Modeling Complex Systems – concluding assignment

In this assignment we'll simulate an imaginary evolving economy and use that simulation to study the wealth distribution within that economy and the effect of an individual's behavior on that individual's wealth.

The assignment is multifaceted in the sense that it requires skills in several areas: learning new topics, problem-solving, and programming.

You were hired to advise the government of *Zubawewe* – a tiny tribe-nation that has just received its independence from Britain. The new government would like to establish a free economy, yet exercise some level of control over it, in an effort to avoid formation of large disparities and inequalities within its population and, as importantly, attempt

to prevent poverty. Some ideas regarding laws that need to be applied towards that end have been considered. Your mission is to test the long-term effect of these laws (rules), via simulation, and advise the government regarding the repercussions of their application.

As part of the process of forming a new economy in the state, a new currency will be created and the government wishes to hand out an equal amount of the new currency to all its current citizens. Moreover, it also wishes to grant a fixed amount to each newborn child (as long as both parents are citizens). The ideas the government wishes to test in are detailed below.

The required simulation has two main components –

- a growing/changing **network**, where each node represents an agent (person) and the edges connected to a node represent this agent's connections;
- A system of inter-agent wealth transactions wherein agents transact some of their wealth.

A third component of the simulation is taxation, as explained below. (This part is canceled.)

The implementation should be done in Python. It is recommended that you use the *Networkx* package to implement the network, although you can use/write other tools for that purpose, if you so wish.

Each node has several characteristics (traits), as detailed below. Three of these traits need to be drawn from random distributions (which are based on behaviors observed in surveys of the *Zubawewean* population). For **each node** ("individual"), you should keep track of the parameters characterizing that node. The parameters include:

1. **saving-propensity** σ_p – this value represents the agent's inclination to save and is randomly drawn, for each node, from a beta(2, 10.7) distribution, shifted by 0.01, thereby forcing a minimum saving requirement, and capped at 0.7.
   i.e., $min(0.01 + np.random.beta(2, 10.7), 0.7)$.

2. **Social-propensity** ϕ – this value represents the agent's social inclination (inclination towards connecting with other agents) a randomly drawn standard-Normal distributed value. (Thus, the average agent social propensity is 0)

3. **Business-propensity** β – this value represents the agent's business inclination (or willingness to take risks) and is randomly drawn, for each node, from a lognorml distribution, using the following formula:
   $$min((np.random.lognormal(mean = 3.4, sigma = .55) + 1.4)/350, 0.5)$$

4. **Birth** – a timestamp of when the node was created,
5. **Wealth** w – initially, at "birth", set to 100. (This is the amount the government intends to grant each existing citizen as well as each newborn.)

The simulation should be run for 35,500 timesteps. Throughout the run, *two* processes take place:

1. Network growth/alteration,
2. Inter-agent wealth transactions,

3. A third process – taxation, takes place every 1,000 steps (this part was removed from the work and shouldn't be implemented).

Initial network:

Initially, the network is a 200-node E-R graph with $p = 0.15$, ie, each existing node is connected to an average of 30 nodes. The internode connections (edges) are to be randomly generated. This initial state represents the current population of Zimbawewe

Network dynamics:
Once the initial network is set up, the dynamics of the system are characterized by **three** processes that take place at various times; namely: node deletion, node generation, connections (edges) alterations.

I. Nodes get deleted (die) when they reach an "age" of $t = 18{,}250$. At that point, their wealth is equally distributed between all of their immediate $(k = 1)$ connections.
NOTE: removal of "deceased" nodes should be done *prior* to addition of new nodes, i.e., the calculation of the number of nodes that need to be added at time t should be done after the removal of deceased nodes.

II. Upon the removal of a node, its entire wealth should be equally distributed between all of their immediate $(k = 1)$ connections

III. If a node doesn't have any connections (ie, has 0 K=1 neighbors), its entire wealth should be allocated to an "account" called *treasury*, which accrues all such funds.

IV. Once the initial network is set-up, every timestep $(1 \dots N)$ new nodes are generated, according to

$$N(t) = int\left(nodes_i + \left(nodes_f - nodes_i\right) * \left(\frac{1}{1+\exp\left(-\left(\frac{t}{3500}-1\right)\right)} \right) \right)$$

where, $N(t)$ is the required number of nodes at time t, $nodes_i = 200$ is the number of nodes in the initial network, $nodes_f = 41000$ is a limiting factor on the total number of nodes.

The number of required nodes (according to the above expression) is to be compared to the actual number of nodes and the difference, if any, should be added. For example, if at some point in time 232 nodes are required but only 230 exist, 2 nodes should be added.

The upper cap on the population is a result of the governments belief that this is the largest population *Zubawewe's* natural resources can support and they intend to enforce reproduction limit laws if/when the population reaches that level.

V. Each *newly generated* node (i.e. a node added after the initial network is set) is initially connected to 2 randomly selected **interconnected nodes** ("parents") and to *all* of the immediate $(k = 1)$ connections of those two nodes. Note that the 2 parents must be interconnected, i.e. 1st degree neighbors.

Each newly generated node is assigned values for σ_p, β, and ϕ, using the same formulas used for the initial network, and an initial wealth $w = 100$.

VI. Every 500 timesteps since a node’s "birth", a node makes (or removes) connections, according to the description below, with the exception that a node must maintain a minimum of 2 connections. To clarify, each existing node modifies its connections when its age is 500, 1000, 1500 etc. The number of nodes to add/remove is determined by:

$$int(np.round(np.random.normal(loc = phi\_i),\ 0))$$

where ϕ_i is the node's value of ϕ. Once the number of nodes to add/remove is determined, addition/removal takes place.

If connections (edges) are to be removed, the edges for removal are chosen randomly from among the node's NN connections (ie, $k = 1$) and removed.

If nodes are to be added, then the nodes to connect to are randomly chosen from within nodes that are **not** currently NN of the relevant node (ie, **not** $k = 1$). For example, if we need to add 1 node to node #55, then we need to randomly select 1 node from within the nodes that are not currently connected to #55 and add a connection (edge) between #55 and that node.

Transactions:

I. Transactions between agents are made as described below. The transactions should be made *after* all network updates for that timestep have been made.

II. Transactions are made every 3 timesteps (ie, at times 3, 6, 9, 12, etc). At each transaction time, we first divide the existing nodes into two equal-sized groups, which we'll designate **A** and **B**. If the overall number of nodes at that time is odd, one node should be randomly selected and removed from the transacting nodes. NOTE: the node itself remains, only it doesn't participate in the transactions at that timestep.
At each transaction time, payments are made by members of group **A** to members of group **B**, ie, unilateral payments. (Since at each transaction the nodes are assigned into the paying and receiving groups anew, averaging over time, each node makes and receives payments approximately equal number of times.)

III. Coupling the payer node with a receiver node involves their betas (β) and is done in the following manner: Each group is sorted according to its members' beta and matching nodes (by rank) are coupled.
For example, if node #55 has the highest beta among the payers group and node #237 has the highest beta among the recipients group, then these two nodes are "coupled" for that transaction.

IV. The size (amount) of **each** transaction is determined by:
$payment = payer\_wealth * (1 - payer\_sigma) * (mean_{beta} * max(np.random.beta(2,5),\ 0.01))$
where payer-sigma is the sigma of the payer, np.random.beta(2,5) is a random draw from the beta(2,5) distribution, and $mean_{beta}$ is the arithmetic mean of the beta values of the payer and receiver nodes.

V. A transaction only takes place when the paying agent has sufficient wealth to make it, i.e., its post-transaction wealth >= 0. When a payer doesn't have sufficient funds to transfer the amount determined in IV above, the transaction is canceled (skipped).

Taxation: (No need to implement!)

Simulation:

Run the simulation 35,500 timesteps (if you have access to a powerful platform, you can run 55E3 timesteps). Every 5E3 steps (as well as at time=0 and at time=35,500) report the following statistics:

- Plot a histogram of the wealth distribution
- Plot the wealth as a function of sigma (the saving predisposition) + a linear regression line
- Plot the wealth as a function of beta (the business/risk predisposition) + a linear regression line
- Plot the wealth as a function of phi (the social predisposition) + a linear regression line

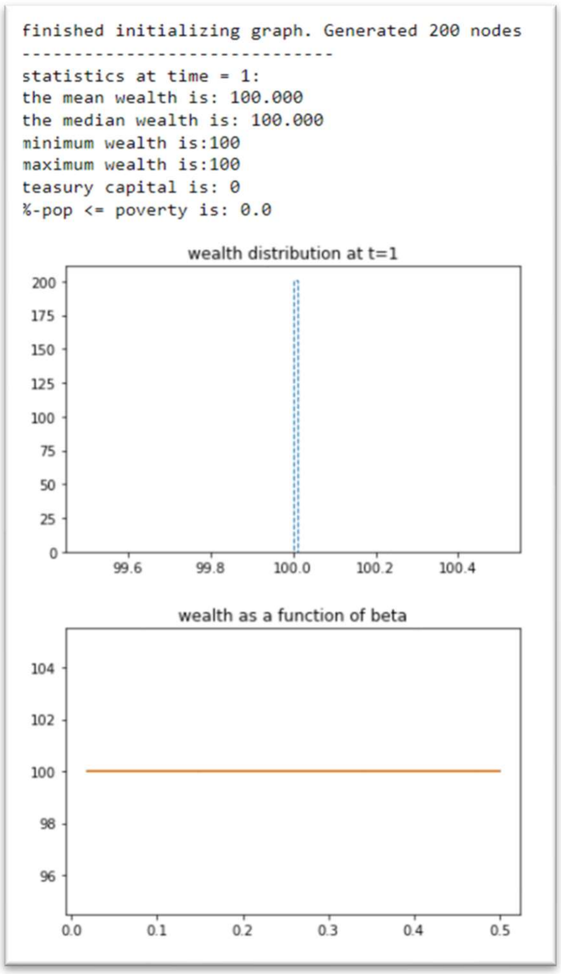
In addition, print the following statistics (as numerical values):

- mean wealth,
- median wealth,
- min(wealth), i.e., agent with the smallest wealth,
- max(wealth), i.e., the agent with the highest wealth,
- percent population below the poverty line (defining the poverty line as 35)
- (treasury capital. – no need to implement this).

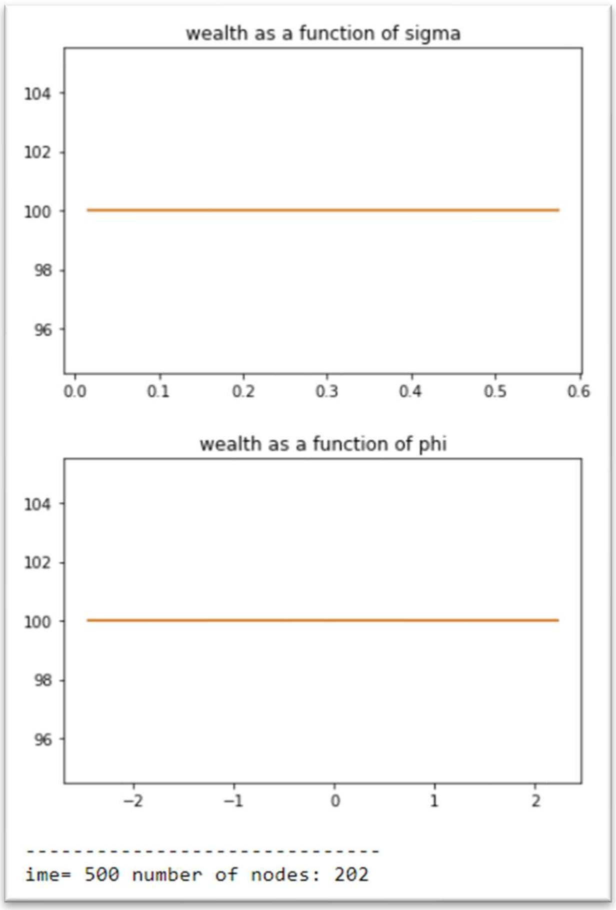
GOOD LUCK

Sample screenshots of the required output

Below are some screenshot samples of the required output. These are merely samples. While your output needs to contain the same information (as described above), there's no need to adopt the format shown below.



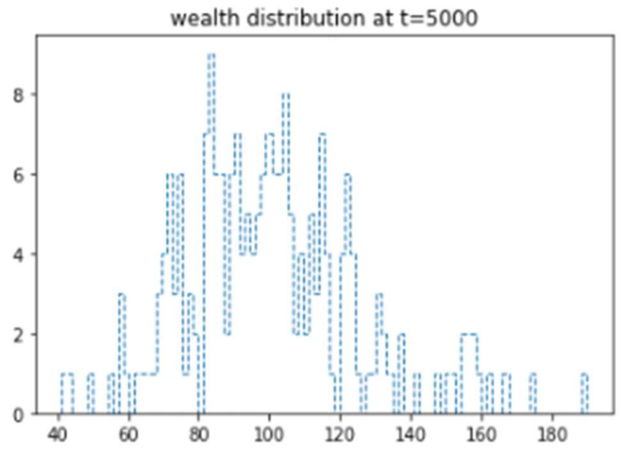
```
finished initializing graph. Generated 200 nodes
------------------------------
statistics at time = 1:
the mean wealth is: 100.000
the median wealth is: 100.000
minimum wealth is:100
maximum wealth is:100
teasury capital is: 0
%-pop <= poverty is: 0.0
```

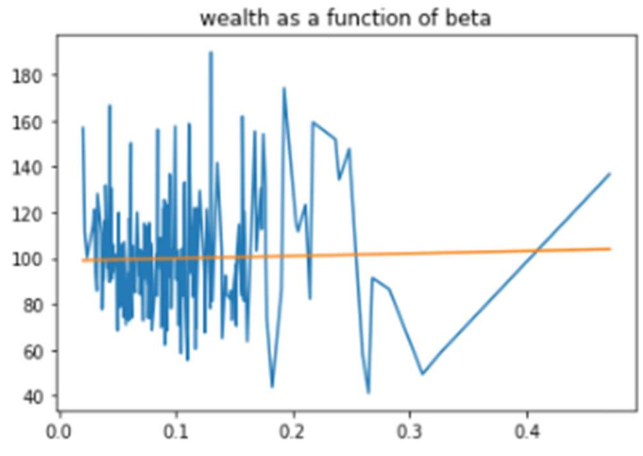


```
------------------------------
ime= 500 number of nodes: 202
```

Below are additional screenshots taken some time later in the run.

```
time= 4500 number of nodes: 206
time= 5000 number of nodes: 207
------------------------------
statistics at time = 5000:
the mean wealth is: 100.000
the median wealth is: 98.378
minimum wealth is:41.05492512149619
maximum wealth is:189.7848191421263
teasury capital is: 0
%-pop <= poverty is: 0.0
```


wealth distribution at t=5000


wealth as a function of beta