1 INTRODUCTION

We will be analysing the impacts of a waste-tax. to model this, we will be using a technique I had come across at UChicago: Agent Based Models (ABMs) and their application to modelling policy impacts. The ABM works by simulating hundreds of agents, with varying resources and preferences (representative of the US economy), and seeing how this simulation economy evolves. We will see that this has implications for waste rates and more importantly may disproportionally affect certain households, which will be our focus.

Hypothesis: since waste is taxed solely based on waste produced, and waste production is range-bound to a certain extent (diminishing marginal waste), lower-income households may be hit harder. This may unintentionally have a regressive effect, though we may observe the intended waste reduction.

2 THE MODEL

In the ABM[1], we create 100 agents, each present in communities of 4–7 agents (based on Dunbar's number for social relations and netowrk built on a Watts-Strogatz structure). Agents have incomes sampled from a distribution based on U.S. Census data, and produce waste in each period based on income and a base waste production level. Every period, they have a choice to recycle (reduce waste) or produce waste at original levels, and this choice is determined by social factors and utility maximization.

2.1 KEY ASSUMPTIONS

1. Agents seek to maximize utility and recycling creates a fixed percentage utility cost. A tax on waste also comes at a percentage utility cost, which can be avoided by reducing waste.
2. Social pressure: agents respond to community changes in recycling, influenced by peer interactions (e.g. a peer who believes in sustainability/recycling). The peers of an agent change over time (create and break ties) every 10 periods.
3. The initial economy is based on present figures (propensity to recycle, income distribution), and we can sample these from known distribution data, using a log-normal distribution

The tax will be progressive on the waste produced, as at higher waste levels, an additional unit of waste will have greater externalities than at the first unit of waste.

WASTE UNITS	0 to 2	2 to 5	>5
WASTE TAX	0.5%	1%	2%

An agent's decision to recycle depends on peer behavior (social pressure), the cost of recycling, and tax incentives. A sigmoid function introduces realistic decision making (combination of randomness and external factors).

Algorithm 1 Agent's decision to recycle

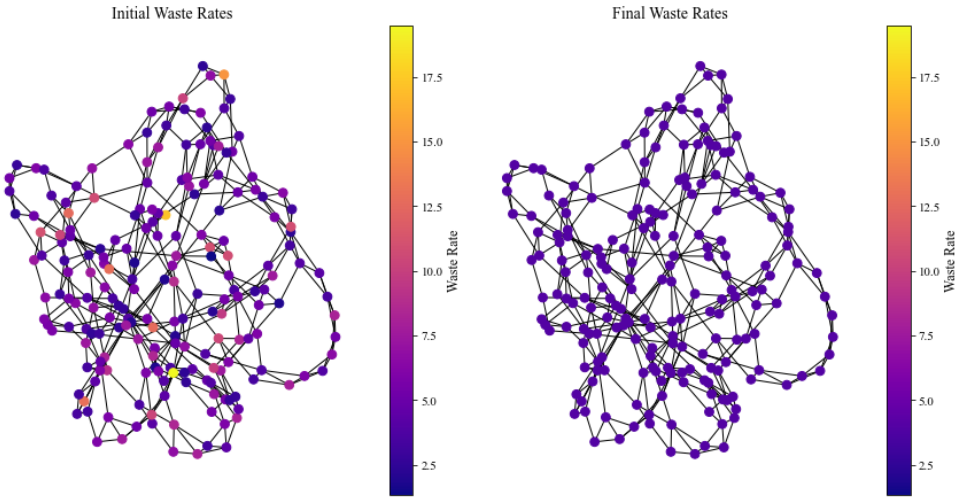
function TORECYCLE(self, neighbours, taxSavings, costOfRecycling)
    utility = self.attitude * neighbours + (benefit / self.income) - costOfRecycling
    p = 1.0 / (1.0 + $e^{(-10*(\text{utility}-0.5))}$) ▷ sigmoid activation function for utility
    self.recycling = random.random() < p
end function

This helps us model how agents make recycling decisions based on how much they can save, the cost (effort) of recycling, and if they are close to someone who believes in recycling. The economy runs for 150 periods, and we compare initial and final waste rates, Gini coefficients, and waste distributions.

[1]Implemented in Python using NumPy, SciPy, NetworkX, and Matplotlib. Agent-based simulation run with dynamic social networks, see code in Appendix

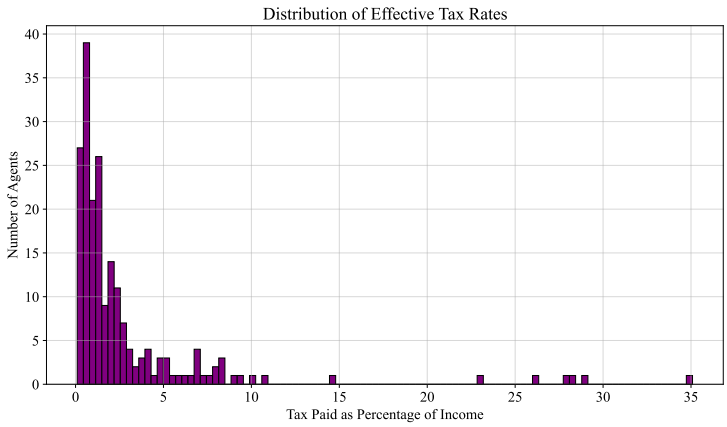
3 RESULTS

Below, we see a representation of the network of agents. Note that this is a stochastic process; results vary each time the simulation is run, and the particular run shown below shows the impact clearly. The waste tax is effective in that it has significantly reduced the average tendency to waste in the economy, reducing the average from 6.1 units to 4.3 units, suggesting positive behavioural changes in response to the tax.



We observe that the tax disproportionally affects certain individuals (seen in the right skew below), who pay a significant percentage of tax, even though tax production is largely range bound and similar for all agents. The initial and final Gini coefficient (post tax) are 0.38 and 0.44 respectively, suggesting an increase in income inequality.

A compression of the waste rate distribution is also observed (initial and final distributions shown in appendix), with extreme waste producing individuals scaling back waste production.



4 IMPLICATIONS AND QUALITATIVE DISCUSSION

Interestingly, in the long term, the ABM seems to "imitate" others' behaviours regarding recycling, which represents a behavioural and cultural change to the perception of recycling (emergent social behaviour), and this behaviour can be exploited using behavioural approaches instead of market based taxes.

An equity consideration to improve the ABM is the responsiveness of agents to adapt to the new waste tax: lower-income agents may have trouble finding time or buying new technology to accommodate for recycling the new tax, and may take longer to adapt, as compared to higher-income households. We may also incorporate waste externalities that are shared within a community (e.g. neighbourhood waste affects individuals). We must consider the Tragedy of the Commons problem where shared resources (e.g. environment or landfills in this case) are overused when left to private individuals, and lead to the collapse of a community, and this impact may be argued to be more significant than that of the income inequality observed.

Overall, the policy is a piecewise Pigouvian tax, effective at reducing negative externalities by internalizing and accounting for the externality; however, ideal tax rates are difficult to precisely calculate and are regressive, as seen in the ABM. Ultimately, an agent's decision to recycle comes down to the marginal cost

of recycling versus the marginal benefit from avoiding taxation (seen in the ABM); this represents a precise tax threshold at which this switch to recycle occurs.

5 BIBLIOGRAPHY

1. Axtell, Robert L., and J. Doyne Farmer. 'Agent-Based Modeling in Economics and Finance: Past, Present, and Future'. Journal of Economic Literature, vol. 63, no. 1, Mar. 2025, pp. 197–287, https://doi.org/10.1257/jel.202213199.

2. 'Dunbar's number' deconstructed.. https://royalsocietypublishing.org/doi/10.1098/rsbl.2021.0158

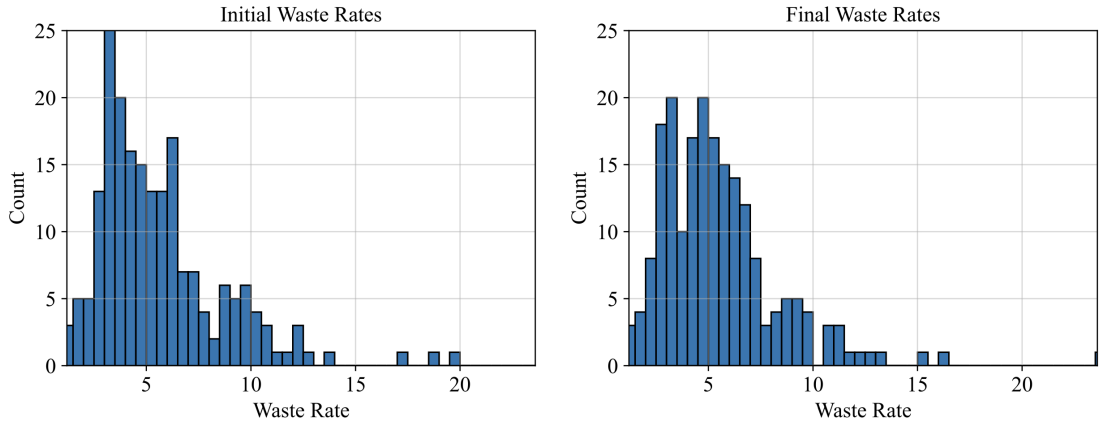
3. Hoeven, E., Kwakkel, J., Hess, V., Pike, T., Wang, B., rht, & Kazil, J. (2025). Mesa 3: Agent-based modeling with Python in 2025. Journal of Open Source Software, 10(107), 7668.

4. US Census Bureau. "Percentage Distribution of Household Income in The United States in 2023." Statista, Statista Inc., 16 Sep 2024, https://www.statista.com/statistics/203183/percentage-distribution-of-household-income-in-the-us/

6 Appendix & Code Sample

```python
mport random
import numpy as np
import scipy.stats as st
import networkx as nx
import matplotlib.pyplot as plt

income_brackets = [(0, 14999), (15000, 24999), (25000, 34999), (35000, 49999), (50000, 74999)
                   , (75000, 99999), (100000, 149999), (150000, 199999), (200000, 350000)]
bracket_probs = np.array([7.4, 6.7, 6.9, 10.3, 18.7, 12.1, 17.0, 9.5, 11.4]) / 100
num_agents = 100

def gen_income():
    i = np.random.choice(len(income_brackets), p=bracket_probs)
    low, high = income_brackets[i]
    return np.random.uniform(low, high)

class agent:
    def __init__(self, node_id, waste_rate,
                 attitude=None, income=None):
        self.id = node_id
        self.waste_rate = waste_rate
        self.attitude = attitude
        self.income = income
        self.recycling = False
        self.total_waste = 0.0
        self.total_tax = 0.0

    def generate_waste(self):
        base = max(0.0, random.gauss(self.waste_rate, 0.5))
        amount = base * (0.6 if self.recycling else 1.0) + base * self.income/350000
        self.total_waste += amount
        return amount

    def pay_tax(self, amount, brackets):
        tax = 0.0
        prev_limit = 0.0
        remaining = amount
        for limit, rate in brackets:
            width = limit - prev_limit
            taxed = min(remaining, width)
            tax += taxed * rate
            remaining -= taxed
            prev_limit = limit
            if remaining <= 0:
                break
        self.total_tax += tax
        return tax
```

```python
    def recycle(self, neighbors, tax_savings_factor=1.0, cost_of_recycling=0.05):
        peer_frac = np.mean([n.recycling for n in neighbors]) if neighbors else 0.0
        benefit = self.waste_rate * tax_savings_factor * (1 - 0.6)
        # normalized utility
        utility = self.attitude * peer_frac + (benefit / self.income) - cost_of_recycling
        p = 1.0 / (1.0 + np.exp(-10 * (utility - 0.5))) # sigmoid activation function for utility
        self.recycling = random.random() < p

        G = nx.watts_strogatz_graph(num_agents, neighbors, p)
        agents = {}
        shape, loc, scale = 0.5, 0, 5.0
        for n in G.nodes:
            rate = st.lognorm.rvs(shape, loc=loc, scale=scale)
            attitude =  np.random.binomial(1, 0.4)
            income = gen_income()
            # rate = (income / 50000) * 5.0
            agents[n] = agent(n, waste_rate=rate, attitude=attitude, income=income)
        return G, agents


def simulate(num_agents=100, steps=100, controller=None, dynamic_network=False):
    G, agents = build_network(num_agents, dynamic=dynamic_network)
    base_brackets = [(2, 0.5), (5, 1.0), (np.inf, 2.0)]
    tax_saving_factor = 10.0

    history = {'avg_waste': [], 'avg_tax': [], 'recycle_rate': []}

    for t in range(steps):
        w_sum = tax_sum = 0.0
        for i, agent in agents.items():
            w = agent.generate_waste()
            tax = agent.pay_tax(w, base_brackets)
            w_sum += w
            tax_sum += tax
        avg_w = w_sum / num_agents
        history['avg_waste'].append(avg_w)
        history['avg_tax'].append(tax_sum / num_agents)
        history['recycle_rate'].append(np.mean([a.recycling for a in agents.values()]))

        if dynamic_network and t % 10 == 0: # every 10
            to_be_removed = random.sample(list(G.edges), int(0.01*G.number_of_edges()))
            for u, v in to_be_removed:
                G.remove_edge(u, v)
                w = random.choice(list(G.nodes)) # choose another node
                G.add_edge(u, w)

        for i, agent in agents.items():
            neigh = [agents[n] for n in G.neighbors(i)]
            agent.recycle(neigh, tax_saving_factor)

    return agents, history, G

agents, history, G = simulate(num_agents, steps=150, dynamic_network=True)
```


Initial Waste Rates
Count
Waste Rate
0
5
10
15
20
25
5
10
15
20

Final Waste Rates
Count
Waste Rate
0
5
10
15
20
25
5
10
15
20