Assignment #3

MACS 30000, Dr. Evans

Sanittawan Tan (sanittawan)

Note: I still have errors converting Jupyter notebook to pdf. Some parts of part 1 will be cut off. Please refer to the pdf file of Part 1 instead. Thank you for your understanding and apologies for the inconvenience.

Part 1: A Response to Moretti's Simulation in Sociology paper

Please see attached pdf file

In her article *Computer Simulation in Sociology: What Contribution?*, Moretti surveyed four main techniques of simulation that have been used in sociology, namely system dynamics, multiagent systems, cellular automata, and genetic algorithms, and discuss their contribution to sociology. Although computer simulation is a promising method that can be beneficial for many fields in the social sciences, some weaknesses still remain. This paper discusses some of the potential weaknesses of certain methods in simulation, gives an example of the application of dynamic feedback in Sociology and explores future research questions in Political Science.

As Moretti nicely summarized, the main weakness of a computer simulation regardless of techniques is that the validity of a simulation largely depends on two elements: the validity of the theory that the model is based on and the validity of the computational tools used for such simulation. In other words, if the theory that supports a simulation does not accurately capture the reality, the simulation will not be valid. Take multiagent systems technique as a case in point. Simulations based on multiagent systems are very useful for studying how autonomous agents in a system interact and causes a structure to emerge over time. However, its potential weakness is that such simulation must include "a detailed list of rules of agents' behavior, that is, protocols of communications and decision-making procedures that consider environmental changes and all interactions with other agents in that systems." The model relies on researchers' understanding of these rules and, additionally, accurately model them into the simulation. Moreover, researchers also have to make assumptions regarding an agent in the system; for instance, some models begin from the assumption that an agent is a rational actor. Although this is a fairly standard assumption across many theories in the social sciences and can easily be modeled into a simulation, one may question if an agent is indeed *always* a rational actor.

Cellular automata technique which is a particular kind of multiagent system may also suffer from similar problems. This is because cellular automata assume that agents in a system have "a specific and determined position in a lattice" and behave and interact homogeneously. What if agents do not act homogeneously? How do we deal with outliers? It seems to me that its strength, which is simplicity, is also its weakness. As Moretti pointed out, cellular automata assume "synchronous updating" as in biology, but individual agents in a society may have a different timeline in "updating" such as developing opinions. Another limitation of cellular automata that the author discussed is the difficulty in defining neighborhood or boundary that individuals in a society interact. This stems from the fact the model establishes that individuals interact solely with a subset of a population. However, in reality, that subset is hard to define taken into account the advent of the internet and technology and that particular subset also changes over time.

Another problem of simulation validity is that it is difficult to set criteria and measurement for validity. It seems that social scientists who wish to use simulation still have to develop standardized tools or methods for evaluating if a study is valid. Although potential weaknesses discussed so far remain challenges for future research, some models such as system dynamics can still be useful for explaining and discovering relationships in a complex system, especially how an action causes a change in the system which, in turn, causes another change.

Developed by Jay Forrester of MIT Sloan School of Management in the 1950s, system dynamics is a model for studying complex systems taken into account agents' actions. System dynamics has wide application in Sociology. For example, the model translates conflict theory, which posits that society functions in a way that individuals and their respective groups maximize their benefits which lead to conflicts among groups and eventually social change, propagated by Marx and Weber into a practical simulation. Researchers may use a toolkit such as SWARM developed by the Santa Fe Institute to model how interests groups compete in the policy-making process. An example of sociological models concerning dynamic feedback that Moretti cited is

the work of Doran et al. (1994), *The EOS project: Modelling Upper Paleolithic social change*, which sought to understand circumstances that contribute to the formation of hierarchies in a society. Without simulation, it is nearly impossible to recreate the circumstances of the Upper Paleolithic era.

A potential application of system dynamics in Political Science, especially the feedback loops, leads to many interesting questions. For instance, in International Relations subfield, one may study how the norm of Responsibility to Protect (R2P) emerged from the struggle between civil societies and states and how R2P shapes and changes state behaviors and the concept of sovereignty. The acceptance of R2P, which maintains that every nation has the responsibility to protect all of their citizens regardless of political, racial and religious affiliations to prevent genocide, ethnic cleansing and other human rights violations, at the United Nations is a major milestone because it allows the United Nations and other countries to intervene to stop crimes against humanities in a country. Since R2P clearly weakens the concept of sovereignty, we may be able to use system dynamics to simulate how this change affects behaviors of all states and the international system and if the norm leads to a formation of a group of states that are working on undermining the norm. Another interesting research question could be the study of how increasing the number of immigrant workers or accepting more refugees into a country affects demography and politics. For instance, countries like Japan and Singapore which are facing population problems because there are increasingly fewer young workers may use simulation to understand how increasing the number of immigrant workers can affect the country's demography and eventually politics in the long run. In the short run, one may think that importing workers can solve labor supply shortages in key industries. However, there are potential long-run effects that result from this change in population. Will more immigrant workers who are racially and culturally different from its citizens give rise to a right-wing nationalist party because the country's citizens feel threatened by the potential loss of homogeneity? Or will increased immigrants indeed economically benefit the country by raising productivity level and quality of life and contribute to more multiculturalism? Simulated data and simulation are clearly useful in both cases because policymakers can look at predicted results without implementing the policy if researchers model states and citizens behaviors accurately. Moreover, there are dynamic feedbacks as one change leads to another change in both cases.

Despite its weaknesses, computer simulation can be applied to various study in the social sciences. Its main strength is that it enables social scientists to simulate the real world without making an impact. In other words, computer simulation makes possible studies that are impossible to do by observational studies and experiments. However, researchers have to pay attention to underlying assumptions and limitations of simulation before generalizing the results.

Part 2: Silmulating your income

In [1]:
```
# Import initial packages
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator
```

part (a)

In [2]:
```
p = {
    'w0': 80000,
    'gr': 0.025,
    'st_year': 2020,
    'lf_years': 40,
    'num_draws': 10000,
    'rho': 0.4,
    'mean': 0,
    'sd': 0.13,
    'size': (40,10000)
}
```

In [3]:
```
np.random.seed(524)
errors = np.random.normal(p['mean'], p['sd'], p['size'])
errors
```

Out[3]:
```
array([[-0.1861917 ,  0.20573424,  0.24235592, ...,  0.21026495,
        -0.09975796, -0.15212711],
       [ 0.05973327, -0.2689247 , -0.05653301, ..., -0.25798775,
        -0.04607823,  0.05277632],
       [-0.08396151, -0.15168482,  0.08169169, ..., -0.13978135,
         0.11631367, -0.02473229],
       ...,
       [ 0.3148506 ,  0.0311664 , -0.12613043, ..., -0.22229771,
         0.20639003,  0.00992186],
       [ 0.00219802, -0.04190169,  0.00128297, ...,  0.06673485,
        -0.06246006, -0.01250838],
       [-0.10437421, -0.2318872 , -0.17819712, ...,  0.18351762,
         0.10146635,  0.05648267]])
```

In [4]:
```
errors.shape
```

Out[4]:
```
(40, 10000)
```

In [5]:
```
ln_income_matrix = np.zeros((p['lf_years'], p['num_draws']))
ln_income_matrix
```

Out[5]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [6]:
```
ln_income_matrix.shape
```

Out[6]:
```
(40, 10000)
```

In [7]:
```
ln_income_matrix[0, :] = np.log(p['w0']) + errors[0, :]
ln_income_matrix
```

Out[7]:
```
array([[11.10359022, 11.49551615, 11.53213783, ..., 11.50004686,
        11.19002396, 11.1376548 ],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       ...,
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ]])
```

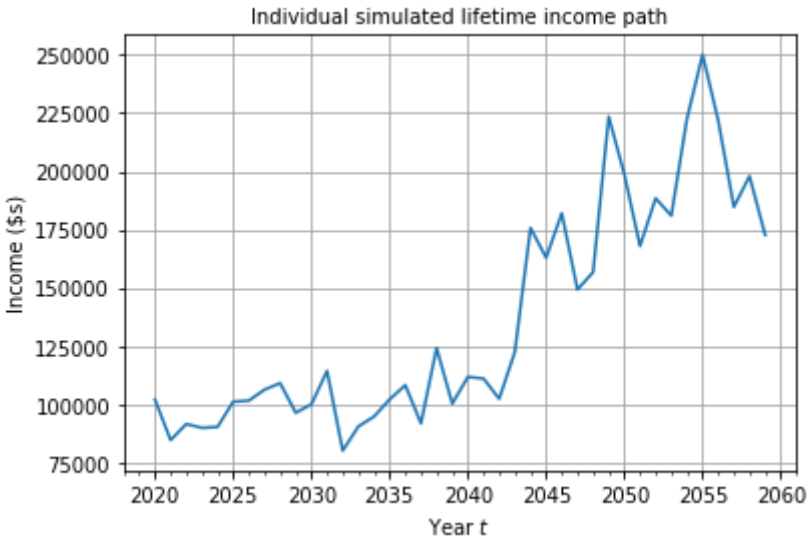
In [8]:
```
for yr in range(1, p['lf_years']):
    ln_income_matrix[yr, :] = (1 - p['rho'])*(np.log(p['w0']) + p['gr']*
yr) + \
         p['rho']*(ln_income_matrix[yr-1, :]) + errors[yr, :]
ln_income_matrix = np.exp(ln_income_matrix) #dealing with large numbers
 so put in terms of 10k's
```

In [9]:
```
ln_income_matrix
```

Out[9]:
```
array([[ 66409.15585396,  98274.13534194, 101939.81109509, ...,
         98720.39690442,  72404.51636886,  68710.32820307],
       [ 80020.53020329,  67383.19350738,  84557.85626308, ...,
         68247.7770509 ,  74518.33613244,  80555.96068584],
       [ 75805.26636606,  66134.42494243,  91458.20304692, ...,
         67268.53350159,  90012.42673528,  80645.62355527],
       ...,
       [272690.56519108, 217821.73027242, 184724.24512469, ...,
        159922.45424852, 253961.68337673, 209741.55004062],
       [231539.17420799, 202509.15149494, 197955.96626493, ...,
        199502.43481758, 210951.71828579, 205420.27946389],
       [197895.95201384, 165115.10025278, 172644.86927513, ...,
        248654.44847819, 234237.14656466, 221566.29879732]])
```

In [10]:
```
# Plotting one income path
%matplotlib inline
year_vec = np.arange(p['st_year'], p['st_year'] + p['lf_years'])
individual = 2
fig, ax = plt.subplots()
plt.plot(year_vec, ln_income_matrix[:, individual])
minorLocator = MultipleLocator(1)
ax.xaxis.set_minor_locator(minorLocator)
plt.grid(b=True, which='major', color='0.65', linestyle='-')
plt.title('Individual simulated lifetime income path', fontsize=10)
plt.xlabel(r'Year $t$')
plt.ylabel(r'Income (\$s)')
```

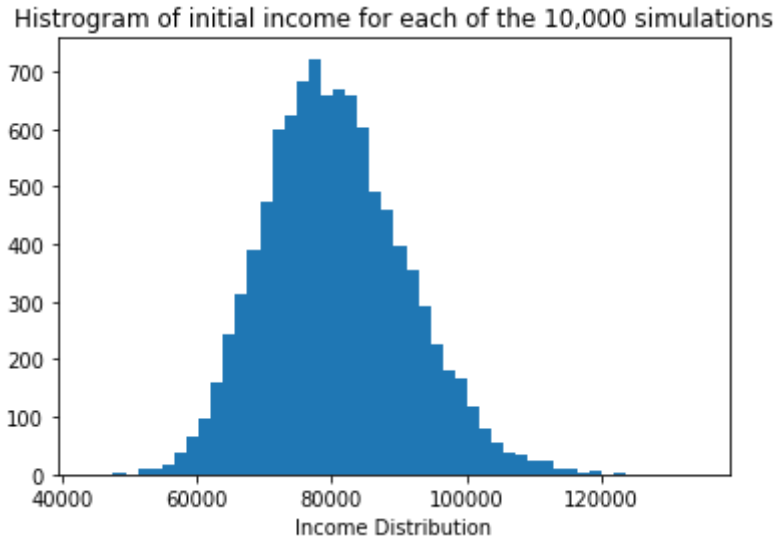
Out[10]:
```
Text(0, 0.5, 'Income (\\$s)')
```



Part (b): Plot a histogram and calculate percentages of the class

In [11]:
```
plt.hist(ln_income_matrix[0,:], bins=50)
plt.xlabel("Income Distribution")
plt.title("Histrogram of initial income for each of the 10,000 simulatio
ns")
```

Out[11]:
```
Text(0.5, 1.0, 'Histrogram of initial income for each of the 10,000 sim
ulations')
```



In [12]:
```
np.mean(ln_income_matrix)
```

Out[12]:
```
134881.70610587983
```

In [13]:
```
np.median(ln_income_matrix)
```

Out[13]:
```
128097.2995056228
```

Because we found that mean is greater than median, this distribution is right skewed.

Percentage of class who will earn more than 100,000 dollars

In [14]:
```
len(ln_income_matrix[0, :] [ln_income_matrix[0, :] > 100000]) / len(ln_i
ncome_matrix[0, :])
```

Out[14]:
```
0.0417
```

The percentage of class who will earn more than 100,000 dollars is 4.17%.

Percentage of class who will earn less than 70,000 dollars

In [15]:
```
len(ln_income_matrix[0, :] [ln_income_matrix[0, :] < 70000]) / len(ln_in
come_matrix[0, :])
```

Out[15]:
```
0.1512
```

The percentage of class who will less than 70,000 dollars is 15.12%.

Part (C): Simulate debt payoff with the starting salary of 80,000 dollars

In [16]:
```
# write a function to calculate the number of years taken to payoff the
 debt
def year_payoff_debt(principle, salary_list):
    year = 0
    for j in salary_list:
        if principle > 0:
            new_principle = principle - (0.1) * (j)
            principle = new_principle
            year += 1
        else:
            break
    return year
```

In [17]:
```
# Check income matrix of the first set of simulated income
ln_income_matrix[:, 0]
```

Out[17]:
```
array([ 66409.15585396,  80020.53020329,  75805.26636606,  88075.026533
66,
       106861.63415772, 109595.32218589,  91254.89677608, 109756.188862
87,
        90737.58945859,  86940.93993646,  92663.92019855,  98052.981488
84,
        81286.30857991,  74947.88506476,  88636.79143603,  81812.314390
37,
       122584.4464533 , 141450.04880919, 165005.91290055, 153472.719534
7 ,
       144853.8589323 , 132509.14721066, 126880.53955683, 116396.880002
57,
       125635.36389956, 140207.1445515 , 210435.37935251, 189343.852893
87,
       182646.86442959, 216001.57150949, 212982.60688553, 209937.619806
41,
       155059.70836031, 148085.1913182 , 168596.47052775, 215862.030925
73,
       195034.75599749, 272690.56519108, 231539.17420799, 197895.952013
84])
```

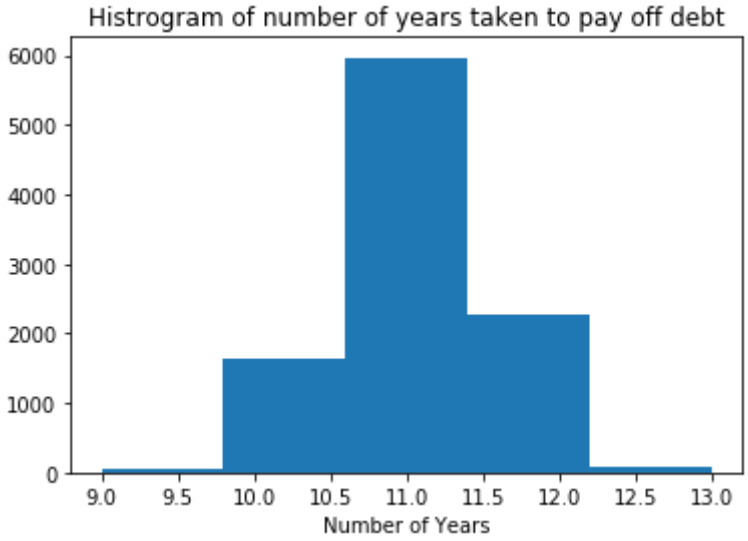
In [18]:
```
all_year = []
for i in range(0, p['num_draws']):
    year = year_payoff_debt(95000,ln_income_matrix[:, i])
    all_year.append(year)
```

In [20]:
```
print("minimum number of years is", min(all_year))
print("maximum number of years is", max(all_year))
```

```
minimum number of years is 9
maximum number of years is 13
```

In [21]:
```
plt.hist(all_year, bins=5)
plt.xlabel("Number of Years")
plt.title("Histrogram of number of years taken to pay off debt")
```

Out[21]:
```
Text(0.5, 1.0, 'Histrogram of number of years taken to pay off debt')
```



In [22]:
```
payoff_ten_years = [x for x in all_year if x <= 10]
```

In [23]:
```
# Calculate the percentage of the simulations
len(payoff_ten_years) / len(all_year)
```

Out[23]:
```
0.1678
```

In 16.78 percent of the simulations, I will be able to pay off $ 95,000 debt in ten years.

Part (d): Simulate debt payoff with the starting salary of 90,000 dollars

In [24]:
```
new_p = {
    'w0': 90000,
    'gr': 0.025,
    'st_year': 2020,
    'lf_years': 40,
    'num_draws': 10000,
    'rho': 0.4,
    'mean': 0,
    'sd': 0.17,
    'size': (40,10000)
}
```

In [25]:
```
np.random.seed(524)
new_errors = np.random.normal(new_p['mean'], new_p['sd'], new_p['size'])
```

In [26]:
```
new_ln_income_matrix = np.zeros((new_p['lf_years'], new_p['num_draws']))
new_ln_income_matrix[0, :] = np.log(new_p['w0']) + new_errors[0, :]
```

In [27]:
```
for yr in range(1, new_p['lf_years']):
    new_ln_income_matrix[yr, :] = (1 - p['rho'])*(np.log(new_p['w0']) +
new_p['gr']*yr) + \
         p['rho']*(new_ln_income_matrix[yr-1, :]) + new_errors[yr, :]
new_income_matrix = np.exp(new_ln_income_matrix)
new_income_matrix
```

Out[27]:
```
array([[ 70550.46142451, 117783.33011091, 123561.20729139, ...,
        118483.24080508,  78992.81966812,  73764.25171169],
       [ 89615.63768821,  71575.56495871,  96317.75493523, ...,
         72778.88084775,  81644.3347736 ,  90400.57899801],
       [ 82955.30101689,  69396.06916251, 106035.55593099, ...,
         70956.3661129 , 103848.93176006,  89949.09077038],
       ...,
       [338309.11761165, 252187.52025149, 203293.03644369, ...,
        168361.21927259, 308250.29858492, 240024.49205936],
       [271061.07048342, 227502.32436192, 220836.5697397 , ...,
        223095.32811759, 239983.96514044, 231788.44418303],
       [219057.46748997, 172865.33333479, 183245.71710131, ...,
        295275.8618388 , 273090.00167035, 253934.86273481]])
```

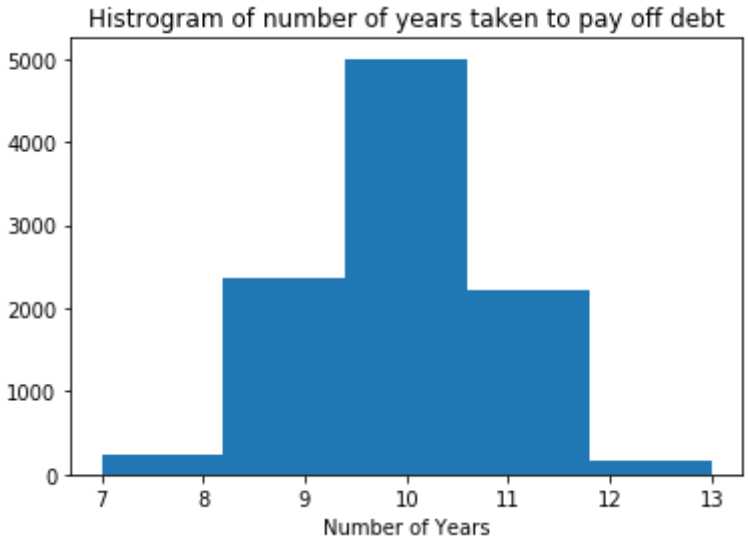
In [28]:
```
new_all_year = []
for i in range(0, p['num_draws']):
    year = year_payoff_debt(95000,new_income_matrix[:, i])
    new_all_year.append(year)
```

In [29]:
```
print("minimum number of years is", min(new_all_year))
print("maximum number of years is", max(new_all_year))
```

```
minimum number of years is 7
maximum number of years is 13
```

In [30]:
```
plt.hist(new_all_year, bins=5)
plt.xlabel("Number of Years")
plt.title("Histrogram of number of years taken to pay off debt")
```

Out[30]: Text(0.5, 1.0, 'Histrogram of number of years taken to pay off debt')



In [31]:
```
new_payoff_ten_years = [x for x in new_all_year if x <= 10]
```

In [32]:
```
len(new_payoff_ten_years) / len(new_all_year)
```

Out[32]: 0.7602

In 76.02 percent of the simulations, I will be able to pay off the debt in 10 years.