Experiment report: Social Network Analysis

Done by student: **Bylkova Kristina (伯汀娜), 1820249049**
Course: **Big Data Analysis Technology**
Date: **November 2024**

Task 1

Question: What is Social Network Analysis? For a single node, what can we analyze in Social Network? For the structure of Social Network, what can we analyze for Social Network? Explain all the metrics with example.

Answer:

Social Network Analysis is a crucial method used to examine the structure and dynamics of social networks by analyzing the relationships and interactions among individuals, groups, or entities. It applies graph theory to model real-world networks, making it a powerful tool in different fields such as sociology, business, biology, and computer science.

For a single node, we can analyze:

1. Degree Centrality: the number of connections (edges) a node has in a network. Example: many people know superstars (high degree centrality);

2. Betweenness Centrality: the proportion of the number of paths passing through a node to the total number of shortest paths. Example: intersections in a traffic network that connect different areas (higer betweeness centrality);

3. Closeness Centrality: the average shortest paths from a node to all other nodes in the network. Example: shopping malls are usually located in areas where more residents can reach them quickly (high clossness centrality);

4. Eigenvector Centrality: a reflection of the sum of the centrality scores of its neighbors. Example: people who are associated with influential people are usually also influential (high eigenvector centrality).

For the structure of Social Network, we can analyze:

1. Average degree & Average weighted degree: the average of the degrees of all nodes in the network & the average of the weights of all nodes in the network;

2. Density: the ratio of the actual number of edges to the maximum possible number of edges. Example: a high density of social networks helps employees access more career opportunities and resources;

3. Clustering: the degree to which a node's neighbors are connected to each other. Example: recommend related products or services to users with common interests. These people have high clustering coefficient;
4. Diameter: the longest shortest path (distance) between any two nodes in a network. Example: in a small diameter social network, a piece of news or message may be quickly known by all nodes. In a larger diameter network, the speed of information transmission is hindered.
5. Average distance: the average shortest path length between any two nodes. Example: in a residential community, if the average distance is small, the community is relatively close, and the communication and interaction between residents are more frequent and convenient.

Task 2

Question: Compared Social Network with other Network, what special characteristics does the social Network have?

Answer:

Characteristics:

1. Small Worlds: information spreads quickly and neighbors of nodes are often connected to each other, forming a tight local group;
2. Scale-Free: each time a new node is added, the new node establishes a connection with the existing node. New nodes are more likely to connect to existing nodes with higher degrees.That is, ”popular nodes” become more popular, forming Hub nodes;
3. Core-Periphery: core nodes are closely connected to form a group. They can interact frequently with other core nodes. Periphery nodes have some connections with the core, but have few or no connections between periphery nodes.

Experiment 1: Community Detection with NetworkX in Python

Objective: work with NetworkX, a Python library designed for the creation, manipulation, and study of complex networks. Run a community detection algorithm and observe how different communities emerge within a given graph.

Code:

```python
import networkx as nx
import matplotlib.pyplot as plt
from networkx.algorithms import community

# Create graph G
G = nx.Graph()  # Undirected graph without multiple edges
G.add_node(1)
G.add_nodes_from([2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])
G.add_edges_from([(1, 2), (4, 5), (5, 6), (2, 4), (1, 3), (2, 3),
                  (6, 7), (7, 8), (8, 9), (9, 10), (5, 10), (3, 7),
                  (1, 11), (11, 12), (12, 4), (12, 13), (13, 6),
                  (13, 14), (14, 7), (14, 15), (9, 15)])

# Use Girvan-Newman algorithm to divide communities
communities_generator = community.girvan_newman(G)
top_level_communities = next(communities_generator)
community_mapping = {node: i for i, community in enumerate(top_level_communities) for node in community
    }

# Sets the color of the node, based on the community division
node_colors = [community_mapping[node] for node in G.nodes]

# Visualize the network and mark communities with different colors
plt.figure(figsize=(8, 6))
pos = nx.spring_layout(G)  # Plot using the spring layout
nx.draw(G, pos, with_labels=True, node_color=node_colors, cmap=plt.cm.tab10, node_size=500, font_size
    =10)
plt.title("Community Detection in Modified Graph")
plt.show()

# Print the community division results
print("The results of community segmentation: ")
for i, community in enumerate(top_level_communities):
    print(f"Community {i+1}: {sorted(community)}")
```

Result:

During the experiment, undirected graph without multiple edges was created and Girvan-Newman algorithm was used to divide communities. As a result, I got a graph, which visualize the network and mark communities with different colors.

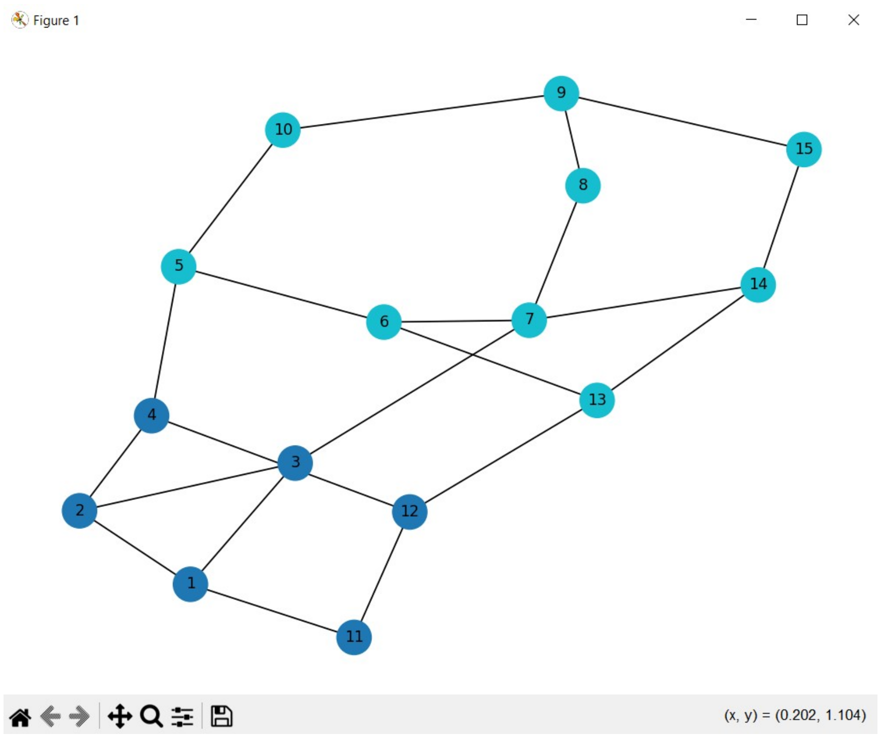
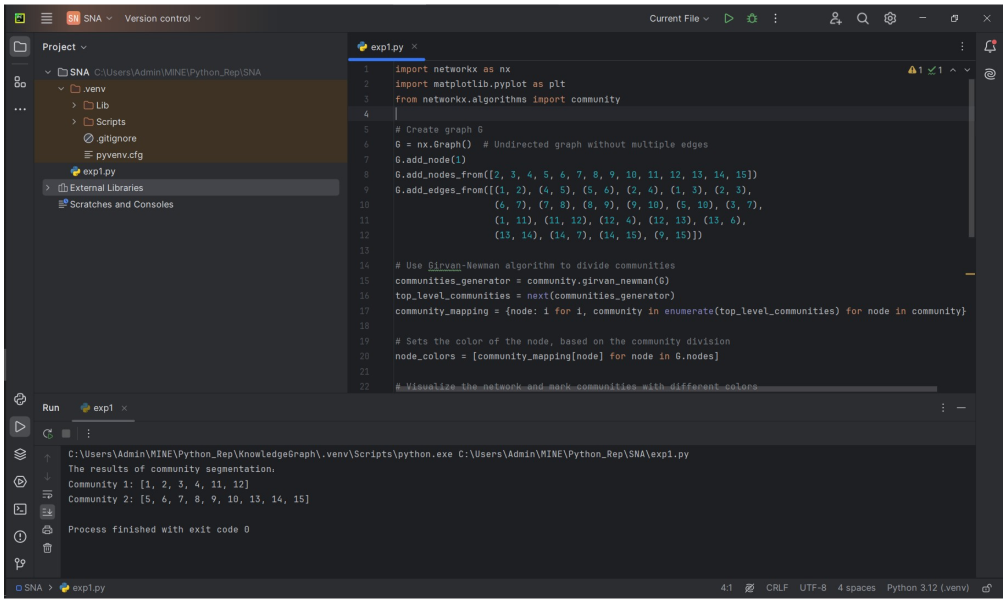


Figure 1: Sample result graph generated by the code



```python
import networkx as nx
import matplotlib.pyplot as plt
from networkx.algorithms import community

# Create graph G
G = nx.Graph()  # Undirected graph without multiple edges
G.add_node(1)
G.add_nodes_from([2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])
G.add_edges_from([(1, 2), (4, 5), (5, 6), (2, 4), (1, 3), (2, 3),
                  (6, 7), (7, 8), (8, 9), (9, 10), (5, 10), (3, 7),
                  (1, 11), (11, 12), (12, 4), (12, 13), (13, 6),
                  (13, 14), (14, 7), (14, 15), (9, 15)])

# Use Girvan-Newman algorithm to divide communities
communities_generator = community.girvan_newman(G)
top_level_communities = next(communities_generator)
community_mapping = {node: i for i, community in enumerate(top_level_communities) for node in community}

# Sets the color of the node, based on the community division
node_colors = [community_mapping[node] for node in G.nodes]

# Visualize the network and mark communities with different colors
```

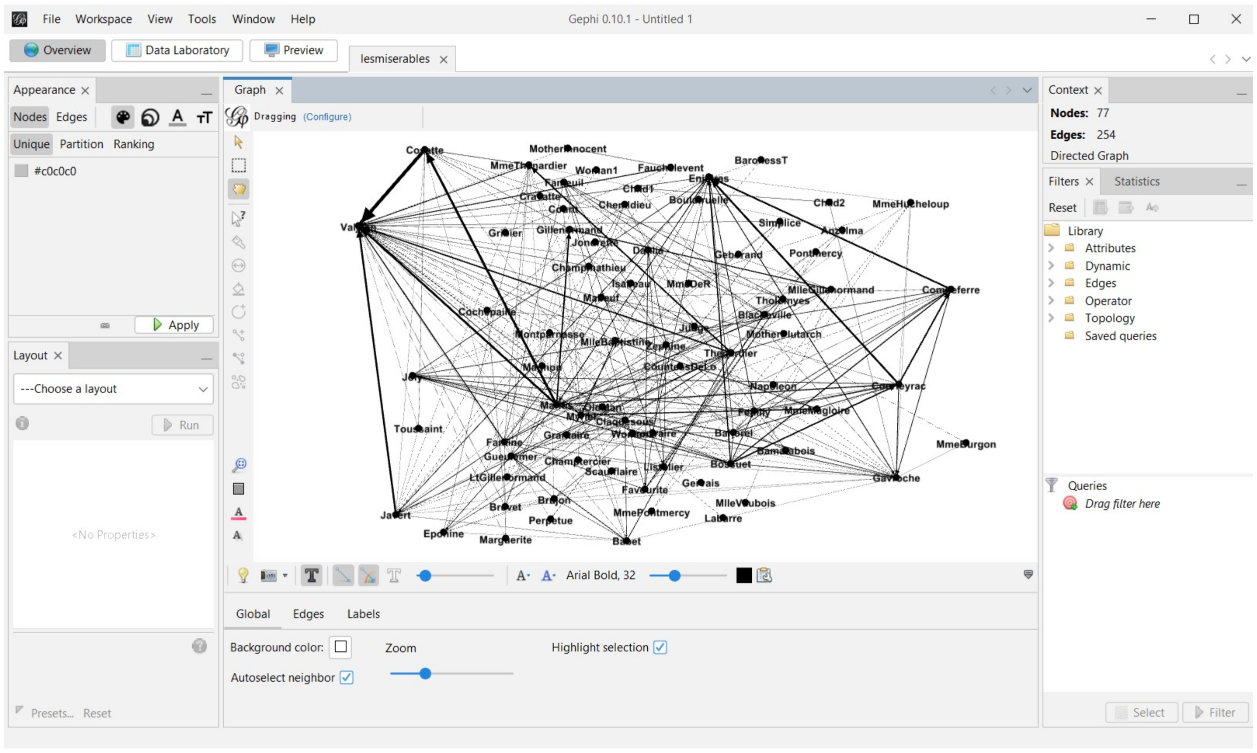
```
C:\Users\Admin\MINE\Python_Rep\KnowledgeGraph\.venv\Scripts\python.exe C:\Users\Admin\MINE\Python_Rep\SNA\exp1.py
The results of community segmentation:
Community 1: [1, 2, 3, 4, 11, 12]
Community 2: [5, 6, 7, 8, 9, 10, 13, 14, 15]

Process finished with exit code 0
```

Figure 2: Result of community segmentation

Experiment 2: SNA Graph Analysis Using Gephi

Objective: use Gephi, a visualization tool for graph analysis, to load and explore a dataset, compute key graph statistics (such as PageRank and modularity), and adjust the graph's appearance for better readability.

Step 1: Load the Dataset



Gephi 0.10.1 - Untitled 1

lesmiserables

#c0c0c0

---Choose a layout

Arial Bold, 32

Background color: ☐ Zoom Highlight selection ✓

Autoselect neighbor ✓

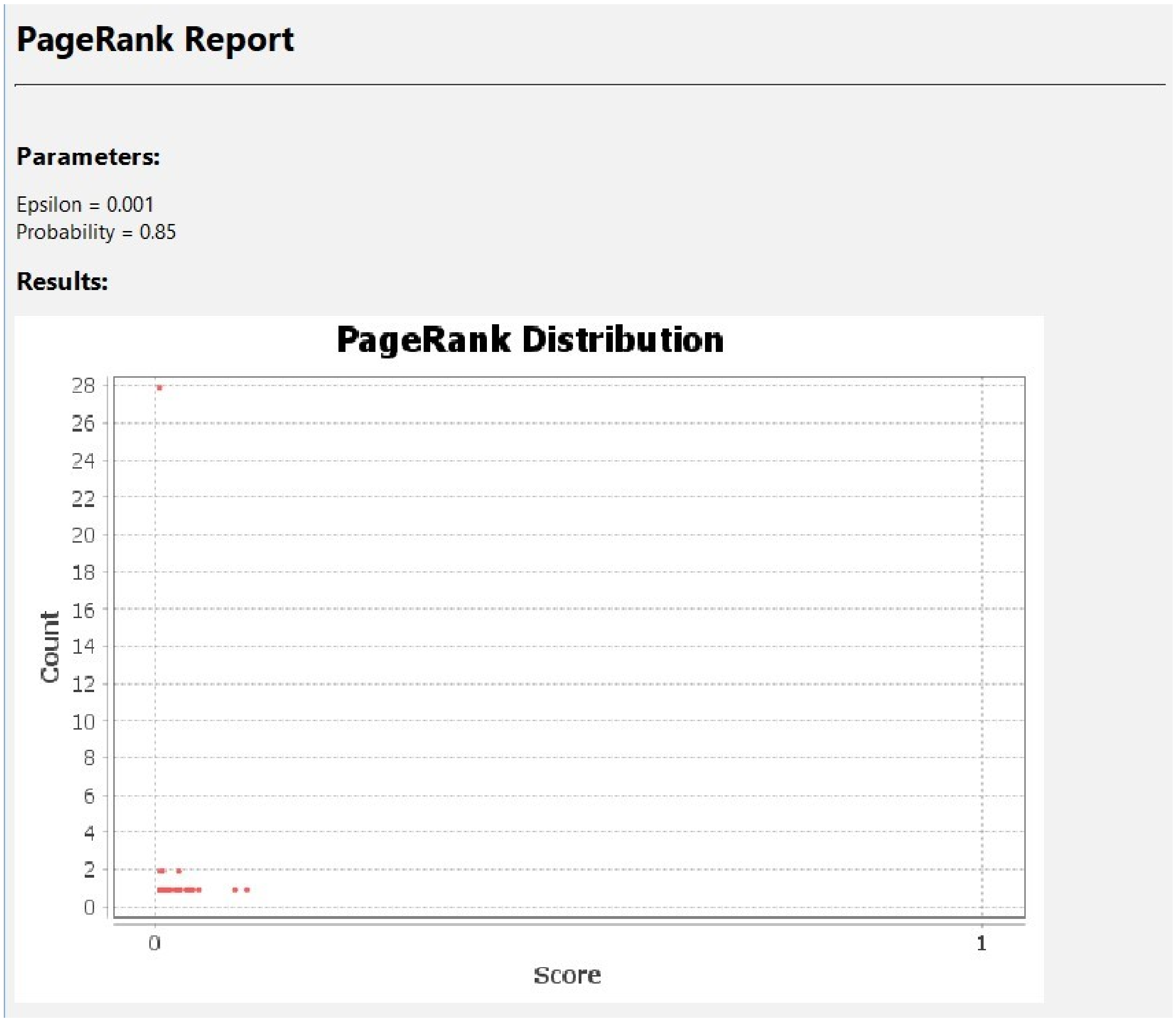
Nodes: 77

Edges: 254

Directed Graph

Figure 3: Graph in Gephi

Step 2: Analyze Statictics



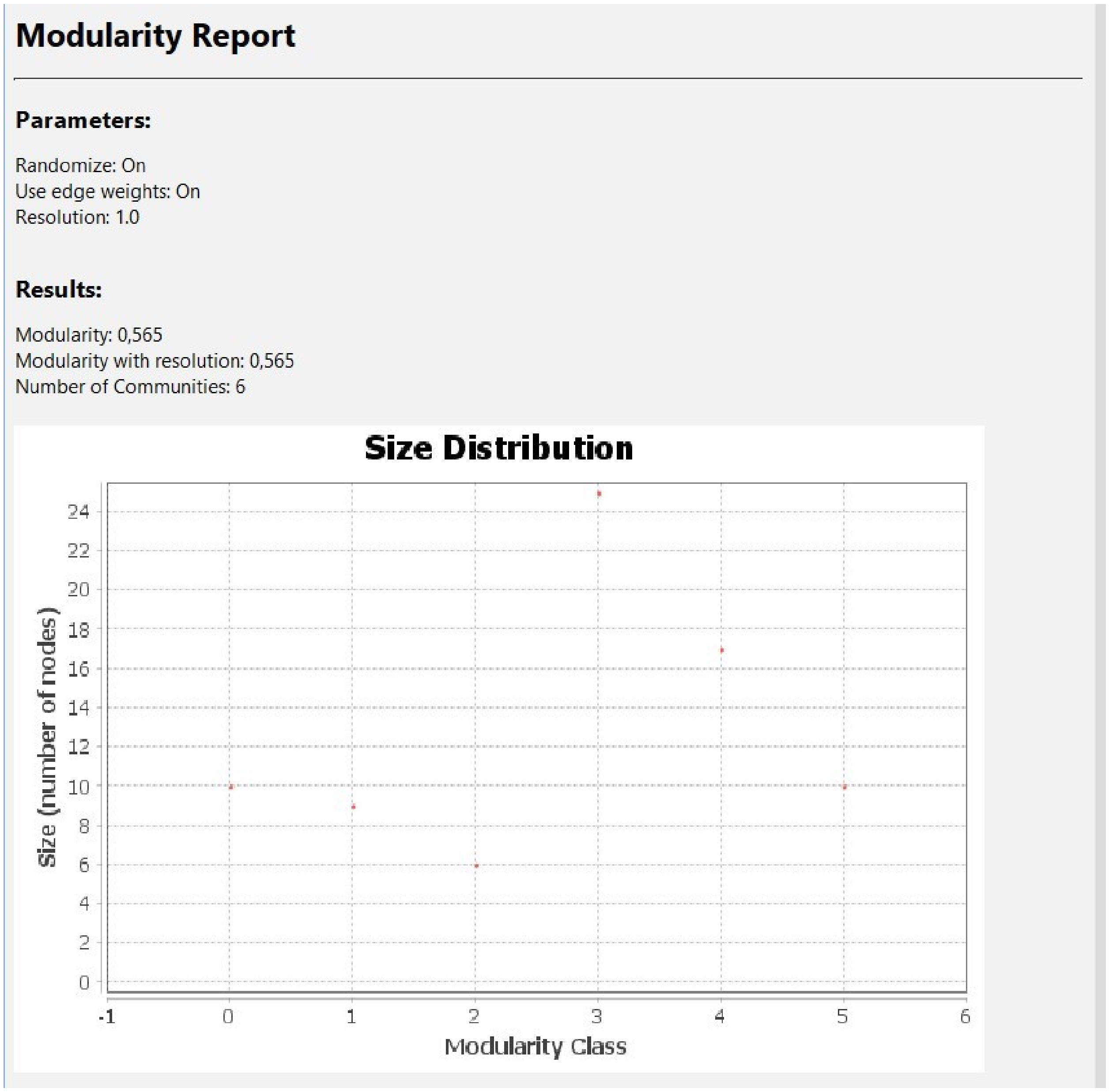
PageRank Report

Parameters:

Epsilon = 0.001
Probability = 0.85

Results:

Figure 4: PageRank



Modularity Report

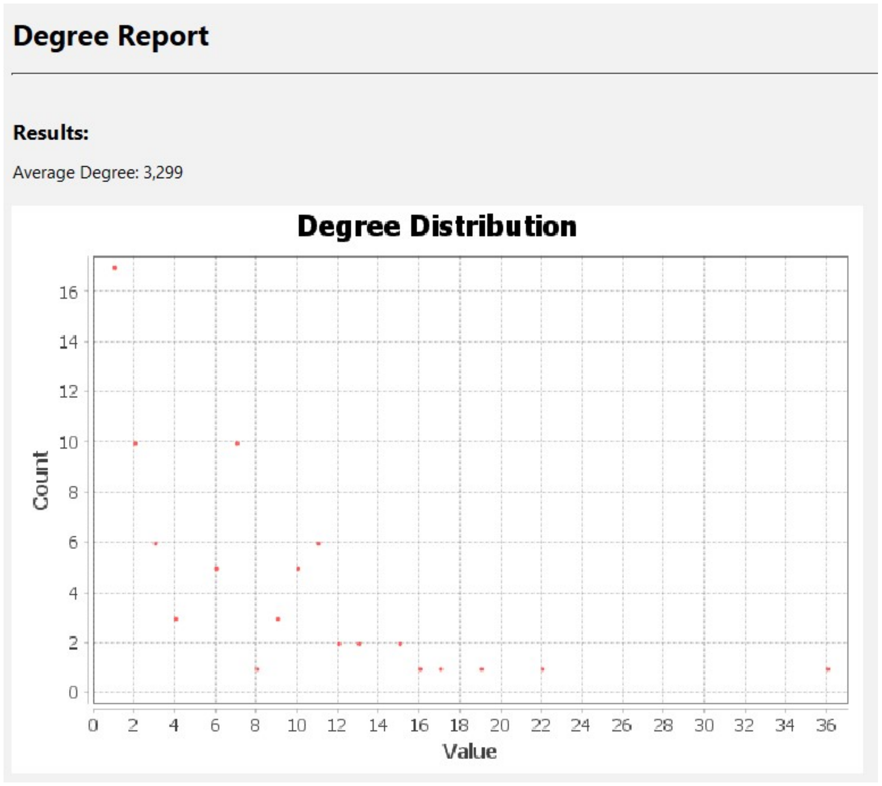
Parameters:

Randomize: On
Use edge weights: On
Resolution: 1.0

Results:

Modularity: 0,565
Modularity with resolution: 0,565
Number of Communities: 6

Figure 5: Modularity


Degree Report

Results:

Average Degree: 3,299

Figure 6: Average Degree: part 1

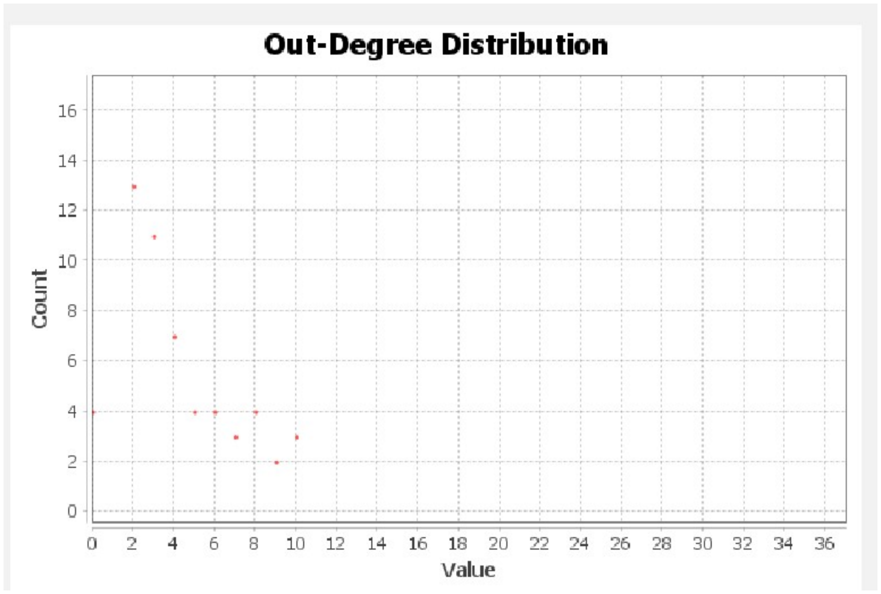
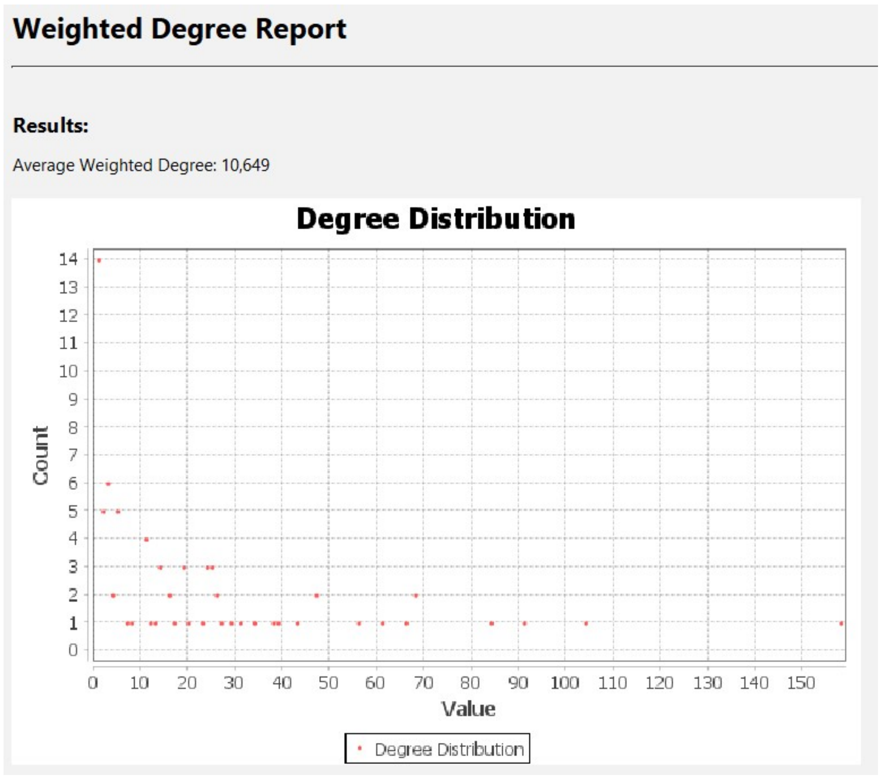


Figure 7: Average Degree: part 2


Weighted Degree Report

Results:

Average Weighted Degree: 10,649

Figure 8: Average Weighted Degree: part 1

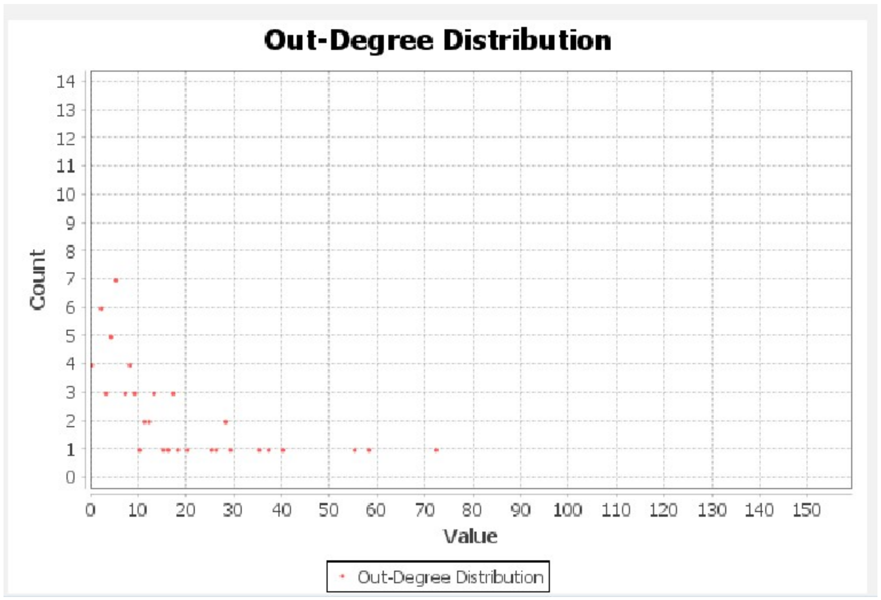
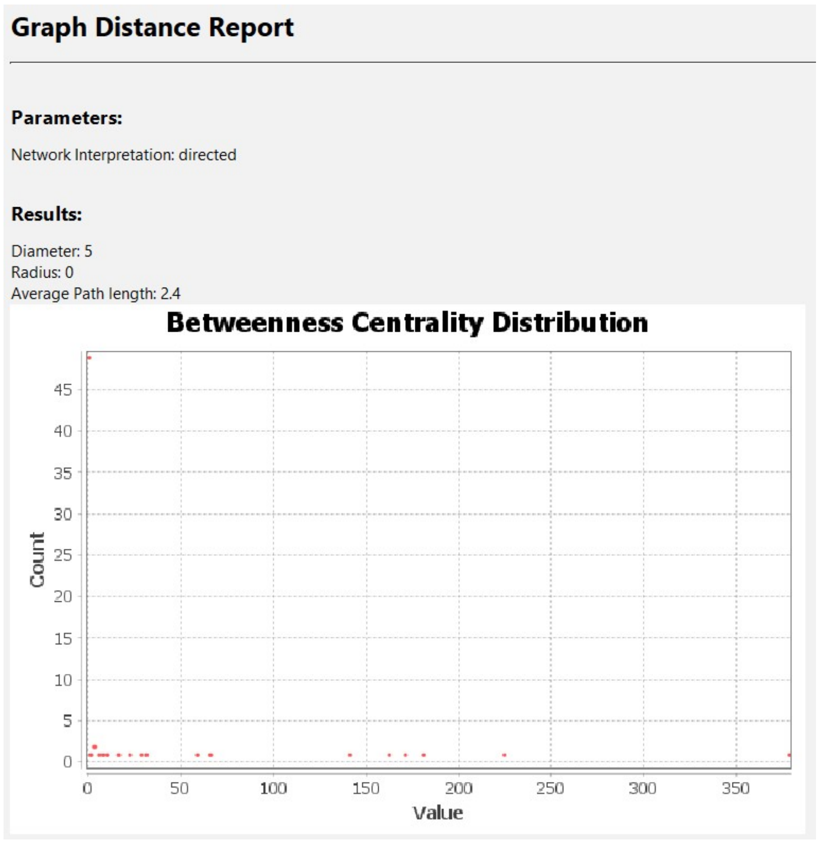


Figure 9: Average Weighted Degree: part 2



Graph Distance Report

Parameters:

Network Interpretation: directed

Results:

Diameter: 5
Radius: 0
Average Path length: 2.4

Figure 10: Network Diameter: part 1

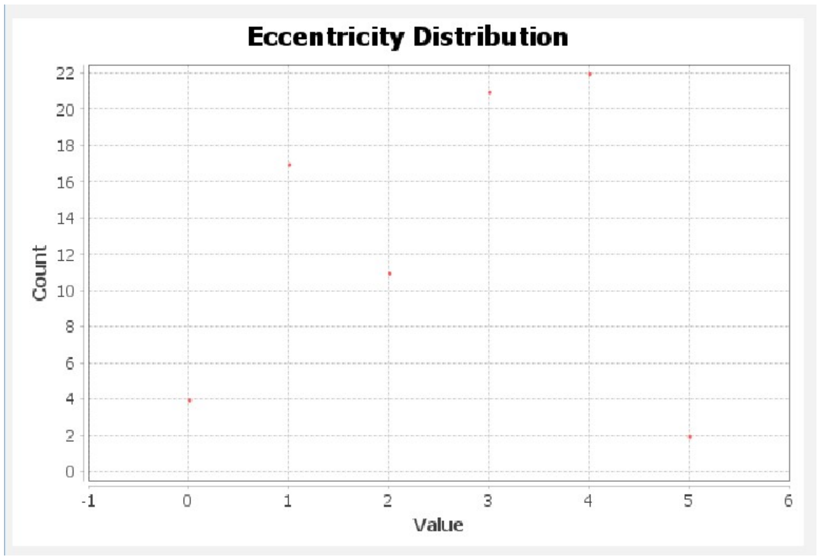
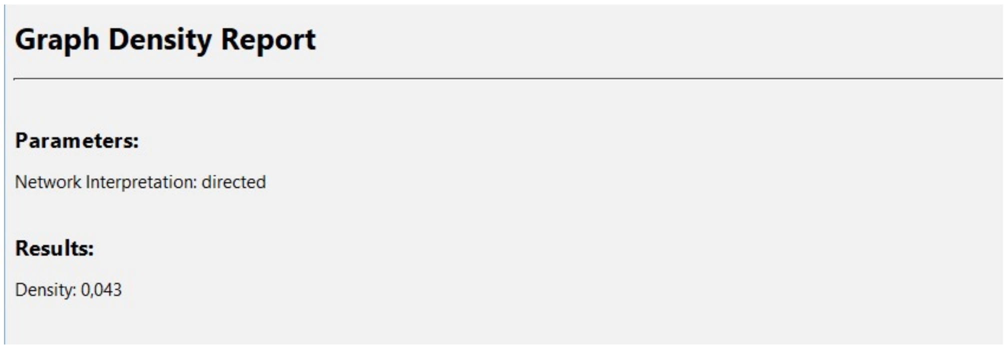


Figure 11: Network Diameter: part 2



Graph Density Report

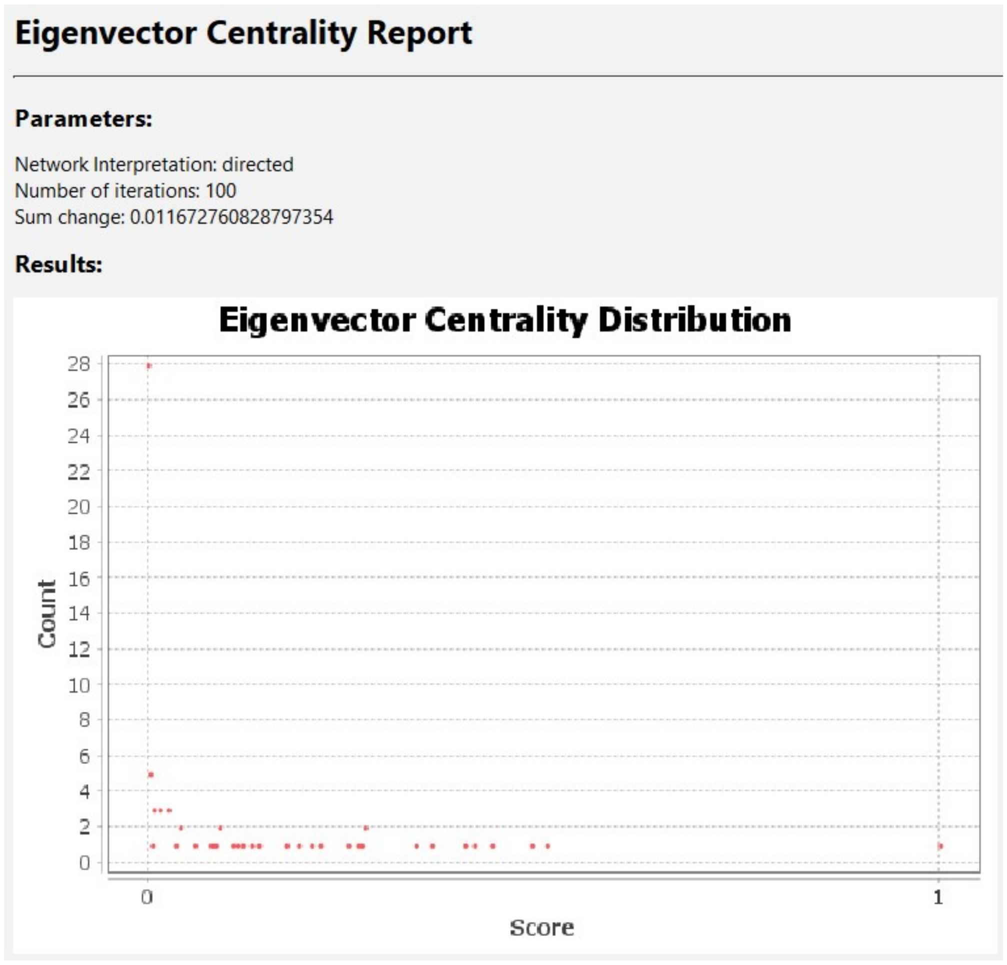
Parameters:

Network Interpretation: directed

Results:

Density: 0,043

Figure 12: Graph Density



Eigenvector Centrality Report

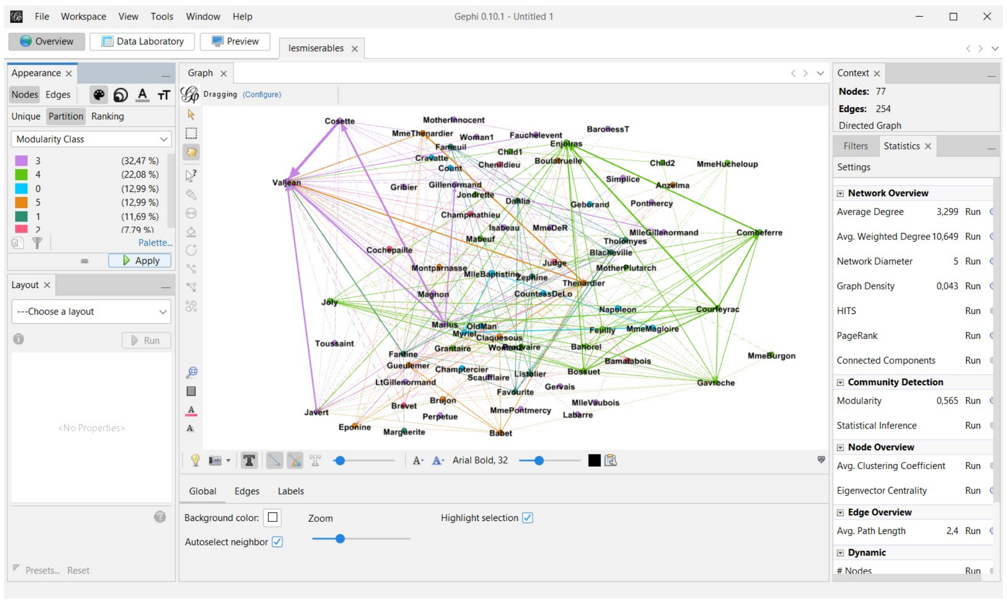
Parameters:

Network Interpretation: directed
Number of iterations: 100
Sum change: 0.011672760828797354

Results:

Figure 13: Eigenvector Centrality

Step 3: Preview Appearance Adjustment


Gephi 0.10.1 - Untitled 1

Appearance: Nodes — Partition — Modularity Class

🟪	3	(32,47 %)
🟩	4	(22,08 %)
🟦	0	(12,99 %)
🟧	5	(12,99 %)
🟩	1	(11,69 %)
🟥	2	(7,79 %)

Layout: ---Choose a layout

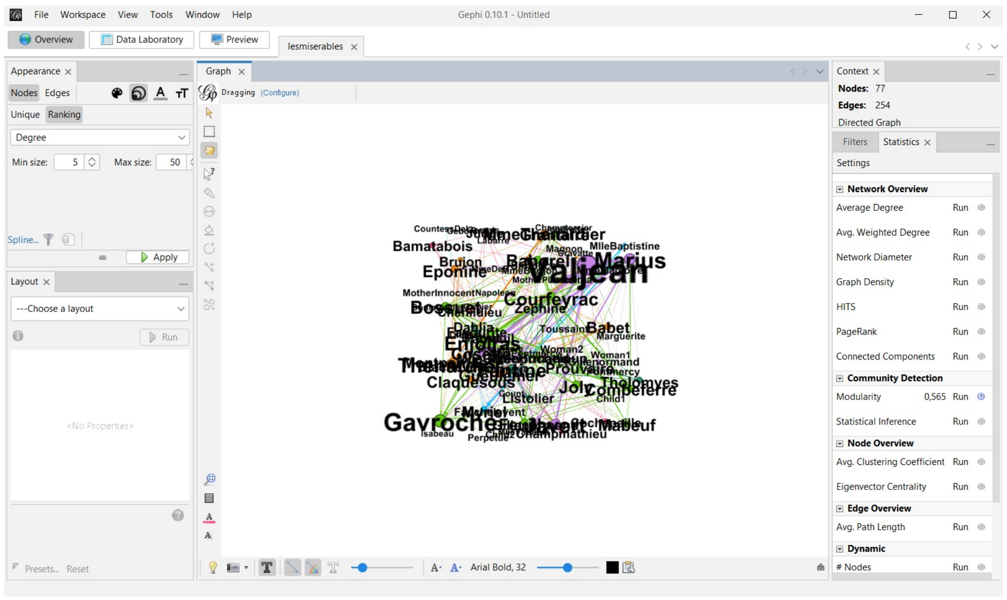
Graph: Global — Background color: ☐ Zoom Highlight selection ✓ Autoselect neighbor ✓

Context — Nodes: 77 Edges: 254 Directed Graph

Statistics

Network Overview		
Average Degree	3,299	Run
Avg. Weighted Degree	10,649	Run
Network Diameter	5	Run
Graph Density	0,043	Run
HITS		Run
PageRank		Run
Connected Components		Run
Community Detection		
Modularity	0,565	Run
Statistical Inference		Run
Node Overview		
Avg. Clustering Coefficient		Run
Eigenvector Centrality		Run
Edge Overview		
Avg. Path Length	2,4	Run
Dynamic		
# Nodes		Run

Figure 14: Nodes adjustments


Gephi 0.10.1 - Untitled

Appearance: Nodes — Ranking — Degree

Min size: 5 Max size: 50

Layout: ---Choose a layout

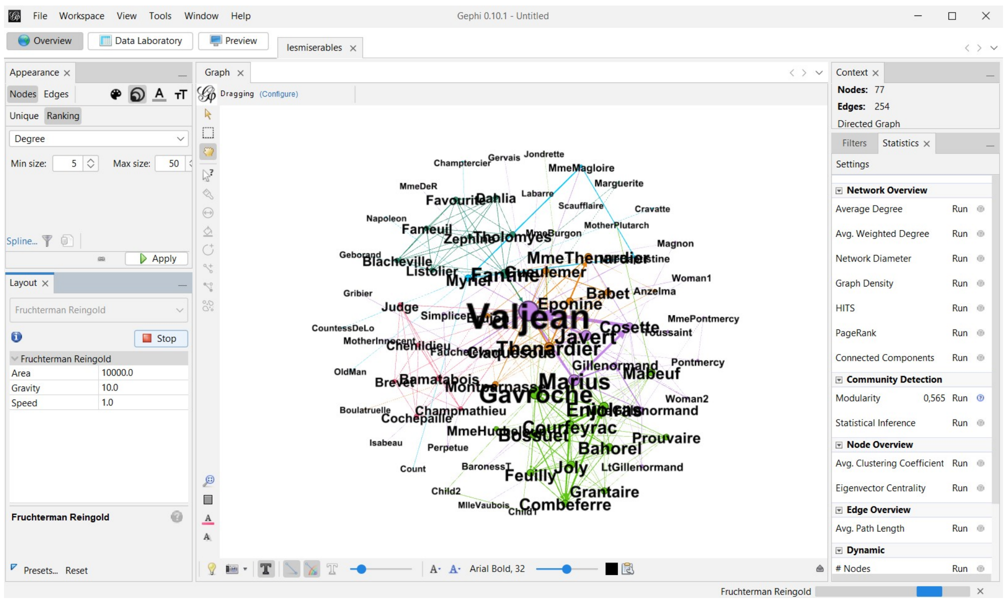
Context — Nodes: 77 Edges: 254 Directed Graph

Statistics

Network Overview		
Average Degree		Run
Avg. Weighted Degree		Run
Network Diameter		Run
Graph Density		Run
HITS		Run
PageRank		Run
Connected Components		Run
Community Detection		
Modularity	0,565	Run
Statistical Inference		Run
Node Overview		
Avg. Clustering Coefficient		Run
Eigenvector Centrality		Run
Edge Overview		
Avg. Path Length		Run
Dynamic		
# Nodes		Run

Figure 15: Labels adjustments

Step 4: Select Layout



Degree

Min size: 5 Max size: 50

Fruchterman Reingold

Fruchterman Reingold	
Area	10000.0
Gravity	10.0
Speed	1.0

Nodes: 77
Edges: 254
Directed Graph

Network Overview
- Average Degree
- Avg. Weighted Degree
- Network Diameter
- Graph Density
- HITS
- PageRank
- Connected Components

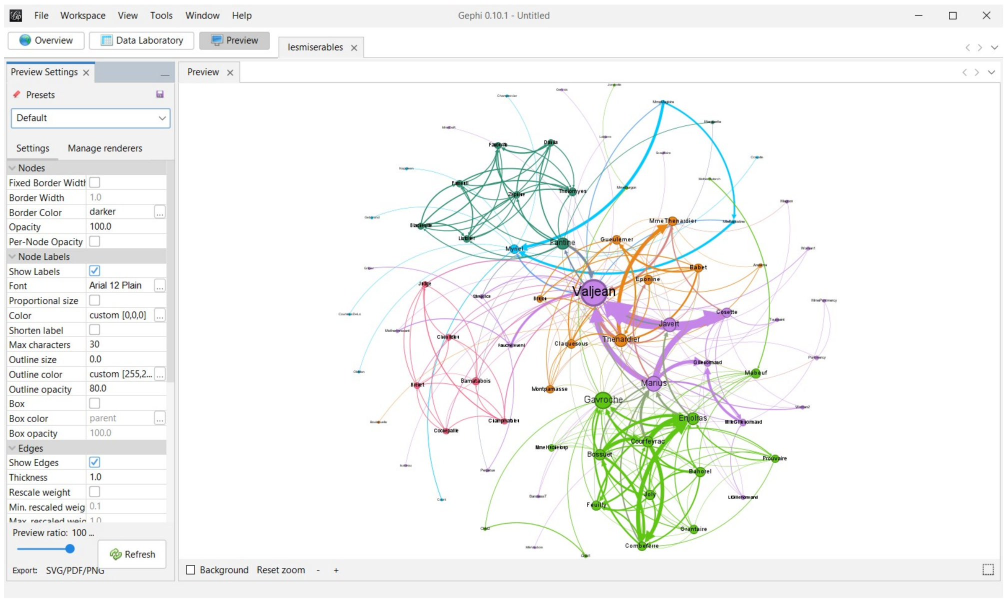
Community Detection
- Modularity 0,565
- Statistical Inference

Node Overview
- Avg. Clustering Coefficient
- Eigenvector Centrality

Edge Overview
- Avg. Path Length

Dynamic
- # Nodes

Figure 16: Fruchterman Reingold layout



Default

Nodes	
Fixed Border Width	☐
Border Width	1.0
Border Color	darker
Opacity	100.0
Per-Node Opacity	☐
Node Labels	
Show Labels	✓
Font	Arial 12 Plain
Proportional size	☐
Color	custom [0,0,0]
Shorten label	☐
Max characters	30
Outline size	0.0
Outline color	custom [255,2...
Outline opacity	80.0
Box	☐
Box color	parent
Box opacity	100.0
Edges	
Show Edges	✓
Thickness	1.0
Rescale weight	☐
Min. rescaled weig	0.1
Max. rescaled weig	1.0

Preview ratio: 100 ...

Figure 17: Adjusting preview settings

Result:

During this experiment, I loaded the dataset into Gephi, used analyzing tools to get statistics of the graph and customized the graph’ s appearance. I also understood how to select graph layouts and saved the final visualization of the graph.

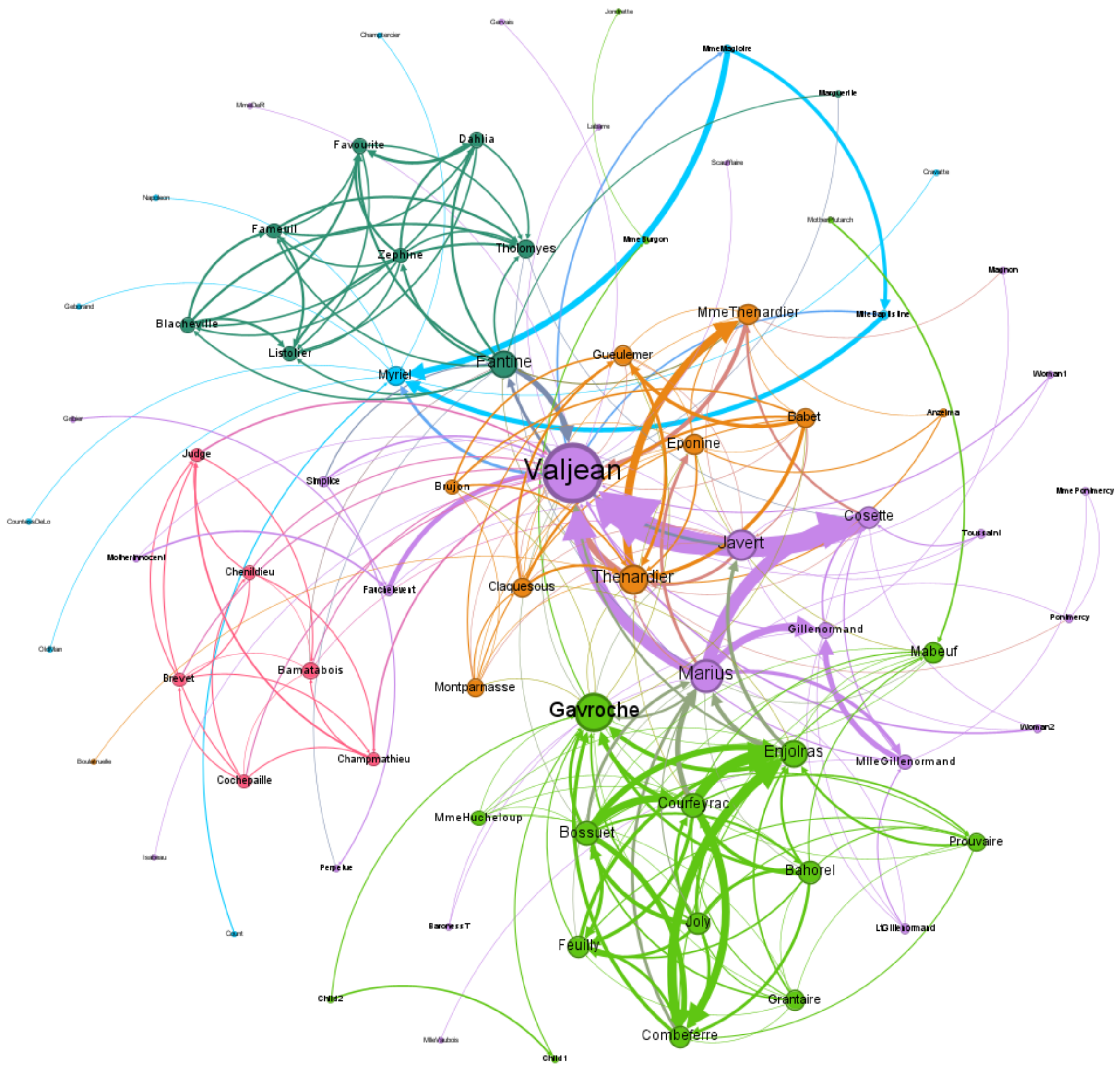


Figure 18: Final graph saved in PNG