finally, when taking a big picture view at the results, a consistent meta-pattern start to emerge. This meta-pattern can best be described as indicating that the codetables are geared towards best explaining the dataset as a whole. To elaborate, rather than describing all the interrelations between the attributes and reporting a descriptive summary of the interrelations, the results seem to describe some greater underlying trend.

This is, for example, displayed within the Dow-Jones code table. The patterns found between UnitedHealth, McDonald's and Travellers have almost nothing in common with the results found when analyzing the stock. This would not be the case if the description was simply a sum of the parts.

Rather, the codetables describe the underlying information of their respective dataset. In the case of the stock comparisons, it is the parts the stocks have in common. However in the case of the index, the codetable describes how the stock partakes to form the index, and as such what the stock have in common does not matter. Indicating that what structures on small financial dataset, does not implicitly structure a larger financial dataset that it is part of.

To conclude, the PYCCOLI method provides highly effective compression of the datasets and presents an interesting range of results. The patterns that result from the compression are well suited for data analysis and reveal interesting underlying relations within the data. Most importantly however, the results prove that stock data of any type contains information that can be extracted into concrete patterns.

## 6.8 On the PYCCOLI implementation

PYCCOLI does not make connections by random chance. Due to it being an extension of the KRIMP algorithm, it in essence sifts out all of the order from the dataset, leaving only noise behind.

A couple of caveats come in to play however; first, as shown within the method PYCCOLI does not create every combination possible. PYCCOLI also does not allow gaps within its patterns and for these two reasons there may be a lot of interesting patterns left out of the codetable.

These patterns could very well compress the dataset much better than the results achieved by PYCCOLI. As such, PYCCOLI must be taken as the worst-case scenario codetable. Although this sounds bad, it actually is encouraging as it means that there is ample space left for improvement.

Second, accurate rendition of the structure underlying the dataset may require multiple levels of analysis. As can be seen from the results, the patterns between two stocks are very different from the patterns returned from an index where they are both part of.

It is thus possible that there is a significant difference between the structure of the whole and the structure that results from a summation of the parts. As a consequence, to offer a complete image of how the stocks are interconnected to others as well as to themselves, it might be necessary to run the stocks not only within their index but also by themselves and perhaps even with stocks within the same sector.

Another point that must be addressed is that of cover order. Although no results are given in regards to differences in cover order, it should be mentioned that during experimentation cover order made a significant impact on the patterns found. The solution proposed by KRIMP is that ideally all cover order should be run, however KRIMP also very correctly follows this point with the observation that this makes for a very heavy computation. Seeing how the current algorithm already had difficulty running through the dataset with daily intervals rather than hourly or minute interval, it is hard to imagine this as a possibility.

Perhaps an alternative would be to run a certain amount of different orders and compare the results, or to order the patterns based on interestingness. The latter was done within this paper to prioritize patterns that have a longer time length over patterns that span over multiple attributes. Finding patterns that last over multiple days diminished the compression performance in comparison to simply finding the longest patterns. However, no patterns that spanned multiple days were found until the cover algorithm prioritized time length over pattern length.

Furthermore, as a consequence of taking the difference in comparison to the previous day the resulting data points are in essence always linked to the previous day (this can also be observed within the graphs). As such, the patterns in effect capture movement within the dataset. Thus they are not focused on individual events happening at a fixed point but actual movement. Although this may seem obvious and has been mentioned before within the method, it still is interesting that PYCCOLI in effect captures the activity of the data.

Some final remarks on the PYCCOLI algorithm are that despite its flaws, there is genuine potential for usage of the algorithm as a financial analysis tool. The findings displayed within the result section are but very few of the total results produced by the algorithm and a thorough exploration of these findings has not been done. As such, there is genuine merit to continued use of the algorithm, or at least of DITTO-like algorithms within finance.

### 6.9 Suggestions for future work

It stands to reason from the points mentioned within the previous sections that PYCCOLI could very well be used as a starting point when analyzing a financial dataset. Within this domain there are many applications that PYCCOLI could have, for example as a method to find similarity between two stock. However, it could also be used as a tool to compare markets overall. As such, the European market could be compared to the American one to see if there is a difference in compression. Of course this could be done between any two markets. The results from which would be a metric that gives an objective measure as to how structured the market is as a whole.

Some further more obvious research that has not been explored is to, for example, run the algorithm on the US market as a whole. Or to run minute data, rather than daily data. It would also be interesting to see if, when only using small market capitalization stock, the price and volume would be combined within the patterns as was the case within Bitcoin.

Finally, it would be interesting to find out how well the results from PYCCOLI could be used for classification. KRIMP shows within its paper [2] that it is on par with the best of classifiers. The financial industry, as well as academia, lacks such classifiers, making PYCCOLI an obvious choice. As such, applying PYCCOLI to classify any kind of financial data should provide interesting results.

Beyond its use within academia or the private domain there is also the question of expanding the algorithm to allow patterns with gaps. Or perhaps even more interesting, to look at patterns independent from the attribute. That is to say, to encode the entire dataset as a single cluster and then allow patterns to fit over any lines as long as the shape is correct.

Given the MDL formulation, it would be assumed that this kind of universal pattern would give shorter MDL. Furthermore, the pattern, rather than describing a relation between two attributes, would be describing a universal pattern within the financial market that transcends a single stock.

As such many paths have been suggested for future work, however the simplest and perhaps most entertaining is simply running PYCCOLI on your own time series and being surprised by the results.

## 7 Acknowledgements

The author wishes to thank Arno Siebes, for his excellent supervision and continued guidance throughout the project. His parents Corinne Girard and Giovanni Vaglienti and his sister Clementine Vaglienti, for the love and support they have continually offered. Folco Vaglienti, for the inspiration to fully immerse oneself into hard work and science. And finally, Terri Modrakowski and Dr. Amelia Beans for their continued love and support.

## References

[1] R. Bertens, J. Vreeken, and A. Siebes, “Keeping it short and simple: Summarising complex event sequences with multivariate patterns,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 735–744, 2016.

[2] J. Vreeken, M. Van Leeuwen, and A. Siebes, “Krimp: mining itemsets that compress,” *Data Mining and Knowledge Discovery*, vol. 23, no. 1, pp. 169–214, 2011.

[3] K. Smets and J. Vreeken, “Slim: Directly mining descriptive patterns,” in *Proceedings of the 2012 SIAM international conference on data mining*, pp. 236–247, SIAM, 2012.

[4] N. Tatti and J. Vreeken, “The long and the short of it: summarising event sequences with serial episodes,” in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 462–470, 2012.

[5] E. F. Fama, “The behavior of stock-market prices,” *The journal of Business*, vol. 38, no. 1, pp. 34–105, 1965.

[6] E. F. Fama, “Random walks in stock market prices,” *Financial analysts journal*, vol. 51, no. 1, pp. 75–80, 1995.

[7] J. Lin, E. Keogh, L. Wei, and S. Lonardi, “Experiencing sax: a novel symbolic representation of time series,” *Data Mining and knowledge discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[9] R. Agrawal, R. Srikant, *et al.*, “Fast algorithms for mining association rules,” in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, pp. 487–499, 1994.

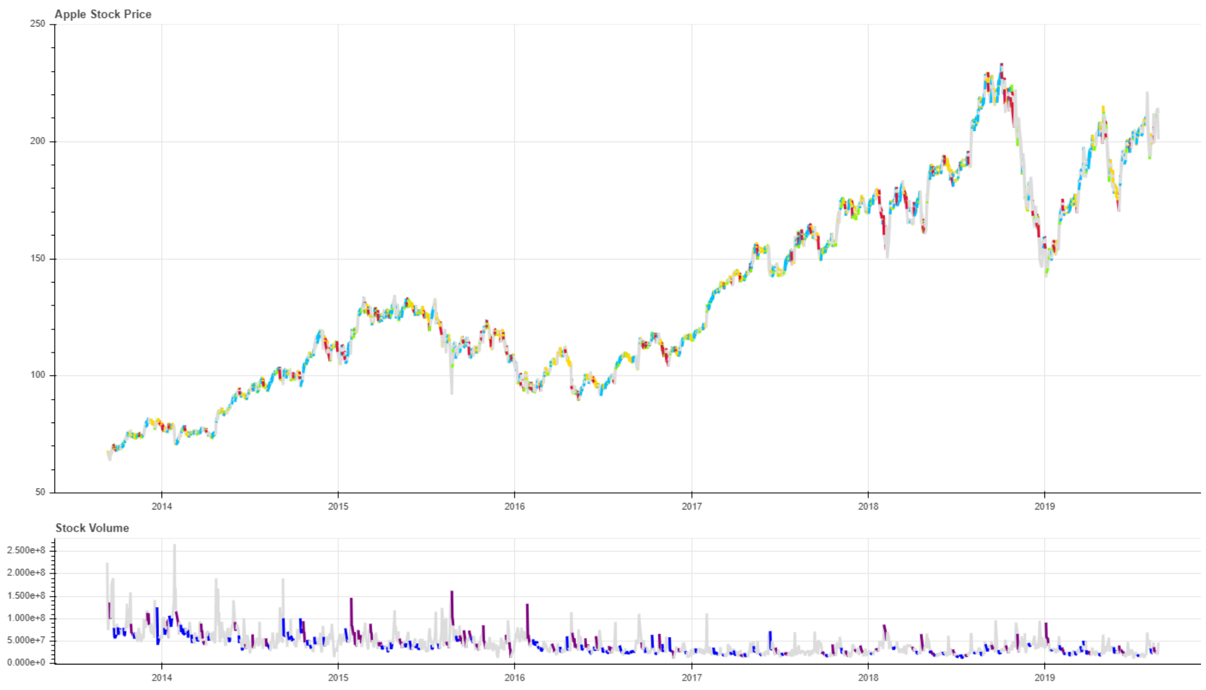

Figure 4: *The High, Low and Volume of Apple stock, full version online*

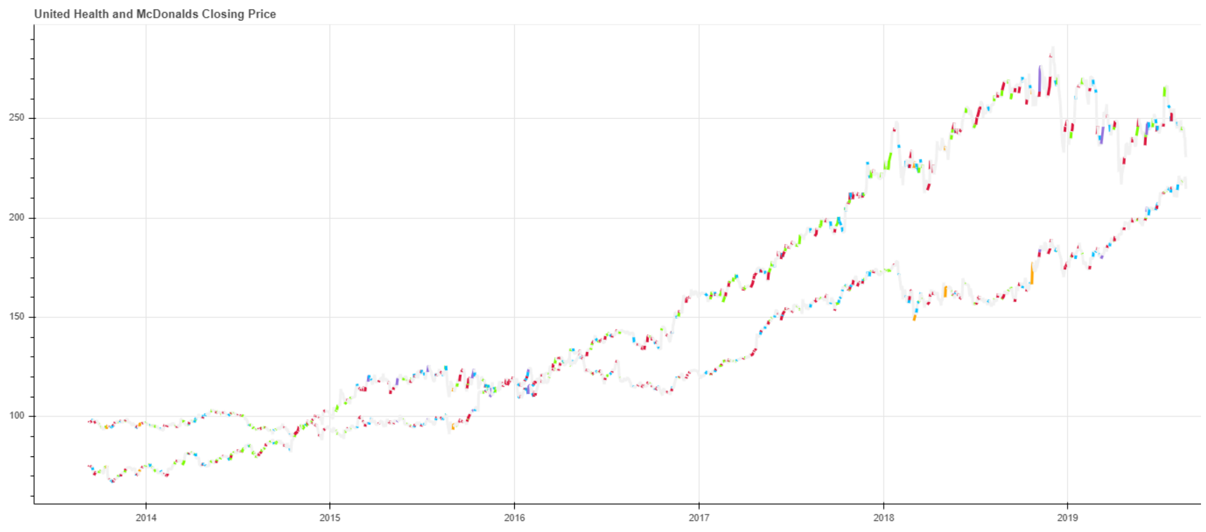

Figure 5: *The Close of McDonalds and UnitedHealth stock pattern found through the Dow Jones index, full version online*

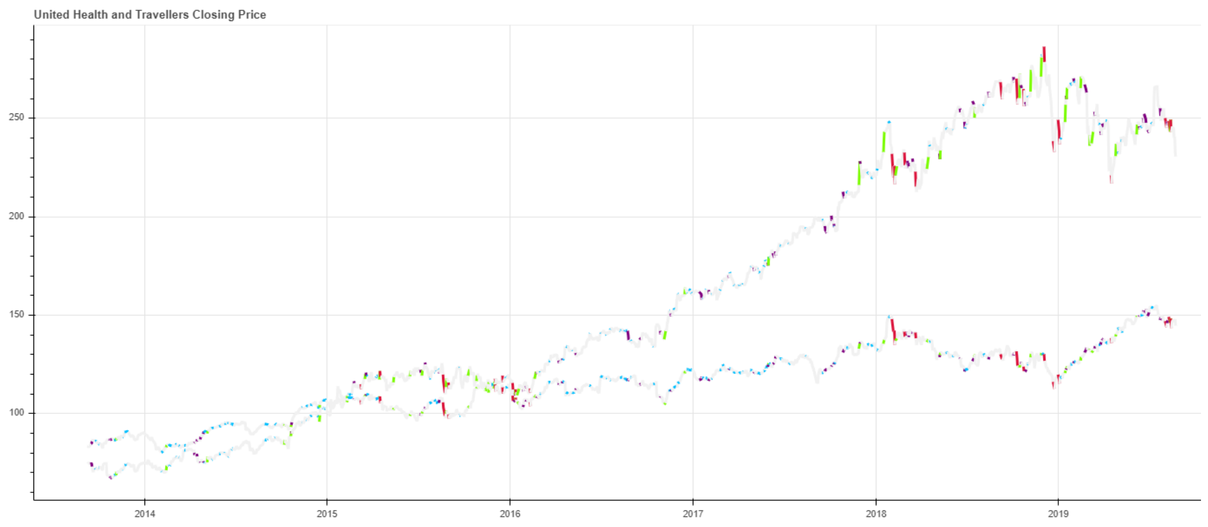

Figure 6: *The Close of UnitedHealth and Travellers stock with the pattern taken from the total index, full version online*

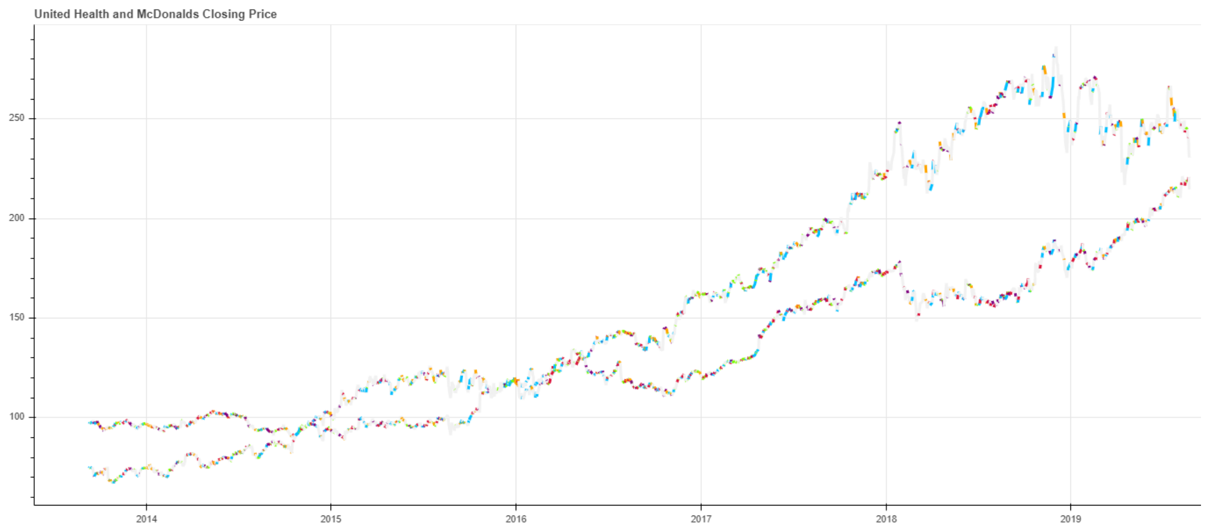

Figure 7: *The Close of McDonalds and UnitedHealth stock pattern found through their own closing price, full version online*

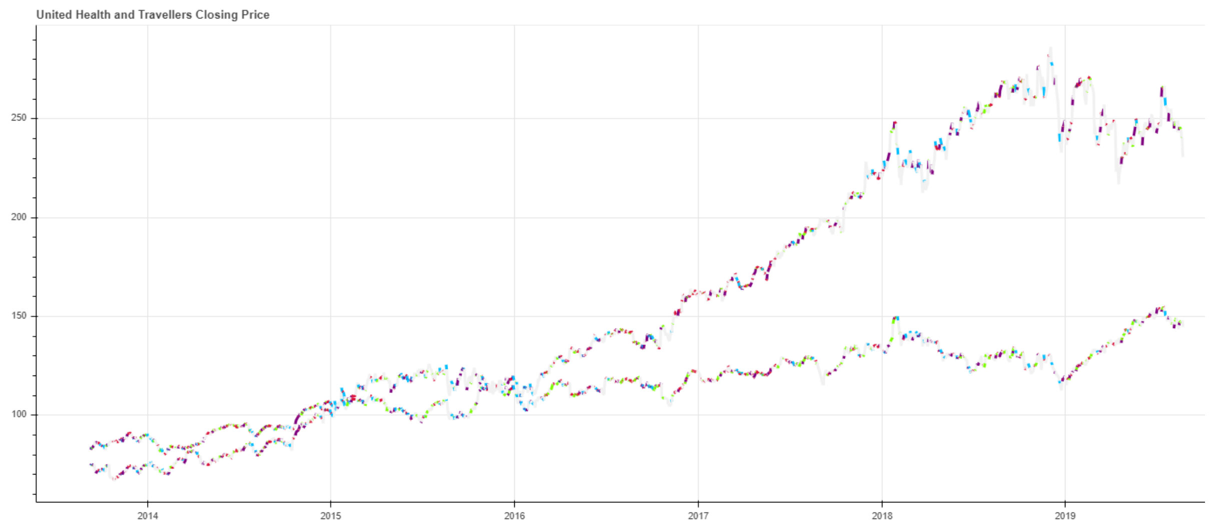

Figure 8: *The Close of UnitedHealth and Travellers stock where the patterns are found through the stock, full version online*

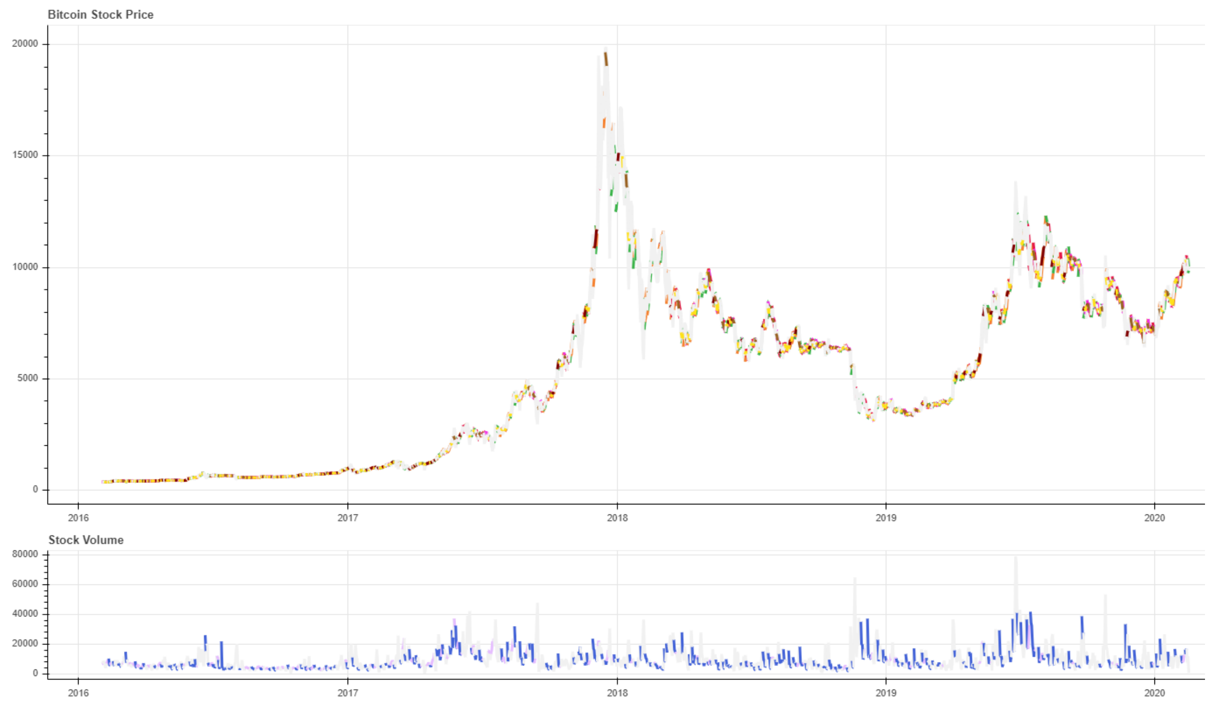

Figure 9: *The Open,Close, High, Low and Volume of Bitcoin full version online*

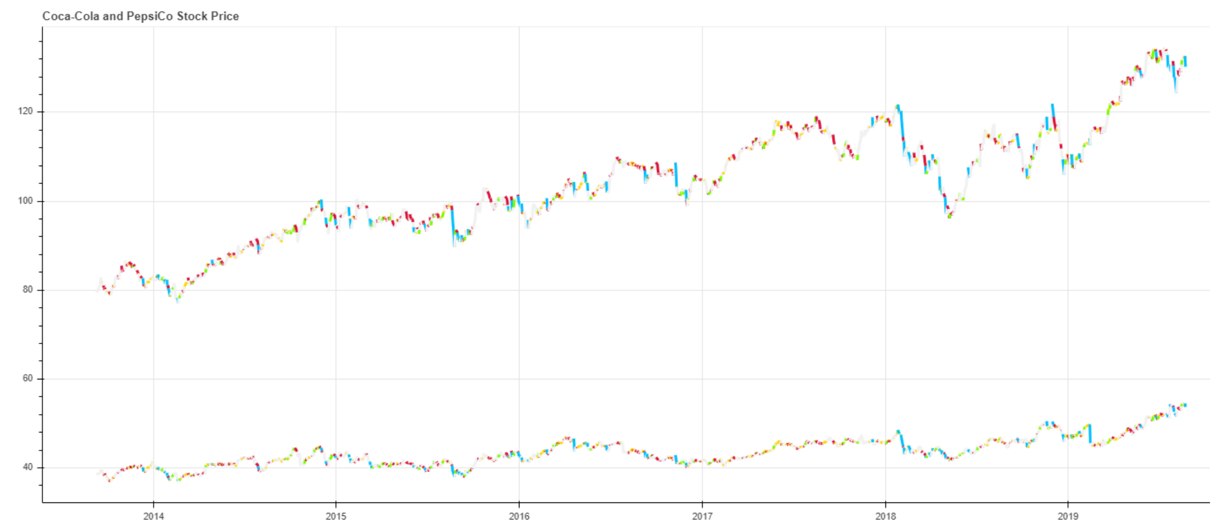

Figure 10: *The close prices of Pepsi and Coca-Cola full version online*

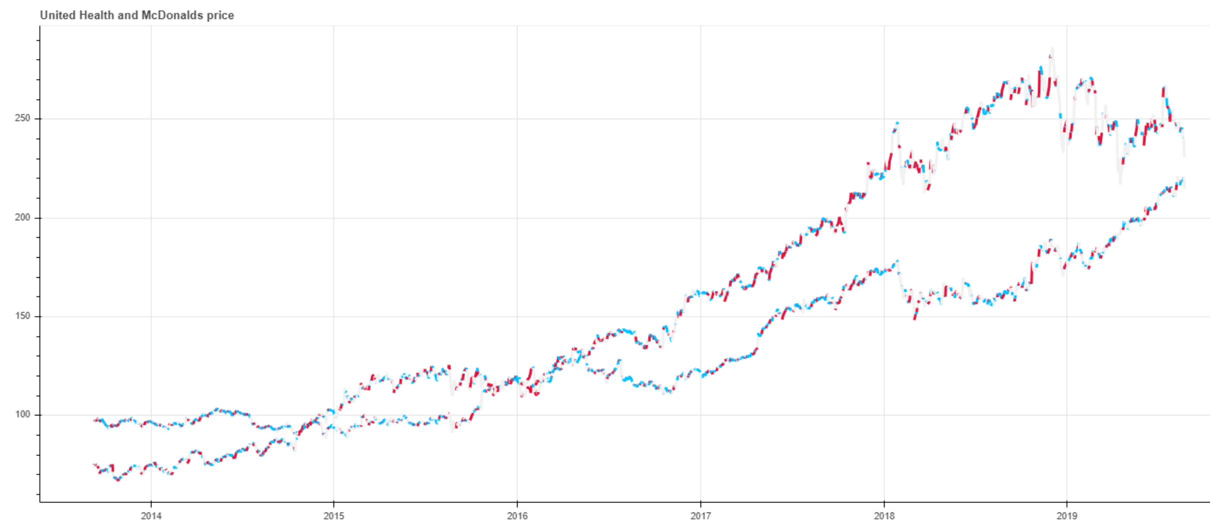

Figure 11: *The close prices of McDonald's and UnitedHealth where index patterns are in red and stock patterns are in blue full version online*

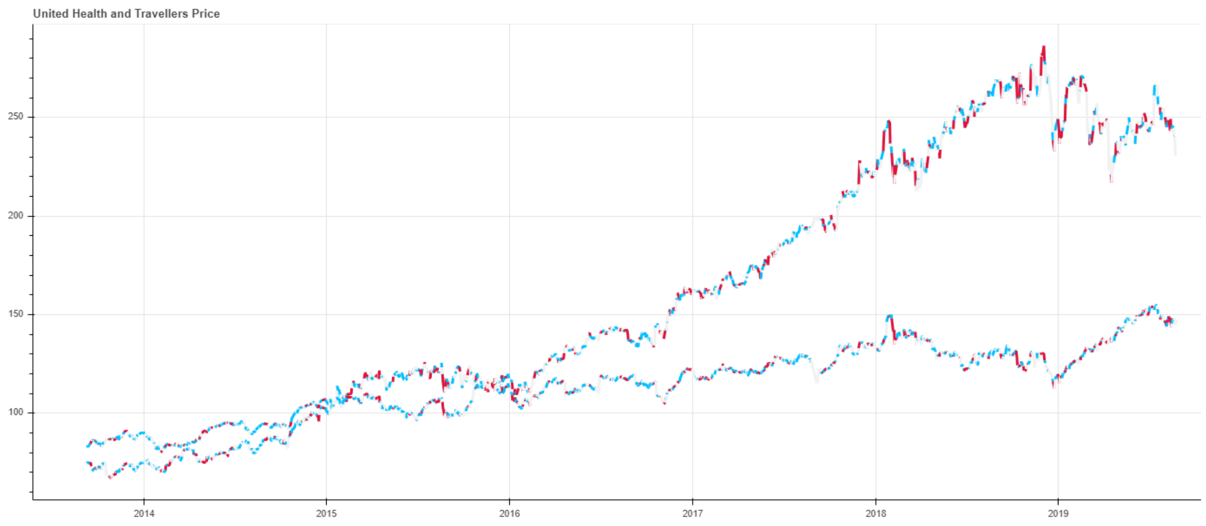

Figure 12: *The close prices of Travellers and UnitedHealth where index patterns are in red and stock patterns are in blue full version online*