Network Analysis Project

CAO Anh Quan
M2 D&K Social Data Management

January 23, 2018

Contents

1 Network Introduction

In this project, I select the **email-Eu-core** which is generated using email data of a big European research institution. The network only represents the communication between the members of the institution and no communication with the rest of the world.

The network contains 42 ground-truth communities which are corresponding the 42 departments in the institution. In this report, we will refer this network as the **real network**.

2 Show the number of nodes and edges

- Number of nodes: **1005**
- Number of edges: **16706**

```
num_nodes = len(G.nodes)
print("Number of nodes", num_nodes)
# Number of nodes 1005

num_edges = len(G.edges)
print("Number of edges", num_edges)
# Number of edges 16706
```

3 Draw the graph

```
nx.draw(G, with_labels=True)
```

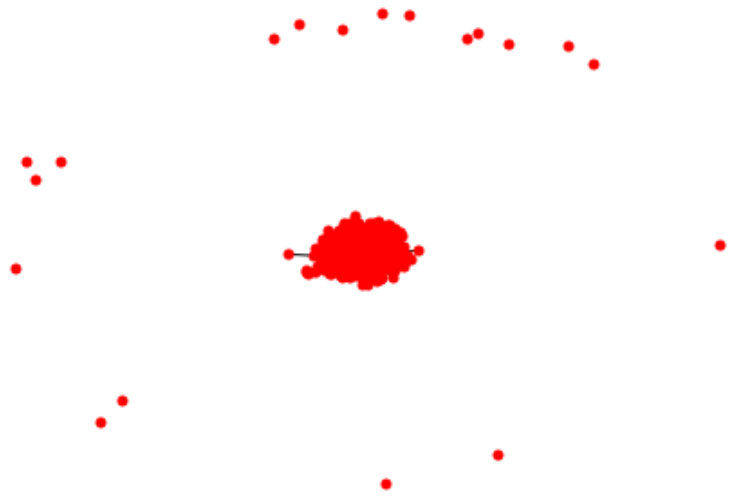


Figure 1: Draw the real network

We can see that there are many isolated nodes in the network. Hence, I will remove all these nodes and draw the graph again.

```
subgraphs = list(nx.connected_component_subgraphs(G))
for g in subgraphs:
    if len(list(g.nodes())) == 1:
        G.remove_nodes_from(g.nodes())
nx.draw(G, node_size = 25)
```

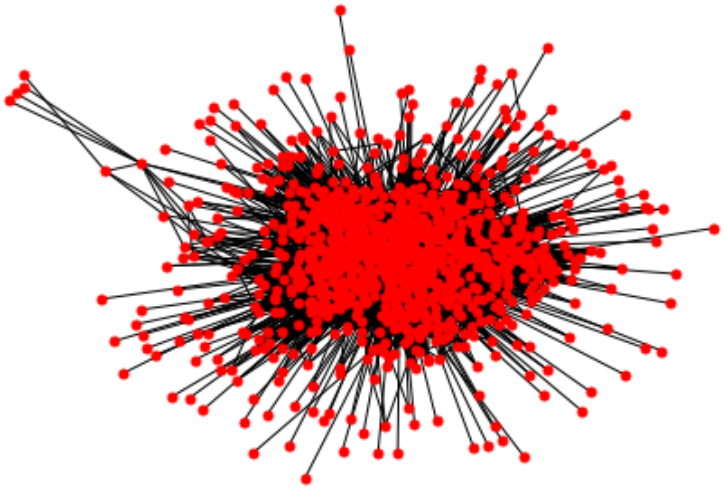


Figure 2: Draw the real network after removing isolate nodes

4 Draw the histogram of degrees. Compare the distribution with the distribution for a random graph having the same average degree. Discuss the results.

4.1 Real network

This is directed network, so the degree k_i of a node i is the sum of outgoing k_i^{out} and ingoing k_i^{in} degrees.

$$k_i = k_i^{out} + k_i^{in}$$

The average degree of the real network is **33.246**

```
def degrees_to_list(G):
    return list(map(lambda x: x[1], list(nx.degree(G))))
degrees = degrees_to_list(G)
sns.distplot(degrees)

average_degree = np.mean(degrees)
print("Average degree", average_degree)
# Average degree 33.246
```

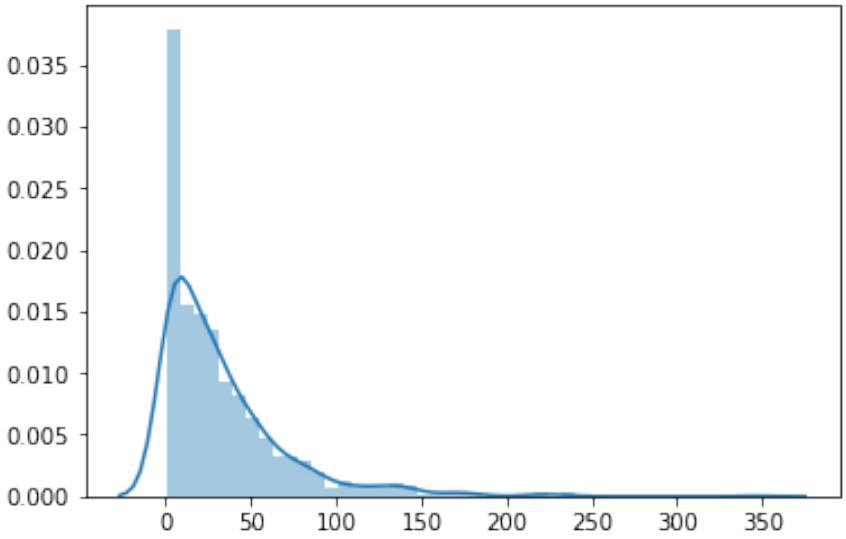


Figure 3: Degree distribution of the real network

4.2 Random Network

```
# Generate random network
p = average_degree / (num_nodes - 1)
print("p:", p)
# p: 0.03311331787278746
RG = nx.gnp_random_graph(num_nodes, p)

# Draw the histogram of degrees
r_degrees = degrees_to_list(RG)
sns.distplot(r_degrees)
```

The probability that 2 nodes connect to each other is $p = 0.03311331787278746$

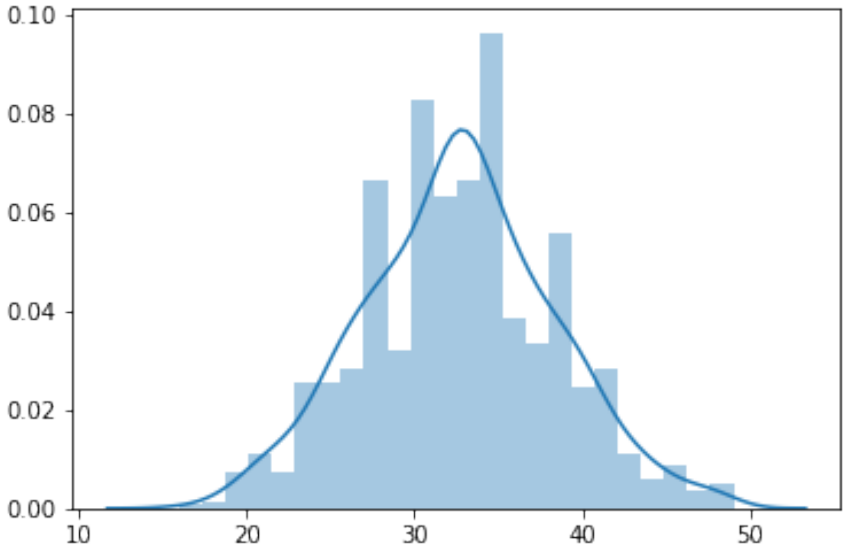


Figure 4: Degree distribution of the random network

4.3 Discussions

We can see from the figure 4 that the random network has the Poisson distribution with most of the nodes have the degree in the vicinity of the average degree. However, base on the figure 3 the real network that we showed above does not have the Poisson distribution.

The random network underestimates the size and the frequency of both **high degree (hub)** and **low degree nodes**. According to the figures 3 and 4, the real network has nodes with the degree of over 300 (hub), but the random network only has the node with the degree less than 60. We also see that in the real network, we have a large number of nodes with the degrees close to 0. However, this is not the case for the random network as we have just a few nodes close to zero. Because this is the email communication in an institution, there are some people with a very high degree - received a lot of emails like the director or the secretary. Researcher usually works in a small group of about 5 to 10 people which lead to the case that we have a lot of nodes with degree close to 0.

In conclusion, the random network does not capture the degree of the real network.

5 Draw the histogram of the clustering coefficient, and the average clustering coefficient. Compare it with the one of a random graph and discuss the results.

5.1 Real Network

The average clustering coefficient of the real network is **0.399**

```
G_cc_values = list(nx.clustering(G).values())
avg_G_cc_value = np.mean(G_cc_values)
print("Average clustering coefficient", avg_G_cc_value)
```

```
# Average clustering coefficient 0.39935496642215434
```

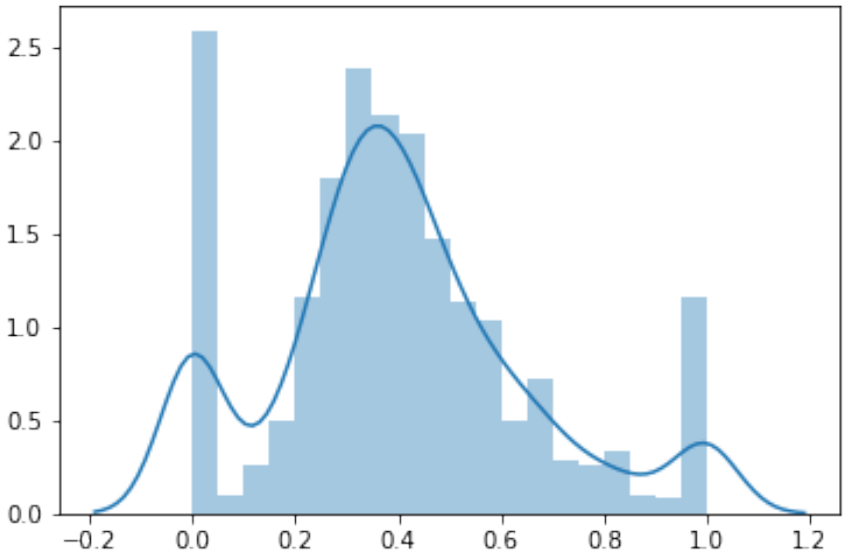


Figure 5: Clustering coefficient distribution of the random network

5.2 Random Network

The average clustering coefficient of the random network is **0.033**

```
RG_cc_values = list(nx.clustering(RG).values())
avg_RG_cc_value = np.mean(RG_cc_values)
print("Average clustering coefficient", avg_RG_cc_value)

# Average clustering coefficient 0.03316141649539389

sns.distplot(RG_cc_values)
```

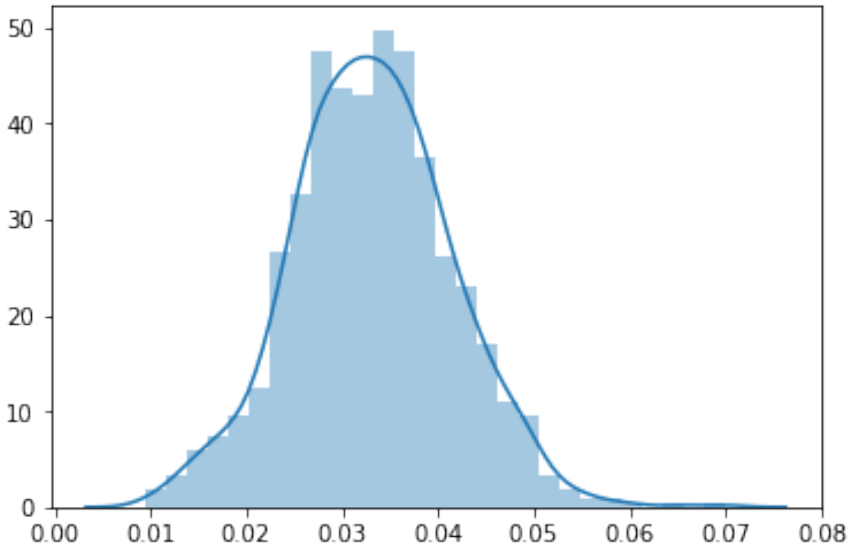


Figure 6: Clustering coefficient distribution of the random network

5.3 Discussion

According to the figures 5 and 6, the real network has much higher clustering coefficient than the random network because researchers work in a research team and people in a teamwork with each other.

There are many nodes in the real network has the clustering coefficient of about one because all the people in a team usually work together which means that they all know others. On the other hand, There are nodes with a clustering coefficient close to 0 seems to represent the staff like the secretary. They receive and send email to many departments, and the people in different department often do not each other.

In conclusion, the random network model does not capture the clustering of the real-world network.

6 Draw the histogram of distances in the graphs, the diameter, and the average distance. Compare with random graphs and discuss the results.

6.1 Real Network

The diameter of the network is **7**. The average distance is **2.587**

```
# we need to check if the graph is connected
print("Graph is connected:", nx.is_connected(G))
# Graph is connected: False
```

Because the graph is not connected, we will find all the connected component and then compute the diameter of each connected component

```
connected_components = nx.connected_component_subgraphs(G)
diameters = []
count = 0
for com in connected_components:
    d = nx.diameter(com)
    diameters.append(d)
    count += 1
print("Diameters", diameters)
print("Found", count, "connected component")
# Diameters [7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
# Found 20 connected component

paths = nx.shortest_path_length(G)
total = 0
count = 0
distances = []
for node, lengths in paths:
    values = lengths.values()
    count += len(values) - 1 # -1 to not count the path to itself
    total += sum(values)
    distances = distances + list(values)

# note that we add the distance of each path twice and we also count each path
    twice
# Therefore when we compute total/count the result is correct
print("Average distance", total / count)
# Average distance 2.586933824816466
```

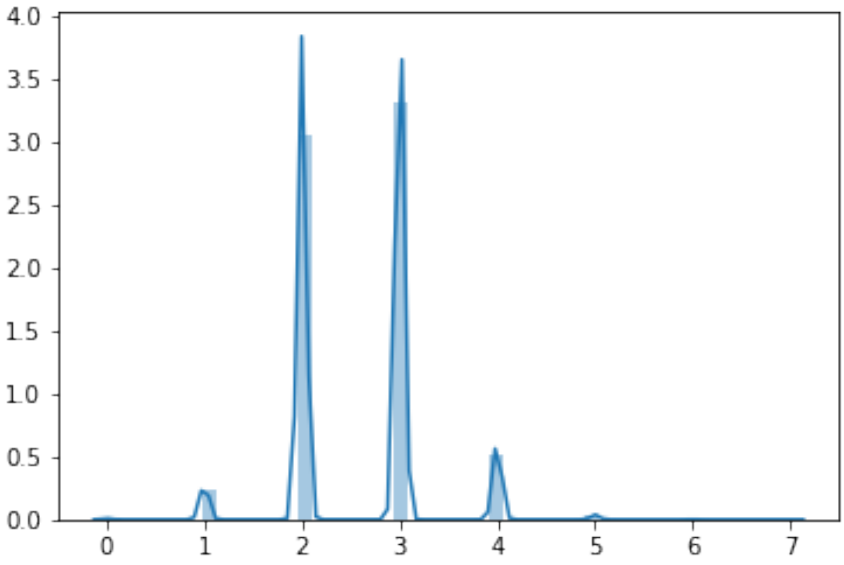


Figure 7: The histogram of distances in the real network

6.2 Random Network

The diameter of the network is **3**. The average distance is **2.286**

```
# We need to check if the graph is connected
print("Graph is connected:", nx.is_connected(RG))
# Graph is connected: True

print("Diameter", nx.diameter(RG))
# Diameter 3

paths = nx.shortest_path_length(RG)
total = 0
count = 0
distances = []
for node, lengths in paths:
    values = lengths.values()
    count += len(values) - 1 # -1 to not count the path to itself
    total += sum(values)
    distances = distances + list(values)

# note that we add the distance of each path twice and we also count each path
    twice
# Therefore when we compute total/count the result is correct
print("Average distance", total / count)
# Average distance 2.2860696517412937

# note that we add the distance of each path twice and we also count each path
    twice
# when we plot the distribution, all the distances are normalized => the
   result is correct
sns.distplot(distances)
```

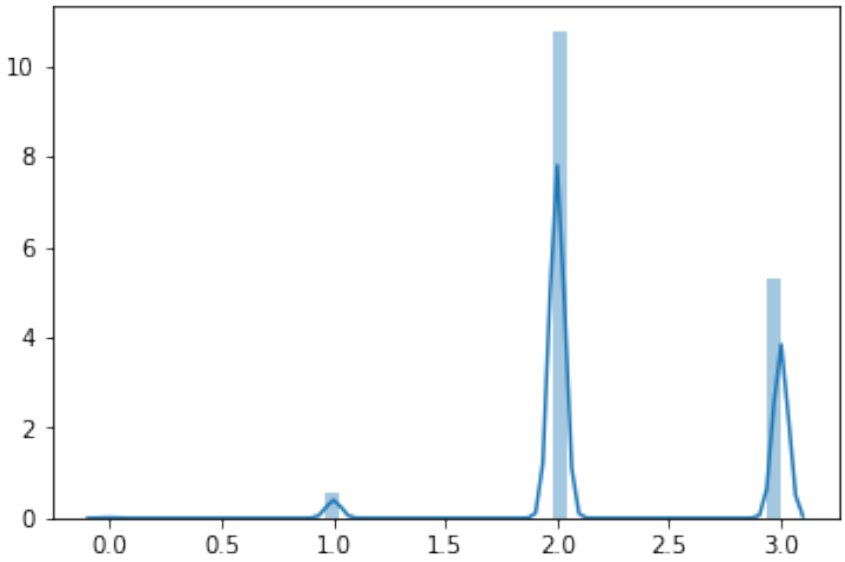


Figure 8: The histogram of distances in the random network

6.3 Discussion

First of all, I would like to mention the six degrees of separation (also known as small world phenomenon): Two individuals anywhere in the world, can be connected through a chain of six or fewer acquaintances. In network science, the six degrees of separation implies that the distance between two randomly chosen nodes in a network is short.

We can observe this phenomenon clearly in both random network and real network. In the real network, according to figure 7, most of the distances are 2 and 3, we have a small number of 1 and four distances. We have very few distances 5, 6 and 7. In the random network, according to the figure 8, we also observe a large number of nodes with distance 2 and 3. However, there is no distances 4, 5, 6, seven as in our real network. The random network has the average distance (2.286) approximately equal to the real network (2.587) For example, our random network has the average degree of **33.246**, for one node, we have:

- At distance 1: $33^1 = 33$ nodes
- At distance 2: $33^2 = 1089$ nodes
- At distance 3: $33^3 = 35937$ nodes

We can see that even at only distance 2, we have 1089 nodes which is much higher than the number of nodes in our network which is 1005. We have the maximum distance in a random network follows the equation:

$$d_{max} \approx \frac{ln(N)}{ln < k >} = \frac{ln(1005)}{ln(33.246)} = 1.973$$

With N is the number of nodes and $< k >$ is the average degree in the network. Therefore, we can observe that the small world phenomena are considerably well captured in the random

network model: Thanks to the fact that the number of nodes at distance d from a node increases exponentially with d.

Also, there is a difference between the two networks: the real network has a diameter of 7 while the random network has a diameter of 3. In my opinion, since our real network is from the research institution with 42 departments. People in one department do not know people in other departments and are connected through some people such as the secretary, staff. On the other hand, in the random network, all the nodes have the same probability to connect to other nodes which decrease the chance to have considerable distance. Therefore, we have more large distances 4, 5, 6, 7 and bigger diameter in the real network than the random network.

In conclusion, Only one property which is the small world phenomena is considerably demonstrated by the random network. All the other network characteristics: degree distribution, clustering coefficient is different in the real network.

7 Analyze the degree correlations of the graph.

7.1 Real Network

The real network has the degree assortativity coefficient of **-0.011**.

```
# Degree Assortativity
nx.degree_assortativity_coefficient(G)
# -0.010990490627931119
```

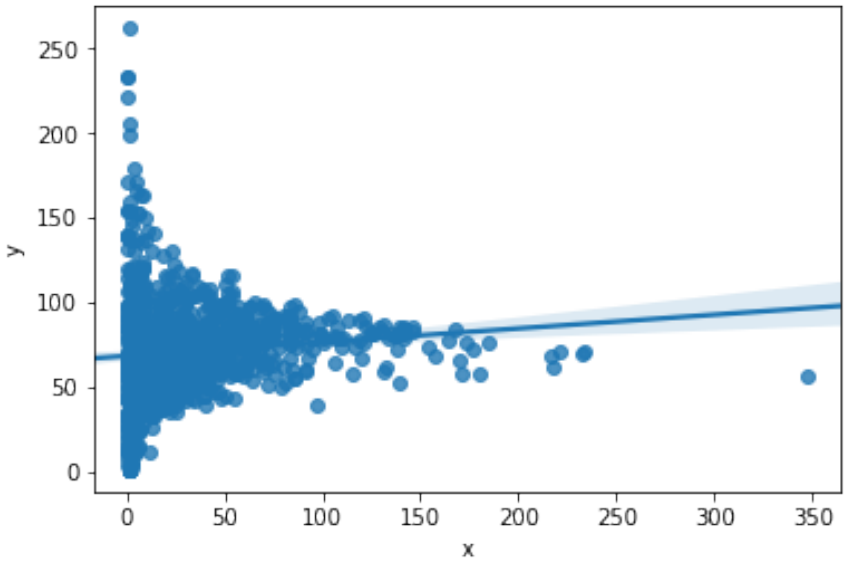


Figure 9: The degree correlation of the real network

7.2 Random Network

The random network has the degree assortativity coefficient of -0.004

```
# Degree correlation in Random Network
nx.degree_assortativity_coefficient(RG)
# -0.003987301784560138
```

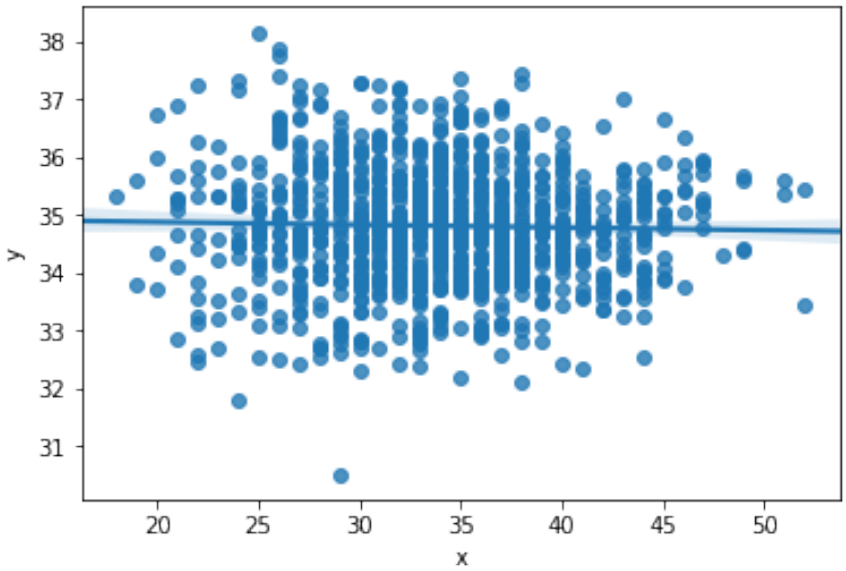


Figure 10: The degree correlation of the random network

7.3 Discussion

The real network has the assortativity coefficient of **-0.011** and the random network assortativity coefficient of **-0.004**. According to the assortativity coefficient and figure 10, the random network is a neutral network with a little disassortative because the value of the assortativity coefficient is very close to 0 (-0.004). In the random network, we place the edges randomly without regard the degree of the nodes.

Base on the assortativity coefficient and the figure 9, the real network is more disassortative than the random network (-0.011 compare to -0.004). There are are many nodes with the low degree (e.g., degree close to 0) connects to high degree nodes (e.g., an average degree of more than 150). However, there is also a reasonable number of nodes connected to nodes with the same degree, especially the nodes in the range of degree from 0 to 100.

The reason for the case that many low degree nodes connect to high degrees nodes is that: for instance, the secretary, the staffs of the institution is the hubs. There are many people such as the interns, invited researchers come to the institution, and they only communicate with the staff and few people in their team which mean that they only connect to few people with a very high degree. Therefore, they are the nodes with a low degree but have a very high average degree of neighbors.

For the case that many nodes with the same degree connect, I believe the reason could be the fact that researchers communicate with all other researchers in their team and those researchers also interact with all the researcher's team. Thus, they will likely to have the same degree, and they also are neighbors of other researchers in the group. Hence, their neighbors will have the same degree as them which cause the case that many nodes with the degree from 0 to 100 connect to the nodes with the same degree.

Besides, there is the case that some people with a very high degree but their neighbors have a much lower degree than them. They could be the director, the secretary, the staffs. They have a very high degree, and they have to communicate with almost everyone in the institution.

8 Detect the communities in the graph, and discuss the results.

8.1 Detect the communities

To detect the communities in the graph, I use the greedy modularity communities algorithm.

1. Assign each node as one community.
2. Iterate over all the connected community pair. Find the one that if we merge, the modularity increases the most.
3. Repeat until we have only one community which contains all the nodes.
4. Select the partition with the highest modularity.

```
G = nx.read_edgelist("email-Eu-core.txt")
communities = [list(c) for c in nx.algorithms.community.modularity_max.
    greedy_modularity_communities(G)]

clustering = []
avg_distances = []
diameters = []
d = dict()
for c in communities:
    if len(c) in d:
        d[len(c)] += 1
    else:
        d[len(c)] = 1
    if len(c) > 1:
        subgraph = G.subgraph(c)
        clustering.append(nx.average_clustering(subgraph))
        avg_distances.append(nx.average_shortest_path_length(subgraph))
        diameters.append(nx.diameter(subgraph))
print("Number of communities:", len(communities))
print(d)
print("Average clusterings", clustering)
print("Average distances", avg_distances)
```

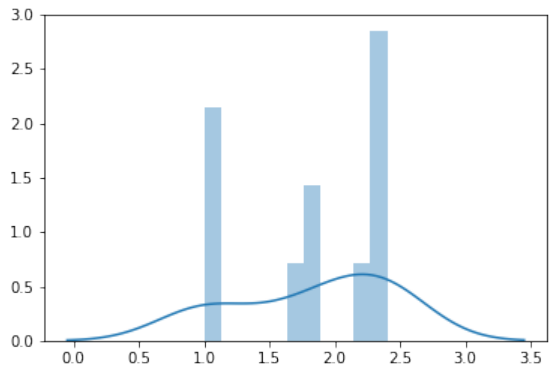
```
print("diameters", diameters)
```

```
Result:
Number of communities: 44
{368: 1, 321: 1, 103: 1, 89: 1, 58: 1, 9: 3, 2: 3, 1: 33}
Average clusterings [0.45288246251167213, 0.4332951307815015,
    0.5049759456222491, 0.513898657705322, 0.7014639740718691,
    0.5814814814814815, 0.30687830687830686, 0.0, 0.0, 0.0, 0.0]
Average distances [2.4019962089799787, 2.3896806853582553, 2.190938511326861,
    1.8730847803881512, 2.3914095583787054, 1.6388888888888888,
    1.8888888888888888, 2.388888888888889, 1.0, 1.0, 1.0]
diameters [5, 5, 5, 4, 6, 3, 3, 5, 1, 1, 1]
```

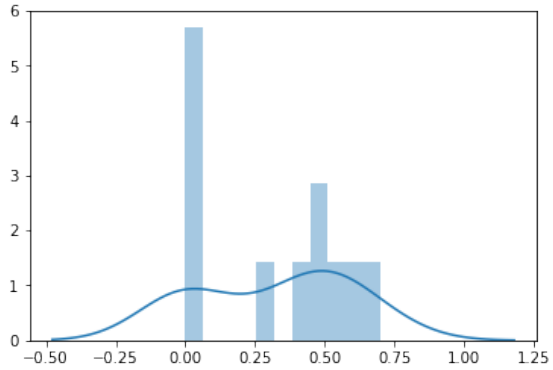
Modularity is the measure of the graph structure which is used to measure the quality of the division of the network into communities. The network with higher modularity has the denser connection between nodes in the module while the sparser connection between nodes in a different module.

The greedy modularity algorithm finds 44 communities in the real network.

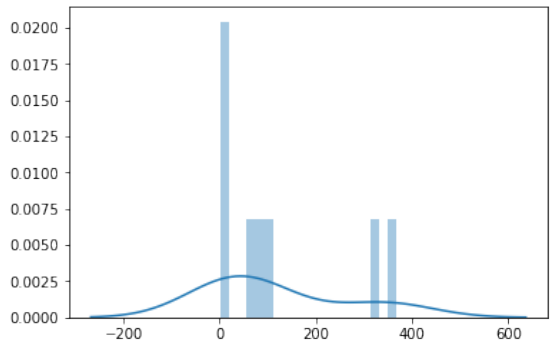
- 1 community with 368 nodes
- 1 community with 312 nodes
- 1 community with 103 nodes
- 1 community with 89 nodes
- 1 community with 58 nodes
- 3 community with 9 nodes
- 3 community with 2 nodes
- 33 community with 1 nodes



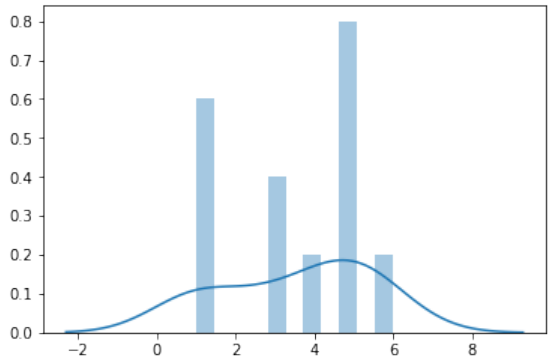
(a) The average distances distribution of the communities



(b) The average clustering coefficient distribution of the communities



(c) The communities size distribution



(d) The communities diameters distribution

Figure 11: The size, average clustering coefficient, diamters and average distribution of the communities

We can see that the real network has the average clustering coefficient of 0.4. The communities have much higher clustering coefficient than the real network from 0.5 to 0.8 because the connection in the community is much denser. There are communities with 0 clustering coefficient because they have only 2 or 3 nodes and their link is a line. The communities have a smaller diameter (most of the communities has the diameter of 1, 3 and 5) and average distances (1.5 to 2.5) than the original network (average distance: 2.587 and diameter: 7) which make sense because the community is the subgraph of the original graph.

8.2 Ground-truth communities

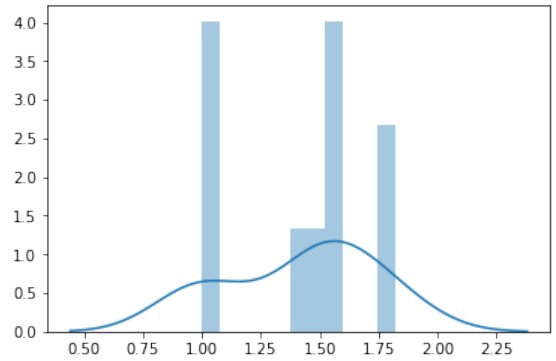
There are 42 ground-truth communities in the dataset corresponding to the 42 departments in the institution. Next, I will read the ground-truth communities and compare with the algorithm.

```
G = nx.read_edgelist("email-Eu-core.txt")
clustering = []
avg_distances = []
diameters = []
subgraphs_not_connected = 0
```

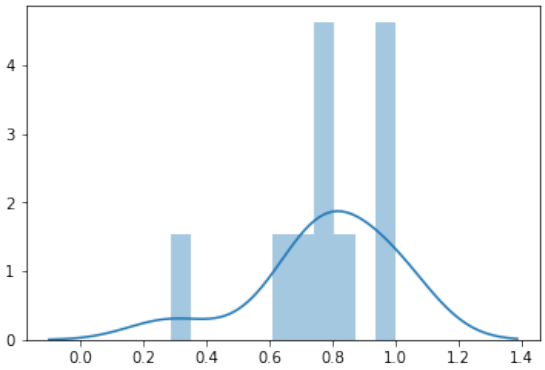
```
communities = list(deps.values())
for c in communities:

    if len(c) > 1:
        subgraph = G.subgraph(c)

        if nx.is_connected(subgraph):
            clustering.append(nx.average_clustering(subgraph))
            avg_distances.append(nx.average_shortest_path_length(subgraph))
            diameters.append(nx.diameter(subgraph))
        else:
            subgraphs_not_connected += 1
print("Number of communities:", len(communities))
print("Average clusterings", clustering)
print("Average distances", avg_distances)
print("diameters", diameters)
print(subgraphs_not_connected, "not connected communities")
```

```
Result
Number of communities: 42
Average clusterings [1.0, 0.825560036393977, 0.7888430880655762,
    0.7911976911976913, 0.7321259150206518, 0.6346153846153846, 0.2875, 1.0,
    0.7835467715316489, 1.0]
Average distances [1.0, 1.5270935960591132, 1.4891774891774892,
    1.438095238095238, 1.543859649122807, 1.7692307692307692,
    1.8214285714285714, 1.0, 1.583743842364532, 1.0]
diameters [1, 3, 3, 3, 3, 3, 3, 1, 3, 1]
30 not connected communities
```

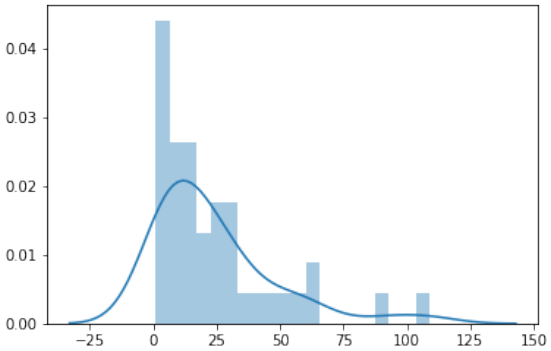
We can see that there are 30 disconnected communities. Thus, there are only 12 communities left which is close to the number of communities that we have predicted. Those disconnected communities make sense because researchers work in a team and there are many teams in each department. It is possible that researchers in a team only communicate with each other and do not communicate with people in other groups. Therefore, the community corresponding to this department is disconnected.



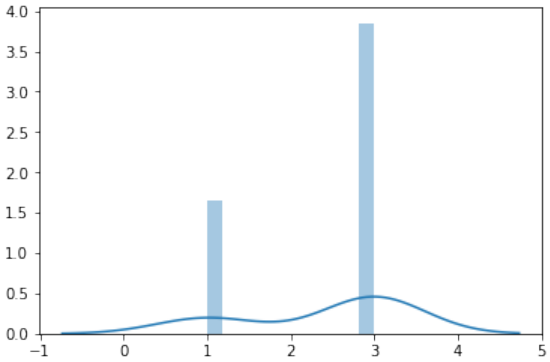
(a) The average distances distribution of the ground-truth communities



(b) The average clustering coefficient distribution of the communities



(c) The communities size distribution



(d) The communities diameters distribution

Figure 12: The size, average clustering coefficient, diamters and average distribution of the ground-truth communities

We can see that the ground-truth communities have smaller average distances than the ones that we predicted because there could be some departments have a strong connection with other departments such as the administrative department and our algorithm possibly group some departments that are strongly connected into a community and this increase the average distance in the community.

The ground-truth communities have smaller communities size with most of the communities have a small size in the range from 0 to 50 which result in larger clustering coefficient and a smaller diameter in the communities because the ground-truth communities capture the real departments and we can see that the size of the department is small. Therefore, people in those departments are more likely to work and communicate with each other.

The communities that we predicted have larger diameters (up to 5) and smaller average clustering coefficient (up to 0.8) because the algorithm probably merges many departments because in reality there are departments that communicate a lot with other departments such as the administrative department that I mentioned above. Thus, we can see that the algorithm predicted two huge communities with the size close to 400 and these communities could be the combination of many departments that have a strong connection.

9 Count the number the triangles in the graph, and compare to a random graph.

9.1 Real Network

```
triangles = nx.triangles(G)
print("Total triangles", sum(triangles.values()) / 3)
# Total triangles 105461.0
```

9.2 Random Network

```
triangles = nx.triangles(RG)
print("Total triangles", sum(triangles.values()) / 3)
# Total triangles 6021.0
```

9.3 Discussion

There are 105,461 triangles in the real network which is almost 20 times larger than 6,021 triangles in the random network. In my opinion, the first reason is that the clustering coefficient in the real network is much higher than the random network so that there are more connections in node's neighbor which increase the chance of appearing the triangle. In reality, if person A know person B and person B know person C, there will be a high probability that person C know person A. Another reason is that in the real network, there are hubs which are nodes has a very high degree (many connections). Hence, there are many nodes link to these nodes which increase the probability of forming a triangle because between all the nodes connected to a hub, and we only need one more connection between every two nodes to form a triangle.

Furthermore, the expected number of the triangle in our random network is small because it depends on the exponential of p which is the probability of having an edge between 2 nodes. In our example, we have t is the number of combinations of 3 nodes in the random network

$$t = \binom{1005}{3} = 168,674,510$$

We have $p = 0.03311331787278746$ is the probability have an edge between 2 nodes. Hence, the expected number of triangles in the random network is:

$$t \times p^3 = 6124.315$$

We can see that this number is close to the number of triangles in our random network.

10 Compute and discuss other centrality measures: betweenness, PageRank

10.1 Betweenness

The betweenness is computed as follows:

$$g(v) = \sum_{s \neq v \neq t} = \frac{\sigma_{st}(v)}{\sigma_{st}}$$

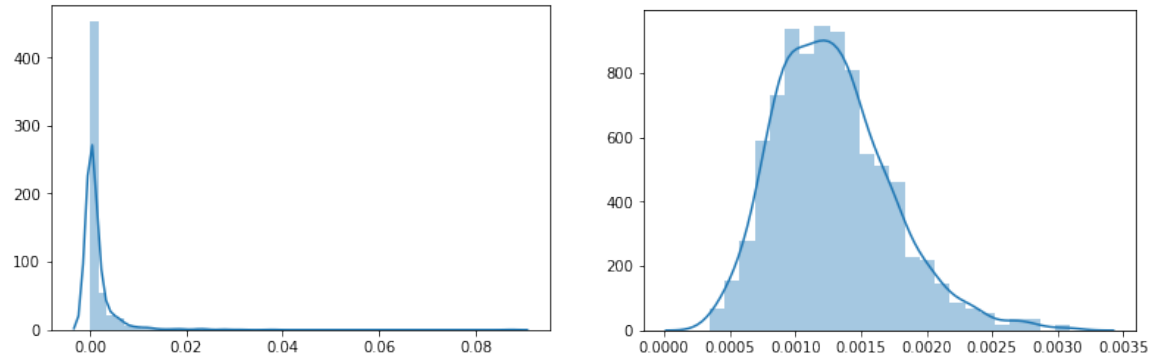
where σ_{st} is the total number of shortest paths from node s to node t and $\sigma_{st}(v)$ is the number of those paths that pass through v.

Real Network

```
G = nx.read_edgelist("email-Eu-core.txt")
btw = nx.betweenness_centrality(G)
sns.distplot(list(btw.values()))
print("Average betweenness", np.mean(list(btw.values())))
# Average betweenness 0.0015228995378930303
```

Random Network

```
RG = nx.gnp_random_graph(num_nodes, p)
btw = nx.betweenness_centrality(RG)
sns.distplot(list(btw.values()))
print("Average betweenness", np.mean(list(btw.values())))
# Average betweenness 0.0012859757727076446
```



(a) The betweenness distribution of the real network

(b) The betweenness distribution of the random network

Figure 13: The betweenness distribution of the real and random network

The real network has an average betweenness of 0.0015, and the random network has an average betweenness of 0.0013 which are quite similar. However, they have very different

betweenness distribution. We can see from the two figures 14a and 14b that the real network has some nodes with very high betweenness (bigger than 0.08). On the other hand, the random network only has the betweenness up to 0.0035. I believe the reason is that the real network has hubs which have a lot of connections so it will have a high probability to be in the shortest path of other nodes. Those hubs are the ones that have very high betweenness from 0.04 to over 0.08. It is not the case for the random network because it does not have the hub. The betweenness of the random network follows the Gaussian distribution because the edges in the random network are generated randomly with most of the nodes have the betweenness around 0.0013

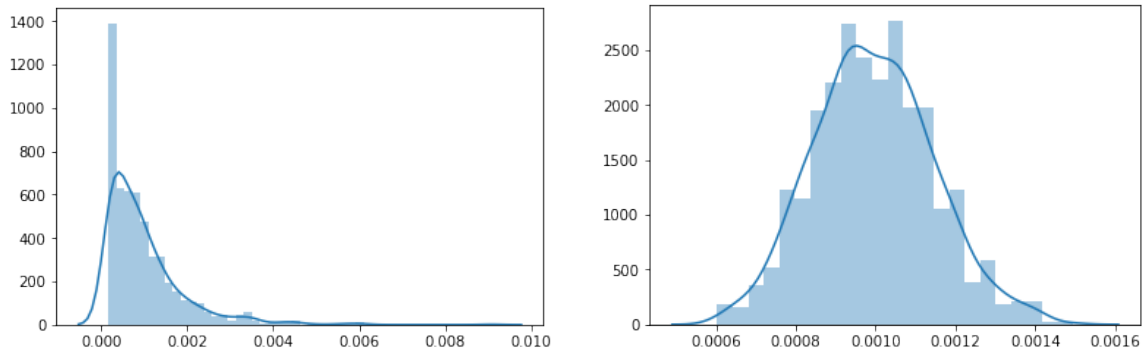
10.2 PageRank

Real Network

```
ranks = nx.pagerank(G)
rank_values = list(ranks.values())
print("Average rank", np.mean(rank_values))
# Average rank 0.0009950248756218903
sns.distplot(rank_values)
```

Random Network

```
ranks = nx.pagerank(RG)
rank_values = list(ranks.values())
print("Average rank", np.mean(rank_values))
# Average rank 0.0009950248756218903
sns.distplot(rank_values)
```



(a) The PageRank distribution of the real network

(b) The PageRank distribution of the random network

Figure 14: The PageRank distribution of the real and random network

According to the figure 14, we have the same situation as with the betweenness centrality. The real network and random network have the same average PageRank of 0.001. There are

some nodes in the real network have very high page rank (up to 0.01) while the maximum PageRank in the random network is 0.0016. I believe the reason is there are hubs in the real network which have a very high degree (a lot of connections). Therefore, the hubs will also have high PageRank. On the other hand, the PageRank of the random network follows the Gaussian distribution because the edges in the random network are generated randomly with most of the nodes have the rank around 0.001.

11 Do a comparative analysis of your social dataset and a non-social one (e.g., transport, Web).

In this section, I choose the US Power Grid Network for comparison. The network illustrates the power grid of the Western States of the USA. An edge represents a power supply line, and a node is either a generator, a transformator or a substation.

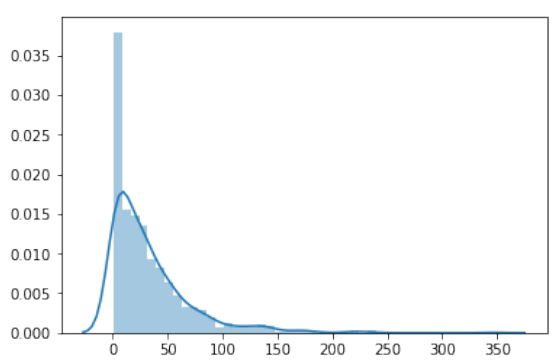
11.1 Number of Nodes and Edge

```
grid_G = nx.read_edgelist("out.opsahl-powergrid")
print("Number of Edges:", len(grid_G.edges()))
print("Number of Nodes:", len(grid_G.nodes()))
# Number of Edges: 6594
# Number of Eodes: 4941
```

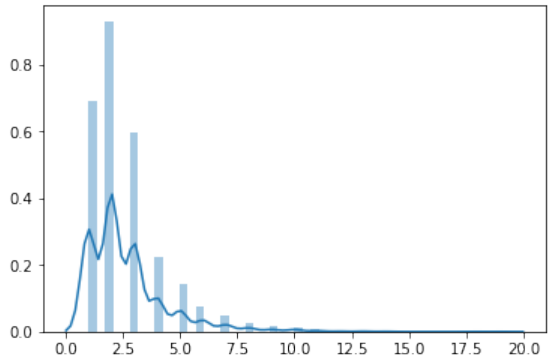
There are **6,594** nodes and **4.941** edges in this graph.

11.2 The degree

```
grid_degrees = list(map(lambda x: x[1], list(nx.degree(grid_G))))
sns.distplot(grid_degrees)
print("Average degree", np.mean(grid_degrees))
# Average degree 2.66909532483303
```



(a) The degree distribution of the email network



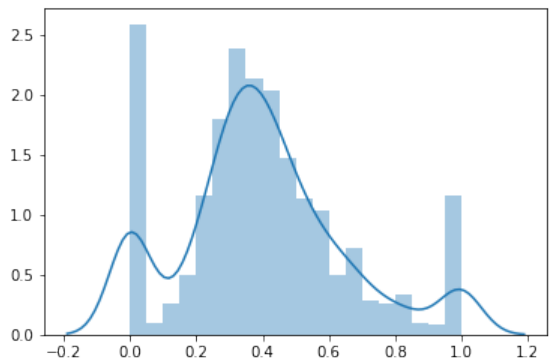
(b) The degree distribution of the Power Grid network

Figure 15: The degree distribution of the email and power grid networks

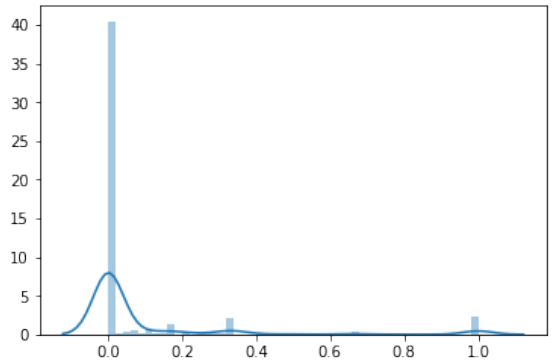
The average degree of the power grid network is **2.67** which is much smaller than the average degree of the email network (**33.246**). We also see from the figure **??** that the power grid network has a much smaller degree distribution than the email network. Most of the nodes have the degree from 1 to 7, and the largest degree is 20. The reason is that the power grid network is a **physical network**: Each node in this network is a generator, a transformator or a substation and we cannot have too many power supply lines connect to 1 generator because of physical limitation. We have a lot of nodes with degree 1, 2 and three which means that we have many generators with 1,2 and three power supply lines and it makes sense because it is common in reality. There are some nodes with higher degree up to 20 (**hubs**). They could be a huge generator in the power plants. On the other hand, for the email network, we do not have this physical limitation we can have as many connections as we want at a node: We can send email to every people in the institution.

11.3 Clustering coefficient

```
grid_G_cc_values = list(nx.clustering(grid_G).values())
avg_grid_G_cc_value = np.mean(grid_G_cc_values)
print("Average clustering coefficient", avg_grid_G_cc_value)
# Average clustering coefficient 0.08010361108159712

sns.distplot(grid_G_cc_values)
```



(a) The clustering coefficient distribution of the email network



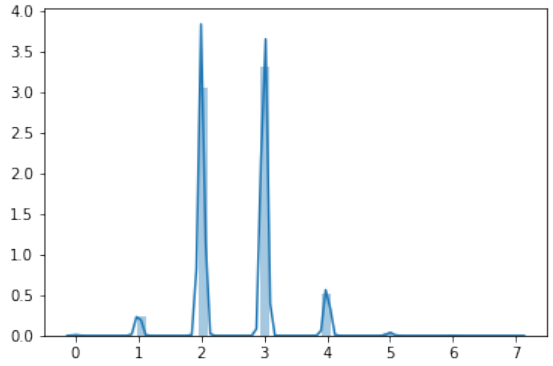
(b) The clustering coefficient distribution of the Power Grid network

Figure 16: The clustering coefficient distribution of the email and power grid networks

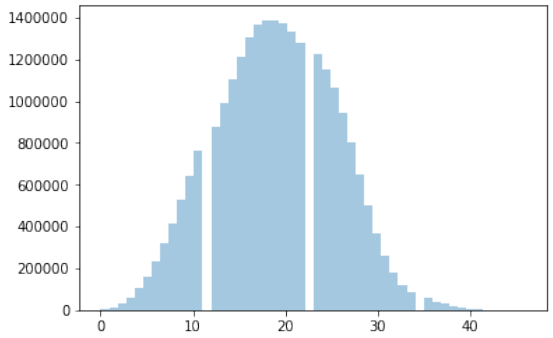
We can see that the email network (average clustering coefficient of **0.4**) has much higher clustering coefficient than the power grid network (average clustering coefficient of **0.0801**) because researchers work in a team and all the team members usually know each other which result in high clustering coefficient. However, it is not the case for the power grid network. Most of the nodes have the clustering coefficient close to zero. For example, we have a generator which sends the power to its neighbors. Those neighbors will send the power further instead of send to each other because it makes no sense to send to other neighbors who also received the energy from the generator. We also have some nodes in the power grid network have the clustering coefficient close 1. They could be in the power plant where we have a dense connection of electric devices.

11.4 Distance and diameter

```
print("Diameter", nx.diameter(grid_G))
# Diameter 46

paths = nx.shortest_path_length(grid_G)
total = 0
count = 0
distances = []
for node, lengths in paths:
    values = lengths.values()
    count += len(values) - 1 # -1 to not count the path to itself
    total += sum(values)
    distances = distances + list(values)

# note that we add the distance of each path twice
# and we also count each path twice
# Therefore when we compute total/count the result is correct
print("Average distance", total / count)
# Average distance 18.989185424445708

sns.distplot(distances, kde=False)
```



(a) The distance distribution of the email network



(b) The distance distribution of the Power Grid network

Figure 17: The clustering coefficient distribution of the email and power grid networks

We have the average distance of the power grid network is **18.981** which is higher than the average distance of the email network which is **2.587**. The same situation happens with the distance distribution: The email network distance distribution is much lower than the distance distribution of the Power Grid Network. In reality, to transfer the electricity in long distance, we will need many generators, transformators, and substations. We cannot use only one power line connecting the two nodes which are very far directly due to the physical limitation. Therefore, we have a large distance in the power grid network.

11.5 Degree Correlation

```
# Degree Assortativity
nx.degree_assortativity_coefficient(grid_G)
# 0.00345698774420483

knns = nx.average_neighbor_degree(grid_G)
degrees = nx.degree(grid_G)
dc = []
for node in knns:
    dc.append([degrees[node], knns[node]])
data = pd.DataFrame(dc)
data.columns = ["k", "knn"]
sns.regplot(x="k", y="knn", data=data)
```

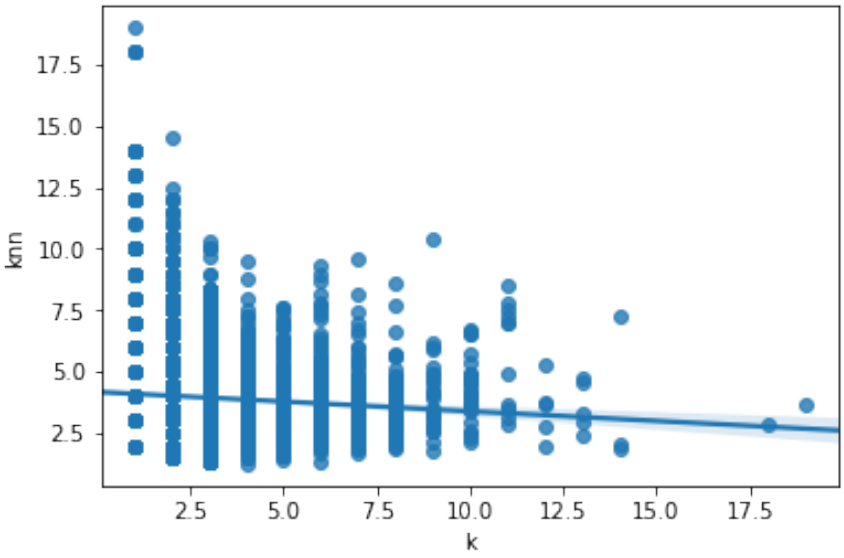


Figure 18: The degree correlation of the Power Grid network

The degree assortativity coefficient is **0.0034** which is slightly bigger than 0 indicate that the power grid network is slightly assortative. However, base on the figure 18, the power grid network seems to be disassortative with many low degree nodes connect to high degree nodes which could be the case in reality when we have 1 big generator in power plant (High degree nodes) connect to many other substations and transformators (low degree nodes). There are also some nodes connect to nodes with the same degree which could be the case in the residential area, we have many substations connect to others, and each of them provides electricity to the almost identical number of households.

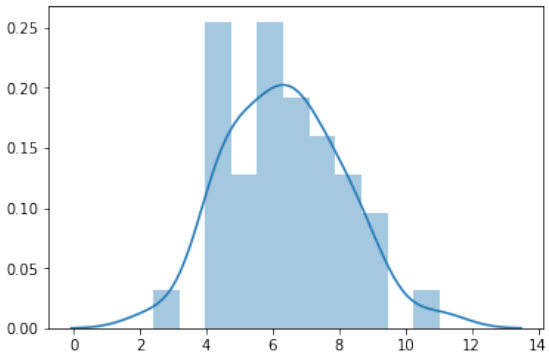
11.6 Communities Detection

```
grid_G = nx.read_edgelist("out.opsahl-powergrid")
communities = [list(c) for c in nx.algorithms.community.modularity_max.
    greedy_modularity_communities(grid_G)]

clustering = []
avg_distances = []
diameters = []
d = dict()
for c in communities:
    if len(c) in d:
        d[len(c)] += 1
    else:
        d[len(c)] = 1
    if len(c) > 1:
        subgraph = grid_G.subgraph(c)
        clustering.append(nx.average_clustering(subgraph))
        avg_distances.append(nx.average_shortest_path_length(subgraph))
        diameters.append(nx.diameter(subgraph))
print("Number of communities:", len(communities))
```

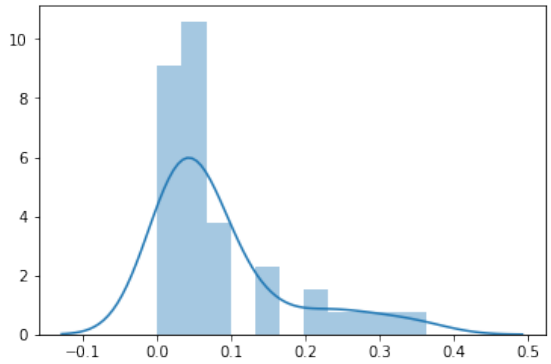
```
print(d)
print("Average clusterings", clustering)
print("Average distances", avg_distances)
print("diameters", diameters)
```

Result

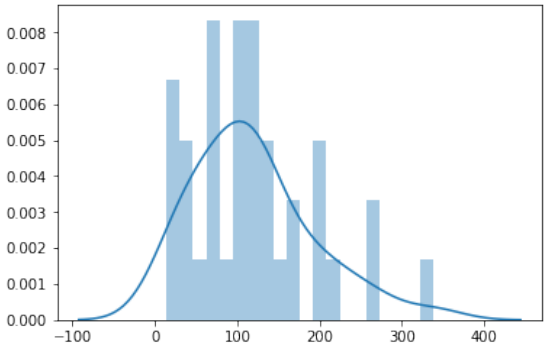
```
Number of communities: 40
{338: 1, 262: 1, 261: 1, 224: 1, 202: 1, 198: 1, 197: 3, 166: 1, 164: 1, 145: 2, 137: 1, 136: 1, 131: 1, 125: 1, 123: 1, 118: 1, 114: 1, 112: 1, 110: 1, 109: 1, 105: 1, 104: 1, 98: 1, 93: 1, 78: 1, 76: 1, 72: 1, 68: 1, 66: 1, 61: 1, 42: 1, 35: 1, 33: 1, 30: 1, 29: 1, 26: 1, 14: 1}
Average clusterings [0.07571829879522186, 0.026517629952744457, 0.018780027975430276, 0.216113474818832, 0.01155115511551155, 0.09794742294742297, 0.05912496978486825, 0.02762468777697204, 0.05400048344210779, 0.015313635494358385, 0.03660472319008904, 0.13583519928347515, 0.05794934950107362, 0.018004866180048658, 0.011274509803921568, 0.13887677208287894, 0.2503114452798663, 0.062085430378113295, 0.3186083690320979, 0.018525480367585632, 0.1451754099075528, 0.05636363636363637, 0.042223678462210576, 0.28620295048866473, 0.043269230769230775, 0.006122448979591836, 0.2193036354326677, 0.0, 0.010964912280701754, 0.09351851851851851, 0.04689542483660131, 0.0562049062049062, 0.3632969034608378, 0.08095238095238096, 0.0, 0.06565656565656566, 0.03333333333333333, 0.06321839080459771, 0.08717948717948718, 0.04047619047619048]
Average distances [7.726616683932365, 7.482085929045655, 8.666430887120542, 6.063781229980782, 8.83222501354613, 6.263344100907553, 6.957733347145965, 7.143426913912773, 8.034497047549985, 7.8326396495071196, 6.7180158611402065, 6.586494252873563, 5.360153256704981, 9.24463288965221, 8.676252723311547, 6.797298884321785, 5.038838709677419, 6.786218845795015, 5.910473707083876, 5.846141903431144, 5.798584298584299, 8.330108423686406, 6.2716615698267075, 4.4774725274725276, 6.08159073935773, 11.024405638544078, 5.724403927068724, 7.942723942723942, 7.153333333333333, 4.547339593114241, 4.2949956101843725, 5.207459207459207, 4.466666666666667, 4.200929152148665, 4.460504201680672, 6.746212121212121, 4.995402298850575, 4.440886699507389, 4.227692307692307, 2.3956043956043955]
diameters [18, 19, 25, 15, 20, 17, 15, 15, 19, 18, 15, 15, 12, 23, 24, 16, 14, 16, 14, 12, 15, 21, 14, 13, 13, 29, 11, 21, 22, 11, 10, 13, 11, 10, 10, 18, 13, 12, 10, 5]
```



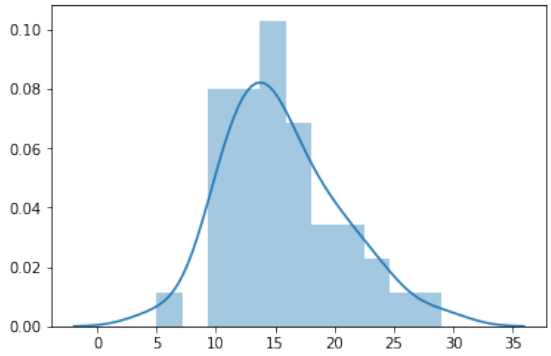
(a) The average distances distribution of the Power Grid communities



(b) The average clustering coefficient distribution of the Power Grid communities



(c) The communities size distribution



(d) The communities diameters distribution

Figure 19: The size, average clustering coefficient, diamters and average distribution of the ground-truth communities

There are 40 communities in the power grid network. In my opinion, each community is a city or a residential area in reality. Hence, they have the denser connection in the cities and the sparser connection between the cities. Most of them have a size between 0 and 200. We have some large communities with the size larger than 250. The size of the communities corresponding to the size of the cities. The bigger the city, the more electric devices we need, the bigger the communities. It makes sense because, in reality, we have only a few megacities, most of the cities are small. We have the same situation for the average distances, diameter and clustering coefficient.

11.7 Number of Triangles

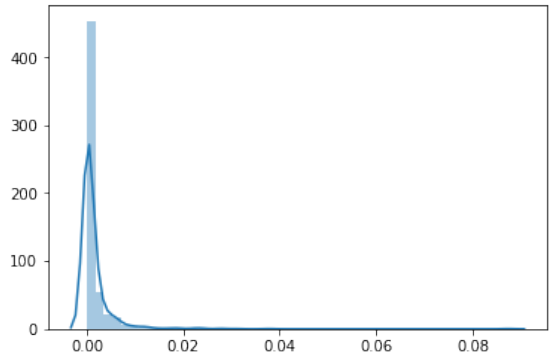
```
triangles = nx.triangles(grid_G)
print("Total triangles", sum(triangles.values()) / 3)
# Total triangles 651.0
```

We have the number of triangles in the email network is **105461** which is hugely more significant than the amount of the triangle in the power grid network which is **651** because

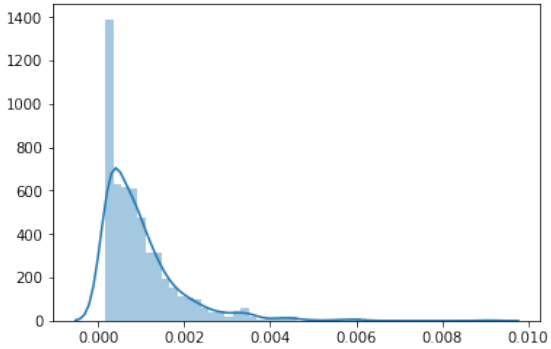
the email network is the social network of humans. In the research institution, if person A knows person B and person B knows person C, it is very likely that person C also knows person A which create a triangle. However, the situation is inverse for the power grid network. If we connect a substation A to substation B and substation B to substation C, It will be not likely that we will also connect the substation C to substation A.

11.8 Centrality Measures: Betweenness, PageRank

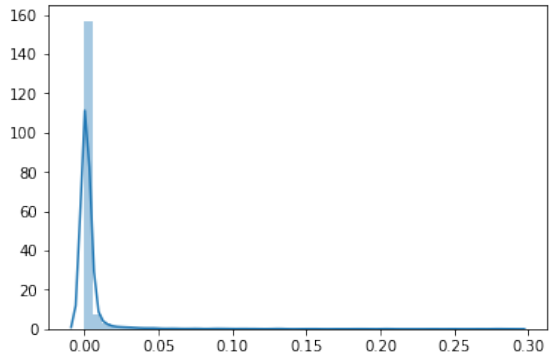
```
grid_G = nx.read_edgelist("out.opsahl-powergrid")
btw = nx.betweenness_centrality(grid_G)
sns.distplot(list(btw.values()))
print("Average betweenness", np.mean(list(btw.values())))
# Average betweenness 0.0036422728132103135

ranks = nx.pagerank(grid_G)
rank_values = list(ranks.values())
print("Average rank", np.mean(rank_values))
# Average rank 0.00020238818053025704
```



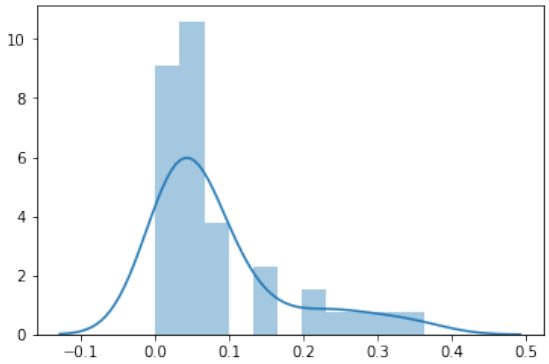
(a) The betweenness distribution of the Email network



(b) The PageRank distribution of the Email network



(c) The betweenness distribution of the Power Grid network



(d) The PageRank distribution of the Power Grid network

Figure 20: The betweeness and PageRank distribution of the Power Grid Network

The email network has the average betweenness of **0.0015** which is smaller than the average betweenness of the power network which is **0.003**. We also see that the betweenness and PageRank distribution of the power grid network is much higher than the email network although they have quite a similar shape of the distribution with most of the nodes have low betweenness and PageRank and few nodes have high betweenness and PageRank. In reality, if we have a power plant and a city need to be supplied, we will first build a power line to a substation in that city and from that substation, we transfer the electricity to other substations to power all the city and the structure of the grid would like a tree with many branches. Therefore, there are probably only a few shortest paths between two nodes which result in high betweenness and PageRank.

12 Other analysis: How to prevent the bombing?

In this section, I will use another network Train bombing. This network represents the contacts between terrorists involved in the train bombing of Madrid on March 11, 2004. The nodes are the terrorists, and the edges are the contacts between the terrorists. The edge weights denote how 'strong' a connection was.

Our target is to find which is the most three influence terrorists so that we can capture them to prevent the bombing.

12.1 Number of terrorists and contacts

```
G = nx.read_weighted_edgelist("out.moreno_train_train")
print("Number of terrorists", len(G.nodes()))
print("Number of contacts", len(G.edges()))
# Number of terrorists 64
# Number of contacts 243

nx.draw(G, with_labels=True)
```

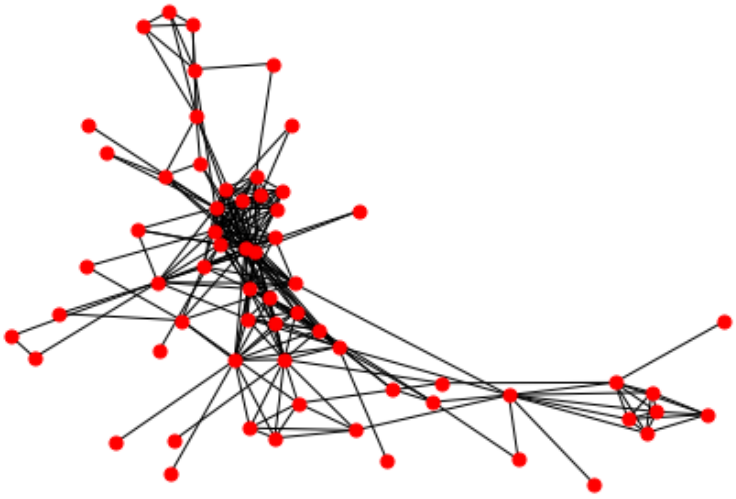


Figure 21: The terrorists graph

There are **64** terrorists and **243** contacts. We convert all the weights of the edges to probabilities to make it a uncertain graph. The weights of the graph become the probability that 2 terrorist has contact.

```
# convert the weight to probability to create uncertain graph
for u,v,d in G.edges(data=True):
    d['weight'] /= 4
```

12.2 Reachability and Expected Spread

I evaluate the reachability of expected spread of each terrorist to find the most 5 influence terrorists.

```
def can_reach(G, s, t):
    Q = deque([s])
    visited = set()
    while len(Q) != 0:
        u = Q.popleft()
        if u == t:
            return 1
        visited.add(u)
        N = G.neighbors(u)
        for v in N:
            if v not in visited:
                dice = rnd.random()
                p = G[u][v]['weight']
                if dice <= p:
                    Q.append(v)
    return 0

def reachability(G,s,t,rounds=100):
    good = 0
    for i in range(rounds):
        good += can_reach(G, s, t)
    return good / rounds

def expected_spreads(G, num_tops = 10):
    nodes = G.nodes()
    spreads = []
    for node1 in nodes:
        reach = 0
        for node2 in nodes:
            if node1 != node2:
                reach += reachability(G, node1, node2)
        spreads.append((node1, reach))
    sorted_spreads = sorted(spreads, key = lambda x: x[1], reverse=True)
    top_spreads = sorted_spreads[:num_tops]
    top_nodes = list(map(lambda x: x[0], top_spreads))
    return top_nodes


top_terrorists = expected_spreads(G, num_tops = 5)
print(top_terrorists)
# ['14', '13', '15']
```

By evaluating the reachability and expected spread, we discover the most 5 influence terrorists that we need to capture to prevent the bombing are **'13', '14', '15'**. Let's see these terrorists in the graph.

```
def draw_top_reachabilities_nodes(G, top_nodes):
    color_map = []
    for node in G:
        if node in top_nodes:
            color_map.append('green')
        else: color_map.append('red')
```

```
    nx.draw(G, with_labels=True, node_color = color_map)
    plt.show()

draw_top_reachabilities_nodes(G, top_terrorists)
```

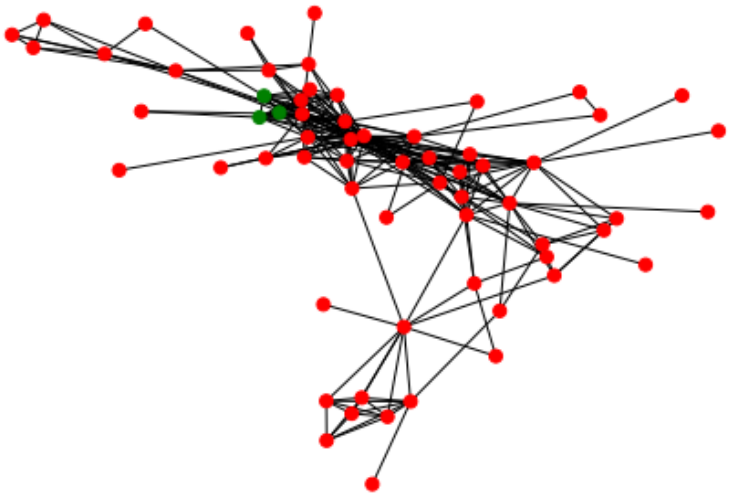


Figure 22: The most important terrorists in the network. The green nodes is the 3 most influence terrorists in the network

12.3 Influence Maximization

In this section, I implement the Influence Maximization: Greedy Algorithm to find the most 10 influence terrorists in the network.

```
def can_reach_from_seeds(G, seed_nodes, t):
    Q = deque(set(seed_nodes))
    visited = set()
    while len(Q) != 0:
        u = Q.popleft()

        if u == t:
            return 1
        visited.add(u)
        N = G.neighbors(u)
        for v in N:
            if v not in visited:
                dice = rnd.random()
                p = G[u][v]['weight']
                if dice <= p:
                    Q.append(v)
    return 0

```

```
def reachability_seeds(G, seed_nodes, t, rounds = 100):
    good = 0
    for i in range(rounds):
        good += can_reach_from_seeds(G, seed_nodes, t)
    return good / rounds

def spread(G, seed_nodes, rounds = 100):

    spread = 0
    for target_node in G.nodes():
        if target_node not in seed_nodes:
            spread += reachability_seeds(G, seed_nodes, target_node, rounds =
   rounds )
    return spread

def greedyIM(G, k = 3):
    S = []
    total_spread = 0
    while len(S) < k:
        print(len(S))
        max_spread = 0
        max_u = None
        for u in G.nodes():
            m_spread = spread(G, S + [u]) - total_spread
            if m_spread > max_spread:
                max_spread = m_spread
                max_u = u
        S = S + [max_u]
        total_spread = total_spread + max_spread
    return S, total_spread

S, total_spread = greedyIM(G, k = 3)
print("Most Influence Nodes:",S)
print("Total spread:", total_spread)
# Most Influence Nodes: ['14', '11', '10']
# Total spread: 52.830000000000005
```

By using the Influence Maximization Greedy Algorithm, we discover 3 most influence terrorist: **'14', '11', '10'**. Let's see them in the graph.

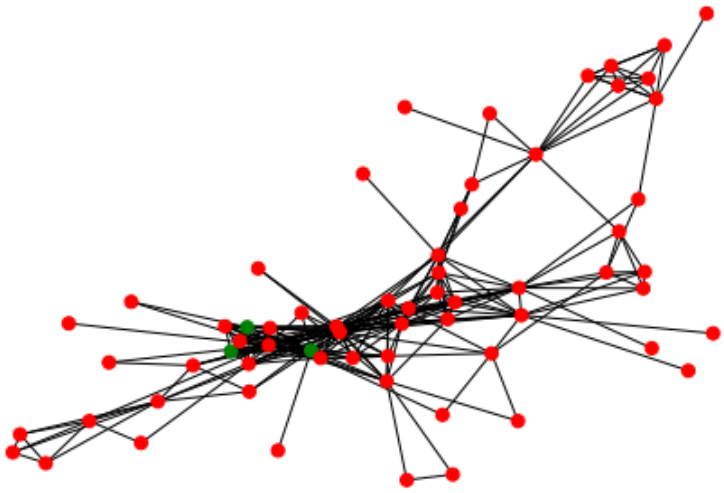


Figure 23: The most important terrorists in the network. The green nodes is the 3 most influence terrorists in the network

12.4 Compare the Terrorists returned by the two algorithm

```
# Evaluating Reachability and Expected Spread
print("spread", spread(G, top_terrorists))
# spread 52.08

# Greedy Influence Maximization
print("spread", total_spread)
#spread 52.830000000000005
```

The first algorithm: **Evaluating Reachability and Expected Spread**:

- Terrorists: 13, 14, 15
- Spread: 52.08

The second algorithm: **Influence Maximization Greedy Algorithm**:

- Terrorists: 14, 11, 10
- Spread: 52.83

We can see that the solution return by the Influence Maximization Greedy Algorithm is slightly better than the answer returned by evaluating Reachability and Expected Spread. However, the difference is very small, and they share one common terrorist: 14.