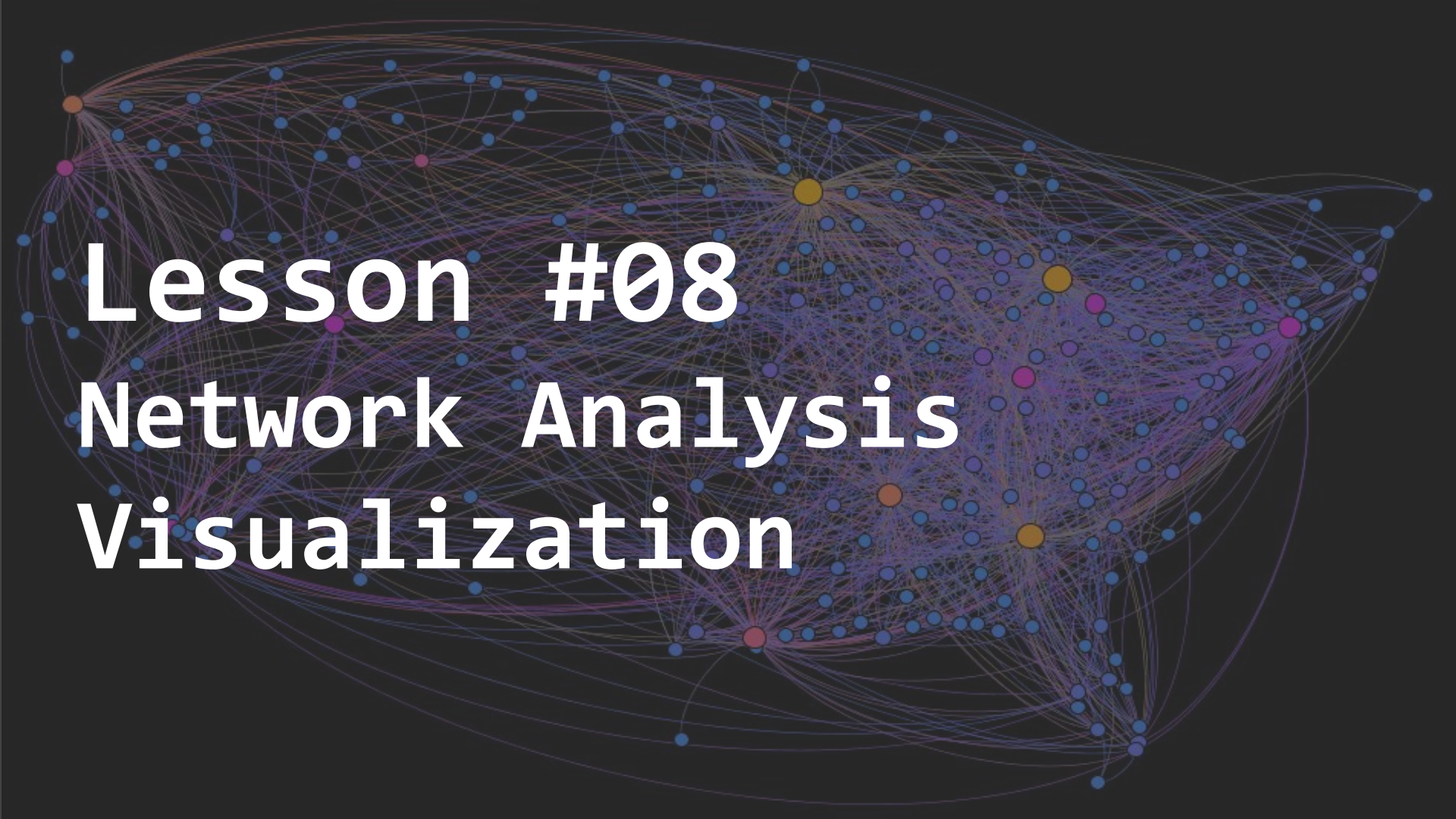
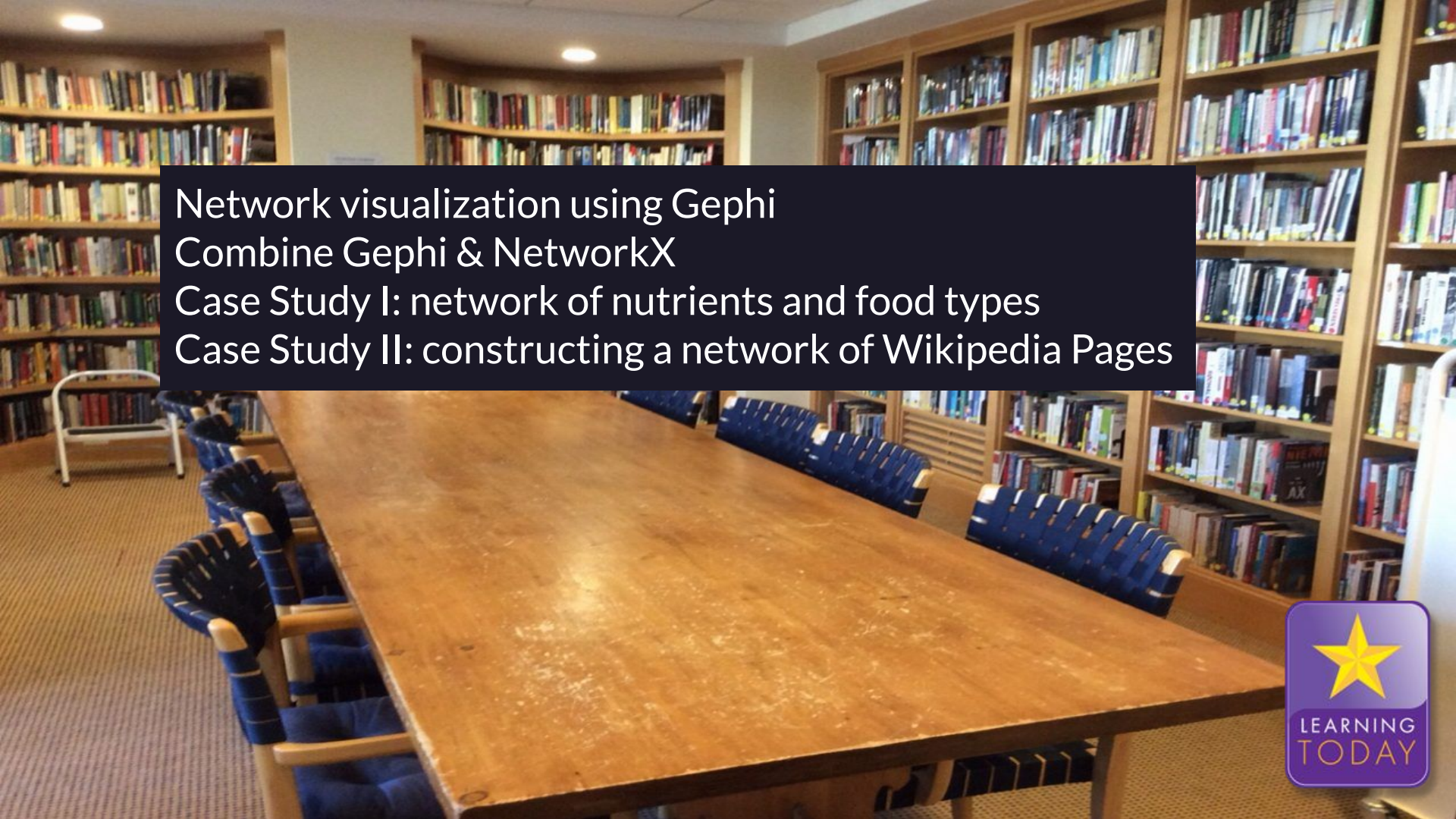
Lesson #08
Network Analysis Visualization



Network visualization using Gephi
Combine Gephi & NetworkX
Case Study I: network of nutrients and food types
Case Study II: constructing a network of Wikipedia Pages



The Open Graph Viz Platform

Gephi is the leading visualization and exploration software for all kinds of graphs and networks. Gephi is open-source and free.

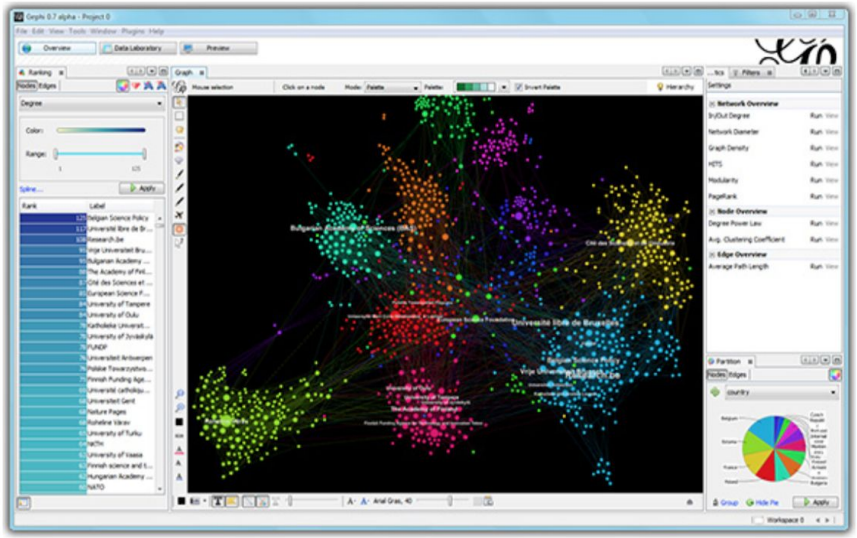
Runs on Windows, Mac OS X and Linux.

Learn More on Gephi Platform »

Download FREE
Gephi 0.9.2

Release Notes | System Requirements

- Features
- Quick start
- Screenshots
- Videos



Support us! We are non-profit. Help us to innovate and empower the community by donating only 8€:

Donate

APPLICATIONS

- **Exploratory Data Analysis**: intuition-oriented analysis by networks manipulations in real time.
- **Link Analysis**: revealing the underlying structures of associations between objects.
- **Social Network Analysis**: easy creation of social

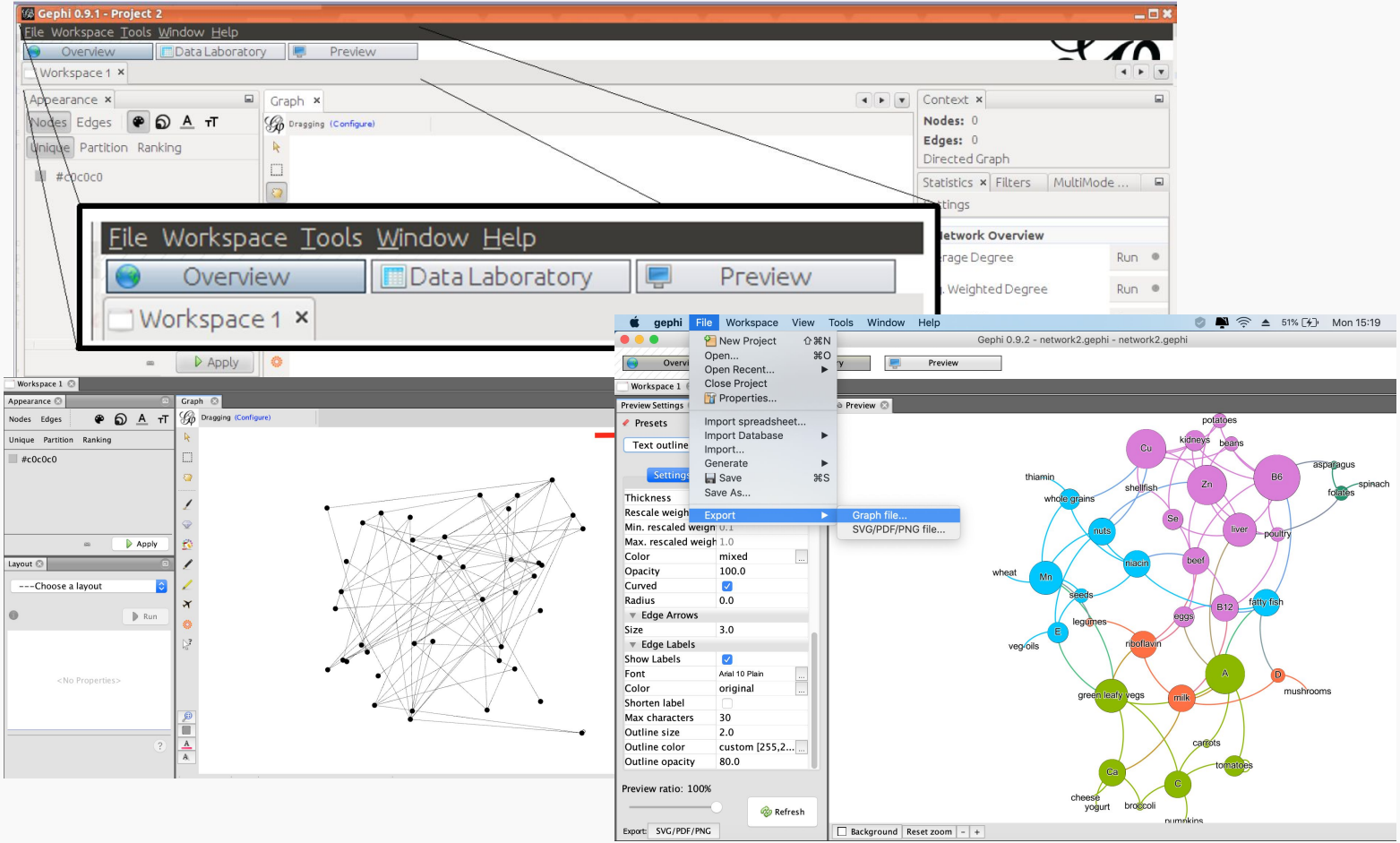
Like Photoshop™ for graphs.

— the Community

LATEST NEWS

- Gephi updates with 0.9.2 version

PAPERS

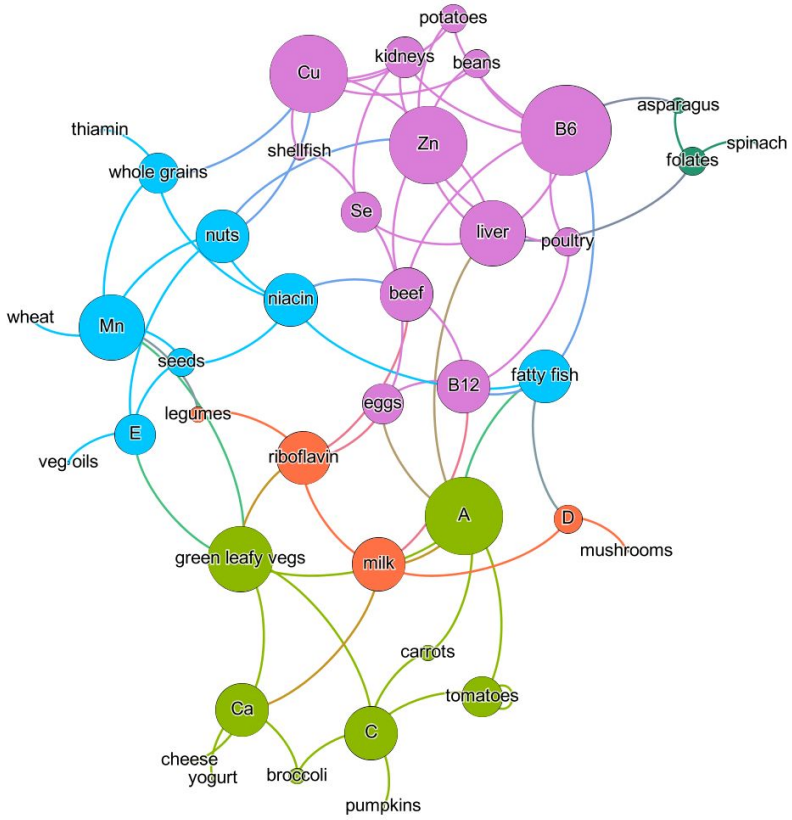


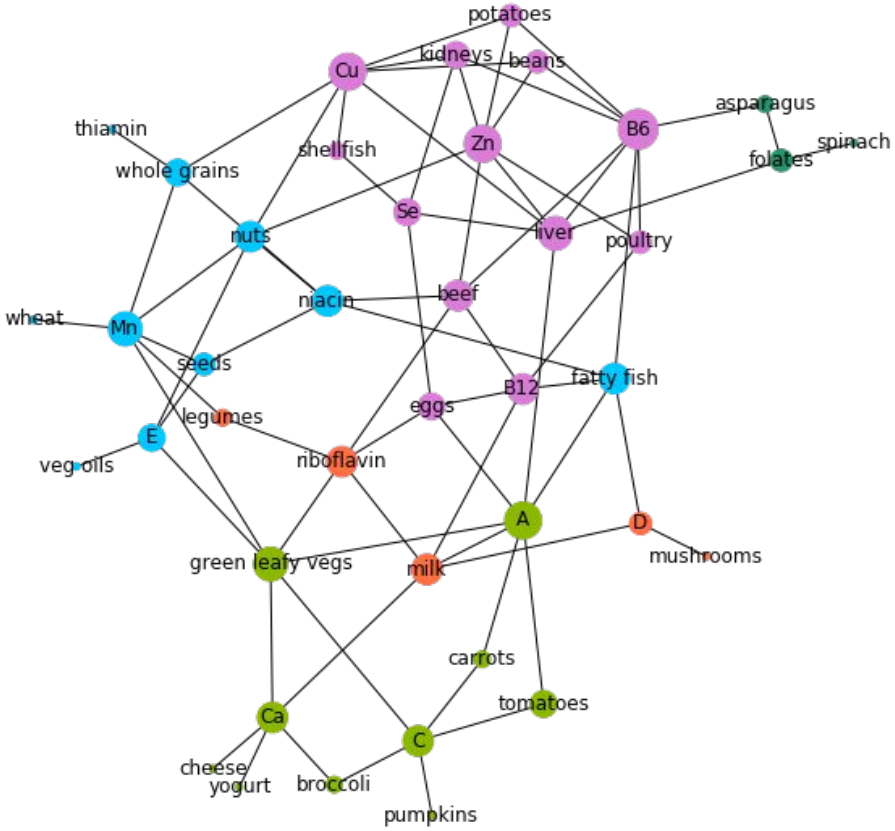
Tutorial Hands on Gephi

1. Install gephi
2. Import nutrients.csv
3. Modify a simple network
4. Explore the network
5. Sketch the network
6. Prepare a presentation-quality image
7. Export the network


KEEP
CALM
AND
LETS DO IT
TOGETHER

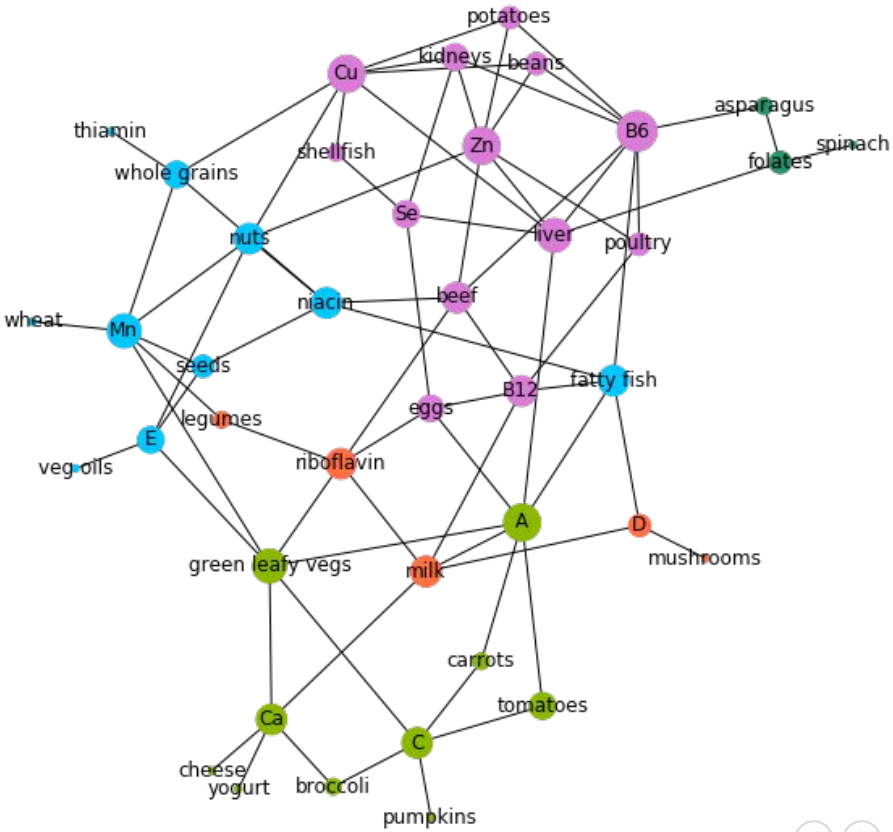
Combine Gephi & NetworkX



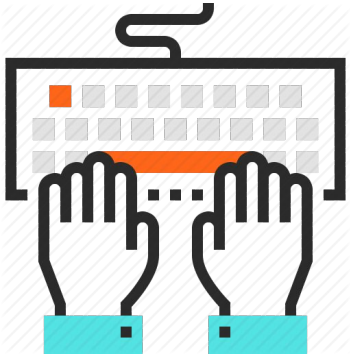


```python
import networkx as nx
# import the network generate by gephi
g = nx.read_graphml("nutrients.graphml")

# g maintains the attributes created by gephi
g.nodes["eggs"]
{'Modularity Class': 0,
 'b': 216,
 'g': 125,
 'label': 'eggs',
 'r': 217,
 'size': 26.857143,
 'x': -14.69237,
 'y': -88.377914
}
```

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(10,10))
nx.draw_networkx(g, pos=pos,
                 labels=labels,
                 node_size=node_size,
                 ax=ax,
                 node_color=node_color)

plt.axis("off")
plt.show()
```




Let's
CODE It.



Case Study: Constructing a Network of Wikipedia Pages

WIKIPEDIA

The Free Encyclopedia

English
6 195 000+ articles

Português
1 047 000+ artigos

Español
1 641 000+ artículos

日本語
1 239 000+ 記事

Deutsch
2 503 000+ Artikel

Русский
1 678 000+ статей

Français
2 270 000+ articles

Italiano
1 652 000+ voci

中文
1 159 000+ 條目

Polski
1 439 000+ haseł

Search and add articl

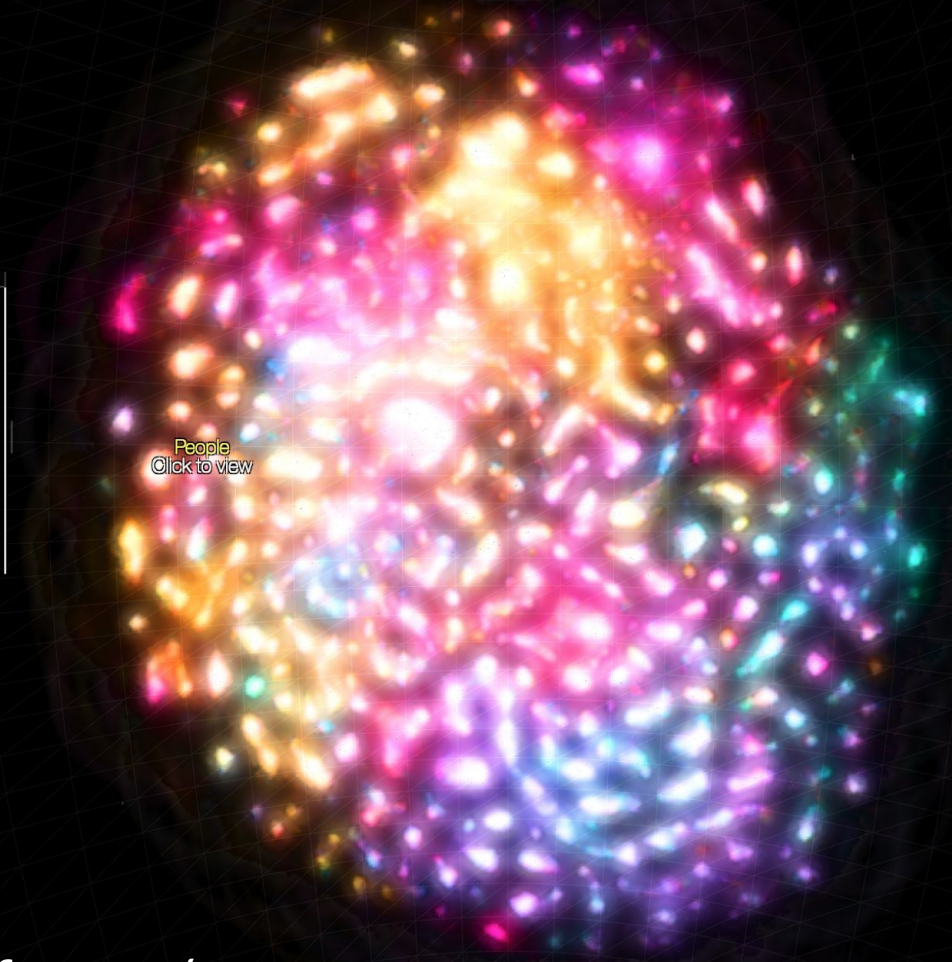
fly map home help about UI off full

Article links

Pick an article

Welcome.

WikiGalaxy is a 3D web experiment that visualizes Wikipedia as a galactic web of information. With it I aim to show the world the beauty and variety of knowledge that is available at our fingertips.
I used 100,000 of 2014's most popular articles, all clustered with hyperlinks. In this world Wikipedia articles are stars, interests are nebulas and you are on a journey through knowledge.

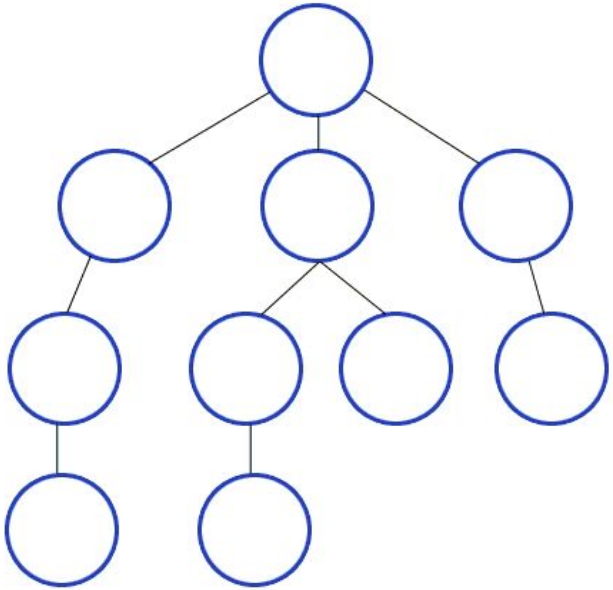


Use the mouse to see a preview of articles in each cluster
Click anywhere on the map to fly there

http://wiki.polyfra.me/

Get the data, build the network

Snowballing process (breadth-first search - BFS)

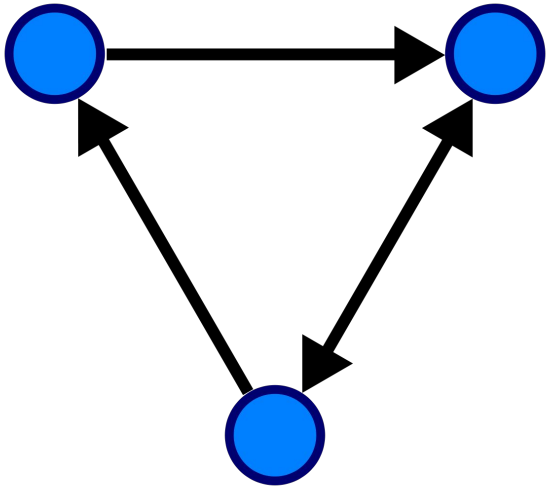


```
SEED = "Complex network".title()
STOPS = ("International Standard Serial Number",
         "International Standard Book Number",
         "National Diet Library",
         "International Standard Name Identifier",
         "International Standard Book Number (Identifier)",
         "Pubmed Identifier",
         "Pubmed Central",
         "Digital Object Identifier",
         "Arxiv",
         "Proc Natl Acad Sci Usa",
         "Bibcode",
         "Library Of Congress Control Number",
         "Jstor")
```

Wikipedia size and users (update)	
English articles:	6,196,713
Total wiki pages:	51,996,674
Average revisions:	18.96
Total admins:	1,123
Total users:	40,387,815
UTC time: 00:54 on 2020-Nov-26	

Layer 0: 1
Layer 1: N
Layer 2: N + N*N
....

```
todo_lst = [(0, SEED)] # The SEED is in the layer 0
todo_set = set(SEED) # The SEED itself
done_set = set() # Nothing is done yet
```



```
g = nx.DiGraph()
layer, page = todo_lst[0]
```

We choose a directed graph because the edges that represent HTML links are naturally directed: a link from page A to page B does not imply a reciprocal link.

2min

```
while layer < 2:
  # Remove the name page of the current page from the todo_lst,
  # and add it to the set of processed pages.
  # If the script encounters this page again, it will skip over it.
  del todo_lst[0]
  done_set.add(page)

  # Show progress
  print(layer, page)

  # Attempt to download the selected page.
  try:
    wiki = wikipedia.page(page)
  except:
    layer, page = todo_lst[0]
    print("Could not load", page)
    continue

  for link in wiki.links:
    link = link.title()
    if link not in STOPS and not link.startswith("List Of"):
      if link not in todo_set and link not in done_set:
        todo_lst.append((layer + 1, link))
        todo_set.add(link)
      g.add_edge(page, link)
  layer, page = todo_lst[0]
```

13244 nodes
25566 edges

Eliminate Duplicates

```
# remove self loops
g.remove_edges_from(g.selfloop_edges())

# identify duplicates like that: 'network' and 'networks'
duplicates = [(node, node + "s")
              for node in g if node + "s" in g
             ]

for dup in duplicates:
  # *dup is a technique named 'unpacking'
  g = nx.contracted_nodes(g, *dup, self_loops=False)

duplicates = [(x, y) for x, y in
              [(node, node.replace("-", " ")) for node in g]
                if x != y and y in g]

for dup in duplicates:
  g = nx.contracted_nodes(g, *dup, self_loops=False)

# nx.contracted creates a new node attribute called contraction
# the value of the attribute is a dictionary, but GraphML
# does not support dictionary attributes
nx.set_node_attributes(g, 0,"contraction")
```

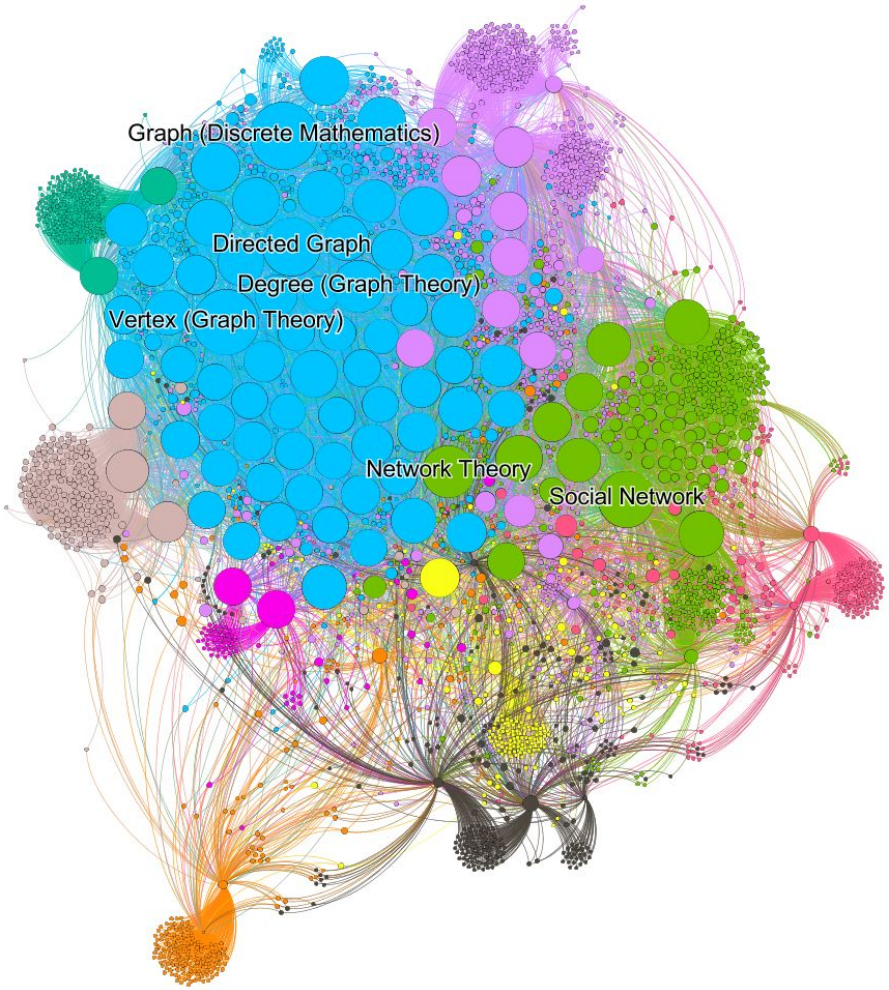
Before:
13244 nodes
25566 edges

After:
13128 nodes
24191 edges

```python
# filter nodes with degree greater than or equal to 2
core = [node for node, deg in dict(g.degree()).items() if deg >= 2]

# select a subgraph with 'core' nodes
gsub = nx.subgraph(g, core)

print("{} nodes, {} edges".format(len(gsub), nx.number_of_edges(gsub)))

nx.write_graphml(gsub, "cna.graphml")
```

Truncate the Network (4.43 edges per node)

Before:	After:
13563 nodes	3222 nodes,
26032 edges	14285 edges



Explore the Network in Gephi

```
top_indegree = sorted(dict(gsub.in_degree()).items(),
                      reverse=True, key=itemgetter(1))[:100]
print("\n".join(map(lambda t: "{} {}".format(*reversed(t)), top_indegree)))
```

```
70 Graph (Discrete Mathematics)
66 Vertex (Graph Theory)
59 Directed Graph
57 Pmc (Identifier)
56 Issn (Identifier)
56 Social Network
55 Network Theory
54 Degree (Graph Theory)
52 Adjacency Matrix
52 Network Science
51 Complex Network
51 Bipartite Graph
51 Complete Graph
```

Now is your turn!!!