Time Series Analysis with Pandas

There is a time to wake up, a time to go to bed, a time to relax, and a time to get going. Indeed, time has a substantial effect on our lives and the decisions we make, and as FinTech professionals our ability to analyze and understand events over time is paramount in supporting better decision making. This is known as **time series analysis**—the exploration and understanding of data that is ordered over time (**time series data**). By analyzing time series data**,** we are able to report and visualize trends, detect anomalies, and quantify relationships between metrics.

In this lesson, you will learn to prepare, visualize, interpret, and decompose time series data to identify the three major components of time series analysis—trend, seasonality, and noise. You'll also explore how to use correlation to recognize and quantify relationships, and how to qualify and understand your findings by referencing real-world examples.

MVRV: A New Method of Analysis

Before we start our analysis, let's take a moment to learn a little bit about our dataset and some new metrics we will be exploring.

In 2018, Nic Carter of Castle Island Ventures presented his newly-termed concept **realized value** (also referred to as *realized cap*), representing an indicator that would help adjust for lost coins and coins used for **hodling** in cryptocurrency markets.

- *Hodling*: term commonly seen as derived as a misspelling of HOLD and forming the acronym HODL (Hold On for Dear Life). It refers to an investor holding a cryptocurrency for future profits and not selling.

Soon after, the **MVRV** (Market-Value-to-Realized-Value) ratio was introduced as an instrument to determine whether Bitcoin was over or undervalued. To understand how this works, let's define a couple of terms.

- *Market Value (Market Capitalization)*: classically, the value of a company that is traded on the stock market, calculated by multiplying the total number of outstanding shares by the present share price. In crypto markets, it represents the total value of a cryptocurrency, calculated by multiplying the total number of coins in circulation by the current coin price.

- *Realized Value (Realized Capitalization*): used exclusively for cryptocurrency analysis, a variation of market capitalization that values each coin based on the price that it was last transacted—its acquisition price—as opposed to its current value. As such, it is often used to contrast the value of coins in their network (sometimes referred to as *fair value*) to their overall market value.

To illustrate the concept, let's assume you have 20 BTC in your digital wallet and they were acquired in three different transactions:

1. 5 bitcoins were bought at $5,000.00/bitcoin
2. 10 bitcoins were bought at $15,000/bitcoin
3. 5 coins were bought at $30,000/bitcoin

If the price of bitcoin were currently $25,000, then we could calculate our market value as:

(20 x $25,000) = $500,000

Our realized value is found by multiplying by the acquisition cost of all the bitcoins in your wallet. In this case we find that realized value is:

(5 x $5,000) + (10 x $15,000) + (5 x $30000) = $325,000

This means that you paid average value of $325,000 / 20 = $16,250 per bitcoin, though current market prices are $25,000 per bitcoin, suggesting overvaluation of the market.

The MVRV ratio is calculated by dividing the market value by the realized value on an ongoing basis. In general, Bitcoin is generally considered overvalued when MVRV exceeds 3.7, and is considered undervalued when MVRV is at or below 1.

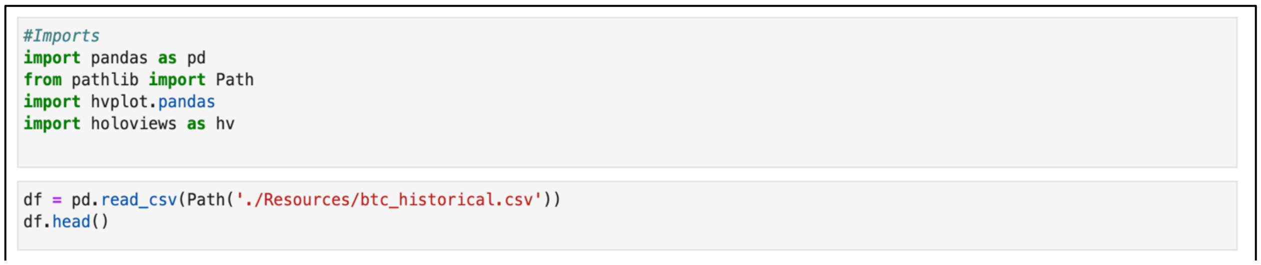
Let's now move to preparing our data for time series analysis, allowing us to identify opportunities when Bitcoin has been undervalued and overvalued the past 5 years, and to identify patterns that may allow us to predict buy/sell opportunities in the future!

Making Time: Using Pandas to Prepare Time Series Data

FOLLOW ALONG:

Access to all CSVs can be found at the following link:
https://github.com/AbsurdSophist/Time_Series_Analysis_Lesson

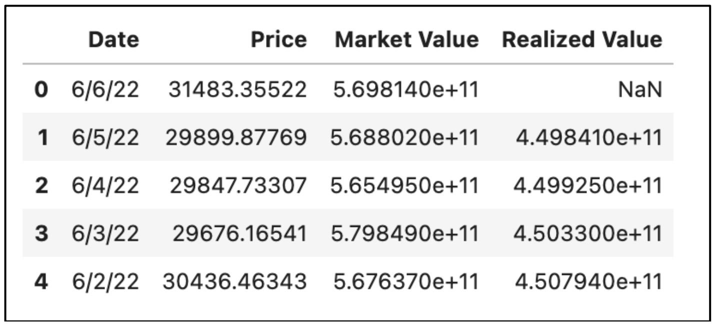
Let's start our time series analysis by setting up our initial imports and using Pandas to read our Bitcoin CSV file ("btc_historical.csv") into a DataFrame, as shown below:



```
#Imports
import pandas as pd
from pathlib import Path
import hvplot.pandas
import holoviews as hv
```

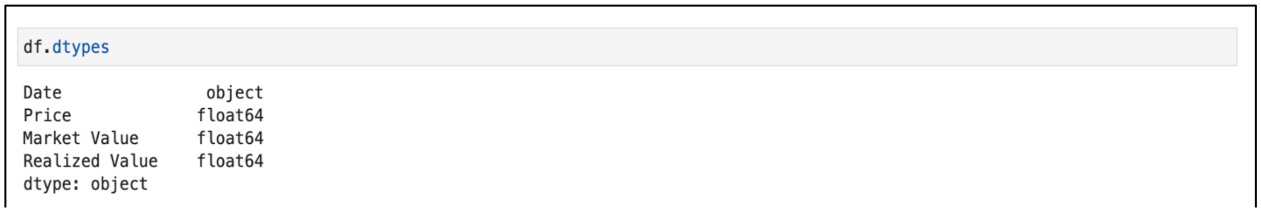
```
df = pd.read_csv(Path('./Resources/btc_historical.csv'))
df.head()
```

After running our code, we produce the following image showing the first five rows of the resulting DataFrame:



	Date	Price	Market Value	Realized Value
0	6/6/22	31483.35522	5.698140e+11	NaN
1	6/5/22	29899.87769	5.688020e+11	4.498410e+11
2	6/4/22	29847.73307	5.654950e+11	4.499250e+11
3	6/3/22	29676.16541	5.798490e+11	4.503300e+11
4	6/2/22	30436.46343	5.676370e+11	4.507940e+11

Before moving forward, let’s ensure that our “Date” column is a **datetime** object by using **dtypes**, as shown below:



```
df.dtypes
```

```
Date               object
Price             float64
Market Value      float64
Realized Value    float64
dtype: object
```

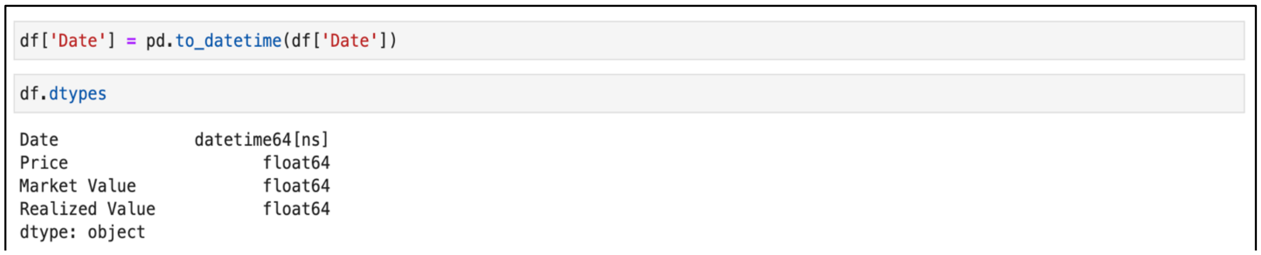
As you may have noticed, our “Date” column is an **object** data type. But don’t we need it to be a **datetime** series? What’s going on here?

In Pandas, an **object** data type is the equivalent to a string or mixed data type in Python and will not allow us to let us take advantage of our built-in datetime functions. So, to start, let’s try converting our “Date” column into a datetime data type.

PRO TIP:

Always be sure to check your data types when working with new data. Having the wrong data type can lead to values not functioning as expected. For example, the number 2 can be read in as a string or as an integer in Python, but you can only perform mathematical functions on 2 if it is an integer. datetime data types are the same and failing to convert can lead to reduced functionality and intended errors when running your code.

Fortunately for us, Pandas has the **to_datetime** function, which will allow us to convert our "Date" column data from an **object** to a **datetime** data type. We can then use take advantage of our time-related functions as we begin our time series analysis.
The code below shows how we can convert our data into a **datetime** series using the **pd.to_datetime** function, as well as verify that the data type transformation has taken place:

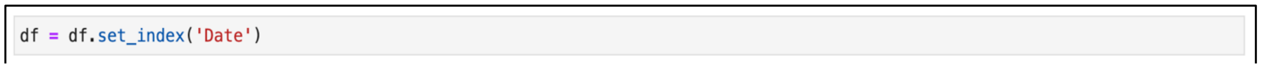


```
df['Date'] = pd.to_datetime(df['Date'])
```

```
df.dtypes
```

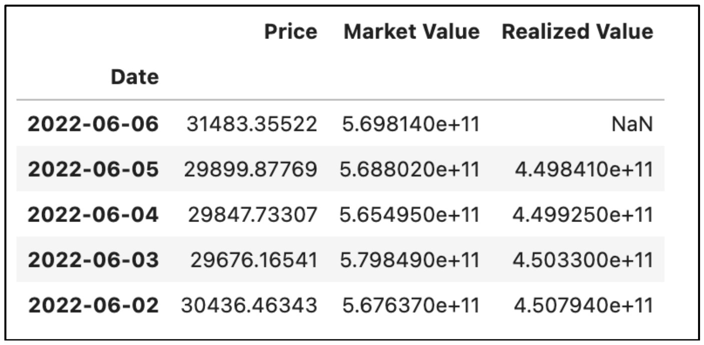
```
Date              datetime64[ns]
Price                    float64
Market Value             float64
Realized Value           float64
dtype: object
```

With our data read in and our datatypes verified, there is only one thing left for us to do: set up our **DatetimeIndex**. Lucky for us, Pandas makes this a breeze. The **set_index** function will allow us to pass our "Date" column name as the parameter, as shown below:



```
df = df.set_index('Date')
```

This results in, you guessed it, a **DatetimeIndex**, as can be seen in the following image:



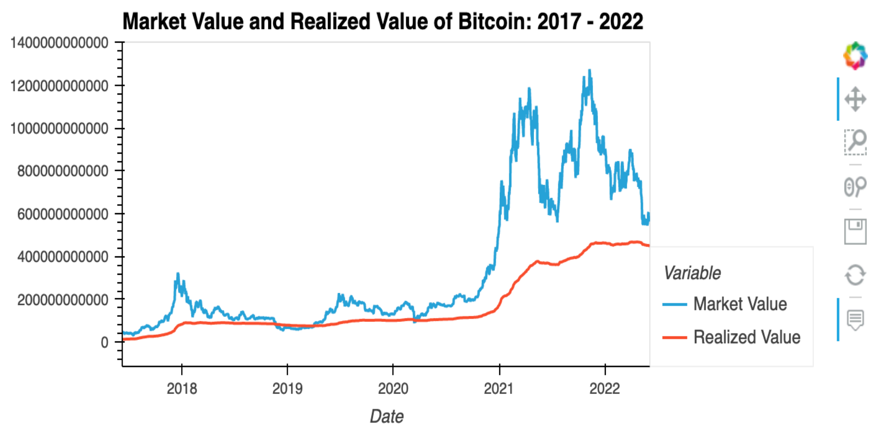
	Price	Market Value	Realized Value
Date			
2022-06-06	31483.35522	5.698140e+11	NaN
2022-06-05	29899.87769	5.688020e+11	4.498410e+11
2022-06-04	29847.73307	5.654950e+11	4.499250e+11
2022-06-03	29676.16541	5.798490e+11	4.503300e+11
2022-06-02	30436.46343	5.676370e+11	4.507940e+11

Now that we set up our data with a **DatetimeIndex**, we can use Pandas slicing features to decompose and analyze the data across time.

As Time Goes By: Trends, Seasonality, and Noise

Let's begin by plotting the market cap and realized cap as shown below:

```
df.hvplot(y = ['Market Value', 'Realized Value'],
          title = 'Market Value and Realized Value of Bitcoin: 2017 - 2022',
          yformatter = '%.0f')
```



What do you see? Are there any patterns we should notice? Are there any anomalies that we can examine more closely?

We can start by looking at whether our market and realized values have an increasing or decreasing **trend** over time.

- *Trend*: shows whether the data is consistently increasing (upward trend), decreasing (downward trend), or constant (no trend) over time.

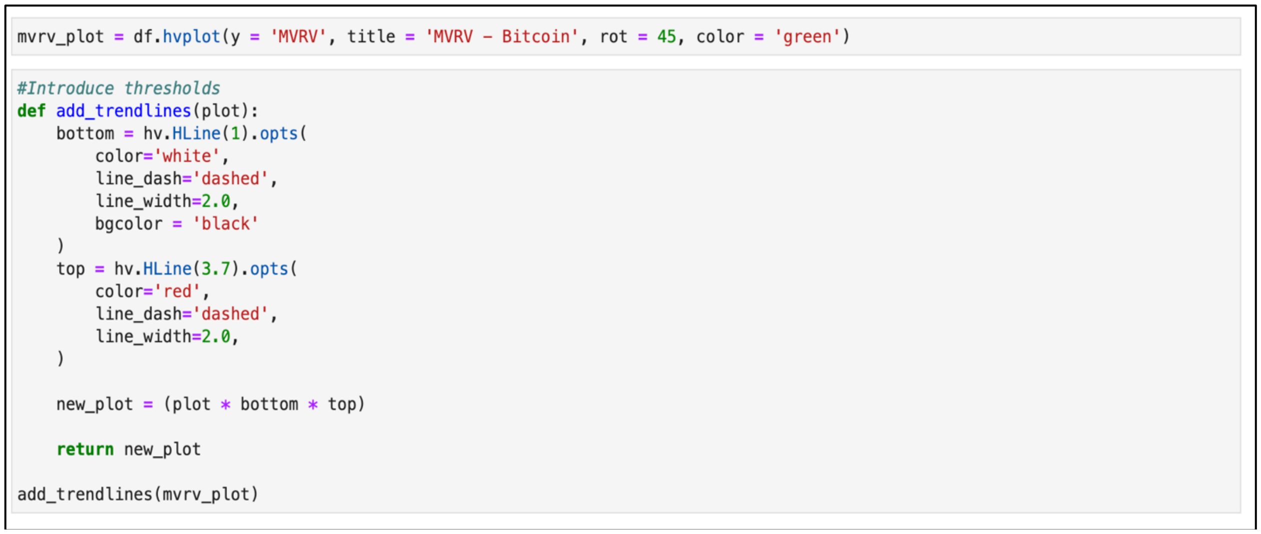
Looking at our plot, we can see that realized value appears to have a long-term upward trend: suggesting coins are being purchased and held at an increasingly higher value. Market value, however, starts as an upward trend and then becomes increasingly unpredictable after 2021. Now let's create a new column for our MVRV ratio and see if we can identify additional patterns.

Begin by creating a column for and calculating our MVRV ratio:

```
df['MVRV'] = df['Market Value'] / df['Realized Value']
df.head()
```

Date	Price	Market Value	Realized Value	MVRV
2022-06-06	31483.35522	5.698140e+11	NaN	NaN
2022-06-05	29899.87769	5.688020e+11	4.498410e+11	1.264451
2022-06-04	29847.73307	5.654950e+11	4.499250e+11	1.256865
2022-06-03	29676.16541	5.798490e+11	4.503300e+11	1.287609
2022-06-02	30436.46343	5.676370e+11	4.507940e+11	1.259194

Using hvplot and holoviews, we can plot our MVRV ratio with our valuation thresholds using the code shown below:



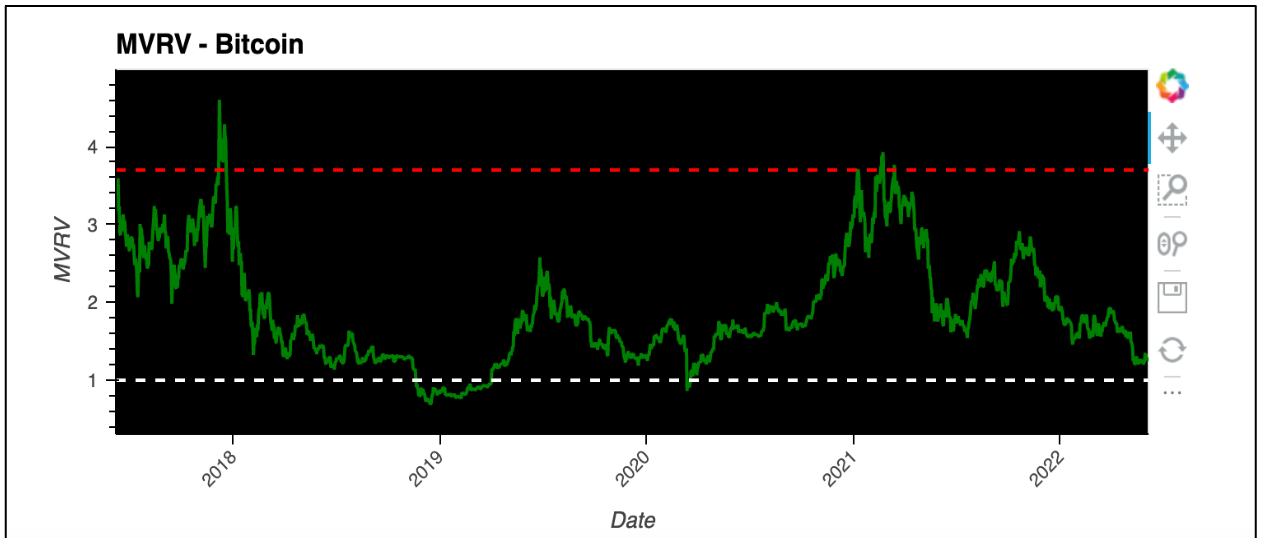
```
mvrv_plot = df.hvplot(y = 'MVRV', title = 'MVRV - Bitcoin', rot = 45, color = 'green')
```

```
#Introduce thresholds
def add_trendlines(plot):
    bottom = hv.HLine(1).opts(
        color='white',
        line_dash='dashed',
        line_width=2.0,
        bgcolor = 'black'
    )
    top = hv.HLine(3.7).opts(
        color='red',
        line_dash='dashed',
        line_width=2.0,
    )

    new_plot = (plot * bottom * top)

    return new_plot

add_trendlines(mvrv_plot)
```

DEEPER DIVE:

Notice how we use the **opts** function to help stylize our threshold lines and even set our background color? Check out https://holoviews.org/user_guide/Customizing_Plots.html to discover more options for customizing plots.

And now to view our resulting plot:



What do you see? Is there a pattern or does MVRV appear random over time? Unlike our previous plot, there does not appear to be a long-term upward and downward trend, but there does appear to be a pattern. Our job now is to investigate whether the pattern is appearing regularly (periodically) at fixed time intervals, or if the pattern appears over unpredictable, unfixed intervals. We refer to the regular and predictable changes of data in a time series as **seasonality**, and the unfixed but patterned changes as **cyclicality**.

- *Seasonality*: a characteristic of a time series in which the data experiences regular and predictable changes that recur every calendar year. For example, increased retail sales during the last quarter of the year due to the holidays.

- *Cyclicality*: much like seasonality, but with no fixed or known period. For example, a boost in sales due to a new marketing campaign.

We should also consider whether there are random fluctuations in our data, with no discernable patterns present. We refer to these random fluctuations as **noise** (sometimes referred to as white noise).

- *Noise*: random fluctuations in time series data outside of seasonal and cyclic patterns. For example, Elon Musk tweeting about Bitcoin, sometimes referred to as the "Elon effect." These tweets have caused Bitcoin to fluctuate at times over 50% within a day!

DEEPER DIVE:

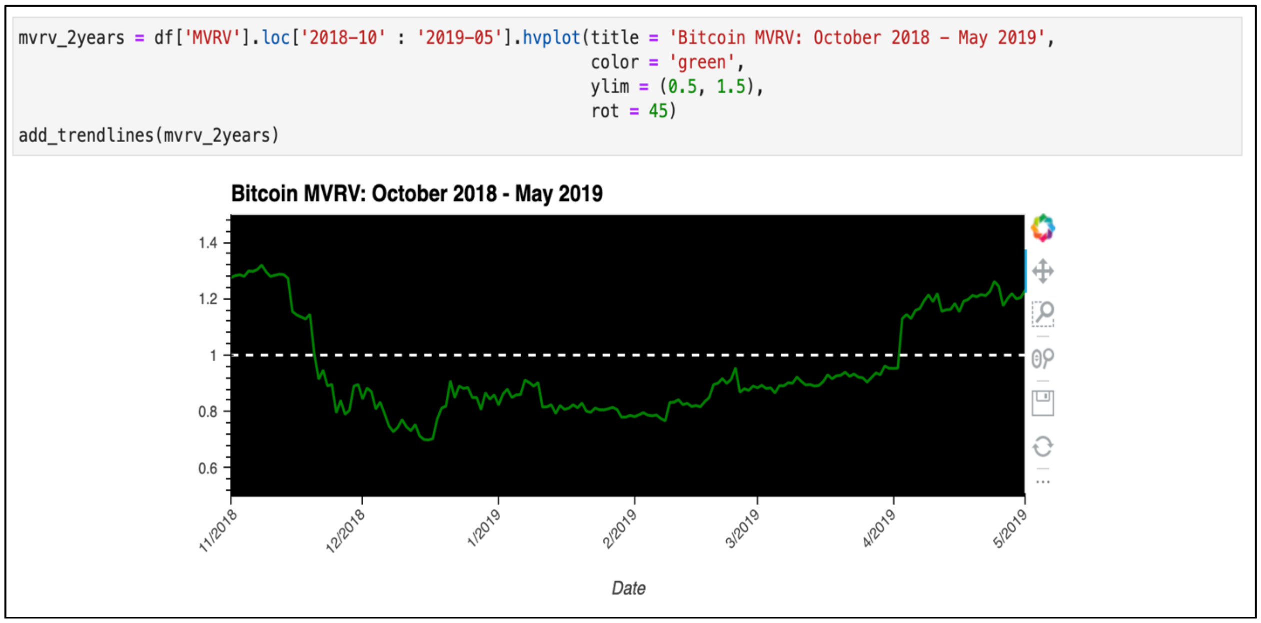
To read more about the "Elon effect" check out this article on CoinDesk: https://www.coindesk.com/layer2/culture-week/2021/12/14/the-elon-effect-how-musks-tweets-move-crypto-markets/

We have now explored our time series data using our three major components: trend, seasonality, and noise. Now we can slice our time series into different time scales to further investigate and identify patterns, and to discover quantifiable relationships in our data.

Finding the Right Time: Slicing and Grouping Time Series Data

As fintech professionals, we will rarely find an initial observation of time series data sufficient for a complete analysis, and so we are tasked with breaking down and grouping our data into different timescales. Fortunately for us, Pandas **DatetimeIndex** attributes and the **groupby** function are all we need to start our deeper dive into our dataset.

Let's focus our analysis by first slicing our time series using the Pandas **loc** function. Let's assume we want to look at a period when the Bitcoin market was undervalued. Using our MVRV plot, we can quickly find that such an opportunity existed between October 2018 and May 2019. We can slice and plot these months to begin our investigation, giving us the opportunity to investigate the period more closely. See the code below and the resulting plot:



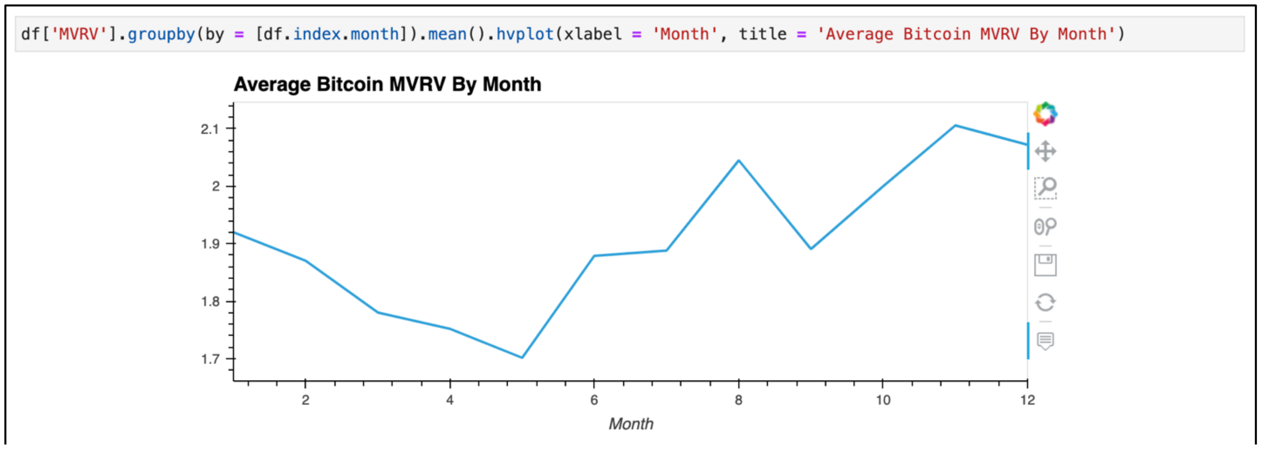
```
mvrv_2years = df['MVRV'].loc['2018-10' : '2019-05'].hvplot(title = 'Bitcoin MVRV: October 2018 - May 2019',
                                                           color = 'green',
                                                           ylim = (0.5, 1.5),
                                                           rot = 45)
add_trendlines(mvrv_2years)
```

DEEPER DIVE:

The CDC confirmed the first US Coronavirus case on January 21, 2022. By March, statewide shutdowns for nonessential workers begin, forcing many to work from home. Such events are referred to as **black swan events**—unpredictable events that are beyond what is normally expected of a situation and have potentially sever consequences. Does are data set suggest a **black swan event** in 2020? What if we look at market value, realized value, and price independently?

We can also utilize Pandas **DatetimeIndex** attributes by using the **groupby** function. Consider our original time series dataset, what if we want to analyze how the average MVRV is changing by month, by quarter, or by year so we can better understand trending and patterns within our dataset? Using Pandas, we can quickly compute and visualize these averages, giving us the freedom to explore multiple ranges and levels of our time series data.

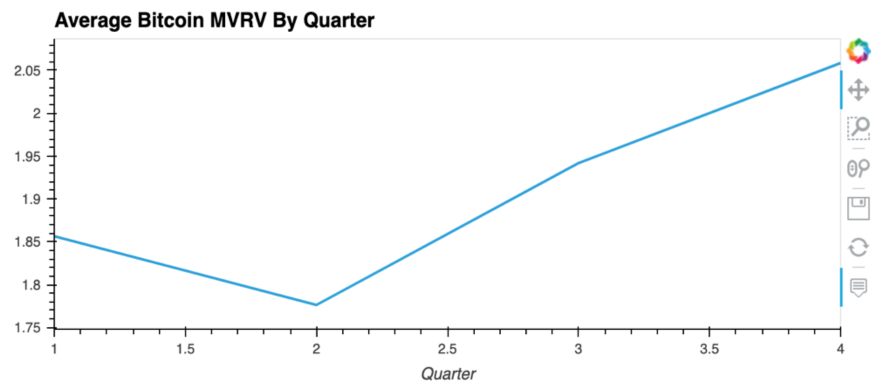
Let's start by grouping and plotting our original dataset by the average MVRV per month:



```
df['MVRV'].groupby(by = [df.index.month]).mean().hvplot(xlabel = 'Month', title = 'Average Bitcoin MVRV By Month')
```

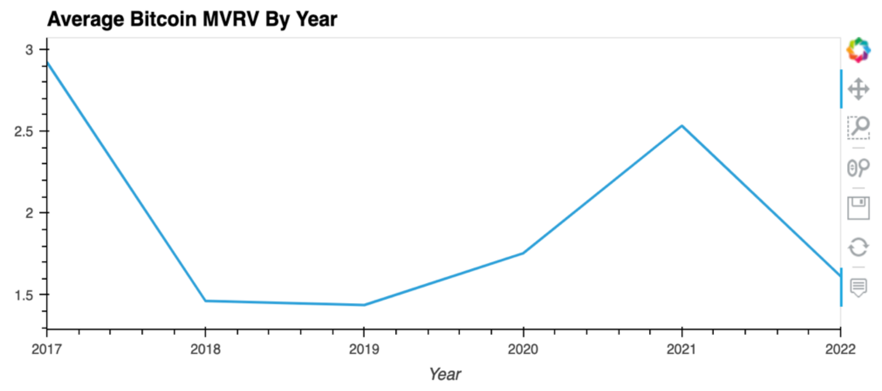
Now by quarter:

```
df['MVRV'].groupby(by = [df.index.quarter]).mean().hvplot(xlabel = 'Quarter', title = 'Average Bitcoin MVRV By Quarter')
```



And finally, by year:

```
df['MVRV'].groupby(by = [df.index.year]).mean().hvplot(xlabel = 'Year', title = 'Average Bitcoin MVRV By Year')
```

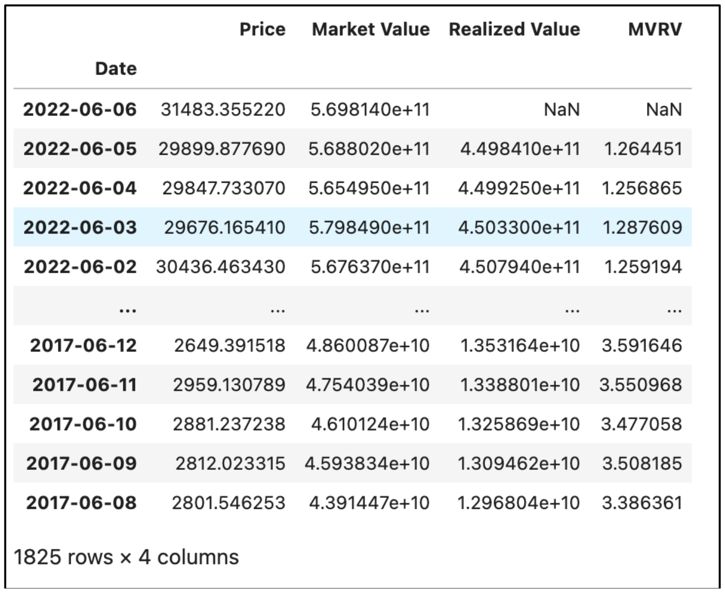


Investigating our new datasets, we can now look to identify additional patterns and to decompose our data to consider our three major time series components: trend, seasonality, and noise. What do you notice? It seems we have an upward trend (on average) across all groupings, suggesting (in the case of MVRV) that market value is outpacing realized value in the long-term, and implying an increased motive for selling in the market. We can also see that we are quickly approaching our undervalued threshold as we continue into 2022, suggesting a buy or hold strategy moving forward. Can you find any other patterns?

While grouping, slicing, and visualizing time series data gives us a method for identifying trends and patterns, we have, so far, neglected to identify and quantify correlations within our dataset. Let's now dive deeper and explore how Bitcoin prices are affected by MVRV, and how we can quantify these relationships using the Pandas **corr** function.

With the Times: Identifying Correlations

We've now created and manipulated time series data, identified major components in our time series analysis, and learned how to break down and group our time series elements to further investigate our data. But before we call-it-a-day, we need to make sure we haven't missed any patterns that may be significant to our time series investigation. Let's take a look at our current DataFrame once more and find if there is anything worth exploring further:



	Price	Market Value	Realized Value	MVRV
Date				
2022-06-06	31483.355220	5.698140e+11	NaN	NaN
2022-06-05	29899.877690	5.688020e+11	4.498410e+11	1.264451
2022-06-04	29847.733070	5.654950e+11	4.499250e+11	1.256865
2022-06-03	29676.165410	5.798490e+11	4.503300e+11	1.287609
2022-06-02	30436.463430	5.676370e+11	4.507940e+11	1.259194
...	...	...	...	...
2017-06-12	2649.391518	4.860087e+10	1.353164e+10	3.591646
2017-06-11	2959.130789	4.754039e+10	1.338801e+10	3.550968
2017-06-10	2881.237238	4.610124e+10	1.325869e+10	3.477058
2017-06-09	2812.023315	4.593834e+10	1.309462e+10	3.508185
2017-06-08	2801.546253	4.391447e+10	1.296804e+10	3.386361

1825 rows × 4 columns

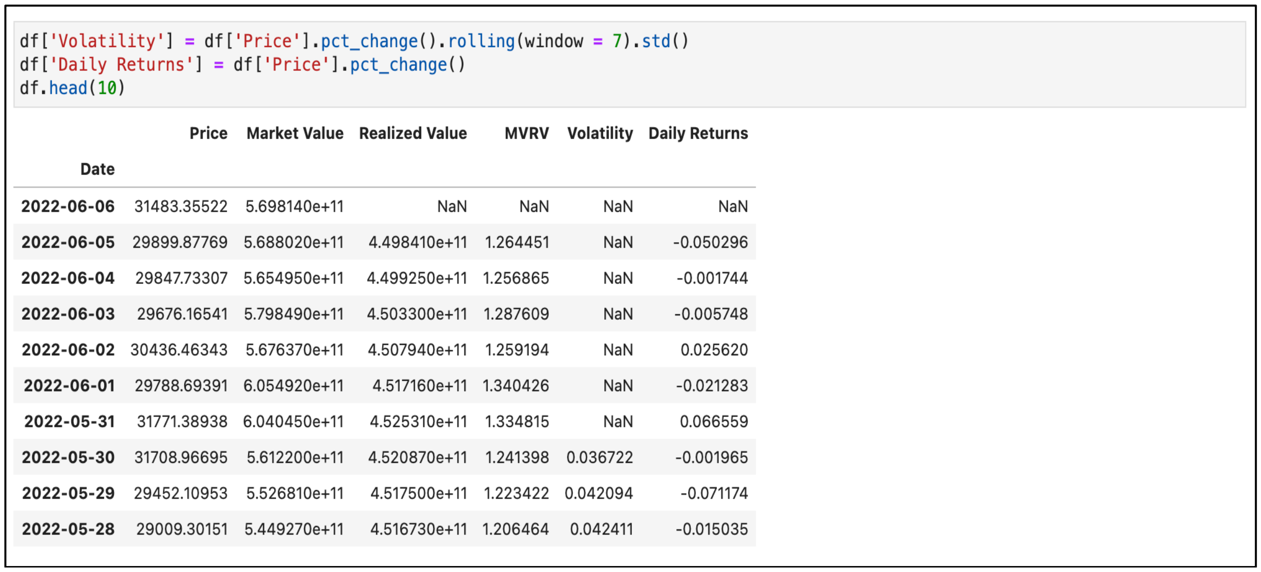
Notice anything? We have neglected to investigate if Bitcoin's price has any relation to—**correlated** to—our MVRV ratio.

- *Correlation:* a statistic that measures the degree to which two variables move in relation with each other. Represented as a **correlation coefficient**, which is value falls between -1.0 and 1.0. Positive correlations imply direct relationship between variables over time (variables rise and fall together), while negative correlations imply an inverse relationship (variables rise and fall counter to each other).

NOTE:

Hang around a statistics class long enough and you're bound to hear the phrase, "Correlation does NOT mean causation." To illustrate, consider that shark attacks and ice cream sales are positively correlated, and have been for decades. Does this mean that shark attacks are somehow caused by ice cream? Let's instead consider seasonality, and that shark attack and ice creams sales both rise over the summer! While correlations can be powerful, we should always be cautious of confusing relations and causation.

We also haven't considered how volatility and daily returns may be related to our ratio as well. Let's start by adding a few more columns to our time series data, introducing daily returns and volatility. Follow along with the code below:



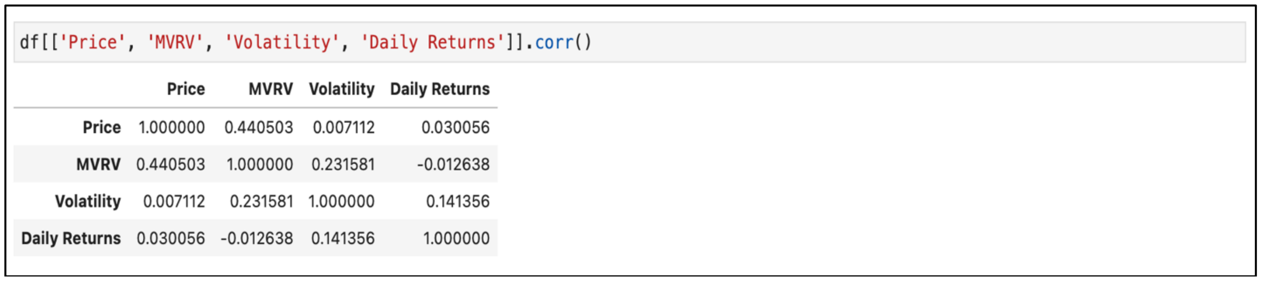
```
df['Volatility'] = df['Price'].pct_change().rolling(window = 7).std()
df['Daily Returns'] = df['Price'].pct_change()
df.head(10)
```

	Price	Market Value	Realized Value	MVRV	Volatility	Daily Returns
Date						
2022-06-06	31483.35522	5.698140e+11	NaN	NaN	NaN	NaN
2022-06-05	29899.87769	5.688020e+11	4.498410e+11	1.264451	NaN	-0.050296
2022-06-04	29847.73307	5.654950e+11	4.499250e+11	1.256865	NaN	-0.001744
2022-06-03	29676.16541	5.798490e+11	4.503300e+11	1.287609	NaN	-0.005748
2022-06-02	30436.46343	5.676370e+11	4.507940e+11	1.259194	NaN	0.025620
2022-06-01	29788.69391	6.054920e+11	4.517160e+11	1.340426	NaN	-0.021283
2022-05-31	31771.38938	6.040450e+11	4.525310e+11	1.334815	NaN	0.066559
2022-05-30	31708.96695	5.612200e+11	4.520870e+11	1.241398	0.036722	-0.001965
2022-05-29	29452.10953	5.526810e+11	4.517500e+11	1.223422	0.042094	-0.071174
2022-05-28	29009.30151	5.449270e+11	4.516730e+11	1.206464	0.042411	-0.015035

IN CASE YOU MISSED IT:

Notice that we set our rolling window equal to 7 days instead of the usual 4 we use for stock analysis. Why is this? Because cryptocurrency markets are open 24/7!

Now, let's use Pandas **corr** function and see if we can find any meaningful correlations in are data:



```
df[['Price', 'MVRV', 'Volatility', 'Daily Returns']].corr()
```

	Price	MVRV	Volatility	Daily Returns
Price	1.000000	0.440503	0.007112	0.030056
MVRV	0.440503	1.000000	0.231581	-0.012638
Volatility	0.007112	0.231581	1.000000	0.141356
Daily Returns	0.030056	-0.012638	0.141356	1.000000

Using our correlations, we can see that Bitcoin's price, volatility, and MVRV are positively correlated, meaning that there is a direct relationship between these variables. This would suggest that, based on our MVRV metric, Bitcoin will be undervalued and have increased incentive towards buying when it is less volatile, and price has gone down.

As fintech professionals, we will regularly use correlations as a basis for determining relationships between current and future values in time series data. In the next activity we will look at Netflix (NFLX) closing prices and Google Search data over time, and use our time series analysis skills to explore, break down, and find patterns in our datasets.