

CryptoWolf

Student Presentation for Cryptowatch API
Social Media Management AA 2018/2019

Students:
Gabriele Marino
Maria Ausilia Napoli Spatafora
Rosario Scalia



What is Cryptowatch?

Cryptowatch is a service that provides real-time cryptocurrency market data, charting and trading services. The real-time data on Cryptowatch is provided directly from cryptocurrency exchanges via their APIs and covers over 700 markets available on 25+ exchanges. The term market refers to a pair listed on an exchange. For example, the trading pair BTC/USD on exchange Kraken is a market.



Market Data Rest API

Cryptowatch offers a general use public market REST API, providing basic information about all markets on our platform.

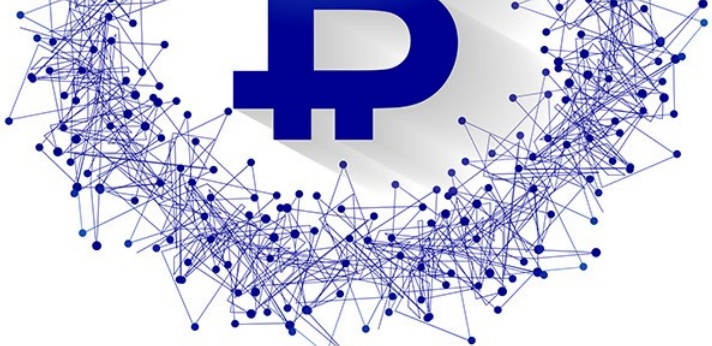
<https://cryptowat.ch/docs/api>

A large blue Bitcoin logo is partially visible on the left side of the slide. Overlaid on and around the logo is a complex network diagram consisting of numerous blue dots (nodes) connected by thin blue lines, representing a decentralized network structure.

Rate limit

The API is rate limited by a CPU allowance, rather than a fixed number of calls per time window. Some API requests take longer to fetch than others, so these cost more allowance.

Each client has an allowance of 80000000000 nanoseconds (8 seconds) of CPU time per hour. The allowance is reset every hour on the hour.



```
{  
  "result": {  
    ...  
  },  
  "allowance": {  
    "cost": 16767,  
    "remaining": 1999983233  
  },  
}
```

The cost is how many nanoseconds that request took in nanoseconds, and remaining is how many nanoseconds remain in your allowance. You can use this information, along with the current time, to have your application self-regulate its request rate.



Asset

An asset can be a crypto or fiat (currency established as money by government regulation) currency.

Asset Index:

<https://api.cryptowat.ch/assets>

Returns all assets (in no particular order)

Asset Details:

<https://api.cryptowat.ch/assets/btc>

Returns a single asset. Lists all markets which have this asset as a base or quote.



Pairs

A pair of assets. Each pair has a base and a quote. For example, btceur has base btc and quote eur.

Pairs Index:

<https://api.cryptowat.ch/pairs>

Returns all pairs (in no particular order).

Pairs Details:

<https://api.cryptowat.ch/pairs/ethbtc>

Returns a single pair. Lists all markets for this pair.



Exchanges

Exchanges allow customers to trade cryptocurrencies for other assets (such as fiat money or other cryptocurrencies).

Exchange Index:

<https://api.cryptowat.ch/exchanges>

Returns a list of all supported exchanges.

Exchange Details:

<https://api.cryptowat.ch/exchanges/kraken>

Returns a single exchange, with associated routes.



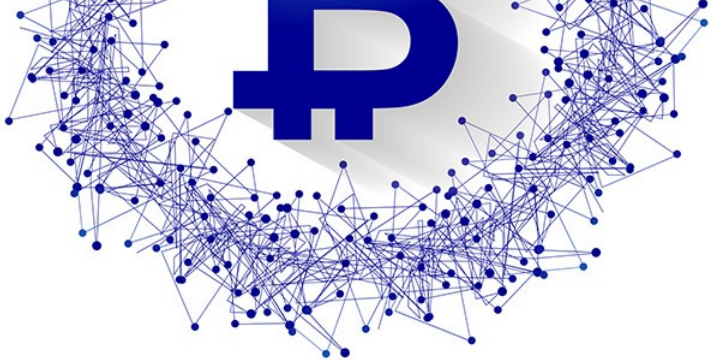
Markets

A market is a pair listed on an exchange. For example, pair btceur on exchange kraken is a market.

Market Index:

<https://api.cryptowat.ch/markets>

Returns a list of all supported markets.



Market Details:

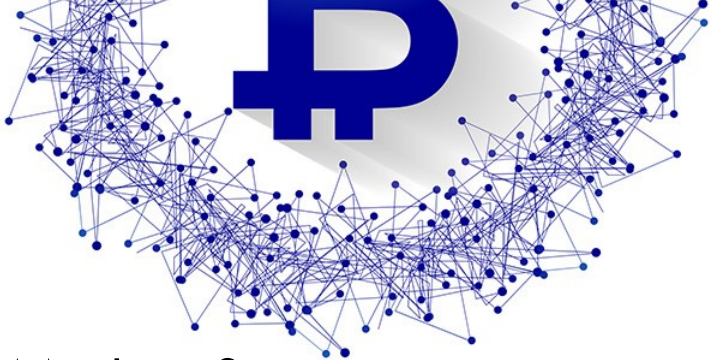
<https://api.cryptowat.ch/markets/coinbase-pro/btcusd>

Returns a single market, with associated routes.

Market Price:

<https://api.cryptowat.ch/markets/coinbase-pro/btcusd/price>

Returns a market's last price.

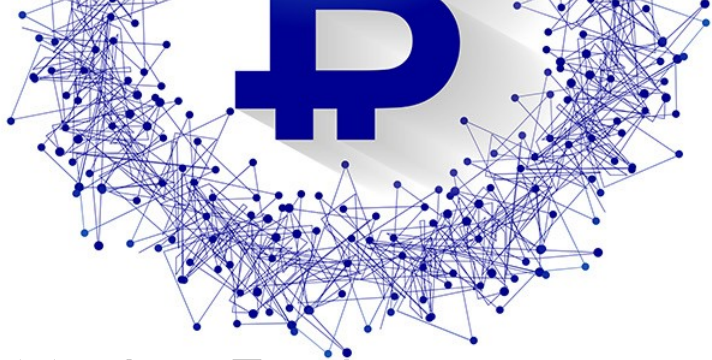


Market Summary:

<https://api.cryptowat.ch/markets/coinbase-pro/btcusd/summary>

Returns a market's last price as well as other stats based on a 24-hour sliding window:

- High price
- Low price
- % change
- Absolute change
- Volume



Parameters supported:

Param	Description	Format	Example
limit	Limit amount of trades returned. Defaults to 50.	Integer	100
since	Only return trades at or after this time.	UNIX timestamp	1481663244

Market Trades:

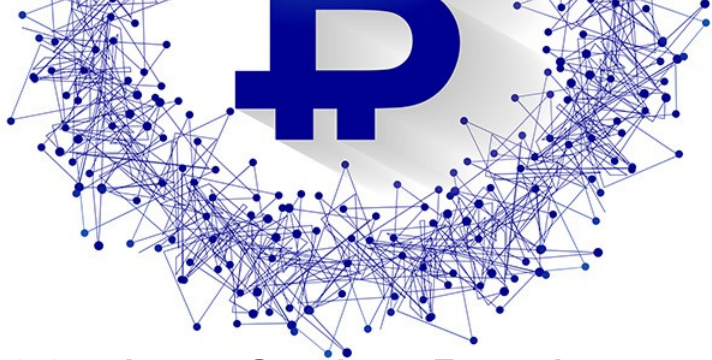
[https://
api.cryptowat.ch/
markets/coinbase-pro/
btcusd/trades](https://api.cryptowat.ch/markets/coinbase-pro/btcusd/trades)

Returns a market's most recent trades, incrementing chronologically.

Trades are lists of numbers in this order:

[ID, Timestamp, Price, Amount]

Note some exchanges don't provide IDs for public trades.



Market Order Book:

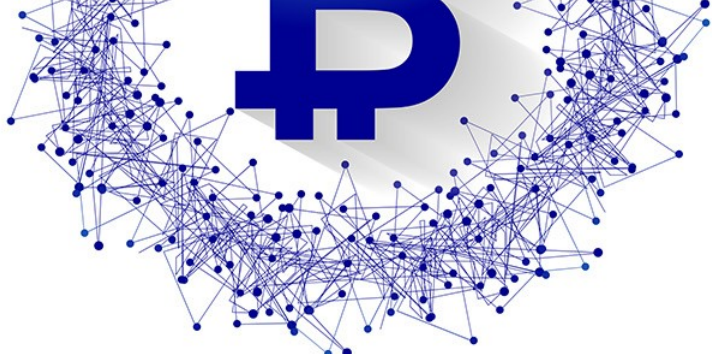
[https://
api.cryptowat.ch/
markets/coinbase-pro/
btcusd/orderbook](https://api.cryptowat.ch/markets/coinbase-pro/btcusd/orderbook)

Returns a market's order book (list of open trades).

Parameters supported:

Param	Description	Format	Example
limit	Only return n orders on each side	Integer	10
depth	Only return orders cumulating up to n size	float	20,4
span	Only return orders within n% of the midpoint	float	0,5 (meaning 0.5%, not 50%)

Orders are lists of numbers in this order:
[Price, Amount]



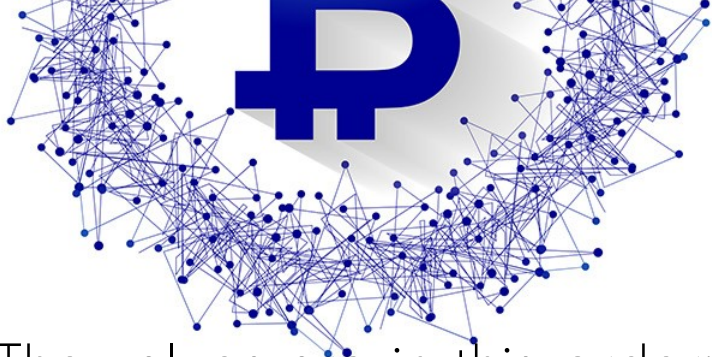
Market OHCL:

[https://api.cryptowat.ch/
markets/coinbase-pro/btcusd/
ohlc](https://api.cryptowat.ch/markets/coinbase-pro/btcusd/ohlc)

Returns a market's OHLC candlestick data (OHLC charts are a type of bar chart that shows open, high, low and closing prices). Returns data as lists of lists of numbers for each time period integer.

Parameters supported:

Param	Description	Format	Example
before	Only return candles opening before this time	UNIX timestamp	1481663244
after	Only return candles opening after this time	UNIX timestamp	1481663244
periods	Only return these time periods	Comma-separated integers	60,180,108000



All periods supported:

The values are in this order:
[CloseTime, OpenPrice, HighPrice, LowPrice, ClosePrice, Volume]

```
{
  "result": {
    "60": [
      [1481634360, 782.14, 782.14, 781.13, 781.13, 1.92525],
      [1481634420, 782.02, 782.06, 781.94, 781.98, 2.37578],
      [1481634480, 781.39, 781.94, 781.15, 781.94, 1.68882],
      ...
    ],
    "180": [...],
    "300": [...],
    ...
    "604800": [...]
  }
}
```

value	label
60	1m
180	3m
300	5m
900	15m
1800	30m
3600	1h
7200	2h
14400	4h
21600	6h
43200	12h
86400	1d
259200	3d
604800	1w

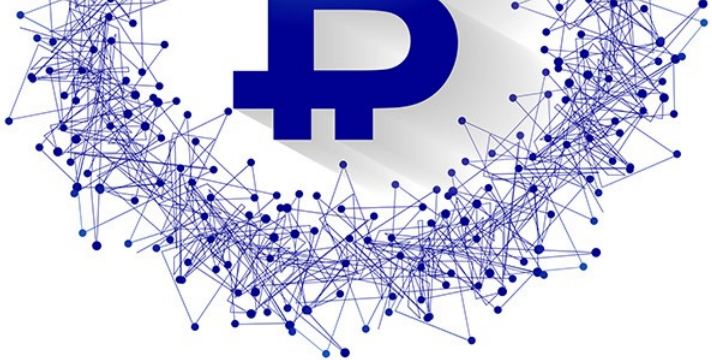


Aggregate Endpoints

You can also retrieve the prices and summaries of all markets on the site in a single request. These responses are cached and may be out of date by a few seconds.

Markets are identified by a slug, which is the exchange name and currency pair concatenated with a colon, like so:

coinbase-pro:btccusd



Prices:

<https://api.cryptowat.ch/markets/prices>

Returns the current price for all supported markets. Some values may be out of date by a few seconds.

Summaries:

<https://api.cryptowat.ch/markets/summaries>

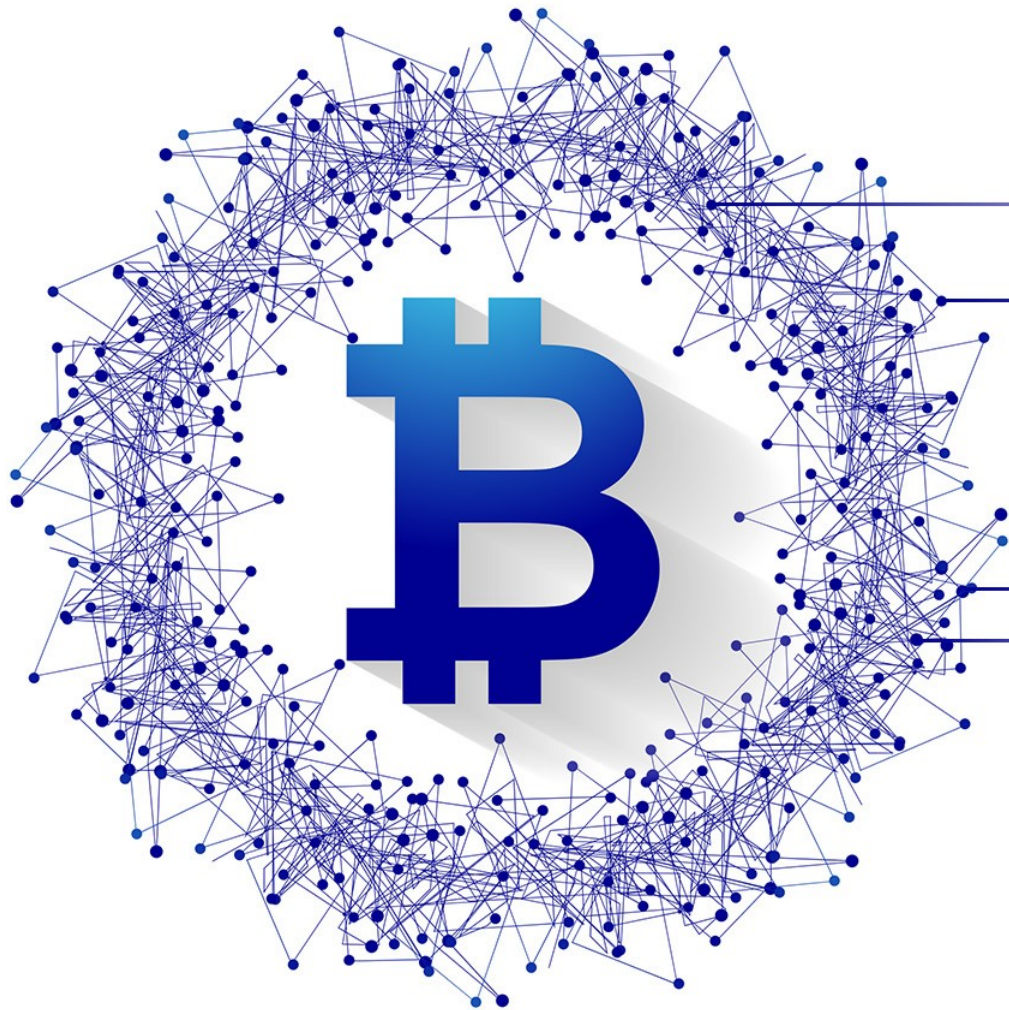
Returns the market summary for all supported markets. Some values may be out of date by a few seconds.



Why Cryptowatch?

We have chosen Cryptowatch because of its free API. Actually, we tried other socials failing:

- Booking: it requires site with 500 order/month
- Google News: it gives only 30 words with free subscription (premium subscription costs \$\$\$)
- Reddit: it never answer our request.



Google
Chart



First Step

We have tried to extract json through AJAX, but our request has been blocked by CORS and CORB problem (i.e. browser security sistem).

```
1 var jsonData = getJSON("https://api.cryptowat.ch/markets/coinbase-  
  pro/btcusd/ohlcv?before=1546297140&after=1543618800&periods=86400")  
2 function getJSON(url){  
3   var xml = new XMLHttpRequest();  
4   var json = "{}"  
5   xml.onreadystatechange = function(){  
6     xml.open("GET", url, true);  
7     xml.send();  
8     return JSON.parse(json);  
9 }
```


A large blue Bitcoin logo is partially visible on the left side of the slide. Overlaid on and around the logo is a complex network graph consisting of numerous blue dots (nodes) connected by thin blue lines (edges), creating a dense, web-like structure.

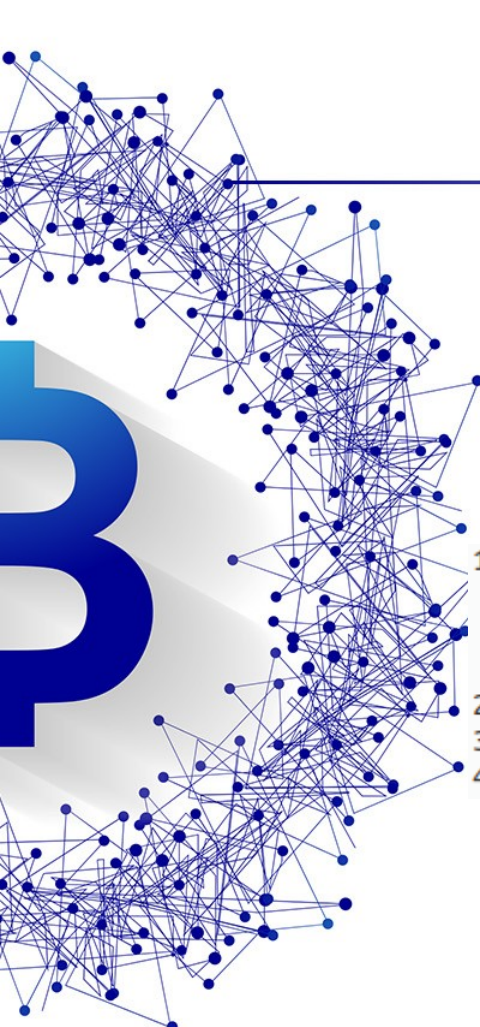
Second Step

We have used JSONP type for AJAX request, but failing in data visualization (and usage).

```
1 var jsonData;  
2 function logResults(json){  
3     console.log(json);  
4     jsonData = json;  
5 }  
6 $.ajax({  
7     url: "https://api.cryptowat.ch/markets/coinbase-pro/btcusd/ohlcv?  
before=1546297140&after=1543618800&periods=86400",  
8     dataType: "jsonp",  
9     jsonpCallback: "logResults"  
10 });
```

Third Step

<http://allorigins.ml/> is the solution to all bad guys!!



```
1 obj = $.getJSON('http://api.allorigins.ml/get?url=https%3A//api.cryptowat.ch/  
markets/coinbase-pro/btcusd/  
ohlcv%3Fbefore%3D1546297140%26after%3D1543618800%26periods%3D86400&callback=?',  
function(data){  
2     txt = JSON.parse(data.contents)  
3     console.log(txt)  
4 }
```

Fourth Step

Problems are infinity...

GChart methods accepts only two-dimensional array and it's OK after several test...

```
1 var data = google.visualization.arrayToDataTable(jsonData.result[period], true);
```

But

Last domain does not have enough data columns (missing 2) ✕

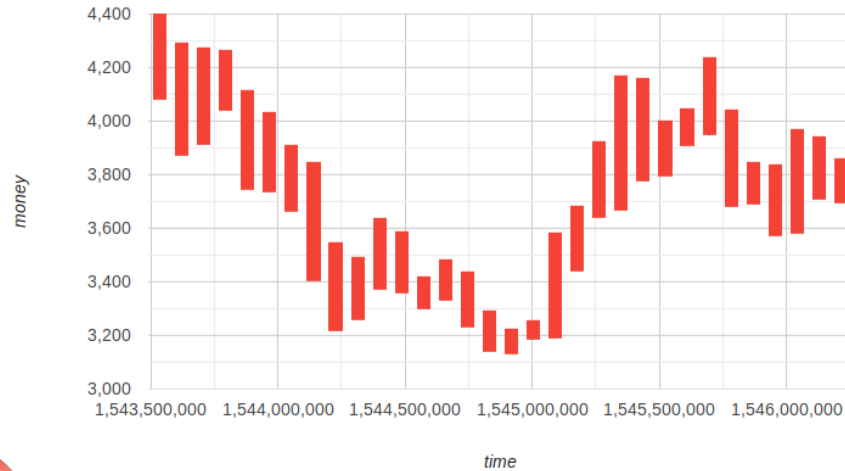
Fifth Step

We have bypassed the problem creating empty DataTable, populating it and removing the two excess columns

```
1 var data = new google.visualization.DataTable();
2 data.addColumn({type: 'number', label: 'time'});
3 data.addColumn({type: 'number', label: 'open price'});
4 data.addColumn({type: 'number', label: 'high price'});
5 data.addColumn({type: 'number', label: 'low price'});
6 data.addColumn({type: 'number', label: 'close price'});
7 data.addColumn({type: 'number', label: 'volume'});
8 data.addColumn({type: 'number', label: 'market cap'});
9 data.addRows(jsonData.result[period]);
10 data.removeColumn(6);
11 data.removeColumn(5);
```


Sixth Step

The visualization problem seemed resolved, but... NO!

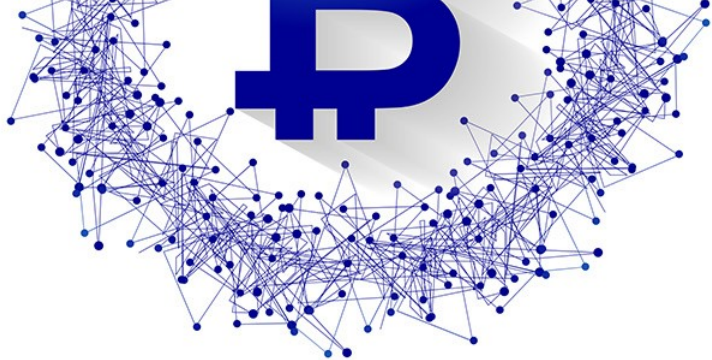




Why?

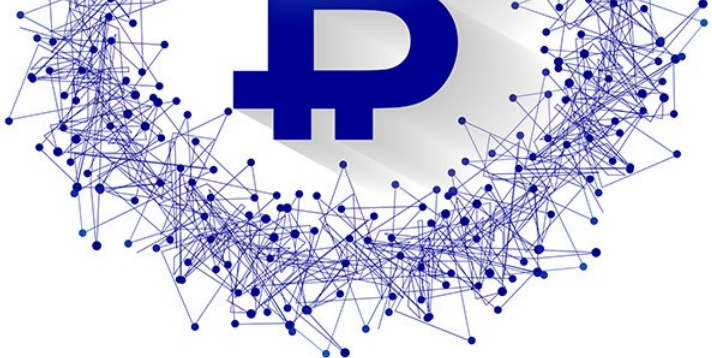
Google Chart accepts array format as [time, low price, open price, close price, high price, [tooltip]]. Our JSON's format is [time, open price, high price, low price, close price, [volume, market cap]].

We have tried to exchange the columns properties in different ways, but we have failed...



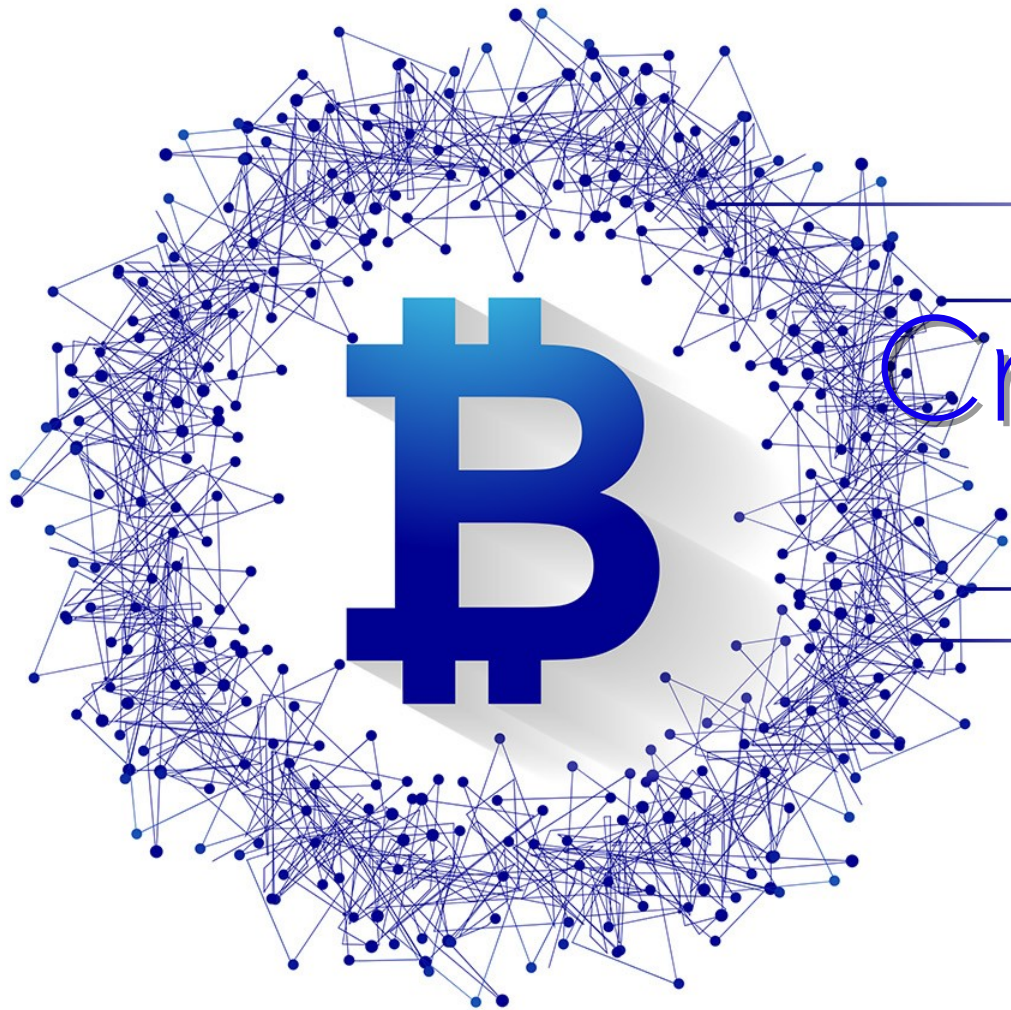
1° way

```
1 /* openprice <-> lowprice*/
2 data.setColumnProperties(5, data.getColumnProperties(1)); //save LowPrice in tmpColumn
3 data.setColumnProperties(1, data.getColumnProperties(2)); //switch OpenPrice with LowPrice
4 /*lowprice <-> closeprice*/
5 data.setColumnProperties(5, data.getColumnProperties(3)); //save ClosePrice in tmpColumn
6 data.setColumnProperties(3, data.getColumnProperties(5)); //switch ClosePrice with LowPrice
7 /*closeprice <-> highprice*/
8 data.setColumnProperties(5, data.getColumnProperties(4)); //save HighPrice in tmpColumn
9 data.setColumnProperties(4, data.getColumnProperties(5)); //switch ClosePrice with HighPrice
10 /*highprice <-> openprice*/
11 data.setColumnProperties(5, data.getColumnProperties(2)); //save OpenPrice in tmpColumn
12 data.setColumnProperties(2, data.getColumnProperties(5)); //switch OpenPrice with HighPrice
13
```



2° way

```
1 var dataTmp = new google.visualization.arrayToDataTable([[ 'Mon', 20, 28, 38, 45 ]], true);  
2 //tmp dataTable to do change column properties  
3 data.setColumnProperties(1, dataTmp.getColumnProperties(2)); //set openprice  
4 data.setColumnProperties(2, dataTmp.getColumnProperties(4)); //set highprice  
5 data.setColumnProperties(3, dataTmp.getColumnProperties(1)); //set lowprice  
6 data.setColumnProperties(4, dataTmp.getColumnProperties(3)); //set closeprice
```

CryptoWatch
Embed



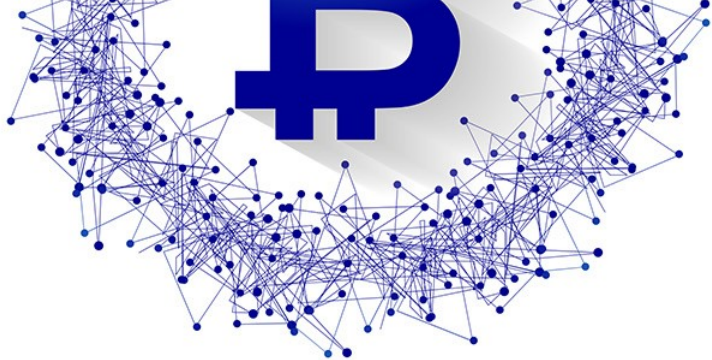
What is?

CryptoWatch Embed is small library for embedding Cryptowatch charts on a website in real time:

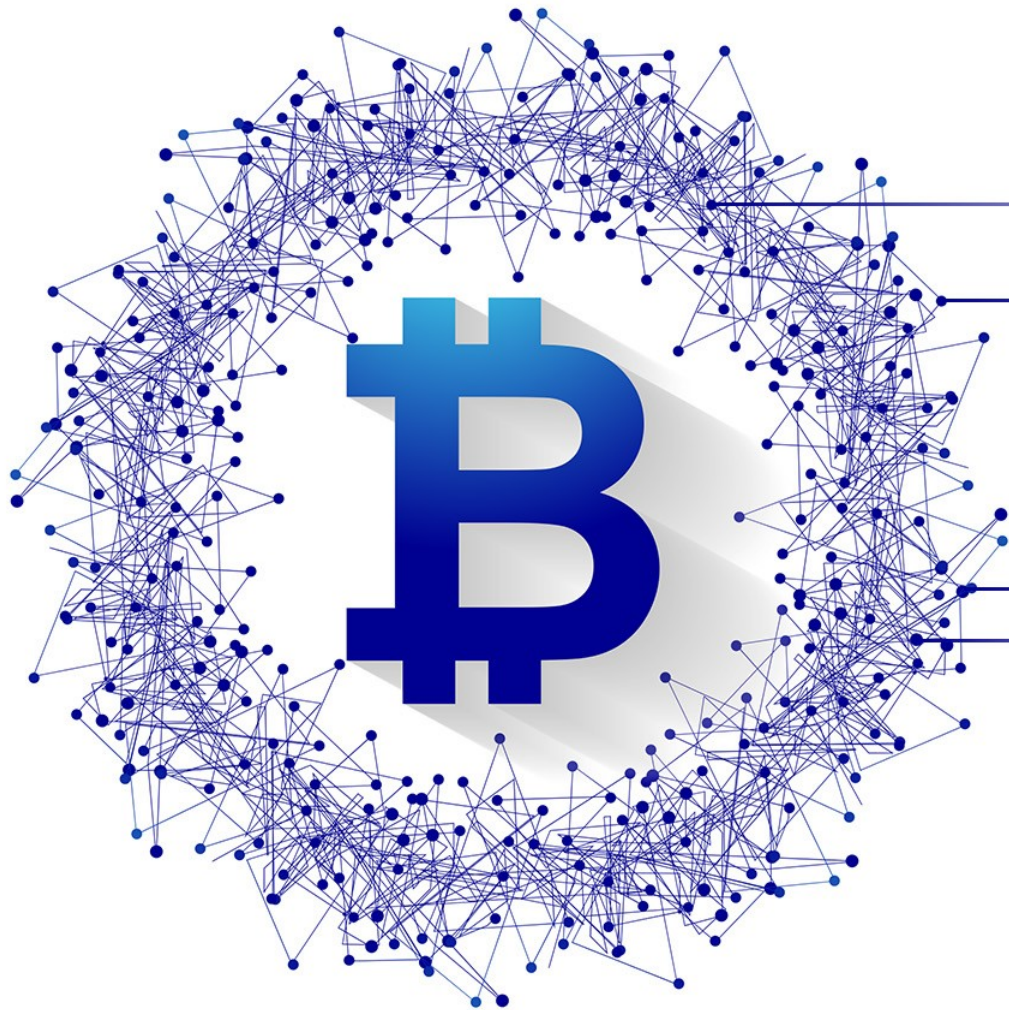
<https://github.com/cryptowatch/embed>

Few options can be provided to configure the chart:

- Exchange and currency pair;
- Width and height;
- Time Period (it can be set in real time)
- Color Scheme.



```
1 <!DOCTYPE html>
2 <html lang="en_UK">
3   <head>
4     <meta charset="utf-8">
5     <div id="chart-container" style="width: 900px; height: 500px;"></div>
6     <script type="text/javascript" src="https://static.cryptowat.ch/assets/scripts/embed.bundle.js"></script>
7     <script>
8       var chart = new cryptowatch.Embed('kraken', 'btcusd', {
9         timePeriod: '1d',
10        width: 650,
11        presetColorScheme: 'ballmer'
12      });
13      chart.mount('#chart-container');
14    </script>
15
16  </head>
17  <body>
18  </body>
19 </html>
```

Index.py
class



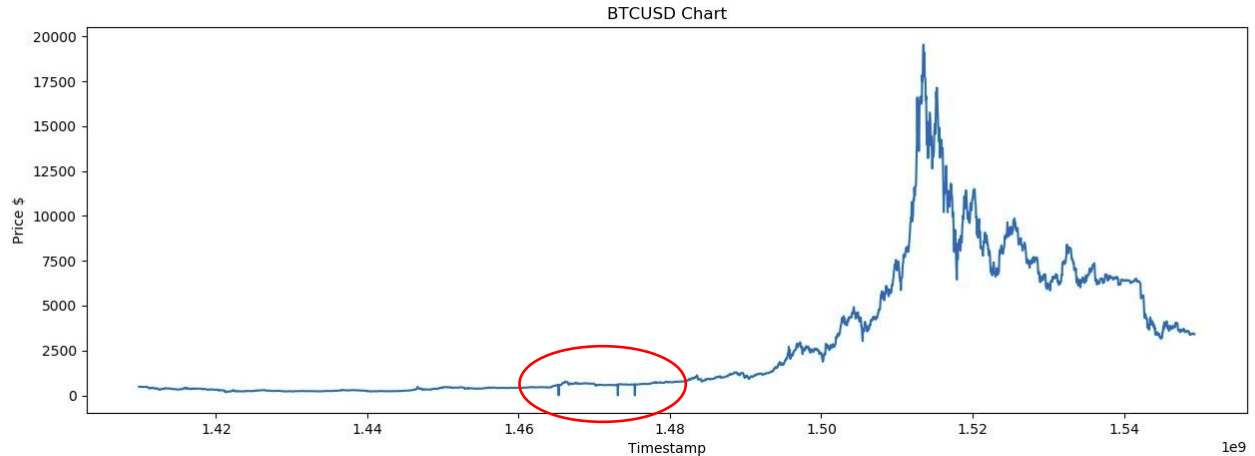
What is?

We have used several instruments to describe distribution shape, central tendencies and dispersion indices computed in **Index** class.

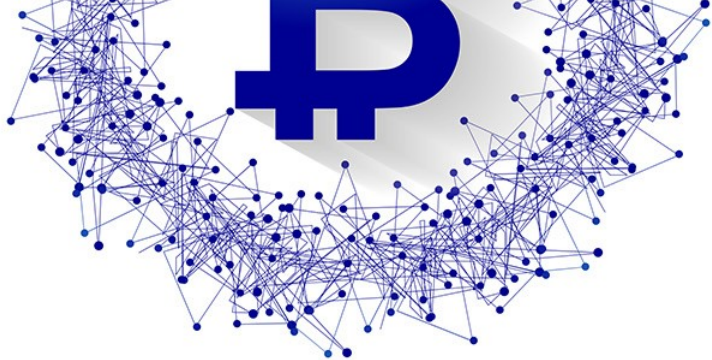
After extracting data, this is converted in numpy array to perform computation; data are 12-hour candle from 01-01-2011, 00:00 to 04-02-2019, 01:00 previously saved in a .pkl file by **Dataset_Creator**. The data are Open Price's vector and Close Price's vector.

Data visualization

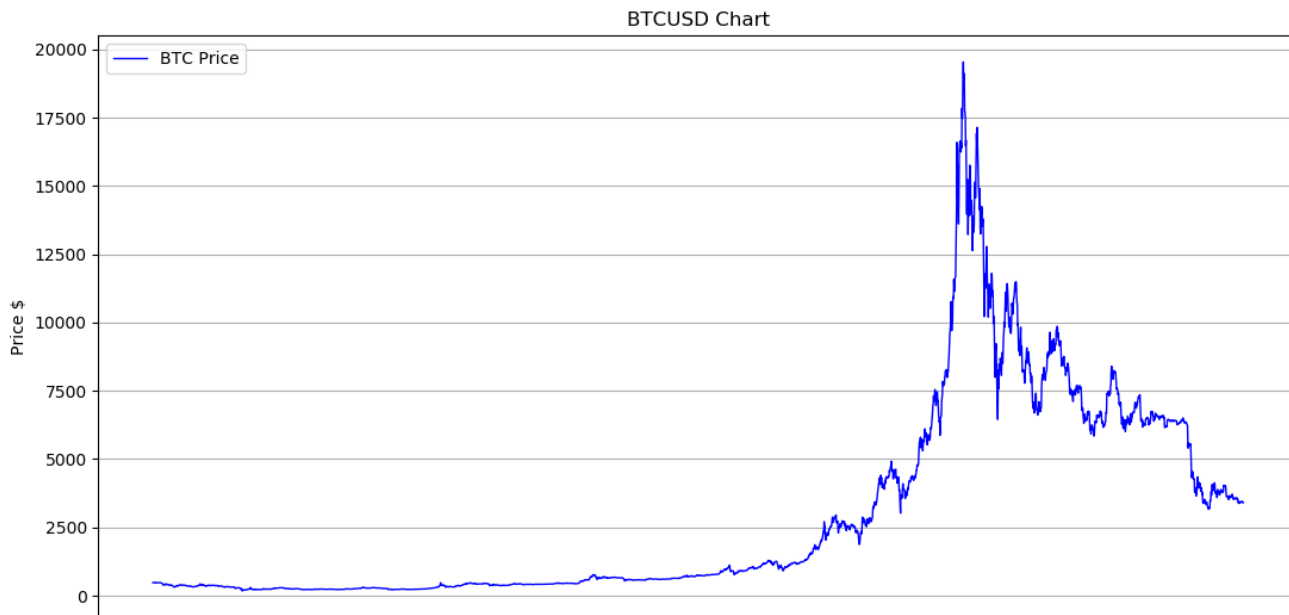
Firstly, we have plotted close time VS close price to check if our data are correct:

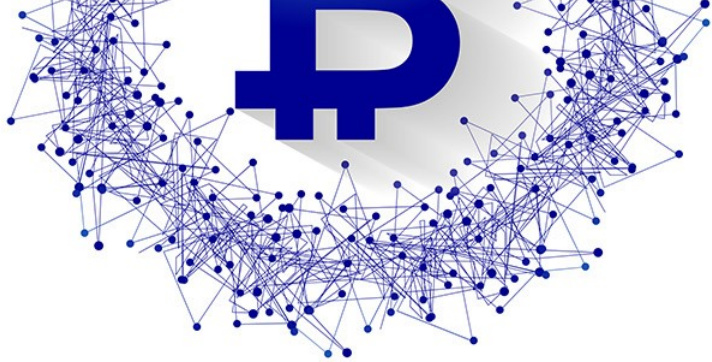


But we have realized that some candles are dirty because of API Cryptowatch: they have close price setted to 0.0.



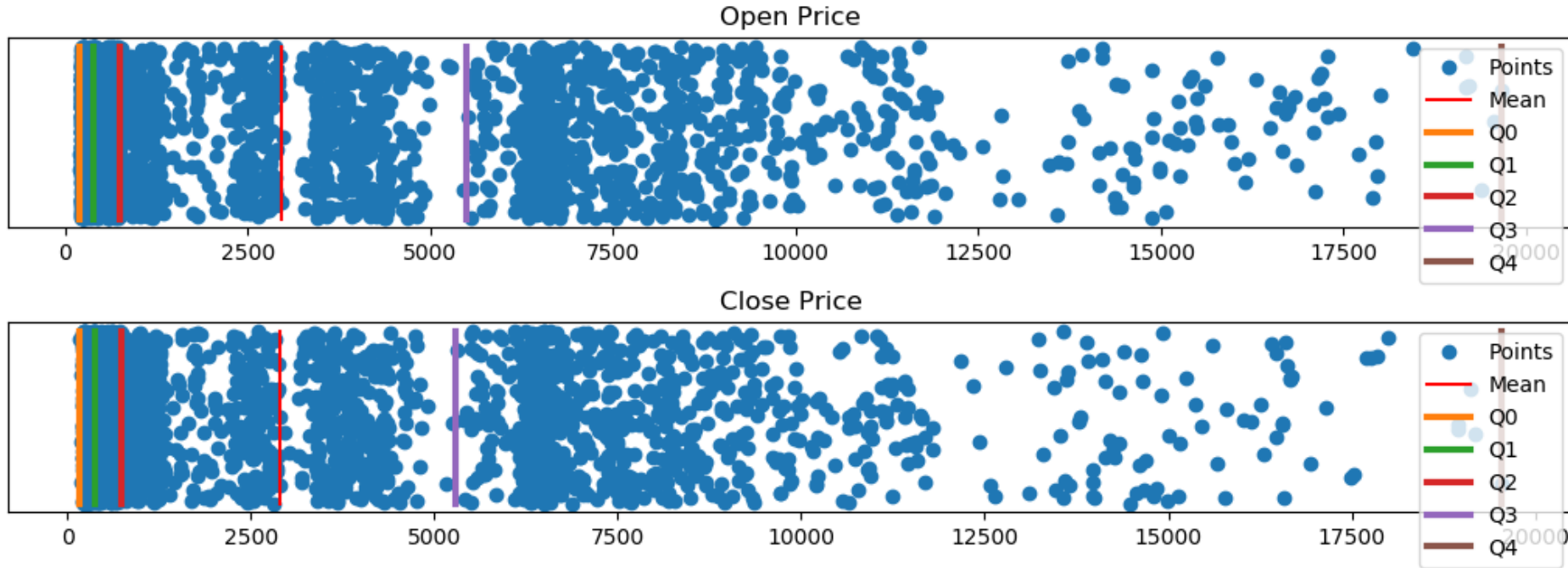
So, we have created the new dataset removing this dirty candles and we have plotted again:

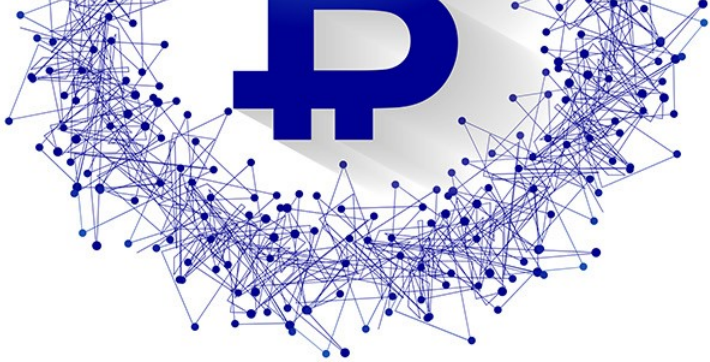




It is used dotplot to show data marked mean and quartiles (0%, 25%, 50%, 75%, 100%).

Mean and median are in according to dotplot's density.

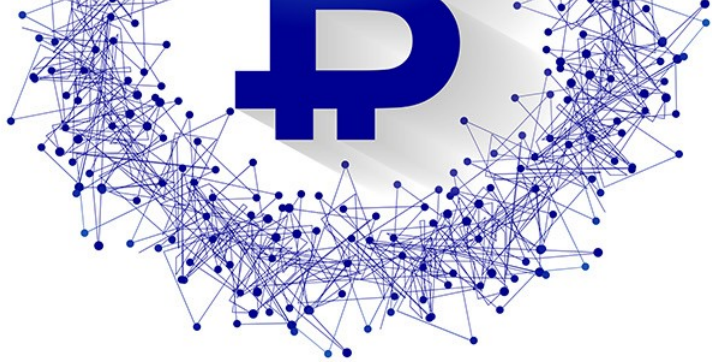




The figure shows all descriptive indices calculated:

- Minimum, maximum and range
- Mean and Median
- Quartiles (0%, 25%, 50%, 75%, 100%)
- Variance
- Standard Deviation

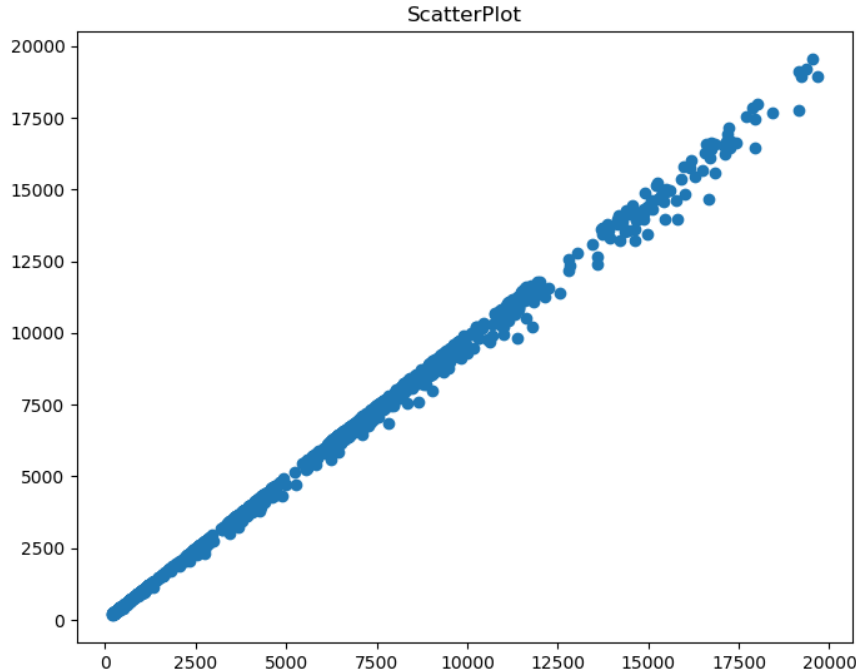
```
Open Price Minimum: 201.96
Close Price Minimum: 172.0
Open Price Maximum: 19666.0
Close Price Maximum: 19546.88
Open Price Range: 19464.04
Close Price Range: 19374.88
Open Price Mean: 2967.0544313238042
Close Price Mean: 2901.2639745183346
Open Price Median: 749.98
Close Price Median: 742.0
Q0: 0% of Open Price: 201.96
Q1: 25% of Open Price: 381.4475
Q2: 50% of Open Price: 749.98
Q3: 75% of Open Price: 5499.4349999999995
Q4: 100% of Open Price: 19666.0
Q0: 0% of Close Price: 172.0
Q1: 25% of Close Price: 375.655000000000003
Q2: 50% of Close Price: 742.0
Q3: 75% of Close Price: 5290.8099999999995
Q4: 100% of Close Price: 19546.88
Open Price Variance: 14007498.044542884
Close Price Variance: 13240491.67805863
Open Price STD: 3742.659221000876
Close Price STD: 3638.74864178039
```



To measure correlation between data series is done the Scatter plot: data are aligned along a straight line.

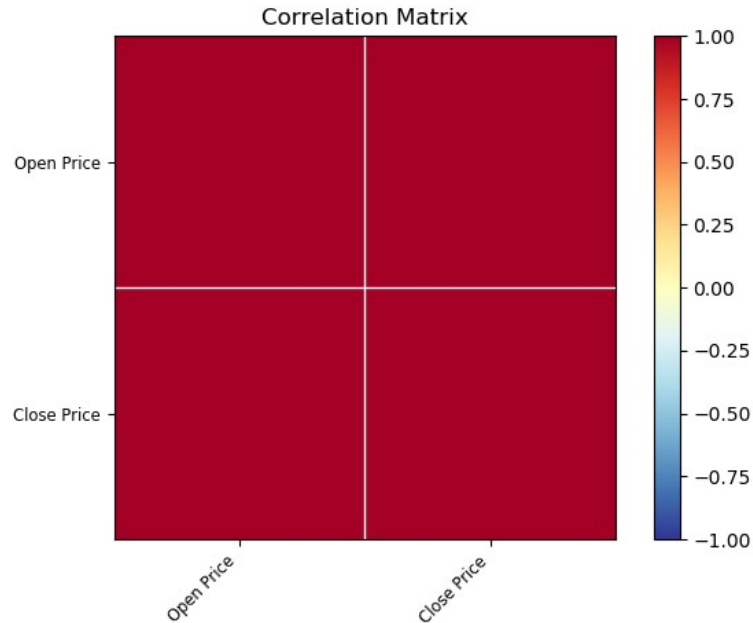
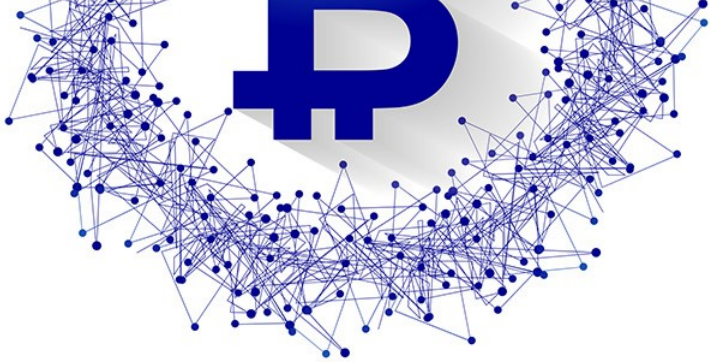
The covariance is computed through the covariance matrix which reports:

- $\text{Cov}(X,X)$ and $\text{Cov}(X,Y)$ on first row;
- $\text{Cov}(Y,X)$ and $\text{Cov}(Y,Y)$ on second row.



The main diagonal contains the variance of two data series; the other diagonal contains the covariance: this value is positive and so the characters grow each other.

```
Covariance: [[14011851.47646659 13615441.7292866 ]  
             [13615441.7292866  13244606.88651654]]
```

Other correlation coefficients are:

- Pearson: its values is almost 1 and so implies an exact linear relationship (0.9994577431432222);
- Spearman rank-order: its values is almost 1 and so implies an exact monotonic relationship (0.9996041721721992);
- Kendall: its values is almost 1 and so implies strong agreement (0.9874331459000218);
- Matrix correlation: it can be observed strong correlation between data series, again.



Our contacts



Gabriele Marino

marino.gabri97@gmail.com



Maria Ausilia Napoli Spatafora

ausilianapoli@gmail.com



Rosario Scalia

rosarioscalia@outlook.com