

Social Network Analysis

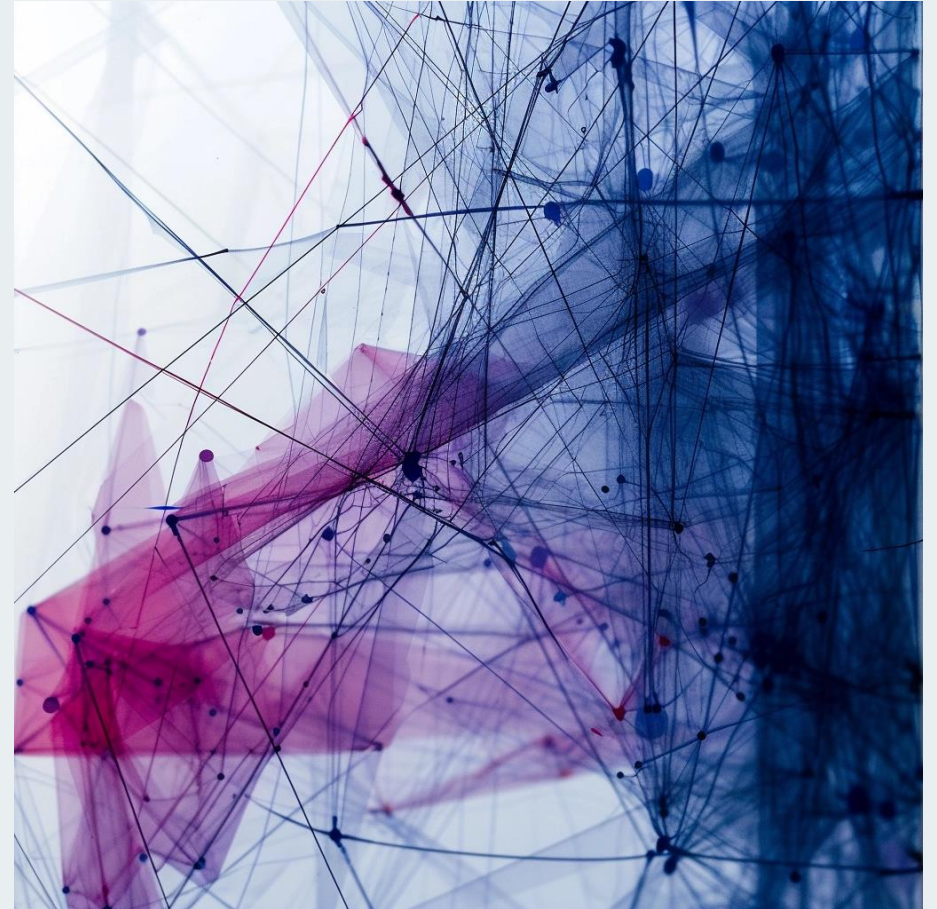
Network Visualization

Dr. Chun-Hsiang Chan

Department of Geography,
National Taiwan Normal University

Outline

- Why We Need Graph Visualization?
- Task 1: Create a Random Graph
- Task 2: Visualize with Networkx
- Task 3: Visualize with pyVis
- Task 4: Visualize with Jaal
- Task 5: Visualize with Gephi



Why We Need Graph Visualization?

- Graph visualization is used to demonstrate the relationship between nodes with node-edge quantification.
- Different to map visualization, the closer nodes does not indicate that these nodes have geographical proximity, only means that there relationship might be stronger than others.
- However, different graph visualization may illustration different story of your graph; therefore, you need to choose the best way to present your graph.



Why We Need Graph Visualization?

- In today's class, we are going to introduce graph-based visualization, e.g., networkx, pyvis, and Jaal.
- If you want to know how to plot a graph onto a map, please refer to Python Programming lectures. Basically, we use `plt.plot` and `plt.scatter` to draw the straight lines and nodes based on the attributes and geometry of provided GeoDataFrame, respectively.



Task 1: Create a Random Graph

– First, we need to import necessary packages:

```
# import packages  
import numpy as np  
import pandas as pd  
import networkx as nx  
import matplotlib.pyplot as plt
```



Task 1: Create a Random Graph

- Create a random graph with **Erdos Renyi**:

```
# create Erdos Renyi random graph
```

```
# a directed graph with 50 nodes and 0.2 edge-link probability
```

```
G = nx.erdos_renyi_graph(50, 0.2, seed=21, directed=False)
```



Task 2: Visualize with Networkx

- **Networkx** is a useful Python graph packages, which is convenient for social/complex network analysis via numerous quantitative indicators.
- **Installation**

```
pip install networkx
```



Task 2: Visualize with Networkx

– Configuration for visualization

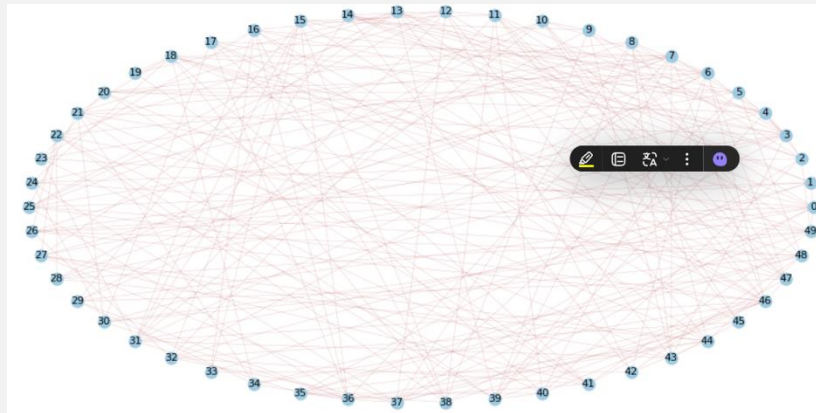
```
# set options
options = {
    "font_size": 8,
    "node_size": 100,
    "node_color": "#A0CBE2",
    "edge_color": "brown",
    "linewidths": 0.1,
    "width": 0.08,
}
```



Task 2: Visualize with Networkx

– Circular Style

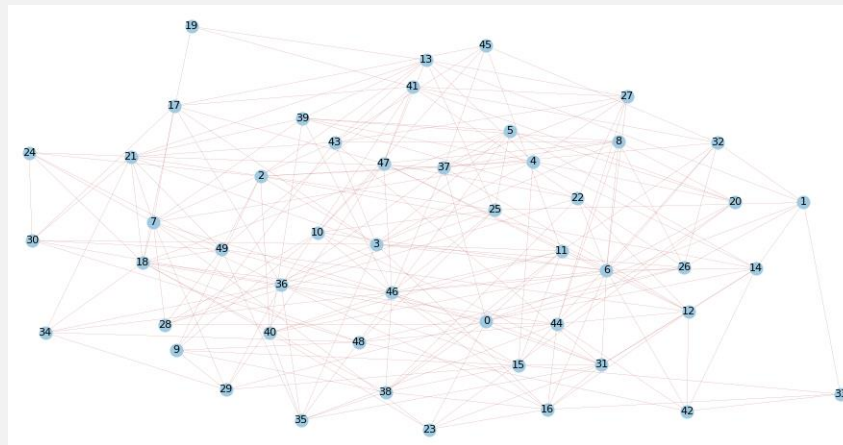
```
# circular
plt.subplots(figsize=[12,6], dpi=100)
nx.draw_circular(G, **options, with_labels = True)
ax = plt.gca()
ax.margins(0.001)
plt.axis("off")
plt.show()
```



Task 2: Visualize with Networkx

– Kamada Kawai Style

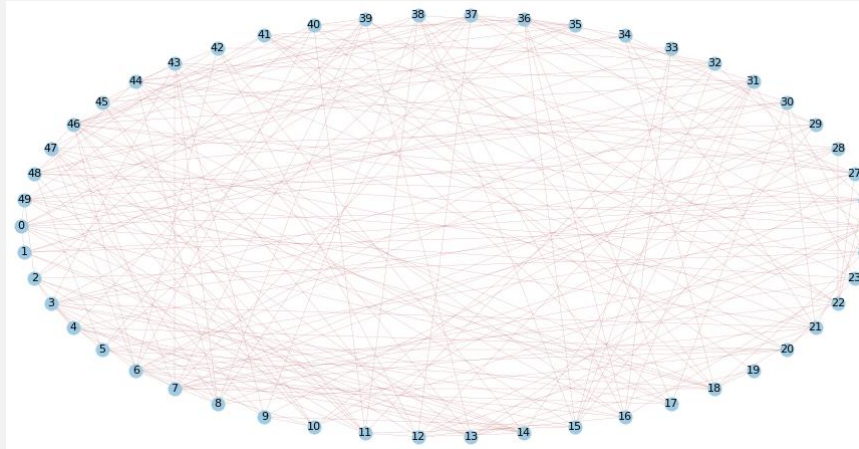
```
# kamada kawai  
plt.subplots(figsize=[12,6], dpi=100)  
nx.draw_kamada_kawai(G, **options, with_labels = True)  
ax = plt.gca()  
ax.margins(0.001)  
plt.axis("off")  
plt.show()
```



Task 2: Visualize with Networkx

– Shell Style

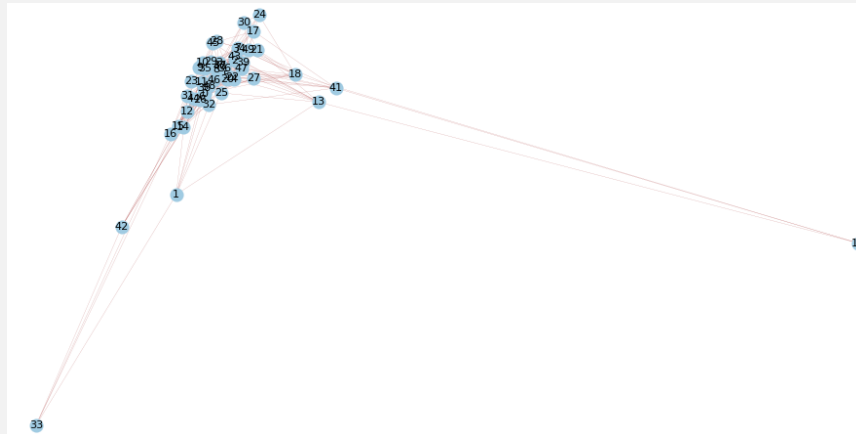
```
# Shell
plt.subplots(figsize=[12,6], dpi=100)
nx.draw_shell(G, **options, with_labels = True)
ax = plt.gca()
ax.margins(0.001)
plt.axis("off")
plt.show()
```



Task 2: Visualize with Networkx

– Spectral Style

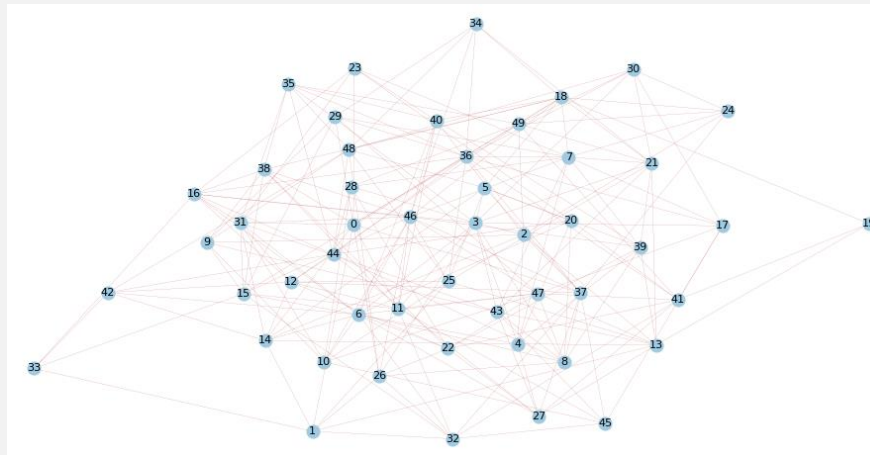
```
# Spectral
plt.subplots(figsize=[12,6], dpi=100)
nx.draw_spectral(G, **options, with_labels = True)
ax = plt.gca()
ax.margins(0.001)
plt.axis("off")
plt.show()
```



Task 2: Visualize with Networkx

– Spring Style

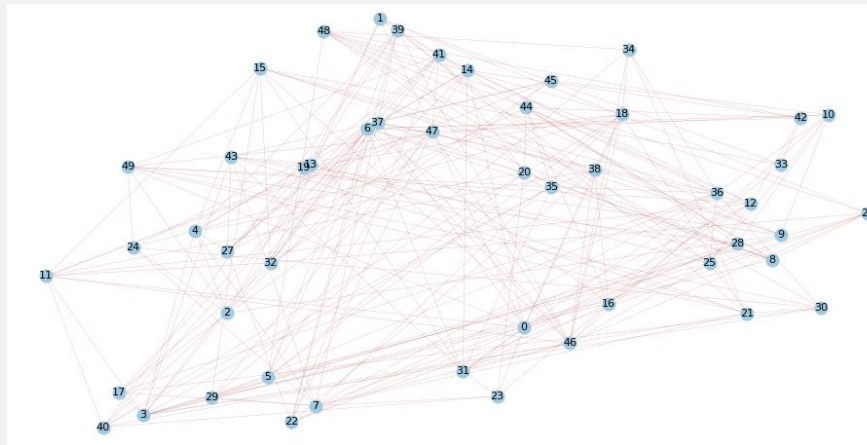
```
# Spectral  
plt.subplots(figsize=[12,6], dpi=100)  
nx.draw_spring(G, **options, with_labels = True)  
ax = plt.gca()  
ax.margins(0.001)  
plt.axis("off")  
plt.show()
```



Task 2: Visualize with Networkx

– Random Style

```
# Random
plt.subplots(figsize=[12,6], dpi=100)
nx.draw_random(G, **options, with_labels = True)
ax = plt.gca()
ax.margins(0.001)
plt.axis("off")
plt.show()
```



Task 3: Visualize with pyVis

- The developers of Networkx emphasize that the library is designed for quantifying and analyzing network structures and topological properties, rather than for visualization purposes.
- In many cases, however, visualization is essential for understanding interactions among nodes. Static visualization methods often fall short in effectively conveying these relationships.
- To address this limitation, we introduce PyVis, a dynamic visualization tool that enables interactive exploration of graphs. It can display networks directly within a Jupyter Notebook or export them as standalone HTML files for broader accessibility.



Task 3: Visualize with pyVis

– Installation

```
pip install pyvis
```

- Before starting to visualize our graph, pyvis only conducts its own node and edge format.



Task 3: Visualize with pyVis

- Import graph from network

```
# create vis network
```

```
visG = net.Network(notebook=True, height="800px", width="100%",  
                  bgcolor="#222222", font_color="white",  
                  filter_menu=True, select_menu=True,  
                  cdn_resources='remote')
```

```
# import graph from networkx
```

```
visG.from_nx(G)
```

```
# show
```

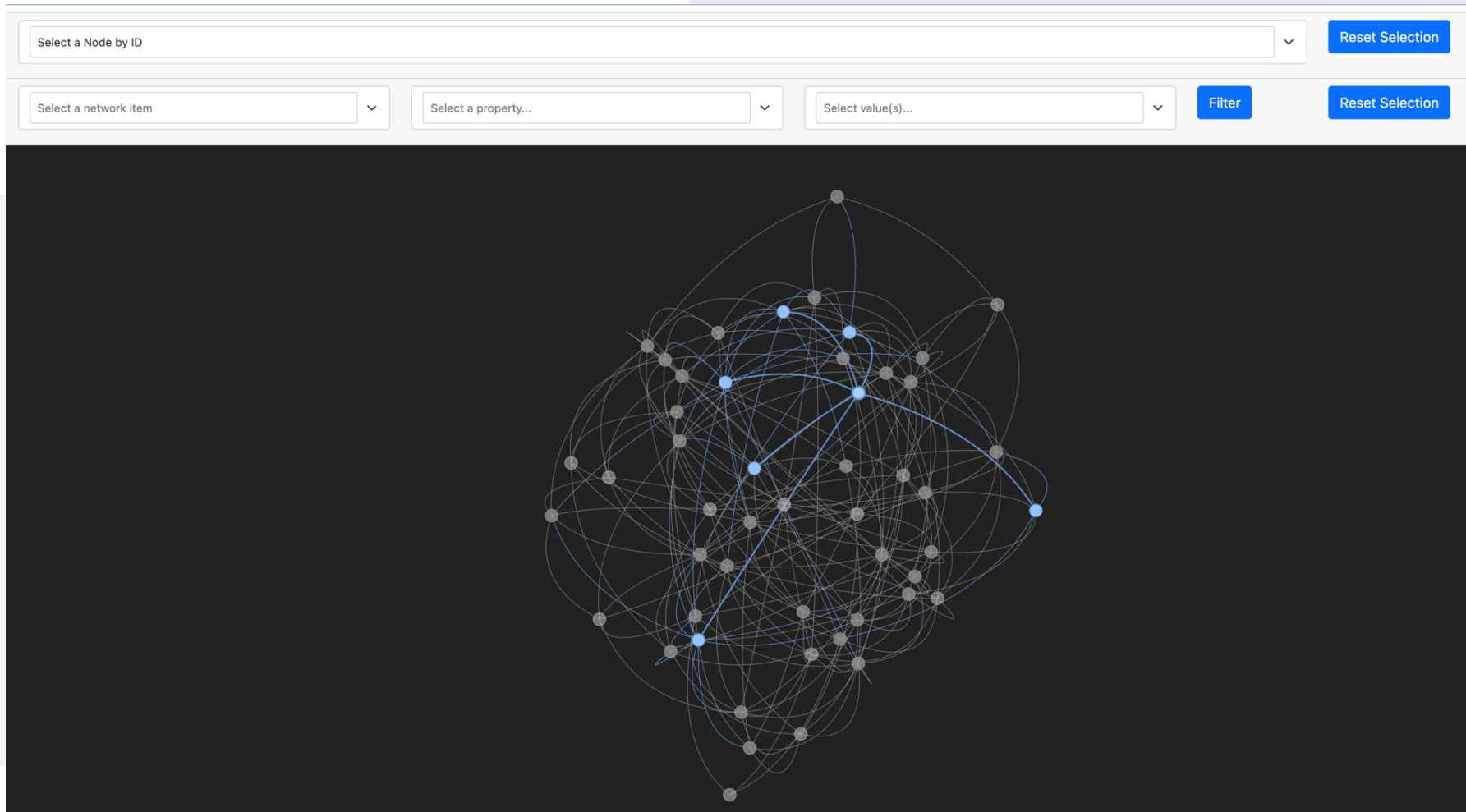
```
visG.show_buttons()
```

```
visG.show('pyvis_ex.html')
```



Task 3: Visualize with pyVis

– Results



Task 3: Visualize with pyVis

- Node Color and Size with Group Label and Importance
- To demonstrate the **size** attribute, we use a random number to assign the size to each node.

```
# extract networkx node to dataframe
```

```
G_nodes = pd.DataFrame.from_records(np.vstack(list(G.nodes)),  
                                     columns=['ID'])
```

```
# add size with random values
```

```
G_nodes['importance'] = abs(np.random.normal(15, 12,  
                                             G_nodes.shape[0]))
```



Task 3: Visualize with pyVis

- Node Color and Size with Group Label and Importance
- To demonstrate the **group** attribute, we use a random number to assign the group label to each node.

```
# add group with random values
```

```
G_nodes['group'] = np.random.randint(5, size=G_nodes.shape[0])+1
```

```
# add node label with random values
```

```
G_nodes['code'] = 'S_'+G_nodes['ID'].astype('str')
```

```
# preview data
```

```
G_nodes.head()
```

	ID	importance	group	code
0	0	22.095438	3	S_0
1	1	4.494944	2	S_1
2	2	10.828733	1	S_2
3	3	13.368127	3	S_3
4	4	24.870871	4	S_4



Task 3: Visualize with pyVis

- Edge Width with Weight
- To demonstrate the **edge** attribute, we use a random number to assign the weight to each edge.

```
# extract networkx edge to dataframe
```

```
G_edges = pd.DataFrame.from_records(np.vstack(list(G.edges)),  
columns=['source', 'target'])
```

```
# add weight with random values
```

```
G_edges['weight'] = abs(np.random.normal(100, 25,  
G_edges.shape[0]))**2/1000
```

	source	target	weight
0	0	1	5.494500
1	0	11	3.121376
2	0	14	8.873845
3	0	18	6.916639
4	0	23	14.928490



Task 3: Visualize with pyVis

– Assign Colors and Settings

```
color_map = {1 : '#CCFF00', 2 : '#CCCC33', 3 : '#CC9966', 4 : '#CC6699',  
             5 : '#CC33CC', 6 : '#CC00FF', 7 : '#9900FF', 8 : '#6600CC',  
             9 : '#660066', 10 : '#660033', 11 : '#CC99FF', 12 : '#FFCCFF',  
             13 : '#FFFFCC', 14 : '#FF3300', 15 : '#333333', 16 : '#666666'}  
  
options = {'edge_color': '#FFDEA2', 'width': .5, 'with_labels': False,  
          'font_weight': 'regular'}
```



Task 3: Visualize with pyVis

– Starting to Plot

```
# initialize
g = net.Network(height="800px", width="100%", bgcolor="#222222",
                font_color="white", filter_menu=True, select_menu=True,
                cdn_resources='remote')

# add node
for i in range(len(G_nodes['code'])):
    g.add_node(int(G_nodes['ID'][i]), size=int(G_nodes['importance'][i]),
               label=G_nodes['code'][i], group=int(G_nodes['group'][i]))
```



Task 3: Visualize with pyVis

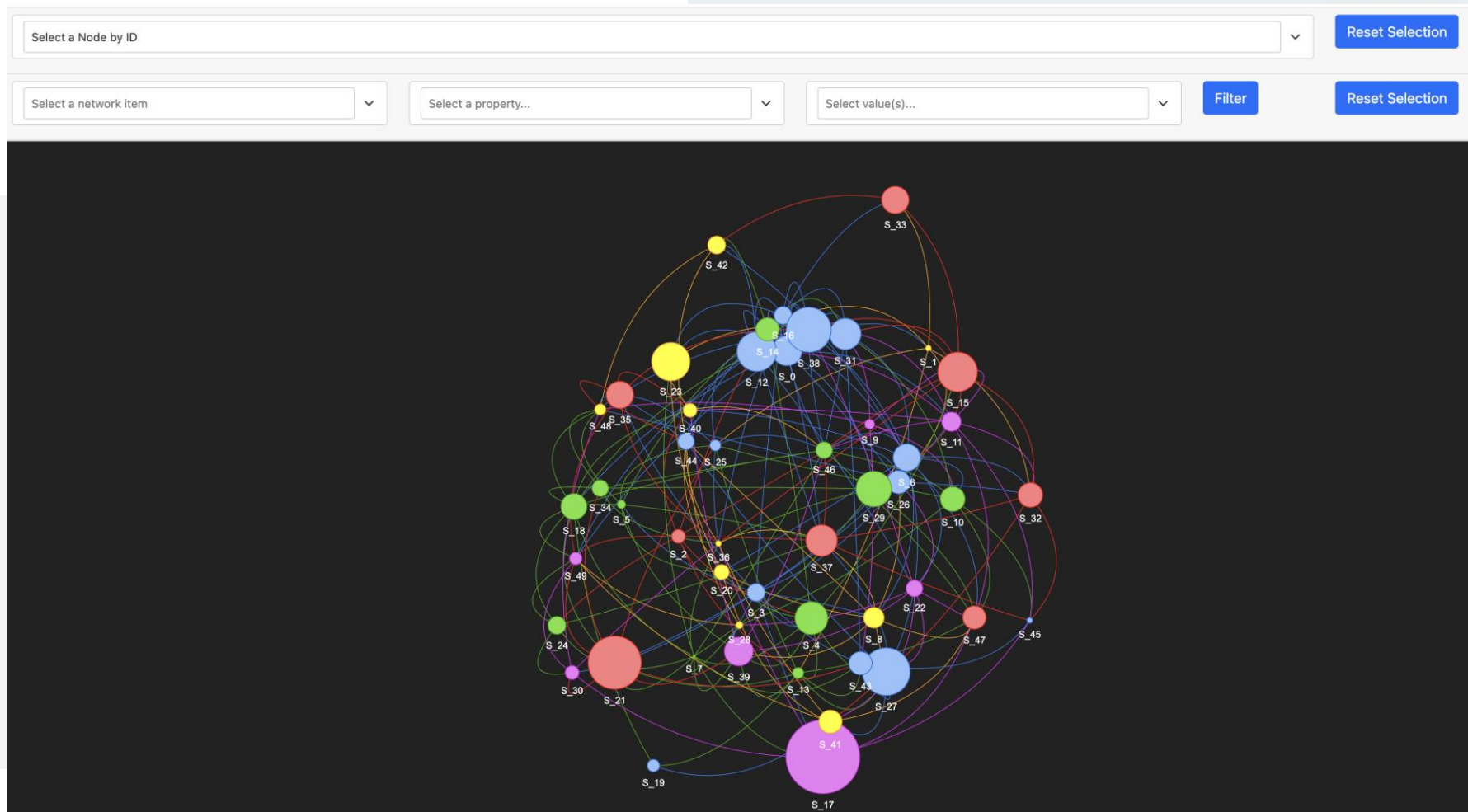
– Starting to Plot

```
# add edge
for i in range(len(G_edges)):
    g.add_edge(int(G_edges['source'][i]), int(G_edges['target'][i]),
               weight=G_edges['weight'][i])
# add options
g.set_template_dir = None
g.show_buttons()
# export
g.show('adv_ex.html', notebook=False)
```



Task 3: Visualize with pyVis

– Results



Task 4: Visualize with Jaal

- Sometimes we do not need several parameters to adjust the visualization. Here, we are going to introduce another graph visualization package – Jaal.
- This package is designed to plot your graph inside your Jupyter Notebook directly with minimal (critical) filters and functions for visualization adjustment.



Task 4: Visualize with Jaal

- In Jaal, the allowed input argument names are very restricted, especially for column names; for example, the **source** and **target** should be named **from** and **to**, respectively.
- Additionally, the term of “**size**” cannot be used for edge or node column names.

```
# rename columns for Jaal
```

```
# Jaal reinforce source & target should be named as from and to
```

```
G_edges.columns = ['from', 'to', 'weight']
```

```
G_nodes.columns = ['id', 'importance', 'group', 'code']
```



Task 4: Visualize with Jaal

- Group number or label should be a string.

```
# change the data type of the group column into String  
G_nodes['group'] = G_nodes['group'].astype(str)
```

- Define edge weight divisions via mean and standard deviations

```
# compute the mean and standard deviation for division  
mean = np.mean(G_edges['weight'])  
stddev = np.std(G_edges['weight'], ddof=1)
```



Task 4: Visualize with Jaal

– Simple Plot

simple demo

```
Jaal(G_edges, G_nodes).plot()
```

Search

Search node in graph...

Show the node you are looking for

Filter Hide/Show

Color Hide/Show Legends

Color nodes by None

Select the categorical node property to color nodes by

Color edges by None

Select the categorical edge property to color edges by

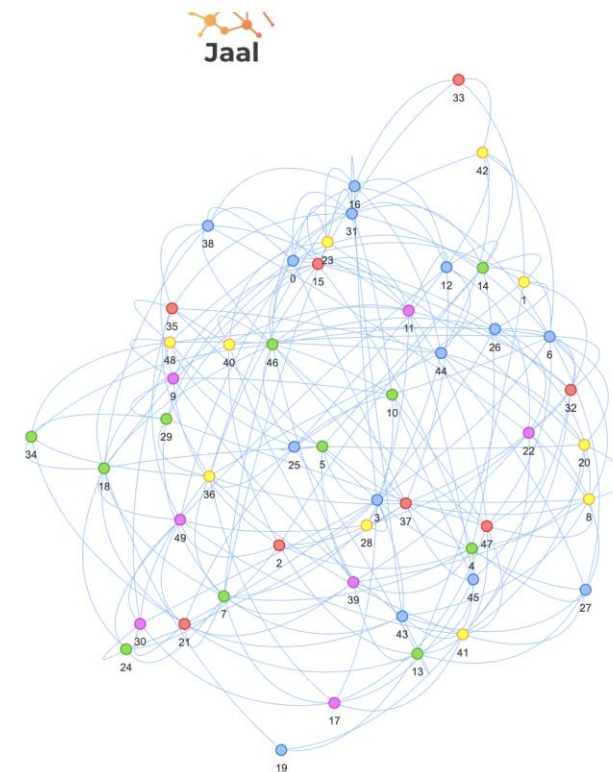
Size Hide/Show

Size nodes by None

Select the numerical node property to size nodes by

Size edges by None

Select the numerical edge property to size edges by



Task 4: Visualize with Jaal

– Adjustment

Parsing the data...Done

Search

Search node in graph...

Show the node you are looking for

Filter

Hide/Show

Color

Hide/Show

Legends

Color nodes by

group

Select the categorical node property to color nodes by

Color edges by

strength

Select the categorical edge property to color edges by

Size

Hide/Show

Size nodes by

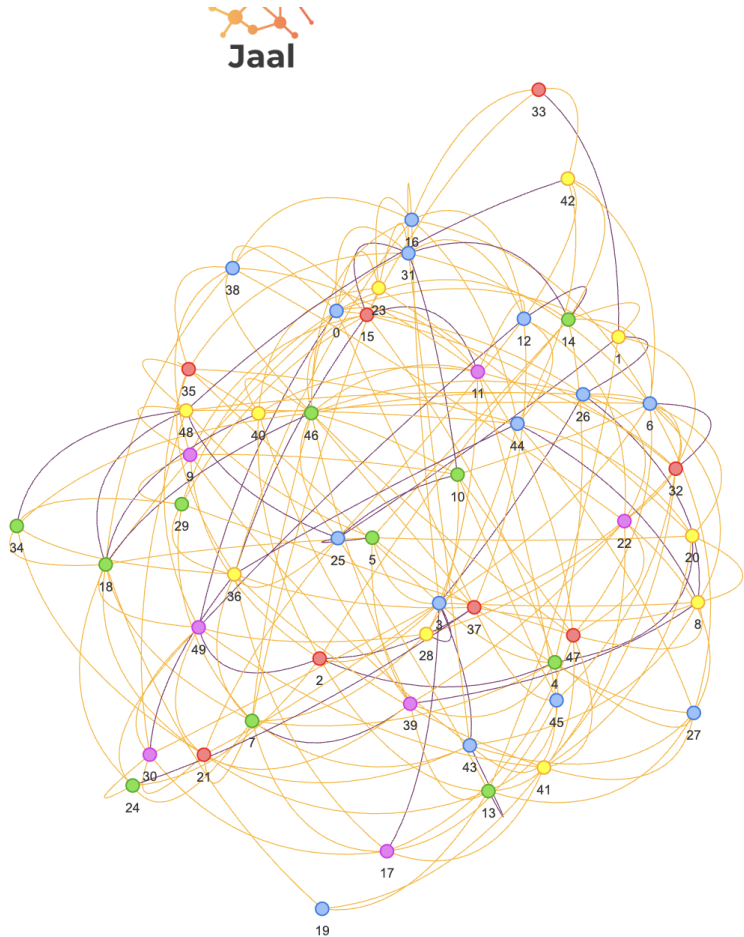
None

Select the numerical node property to size nodes by

Size edges by

None

Select the numerical edge property to size edges by



Task 4: Visualize with Jaal

– Adjustment

Parsing the data...Done

Search

Search node in graph...

Show the node you are looking for

Filter Hide/Show

Color Hide/Show Legends

Color nodes by group

Select the categorical node property to color nodes by

Color edges by strength

Select the categorical edge property to color edges by

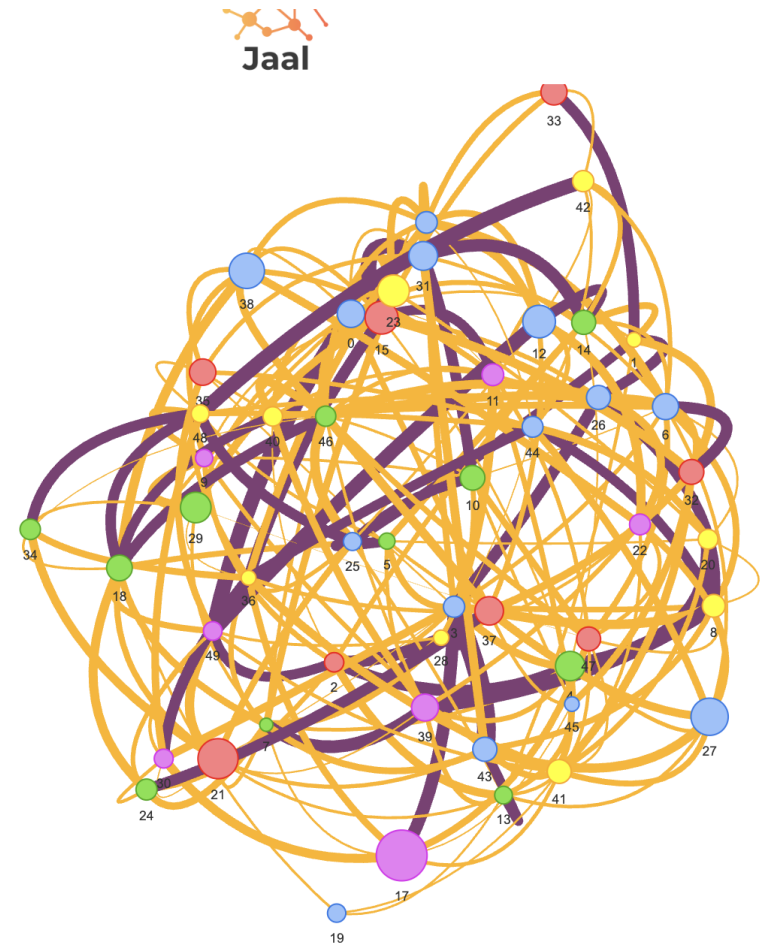
Size Hide/Show

Size nodes by importance

Select the numerical node property to size nodes by

Size edges by weight

Select the numerical edge property to size edges by



Task 4: Visualize with Jaal

– For Directed Graph

set for directed graph

```
Jaal(G_edges, G_nodes).plot(directed=True)
```

Search

Search node in graph...

Show the node you are looking for

Filter Hide/Show

Color Hide/Show Legends

Color nodes by ⌵

Select the categorical node property to color nodes by

Color edges by ⌵

Select the categorical edge property to color edges by

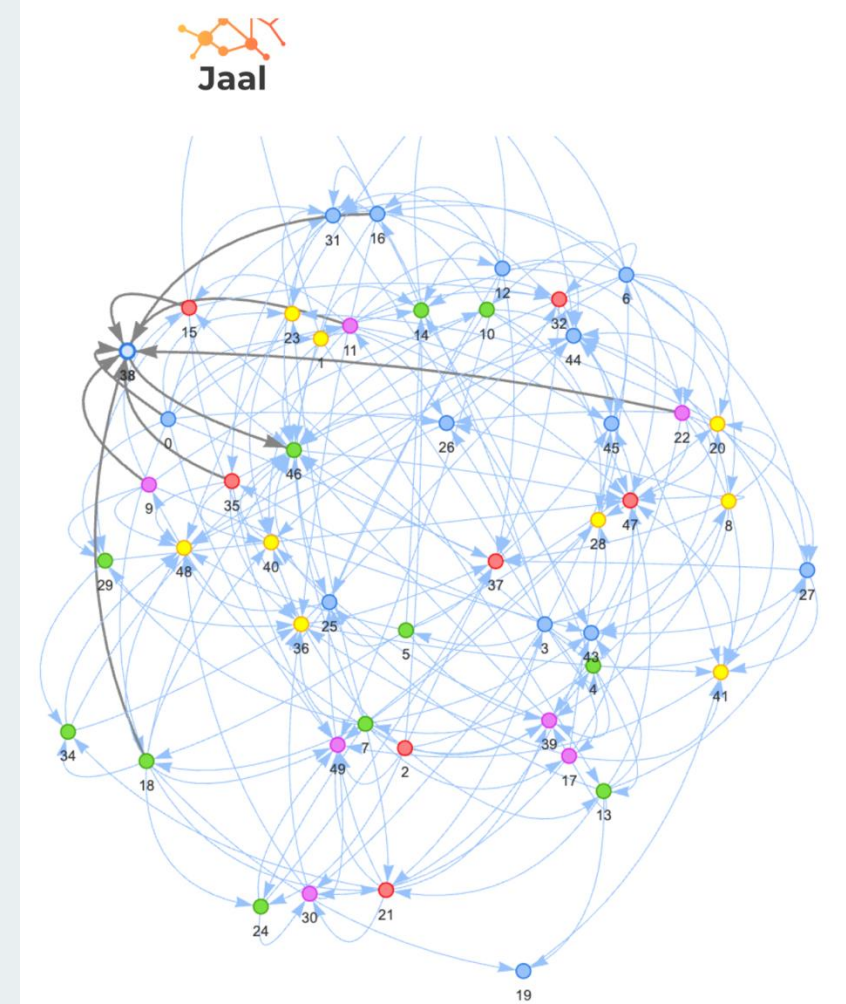
Size Hide/Show

Size nodes by ⌵

Select the numerical node property to size nodes by

Size edges by ⌵

Select the numerical edge property to



Task 4: Visualize with Jaal

– Adjustment

Parsing the data...Done

Search

Show the node you are looking for

Filter

Color

Color nodes by **group** ▾

Select the categorical node property to color nodes by

Color edges by **strength** ▾

Select the categorical edge property to color edges by

Size

Size nodes by **importanc** ▾

Select the numerical node property to size nodes by

Size edges by **weight** ▾

Select the numerical edge property to



Task 4: Visualize with Jaal

– Add More Settings

init Jaal and run server

```
Jaal(G_edges, G_nodes).plot(vis_opts={'height': '600px', # change height  
                                   'interaction':{'hover': True}, # turn on-off the hover  
                                   'physics':{'stabilization':{'iterations': 100}}},  
    ) # define the convergence iteration of network
```



Task 4: Visualize with Jaal

– Results

Parsing the data...Done

Search

Search node in graph...

Show the node you are looking for

Filter Hide/Show

Color Hide/Show Legends

Color nodes by ⌵

Select the categorical node property to color nodes by

Color edges by ⌵

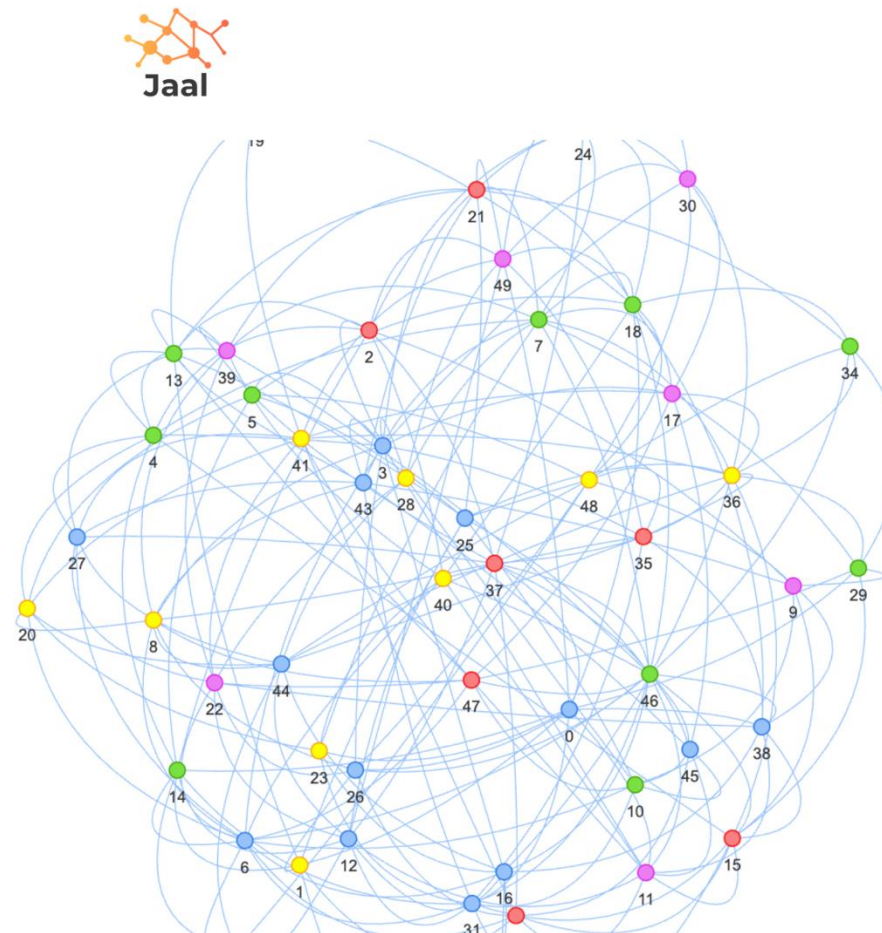
Select the categorical edge property to color edges by

Size Hide/Show

Size nodes by ⌵

Select the numerical node property to size nodes by

Size edges by ⌵



Task 4: Visualize with Jaal

– Results

Parsing the data...Done

Search

Search node in graph...

Show the node you are looking for

Filter Hide/Show

Color Hide/Show Legends

Color nodes by group ▾

Select the categorical node property to color nodes by

Color edges by strength ▾

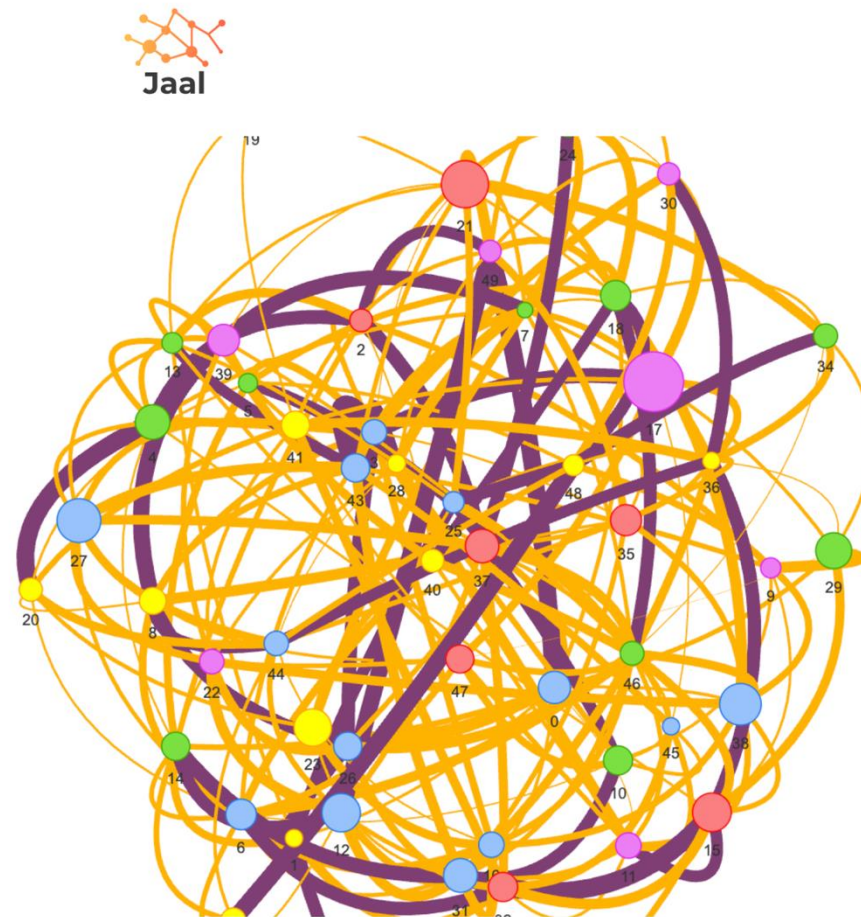
Select the categorical edge property to color edges by

Size Hide/Show

Size nodes by importanc ▾

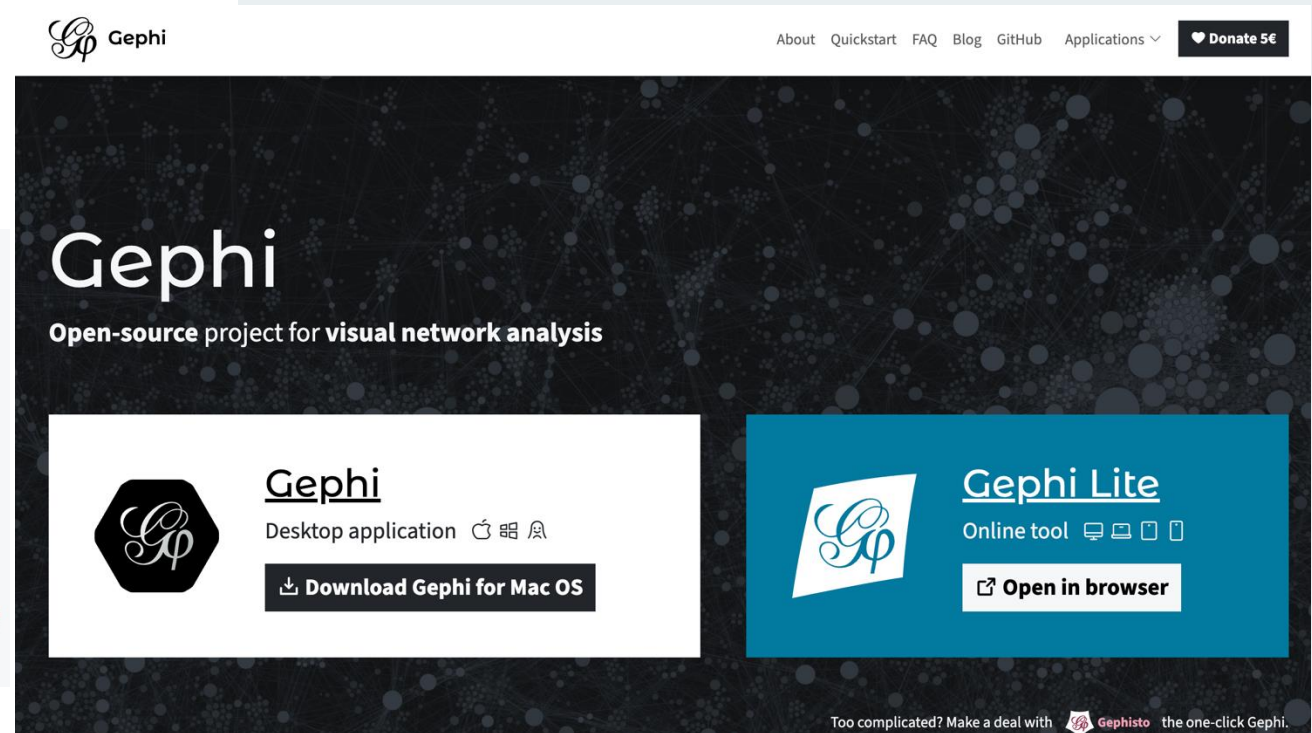
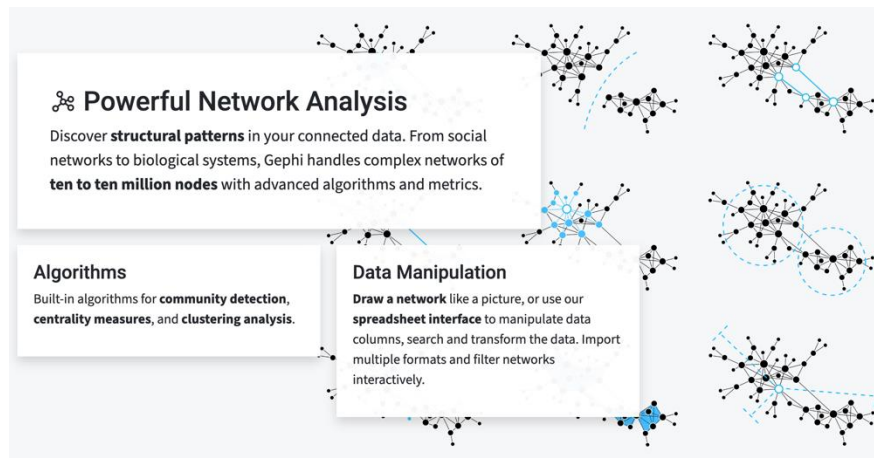
Select the numerical node property to size nodes by

Size edges by weight ▾



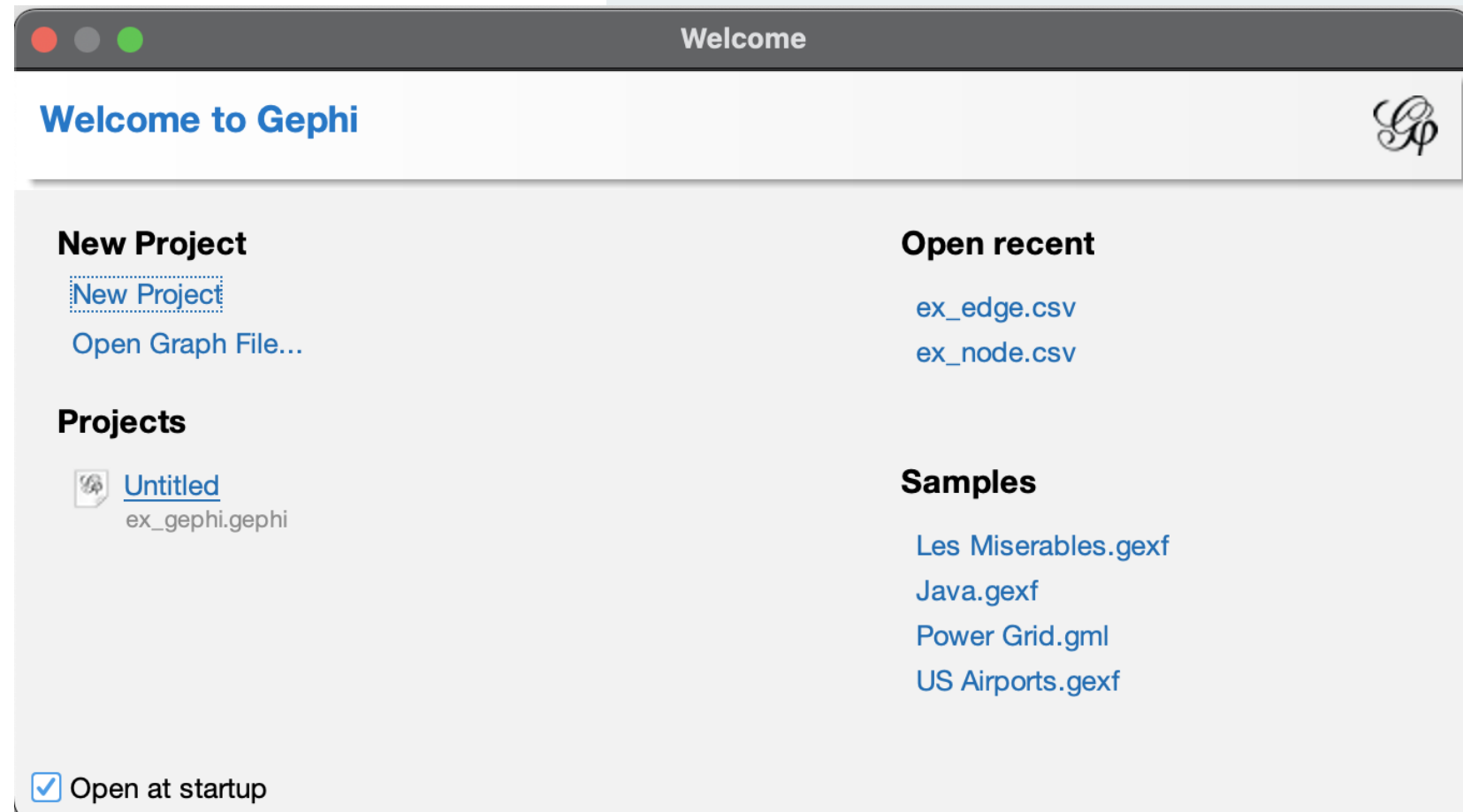
Task 5: Visualize with Gephi

- In addition to Python-based visualization, we introduce Gephi, a software tool for graph visualization.
- Installation: <https://gephi.org/>

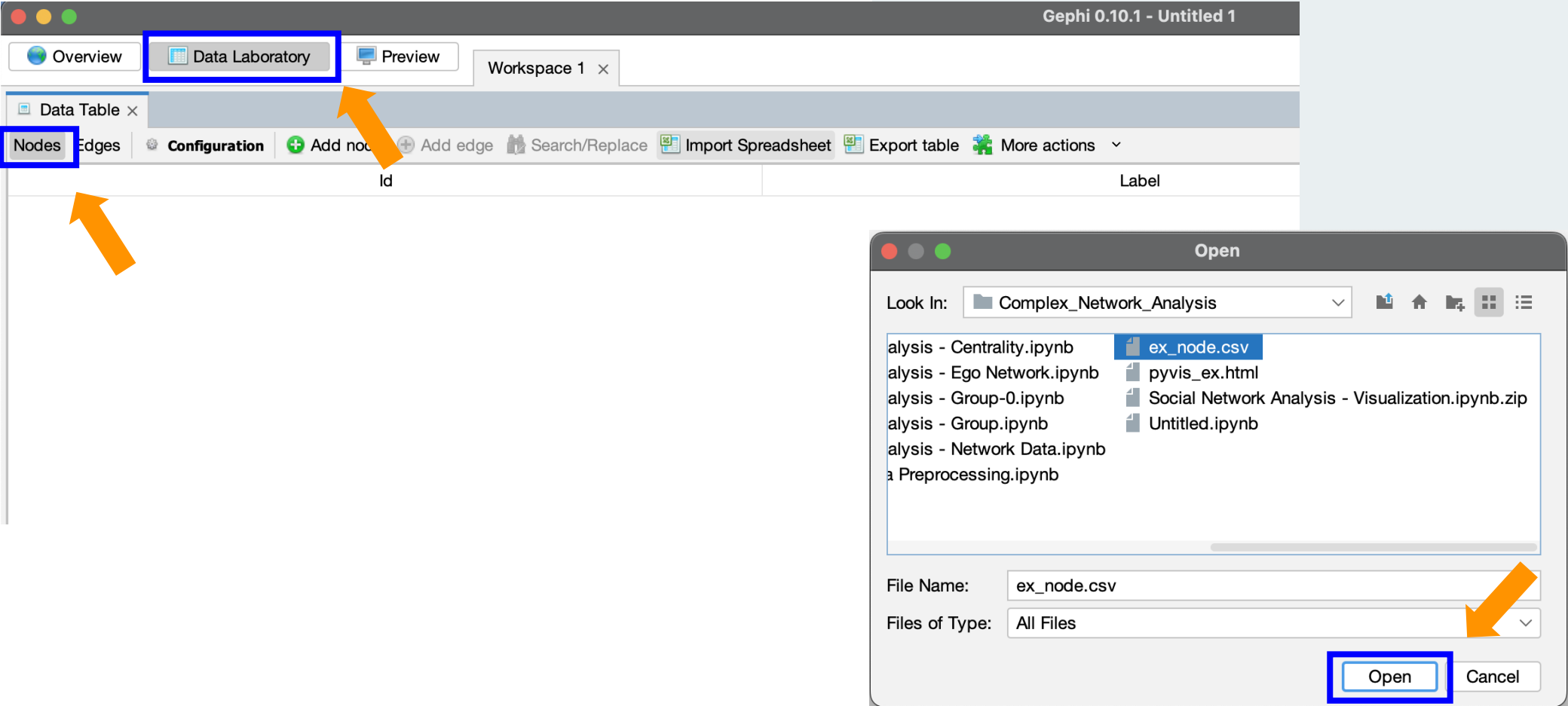


Task 5: Visualize with Gephi

– New Project



Task 5: Visualize with Gephi



Task 5: Visualize with Gephi

Spreadsheet (CSV)...

Steps

1. General CSV options

2. Import settings

General CSV options (1 of 2)

CSV file to import:

/Users/toodou/Documents/python/Complex_Network_Analysis/ex_node.csv

Separator:

Comma

Import as:

Nodes table

Charset:

UTF-8

Preview:

	ID	importance	group	code
0	22.09543...	3	S_0	
1	4.494944...	2	S_1	
2	10.82873...	1	S_2	
3	13.36812...	3	S_3	
4	24.87087...	4	S_4	
5	6.426776...	4	S_5	

< Back

Next >

Finish

Cancel

Help

Spreadsheet (CSV)...

Steps

1. General CSV options

2. Import settings

Import settings (2 of 2)

Time representation

Intervals

Imported columns:

☒ ID

☒ importance

☒ group

☒ code

Double

Integer

String

< Back

Next >

Finish

Cancel

Help



Task 5: Visualize with Gephi

Import report

Source: ex_node.csv

Issues

Report

No issue found during import

Graph Type: Mixed

More options...

of Nodes: 50

of Edges: 0

Dynamic Graph: no

Dynamic Attributes: no

Multi Graph: no

☒ New workspace

☐ Append to existing workspace

OK

Cancel



Gephi 0.10.1 - Untitled 1

Overview

Data Laboratory

Preview

Workspace 1 x ex_node x

Data Table x

Nodes	Edges	Configuration	Add node	Add edge	Search/Replace	Import Spreadsheet	Export table	More actions	Filter:	Id
Id	Label	Interval	importance	group	code					
0			22.095438	3	S_0					
1			4.494944	2	S_1					
2			10.828733	1	S_2					
3			13.368127	3	S_3					
4			24.870871	4	S_4					
5			6.426777	4	S_5					
6			20.160018	3	S_6					
7			2.254607	4	S_7					
8			15.717886	2	S_8					
9			7.855619	5	S_9					
10			18.311433	4	S_10					
11			14.204491	5	S_11					
12			29.439554	3	S_12					
13			8.521471	4	S_13					
14			17.384215	4	S_14					
15			29.793572	1	S_15					
16			13.852877	3	S_16					
17			54.816701	5	S_17					
18			19.700359	4	S_18					
19			9.376037	3	S_19					
20			11.402844	2	S_20					
21			39.925948	1	S_21					
22			12.884442	5	S_22					
23			28.291281	2	S_23					

Add column

Merge columns

Delete column

Clear column

Copy data to other column

Fill column with a value

Duplicate column

Create a boolean column from regex match

Create column with list of regex matching groups



Task 5: Visualize with Gephi

Overview

Data Laboratory

Preview

Workspace 1 x ex_node x

Data Table x

Nodes Edges Configuration

+ Add node

+ Add edge

Search/Replace

Import Spreadsheet

Export table

More actions

Filter:

Source

Target

Type

Id

Label

Interval

Open

Look In: Complex_Network_Analysis

centrality

lib

[Social Network Analysis] Centrality.ipynb

[Social Network Analysis] Ego Network.ipynb

[Social Network Analysis] Network Data.ipynb

[Social Network Analysis] Visualization.ipynb

adv_ex.html

centrality.zip

Complex Network Analysis

Complex Network Analysis

Complex Network Analysis

Complex Network Analysis

Complex Network Analysis

Demo OneWorld Data Pr

email-Eu-core.txt

ex_edge.csv

File Name:

Files of Type:

All Files

Open

Cancel

Spreadsheet (CSV)...

Steps

1. General CSV options

2. Import settings

General CSV options (1 of 2)

CSV file to import:

/Users/toodou/Documents/python/Complex_Network_Analysis/ex_edge.csv

Separator: Comma

Import as: Edges table

Charset: UTF-8

Preview:

source	target	weight
0	1	5.494500...
0	11	3.121376...
0	14	8.873844...
0	18	6.916639...
0	23	14.92848...
0	26	8.437991...

< Back

Next >

Finish

Cancel

Help

Add column

Merge columns

Delete column

Clear column

Copy data to other column

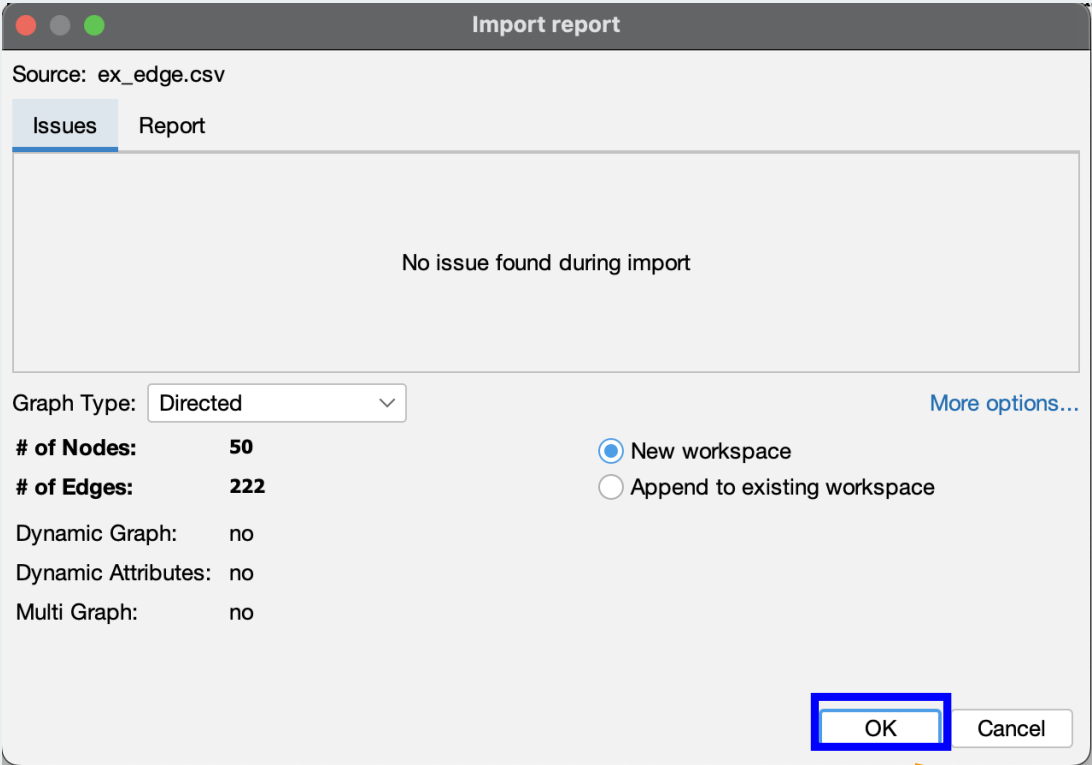
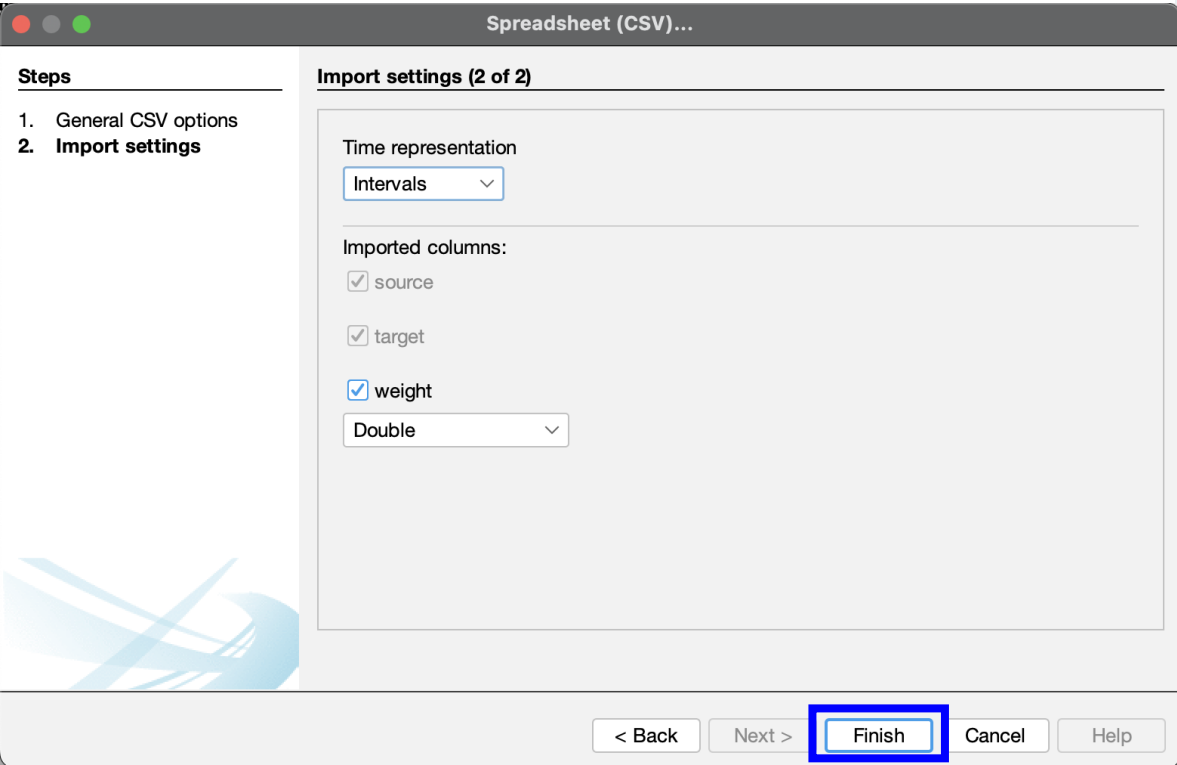
Fill column with a value

Duplicate column

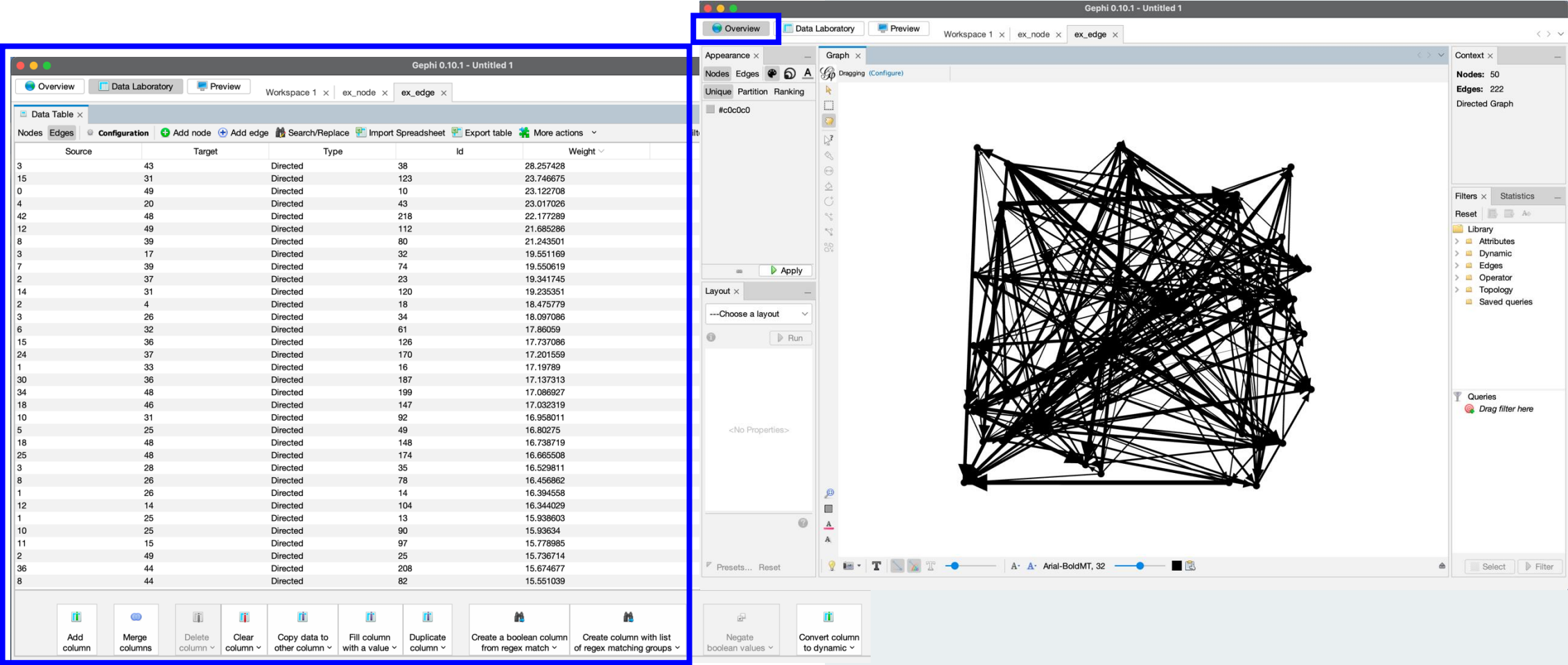
Create a boolean column from regex match

Create column with list of regex matching groups

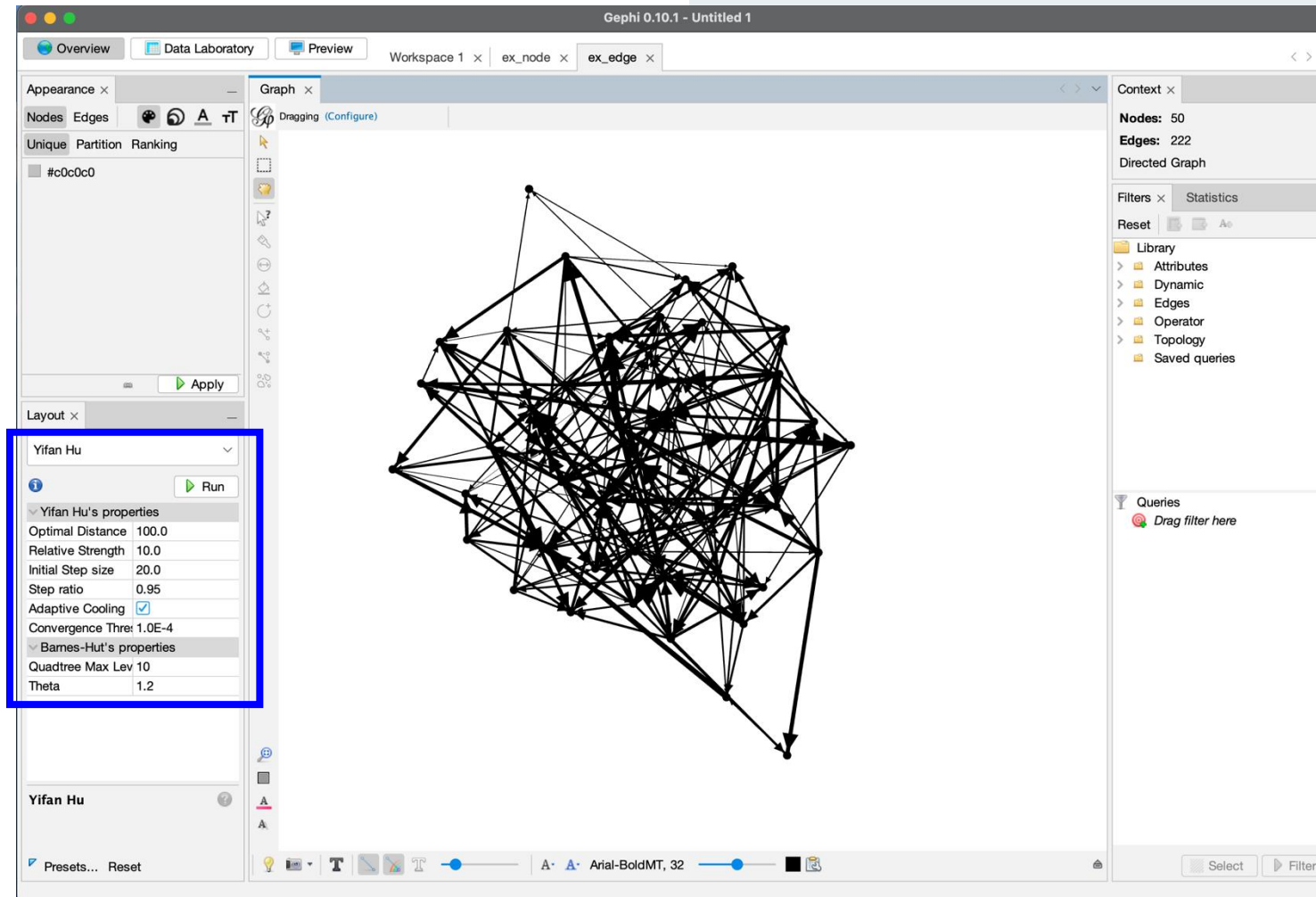
Task 5: Visualize with Gephi



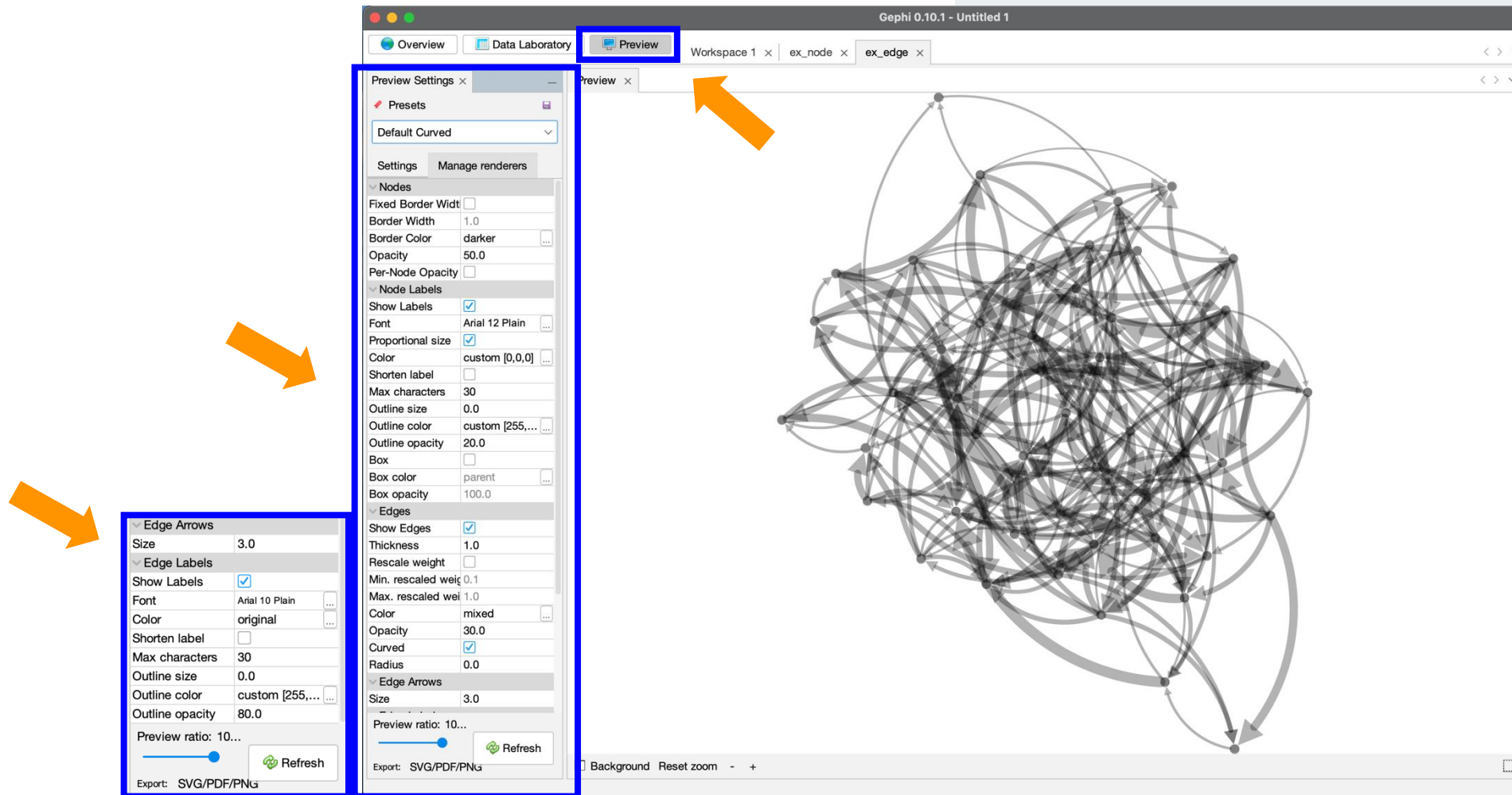
Task 5: Visualize with Gephi



Task 5: Visualize with Gephi



Task 5: Visualize with Gephi



Task 5: Visualize with Gephi

Search

Gephi version

☐ 0.10.0

☐ 0.9.6

☐ 0.9.3

☐ 0.9.2

☐ 0.9.1

☐ 0.9.0

☐ 0.8.2

Plugin category

☐ Layout

☐ Metric

☐ Import

☐ Tool

☐ Export

☐ Clustering

☐ Generator

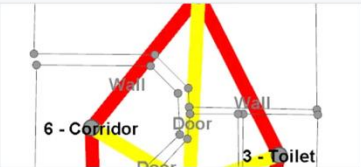
☐ Preview

☐ Filter

☐ Other Category

☐ Data Laboratory

0.8.2



Generator

Architectural GraphML

last updated on November 13, 2014

0.8.2



Layout

Alphabetical Sorter

last updated on October 19, 2014

0.8.2

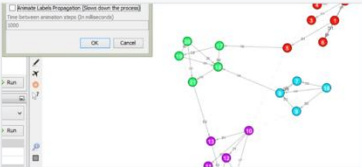


Preview

Image Preview

last updated on July 19, 2014

0.8.2

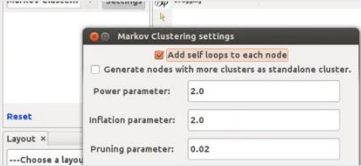


Clustering

Label Propagation Clustering

last updated on June 23, 2014

0.8.2



Clustering

Markov Cluster Algorithm (MCL)

last updated on May 30, 2014

0.8.2

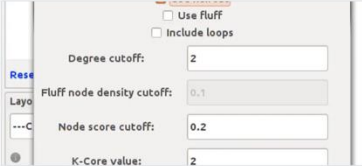


Clustering

Girvan Newman Clustering

last updated on May 30, 2014

0.8.2



Clustering

Molecular Complex Detection (...)

last updated on May 29, 2014

0.8.2

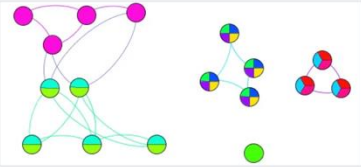


Layout

Random 3D Layout

last updated on May 14, 2014

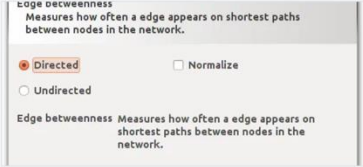
0.8.2



Preview

Multi Colour Renderer

0.8.2



Metric

Edge Betweenness Metric

0.8.2



Layout

GC-Viz

0.8.2



Metric

Social Network Analysis

The End

Thank you for your attention!



Email: chchan@ntnu.edu.tw
Website: toodou.github.io