Problem statement:

Analyzing the Ethereum blockchain.

Analysis: Price volatility, Centrality of wealth, smart contracts, Tokens, mining centrality

Concepts: Ethereum, Tokens, Smart contracts, ERC20

Example of a smart contract

Motivation: The ICO hype

Data: Blockchain is stored in leveldb binary format. No JSON or any text format available

Collecting data: Challenging

Crawling: Finding the pull request. The original github repo we found was taking more than 50 days. Reduced 90%

The windows challenge: Geth worked on windows. Pyetherem doesn’t

Getting 50Gb of data on a linux box would take ages!

Setting up a cloud VM was not the best timely option as well. Also setting up my windows box to have an external IP would be a pain.

Solution: setting up a Linux virtualbox to run the code. And leaving mongo and geth on windows.

Fixing the code:

The repo we found was the ONLY github repo that does Ether crawling. The code was not documented, and the author does not maintain it any longer.

It is designed to run on Linux so all directory structure and environment variables were on Linux format.

It was also designed to run on Python 2 so we had to replace some python modules and attempt to restructure the environment variables to run on windows.

This went well until we cloud not build pyethereum on windows.

When we decided to go with the virtualbox solution. We faced another hurdle: The code was slow as it sends an http request to the geth JSON-RPC server to get one block at a time. The pull request we found used a more robust approach by using session instead of http request.

We built over that pull request and we found out a few bugs that we had to fix first. The code only stored the transactions inside each block without any block details. So that was the first thing we fixed.

Setting up Geth

JSON-RPC

Mongodb

Crunching the blockchain

Leveldb

Model Scarcity: we haven’t found any public analysis

Setting up the environment

Data used: The blockchain has more than 5 million blocks with a new block added every 15 seconds. Each block permanently stores a list of confirmed transactions and it is linked to the next block. We have started our analysis from block 2,900,000 that was mined on the 29th of December 2016. The more recent in time the higher the no. of transactions stored in each block which makes the average crawling time to increase. We started by 22 iterations/second and as we moved along it went down to 6 iterations/second

External code

Libraries used: pyethereum, pymongo, tqdm, …

Visualization: Piecharts, timeseries, histogram

References:

**Correlation and Autocorrelation**

Statistical correlation summarizes the strength of the relationship between two variables.

We can assume the distribution of each variable fits a Gaussian (bell curve) distribution. If this is the case, we can use the Pearson’s correlation coefficient to summarize the correlation between the variables.

The Pearson’s correlation coefficient is a number between -1 and 1 that describes a negative or positive correlation respectively. A value of zero indicates no correlation.

We can calculate the correlation for time series observations with observations with previous time steps, called lags. Because the correlation of the time series observations is calculated with values of the same series at previous times, this is called a serial correlation, or an autocorrelation.

A plot of the autocorrelation of a time series by lag is called the AutoCorrelation Function, or the acronym ACF. This plot is sometimes called a correlogram or an autocorrelation plot.