Network Analysis and Visualization with Python

Veera Muangsin

Python Libraries for Network Analysis & Visualization

- NetworkX
  - Network analysis
  - https://networkx.org/
- Igraph
  - Network analysis
  - https://igraph.org/python/
- PyVis
  - interactive visualization
  - https://pyvis.readthedocs.io/
- GraphViz
  - static visualization
  - https://www.graphviz.org/
- Plotly
  - interactive visualization
  - https://plotly.com/python/network-graphs/

Python Libraries for Network Analysis & Visualization

Purpose	Package	Description
Network manipulation and analysis	NetworkX	Core Python library for network analysis and manipulation https://networkx.org/
Network manipulation and analysis	igraph	High-performance graph analysis library for large graphs https://igraph.org/python/
Interactive visualization	PyVis	Interactive network visualization based on javascript https://pyvis.readthedocs.io/
static visualization	GraphViz	High quality static graph visualization https://www.graphviz.org/
Interactive visualization	Plotly	Full-featured data visualization library with network graph support https://plotly.com/python/network-graphs/

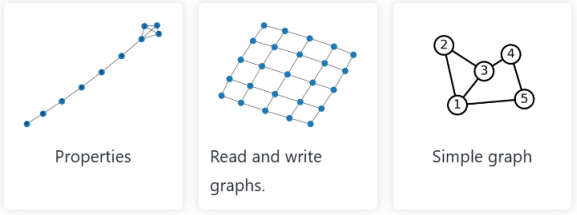
Data analysis and visualization projects

- Les Misérables Social Network
  - https://towardsdatascience.com/les-mis%C3%A9rables-social-network-analysis-using-marimo-notebooks-and-the-networkx-python-library-%EF%B8%8F-%EF%B8%8F-3f433216412f
- Bibliometric Network
  - https://www.kaggle.com/code/kruttika17/bibliometric-network-analysis-topic-modelling
  - https://github.com/mclevey/metaknowledge_article_supplement/blob/master/4_network_analysis.ipynb
- Game of Thrones
  - https://www.kaggle.com/code/mmmarchetti/game-of-thrones-network-analysis/notebook

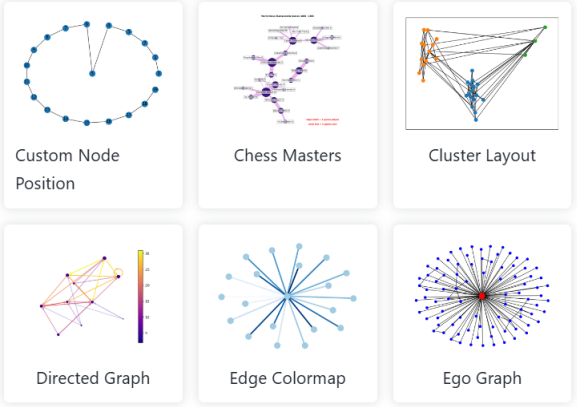
NetworkX

- NetworkX is a python package for creation, manipulation, and study of the structure, dynamics, and functions of complex networks.
  - https://networkx.org/
  - https://networkx.org/documentation/stable/auto_examples/index.html
  - https://networkx.org/documentation/stable/tutorial.html
- `pip install networkx`

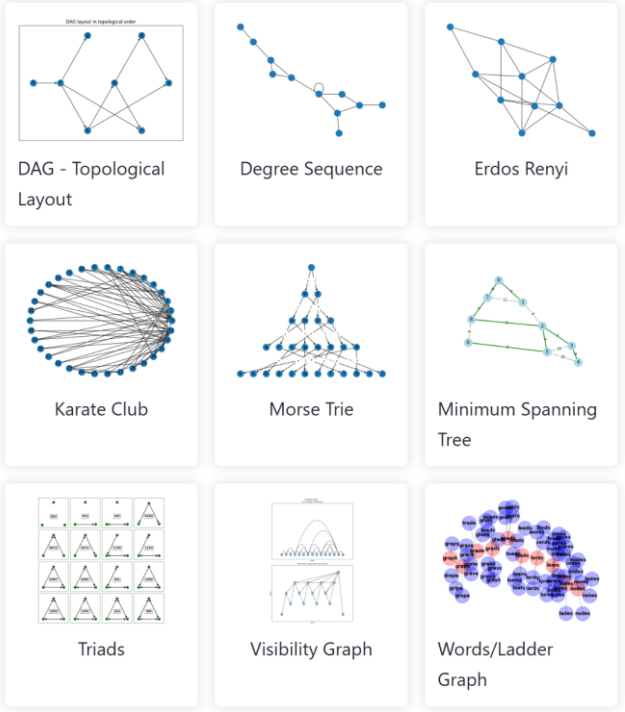
Basic



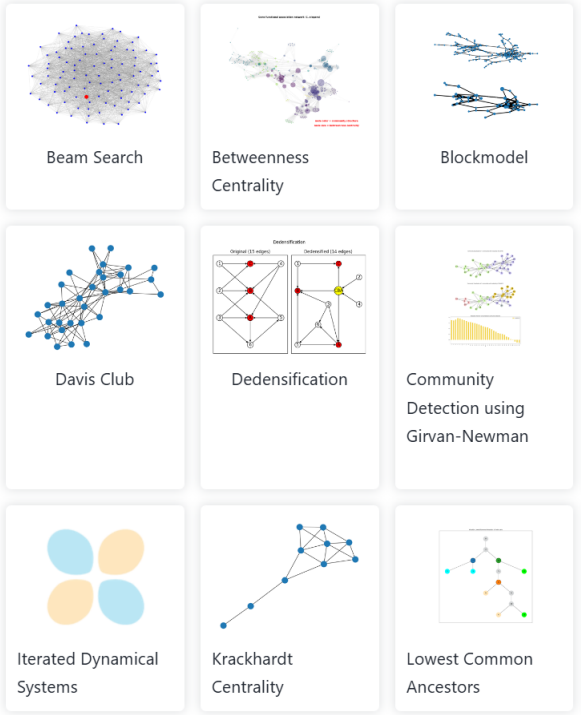
Drawing



Graph



Algorithms



NetworkX Graph Components

- Graph Objects
  - The containers for networks

```
G = nx.Graph()          # Undirected graph
G = nx.DiGraph()        # Directed graph
```

- Node
  - The vertices of the graph

```
G.add_node(1)
G.add_nodes_from([2, 3, 4])
```

- Edges
  - The connections between nodes

```
G.add_edge(1, 2)
G.add_edges_from((2, 3), (3, 4)]
```

- Attributes
  - Data attached to nodes or edges

```
G.graph['name'] = 'My Graph'
G.nodes[1]['label'] = 'Start'
G.nodes[1]['color’] = 'blue'
G.edges[1, 2]['weight'] = 1.0
```

Graph Generators

- NetworkX provides functions to create various types of network graphs
- https://networkx.org/documentation/stable/reference/generators.html

Function	Return
complete_graph(n)	complete graph with n nodes
erdos_renyi_graph(n, p) binomial_graph(n, p)	Erdős and A. Rényi random graph with *n* nodes, where *p* is the probability for edge creation.
watts_strogatz_graph(n, k, p)	Watts–Strogatz small-world graph with *n* nodes. Each node is joined with its *k* nearest neighbors in a ring topology and *p* is the probability of rewiring each edge
barabasi_albert_graph(n, m)	Barabási–Albert scale-free graph with *n* nodes grown by attaching new nodes each with *m* edges that are preferentially attached to existing nodes with high degree.
karate_club_graph()	Zachary's Karate Club graph
les_miserables_graph()	Coappearance network of characters in the novel Les Miserables.

Graph Properties

```
G.number_of_nodes()          # Number of nodes
G.number_of_edges()          # Number of edges
list(G.nodes())              # List all nodes
list(G.edges())              # List all edges
G.degree(1)                  # Degree of node 1
G.neighbors(1)               # Neighbors of node 1
```

Graph Visualization

- NetworkX provides only basic graph visualization since its main goal is graph analysis.

- Layout
  - Node positioning algorithms for graph drawing.

```
pos = nx.xxx_layout()
pos = nx.spring_layout()
pos = nx.circular_layout()
pos = nx.forceatlas2_layout(G)
```

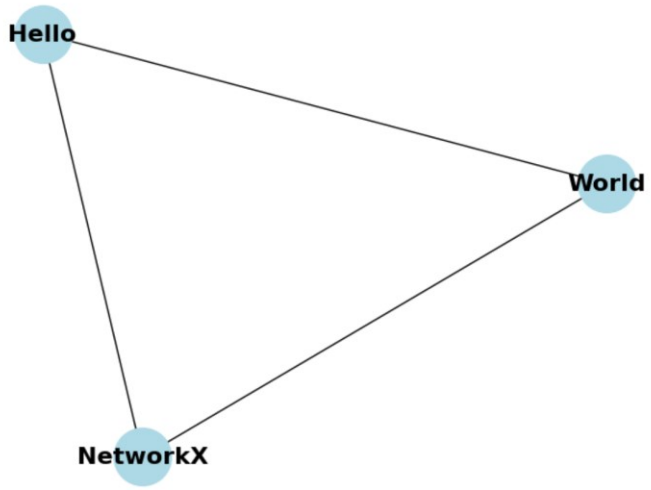
  - https://networkx.org/documentation/stable/reference/drawing.html#module-networkx.drawing.layout

- Draw
  - Draw graph with Matplotlib

```
nx.draw(G)              # default layout is spring
nx.draw(G, pos=pos)
```

NetworkX: Simple Example

```python
import networkx as nx

# Create a new empty graph
G = nx.Graph()

# Add some nodes
G.add_node("Hello")
G.add_node("World")
G.add_node("NetworkX")

# Add edges between the nodes
G.add_edge("Hello", "World")
G.add_edge("World", "NetworkX")
G.add_edge("Hello", "NetworkX")

# Draw the graph with spring_layout
pos = nx.spring_layout(G)
nx.draw(
    G,
    pos,
    with_labels=True,      # Show node labels
    node_color='lightblue',# Make nodes light blue
    node_size=1500,        # Make nodes a bit bigger
    font_size=16,          # Make labels readable
    font_weight='bold'     # Make labels stand out
)

# Print some basic information about the graph
print(f"Number of nodes: {G.number_of_nodes()}")
print(f"Number of edges: {G.number_of_edges()}")
print(f"Nodes: {list(G.nodes())}")
print(f"Edges: {list(G.edges())}")
```




```
Number of nodes: 3
Number of edges: 3
Nodes: ['Hello', 'World', 'NetworkX']
Edges: [('Hello', 'World'), ('Hello', 'NetworkX'), ('World', 'NetworkX')]
```

Centrality

- NetworkX provides functions to calculate network centrality
- https://networkx.org/documentation/stable/reference/algorithms/centrality.html

degree_centrality(G)

closeness_centrality(G)

betweenness_centrality(G)

eigenvector_centrality(G)

Community Detection

- NetworkX provides functions for community detection
- https://networkx.org/documentation/stable/reference/algorithms/community.html

louvain_communities(G)

greedy_modularity_communities(G)

PyVis

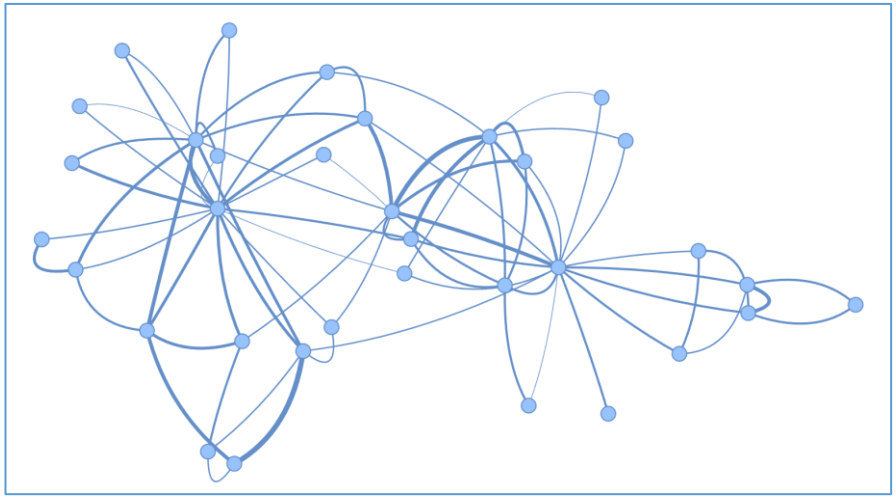
- PyVis is a python package package for interactive visualization based on JavaScript.
  - https://pyvis.readthedocs.io/
- `pip install pyvis`
- `from pyvis.network import Network`
- A typical use case is to create a network and set its layout using NetworkX, then converting to a PyVis network for interactive visualization.

PyVis: simple example

Create html

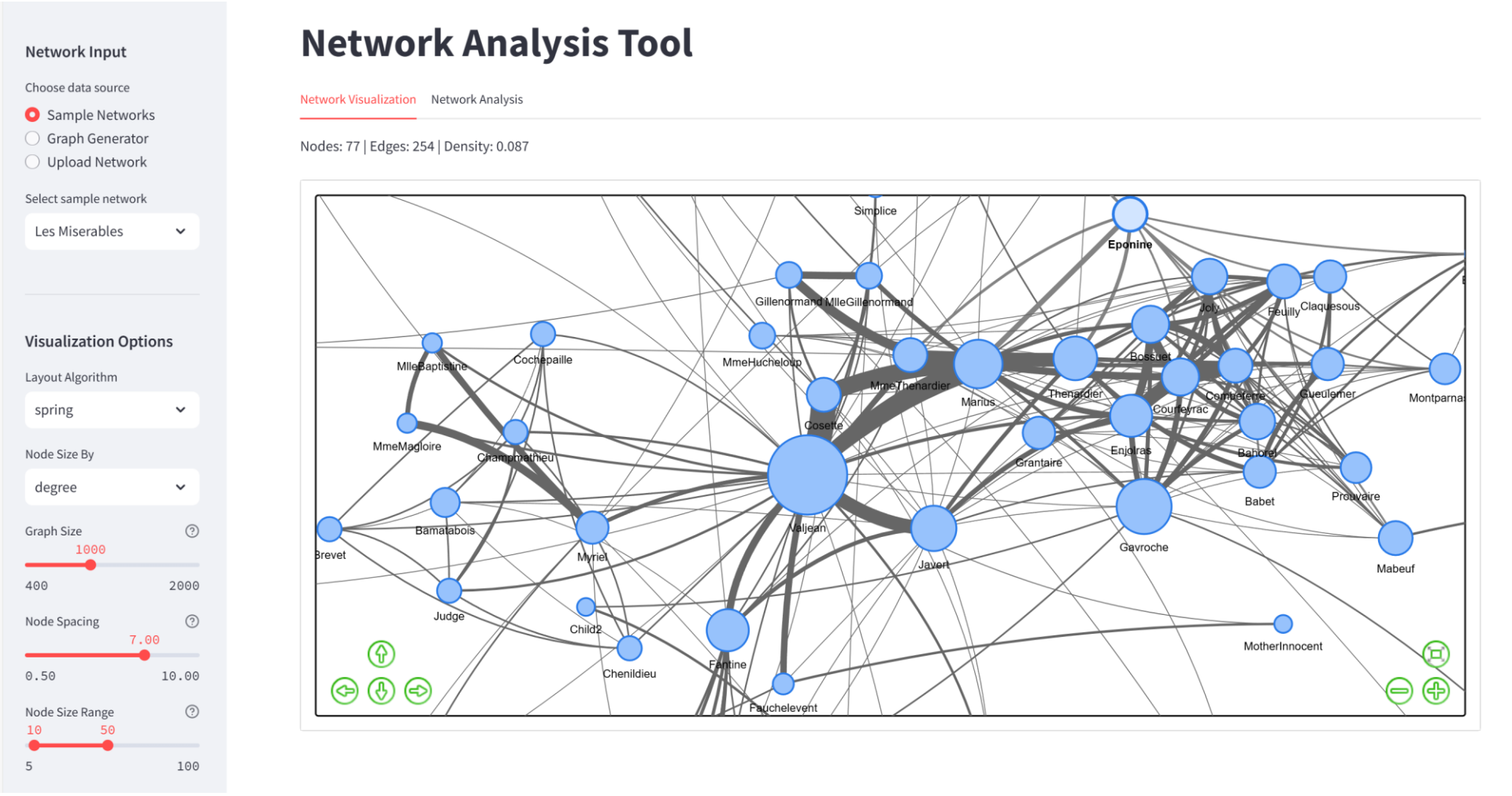
```
import networkx as nx
from pyvis.network import Network

G = nx.karate_club_graph()
net = Network(notebook=True)  # to run on Jupyter Notebook
net.from_nx(G)
net.show('karate_club.html')
```

```
import networkx as nx
from pyvis.network import Network

G = nx.karate_club_graph()
net = Network()
net.from_nx(G)
net.write_html('karate_club.html')
```



Streamlit + NetworkX + PyVis

streamlit_network_analysis.py



Streamlit + NetworkX + PyVis

streamlit_network_analysis.py

Network Input

Choose data source

- (•) Sample Networks
- ( ) Graph Generator
- ( ) Upload Network

Select sample network

Les Miserables

Visualization Options

Layout Algorithm

spring

Node Size By

degree

Graph Size

1000

400 2000

Node Spacing

2.00

0.50 10.00

Node Size Range

10 50

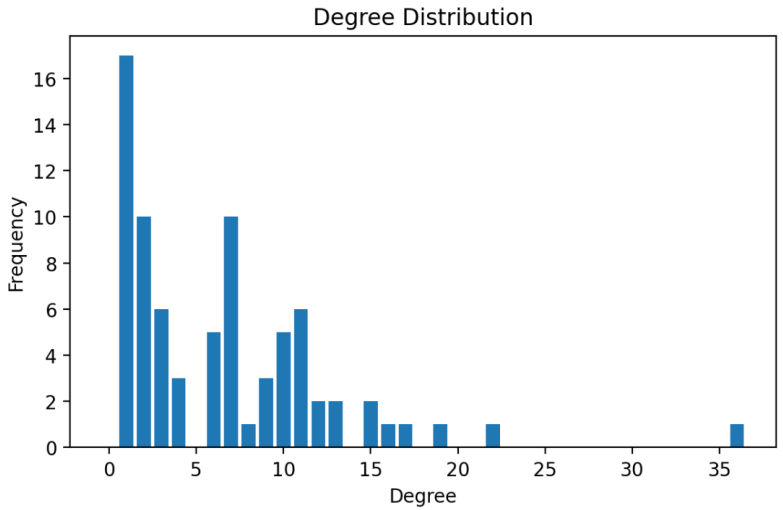
5 100

Network Analysis

Basic Statistics

Nodes	Edges	Density	Diameter
77	254	0.087	5

Degree Distribution



Centrality Analysis

	Node	↓ Degree	Betweenness	Closeness	PageRank
10	Valjean	0.4737	0.57	0.6441	0.0996
48	Gavroc	0.2895	0.1651	0.5135	0.0283
55	Marius	0.25	0.132	0.5315	0.0517
27	Javert	0.2237	0.0543	0.517	0.0268
25	Thenar	0.2105	0.0749	0.517	0.0357
58	Enjolra	0.1974	0.0426	0.481	0.0366
23	Fantine	0.1974	0.1296	0.4606	0.0272
64	Bossue	0.1711	0.0308	0.475	0.0262
62	Courfey	0.1711	0.0053	0.4	0.033
65	Joly	0.1579	0.0022	0.3938	0.0177