Stellar Analytics Library

This library is being created with the following goals:

* Ability to easily query information from the stellar network
* Define the activity of the Stellar users
* Draw conclusions from that activity
* Leverage that information to best understand the most effective ways of amplifying usage, effectiveness, and awareness of the stellar network
* The library may be used to store network data in a relational database in order to run real time analytics on how the network and users are behaving

In order to best achieve these goals the network data must be transformed and organized in a manner where we can easily query for specific information and define certain behaviors. Some examples might be:

* What is the breakdown of account balances by region, country, age , sex etc.
* What is the average transaction amount, outflow amount, inflow amount for each account
* When in the lifetime of an account did the account balance reach its max balance point, what about the minimum balance point?
* How much time elapsed from the max account balance time to the min (can identify m-turkers)
* What is the frequency of each transaction inflow amount (e.x. how many inflows were there of exactly 3500 STR – can identify users soliciting m-turkers/odeskers)
* What is the gross STR volume of a given account?
* What is the average daily gross volume for a given account?
* What does the time series of the account balance look like ?
* What type of account did the majority of the outflows go to (exchanges?)
* What is the breakdown of account behavior by demographic?

The library may be used to compile a unique set of all account ids that have ever appeared on a ledger. This can be done by traversing through every ledger through the “ledger” api call, retrieve the “transactions” vector (by way of “result” and “ledger”) and appending each unique “Destination” onto a map. We use “Destination” because presumably each active account must have been sent some STR at some point except (even the “genesis” account has received STR). The unique account keys should be stored in a database and only updated for new accounts so that the list does not need to be re-populated every time we want to run a query on the data

For a given set of account ids, the library may be used to organize the data into the following tree (map in clojure) where each account id is a key to a map with that account’s relevant information on it. Currently the account has :transactions, :statistics, and :other-data maps defined but the structure is dynamic such that additional categories can be defined.

Account-id (the account id ex “gwJSfNRWbYKSSF4Z6GWfv5wMsG72hQ2DzZ”)

:transactions (contains a vector of maps for each transaction ever made in the given account. This is the same as the “tx” map in the “transactions” vector that is retrieved from the “account-tx” API method by way of :result. Each transaction map contains the following info:)

:TransactionType (e.g. “Payment” “AccountSet”)

:date (in seconds since Jan 1 2000)

:Amount (amount of STR that was transaction in mm)

:Account (the account that sent the STR)

:Fee (the transaction fee in STR)

:Destination (the receiver of the STR in the transaction)

:inLedger (the ledger index of this transaction)

:ledger\_index (not sure how this is different from above)

:LastLedgerSequence

:TxnSignature

:hash

:Flags

:SigningPubKey

:Sequence

:statistics (contains a map of descriptive statistics about the transactions in the given account)

:inflows (contains a map of inflow statistics)

:total-coins (the gross number of STR that has come into this account)

:total-txns (the total number of inflow transactions)

:avg-txn (the average amount each inflow transaction was)

:txn-frqcy (contains a map with keys equal to inflow transaction amounts and values equal to the number of times an inflow transaction on the ‘key’ amount occurred in the account)

:outflows (contains a map with the same keys as inflows but for outflow data)

:max-bal (the highest account balance this account has had)

:min-bal (the lowest account balance this account has had)

:max-time (the date/time corresponding to when the account was at max balance)

:min-time (the date/time corresponding to when the account was at min balance)

:avg-balance (the average balance calculated as the sum of the balances each time the account had a transaction divided by the total number of transactions)

:current-balance (the current balance of the account)

:time-dif-max-min (the max-time – the min-time)

:balance-over-time (contains a vector of vectors where for each subvector the first value is the date/time and the 2nd value is the balance of the account at the given date/time

:demographics (contains a map of demographic and other data about the account – can be populated once integrated to facebook api)

:age

:location

:sex

:ethnicity

:education

:other-data (other data not currently populated)

:email-address

:facebook-id

:trust-lines (contains a vector of maps with the following keys)

:trust-acct

:trust-amt

:trust-ccy

Once all the transaction data is processed per above I would define certain behavior types and cluster the accounts by those defined types. I would likely add additional mapping to the other-data to laber the accounts by their behaviors. Some possible behavior types might be:

* Connector/Evangelist. To determine if a person is evangelizing Stellar I might look at how many facebook friends of that person also have a Stellar account. Of those facebook friends that do have Stellar accounts, has this person sent or received STR to and from those accounts. A high ratio would indicate this person is a huge proponent of Stellar. I would look to see if there are any common patterns or demographic characteristics of these proponents and see how we could target more of them
* Freeloaders (m-turkers). To determine if someone is a freeloader or someone trying to make a quick buck I would look if their account was depleted soon after their initial Stellars were deposited or after their account achieved max balance. If there accounts were depleted soon after their max balance was achieved I would look to see what accounts those STR were being transferred to. Were they going to an exchange such as Justcoin or were they being transferred to an individual who was taking advantage of these freeloaders. Were they going to someone outside of their facebook network or outside of the country? Once I have identified them I would again look for patterns in the demographics of these type of accounts. I would then aim to educate these types of people about Stellar and try to change their behavior for the better.
* Schemers. These types of individuals are the ones arbitraging the system, getting strangers to sign up for the free giveaway and buying their Stellars from them for below market value. Since we have a frequency map, we can look at the inflows for that account. If this individual has a pattern of receiving a high percentage of a certain amount (5000, 3500) and receiving them from individuals in other countries or outside of their facebook network they are likely the ones doing the arbitraging.
* Traders. These individuals are likely transferring larger quantities or a high percentage of their balances of Stellars to and/or from some of the many exchanges that list Stellars. These individuals are valuable to learn from to help improve the liquidity on the network.
* Noobs. These individuals are new to crypto currencies. They can be characterized by possibly inactivity or by cross referencing their emails to forums and blogs to see if they have posted any questions or are new to the space. The demographics and characteristics of these individuals are extremely important to identify in order to discover what about Stellar they are attracted to and who is most likely to benefit from and use the network.

By mapping the account transaction data per the above schema and characterizing the types of accounts by their transactional behavior we can now answer the example questions we might want to ask with relative ease.

* What is the breakdown of account balances by region, country, age , sex etc.
  + For given demographic keys, sum :statistic-data :current-balance for every account in that demographic and divide by the total of issued STR
* What is the average transaction amount, outflow amount, inflow amount for each account
  + Average inflow/outflow amount is in :statistics :inflows/:outflows :avg-txn
  + Average overall transaction amount is = (:statistics :inflows :total-coins + :statistics :outflows :total-coins) / (:statistics : inflows :total-txns + :statistics :outflows :total-txns)
* When in the lifetime of an account did the account balance reach its max balance point, what about the minimum balance point?
  + This can be found in :statistics :max/min-time
* How much time elapsed from the max account balance time to the min (can identify m-turkers)
  + This can be found in :statistics :time-dif-max-min
* What is the frequency of each transaction inflow amount (e.x. how many inflows were there of exactly 3500 STR – can identify users soliciting m-turkers/odeskers)
  + This can be found in :statistics :inflows :txn-frqcy :(transaction amount i.e. 5000)
  + Divide the above by :statistics :inflows :total-txns to get the percentage of all inflows are accounted by that amount
* What is the gross STR volume of a given account?
  + (:statistics :inflows :total-coins + :statistics :outflows :total-coins)
* What is the average daily gross volume for a given account?
  + This is equal to the above / (current date – date of the first transaction which is :transactions [0 :date]
* What does the time series of the account balance look like?
  + This is stored as a vector of vectors in :statistics :balance-over-time
* What type of account did the majority of the outflows go to (exchanges?)
  + Sum the amount (:transactions [n :Amount]) where :transactions [n :Destination] has the account characteristics you are looking for.
* What is the breakdown of account behavior by demographic?
  + For a given demographic what percentage of accounts fall into each account behavior category.