

UNIVERSITY OF WARSAW

WEBSCRAPPING AND SOCIAL MEDIA SCRAPPING

Quantifying the effect of cryptocurrencies market. Building analytical dashboard for pre-prurchase monitoring is allowing for more thought out decisions.

Author

Mateusz Baryła 427778

Supervisor

Anna Lewczuk
Przemysław Kurek

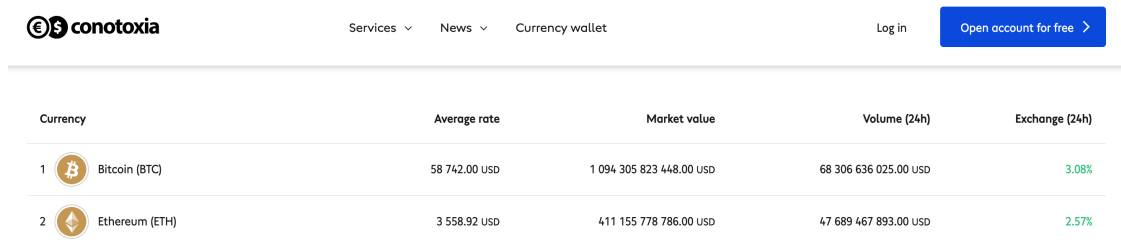
May 8, 2021



1 Project Goals

The increasing role of cryptocurrencies is the main concern of the project. We carefully study the terminology and based on them we choose the website that will allow for robust scrapping. Based on the underlying data we build an analytical dashboard that can be used by non-technical audience.

Taking into account the business perspective there is a need to investigate the current perspectives of cryptocurrencies industry. The main page used for scrapping is <https://conotoxia.com/cryptocurrencies/cryptocurrency-rates>. We have checked in details whether there are any limitations in regards to scrapping and have not found any. We insert a sample screenshot below, for a reader:



The screenshot shows the 'conotoxia' website header with navigation links: Services, News, and Currency wallet. There are 'Log in' and 'Open account for free' buttons. Below the header is a table with the following data:



Currency	Average rate	Market value	Volume (24h)	Exchange (24h)
1  Bitcoin (BTC)	58 742.00 USD	1 094 305 823 448.00 USD	68 306 636 025.00 USD	3.08%
2  Ethereum (ETH)	3 558.92 USD	411 155 778 786.00 USD	47 689 467 893.00 USD	2.57%

Figure 1: Screenshot of the target scrapping page.

Currency	Average rate	Market value	Volume(24h)	Exchange
BTC	58 742.00 USD	1 094 305 823 448.00 USD	68 306 636 025.00 USD	3.08 %
ETH	3 558.92 USD	411 155 778 786.00 USD	47 689 467 983.00 USD	2.57 %
...

Table 1: Desired outcome table.

2 Scrapper mechanics and technical description

Each of three scrappers use very similar mechanism that allows for gathering data in the robust way. We can specify the number of pages that limit scrappers.

Selenium uses chrome executable in our case with `-headless` argument that allows for loading the whole table instead of only a part of it. Later the table rows are extracted and appended to the list of values.

Beautiful Soup content is loaded with Request from urllib library with User-Agent header and Mozilla/5.0 value specified. The rest of the process is similar to the selenium.

Scrapy parses url by url and table in it and at the end yield link which is additional class that consist of the following attributes (currency, average_rate, market_value, volume, exchange).

In terms of processing output we need to perform character escaping for non-unicode characters. There are new line characters that need to be removed as well.

3 Data analysis

The important part for each data science project is of course exploratory data analysis. Keeping in mind the data analysis part that is one page limit we will address the issue of limited resource provided.

Before loading data into Tableau there is a need to process the table. We split specific variables or remove currencies. The python notebook can be found in the repository as well.

For better readability we filter data so that 9 currencies are left at the end. We present several plots such as barchart: normal and stacked versions, mosaic plot and run chart by exchange rate.

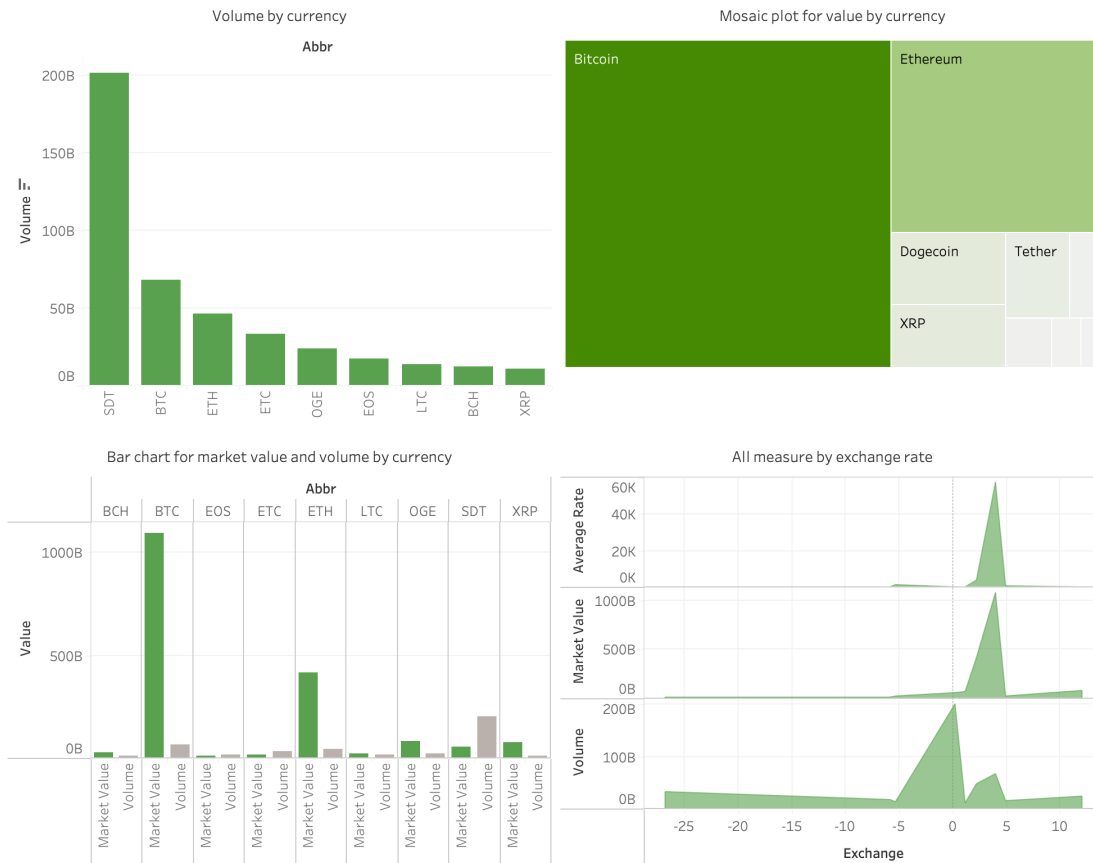


Figure 2: Pre-purchase dashboard of the cryptocurrencies filtered

The so-whats of this analysis are the following:

- Based on market value to volume comparison we see one big outlier - BITCOIN and ETHEREUM. What is interesting is that the volume for BITCOIN although is clearly visible is not the highest one. One's may think about total market share of the particular cryptocurrency which might be sort of linear combination of market value and volume.
- In the mosaic plot we can see that bitcoin has more than 50% of market share when taking into account filtered currencies.

4 Comparing performance of scrapers

a Simulations

We perform time benchmarking in the following settings:

- scrapper: selenium, scrapy, soup
- number of pages
- number of iterations

We keep whole mechanism in the *compare.py* file. Three functions have been create based on the scrapping files. The idea behind this data gathering can lead us to measuring not only results from macro point of view but we can proceed to micro analysis as well. The stabilization performance is an important concern that should not be neglected and this is the reason for performing each simulation several times. Therefore we can not only examine single iterations effects but also the overall stability. That will result in the following table:

scrapper	n_pages	n_iter	time
sel	1	1	$time_{s,1,1}$
sel	2	1	$time_{s,2,1}$
...
sel	10	1	$time_{s,10,1}$
sel	1	2	$time_{s,1,2}$
...
sel	10	10	$time_{s,10,10}$
scrapy	1	1	$time_{s,1,1}$
scrapy	2	1	$time_{s,2,1}$
...
scrapy	10	1	$time_{s,10,1}$
scrapy	1	2	$time_{s,1,2}$
...
scrapy	10	10	$time_{s,10,10}$
soup	1	1	$time_{s,1,1}$
soup	2	1	$time_{s,2,1}$
...
soup	10	1	$time_{s,10,1}$
soup	1	2	$time_{s,1,2}$
...
soup	10	10	$time_{s,10,10}$

Table 2: Benchmark data output

Summary:

- The fastest scrapper is scrapy which beats selenium and soup a lot!
- Taking into account number of pages and time selenium follows a linear trend, the more pages the more time spent on scrapping.

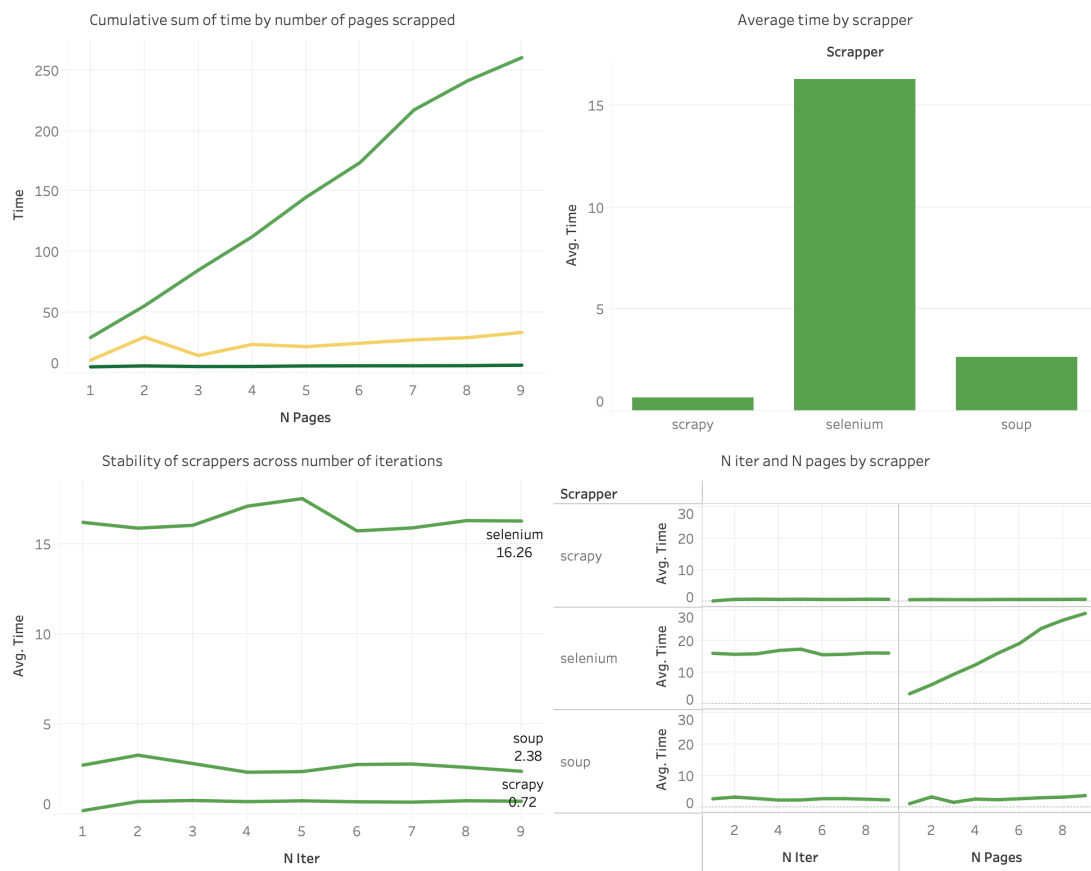


Figure 3: Three scrappers benchmarked in a single dashboard.