Network Analysis and Modeling
CSCI 5352, Fall 2020
Prof. Dan Larremore
Problem Sets 7 and 8
Student: Alex Book; Collaborators: None

1. (70 pts total) *The impact of community structure on spreading processes.* In this question, you will explore via a numerical simulation the impact that community structure has on the dynamics of a spreading process.

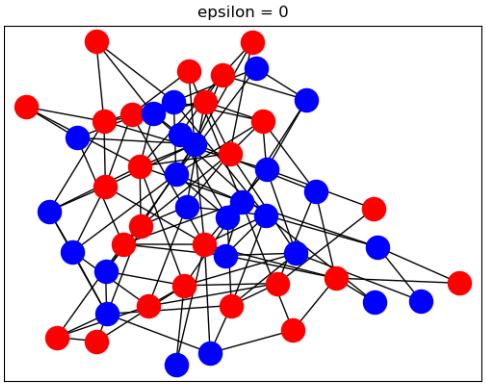
   (a) (10 pts) The "planted partition" model is a one-parameter version of the stochastic block model that generates *simple* synthetic networks with a community structure of varying strength. In this setting "varying strength" means how distinct the communities are- strong community structure means that most edges fall within the communities and a few between, while weak community structure means edges are nearly equally distributed within or between groups.

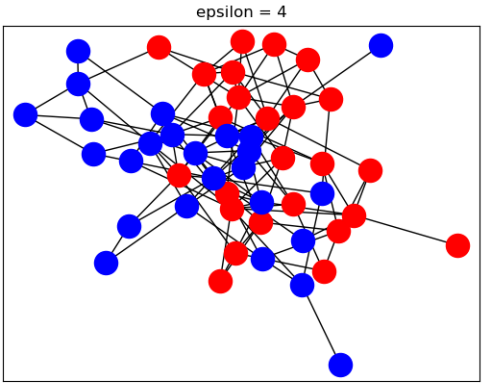
   Let n be a large and even number of vertices, let every vertex have a constant mean degree $c \geq 0$, and let $q = 2$ be the number of equal-sized communities in the model. If we define the probability of an edge existing within a group as $p_{in} = c_{in}/n$ and the probability of an edge existing between two groups as $p_{out} = c_{out}/n$, then the identity $2c = c_{in} + c_{out}$ is implied.

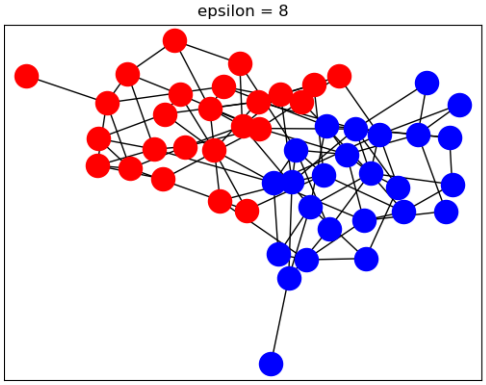
   - Derive expressions for p_{in} and p_{out} in terms of only constants, c, n, and the parameter $\epsilon = c_{in} - c_{out}$, and hence show that this is a one parameter model.

   $$\begin{array}{c|c} 2c = np_{in} + np_{out} & 2c = np_{in} + np_{out} \\ \epsilon = np_{in} - np_{out} & \epsilon = np_{in} - np_{out} \\ 2c + \epsilon = 2np_{in} & 2c - \epsilon = 2np_{out} \\ p_{in} = \frac{2c+\epsilon}{2n} & p_{out} = \frac{2c-\epsilon}{2n} \end{array}$$

   - Generate and create simple visualizations (using an off-the-shelf spring-embedding algorithm) of the three graphs, for $n = 50$, $q = 2$, $c = 5$, and $\epsilon = 0, 4, 8$. Comment on the strength of the community structure each of these graphs exhibits.







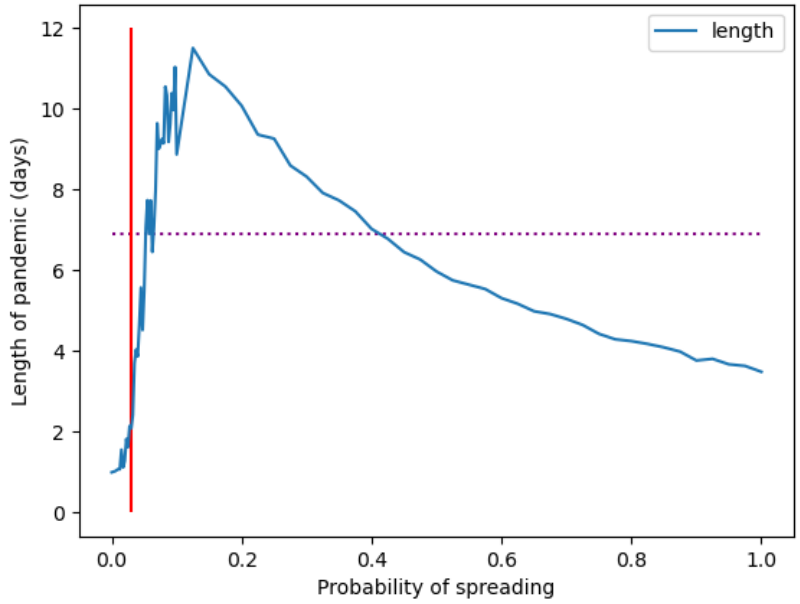
The community structure of each graph seems to strengthen as ϵ increases (there is very little discernible community structure for $\epsilon = 0$, some separation by type for $\epsilon = 4$, and very strong separation for $\epsilon = 8$). This is because as ϵ increases, c_{in} is increasingly greater than c_{out}, which by extension means that nodes are increasingly more likely to be connected within their own group than to outside nodes, yielding a stronger and stronger community structure.

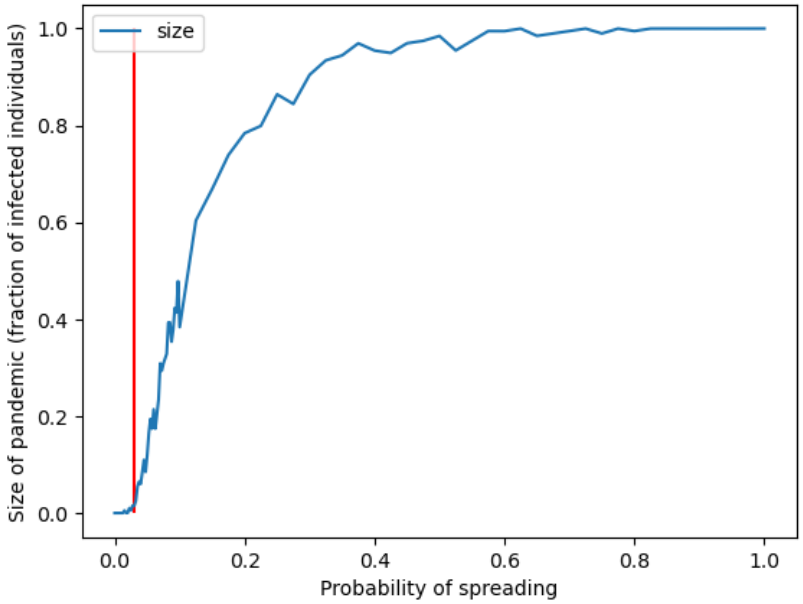
(b) (30 pts) Consider the following rules for a simple discrete time SI spreading process—one which we did *not* explicitly describe in class: the probability that a vertex in the infected state I transmits its infection to a particular uninfected neighbor is a constant p, and we evaluate that transmission possibility at most once for that edge over the lifetime of the epidemic. Initially, at time $t = 0$, all vertices are in the uninfected state S ("susceptible"), time proceeds in discrete steps and vertex state updates are applied simultaneously when moving from $t \to t + 1$. A simulated epidemic is deemed complete when no new infected nodes are produced in a time step. The epidemic's *size* s is the fraction of nodes in the I state when the epidemic is complete, and its length ℓ is the time t=ℓ at which the last node in the epidemic became infected. To begin the epidemic, at time $t = 1$, choose a node uniformly at random to infect.

Using the planted partition model from part (1a), you can generate any number of

synthetic networks to use as a substrate for studying the behavior of the above simple *SI* spreading process. Using $n = 1000$, $c = 8$, and $\epsilon = 0$, measure the average epidemic size $\langle s \rangle$ and epidemic length $\langle \ell \rangle$ as a function of $p \in [0, 1]$

- Make two figures, showing these measured relationships. On the length figure, include a horizontal line showing $\langle \ell \rangle = \log(n)$. On both figures, include a vertical line as the "critical value" of p. (A critical value p_{crit} is where a phase transition occurs.)





- Comment on the qualitative behavior of these measured relationships as a function of the transmission probability p, and discuss your results relative to your expectations.

  The results of the measured relationships are about what I expected. The length of the pandemic should have a middle-ish (yet left-leaning) peak, as very low probabilities of spreading would lead to the pandemic ending fairly quickly as the infection fails to continue spreading, and mid-to-high probabilities of spreading would lead to the pandemic ending somewhat quickly as the every individual becomes infected reasonably quickly. Meanwhile, the size should indeed exhibit an initially-quickly-growing value that grows more slowly (and eventually levels off) as p-values increase (as the pandemic is ended by the entire population becoming infected).

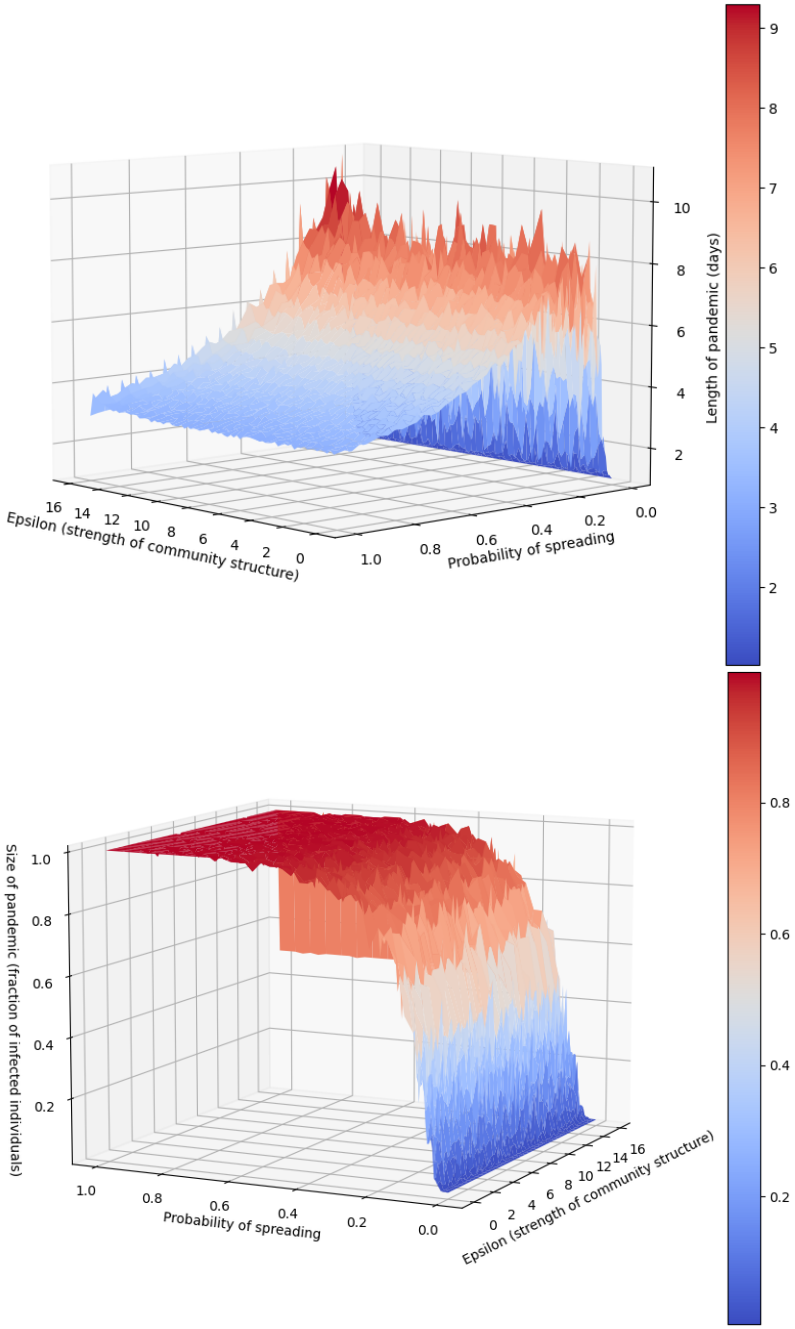
- Give an estimate p_{crit} and an intuitive explanation of why this value is special.

  I estimate p_{crit} to be roughly .03, as that is when the curve sharply turns upward in each figure. This value is special in that it is the threshold at which a spreading disease will start to ravage the population in which it is present (it has a relatively high likelihood of spreading among individuals until everyone is infected).

Hint 1: To get good figures, you will want smooth functions, which means averaging the measured s and ℓ over multiple draws from your data generating process (I used 500 repetitions, which took about 60 CPU minutes), which in this case is the network generator *and* the simulation. You will also want to vary p slowly enough over the $[0, 1]$ interval to get god resolution on how the average epidemic size changes, especially in the regions of p where s or ℓ are changing quickly.
Hint 2: Think carefully about where in $p \in [0, 1]$ the most interesting dynamics will occur. Recall that the critical value or "epidemic threshold" for a spreading process is a mean degree of 1 in the transmission graph.

(c) (30 pts) Now use the planted partition model of part (1a) to investigate whether the strength of community structure enhances, limits, or has no effect on epidemic size s and/or epidemic length ℓ. Let $n = 200$ and $c = 8$, and consider various combinations of the two parameters: $p \in [0, 1]$ and $\epsilon \in [0, 2c]$.

- Present your characterization of how community structure strength ϵ impacts epidemic size s and/or epidemic length ℓ using one or more figures that show the relationship clearly (smooth functions).



- Discuss the qualitative shape of these functions, and how they contrast, if at all, with your results from part (1b).

  The shape of these functions seems to be generally similar to the results from part (1b). The pandemic length once again appears to have a left-leaning middle-ish peak, and the pandemic size has a quickly-large rate of increase, which eventually levels off as it nears a value of 1. However, it's apparent that changing ϵ has an effect on the graph, at least as ϵ approaches $2c$ (this is because p_{out} is approaching zero). In the length graph, there is a clear spike in that region; in the size graph, the size of the pandemic levels off at .5 in that region.

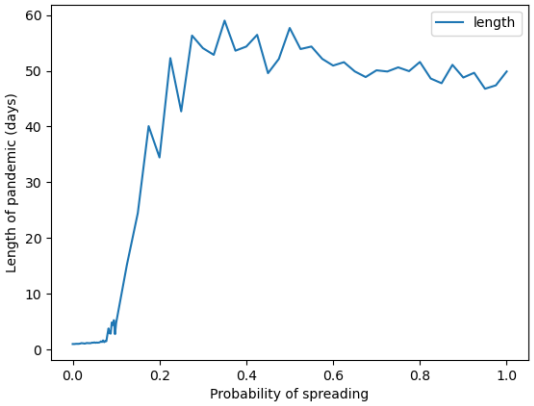
- Provide a brief intuitive explanation for why community structure strength does or does not impact the shape of the epidemic. (A small amount of extra credit if your discussion covers how the transmission rules for this simple epidemic model relate to your results.)

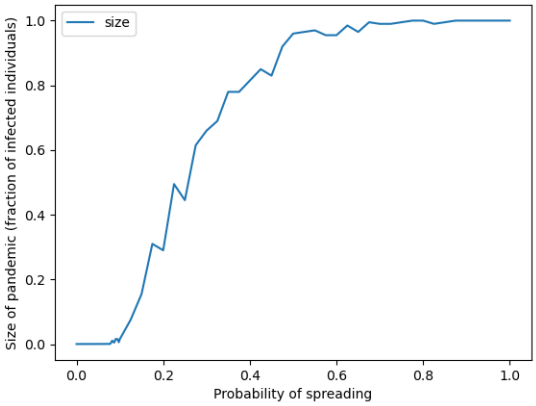
  The results explained above occur because when p_{out} has a value of zero, the disease cannot spread between communities. Since we have two equally-sized communities in this simulation, only half of the total population can become infected. As ϵ approaches $2c$ (and p_{out} approaches zero), this would cause the disease to take noticeably longer to spread to all individuals, and it would altogether fail to do so at $\epsilon = 2c$. Therefore, the results seen in the figures above are as expected.

Hint: Use a non-uniform spacing for choices of ϵ, and note that ϵ does not need to be an integer. I averaged over 2000 repetitions, which took approximately 220 CPU minutes, to produce two 3d plots.

2. (30 pts total) *The impact of long-range edges on spreading processes.* In this question, you will explore via a numerical simulation the impact that long-range links have on the dynamics of a spreading process.

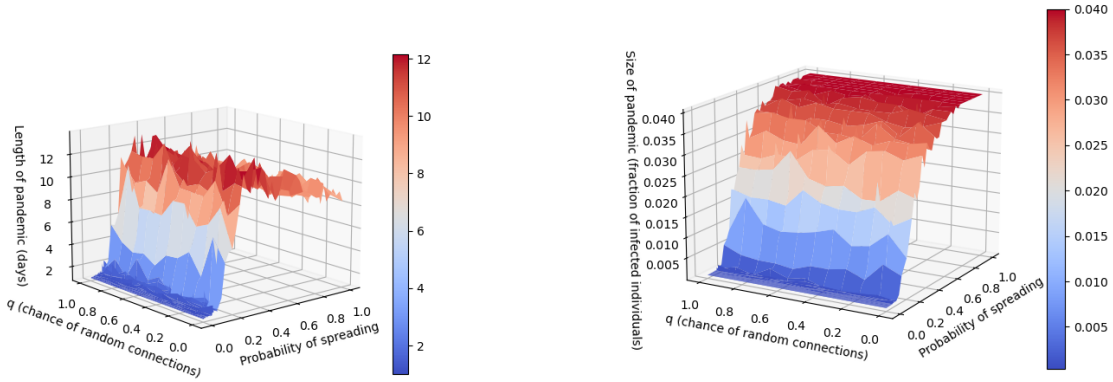
   (a) An $n \times n$ grid or lattice is a very simple network, in which each node, except for those on the boundary, connects to its neighbors above, below, left, and right of itself. Using the same SI model from question (1), investigate how the epidemic size s and epidemic length ℓ vary as a function of p, for $n = 50$. Present two figures showing these relationships and provide a brief interpretation of their shape, based on the way the SI model works and the shape of the network.





The shape of these figures is similar to the results from part (1b). However, it is clear that the p_{crit} value in these figures is noticeably higher (around .1 here, as opposed to roughly .03 in part (1b)). The rate of spread of the infection is also much slower than the rate exhibited in (1b). This is likely due to a lower average degree. This all stems from the structure of the network only allowing a maximum degree of 4, whereas the network in problem (1) allowed for higher degrees, therefore causing quicker spread of the infection. Additionally, the infection can only spread to susceptible individuals immediately next to the infected individuals, causing the infection to take a while to spread across the entirety of the lattice structure (whereas in (1b), infection could hop to the "other side" of the network, allowing the infection to spread more efficiently).

(b) If a network can be embedded in a metric space, as can a grid or lattice, a "long-range" connection is one that connects two vertices that are separated by many steps on the lattice, i.e., two nodes that are "far" apart. Let q represent the probability that some pair of nodes i, j are connected by a long range link. (Note: these links exist independently of the lattice edges.) Investigate how the epidemic size s and epidemic length ℓ depend on the variables q and p. Present your results clearly, and include a brief interpretation of how the long-range links change the epidemic dynamics relative to your results in part (2a).



Although I wasn't able to obtain entirely satisfactory figures to support the following trends (due to various issues with long runtimes causing me to run with a rather severe lack of sampling iterations), I am confident in how epidemic length and size depend on p and q. For any given p, as q increases, size increases much more quickly, as individuals will have higher average degrees (and will therefore infect more neighbors). Following the trend, length will decrease as q increases, as the infection will be able to spread much more quickly, causing the length of time for all individuals to become infected to decrease (also causing infections that don't infect the total population to carry on for longer than they would otherwise).

3. (15 pts extra credit) The *"resolution limit" for modularity maximization.* Consider a "ring graph" made of k cliques, each containing c vertices, arranged in circle, where each clique connects to each of its two nearest neighbors via a single edge. Let each edge have unit weight; let k be an even number; let P_1 be a partition with k groups where each group contains exactly one of the k cliques; and let P_2 be a partition with $k/2$ groups where each group contains one pair of adjacent cliques.

   Derive an expression for the difference in modularity scores $\Delta Q = Q_2 - Q_1$ and show that this difference is positive whenever $k > 2[\binom{c}{2} + 1]$. This is the so-called *resolution limit* of the modularity function, which says that at some size of the network, merging smaller module-like structures-here, the cliques-becomes more favorable under the modularity function than keeping them separate. Thus, finding the partition that maximized Q will miss these small structures.

   Hint: for each partition, begin by writing expressions for e_i, the number of edges with both endpoints in group i and d_i, the number of edges with at least one endpoint in group i.

Code for 1a:

```
import numpy as np
from webweb import Web
import networkx as nx
from pprint import pprint
import matplotlib.pyplot as plt

# choose the number of groups
B = 2
# choose the sizes of groups
n1 = 25
n2 = 25
n = n1+n2
# group memberships b
group_id = np.array([0]*n1 + [1]*n2)
# block matrix, omega
epsilon_list = [0, 4, 8]
for epsilon in epsilon_list:
    within = (10 + epsilon)/(2*n)
    between = (10 - epsilon)/(2*n)
    omega = [
        [within,between,0],
        [between,within,between],
        [0,between,within]]
    # make that happy little SBM!
    edge_list = []
    for i in range(1,n):
        for j in range(i):
            if np.random.rand() < omega[group_id[i]][group_id[j]]:
                edge_list.append([i,j])

    G = nx.Graph()
    G.add_edges_from(edge_list)
    pos_dict = nx.drawing.layout.spring_layout(G)

    nx.draw_networkx_nodes(G, pos_dict, nodelist=np.arange(25), node_color='r')
    nx.draw_networkx_nodes(G, pos_dict, nodelist=np.arange(25, 50), node_color='b')
    nx.draw_networkx_edges(G, pos_dict)

    title = 'epsilon = ' + str(epsilon)
    plt.title(title)
    plt.show()
```

Code for 1b:

```
import numpy as np
import networkx as nx
from pprint import pprint
import matplotlib.pyplot as plt
import pandas as pd

def generate_omega_and_groupid():
    B = 2
    n1 = 500
    n2 = 500
    n = n1 + n2
    group_id = np.array([0]*n1 + [1]*n2)
    epsilon = 0
    c = 8
    within = (2*c + epsilon)/(2*n)
    between = (2*c - epsilon)/(2*n)
    omega = [[within,between,0],
             [between,within,between],
             [0,between,within]]

    return omega, group_id

def generate_graph(omega, group_id):
    edge_list = []
    for i in range(1000):
        for j in range(i):
            if np.random.rand() < omega[group_id[i]][group_id[j]]:
                edge_list.append([i,j])

    G = nx.Graph()
    G.add_edges_from(edge_list)

    return G

def pandemic_simulation(G):
    spread = True
    S_I_list = np.array(['S']*1000)
    length = 0
    num_infected = 0

    while spread == True:
        spread = False
        if length == 0:
```

```
            S_I_list[np.random.randint(1000)] = 'I'
            num_infected += 1
        for node in G.nodes():
            if S_I_list[node] =='I':
                for neighbor in G.neighbors(node):
                    if S_I_list[neighbor] == 'S' and np.random.uniform() <= p:
                        S_I_list[neighbor] = 'I'
                        spread = True
                        num_infected += 1
        length += 1
    return length, num_infected/1000

def make_plots(filename):
    dfNew = pd.read_csv(filename)
    dfNew.columns = ['p', 'length', 'size']

    dfNew.plot(x = 'p', y = 'length')
    plt.xlabel('Probability of spreading')
    plt.ylabel('Length of pandemic (days)')
    plt.hlines(np.log(1000), 0, 1, colors='purple', linestyles='dotted')
    plt.vlines(.03, 0, 12, colors='red')
    plt.show()

    dfNew.plot(x = 'p', y = 'size')
    plt.xlabel('Probability of spreading')
    plt.ylabel('Size of pandemic (fraction of infected individuals)')
    plt.vlines(.03, 0, 1, colors='red')
    plt.show()

if __name__ == '__main__':
    omega, group_id = generate_omega_and_groupid()
    length_size_dict = {}
    p_array = [0, .0025, .005, .0075, .01, .0125, .015, .0175,
               .02, .0225, .025, .0275, .03, .0325, .035, .0375,
               .04, .0425, .045, .0475, .05, .0525, .055, .0575,
               .06, .0625, .065, .0675, .07, .0725, .075, .0775,
               .08, .0825, .085, .0875, .09, .0925, .095, .0975,
               .1, .125, .15, .175, .2, .225, .25, .275, .3, .325, .35, .375,
               .4, .425, .45, .475, .5, .525, .55, .575, .6, .625, .65, .675,
               .7, .725, .75, .775, .8, .825, .85, .875, .9, .925, .95, .975, 1]

    for p in p_array:
        print(p)
        length_array = []
        size_array = []
```

```
    for i in range(200):
        G = generate_graph(omega, group_id)
        length, size = pandemic_simulation(G)
        length_array.append(length)
        size_array.append(size)
    length_size_dict[p] = (np.average(length_array), np.average(size_array))

df = pd.DataFrame.from_dict(length_size_dict)
df.T.to_csv('during_the_day_1b.csv')

make_plots('during_the_day_1b.csv')
```

Code for 1c:

```
import numpy as np
import networkx as nx
from pprint import pprint
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import cm

def pandemic_simulation(G):
    spread = True
    S_I_list = np.array(['S']*200)
    length = 0
    num_infected = 0

    while spread == True:
        spread = False
        if length == 0:
            S_I_list[np.random.randint(200)] = 'I'
            num_infected += 1
        for node in G.nodes():
            if S_I_list[node] =='I':
                for neighbor in G.neighbors(node):
                    if S_I_list[neighbor] == 'S' and np.random.uniform() <= p:
                        S_I_list[neighbor] = 'I'
                        spread = True
                        num_infected += 1
        length += 1
    return length, num_infected/200

def make_plots(filename, a, b):
    dfNew = pd.read_csv(filename)

    X = list(dfNew['epsilon'])
    Y = list(dfNew['p'])
    Z = list(dfNew['length'])
    x = np.reshape(X, (a, b))
    y = np.reshape(Y, (a, b))
    z = np.reshape(Z, (a, b))
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    q = ax.plot_surface(x, y, z, cmap = cm.coolwarm)
    fig.colorbar(q)
    ax.set_xlabel('Epsilon (strength of community structure)')
    ax.set_ylabel('Probability of spreading')
```

```
    ax.set_zlabel('Length of pandemic (days)')
    ax.view_init(15, 15)
    plt.show()

    X = list(dfNew['epsilon'])
    Y = list(dfNew['p'])
    Z = list(dfNew['size'])
    x = np.reshape(X, (a, b))
    y = np.reshape(Y, (a, b))
    z = np.reshape(Z, (a, b))
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    q = ax.plot_surface(x, y, z, cmap = cm.coolwarm)
    fig.colorbar(q)
    ax.set_xlabel('Epsilon (strength of community structure)')
    ax.set_ylabel('Probability of spreading')
    ax.set_zlabel('Size of pandemic (fraction of infected individuals)')
    ax.view_init(15, -15)
    plt.show()

if __name__ == '__main__':
    B = 2
    n1 = 100
    n2 = 100
    n = n1 + n2
    group_id = np.array([0]*n1 + [1]*n2)
    c = 8
    epsilon_array = [0, .025*c, .05*c, .075*c, 0.1*c, .125*c, .15*c, .175*c,
                     .2*c, .225*c, .25*c, .275*c, 0.3*c, .325*c, .35*c, .375*c,
                     .4*c, .425*c, .45*c, .475*c, 0.5*c, .525*c, .55*c, .575*c,
                     .6*c, .625*c, .65*c, .675*c, 0.7*c, .725*c, .75*c, .775*c,
                     .8*c, .825*c, .85*c, .875*c, 0.9*c, .925*c, .95*c, .975*c,
                    1.0*c, 1.025*c, 1.05*c, 1.075*c, 1.1*c, 1.125*c, 1.15*c, 1.175*c,
                    1.2*c, 1.225*c, 1.25*c, 1.275*c, 1.3*c, 1.325*c, 1.35*c, 1.375*c,
                    1.4*c, 1.425*c, 1.45*c, 1.475*c, 1.5*c, 1.525*c, 1.55*c, 1.575*c,
                    1.6*c, 1.625*c, 1.65*c, 1.675*c, 1.7*c, 1.725*c, 1.75*c, 1.775*c,
                    1.8*c, 1.825*c, 1.85*c, 1.875*c, 1.9*c, 1.925*c, 1.95*c, 1.975*c, 2*c]

    p_array = [0, .005, .01, .015, .02, .025, .03, .035, .04, .045,
              .05, .055, .06, .065, .07, .075, .08, .085, .09, .095,
               .1, .125, .15, .175, .2, .225, .25, .275, .3, .325, .35, .375,
               .4, .425, .45, .475, .5, .525, .55, .575, .6, .625, .65, .675,
               .7, .725, .75, .775, .8, .825, .85, .875, .9, .925, .95, .975, 1]

    length_size_dict = {}
```

```
for epsilon in epsilon_array:
    print(epsilon)
    for p in p_array:
        length_array = []
        size_array = []
        for i in range(100):

            within = (2*c + epsilon)/(2*n)
            between = (2*c - epsilon)/(2*n)
            omega = [[within,between,0],
                     [between,within,between],
                     [0,between,within]]

            edge_list = []
            for i in range(200):
                for j in range(i):
                    if np.random.rand() < omega[group_id[i]][group_id[j]]:
                        edge_list.append([i,j])

            G = nx.Graph()
            G.add_edges_from(edge_list)

            length, size = pandemic_simulation(G)
            length_array.append(length)
            size_array.append(size)
        length_size_dict[(epsilon, p)] = (np.average(length_array),np.average(size_array))

df = pd.DataFrame(columns = ['epsilon', 'p', 'length', 'size'])
for key, value in length_size_dict.items():
    df.loc[-1] = [key[0], key[1], value[0], value[1]]
    df.index += 1
df.to_csv('data.csv')

a = len(epsilon_array)
b = len(p_array)
make_plots('data.csv', a, b)
```

Code for 2a:

```
import numpy as np
import networkx as nx
from pprint import pprint
import matplotlib.pyplot as plt
import pandas as pd
from copy import deepcopy

def get_neighbors(A, x, y):
    changes = [(1, 0), (0, 1), (-1, 0), (0, -1)]
    reachable = []
    for pair in changes:
        if (x+pair[0] < len(A)) and (x+pair[0] >= 0) and (y+pair[1] < len(A)) and
        (y+pair[1] >= 0):
            reachable.append((x+pair[0], y+pair[1]))
    return reachable

def pandemic_simulation(p):
    S_I_grid = [['S' for x in range(50)] for y in range(50)]

    spread = True
    length = 0
    num_infected = 0

    while spread is True:
        if num_infected == 2500:
            break
        copy_grid = S_I_grid.copy()
        spread = False
        if length == 0:
            copy_grid[np.random.randint(50)][np.random.randint(50)] = 'I'
            num_infected += 1
        for i in range(50):
            for j in range(50):
                if S_I_grid[i][j] == 'I':
                    for coords in get_neighbors(S_I_grid, i, j):
                        if S_I_grid[coords[0]][coords[1]] == 'S' and np.random.uniform() <= p:
                            copy_grid[coords[0]][coords[1]] = 'I'
                            spread = True
                            num_infected += 1
        length += 1
        S_I_grid = copy_grid.copy()
    return length, num_infected/2500
```

```
def make_plots(filename):
    dfNew = pd.read_csv(filename)
    # print(dfNew)

    dfNew.plot(x = 'p', y = 'length')
    plt.xlabel('Probability of spreading')
    plt.ylabel('Length of pandemic (days)')
    plt.show()

    dfNew.plot(x = 'p', y = 'size')
    plt.xlabel('Probability of spreading')
    plt.ylabel('Size of pandemic (fraction of infected individuals)')
    plt.show()

if __name__ == '__main__':
    p_array = [0, .0025, .005, .0075, .01, .0125, .015, .0175, .02,
                  .0225, .025, .0275, .03, .0325, .035, .0375, .04,
                  .0425, .045, .0475, .05, .0525, .055, .0575, .06,
                  .0625, .065, .0675, .07, .0725, .075, .0775, .08,
                  .0825, .085, .0875, .09, .0925, .095, .0975, .1,
                  .125, .15, .175, .2, .225, .25, .275, .3,
                  .325, .35, .375, .4, .425, .45, .475, .5,
                  .525, .55, .575, .6, .625, .65, .675, .7,
                  .725, .75, .775, .8, .825, .85, .875, .9,
                  .925, .95, .975, 1]

    length_size_dict = {}

    for p in p_array:
        print(p)
        length_array = []
        size_array = []
        for i in range(200):
            length, size = pandemic_simulation(p)
            length_array.append(length)
            size_array.append(size)
        length_size_dict[p] = (np.average(length_array), np.average(size_array))
    df = pd.DataFrame(columns = ['p', 'length', 'size'])
    for key, value in length_size_dict.items():
        df.loc[-1] = [key, value[0], value[1]]
        df.index += 1
    df.to_csv('FIFTYBYFIFTY_0025UpToPoint1_025UpTo1_200iter.csv')

    make_plots('FIFTYBYFIFTY_0025UpToPoint1_025UpTo1_200iter.csv')
```

Code for 2b:

```
import numpy as np
import networkx as nx
from pprint import pprint
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import cm

def get_neighbors(A, x, y):
    changes = [(1, 0), (0, 1), (-1, 0), (0, -1)]
    reachable = []
    for pair in changes:
        if (x+pair[0] < len(A)) and (x+pair[0] >= 0) and (y+pair[1] < len(A)) and
        (y+pair[1] >= 0):
            reachable.append((x+pair[0], y+pair[1]))
    return reachable

def get_random_connections(q):
    A = [[0 for x in range(100)] for y in range(100)]

    for i in range(100):
        for j in range(100):
            if np.random.uniform() <= q:
                A[i][j] = 1

    return A

def pandemic_simulation(p, connections):
    S_I_grid = [['S' for x in range(10)] for y in range(10)]

    spread = True
    length = 0
    num_infected = 0

    while spread is True:
        if num_infected == 100:
            break
        copy_grid = S_I_grid.copy()
        spread = False
        if length == 0:
            copy_grid[np.random.randint(10)][np.random.randint(10)] = 'I'
            num_infected += 1
        for i in range(10):
            for j in range(10):
```

```
                if S_I_grid[i][j] == 'I':
                    for coords in get_neighbors(S_I_grid, i, j):
                        if S_I_grid[coords[0]][coords[1]] == 'S' and np.random.uniform() <= p:
                            copy_grid[coords[0]][coords[1]] = 'I'
                            spread = True
                            num_infected += 1
                    for x in range(len(connections[10*i + j])):
                        if connections[10*i + j] == 1:
                            a, b = str(x).split()
                            if S_I_grid[int(a)][int(b)] == 'S' and np.random.uniform() <= p:
                                copy_grid[int(a)][int(b)] = 'I'
                                spread = True
                                num_infected += 1

        length += 1
        S_I_grid = copy_grid.copy()
    return length, num_infected/2500

def make_plots(filename, a, b):
    dfNew = pd.read_csv(filename)

    X = list(dfNew['p'])
    Y = list(dfNew['q'])
    Z = list(dfNew['length'])
    x = np.reshape(X, (a, b))
    y = np.reshape(Y, (a, b))
    z = np.reshape(Z, (a, b))
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    q = ax.plot_surface(x, y, z, cmap = cm.coolwarm)
    fig.colorbar(q)
    ax.set_xlabel('Probability of spreading')
    ax.set_ylabel('q (chance of random connections)')
    ax.set_zlabel('Length of pandemic (days)')
    ax.view_init(15, -15)
    plt.show()

    X = list(dfNew['p'])
    Y = list(dfNew['q'])
    Z = list(dfNew['size'])
    x = np.reshape(X, (a, b))
    y = np.reshape(Y, (a, b))
    z = np.reshape(Z, (a, b))
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
```

```
    q = ax.plot_surface(x, y, z, cmap = cm.coolwarm)
    fig.colorbar(q)
    ax.set_xlabel('Probability of spreading')
    ax.set_ylabel('q (chance of random connections)')
    ax.set_zlabel('Size of pandemic (fraction of infected individuals)')
    ax.view_init(15, -15)
    plt.show()

if __name__ == '__main__':
    p_array = [0, .0025, .005, .0075, .01, .0125, .015, .0175, .02,
                  .0225, .025, .0275, .03, .0325, .035, .0375, .04,
                  .0425, .045, .0475, .05, .0525, .055, .0575, .06,
                  .0625, .065, .0675, .07, .0725, .075, .0775, .08,
                  .0825, .085, .0875, .09, .0925, .095, .0975, .1,
                  .125, .15, .175, .2, .225, .25, .275, .3,
                  .325, .35, .375, .4, .425, .45, .475, .5,
                  .525, .55, .575, .6, .625, .65, .675, .7,
                  .725, .75, .775, .8, .825, .85, .875, .9,
                  .925, .95, .975, 1]

    q_array = [0, .1, .2, .3, .4, .5, .6, .7, .8, .9, 1]

    length_size_dict = {}

    for p in p_array:
        for q in q_array:
            print(p)
            length_array = []
            size_array = []
            for i in range(50):
                connections = get_random_connections(q)
                length, size = pandemic_simulation(p, connections)
                length_array.append(length)
                size_array.append(size)
            length_size_dict[(p, q)] = (np.average(length_array), np.average(size_array))

    df = pd.DataFrame(columns = ['p', 'q', 'length', 'size'])
    for key, value in length_size_dict.items():
        df.loc[-1] = [key[0], key[1], value[0], value[1]]
        df.index += 1
    df.to_csv('TENBYTEN_0025UpToPoint1_025UpTo1_50iter.csv')

    a = len(p_array)
    b = len(q_array)
    make_plots('TENBYTEN_0025UpToPoint1_025UpTo1_50iter.csv', a, b)
```