PSTAT197A_Project_Part_2

November 25, 2020

1 PSTAT 197A: Project Part 2

Peter Bayerle, Samantha Solomon, & Sophia Sternberg

2 (a)

[1]:
```
import numpy as np
import numpy as np
import scipy
from scipy.integrate import solve_ivp
from scipy.optimize import fmin
from scipy.cluster import hierarchy
from sklearn.metrics import mean_squared_error
import sklearn.cluster
import matplotlib.pyplot as plt
import pandas as pd
```

[2]:
```
file = np.load("part2.npz")
beta_old = file['beta_old']
N = file['N']
Svc_0_PMF = file['Svc_0_PMF']
Lc = file['Lc']
Ic_0 =  file['Ic_0']
gamma = file['gamma']
L_observed = file['L_observed']
nb_nodes = Svc_0_PMF.shape[0]
```

[3]:
```
def SIR(t,Z, N, betas, gamma):
    (S_vec, I_vec, R) = (Z[:16].reshape(4, 4), Z[16:20], Z[-1])
    I = I_vec.sum()
    dSdt = -betas*S_vec*I/N
    dIdt = -dSdt.sum(axis=0)-gamma*I_vec
    dRdt = np.array([gamma*I])
    return np.concatenate((dSdt.flatten(), dIdt, dRdt))
```

[4]:
```
Svc_0_PMF /= Svc_0_PMF.sum(axis=(1, 2), keepdims=True)
Svc_0 = (N-Ic_0.sum(axis=1))[:, np.newaxis, np.newaxis]*Svc_0_PMF
Svc_0 = Svc_0.round().astype(int)
R0 = np.zeros((nb_nodes)).astype(int)

t = np.linspace(0, 199, 200)

S = np.zeros((nb_nodes, 4, 4, len(t)))
I = np.zeros((nb_nodes, 4, len(t)))
R = np.zeros((nb_nodes, len(t)))
L = np.zeros((nb_nodes, len(t)))

for N_node, node_index in zip(N, range(nb_nodes)):

    initial_conditions = np.concatenate(
        (Svc_0[node_index].flatten(), Ic_0[node_index], [R0[node_index]]))

    soln = solve_ivp(SIR, (t[0], t[9]), initial_conditions, args=(N_node,␣
 ↪beta_old, gamma), t_eval=t[:10])

    S[node_index, :, :, :10] = soln.y[:16].reshape(4, 4, -1)
    I[node_index, :, :10] = soln.y[16:20]
    R[node_index, :10] = soln.y[20]
    L[node_index, :10] = Lc@soln.y[16:20]
```

[5]:
```
# initial conditions (start @ day 9)
Svc_9 = S[:,:,:,9]
Ic_9 = I[:,:,9]
R9 = R[:,9]

alpha_predicted = np.zeros((nb_nodes))

def loss_func(alpha, N_node, node_index):
    initial_conditions = np.concatenate(
        (Svc_9[node_index].flatten(), Ic_9[node_index], [R9[node_index]]))

    soln = solve_ivp(
        SIR, (t[9], t[29]), initial_conditions, args=(N_node, alpha*beta_old,␣
 ↪gamma), t_eval=t[10:30])

    S[node_index, :, :, 10:30] = soln.y[:16].reshape(4, 4, -1)
    I[node_index, :, 10:30] = soln.y[16:20]
    R[node_index, 10:30] = soln.y[20]
    L[node_index, 10:30] = Lc@soln.y[16:20]

    mse = np.sum((L_observed[node_index,10:30] - L[node_index,10:30])**2) / 20
    return mse
```

```
for N_node, node_index in zip(N, range(nb_nodes)):
    alpha_predicted[node_index] = fmin(loss_func, 0.5, args=(N_node,␣
 ↪node_index), disp=False)

print('alpha which minimizes error between L_predicted and L_observed:')
print(alpha_predicted)
print(f'max alpha: {max(alpha_predicted)}, min alpha: {min(alpha_predicted)}')
```

```
alpha which minimizes error between L_predicted and L_observed:
[0.23352203 0.3860054  0.22739868 0.70975647 0.67788391 0.20684509
 0.35756836 0.35657654 0.37946777 0.18708496 0.28283691 0.32966309
 0.31420898 0.2162323  0.13437347 0.24160156 0.20031738 0.55296021
 0.58077087 0.18221436 0.26948242 0.30437622 0.48616943 0.30966797
 0.33355408 0.16851807 0.36960449 0.21408691 0.47091064 0.66050568
 0.29938965 0.30321045 0.38393707 0.17612305 0.59567566 0.63848572
 0.39033203 0.17086182 0.28800049 0.86357422 0.41445923 0.43756104
 0.42476196 0.18603516 0.24894104 0.43441162 0.69438477 0.32382202
 0.24464111 0.21859131 0.14959717 0.2128418  0.32988281 0.26757812
 0.17961426 0.65092468 0.19025879 0.52108459 0.27169952 0.55423279
 0.36207275 0.28662109 0.3498291  0.41787415 0.25991821 0.50830688
 0.20330505 0.41876221 0.45647888 0.16158829 0.16303711 0.33074341
 0.2930542  0.44996338 0.40922546 0.2992981  0.56279297 0.64515076
 0.37071838 0.20419922 0.33313293 0.5023941  0.69835815 0.35844421
 0.28005524 0.30422821 0.27301636 0.53604736 0.95327148 0.50705566
 0.17155762 0.35234375 0.25798035 0.23042603 0.44871826 0.20612793
 0.15996704 0.92822266 0.2512085  0.68381958]
max alpha: 0.9532714843750012, min alpha: 0.13437347412109343
```

3 (b)

[6]:
```
def g(money):
    alpha = 1/np.log2(2*10**(-3)*money+2)
    return alpha

def g_inv(alpha):
    money = (2**(1/alpha)-2)/(2*10**(-3))
    return money

def scatter_plot(alpha_afterNPI):
    plt.figure()
    colors=(beta_old[np.newaxis,:,:]*Svc_0_PMF).mean(axis=(1,2))
    plt.scatter(alpha_predicted, alpha_afterNPI, s=N/500, c=colors, cmap='jet',␣
 ↪alpha=0.5)
    plt.colorbar()
    plt.xlabel(r"$\alpha_{own}$")
```

```
    plt.ylabel(r"$\alpha_{after\ NPI}$")

def total_recovered_difference(delta_alpha):
    alpha_own = alpha_predicted
    budget = []
    for alpha in alpha_own:
      new_alpha = alpha - delta_alpha
      x = g_inv(new_alpha)
      budget.append(x)
    budget = np.array(budget)

    difference = abs(np.sum(budget)-1000000)
    return difference

# simulate days 30-200
def simulate(S, I, R, using_alpha=alpha_predicted):
    alphas = using_alpha
    S, I, R = S.copy(), I.copy(), R.copy()

    # initial conditions (start @ day 29)
    Svc_29 = S[:,:,:,29]
    Ic_29 = I[:,:,29]
    R29 = R[:,29]

    for N_node, node_index in zip(N, range(nb_nodes)):
        a = alphas[node_index]

        initial_conditions = np.concatenate(
            (Svc_29[node_index].flatten(), Ic_29[node_index],␣
 ↪[R29[node_index]]))

        soln = solve_ivp(
            SIR, (t[29], t[199]), initial_conditions, args=(N_node, a*beta_old,␣
 ↪gamma), t_eval=t[30:200])

        S[node_index, :, :, 30:200] = soln.y[:16].reshape(4, 4, -1)
        I[node_index, :, 30:200] = soln.y[16:20]
        R[node_index, 30:200] = soln.y[20]

    return S, I, R

S_noNPI, I_noNPI, R_noNPI = simulate(S, I, R)

pd.set_option('display.float_format', lambda x: '%.5f' % x)

def display(S, I, R, policy_num):
  table = pd.DataFrame([
```

```
    ['S', np.sum(S_noNPI[:,:,:,199]), np.sum(S[:,:,:,199])],
    ['I', np.sum(I_noNPI[:,:,199]), np.sum(I[:,:,199])],
    ['R', np.sum(R_noNPI[:,199]), np.sum(R[:,199])]
  ],
  columns=['Compartment', f'Day 200 pop w/out {policy_num}', f'Day 200 pop w/␣
↪{policy_num}']
 )
 table['Difference'] = table[f'Day 200 pop w/out {policy_num}']-table[f'Day␣
↪200 pop w/ {policy_num}']

 print(f'S, I, R values with and without {policy_num}:\n')
 print(table.to_string(index=False))
```

3.0.1 Policy #1

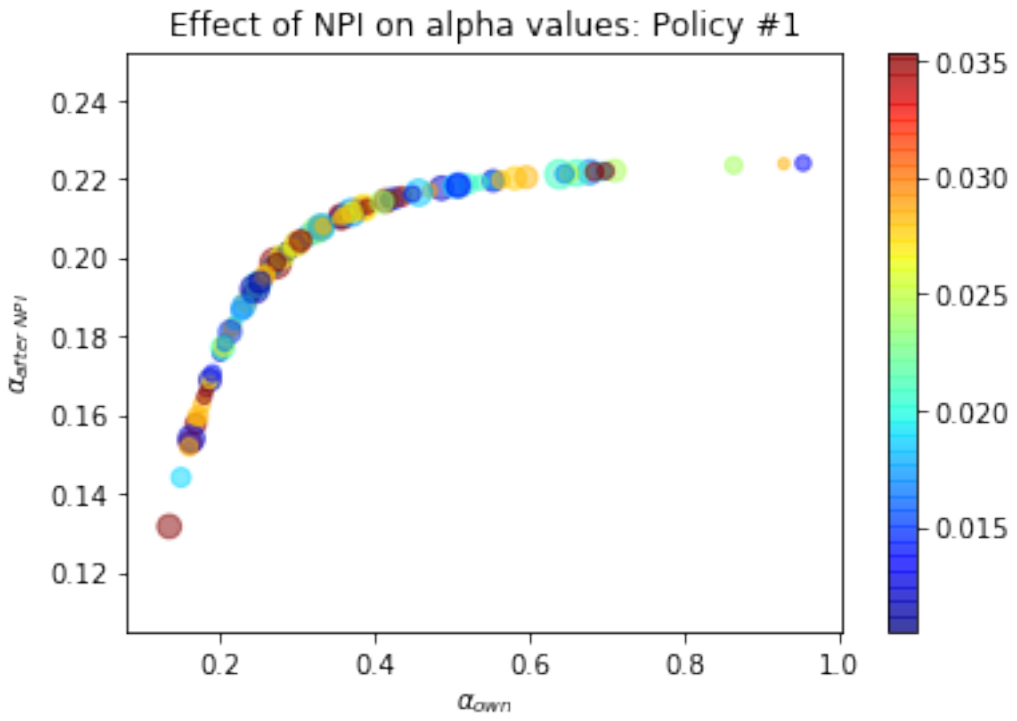
Spend the same amount of money for each node.

1. We see from the plot that nodes with high initial values of α see significant reduction after NPI, but nodes with already small values of α see little change. For example, the node with $\alpha_{own} \approx 1$ reduces to $\alpha_{afterNPI} \approx 0.22$, but nodes with $\alpha_{own} \approx 0.2$ shrink to just $\alpha_{afterNPI} \approx 0.17$.

[7]:
```
npi_budget = 1_000_000

amtPerNode = npi_budget / len(alpha_predicted)
alpha_afterNPI1 = [g(g_inv(alpha) + amtPerNode) for alpha in alpha_predicted]

scatter_plot(alpha_afterNPI1)
plt.title('Effect of NPI on alpha values: Policy #1')
plt.show()
```



2. Change in the number of susceptible, recovered, and infected people in total after 200 days with and without NPI:

- With NPI, at day 200, there are 322,316 more people in the S compartment.
- With NPI, at day 200, there are 1,876 fewer people in the I compartment.
- With NPI, at day 200, there are 320,439 fewer people in the R compartment.

[8]:
```
S_NPI1, I_NPI1, R_NPI1 = simulate(S, I, R, using_alpha=alpha_afterNPI1)
display(S_NPI1, I_NPI1, R_NPI1, 'policy 1')
```

```
S, I, R values with and without policy 1:

Compartment  Day 200 pop w/out policy 1  Day 200 pop w/ policy 1    Difference
          S                1062735.03098             1385051.20997 -322316.17899
          I                   2374.61864                498.05000    1876.56864
          R                1649157.35037             1328717.74003  320439.61035
```

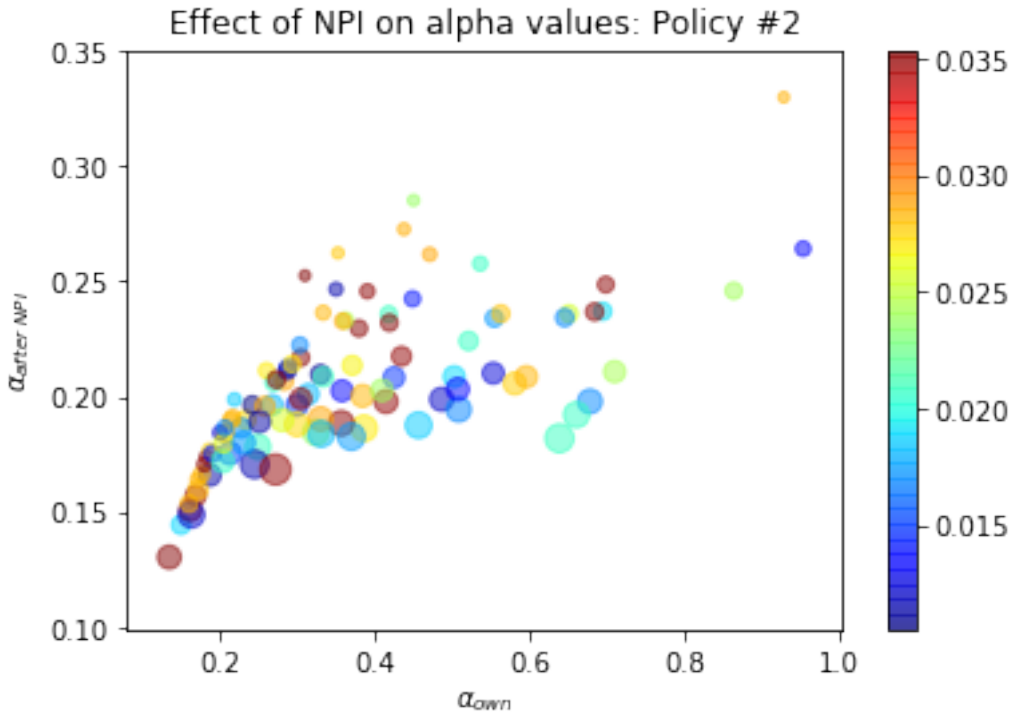
3.0.2 Policy #2

Spend the budget proportional to each node's population.

1. Here, we observe that nodes with larger populations have the largest change in α. The different colored markers are not showing a trend/ pattern. Since the allocation is only based on population size, nodes with more vulnerable populations do not see a guaranteed reduction in α.

[9]:
```
total_pop = sum(N)
budget_2 = [(i/total_pop)*npi_budget for i in N]

alpha_afterNPI2 = [g(g_inv(alpha) + allocation) for allocation, alpha in␣
 ↪zip(budget_2, alpha_predicted)]
scatter_plot(alpha_afterNPI2)
plt.title('Effect of NPI on alpha values: Policy #2')
plt.show()
```



2. Change in the number of susceptible, recovered, and infected people in total after 200 days with and without NPI:

- With NPI, at day 200, there are 337,985 more people in the S compartment.
- With NPI, at day 200, there are 1,991 fewer people in the I compartment.
- With NPI, at day 200, there are 335,993 fewer people in the R compartment.

[10]:
```
S_NPI2, I_NPI2, R_NPI2 = simulate(S, I, R, using_alpha=alpha_afterNPI2)
display(S_NPI2, I_NPI2, R_NPI2, 'Policy 2')
```

```
S, I, R values with and without Policy 2:

Compartment  Day 200 pop w/out Policy 2  Day 200 pop w/ Policy 2    Difference
          S                1062735.03098            1400720.45994 -337985.42896
          I                   2374.61864                382.96548    1991.65317
          R                1649157.35037            1313163.57458  335993.77579
```

3.0.3 Policy #3

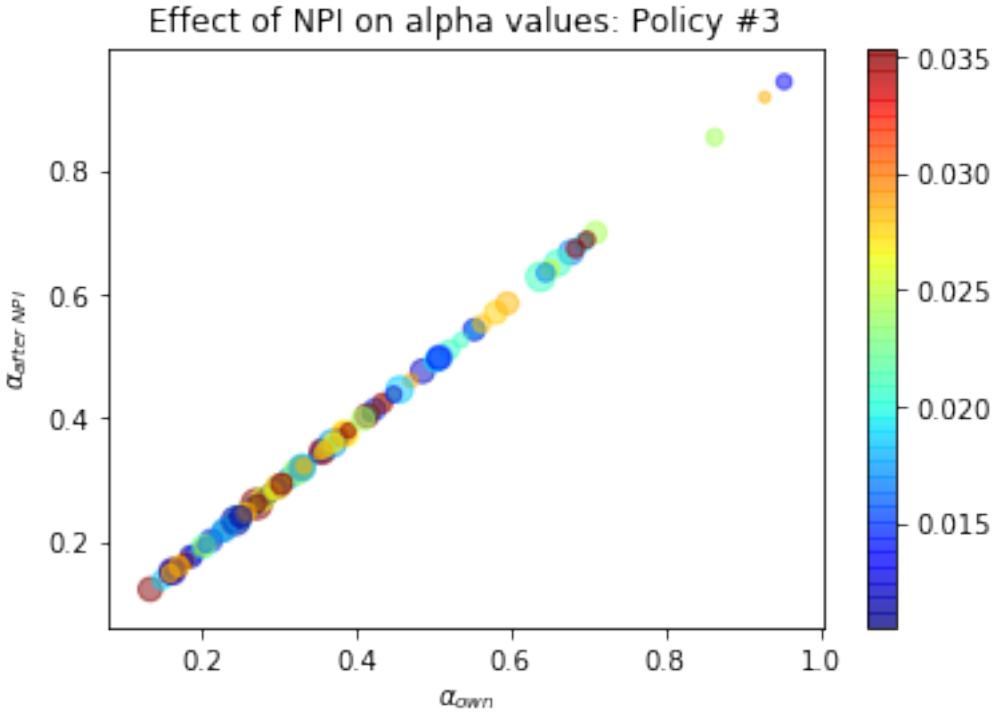
Spend the budget such that $\alpha_{own} - \alpha_{afterNPI}$ is the same for all nodes,that is the change in α induced by the NPI is the same for all nodes.

1. We can see that the change made to the α_{own} is quite small. Because the change in α is the same across all nodes, there is a linear relationship between α_{own} and $\alpha_{afterNPI}$. The original α_{own} values are shifted up by $\Delta\alpha$.

[11]:
```
# starting value for delta_alpha is 0.05
optimal_d_alpha = fmin(total_recovered_difference,.05,disp=False)
alpha_afterNPI3 = [alpha - optimal_d_alpha for alpha in alpha_predicted]
budgets3 = [g_inv(alpha) for alpha in alpha_afterNPI3]
print(f'sum of all budgets allocated for each node is {sum(budgets3)[0]}') #␣
 ↪this checks out, we expect it to be 1mil
```

```
sum of all budgets allocated for each node is 1000000.000076468
```

[12]:
```
scatter_plot(alpha_afterNPI3)
plt.title('Effect of NPI on alpha values: Policy #3')
plt.show()
```



2. Change in the number of susceptible, recovered, and infected people in total after 200 days with and without NPI:

- With NPI, at day 200, there are 22,330 more people in the S compartment.
- With NPI, at day 200, there are 107 fewer people in the I compartment.
- With NPI, at day 200, there are 22,222 fewer people in the R compartment.

[13]:
```
S_NPI3, I_NPI3, R_NPI3 = simulate(S, I, R, using_alpha=alpha_afterNPI3)
display(S_NPI3, I_NPI3, R_NPI3, 'policy 3')
```

```
S, I, R values with and without policy 3:

Compartment  Day 200 pop w/out policy 3  Day 200 pop w/ policy 3   Difference
          S                1062735.03098             1085065.32730 -22330.29632
          I                   2374.61864                2267.15936    107.45929
          R                1649157.35037             1626934.51334  22222.83703
```

3.0.4 Policy #4

Spend the budget proportional to each node's average beta value weighted by the susceptible population size in comorbidity and vulnerability compartments. 1. In the plot, blue-shaded nodes have more people in low socially vulnerable/ comoridity-risk categories. On the contrary, red-shaded nodes have more people in high socially vulnerable/ comorbidity-risk categories. We see that nodes shaded red see a greater reduction in α. For instance a red node with an initial α value of ≈ 0.41 drops to ≈ 0.175 after NPI spending under Policy #4. On the other hand, a blue node with an initial α value of ≈ 0.20 drops to ≈ 0.18. Hence, we can see that Policy #4 is allocating more funds to nodes that are more socially vulnerable/ have higher comorbidity rates.

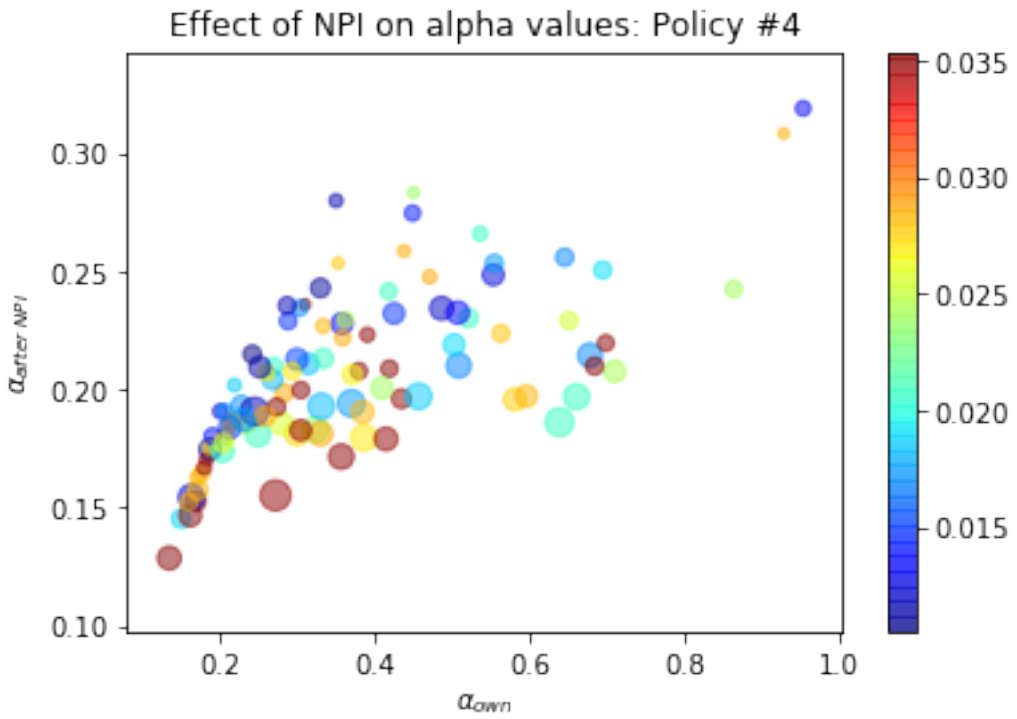
[14]:
```
Svc_0_weighted = Svc_0 * beta_old
S_avg = np.average(Svc_0_weighted, axis=(1,2))
proportions = S_avg / sum(S_avg)

budgets4 = proportions * npi_budget
print(f'sum of all budgets allocated for each node is {sum(budgets4)}') # this␣
 ↪checks out, we expect it to be 1mil

alpha_afterNPI4 = [g(g_inv(alpha) + amtPerNode) for amtPerNode, alpha in␣
 ↪zip(budgets4, alpha_predicted)]

scatter_plot(alpha_afterNPI4)
plt.title('Effect of NPI on alpha values: Policy #4')
plt.show()
```

```
sum of all budgets allocated for each node is 999999.9999999995
```



2. Change in the number of susceptible, recovered, and infected people in total after 200 days with and without NPI:

- Using policy 4, at day 200, there are 333,659 more people in the S compartment.
- Using policy 4, at day 200, there are 1,913 fewer people in the I compartment.
- Using policy 4, at day 200, there are 331,745 fewer people in the R compartment.

[15]:
```
S_NPI4, I_NPI4, R_NPI4 = simulate(S, I, R, using_alpha=alpha_afterNPI4)
display(S_NPI4, I_NPI4, R_NPI4, 'policy 4')
```

```
S, I, R values with and without policy 4:

Compartment  Day 200 pop w/out policy 4  Day 200 pop w/ policy 4    Difference
          S                1062735.03098            1396394.16113 -333659.13015
          I                   2374.61864                461.19549    1913.42315
          R                1649157.35037            1317411.64338  331745.70699
```

4 (c)

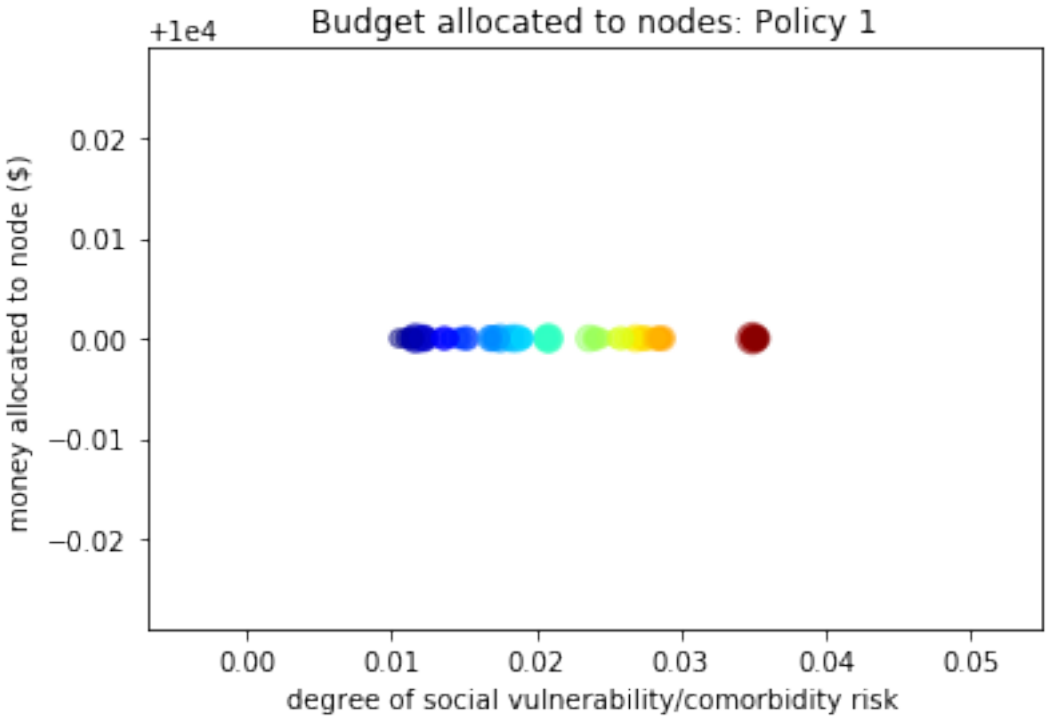
In this section we gauge the fairness of the four policies from section (b). We view a policy as fair if the policy provides the largest share of the budget to nodes most in need of aid. To measure fairness in this way we've created plots whose x-axis is the degree of social vulnerability/comorbidity risk (this is used to color the nodes in the `scatter_plot` function), and whose y-axis is the amount of money allocated to each node. Each dot on the plot represents one of the 100 nodes.

4.0.1 Policy #1

We see from the plot below that all nodes, regardless of size or social vulnerability/comorbidity risk, recieve an identical budget. This means the policy allocates the budget in an equal way, which is by some measures fair. However, it's apparent that this policy is not equitable as nodes most in need of aid (ie the large red dots in the plot) recieve the same amount of aid as the nodes that need little aid (ie the small purple dots in the plot)

[16]:
```
budgets1 = [amtPerNode] * len(alpha_predicted)

plt.figure()
colors=(beta_old[np.newaxis,:,:]*Svc_0_PMF).mean(axis=(1,2))
plt.scatter(colors, budgets1, s=N/500, c=colors, cmap='jet', alpha=0.5)
plt.title("Budget allocated to nodes: Policy 1")
plt.xlabel(r"degree of social vulnerability/comorbidity risk")
plt.ylabel(r"money allocated to node ($)")
plt.show()
```

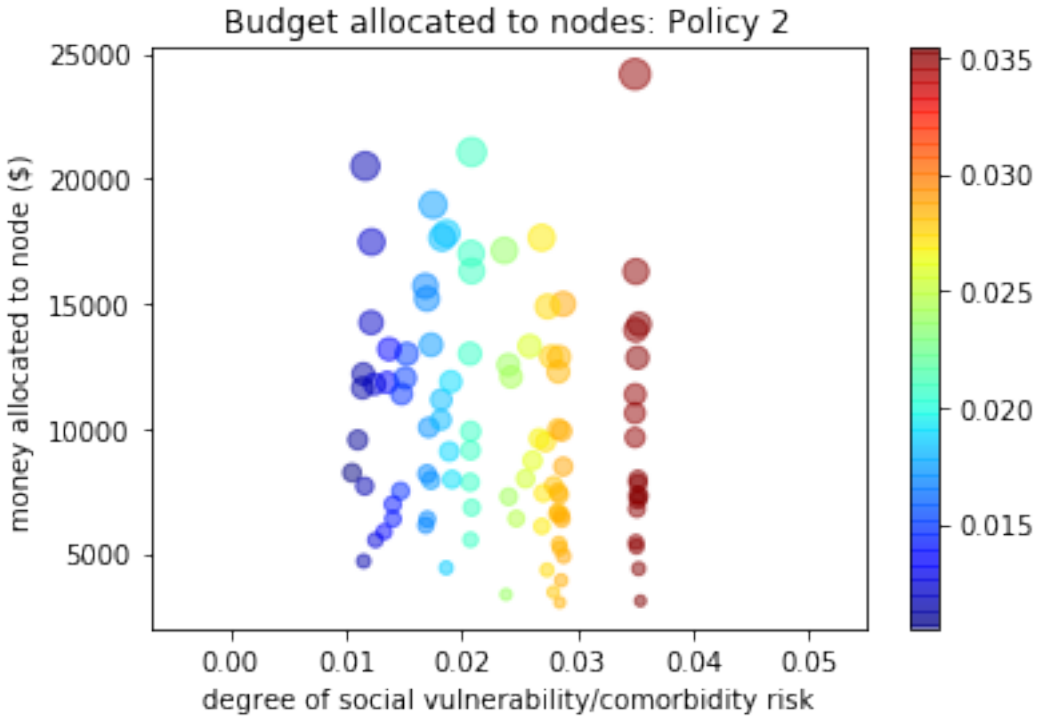


4.0.2 Policy #2

Observing the plot below, we can see that the budget is allocated based on the nodes' populations. Clearly, nodes with larger populations are rewarded with higher allocations, and nodes with lower populations don't receive as much. Policy #2 does not take into account the node's social vulnerability/ comorbidity risk. Also, the smaller dots are closer to the bottom, while the

larger dots are closer to the top. It may be fair in some sense to prioritize nodes with higher populations. However, it is unfair if nodes with highly vulnerable populations are not provided with the most funding.

[17]:
```
plt.figure()
colors=(beta_old[np.newaxis,:,:]*Svc_0_PMF).mean(axis=(1,2))
plt.scatter(colors, budget_2, s=N/500, c=colors, cmap='jet', alpha=0.5)
plt.title("Budget allocated to nodes: Policy 2")
plt.xlabel(r"degree of social vulnerability/comorbidity risk")
plt.ylabel(r"money allocated to node ($)")
plt.colorbar()
plt.show()
```



4.0.3 Policy #3

Policy: Spending the budget such that $\alpha_{own} - \alpha_{afterNPI}$ is the same for all nodes,that is the change in α induced by the NPI is the same for all nodes.

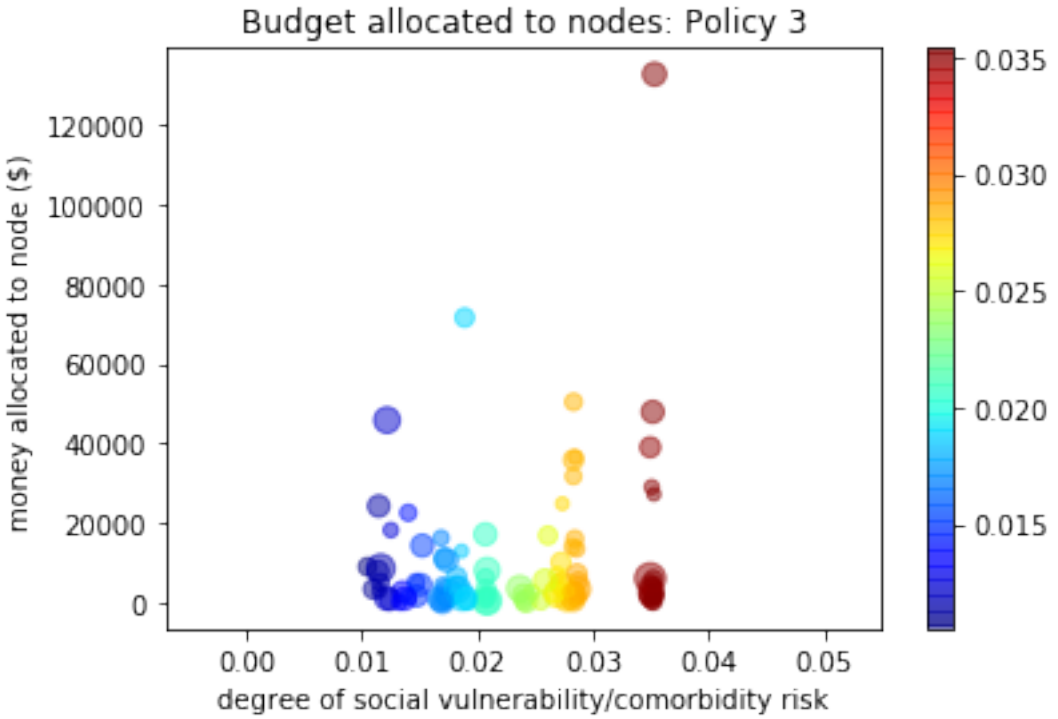
Since the original 'rankings' of the nodes are determined by α_{own}, if we add the same $\Delta\alpha$ to each α_{own}, the original 'rankings' of the nodes will be preserved.

The definition of envy-free fairness is: no group would prefer another group's allocation to their own allocation. In this case, each node has the same $\Delta\alpha$. Although the nodes receive the same adjustment in α, each node has different social vulnerability and comorbidity levels in the corresponding compartments. Thus, for Policy #3, there may be **equality** in how the NPI is allocated, but there might not necessarily be **equity**.

When we consider fairness through awareness, we need to consider the degree to which individuals are similar with respect to the task at hand. Hence, we can possibly measure if two nodes' social vulnerability/comorbidity risk is similar to determine whether the $\Delta\alpha$ should be similar. However, Policy 3 is 'unfair' because two nodes could be very similar or two nodes could be very different in terms of social vulnerability/comborbidity risk. This is not taken into account in Policy#3 as each node receives the same $\Delta\alpha$.

To conclude, Policy #3 is not fair because it does not consider individual fairness. By keeping $\Delta\alpha$ the same for all nodes, we are not taking into account SV/Comorbidity distributions that need more NPI treatment than others.

[18]:
```
plt.figure()
colors=(beta_old[np.newaxis,:,:]*Svc_0_PMF).mean(axis=(1,2))
plt.scatter(colors, budgets3, s=N/500, c=colors, cmap='jet', alpha=0.5)
plt.title("Budget allocated to nodes: Policy 3")
plt.xlabel(r"degree of social vulnerability/comorbidity risk")
plt.ylabel(r"money allocated to node ($)")
plt.colorbar()
plt.show()
```



In the plot above, we see that the nodes we would expect to see the highest allocation- large, red dots- actually receive low allocations for the most part. A 'fair' policy should provide nodes with higher populations of socially-vulnerable/comorbidity-risk people with greater funding. Yet, we see that some of the larger, blue-shaded dots are receiving similar amounts of funding to the red-shaded dots of the same size.

4.0.4 Policy #4

It is apparent in this plot that red nodes (which have more people in high socially vulnerable/ comorbidity-risk categories) recieve the most aid, and purple nodes recieve the least. Moreover, larger nodes recieve more of the budget than smaller nodes. To us this indicates an equitable allocation of the budget under policy 4.

[19]:
```
plt.figure()
colors=(beta_old[np.newaxis,:,:]*Svc_0_PMF).mean(axis=(1,2))
plt.scatter(colors, budgets4, s=N/500, c=colors, cmap='jet', alpha=0.5)
plt.title("Budget allocated to nodes: Policy 4")
plt.xlabel(r"degree of social vulnerability/comorbidity risk")
plt.ylabel(r"money allocated to node ($)")
plt.show()
```