MITx 6.419x Data Analysis: Statistical Modeling and Computation in Applications
Written report -- Homework3_Network_Analysis
Collaborators: none

4/19/21

> ## Problem 1: Suggesting Similar Papers
>
> ### Part (c): (2 points) (100 word limit.)
>
> How does the time complexity of your solution involving matrix multiplication in part (a) compare to your friend's algorithm?

solution:

In my solution,

$$C = A^T \cdot A$$

. Matrix

$$A$$

and

$$A^T$$

both have n rows and n column.
Traversing matrix C uses two cycles:

1. traversing n rows, each row includes n elements, the complexity for that is

$$n \times n$$

2. Each element in matrix C is calculated by multiplying a row of A by a column of B, and a round of cycles use n elements, the complexity for that is n.

So the time complexity of my solution is

$$n \times n \times n$$

.

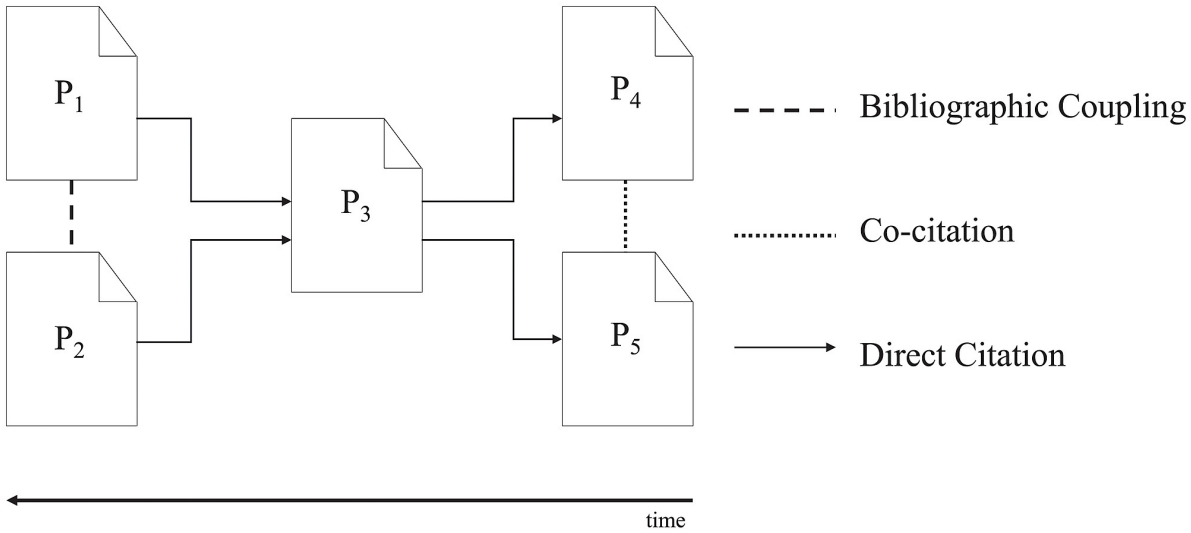
Comparing to my friend's algorithm, the time complexity of both algorithm are

$$n \times n \times n$$

Part (d): (3 points)(200 word limit.)

Bibliographic coupling and cocitation can both be taken as an indicator that papers deal with related material. However, they can in practice give noticeably different results. Why? Which measure is more appropriate as an indicator for similarity between papers?

solution:



The picture above shows the difference between co-citations and Bibliographic coupling.

The more co-citations two documents receive, the higher their co-citation strength, and the more likely they are semantically related. The "coupling strength" of two given documents is higher the more citations to other documents they share.

Both methods are effective. In my opinionis， Co-citation is more appropriate as an indicator for similarity between papers.
Co-citation analysis provides a forward-looking assessment on document similarity in contrast to Bibliographic Coupling, which is retrospective. The citations a paper receives in the future depend on the evolution of an academic field, thus co-citation frequencies can still change. In the adjacent diagram, for example, Doc A and Doc B may still be co-cited by future documents, say Doc F and Doc G. This characteristic of co-citation allows for a dynamic document classification system when compared to Bibliographic Coupling.

Problem 2: Investigating a time-varying criminal network

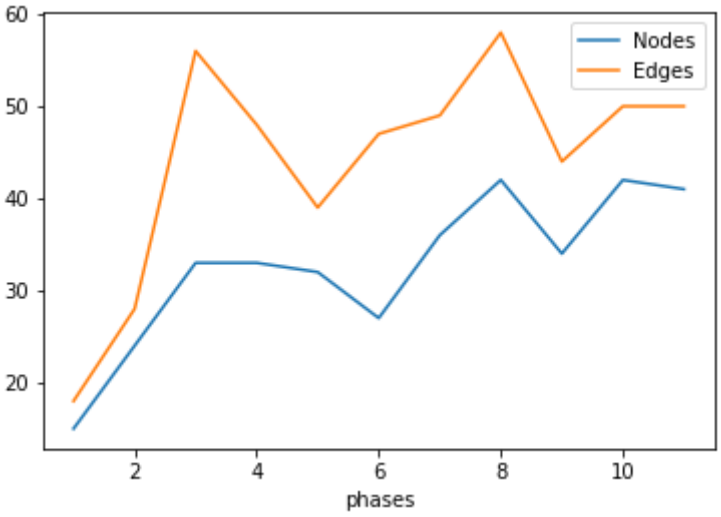
Part (c): (2 points) (~100 words, 200 word limit.)

Observe the plot you made in Part (a) Question 1. The number of nodes increases sharply over the first few phases then levels out. Comment on what you think may be causing this effect. Based on your answer, should you adjust your conclusions in Part (b) Question 5?

solution:

```
## Plotting tools

import matplotlib.pyplot as plt
import matplotlib.colors as mcolors

#Import data processing packages
```

```
import pandas as pd
import numpy as np

#Import network packages
import networkx as nx

# Install pygraphviz

phases = {}
G = {}
G_summaries = {'Nodes': [], 'Edges': []}
for i in range(1,12):
  var_name = "phase" + str(i)
  file_name = "https://raw.githubusercontent.com/ragini30/Networks-
Homework/main/" + var_name + ".csv"
  phases[i] = pd.read_csv(file_name, index_col = ["players"])
  phases[i].columns = "n" + phases[i].columns
  phases[i].index = phases[i].columns
  phases[i][phases[i] > 0] = 1
  G[i] = nx.from_pandas_adjacency(phases[i])
  G[i].name = var_name
  G_summaries['Nodes'].append(G[i].number_of_nodes())
  G_summaries['Edges'].append(G[i].number_of_edges())
plt.plot(range(1,12), [ G[i].number_of_nodes() for i in range(1,12) ],
label="Nodes")
plt.plot(range(1,12), [ G[i].number_of_edges() for i in range(1,12) ],
label="Edges")
plt.legend()
plt.xlabel('phases')
plt.show()
```



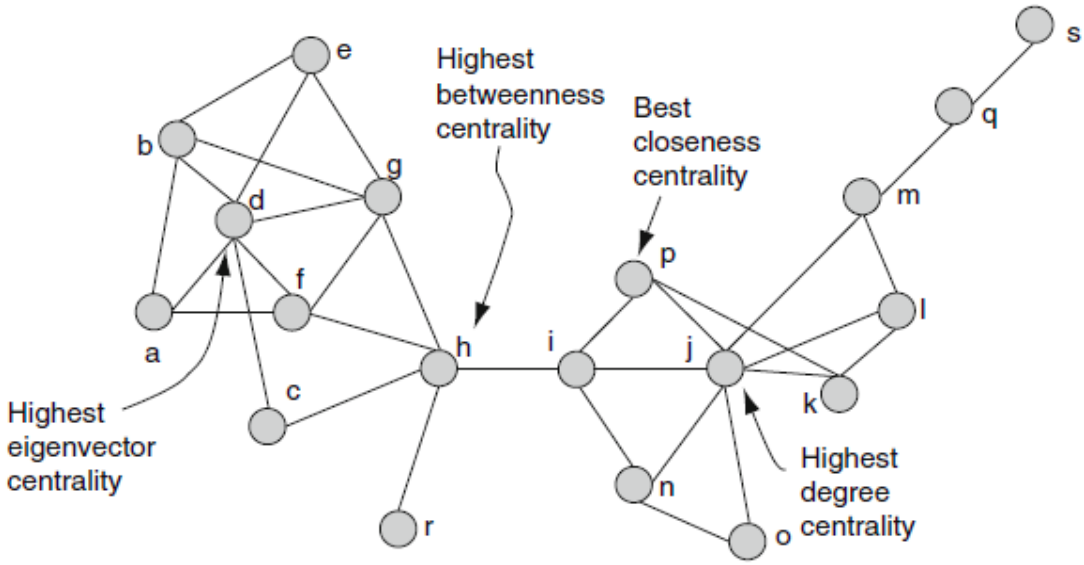
In the process of criminal investigation, the higher the degree of network centralization of criminals, the more difficult it is to be arrested. On the contrary, the farther away from the crime centre, the easier it is to be arrested. Therefore, at the initial stage of the investigation, only a few marginal criminals were arrested, and the criminal network of these criminals was sparse, so there were few nodes and edges. As the investigation proceeded, the criminals on the centre were gradually arrested. The network of centre criminals is very dense, so more nodes and edges can be found.

Therefore, for the accurate calculation of Part (b) Question 5, the centralization weight of the later phase should be increased, which is more reasonable than the mean value.

> ## Part (d): (5 points) (~300 words, 400 word limit.)
>
> In the context of criminal networks, what would each of these metrics teach you about the importance of an actor's role in the traffic? In your own words, could you explain the limitations of degree centrality? In your opinion, which one would be most relevant to identify who is running the illegal activities of the group? Please justify.

Solution:

Each of these metrics shows different roles of an actor in the traffic, it depends on which perspective we choose.
One of the reasons why so many centrality measures have been defined is because all of the measures have limitations. Each works well for probing certain phenomena, but at the same time, each measure also fails to capture other important structural characteristics of a network.

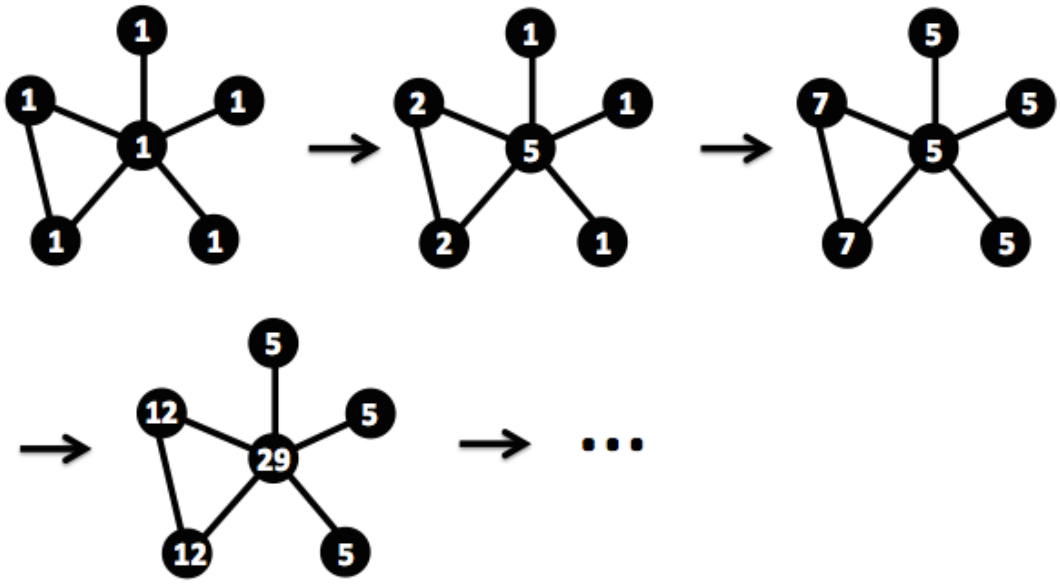


The four centrality measures each identify a different node as the "most central." None of the centrality measures are "wrong"; they just all answer different questions. If we're doing a simple analysis of who's most popular in a social network, prioritizing nodes by betweenness centrality could be misleading. Meanwhile, a lot of immunization strategies in analyses of epidemic spreading rely on immunizing nodes with high betweenness. In this circumstance, immunizing people based on their degree centrality could be sub-optimal. All of the above can be summarized as follows: each centrality measure is attempting to capture some intuition or some set of intuitions, so it's clear that each will be of limited use to address scientific questions where some other intuition is needed.
In my opinion, betweenness centrality would be most relevant to identify who is running the illegal activities of the group.

In addition, there is a subtle limitation in degree centrality, that is, a centrality measure is inadequate to capture even it was intended. In a paper titled Localization and Centrality in Networks., T. Martin, X. Zhang, and M. E. J. Newman argue that this can happen for eigenvector centrality in certain circumstances

> Eigenvector centrality is based on the intuition that a node’s importance in a network is determined by how important its neighbors are. We can imagine initializing the centrality of each node in a network to 1 and then iteratively updating the centrality of each node to be the sum of the centralities of its neighbors’ centralities. We can visualize the first steps of

this process for a simple undirected network here:



The central node in this example passes its centrality along to its neighbors, and then, in the next step, that centrality gets reflected back to the central node. This repeated "reflection" can result in a heavy, and perhaps unjustifiably large, accumulation of eigenvector centrality near high-degree nodes in a network.

Part (e): (3 points) (~100 words, 200 word limit)

In real life, the police need to effectively use all the information they have gathered, to identify who is responsible for running the illegal activities of the group. Armed with a qualitative understanding of the centrality metrics from Part (d) and the quantitative analysis from part Part (b) Question 5, integrate and interpret the information you have to identify which players were most central (or important) to the operation.

Hint: Note that the definition of a player's "importance" (i.e. how central they are) can vary based on the question you are trying to answer. Begin by defining what makes a player important to the group (in your opinion) ; use your answers from Part (d) to identify which metric(s) are relevant based on your definition and then, use your quantitative analysis to identify the central and peripheral traffickers. You may also perform a different quantitative analysis, if your definition of importance requires it.

Solution:

In Part (d), I discussed the different types of quantitative analyses, and I think betweenness centrality would be most relevant to identify who is running the illegal activities of the group.

In Part (b) Question 5, I discussed the reason that The number of nodes increases sharply over the first few phases then levels out. And I mentioned that the centralization weight of the later phase should be increased, which is more reasonable than the mean value.
In Part (b) Question 5,I found that the number of nodes increases as the phase go on, so I choose the change of nodes to calculate the weight of each phase.

$$Weight = \frac{Number_{nodes\ in\ each\ phase}}{Number_{nodes\ in\ all\ phase}}$$

As the phase goes on, the weight value will increase. Of course, I just chose a relatively simple algorithm to calculate the weight value, it can also be calculated by other methods. The code is shown below and we find n1, n12 and n3 were most central (or important) to the operation.

```
sum_nodes = sum(G_summaries['Nodes'])

weight = np.array(G_summaries['Nodes'])/sum_nodes
bet_weight_df = pd.DataFrame()


def compute_metric(metric_fn):
  return { i: metric_fn(G[i]) for i in range(1,12) }

bet = compute_metric(lambda g: nx.betweenness_centrality(g, normalized = True))
## the betweenness values are normalized by 2/((n-1)(n-2)) for graphs, and 1/((n-
1)(n-2)) for directed graphs where n is the number of nodes in G.
bet_df = pd.DataFrame.from_dict(bet, orient='index')
bet_df = bet_df.fillna(0)
bet_df['weight'] = weight.tolist()
# print(bet_df)

for column in bet_df.drop('weight',axis=1):
#     bet_weight_df[column] = bet_df[weight_].multiply(df_bet[column],
axis="index")
    bet_weight_df[column] = bet_df.apply(lambda x: x[column] * x['weight'],
axis=1)
# print(bet_weight_df.sum())
bet_df.drop('weight',axis=1,inplace = True)
bet_statistics = pd.DataFrame([bet_weight_df.sum(),bet_df.mean(),
bet_df.astype(bool).sum(axis=0)], index=['Weight_Mean','Mean', 'Activity'])
bet_statistics = bet_statistics.T.reset_index()

print("Highest Weighted average Centrality : ")
print("Betweenness - ")
print(bet_statistics.sort_values(by=['Weight_Mean'], ascending =
False).head(10))
```

```
Highest Weighted average Centrality :
Betweenness -
   index  Weight_Mean      Mean  Activity
0     n1     0.621548  0.655051      11.0
25   n12     0.182715  0.167562       7.0
4     n3     0.136982  0.129403       8.0
22   n76     0.095690  0.083791       7.0
54   n87     0.073428  0.061327       5.0
84   n41     0.063277  0.050369       1.0
42   n14     0.041787  0.032671       3.0
2    n89     0.038035  0.047948       4.0
50   n82     0.034613  0.029196       3.0
3    n83     0.028829  0.031785       6.0
```

> ## Part (f) Question 2: (3 points) (~200 words, 300 word limit.)
>
> The change in the network from Phase X to X+1 coincides with a major event that took place during the actual investigation. Identify the event and explain how the change in centrality rankings and visual patterns, observed in the network plots above, relates to said event.

Solution:

The top ten betweenness centrality rankings for each phase are output, and a line chart of the change in funds and betweenness centrality of n1 is drawn. From the perspective of phase changes, when criminal funds are small, the network shown by visual patterns is sparse, and centrality is highly concentrated on one criminal. When criminal funds are large, the network shown by visual patterns is dense. Centrality is no longer highly concentrated on one criminal, centrality is diluted by many criminals. Taking criminal n1 as an example, its centrality is inversely related to criminal funds.

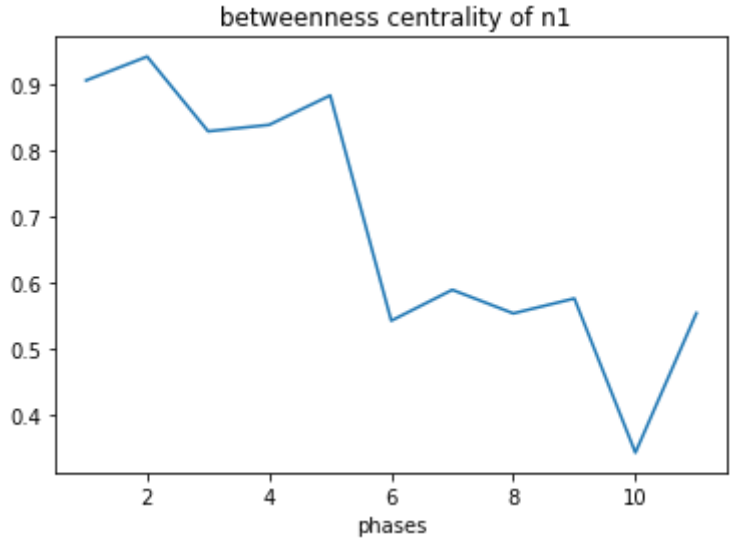
```
i = int(0)
for index, row in bet_df.iterrows():
    i+=1
    row_sort = row.sort_values(ascending = False).head(10)
    print("phase" + str(i))
    print(row_sort)
plt.plot(range(1,12), [ 0.906593, 0.942688, 0.829503, 0.839310, 0.883871,
0.542564, 0.589356, 0.553659, 0.576231, 0.342683, 0.554060 ], label="n1")
plt.xlabel('phases')
plt.title(' betweenness centrality of n1')
```

```
phase1
n1      0.906593
n89     0.142857
n88     0.053114
n83     0.036630
n85     0.036630
n6      0.010989
n49     0.000000
n107    0.000000
n84     0.000000
n109    0.000000
Name: 1, dtype: float64
phase2
n1     0.942688
n89    0.123847
n83    0.086957
n88    0.083004
n76    0.038208
n11    0.006588
n85    0.003294
n8     0.001976
n3     0.001976
n25    0.000000
Name: 2, dtype: float64
phase3
n1     0.829503
n3     0.095497
```

```
n9     0.067540
n5     0.062500
n83    0.046573
n6     0.033333
n85    0.029973
n86    0.028024
n49    0.007527
n2     0.001008
Name: 3, dtype: float64
phase4
n1      0.839310
n89     0.196213
n3      0.090438
n83     0.079589
n13     0.062500
n107    0.062500
n8      0.062500
n86     0.047427
n85     0.016537
n9      0.014761
Name: 4, dtype: float64
phase5
n1     0.883871
n12    0.269892
n83    0.064516
n89    0.064516
n85    0.064516
n31    0.064516
n3     0.044086
n4     0.000000
n63    0.000000
n49    0.000000
Name: 5, dtype: float64
phase6
n1      0.542564
n12     0.382051
n3      0.227179
n76     0.098462
n85     0.010256
n8      0.001026
n109    0.000000
n49     0.000000
n107    0.000000
n50     0.000000
Name: 6, dtype: float64
phase7
n1      0.589356
n76     0.134454
n3      0.073389
n85     0.031653
n12     0.016807
n19     0.011204
n87     0.001961
n5      0.000000
n109    0.000000
n49     0.000000
Name: 7, dtype: float64
phase8
```

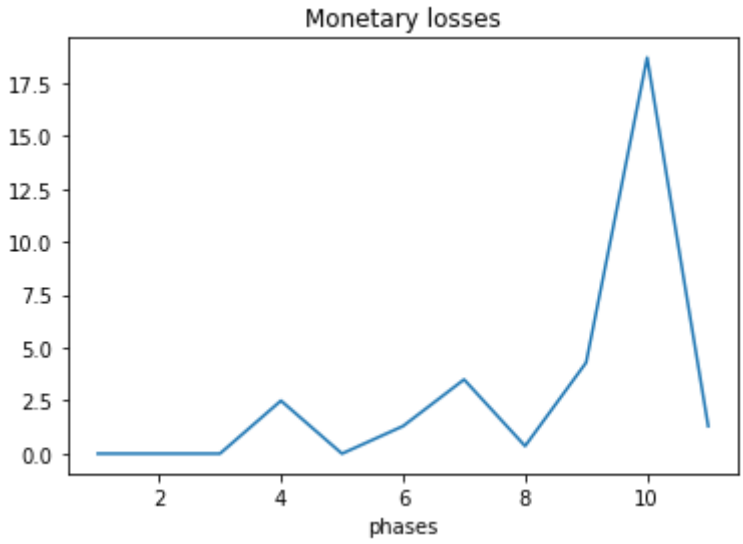
```
n1     0.553659
n12    0.356707
n3     0.314634
n14    0.263415
n87    0.177439
n76    0.118293
n22    0.076829
n2     0.055285
n11    0.048780
n20    0.006098
Name: 8, dtype: float64
phase9
n3     0.576231
n12    0.357323
n1     0.249053
n87    0.236269
n76    0.131944
n82    0.115688
n7     0.087753
n79    0.062500
n85    0.060606
n96    0.038984
Name: 9, dtype: float64
phase10
n1     0.342683
n87    0.188415
n37    0.174390
n82    0.068293
n76    0.068293
n83    0.035366
n71    0.035366
n8     0.035366
n12    0.030488
n14    0.003659
Name: 10, dtype: float64
phase11
n41    0.554060
n1     0.526282
n12    0.429915
n76    0.332051
n79    0.178846
n82    0.137179
n14    0.092308
n87    0.070513
n96    0.047436
n58    0.047436
Name: 11, dtype: float64
```

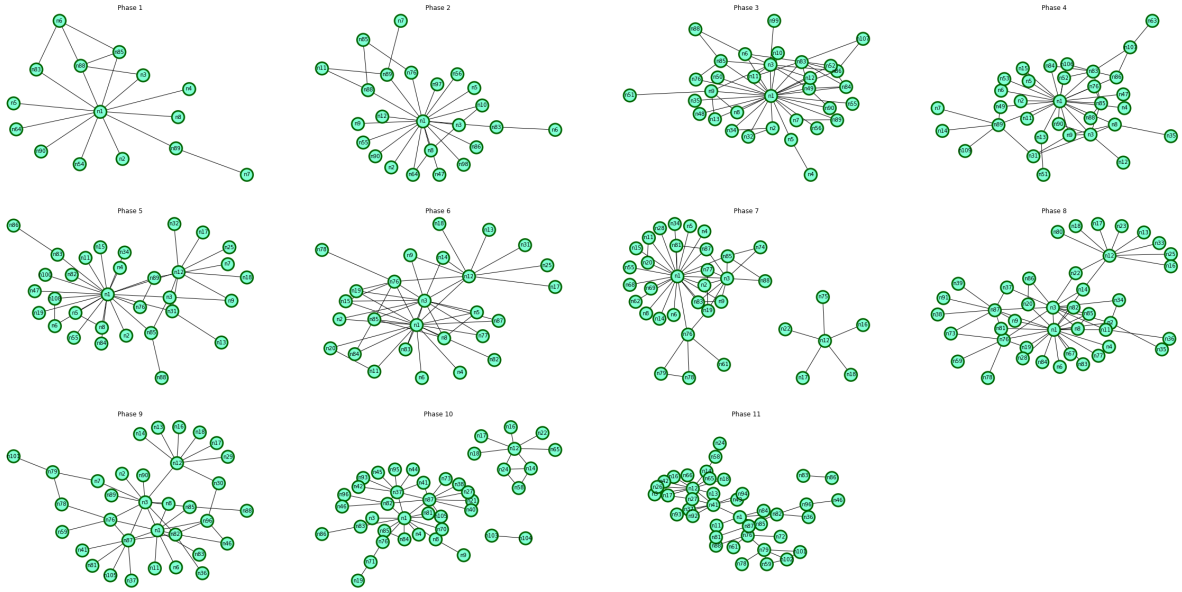
```
Text(0.5, 1.0, ' betweenness centrality of n1')
```




```
plt.plot(range(1,12), [ 0, 0, 0, 2.5, 0, 1.3, 3.5, 0.36, 4.3, 18.7, 1.3, ],
label="Monetary losses")
plt.xlabel('phases')
plt.title('Monetary losses')
```

```
Text(0.5, 1.0, 'Monetary losses')
```





> ## Part (g): (4 points) (~200 words, 300 word limit.)
>
> While centrality helps explain the evolution of every player's role individually, we need to explore the global trends and incidents in the story in order to understand the behavior of the criminal enterprise.
> Describe the coarse pattern(s) you observe as the network evolves through the phases. Does the network evolution reflect the background story?

> Hint: Look at the set of actors involved at each phase, and describe how the composition of the graph is changing. Investigate when important actors seem to change roles by their movement within the hierarchy. Correlate your observations with the information that the police provided in the setup to this homework problem.

Solution:

In this Part, I continue to analyze through the graphs in Part (f). In the early days, the criminal group was a very small organization. N1, as the Mastermind of the network, had connections with almost every other member. As crimes escalated and more criminals participated, many criminals who had been following n1 began to become network centers, such as n12 and n3. In the later phase of the phase, n1 contacted only a few criminals.

> ## Part (h): (2 points) (~50 words, 100 word limit.)
>
> Are there other actors that play an important role but are not on the list of investigation (i.e., actors who are not among the 23 listed above) ? List them, and explain why they are important.

Solution:

Yes, there are other actors that play an important role but are not on the list of investigation. Use the method in Part (e) solution, I sorted the betweenness centrality ranking that shown below and found the criminal list:

n76, \ n41, \ n79, \ n37, \ n8, \ n9

```
print(bet_statistics.sort_values(by=['Weight_Mean'], ascending =
False).head(20))
```

	index	Weight_Mean	Mean	Activity
0	n1	0.621548	0.655051	11.0
25	n12	0.182715	0.167562	7.0
4	n3	0.136982	0.129403	8.0
22	n76	0.095690	0.083791	7.0
54	n87	0.073428	0.061327	5.0
84	n41	0.063277	0.050369	1.0
42	n14	0.041787	0.032671	3.0
2	n89	0.038035	0.047948	4.0
50	n82	0.034613	0.029196	3.0
3	n83	0.028829	0.031785	6.0
59	n79	0.026345	0.021941	2.0
7	n85	0.022335	0.023733	10.0
72	n37	0.020524	0.015951	2.0
14	n8	0.010092	0.009170	4.0
78	n96	0.009110	0.007856	2.0
57	n22	0.008988	0.006984	1.0
10	n7	0.008311	0.007978	1.0
6	n88	0.007830	0.012435	3.0
23	n9	0.007565	0.007482	2.0
15	n86	0.007173	0.007044	3.0

> The remaining two questions will concern the directed graphs derived from the CAVIAR data.
>
> ## Part (i): (2 points) (~150 words, 250 word limit.)
>
> What are the advantages of looking at the directed version vs. undirected version of the criminal network?
>
> Hint: If we were to study the directed version of the graph, instead of the undirected, what would you learn from comparing the in-degree and out-degree centralities of each actor? Similarly, what would you learn from the left- and right-eigenvector centralities, respectively?

Solution:

Directed graphs are more informative than corresponding undirected graphs when the network is sparse. This means that if we treat a sparse directed graph as undirected we probably lose information
Directed graphs apply well to model relationships which are directional and not reciprocal in nature. A good example is a relationship “is a executes of”, upon which we construct criminal trees
Undirected graphs apply well to relationships for which it matters whether they exist or not, but aren’t intrinsically transitive.
We can model the same system as a directed graph in some circumstances and as an undirected graph in others. For example, we can represent a family as a directed graph if we’re interested in studying progeny. If we’re studying clan affiliations, though, we can represent it as an undirected graph

> ## Part (j): (4 points) (~300 words, 400 word limit)(~300 words, 400 word limit)
>
> Recall the definition of hubs and authorities. Compute the hub and authority score of each actor, and for each phase. (Remember to load the adjacency data again this time using create_using = nx.DiGraph().)

> With networkx you can use the nx.algorithms.link_analysis.hits function, set max_iter=1000000 for best results.

> Using this, what relevant observations can you make on how the relationship between n1 and n3 evolves over the phases. Can you make comparisons to your results in Part (g)?

Solution:

With the phases go on, the hub scores of n1 are decrease and hub scores of n3 are increase; the authority scores of n1 are increase and authority scores of n3 are decrease. That means n3 gradually becomes a new contact hub， and n1 trends to give instructions to few people.That increases the criminal concealment of n1.

```
# print(np.array(phases[1]))
# print(phases[1])
graph = {}
phases_values = []
n1_hub, n1_authority, n3_hub, n3_authority = [], [], [], []
j = int(0)
for i in range(1,12):
    j += 1
    phases_name = phases[i].columns.values.tolist()
    graph[i] = nx.from_numpy_matrix(np.array(phases[i]),
create_using=nx.DiGraph)
    v,w=nx.link_analysis.hits(graph[i], max_iter = 1000000)

    phases_v = v.values()
    phases_w = w.values()
    hub = dict(zip(phases_name, phases_v))
    authority = dict(zip(phases_name, phases_w))

    n1_hub.append(hub['n1'])
    n3_hub.append(hub['n3'])
    n1_authority.append(authority['n1'])
    n3_authority.append(authority['n3'])

    print('phase' + str(j))
    print('hubs score:',hub)
    print('authority score:',authority)
plt.plot(range(1,12), n1_hub, label="n1")
plt.plot(range(1,12), n3_hub, label="n3")
# plt.plot(range(1,12), [ G[i].number_of_edges() for i in range(1,12) ],
label="Edges")
plt.legend()
plt.title('hub scores')
plt.xlabel('phases')
plt.show()
plt.plot(range(1,12), n1_authority, label="n1")
plt.plot(range(1,12), n3_authority, label="n3")
```

```
plt.legend()
plt.title('authority scores')
plt.xlabel('phases')
plt.show()
```

```
phase1
hubs score: {'n1': 0.42165703018931017, 'n4': 0.0, 'n89': 0.0422972611685682,
'n83': 0.06687382179337698, 'n3': 0.09447758797853548, 'n5': 0.0, 'n88':
0.1599970764816512, 'n85': 0.1224400601019736, 'n90': 0.0, 'n2': 0.0, 'n7':
0.03669092397798762, 'n54': 0.0, 'n6': 0.05556623830859663, 'n64': 0.0, 'n8':
0.0}
authority score: {'n1': 0.0791014914689256, 'n4': 0.068616896990665, 'n89':
0.07458766747106635, 'n83': 0.0, 'n3': 0.09465346741716718, 'n5':
0.068616896990665, 'n88': 0.1129586190870015, 'n85': 0.09465346741716718, 'n90':
0.068616896990665, 'n2': 0.068616896990665, 'n7': 0.006883098422316634, 'n54':
0.068616896990665, 'n6': 0.05684390978170063, 'n64': 0.068616896990665, 'n8':
0.068616896990665}
phase2
hubs score: {'n1': 0.6730461441120974, 'n89': 0.00028546780996016434, 'n83': 0.0,
'n3': 0.08576021363054886, 'n5': 0.0, 'n88': 0.004480701669845821, 'n85':
0.08576021363054886, 'n90': 0.0, 'n86': 0.004211365637371194, 'n2':
0.004211365637371194, 'n7': 0.04031182054584639, 'n6': 0.04031182054584639,
'n64': 0.0, 'n8': 0.04477542423107923, 'n55': 0.004211365637371194, 'n10': 0.0,
'n56': 0.0, 'n97': 0.0, 'n47': 0.004211365637371194, 'n98': 0.004211365637371194,
'n76': 0.0, 'n9': 0.0, 'n11': 0.0, 'n12': 0.004211365637371194}
authority score: {'n1': 0.006215643548606932, 'n89': 0.05949706790712116, 'n83':
0.05949706790712116, 'n3': 0.05613489177256489, 'n5': 0.05613489177256489, 'n88':
0.06328765589119154, 'n85': 0.0, 'n90': 0.05613489177256489, 'n86': 0.0, 'n2':
0.05613489177256489, 'n7': 2.380923132528853e-05, 'n6': 0.0, 'n64':
0.05986935137284347, 'n8': 0.06328765589119154, 'n55': 0.05613489177256489,
'n10': 0.06328765589119154, 'n56': 0.05613489177256489, 'n97':
0.05613489177256489, 'n47': 0.05613489177256489, 'n98': 0.05613489177256489,
'n76': 0.06328765589119154, 'n9': 0.05613489177256489, 'n11':
0.00039751874256715053, 'n12': 0.0}
phase3
hubs score: {'n1': 0.3597120687924282, 'n4': 0.012299374971739078, 'n89':
0.012299374971739078, 'n83': 0.09233844935515416, 'n3': 0.0845795871570134, 'n5':
0.0, 'n48': 0.0, 'n88': 0.03292335250903222, 'n85': 0.0, 'n90': 0.0, 'n86':
0.05804878167384297, 'n2': 0.02546961406034494, 'n7': 0.0, 'n6':
0.0364355576283346, 'n8': 0.0, 'n55': 0.0, 'n10': 0.0, 'n56': 0.0, 'n76':
0.03362301501283534, 'n9': 0.029605321849396727, 'n34': 0.0, 'n35': 0.0, 'n11':
0.0364355576283346, 'n32': 0.0, 'n84': 0.03276258377388706, 'n49':
0.08557348396552755, 'n107': 0.0, 'n50': 0.016733780788621685, 'n99':
0.019701776839712918, 'n13': 0.015729159511027796, 'n51': 0.015729159511027796,
'n12': 0.0, 'n52': 0.0}
```

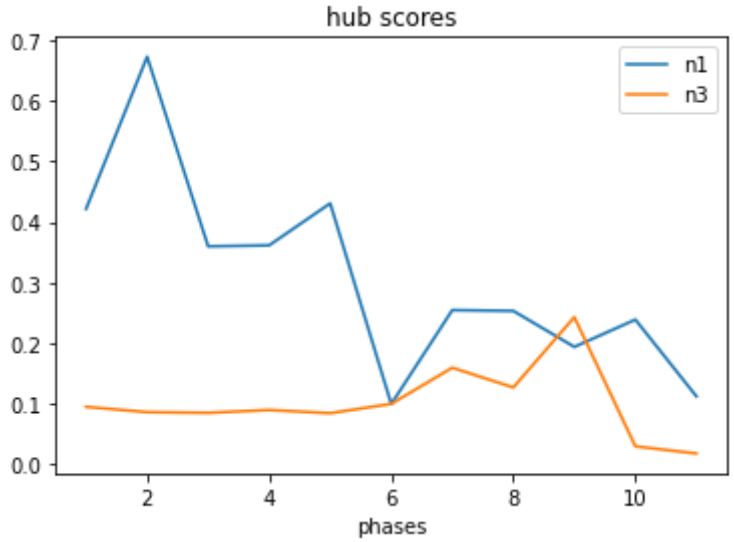
```
authority score: {'n1': 0.0432099146071527, 'n4': 0.0, 'n89':
0.03070938251970205, 'n83': 0.04297072013320254, 'n3': 0.05087386440678324, 'n5':
0.03175940625513424, 'n48': 0.03070938251970205, 'n88': 0.0, 'n85':
0.0436113259655401, 'n90': 0.03801497244318419, 'n86': 0.04581325072776831, 'n2':
0.03070938251970205, 'n7': 0.03175940625513424, 'n6': 0.04140324835371903, 'n8':
0.03323685167889426, 'n55': 0.03070938251970205, 'n10': 0.03070938251970205,
'n56': 0.03070938251970205, 'n76': 0.03070938251970205, 'n9':
0.04061578479478672, 'n34': 0.032883777507500406, 'n35': 0.03070938251970205,
'n11': 0.007883129373528294, 'n32': 0.032883777507500406, 'n84':
0.045898101816712486, 'n49': 0.03070938251970205, 'n107': 0.012838877063546647,
'n50': 0.03070938251970205, 'n99': 0.007220738834537977, 'n13':
0.03070938251970205, 'n51': 0.0, 'n12': 0.03793012135424002, 'n52':
0.04138952320441142}
phase4
hubs score: {'n1': 0.3613662713984412, 'n4': 0.0, 'n89': 0.0445756777840947,
'n83': 0.08937413309639898, 'n3': 0.08930239508755584, 'n5':
0.016715716522847174, 'n88': 0.02435239669834816, 'n85': 0.07298852434482311,
'n90': 0.03705604523082322, 'n86': 0.04100683493268685, 'n2':
0.016715716522847174, 'n7': 0.01559215040602386, 'n6': 0.016715716522847174,
'n8': 0.021240393195678504, 'n47': 0.0, 'n76': 0.02435239669834816, 'n9': 0.0,
'n35': 0.0, 'n11': 0.016715716522847174, 'n53': 0.016715716522847174, 'n84':
0.03561388029918218, 'n49': 0.0, 'n107': 5.124211443706376e-43, 'n13':
5.124211443706376e-43, 'n51': 0.0, 'n63': 0.0, 'n109': 0.0, 'n31':
0.02036182572904807, 'n12': 0.02034032870797604, 'n14': 0.0, 'n52':
0.01889816377633501, 'n106': 0.0, 'n15': 0.0}
authority score: {'n1': 0.04149745216565004, 'n4': 0.03710706586641343, 'n89':
0.03870815317723437, 'n83': 0.046915467025011856, 'n3': 0.05049570064430979,
'n5': 0.03710706586641343, 'n88': 0.04460192691077208, 'n85':
0.06045582405799907, 'n90': 0.03710706586641343, 'n86': 0.04628449083767879,
'n2': 0.03710706586641343, 'n7': 0.0045772745177094865, 'n6':
0.03710706586641343, 'n8': 0.04627712442309975, 'n47': 0.03710706586641343,
'n76': 0.04460192691077208, 'n9': 0.05054906788121054, 'n35':
0.0021810797862575264, 'n11': 0.03710706586641343, 'n53': 0.0, 'n84':
0.009177424971265362, 'n49': 0.04168434038412292, 'n107': 0.013388231167326715,
'n13': 0.03710706586641343, 'n51': 1.2721065171286044e-42, 'n63':
1.2721065171286044e-42, 'n109': 0.0045772745177094865, 'n31':
0.013747333074395813, 'n12': 0.009170058556686326, 'n14': 0.0045772745177094865,
'n52': 0.04628449083767879, 'n106': 0.04628449083767879, 'n15':
0.03710706586641343}
phase5
hubs score: {'n1': 0.4306980722887035, 'n4': 0.0, 'n89': 0.0, 'n83':
2.3220704067611161e-50, 'n3': 0.08400976896180219, 'n5': 0.0, 'n88':
0.023381705306676888, 'n85': 0.02893700197628893, 'n86': 0.023381705306676888,
'n2': 0.014276074957892966, 'n7': 0.025543423316820316, 'n6':
0.014276074957892966, 'n8': 0.023381705306676888, 'n55': 0.0, 'n47': 0.0, 'n76':
0.03981949827471328, 'n9': 0.0, 'n34': 0.0, 'n11': 0.014276074957892966, 'n32':
0.03134582071595375, 'n84': 0.014276074957892966, 'n13': 0.0, 'n31':
0.015051093478720547, 'n12': 0.07339324014441115, 'n108': 0.03843279878539743,
'n100': 0.014276074957892966, 'n18': 0.03134582071595375, 'n17':
0.03134582071595375, 'n25': 0.0, 'n82': 0.014276074957892966, 'n19':
0.014276074957892966, 'n15': 0.0}
```

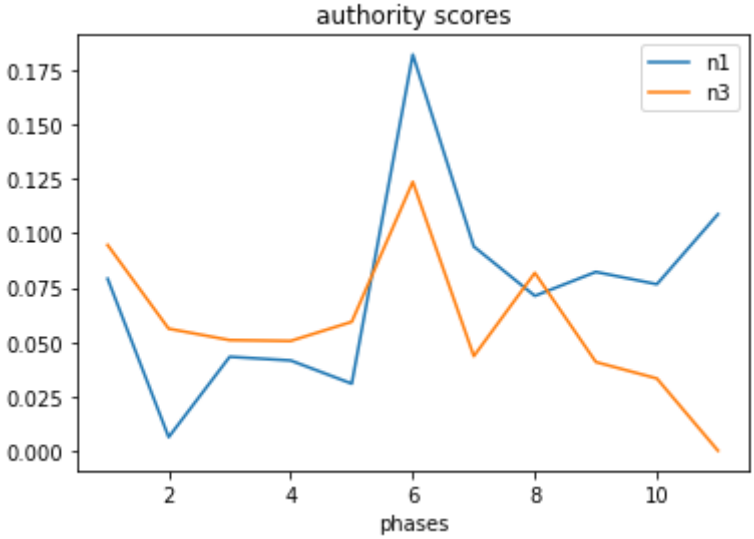
```
authority score: {'n1': 0.030840337441368233, 'n4': 0.04791011917055385, 'n89':
0.05518097914321404, 'n83': 0.05051106019962885, 'n3': 0.05929316082490011, 'n5':
0.05051106019962885, 'n88': 0.0032189027585769665, 'n85': 0.05051106019962885,
'n86': 5.01632522372935e-50, 'n2': 0.04791011917055385, 'n7': 0.0, 'n6':
0.05218531819954345, 'n8': 0.04791011917055385, 'n55': 0.04791011917055385,
'n47': 0.04791011917055385, 'n76': 0.04791011917055385, 'n9':
0.017509242379685794, 'n34': 0.04791011917055385, 'n11': 0.0, 'n32': 0.0, 'n84':
0.0, 'n13': 0.0016742579999145908, 'n31': 0.06541936155023964, 'n12':
0.06771578960659393, 'n108': 0.0, 'n100': 0.0, 'n18': 0.008164138895769292,
'n17': 0.0, 'n25': 0.008164138895769292, 'n82': 0.04791011917055385, 'n19':
0.04791011917055385, 'n15': 0.04791011917055385}
phase6
hubs score: {'n1': 0.09909912077318604, 'n4': 0.0, 'n83': 0.02945727135141846,
'n3': 0.099344049974698, 'n5': 0.06041909548102864, 'n85': 0.055340639481568143,
'n2': 0.04945323665945061, 'n6': 0.02945727135141846, 'n8': 0.02945727135141846,
'n76': 0.05688230076032345, 'n9': 0.0, 'n11': 0.03052252469870637, 'n84':
0.03769875799921189, 'n13': 0.008389827245176106, 'n31': 0.0, 'n12':
0.0943568159339083, 'n14': 0.008389827245176106, 'n18': 0.008389827245176106,
'n17': 0.008389827245176106, 'n25': 0.008389827245176106, 'n82':
0.02945727135141846, 'n19': 0.04945323665945061, 'n78': 0.014917509308571665,
'n77': 0.04945323665945061, 'n87': 0.04945323665945061, 'n20':
0.02945727135141846, 'n15': 0.06437074596802227}
authority score: {'n1': 0.18229305920214234, 'n4': 0.021403257295414747, 'n83':
0.04285941398694522, 'n3': 0.12374281528711809, 'n5': 0.021403257295414747,
'n85': 0.05100152676997799, 'n2': 0.021403257295414747, 'n6': 0.0, 'n8':
0.06786100204315083, 'n76': 0.09231535314641483, 'n9': 0.041835179183163755,
'n11': 0.0, 'n84': 0.0, 'n13': 0.02037902249163329, 'n31': 0.02037902249163329,
'n12': 0.051919516117265516, 'n14': 0.041835179183163755, 'n18':
0.02037902249163329, 'n17': 0.02037902249163329, 'n25': 0.02037902249163329,
'n82': 0.021403257295414747, 'n19': 0.033688598584617535, 'n78':
0.012285341289202787, 'n77': 0.04285941398694522, 'n87': 0.0, 'n20':
0.00659220228465203, 'n15': 0.021403257295414747}
phase7
hubs score: {'n1': 0.2543338088420863, 'n4': 0.0, 'n83': 0.018718130216878043,
'n3': 0.1591585819214289, 'n5': 0.0, 'n88': 0.0, 'n85': 0.037526798957060646,
'n62': 0.0, 'n9': 0.018718130216878043, 'n2': 0.0290087680502963, 'n6':
0.0290087680502963, 'n8': 0.0, 'n55': 0.0290087680502963, 'n76':
0.06048998494980722, 'n34': 0.0290087680502963, 'n11': 0.030321947959427192,
'n12': 1.1402684900008986e-61, 'n14': 0.0, 'n18': 6.481675939090216e-31, 'n17':
6.481675939090216e-31, 'n19': 0.04260240309070448, 'n78': 0.016947280427993237,
'n77': 0.051778361742468026, 'n87': 0.055878013489856417, 'n20':
0.0290087680502963, 'n22': 6.481675939090216e-31, 'n74': 0.018718130216878043,
'n79': 0.00273828356616279, 'n69': 0.0290087680502963, 'n61': 0.0, 'n68':
0.0290087680502963, 'n16': 6.481675939090216e-31, 'n75': 6.481675939090216e-31,
'n28': 0.0, 'n81': 0.0, 'n15': 0.0290087680502963}
authority score: {'n1': 0.09394693267689948, 'n4': 0.03567176058087149, 'n83':
0.060619979320511594, 'n3': 0.043508969219457405, 'n5': 0.03567176058087149,
'n88': 0.027586241330554012, 'n85': 0.060619979320511594, 'n62':
0.03567176058087149, 'n9': 0.060619979320511594, 'n2': 0.057994655291392756,
'n6': 0.03567176058087149, 'n8': 0.03567176058087149, 'n55': 0.0, 'n76':
0.04402394178757703, 'n34': 0.0, 'n11': 0.03567176058087149, 'n12':
2.0991362750303628e-30, 'n14': 0.03567176058087149, 'n18': 0.0, 'n17': 0.0,
'n19': 0.07374092832794266, 'n78': 0.008868123644358907, 'n77':
0.057994655291392756, 'n87': 0.03567176058087149, 'n20': 0.004252825363072142,
'n22': 3.6928395882916526e-61, 'n74': 0.022322894710521263, 'n79':
0.010861016121173399, 'n69': 0.0, 'n61': 0.0084840638268226, 'n68': 0.0, 'n16':
0.0, 'n75': 0.0, 'n28': 0.03567176058087149, 'n81': 0.043508969219457405, 'n15':
0.0}
```

```
phase8
hubs score: {'n1': 0.25302057872067973, 'n4': 0.02357629983393216, 'n83': 0.0,
'n3': 0.1265880329019276, 'n85': 0.06863916917529844, 'n86':
0.010573100575364537, 'n2': 0.06059832172874063, 'n6': 0.02357629983393216, 'n8':
0.0, 'n76': 0.07413575835905538, 'n9': 0.0, 'n34': 0.010573100575364537, 'n35':
0.0, 'n11': 0.028214774838179264, 'n84': 0.0, 'n13': 0.0, 'n12':
0.01813654257057493, 'n14': 0.05365151916989436, 'n18': 0.0029921774493100774,
'n17': 0.0029921774493100774, 'n25': 0.0029921774493100774, 'n82': 0.0, 'n19':
0.02357629983393216, 'n78': 0.010573100575364537, 'n77': 0.0, 'n87':
0.10985316062812676, 'n20': 0.0, 'n22': 0.027083041886652125, 'n16':
0.0029921774493100774, 'n28': 0.02357629983393216, 'n81': 0.0, 'n67': 0.0, 'n38':
0.01655467840693892, 'n73': 0.0, 'n39': 0.0, 'n59': 0.0, 'n37':
0.01655467840693892, 'n23': 0.0029921774493100774, 'n33': 0.0029921774493100774,
'n80': 0.0029921774493100774, 'n91': 0.0, 'n36': 0.0}
authority score: {'n1': 0.07123252371921959, 'n4': 0.03066379489617524, 'n83':
0.03066379489617524, 'n3': 0.08182765901216507, 'n85': 0.0460051141692636, 'n86':
0.015341319273088364, 'n2': 0.031945158605269364, 'n6': 0.0, 'n8':
0.054323557753201246, 'n76': 0.031945158605269364, 'n9': 0.043976999736588036,
'n34': 0.02268528521737617, 'n35': 0.007343965944287804, 'n11':
0.03066379489617524, 'n84': 0.03066379489617524, 'n13': 0.002197984148852161,
'n12': 0.0090404449630833593, 'n14': 0.04820309831811577, 'n18': 0.0, 'n17': 0.0,
'n25': 0.0, 'n82': 0.0460051141692636, 'n19': 0.039648375144585274, 'n78': 0.0,
'n77': 0.03066379489617524, 'n87': 0.050017667343588895, 'n20':
0.031945158605269364, 'n22': 0.002197984148852161, 'n16': 0.0, 'n28':
0.03066379489617524, 'n81': 0.043976999736588036, 'n67': 0.03066379489617524,
'n38': 0.013313204840412799, 'n73': 0.02229778508882283, 'n39':
0.013313204840412799, 'n59': 0.008984580248410033, 'n37': 0.02865452411350116,
'n23': 0.002197984148852161, 'n33': 0.0, 'n80': 0.0, 'n91': 0.013313204840412799,
'n36': 0.0034193743242717264}
phase9
hubs score: {'n1': 0.19356100463852824, 'n89': 0.0, 'n83': 0.0, 'n3':
0.24278577230551454, 'n88': 0.025712720981697617, 'n85': 0.043222252617314924,
'n90': 0.0, 'n2': 0.0, 'n7': 1.6416136874345677e-40, 'n6': 0.0, 'n8':
0.043222252617314924, 'n76': 0.05137943814234242, 'n11': 0.0, 'n13': 0.0, 'n12':
0.022104751115483346, 'n14': 0.0, 'n18': 0.01737898975430699, 'n17':
0.01737898975430699, 'n82': 0.0925765052791319, 'n78': 0.025712720981697617,
'n87': 0.09748579859935619, 'n79': 3.5253385500987172e-31, 'n16':
0.01737898975430699, 'n81': 0.03419565063569325, 'n59': 0.0, 'n37': 0.0, 'n36':
0.0, 'n96': 0.007181009752326349, 'n46': 0.051344163316370574, 'n29':
0.01737898975430699, 'n30': 0.0, 'n105': 0.0, 'n101': 0.0, 'n41': 0.0}
authority score: {'n1': 0.08227168322319874, 'n89': 0.03846462444432177, 'n83':
0.030665929377131376, 'n3': 0.040782257343476895, 'n88': 0.0, 'n85':
0.07320422809758431, 'n90': 0.03846462444432177, 'n2': 0.03846462444432177, 'n7':
0.03846462444432177, 'n6': 0.030665929377131376, 'n8': 0.06913055382145314,
'n76': 0.07320422809758431, 'n11': 0.030665929377131376, 'n13':
0.0035020625529202693, 'n12': 0.04947805916711172, 'n14': 0.0035020625529202693,
'n18': 0.0, 'n17': 0.0035020625529202693, 'n82': 0.09270973043208566, 'n78':
5.018327067102863e-31, 'n87': 0.0973551655953652, 'n79': 4.673681284396781e-40,
'n16': 0.0035020625529202693, 'n81': 0.015444705051783355, 'n59':
0.008140060178366115, 'n37': 0.015444705051783355, 'n36': 0.014666924166686445,
'n96': 0.053467325102667, 'n46': 0.015804613753885394, 'n29':
0.0035020625529202693, 'n30': 0.0046397521401192144, 'n105':
0.015444705051783355, 'n101': 5.018327067102863e-31, 'n41':
0.015444705051783355}
phase10
```

```
hubs score: {'n1': 0.23865332555095556, 'n4': 0.0, 'n83': 0.03085665832332989,
'n3': 0.02921854510355311, 'n85': 0.04512853711191959, 'n86': 0.0, 'n8':
0.03085665832332989, 'n76': 0.02921854510355311, 'n9': 0.0, 'n84':
0.013379898437434848, 'n12': 8.870141645467959e-14, 'n14': 3.135862425825979e-14,
'n18': 0.0, 'n17': 0.0, 'n82': 0.03501393101217885, 'n19': 4.0285956261383554e-
41, 'n87': 0.21833942990305702, 'n22': 0.0, 'n16': 5.686905320065674e-15, 'n81':
0.020766287808134613, 'n38': 0.0, 'n73': 0.0, 'n37': 0.13174886124308113, 'n24':
1.9984813618128442e-14, 'n58': 4.1483273357022985e-14, 'n96':
0.03520155077347411, 'n46': 0.03913812285111873, 'n105': 0.0, 'n41':
0.02885655214877346, 'n21': 0.0, 'n103': 0.0, 'n104': 4.0285956261383554e-41,
'n27': 0.0, 'n95': 0.0, 'n93': 0.0, 'n42': 0.0, 'n71': 0.01590999200836648,
'n40': 0.0, 'n44': 0.02885655214877346, 'n45': 0.02885655214877346, 'n65':
5.686905320065674e-15, 'n70': 0.0}
authority score: {'n1': 0.07659625451299369, 'n4': 0.03321325645848575, 'n83':
0.03321325645848575, 'n3': 0.03321325645848575, 'n85': 0.03507532980987269,
'n86': 0.004294304752629993, 'n8': 0.03321325645848575, 'n76':
0.04170795612353544, 'n9': 0.004294304752629993, 'n84': 0.03321325645848575,
'n12': 1.490819086793195e-14, 'n14': 5.239007845406926e-14, 'n18':
3.0945525294988516e-14, 'n17': 3.0945525294988516e-14, 'n82':
0.09228067081167601, 'n19': 0.0, 'n87': 0.05443870872436184, 'n22':
3.0945525294988516e-14, 'n16': 3.0945525294988516e-14, 'n81':
0.06359943890694011, 'n38': 0.030386182448454355, 'n73': 0.030386182448454355,
'n37': 0.07564729198258896, 'n24': 5.635809339450881e-14, 'n58':
1.0940175927492404e-14, 'n96': 0.010319701946122736, 'n46': 0.004872870185202636,
'n105': 0.06359943890694011, 'n41': 0.0487216014769757, 'n21':
0.030386182448454355, 'n103': 1.0560941169931917e-40, 'n104': 0.0, 'n27':
0.030386182448454355, 'n95': 0.01833541902852135, 'n93': 0.01833541902852135,
'n42': 0.01833541902852135, 'n71': 1.0560941169931917e-40, 'n40':
0.030386182448454355, 'n44': 0.01833541902852135, 'n45': 0.0, 'n65': 0.0, 'n70':
0.03321325645848575}
phase11
hubs score: {'n1': 0.11180235490040877, 'n83': 0.0, 'n3': 0.017407428145584398,
'n88': 0.0, 'n85': 0.05546864930649665, 'n86': 0.0, 'n76': 0.13632374669215083,
'n11': 0.03558046531354339, 'n84': 0.06053525170654886, 'n13': 0.0, 'n12':
0.060021746164025 65, 'n14': 0.01893858287966469, 'n18': 0.017407428145584398,
'n17': 0.017407428145584398, 'n82': 0.073457952283916, 'n78':
0.016724245077697018, 'n87': 0.07872243725463113, 'n79': 0.003958554798140518,
'n61': 0.0, 'n16': 0.017407428145584398, 'n81': 0.0, 'n59': 0.016724245077697018,
'n37': 0.03677957561717103, 'n36': 0.0, 'n24': 0.0, 'n58': 0.00578867460814663,
'n96': 0.01837477400789722, 'n46': 0.0, 'n101': 0.018543501002840447, 'n41':
0.0677081705755356, 'n27': 0.02421808654907531, 'n93': 0.02230934568113811,
'n42': 0.017407428145584398, 'n43': 0.015031570626928432, 'n102':
0.018543501002840447, 'n72': 0.0, 'n26': 0.0, 'n94': 0.0, 'n92': 0.0, 'n66': 0.0,
'n65': 0.017407428145584398}
```

```
authority score: {'n1': 0.10882195417909653, 'n83': 0.0, 'n3': 0.0, 'n88':
0.03370916629210287, 'n85': 0.07632358367092806, 'n86': 0.0, 'n76':
0.06082750824404731, 'n11': 0.027645690976857703, 'n84': 0.0, 'n13':
0.031584145282943614, 'n12': 0.05324017916408891, 'n14': 0.016273132748109372,
'n18': 0.014841750464886286, 'n17': 0.020830226647768277, 'n82':
0.05165519864157247, 'n78': 0.0009788432697482943, 'n87': 0.07951901713915713,
'n79': 0.05115068101238047, 'n61': 0.03370916629210287, 'n16':
0.014841750464886286, 'n81': 0.01946592426344746, 'n59': 0.00556414828140683,
'n37': 0.02225889112173911, 'n36': 0.01816415987019656, 'n24':
0.0014313822832230875, 'n58': 0.0046829980677562544, 'n96': 0.01816415987019656,
'n46': 0.004543583401267301, 'n101': 0.0, 'n41': 0.045973678972119, 'n27':
0.04067873711928 72, 'n93': 0.02583698665440091, 'n42': 0.0, 'n43': 0.0, 'n102':
0.00556414828140683, 'n72': 0.03370916629210287, 'n26': 0.014841750464886286,
'n94': 0.016742394818057327, 'n92': 0.016742394818057327, 'n66':
0.014841750464886286, 'n65': 0.014841750464886286}
```





Problem 3: Co-offending Network

> ### Part (g): (3 points) (~50 words, 100 word limit.)
>
> Plot the degree distribution (or an approximation of it if needed) of G. Comment on the shape of the distribution. Could this graph have come from an Erdos-Renyi model? Why might the degree distribution have this shape?

Solution:

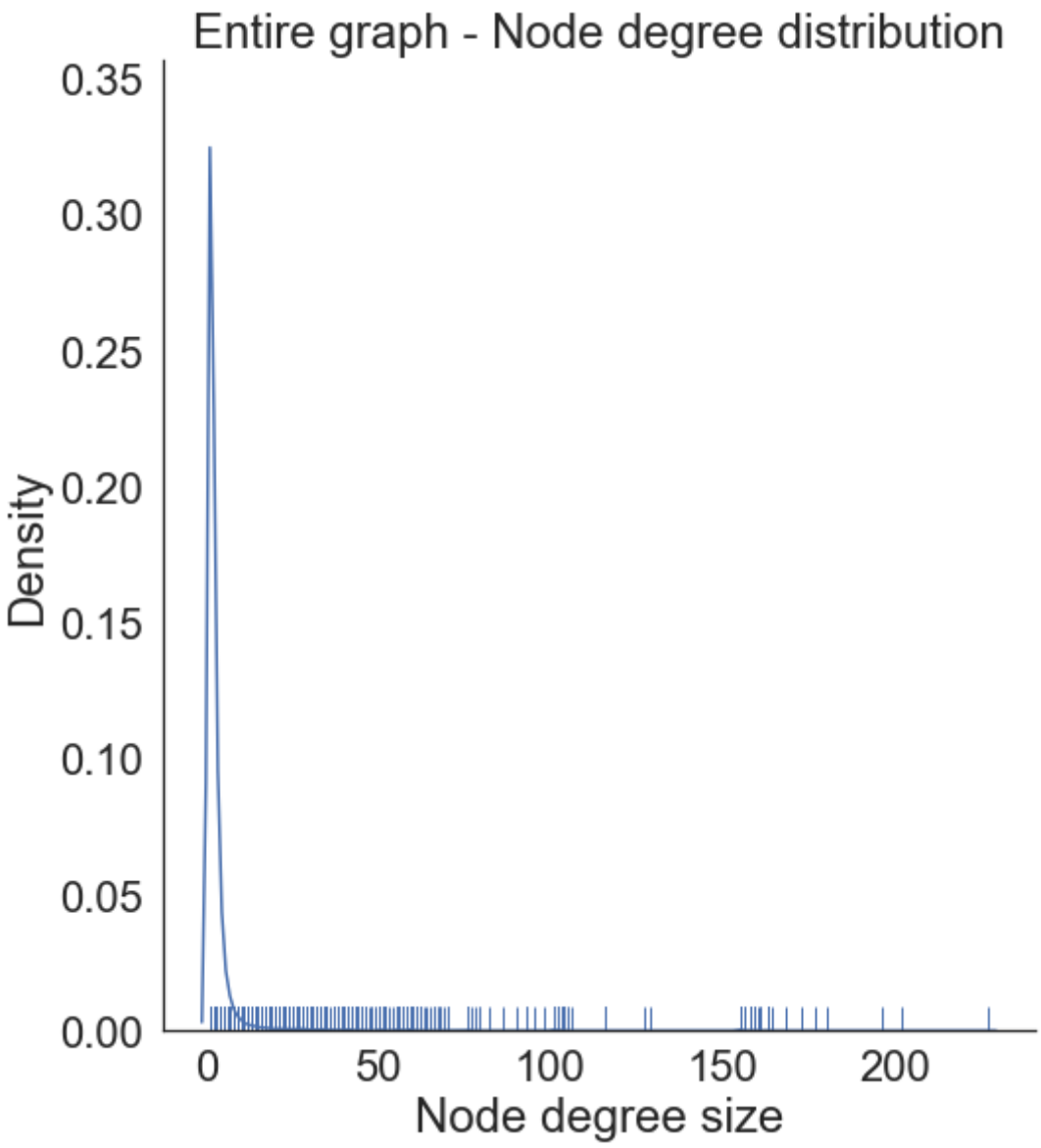
The distribution is a power law distributions. Its key characteristics are as follows:

1. Distribution is right skewed;
2. High ratio of max to min;
3. No matter what scale you look at it, it looks the same.

Because each node has the possibility(p) of connecting with other points. Large node degree size means more nodes connected, that will results in small frequency.

```
import numpy as np
import pandas as pd
import networkx as nx
from scipy.sparse import csr_matrix
import matplotlib
import matplotlib.pyplot as plt
import collections
import scipy
import statistics as stats
import seaborn as sns

df = pd.read_csv("Cooffending.csv")
df.CrimeDate = pd.to_datetime(df.CrimeDate)
df["NOffenders"] = df.NumberYouthOffenders + df.NumberAdultOffenders
n_cases_raw = len(df)
df = df.drop_duplicates()
# There are five people who are listed as both M and F, just drop one of them
df = df.drop_duplicates(subset=["OffenderIdentifier", "CrimeIdentifier"])
n_cases = len(df)
n_criminals = len(df.OffenderIdentifier.unique())
n_crimes = len(df.CrimeIdentifier.unique())
df.head()

# Remap to consecutive identifiers
OffenderIdentifier_dict = {OffenderIdentifier: i for i, OffenderIdentifier in
enumerate(df.OffenderIdentifier.unique())}
CrimeIdentifier_dict = {CrimeIdentifier: i for i, CrimeIdentifier in
enumerate(df.CrimeIdentifier.unique())}

# .replace has a lot of overhead
df.OffenderIdentifier = df.OffenderIdentifier.map(OffenderIdentifier_dict.get)
df.CrimeIdentifier = df.CrimeIdentifier.map(CrimeIdentifier_dict.get)

assert not df.OffenderIdentifier.isnull().any() and not
df.CrimeIdentifier.isnull().any()
assert df.CrimeIdentifier.max() == df.CrimeIdentifier.nunique() - 1
assert df.OffenderIdentifier.max() == df.OffenderIdentifier.nunique() - 1

# Build matrix
row = df.OffenderIdentifier
```

```
col = df.CrimeIdentifier
vals = np.ones(len(row))

# Sparse representation
crime_matrix = csr_matrix((vals, (row, col)), shape=(row.max() + 1, col.max() +
1))

cooffend_matrix = crime_matrix @ crime_matrix.T

# Save an unmodified copy for later
cooffend_matrix_raw = cooffend_matrix.copy()

# convert to (binary) adj. matrix
# Could use the full cooffending matrix for project
cooffend_matrix[cooffend_matrix > 0] = 1
cooffend_matrix.setdiag(0)
cooffend_matrix.eliminate_zeros() # To avoid self loops since setdiag(0) does
not itself change the sparsity pattern

g = nx.from_scipy_sparse_matrix(cooffend_matrix)

g_removed = g.copy()
g_removed.remove_nodes_from(list(nx.isolates(g_removed)))
G3 = g_removed
# print(G3)

degree_sequence = sorted([d for n, d in G3.degree()], reverse=True)
sns.set(font_scale=2)
sns.set_style("white")
ax = sns.displot(data = degree_sequence,  kind="kde", rug = True, height = 8)
ax.set(xlabel="Node degree size",ylabel= "Density",title='Entire graph - Node
degree distribution')
plt.show()
```



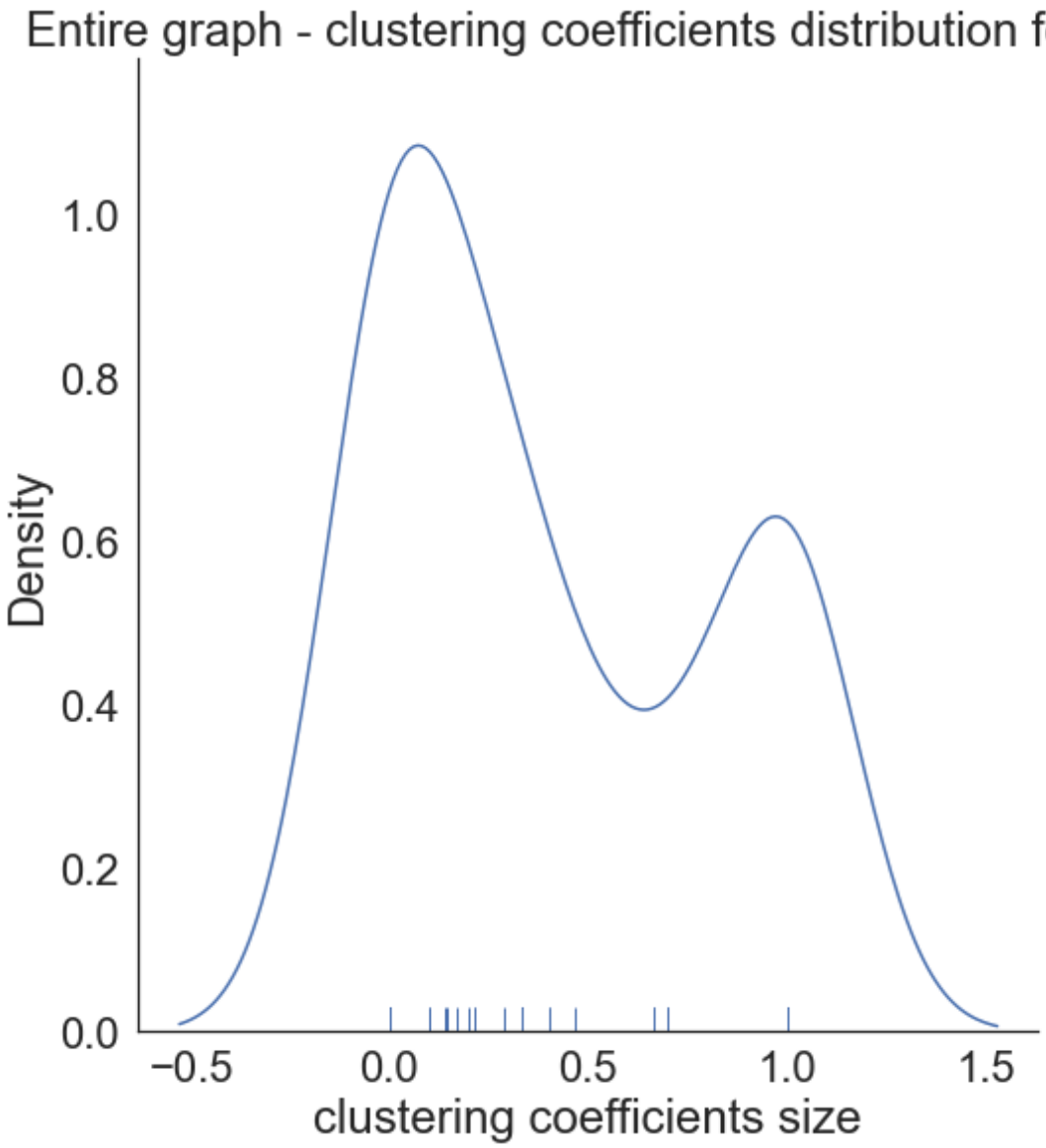
> ### Part (m): (4 points) (~300 words, 400 word limit.)
>
> Plot the distribution of clustering coefficients for each node for Gr and Gnr. What shape do the plots make? What does this tell you about the behavior of the actors? Hint: What does it mean for an actor to have a clustering coefficient of 0.5? Are there as many actors with intermediate clustering coefficients (say, between 0.25 and 0.75) as you expect for each graph?

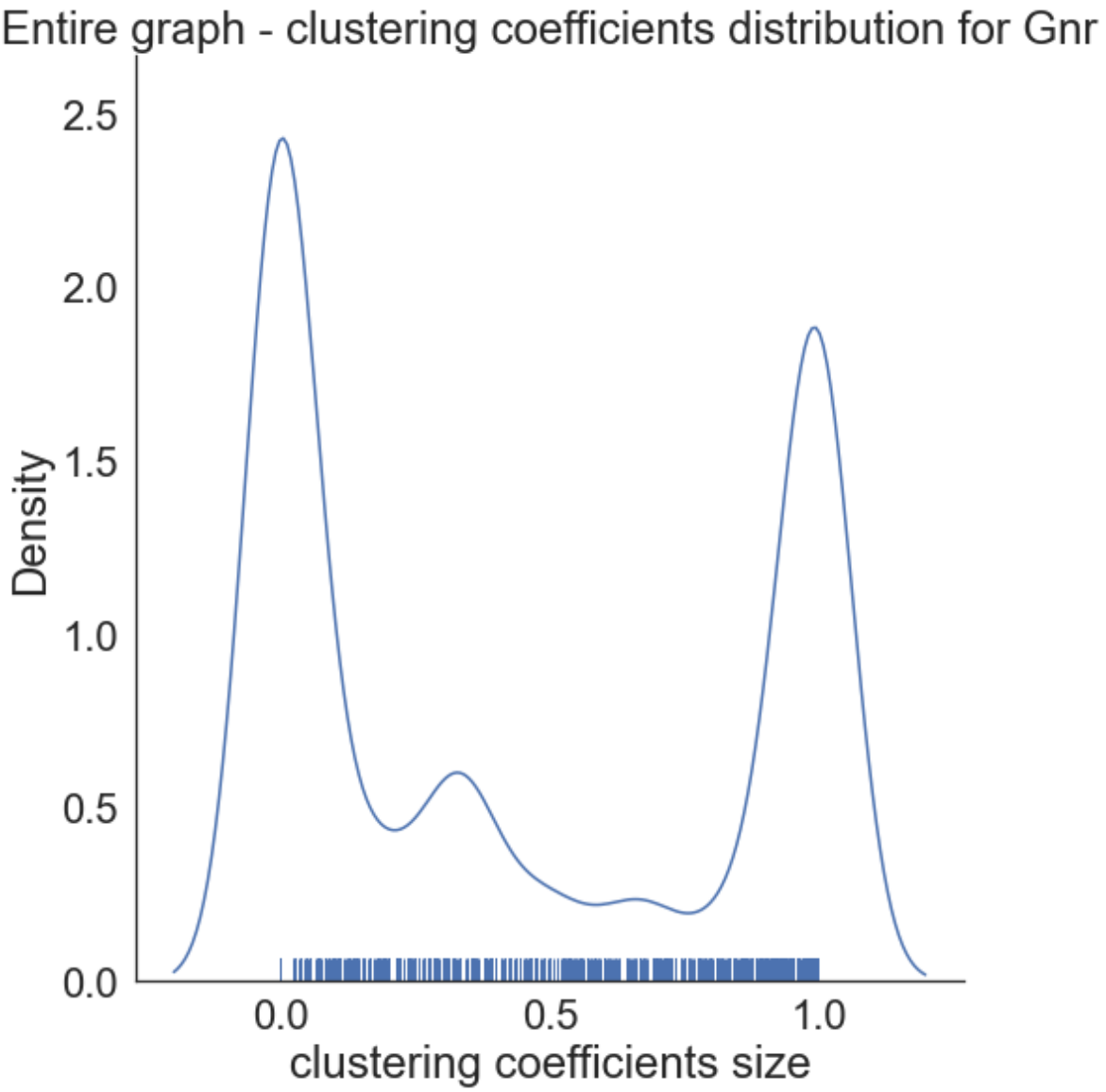
Solution:

The distribution graph shows that most of the clustering coefficients are clustered near 0 and 1. This means that the members of the criminal gang either all commit crimes together, or only cooperate with a few members to commit crimes.

```
# Separate matrices for repeating co-offenders vs non-repeating co-offenders
cooffend_matrix_repeat = cooffend_matrix_raw.copy()
cooffend_matrix_no_repeat = cooffend_matrix_raw.copy()

# Repeating co-offenders: edge strength >= 2
cooffend_matrix_repeat.data[np.where(cooffend_matrix_repeat.data<2)[0]]=0
cooffend_matrix_repeat[cooffend_matrix_repeat > 0] = 1
cooffend_matrix_repeat.setdiag(0)
```

```
cooffend_matrix_repeat.eliminate_zeros() # To avoid self loops since setdiag(0)
does not itself change the sparsity pattern

# Non-repeating co-offenders: edge strength = 1
cooffend_matrix_no_repeat.data[np.where(cooffend_matrix_no_repeat.data!=1)[0]]=0
cooffend_matrix_no_repeat[cooffend_matrix_no_repeat > 0] = 1
cooffend_matrix_no_repeat.setdiag(0)
cooffend_matrix_no_repeat.eliminate_zeros() # To avoid self loops since
setdiag(0) does not itself change the sparsity pattern

g_r = nx.from_scipy_sparse_matrix(cooffend_matrix_repeat)
g_nr = nx.from_scipy_sparse_matrix(cooffend_matrix_no_repeat)

g_r.remove_nodes_from(list(nx.isolates(g_r)))
g_nr.remove_nodes_from(list(nx.isolates(g_nr)))

g_r_component_list = sorted(nx.connected_components(g_r), key=len, reverse=True)
g_nr_component_list = sorted(nx.connected_components(g_nr), key=len,
reverse=True)

g_r_largest_component = g_r.subgraph(g_r_component_list[0])
g_nr_largest_component = g_nr.subgraph(g_nr_component_list[0])

g_r_largest_component_clustering =
list(nx.clustering(g_r_largest_component).values())
g_nr_largest_component_clustering =
list(nx.clustering(g_nr_largest_component).values())

# g_r_largest_component_clustering_sequence = sorted([d for d in
g_r_largest_component_clustering], reverse=True)
# g_nr_largest_component_clustering_sequence = sorted([d for d in
g_nr_largest_component_clustering], reverse=True)


sns.set(font_scale=2)
sns.set_style("white")
ax = sns.displot(data = g_r_largest_component_clustering,  kind="kde", rug =
True, height = 8)
ax.set(xlabel="clustering coefficients size",ylabel= "Density",title='Entire
graph - clustering coefficients distribution for Gr')
plt.show()


ax = sns.displot(data = g_nr_largest_component_clustering,  kind="kde", rug =
True, height = 8)
ax.set(xlabel="clustering coefficients size",ylabel= "Density",title='Entire
graph - clustering coefficients distribution for Gnr')
plt.show()
```


Entire graph - clustering coefficients distribution for Gr
Density
1.0
0.8
0.6
0.4
0.2
0.0
−0.5
0.0
0.5
1.0
1.5
clustering coefficients size



> ### Part (n): (4 points) (~200 words, 300 word limit.)
>
> Pick a centrality measure (degree, eigenvector, betweenness, etc) and compute the scores for the top (largest) component of Gr and Gnr. Compare the distribution of the centrality across nodes (for example, with summary statistics and/or a histogram). Examine the number of crimes committed by the most central actor in the repeat offender graph, does this support your conclusions?.

Solution

It's a pity that my computer is too slow, and this problem can't be calculated normally.

```
bet_gr = nx.degree_centrality(g_r_largest_component, normalized = True)
bet_gn = nx.degree_centrality(g_nr_largest_component, normalized = True)
print(bet_gr)
```

Project:

The last part of this assignment is an open-ended project. Choose a sociologically interesting question about the co-offending network, and try to answer your question using the data. You can subset the data in whichever way you desire as long as it is sociologically meaningful. For example, you can group nodes by attributes such as sex, group edges such as repeat/non-repeating cooffenses, use the weighted or unweighted co-offending networks, focus on the largest connected component, etc.

Project expectations/Rubric:

1. Clearly states a sociological question which is interesting and relevant to the data. The question must be sociologically motivated: for example, “Compare the network structure in 2003 vs 2009" is not a good question, without further context. If you have some reason to believe that the network structure changes in those years, then you should make that your central question: for example, “Did crimes involving youth offenders become more organized and structured over the years" is a better question, from which comparing the structure in different years becomes part of the methodology to answer the question. More examples of possible questions are provided below.

2. (2 points) Describes methodology for network analysis.

   (2 points) Grader is convinced that the methodology makes sense for the question to be answered.

   (1 point) Grader is convinced that no additional methodology within the bounds of techniques taught and discussed in this module could be applied beyond what was described. Additional clarification (April 13): The grader should only consider additional methodology that adds meaningfully to the answer for the question: additions that simply repeat or confirm the presented results should not be considered by the grader. If a justification is provided for why a particular method was not used, the grader should be convinced by that argument.

3. (2 points) Presents results, including figures and/or statistics, which address the question of interest.

   (2 points) The described methodology has been applied in complete and the results shown (that is, the author did not forget to include anything they discussed in the methodology.)

4.Adequately discusses the results obtained.

   (2 points) Question does not need to be successfully answered, but the grader should be convinced that the author has answered the question to the best ability of the methodology presented.

   (3 points) Provides commentary on what was discovered, what were the limitations of the methods, what may have been surprising to discover, etc.

   (1 point) Award this point if the question was successfully answered to the grader's satisfaction.

Solution

In this project, I will study the network containing juvenile delinquency. The purpose of this project is to study the network structure of youth criminals.

> Degree Centrality counts how many direct connections you have (sometimes, for example in a money network, it makes sense to differentiate between in-degree - how many people give you money, and out-degree - how many do you give to).

> Betweenness Centrality measures how often you sit on the shortest path between two others. This is seen as a measure of control. Let's think about the money network again. If you have to give money to someone, but it has to go through me and you have no direct connection to the other person, I can basically just run off with the money and tell the other person I never got it...

> Closeness Centrality: How many steps do you have to take to reach everyone in the network (directly or indirectly). This is a measure of access. Some people say that control (betweenness) and access (closeness) together are a measure of power.

> Eigenvector Centrality: Are you connected to actors who are well connected. A simple example is someone who is faithful but their partner is not. When it comes to their risk of getting HIV/Aids, the faithful person (just one link into the system) is nearly at the same risk as the unfaithful one (many links into the system), because he or she has a high eigenvector centrality.

After separating the data containing juvenile crimes from the original documents, the percentage of youth criminal is 13035/539593.

Among them, 563 criminal have repeated cooperative crimes more than 2 times, and 6329 have repeated cooperative crimes not more than once.

Draw the Node degree distribution, we can see that the number of single-person crimes in youth criminal is only about half of total crimes, and they are more inclined to cooperative crimes. It can be seen from the clustering distribution graph that most teenagers who repeat crimes more than twice commit single-person crimes. Among the teenagers who repeat the crime once, single-person crime and group crime each account for half of them.

As can be seen from the largest component clustering distribution graph, Almost everyone in the criminal group is involved, and it is rare for individual to do it alone.

From these analyses, we can roughly summerize some conclusions as follows:

1. The personal criminal ability of young people is not as good as that of adults, so young people use collective crimes to make up for this shortcoming.
2. Youth criminal gangs are very united, so dismantling the gang will help reduce crime
   3 Teenagers are more likely to commit crimes impulsively. Many people commit a crime once and do not repeat it. These young people tend to repent after committing a crime. The government may reduce crime by strengthening education.
3. Teenagers who repeatedly commit crimes have formed criminal habits, so it is necessary to strengthen control.

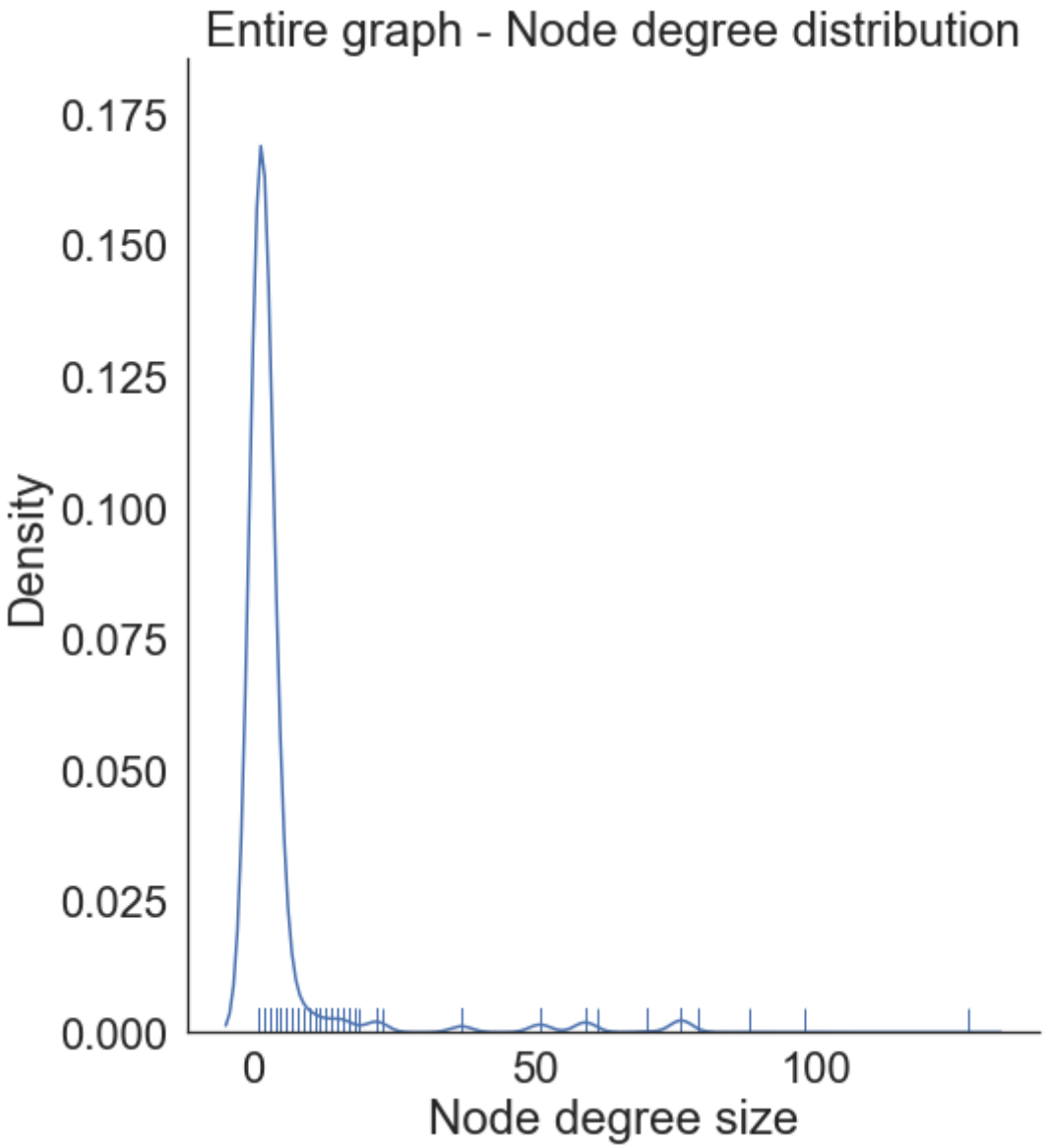
```
import numpy as np
import pandas as pd
import networkx as nx
from scipy.sparse import csr_matrix
```

```
import matplotlib
import matplotlib.pyplot as plt
import collections
import scipy
import statistics as stats
import seaborn as sns

df = pd.read_csv("Cooffending1.csv")
df.CrimeDate = pd.to_datetime(df.CrimeDate)
n_cases_raw = len(df)
df = df.drop_duplicates()
df = df.loc[(df['NumberYouthOffenders']>0.5)]
# There are five people who are listed as both M and F, just drop one of them
df = df.drop_duplicates(subset=["OffenderIdentifier", "CrimeIdentifier"])
n_cases = len(df)
n_criminals = len(df.OffenderIdentifier.unique())
n_crimes = len(df.CrimeIdentifier.unique())
df.head()

# Remap to consecutive identifiers
OffenderIdentifier_dict = {OffenderIdentifier: i for i, OffenderIdentifier in
enumerate(df.OffenderIdentifier.unique())}
CrimeIdentifier_dict = {CrimeIdentifier: i for i, CrimeIdentifier in
enumerate(df.CrimeIdentifier.unique())}

# .replace has a lot of overhead
df.OffenderIdentifier = df.OffenderIdentifier.map(OffenderIdentifier_dict.get)
df.CrimeIdentifier = df.CrimeIdentifier.map(CrimeIdentifier_dict.get)

assert not df.OffenderIdentifier.isnull().any() and not
df.CrimeIdentifier.isnull().any()
assert df.CrimeIdentifier.max() == df.CrimeIdentifier.nunique() - 1
assert df.OffenderIdentifier.max() == df.OffenderIdentifier.nunique() - 1

# Build matrix
row = df.OffenderIdentifier
col = df.CrimeIdentifier
vals = np.ones(len(row))

# Sparse representation
crime_matrix = csr_matrix((vals, (row, col)), shape=(row.max() + 1, col.max() +
1))

cooffend_matrix = crime_matrix @ crime_matrix.T

# Save an unmodified copy for later
cooffend_matrix_raw = cooffend_matrix.copy()

# convert to (binary) adj. matrix
# Could use the full cooffending matrix for project
cooffend_matrix[cooffend_matrix > 0] = 1
cooffend_matrix.setdiag(0)
cooffend_matrix.eliminate_zeros() # To avoid self loops since setdiag(0) does
not itself change the sparsity pattern

g = nx.from_scipy_sparse_matrix(cooffend_matrix)
print("""
Number of nodes in g: {}
```

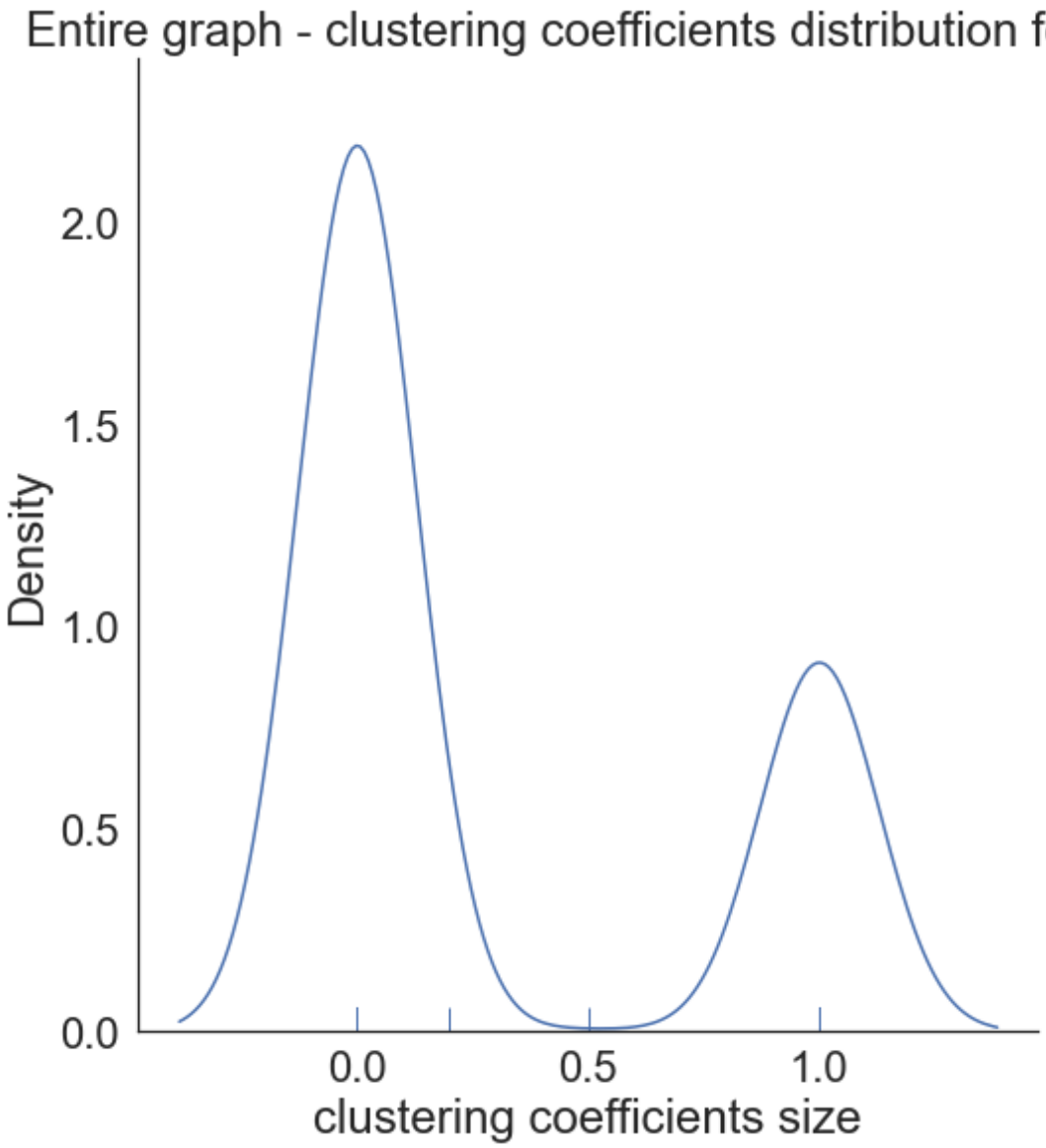
```
""".format(g.number_of_nodes()))

g_removed = g.copy()
g_removed.remove_nodes_from(list(nx.isolates(g_removed)))
G3 = g_removed
# print(G3)
degree_sequence = sorted([d for n, d in G3.degree()], reverse=True)
sns.set(font_scale=2)
sns.set_style("white")
ax = sns.displot(data = degree_sequence,  kind="kde", rug = True, height = 8)
ax.set(xlabel="Node degree size",ylabel= "Density",title='Entire graph - Node
degree distribution')
plt.show()

# Separate matrices for repeating co-offenders vs non-repeating co-offenders
cooffend_matrix_repeat = cooffend_matrix_raw.copy()
cooffend_matrix_no_repeat = cooffend_matrix_raw.copy()

# Repeating co-offenders: edge strength >= 2
cooffend_matrix_repeat.data[np.where(cooffend_matrix_repeat.data<2)[0]]=0
cooffend_matrix_repeat[cooffend_matrix_repeat > 0] = 1
cooffend_matrix_repeat.setdiag(0)
cooffend_matrix_repeat.eliminate_zeros() # To avoid self loops since setdiag(0)
does not itself change the sparsity pattern

# Non-repeating co-offenders: edge strength = 1
cooffend_matrix_no_repeat.data[np.where(cooffend_matrix_no_repeat.data!=1)[0]]=0
cooffend_matrix_no_repeat[cooffend_matrix_no_repeat > 0] = 1
cooffend_matrix_no_repeat.setdiag(0)
cooffend_matrix_no_repeat.eliminate_zeros() # To avoid self loops since
setdiag(0) does not itself change the sparsity pattern

g_r = nx.from_scipy_sparse_matrix(cooffend_matrix_repeat)
g_nr = nx.from_scipy_sparse_matrix(cooffend_matrix_no_repeat)

g_r.remove_nodes_from(list(nx.isolates(g_r)))
g_nr.remove_nodes_from(list(nx.isolates(g_nr)))

print("""
Number of nodes in G_r: {}
Number of nodes in G_nr: {}
""".format(g_r.number_of_nodes(), g_nr.number_of_nodes()))

g_r_clustering = list(nx.clustering(g_r).values())
g_nr_clustering = list(nx.clustering(g_nr).values())

sns.set(font_scale=2)
sns.set_style("white")
ax = sns.displot(data = g_r_clustering,  kind="kde", rug = True, height = 8)
ax.set(xlabel="clustering coefficients size",ylabel= "Density",title='Entire
graph - clustering coefficients distribution for Gr')
plt.show()

ax = sns.displot(data = g_nr_clustering,  kind="kde", rug = True, height = 8)
ax.set(xlabel="clustering coefficients size",ylabel= "Density",title='Entire
graph - clustering coefficients distribution for Gnr')
plt.show()
```

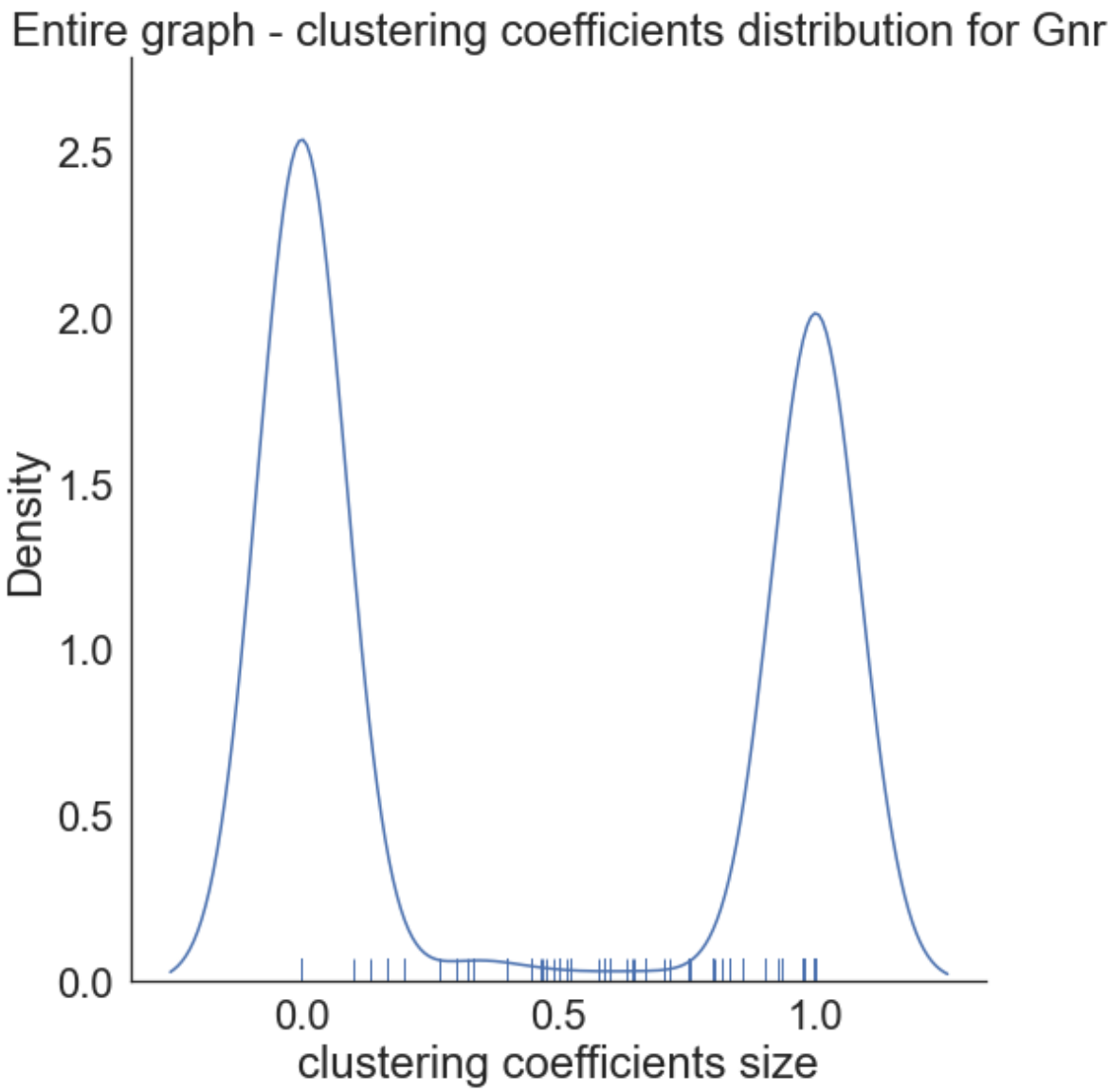
```
g_r_component_list = sorted(nx.connected_components(g_r), key=len, reverse=True)
g_nr_component_list = sorted(nx.connected_components(g_nr), key=len,
reverse=True)

g_r_largest_component = g_r.subgraph(g_r_component_list[0])
g_nr_largest_component = g_nr.subgraph(g_nr_component_list[0])

g_r_largest_component_clustering =
list(nx.clustering(g_r_largest_component).values())
g_nr_largest_component_clustering =
list(nx.clustering(g_nr_largest_component).values())

# g_r_largest_component_clustering_sequence = sorted([d for d in
g_r_largest_component_clustering], reverse=True)
# g_nr_largest_component_clustering_sequence = sorted([d for d in
g_nr_largest_component_clustering], reverse=True)

print("""
Number of nodes in largest_component of G_r: {}
Number of nodes in largest_component of G_nr: {}
""".format(g_r_largest_component.number_of_nodes(),
g_nr_largest_component.number_of_nodes()))

sns.set(font_scale=2)
sns.set_style("white")
ax = sns.displot(data = g_r_largest_component_clustering,  kind="kde", rug =
True, height = 8)
ax.set(xlabel="clustering coefficients size",ylabel= "Density",title='Entire
graph - clustering coefficients distribution for largest_component in Gr')
plt.show()


ax = sns.displot(data = g_nr_largest_component_clustering,  kind="kde", rug =
True, height = 8)
ax.set(xlabel="clustering coefficients size",ylabel= "Density",title='Entire
graph - clustering coefficients distribution for largest_component in Gnr')
plt.show()
```

```
Number of nodes in g: 13035
```



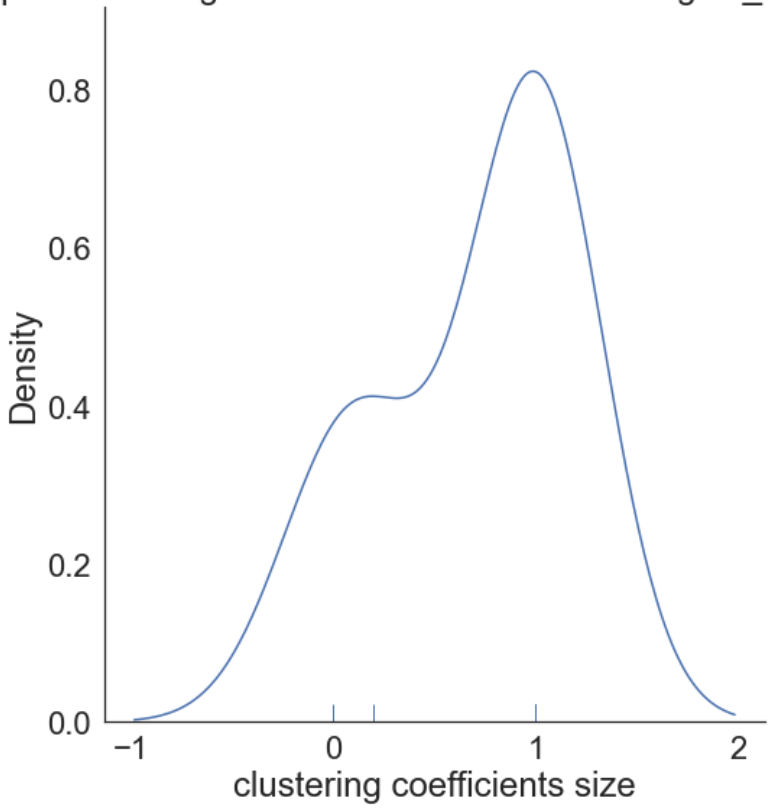

```
Number of nodes in G_r: 563
Number of nodes in G_nr: 6329
```


Entire graph - clustering coefficients distribution for Gr
Density
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
clustering coefficients size




```
Number of nodes in largest_component of G_r: 6
Number of nodes in largest_component of G_nr: 161
```

Entire graph - clustering coefficients distribution for largest_component in Gr



Entire graph - clustering coefficients distribution for largest_component in Gnr