Gitcoin Exploratory Data Analysis Bounty

1. Data collecting and preprocessing
2. Summary dashboards for Gitcoin Rounds
3. Hash popularity distribution algorithm to find potential Sybil attacks wallets:
    1. Description of the algorithm
    2. Example of working on the Gitcoin round 15 dataset
    3. The results of applying the algorithm on the voting datasets
4. Algorithm results verification on the Ethereum transactions dataset:
    1. Generating wallets pairs and features
    2. Output file example and statistics of the Sybil attack wallets received
5. Progress and future work

1. Data collecting and preprocessing steps

Voting datasets:

1. Download datasets for the past Gitcoin Rounds from the Ocean Market: https://market.oceanprotocol.com/profile/0x6fd78613E08FCB92890e65eA14450750aCAFF7b5
2. Drop invalid wallets with length != 42 from the Gitcoin 15 round voting dataset.
3. In this round we also have a small number of rows not in the 'eth_zksync', 'eth_std', 'eth_polygon networks. Let's drop them.
4. Convert date time feature to common format – timestamp.
5. Convert amounts to USD (using rates from the last day of the round for simplicity).

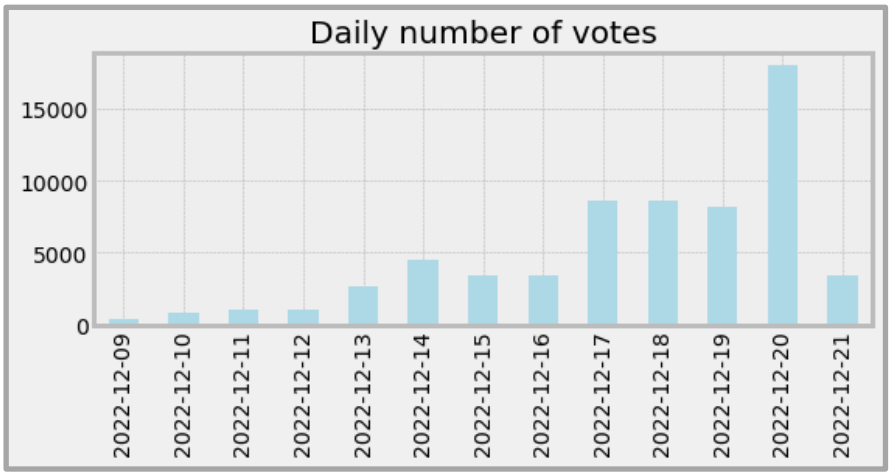
Ethereum transactions dataset:

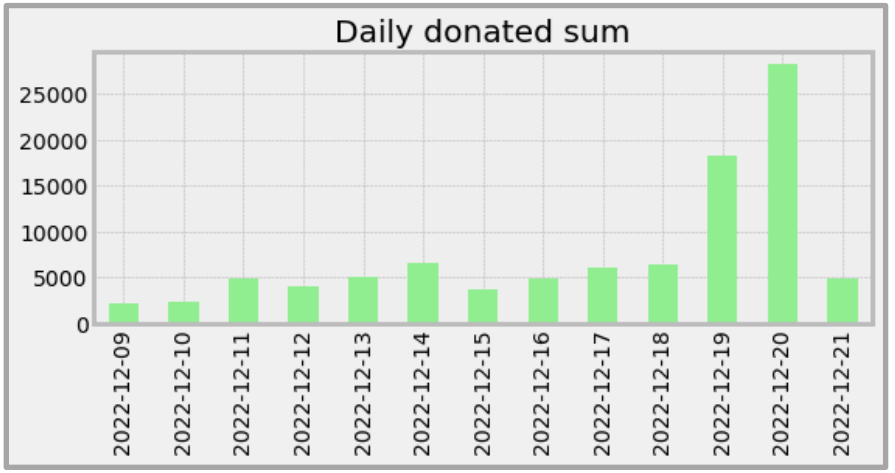
1. Collect all unique wallets from the previous dataset.
2. Use Etherscan API to get transactions by that wallets.
3. Preprocess that transactions to format that "wallet" feature is always the wallet from voting dataset and "wallet_add" is another. Direction = 1 if transaction is from "wallet" to "wallet_add", -1 is the opposite way.
4. Convert transaction value to float format.
5. Add flags if the "wallet" and "wallet_add" are in the rounds datasets.
6. Feature engineering process we will describe later.

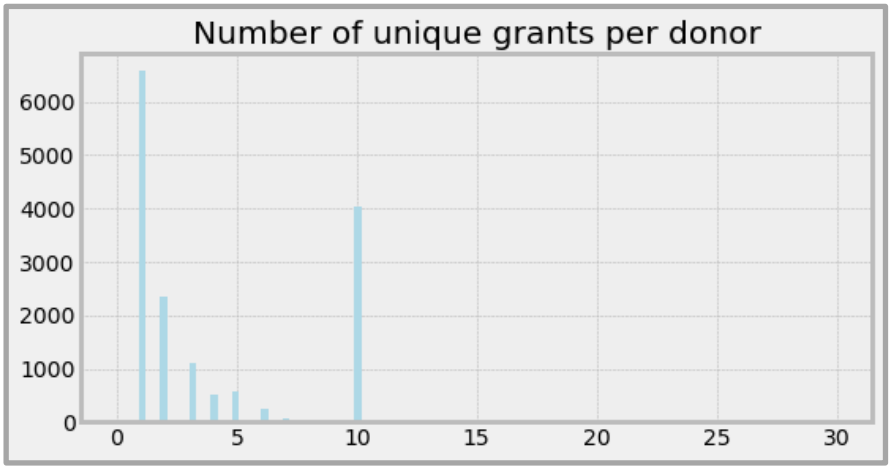
*More details you can find in the following scripts: *01.data_preprocessing.ipynb, 11.parse_etherscan.ipynb, 12.get_transactions.ipynb*

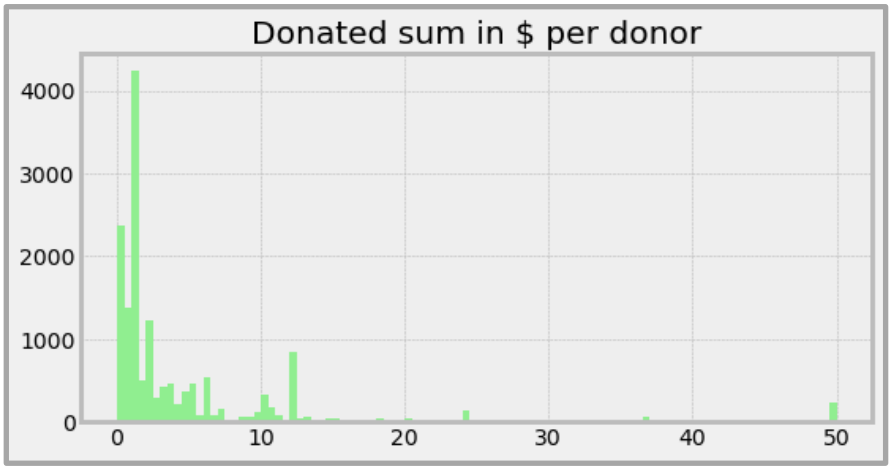
2.1. Summary dashboard for Gitcoin round 15

Number of donors:	Number of projects:	Number of votes:	Donated sum:	Average number of votes per donor:	Average donated sum per donor:
58,409	**1,482**	**453,861**	**$1,291,971**	**7.77**	**$22.12**



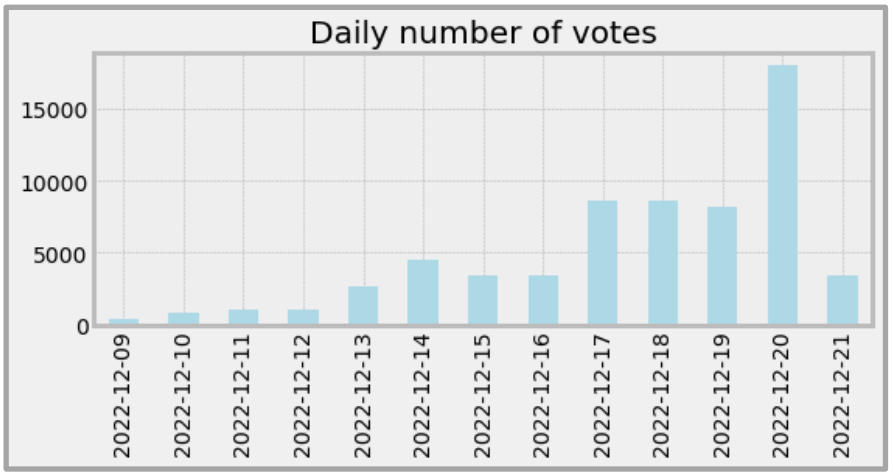


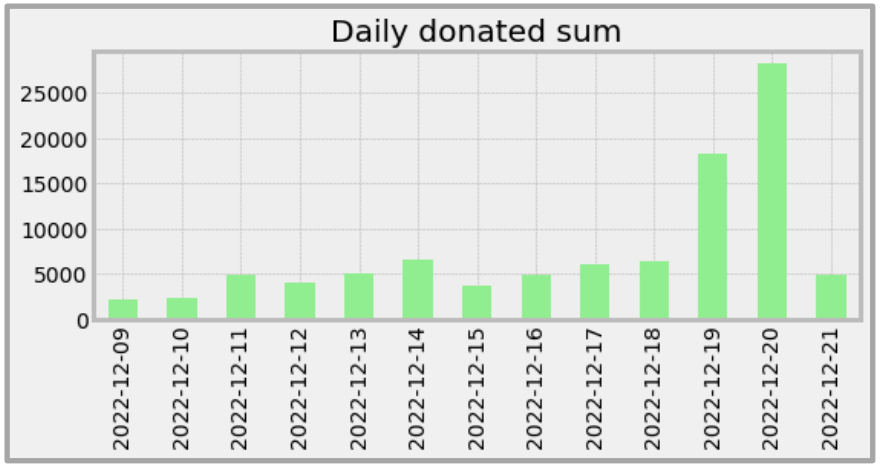


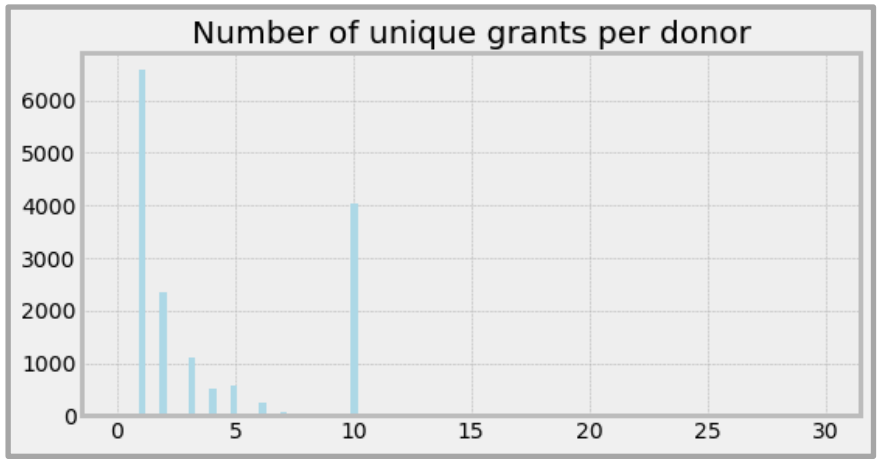


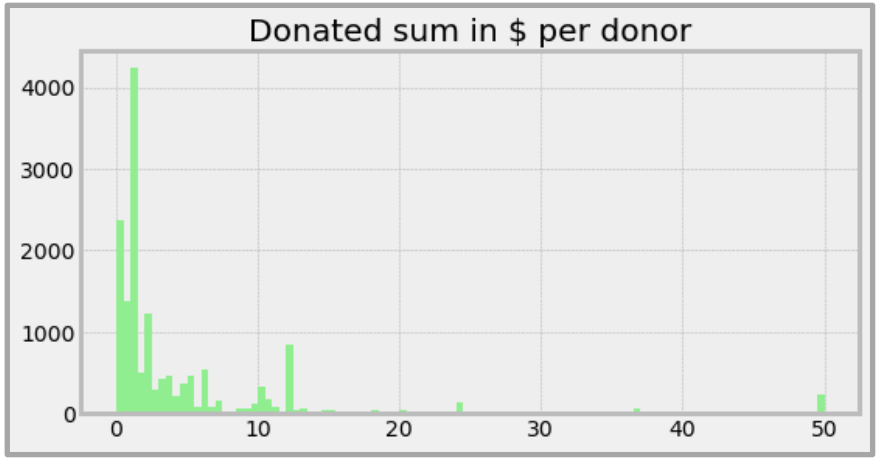
2.2. Summary dashboard for UNICEF round

Number of donors:	Number of projects:	Number of votes:	Donated sum:	Average number of votes per donor:	Average donated sum per donor:
15,510	**10**	**64,180**	**$96,996**	**4.14**	**$6.25**



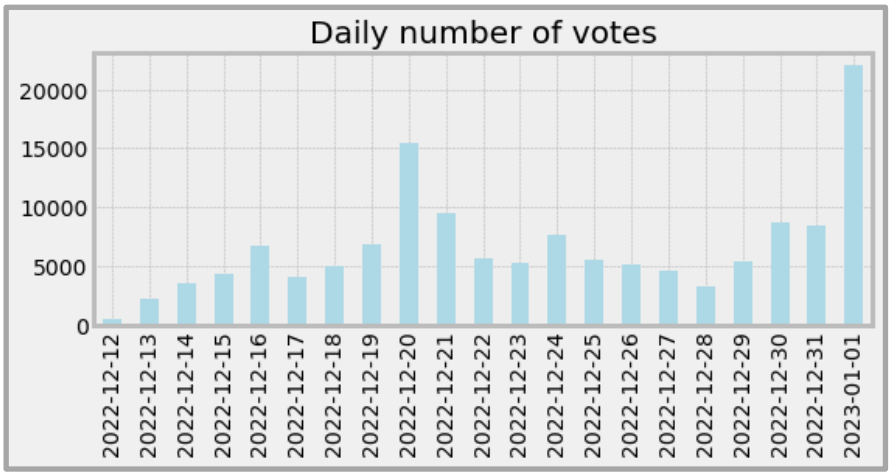


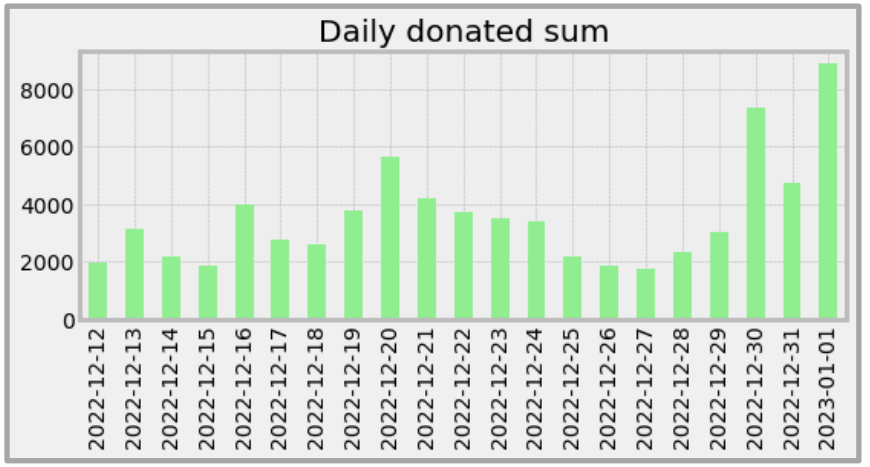


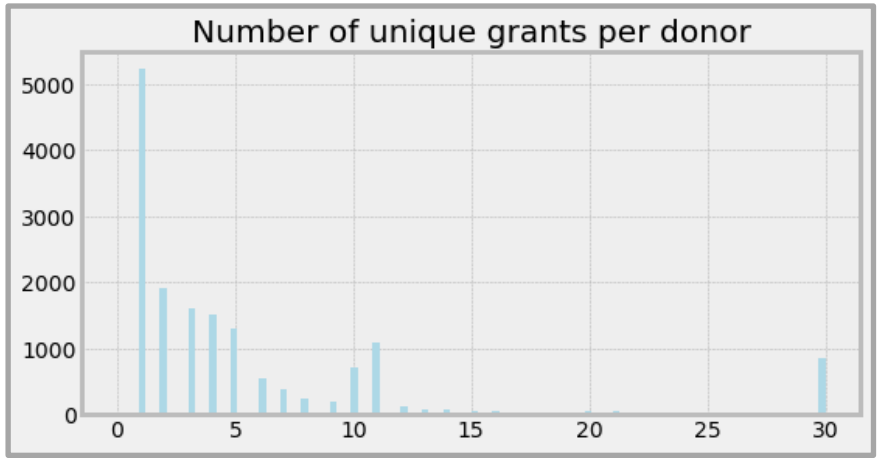


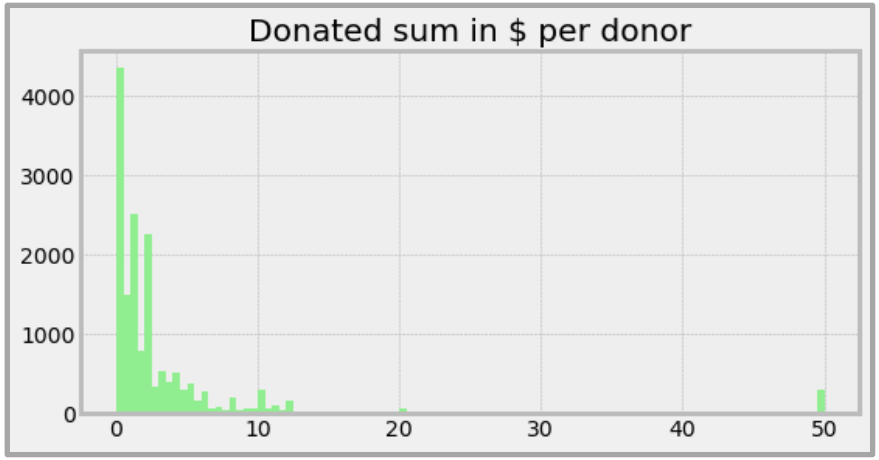
2.3. Summary dashboard for Fantom round

Number of donors:	Number of projects:	Number of votes:	Donated sum:	Average number of votes per donor:	Average donated sum per donor:
16,344	**88**	**139,337**	**$74,704**	**5.18**	**$24.15**



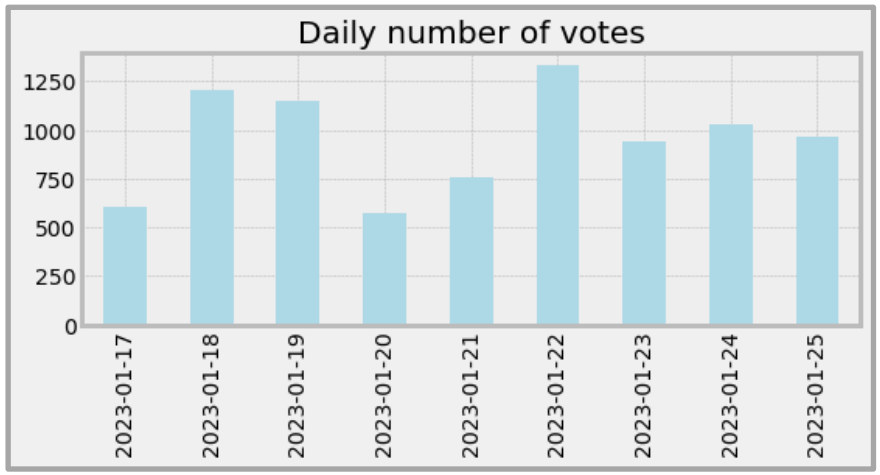


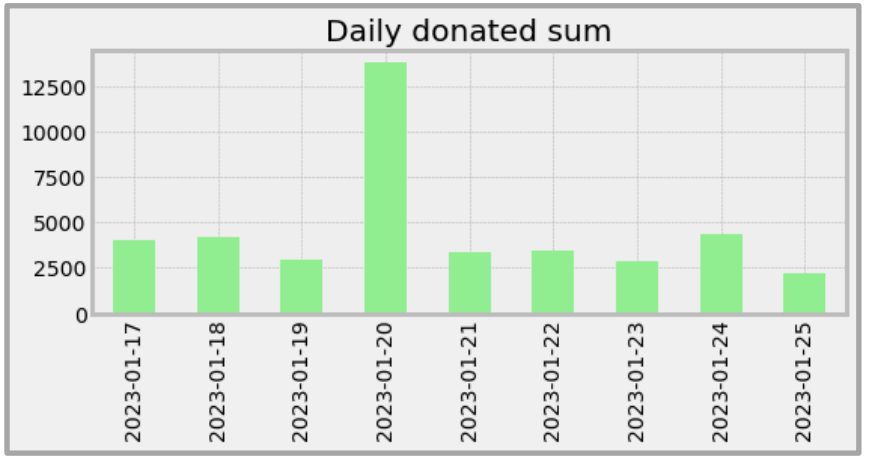


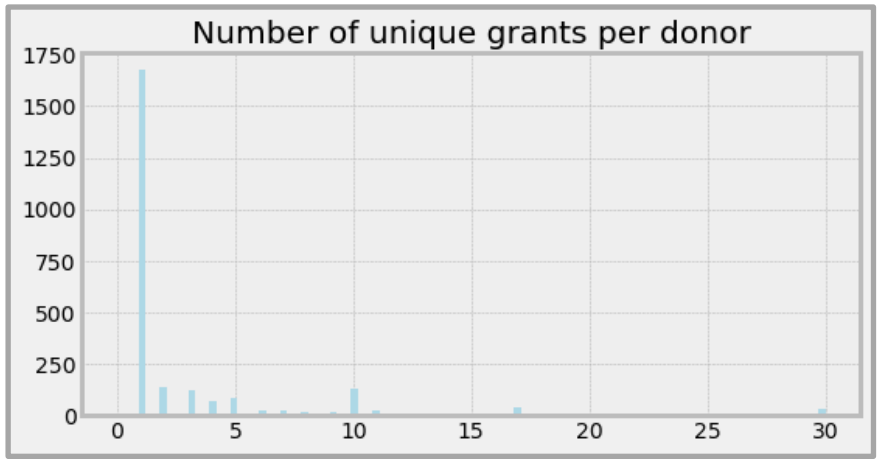


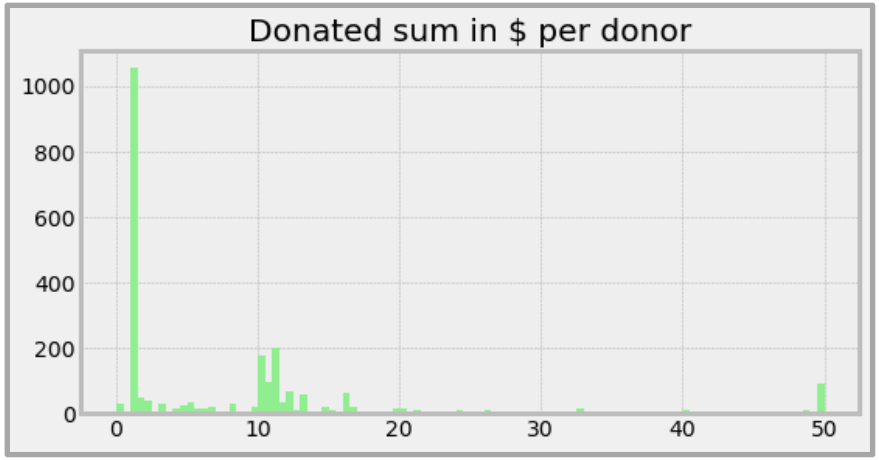
2.4. Summary dashboard for Gitcoin Climate Solutions round (not full)

Number of donors:	Number of projects:	Number of votes:	Donated sum:	Average number of votes per donor:	Average donated sum per donor:
3,399	**66**	**10,462**	**$46,142**	**3.08**	**$13.58**



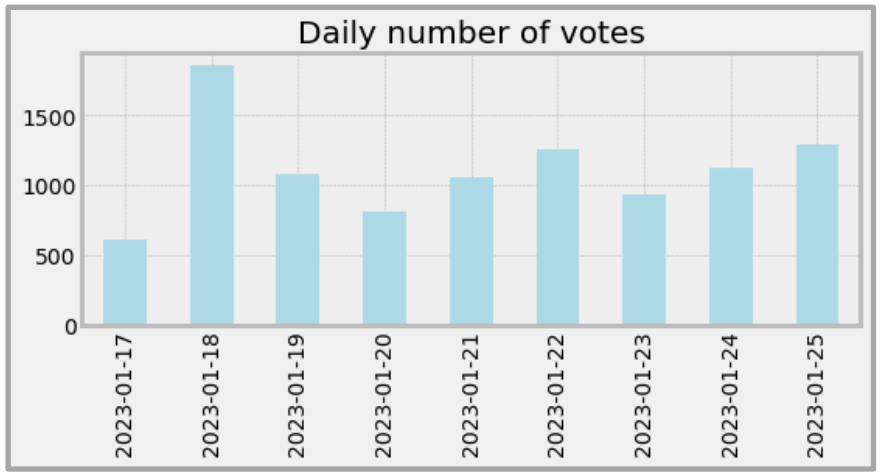


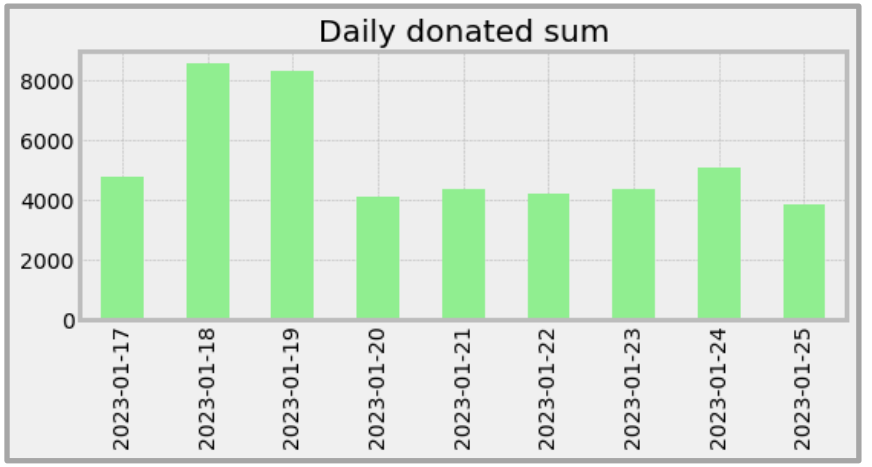


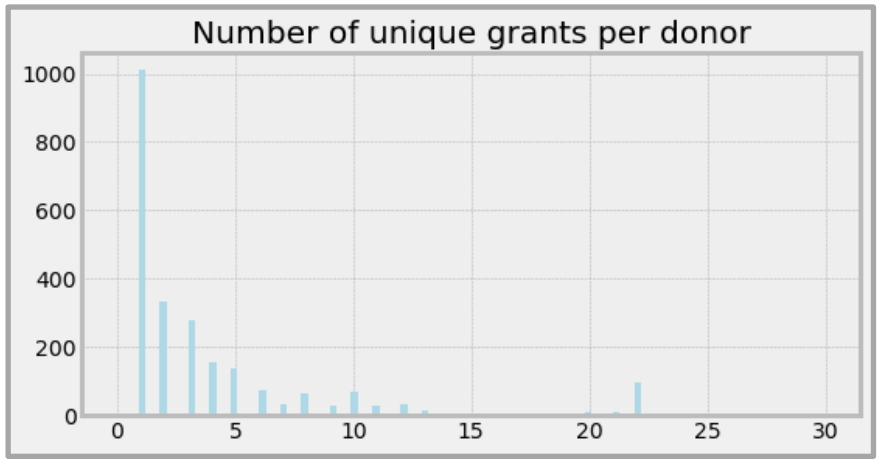


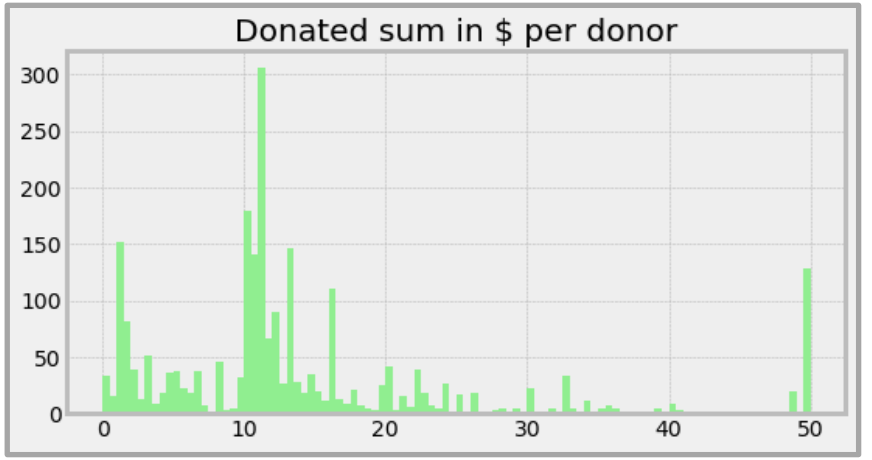
2.5. Summary dashboard for Gitcoin Ethereum Infrastructure round (not full)

Number of donors:	Number of projects:	Number of votes:	Donated sum:	Average number of votes per donor:	Average donated sum per donor:
2,901	**24**	**12,022**	**$54,137**	**4.14**	**$18.66**



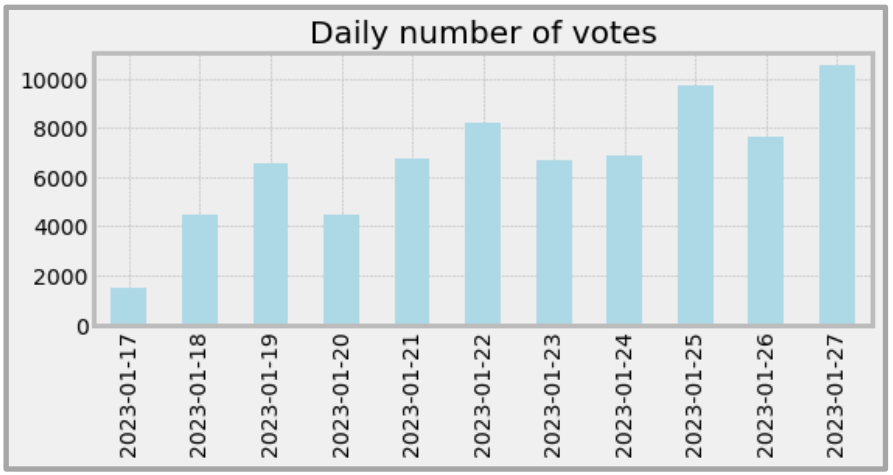


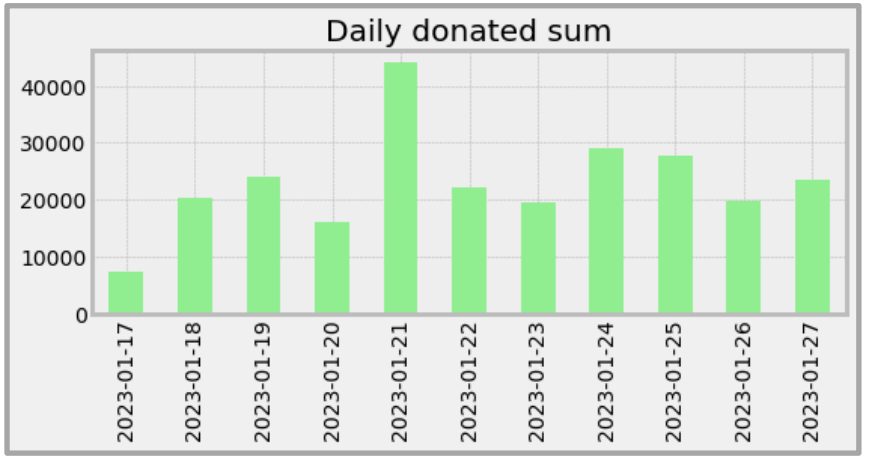


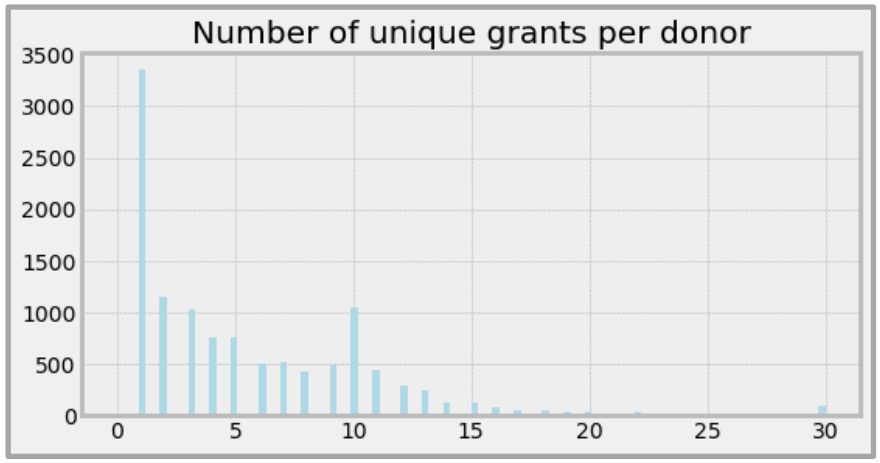


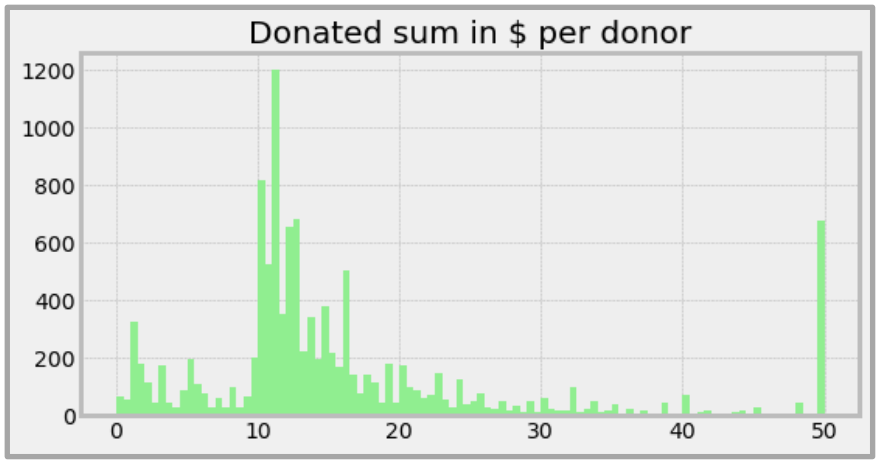
2.6. Summary dashboard for Gitcoin Open Source Software round (not full)

Number of donors:	Number of projects:	Number of votes:	Donated sum:	Average number of votes per donor:	Average donated sum per donor:
11,796	**89**	**73,413**	**$253,288**	**6.27**	**$21.64**









3.1. Hash popularity distribution algorithm to find potential Sybil attacks

For indentifying of Sybil attacking wallets we developed Python script and applied it to the different Gitcoin rounds.

Data preprocessing:

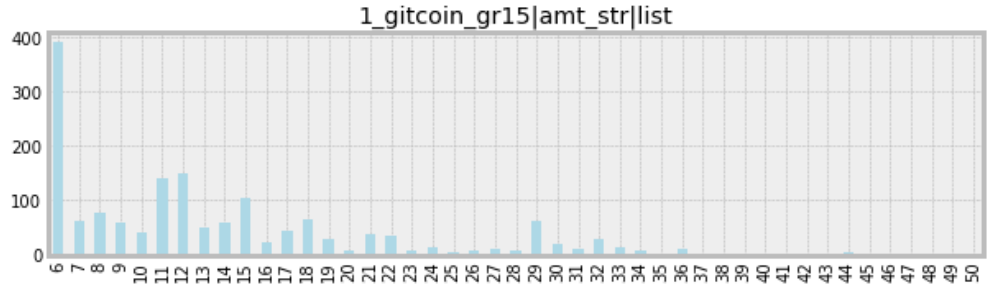
- In the voting dataset let's create new feature "amt_str" by concatenating token and rounded donated sum in usd.
- We applied our algorithm separately for “destination_wallet” and “amt_str” features. For each of them there are “unique” and “list” methods.
- Sort input table by the time of voting for the “list” method. There is no need to sort if we use the “unique” method.

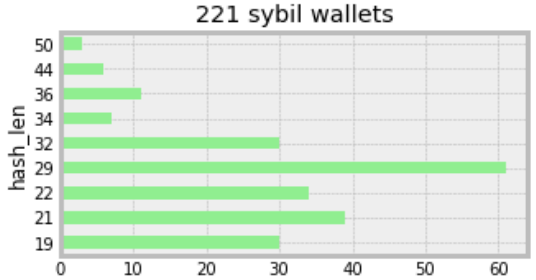
Algorithm:

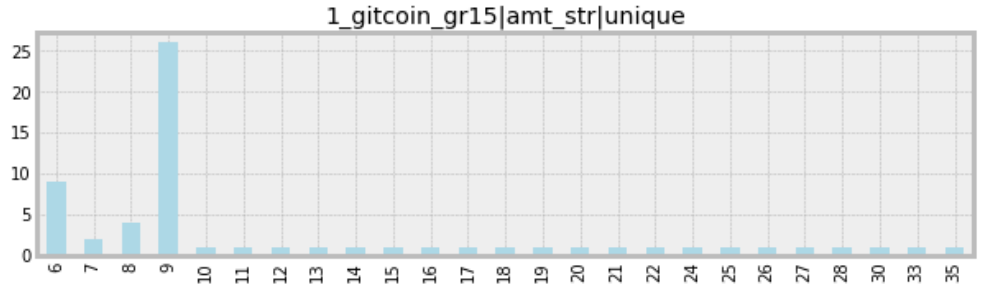
- Group by the source wallet and calculate aggregation as the list of all values (or sorted list of the unique values).
- Create the new hash feature **hash** by concatenating the values in that list.
- Calculate the new count feature **hash_length** as the number of values in that list.
- Group by **hash** and **hash_length**, calculate the number of rows (wallets) in the group.
- Find the maximum number of wallets with the same **hash** grouped by **hash_length**. This series have returned by the Python function for the future visualizations.
- Based on that series values we define potential Sybil attack events the following way:
    - Compare the values from that series with their neighbors and if one of them in **neigh_coef** times less this value then it is Sybil attack case. Experimentally we set **neigh_coef = 3**.
    - Exception: if the **value >= max_value** then it is not Sybil attack. Maybe it is normal situation when a lot of wallets votes for the same top list. Experimentally we set **max_value = 100.**
    - Exception: if **hash_length <= min_hash** then it is not Sybil attack.
    - Based on the visualization we can set the lists what **hash_length** values we want to manually add or drop.
- Then the next step is to find wallets having this hash and return them.

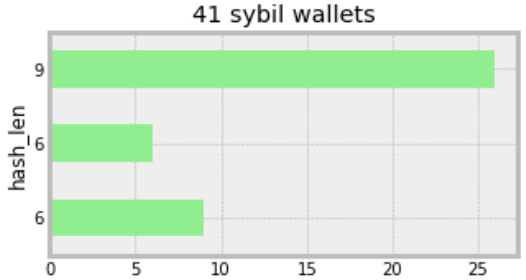
*More details you can find in the following script: *03.sybil_searching.ipynb*

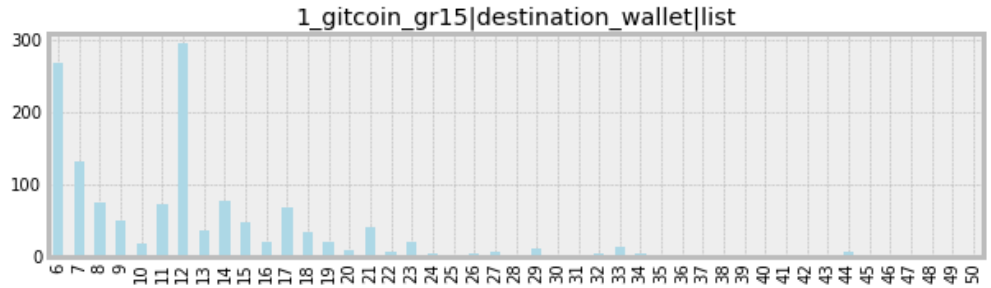
3.2. The algorithm applying to the Gitcoin Round 15

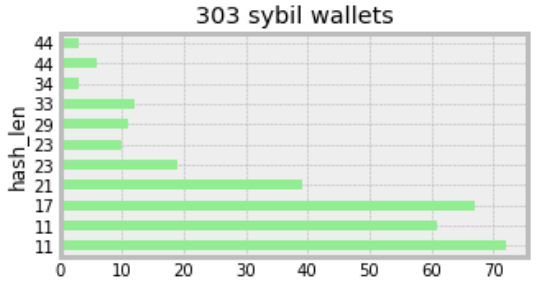


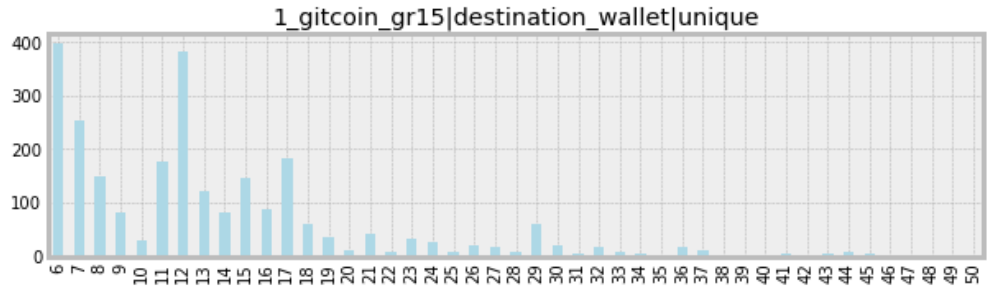


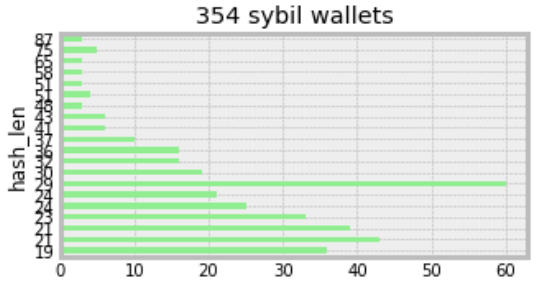












*More details you can find in the following script: *03.sybil_searching.ipynb*

3.3. The algorithm results

- After applying hash popularity distribution algorithm we received the following table:

(1970, 6)

	hash	source_wallet	hash_len	feat_round	feat_target	target_type
0	ETH=1.00;ETH=1.00;ETH=1.00;ETH=1.00;ETH=1.00;E...	0x0d578a998f3e6d1959acf5cf7e83078212b767ca	19	1_gitcoin_gr15	amt_str	list
1	ETH=1.00;ETH=1.00;ETH=1.00;ETH=1.00;ETH=1.00;E...	0x16d7d09d45e76abc63c0999266ec1590f3f08431	19	1_gitcoin_gr15	amt_str	list
2	ETH=1.00;ETH=1.00;ETH=1.00;ETH=1.00;ETH=1.00;E...	0x1a7f8272fb94eff33c51aa1e22cd0c6067f801d9	19	1_gitcoin_gr15	amt_str	list
3	ETH=1.00;ETH=1.00;ETH=1.00;ETH=1.00;ETH=1.00;E...	0x22b2c3055725b847caecebb081feb3da57df3754	19	1_gitcoin_gr15	amt_str	list
4	ETH=1.00;ETH=1.00;ETH=1.00;ETH=1.00;ETH=1.00;E...	0x2532000e2e647b9b6d50eeae39726d6ee029d030	19	1_gitcoin_gr15	amt_str	list

- Then we grouped by the feat_round + hash and collected the set of wallets in this group.
- Then we grouped by round_name and generated all possible pairs sorted(“wallet”+”wallet_add”). We will use them for future research dealing with transactional data.

(23983, 2)

	round_name	wallets
0	1_gitcoin_gr15	0x8e6327a2513c4e9ceaf1080d5cb35442b737bf40_0xc...
1	1_gitcoin_gr15	0x0c85891c036ecd0dd902ebb722a4ecb71e8e0d12_0x8...
2	1_gitcoin_gr15	0x0e5610b7e42e134d9d57224db6b47ce75afa87c9_0x8...
3	1_gitcoin_gr15	0x8e6327a2513c4e9ceaf1080d5cb35442b737bf40_0xa...
4	1_gitcoin_gr15	0x5ba1031bb03110d4443f5df83002338428c4c8eb_0x8...

*More details you can find in the following script: *03.sybil_searching.ipynb*

4.1. Generating wallets pairs and features on the Ethereum transactions dataset

For each Gitcoin round we generated all the pairs of donated wallets.
Then we calculated the following aggregations for the different time periods.

1. **Aggregations:**
   - Tenure – how many days from the first transaction to the end of period
   - Frequency – number of transactions
   - Monetary – transactions amount
   - Consistency – number of active days with transactions
2. **Filters:**
   - All Ethereum transactions for the last year till round ends (_year)
   - All Ethereum transactions for the round period (_round)

Output example:

(37612, 10)

	round_name	wallets	tenure_year	frequency_year	monetary_year	consistency_year	tenure_round	frequency_round	monetary_round	consistency_round
0	1_gitcoin_gr15	0x0000ce08fa224696a819877070bf378e8b131acf_0xa...	297.27	2.00	0.21	1.00	NaN	NaN	NaN	NaN
1	1_gitcoin_gr15	0x0003a2d21b35c7cfc0fb259c9e27dbdb434864bd_0x3...	31.70	1.00	0.01	1.00	NaN	NaN	NaN	NaN
2	1_gitcoin_gr15	0x0003a2d21b35c7cfc0fb259c9e27dbdb434864bd_0x4...	274.64	1.00	0.04	1.00	NaN	NaN	NaN	NaN
3	1_gitcoin_gr15	0x0003a2d21b35c7cfc0fb259c9e27dbdb434864bd_0x4...	274.63	1.00	0.02	1.00	NaN	NaN	NaN	NaN
4	1_gitcoin_gr15	0x00041f83818286276bd5a7507088b7c4dff1c5a4_0xa...	62.99	1.00	0.63	1.00	NaN	NaN	NaN	NaN
5	1_gitcoin_gr15	0x00041f83818286276bd5a7507088b7c4dff1c5a4_0xd...	63.05	1.00	0.63	1.00	NaN	NaN	NaN	NaN
6	1_gitcoin_gr15	0x000566b53e028d21e104e4320de61c2314ef4064_0xf...	275.53	1.00	0.14	1.00	NaN	NaN	NaN	NaN
7	1_gitcoin_gr15	0x000ad8bc3dfbe42d9a87686f67c69001a2006da4_0x1...	182.68	4.00	0.46	4.00	NaN	NaN	NaN	NaN
8	1_gitcoin_gr15	0x000ad8bc3dfbe42d9a87686f67c69001a2006da4_0xd...	88.80	2.00	0.07	2.00	5.73	1.00	0.01	1.00
9	1_gitcoin_gr15	0x000adcacec318ecf629ae42ce1d48d168965bfc5_0x6...	2.57	1.00	0.00	1.00	2.57	1.00	0.00	1.00

*More details you can find in the following script: *13.generate_features.ipynb*

4.2. Output file example and statistics of the Sybil attack wallets received

- We have merged wallet pairs received by the hash algorithm and pairs in the transactional dataset.
- In the output file have collected all the wallets received by the hash algorithm and checked if there is a linked wallet in transactions for the last year (flg_trnx_year) and for the round period (flg_trnx_round).
- We generated the following output file that is available in the Ocean Market here: https://market.oceanprotocol.com/profile/0x4AD829A7609357a7D05737526FE13bbDd18427b5

(1615, 5)

	round_name	wallet	flg_hash_algo	flg_trnx_year	flg_trnx_round
0	1_gitcoin_gr15	0x00a41e72a3bf61c444a021b63a0d5f5af0dc1b83	1	0	0
1	1_gitcoin_gr15	0x00f7b390192724d35b48286406fd6cbe650e0255	1	0	0
2	1_gitcoin_gr15	0x0104d476c19c6c3f9b5536f6aaf62ac4bee9188b	1	1	0
3	1_gitcoin_gr15	0x0212535d92b4fabc447e88e0676cfe3b233730bc	1	1	1
4	1_gitcoin_gr15	0x0356c28c048b52b568e35edd834e463ae2ef723b	1	0	0

- The statistics of the number of wallets in total and the number of potential Sybil attack wallets:

	round_name	Number of donors	Connected donors per year	Connected donors per round	Sybils by hash algorithm	Sybils approved by trnx per year	Sybils approved by trnx per round
0	1_gitcoin_gr15	58409	18154	5722	729	194	99
1	2_unicef	15510	4490	1429	157	12	2
2	3_fantom	16344	4259	1370	251	124	43
3	41_gitcoin_climate	3399	274	99	142	41	27
4	42_gitcoin_ethereum	2901	578	130	167	14	6
5	43_gitcoin_oss	11706	2744	637	169	62	33

*More details you can find in the following script: *14.explore_wallets.ipynb*

5. Progress and future work

Progress:

- We collected all voting datasets together and developed hash algorithm to find potential Sybil attack wallets who have similar donation patterns.
- Based on the analysis of Ethereum transactions we found all the pairs of wallets who were active in every particular round and generated the statistical features for future explorations.
- By merging mapping wallet pairs returned by hash algorithm with transactional pairs we could verify if the potential Sybil wallets connected with each other or not.
- We saved the dataset of potential Sybil wallets on the Ocean market here: https://market.oceanprotocol.com/profile/0x4AD829A7609357a7D05737526FE13bbDd18427b5

Future work:

- We can improve hash algorithm by mapping wallets that doesn't have exactly the same pattern but the similar (let's say 1-2 different positions).
- Now we looked at the voting rounds independently. We can improve it by using statistics on previous rounds.
- Now we have collected only the pairs of wallets who were directly connected with each other. We can improve it by applying graph-based methods.
- Now we have used transactions data only for verifying the existence of connection between wallets. But we can develop semi-supervised machine learning models on the features that we have generated and the targets that we can receive by mapping algorithm.
- If we will find the datasets with Sybil attack wallets it will be possible to train ML model on them.