CSCI2202
Lecture 12: Networks and Final

maguire-lab.github.io/scientific_computing/

Finlay Maguire (finlay.maguire@dal.ca)

TAs: Ehsan Baratnezhad (ethan.b@dal.ca); Precious Osadebamwen (precious.osadebamwen@dal.ca)

Overview

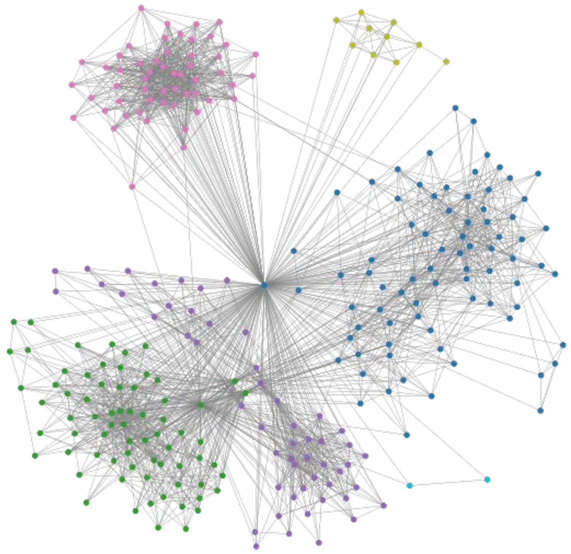
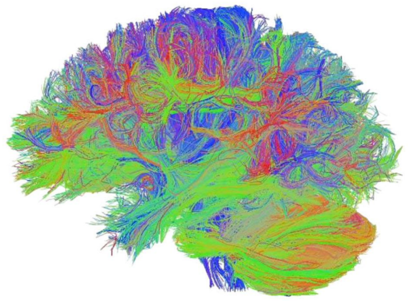
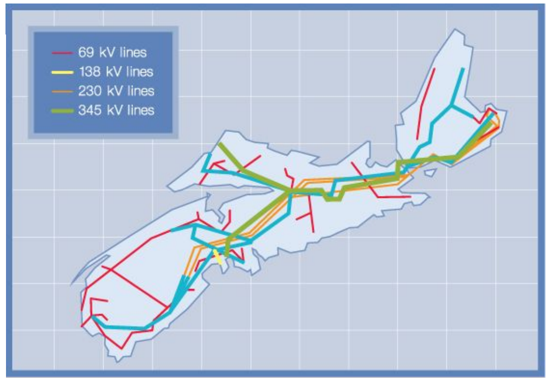
- Graph-structured Data
- Networks/Graphs Basics
- NetworkX
- Nodes/Edges in NetworkX
- Network I/O
- Network Visualization
- Centrality Analyses
- Final Exam Question Examples
  - Mid-Term Style Questions
  - Explain This Concept questions
  - Fix This Bug Questions

Network Data

Lots of datasets are best represented as a network of connected entities:

- Sociology: social networks, web page, research papers…
- Infrastructure: power grids, internet connectivity, human/vehicle flows…
- Life Sciences: neural connectome, landscapes, genome assembly…

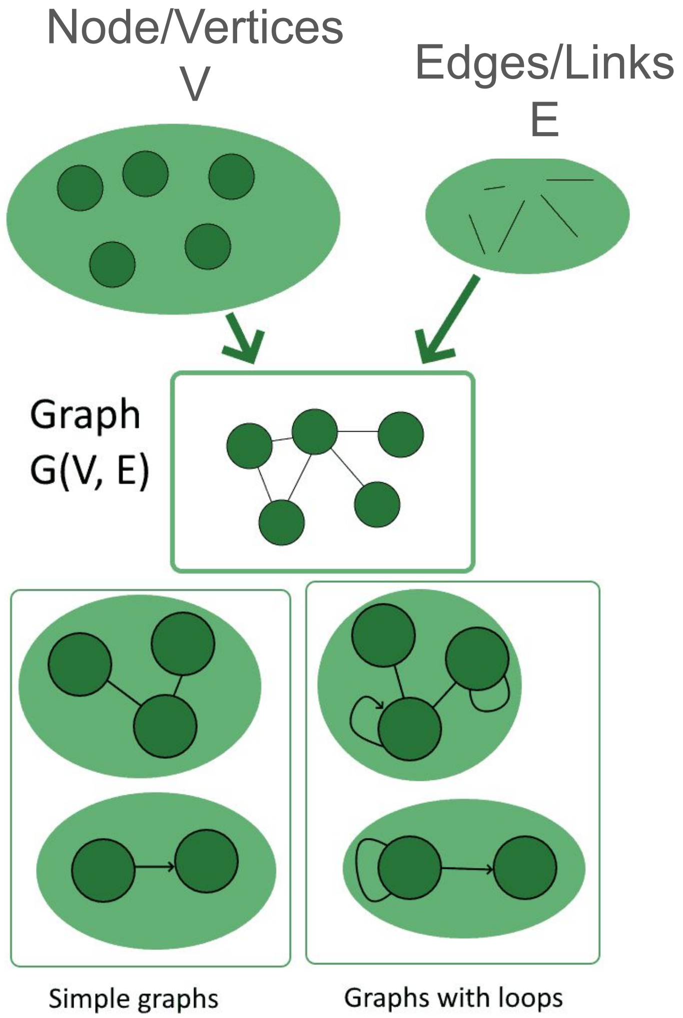
How can we analyze these types of network data?

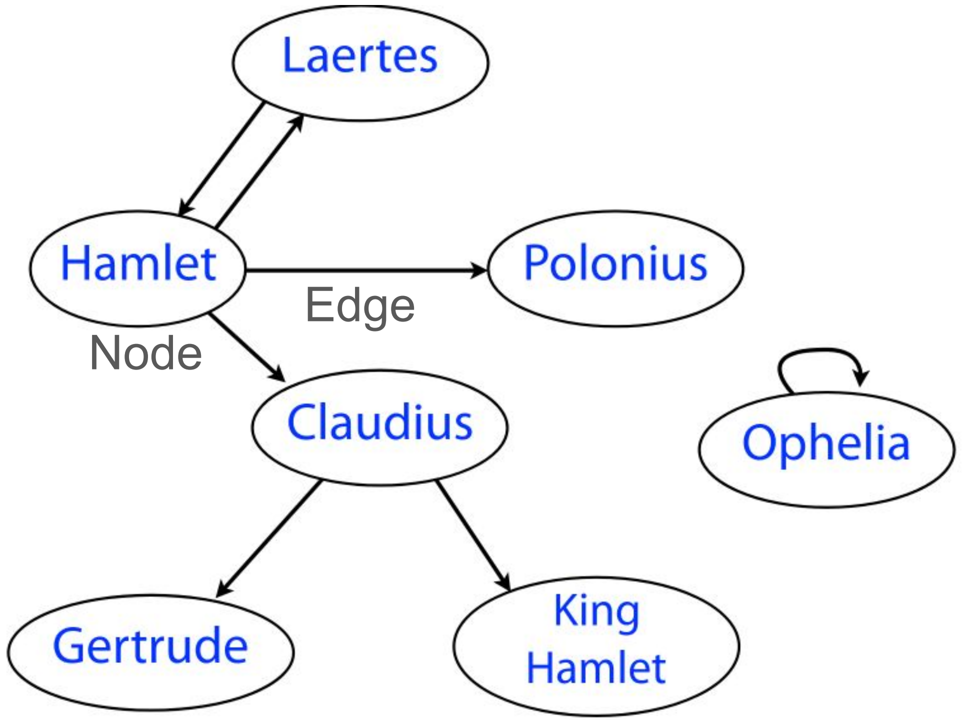
Anatomy of a network/graph

- Networks and graphs are **synonyms**
- Edges/Links can be directed or undirected (and can sometimes form loops)
- Edges/Links can have attributes (e.g., weights corresponding to physical distance or number of links)
- Nodes/Vertices can have attributes (e.g., class of neuron, parent domain etc)
- Graphs don't have to be fully connected
- Unconnected parts are called "components"
- Subset of a graph is called a subgraph

https://www.baeldung.com/cs/graph-theory-intro



Hamlet-onian Directed Graph



https://www.cs.jhu.edu/~langmea/resources/lecture_notes/16_assembly_scs_v2.pdf

Mathematical Graph Representations

Adjacency List:

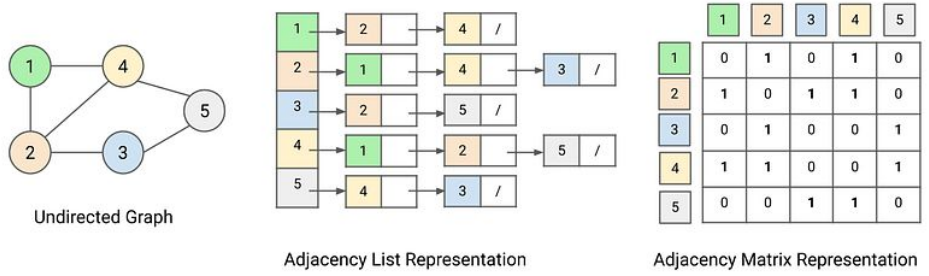
Node -> connected Nodes

Adjacency Matrix:

Node X Node connections

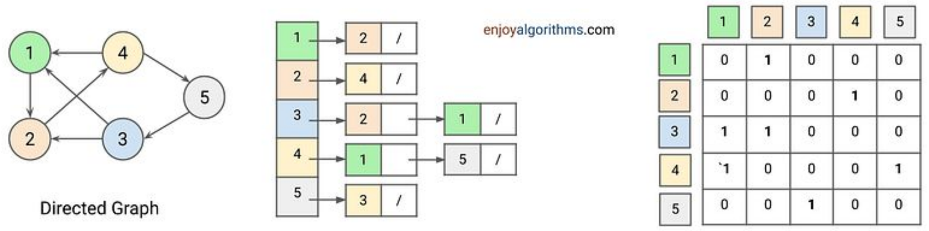
Incidence List:

Node x Edges connections


Undirected Graph
Adjacency List Representation
Adjacency Matrix Representation

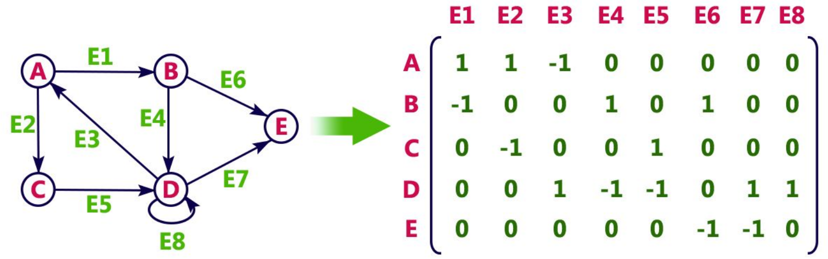
1 → 2 → 4 → /
2 → 1 → 4 → 3 → /
3 → 2 → 5 → /
4 → 1 → 2 → 5 → /
5 → 4 → 3 → /

	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	1	0
3	0	1	0	0	1
4	1	1	0	0	1
5	0	0	1	1	0



1 → 2 → /
2 → 4 → /
3 → 2 → 1 → /
4 → 1 → 5 → /
5 → 3 → /

	1	2	3	4	5
1	0	1	0	0	0
2	0	0	0	1	0
3	1	1	0	0	0
4	`1	0	0	0	1
5	0	0	1	0	0

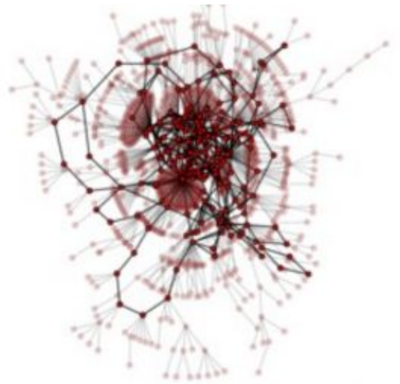


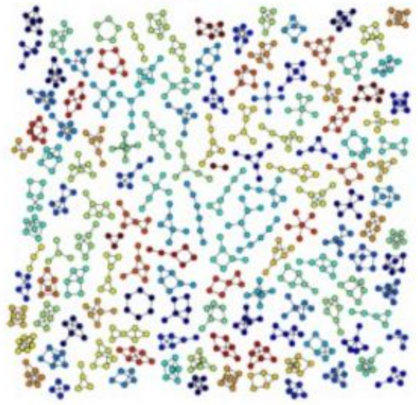
	E1	E2	E3	E4	E5	E6	E7	E8
A	1	1	-1	0	0	0	0	0
B	-1	0	0	1	0	1	0	0
C	0	-1	0	0	1	0	0	0
D	0	0	1	-1	-1	0	1	1
E	0	0	0	0	0	-1	-1	0

https://www.enjoyalgorithms.com/blog/graph-representation-in-data-structures
https://findtodaysnotes.wordpress.com/graph-representations/

NetworkX
Network Analysis in Python

- Data structures for representing many types of networks, or graphs
- Nodes can be any (hashable)[1] object, edges can contain arbitrary data
- Flexibility ideal for representing networks found in many different fields
- Online up-to-date documentation
- First public release in April 2005
- Supports I/O from many common graph formats





1 Objects that implement __hash__ and don't change are hashable i.e., immutable objects. This is the same requirement as objects which can be used as dictionary keys

https://www.cl.cam.ac.uk/~cm542/teaching/2010/stna-pdfs/stna-lecture8.pdf

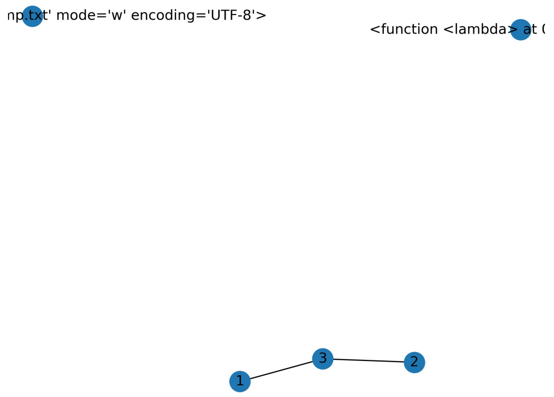
Constructing a simple graph in networkx

NetworkX core objects:

`Graph`, `DiGraph`, `MultiGraph`

Core objects support many attributes and methods e.g., can build a graph by adding nodes and edges

A node can be any hashable object such as strings, numbers, filehandles, functions, and more.



```
import networkx as nx

g = nx.Graph() # or nx.DiGraph() for directed

g.add_node(1) # method of nx.Graph

g.add_nodes_from([2 ,3]) # Add a list of nodes

g.remove_node(2) # remove nodes

g.add_node(lambda x: x+2) # function

g.add_node(open('tmp.txt','w')) # file handle

g.add_edge(1,2)

g.add_edge(*(2,3)) # unpack edge tuple

g.add_edges_from([(1 ,2) ,(1 ,3)]) # List of
edges

g.remove_edge(1,2)
```

Node and Edge attributes and access

Graph is accessed like a dictionary with nodes as primary keys

Nodes can have many different attributes associated with them.

Edges can also have attributes, with one particular special attribute called “weight”.

“weight” should always be numeric and holds values used by algorithms requiring weighted edges (e.g., finding the shortest path through a graph)

```
g.add_node(1, time=’5pm’)

g.node[1][’time’]

’5pm’

g.node[1]

{’time’: ’5pm’}

g.add_edge(1, 2, weight=4.0 )

g[1][2][‘weight’] = 5.0 # edge already added

g[1][2]

{‘weight’: 5.0}
```

Node and Edge Iterators

Many applications require iteration over nodes or over edges.

Graph objects in networkx can be used as customisable iterators

```
for node in g.nodes():

      print(node, g.degree(node))

1, 1

2, 1

g.add_edge(1,3,weight=2.5)

g[1][2]['weight'] = 1.5


for n1, n2, attr in g.edges(data=True):

      print(n1, n2, attr['weight'])

1, 2, 1.5

1, 3, 2.5
```

Graph I/O

General read/write format

```
g = nx.read_format(“path/to/file.txt”,...options...)

nx.write_format(g,“path/to/file.txt”,...options...)
```

Read and write edge lists

```
g = nx.read_edgelist(path, comments='#', create_using=None,

delimiter=' ', nodetype=None, data=True, edgetype=None, encoding='utf-8')

nx.write_edgelist(g, path, comments='#', delimiter=' ', data=True, encoding='utf-8')
```

Many graph operators supported

`subgraph(G, list_of_nodes)` - induce subgraph of G on nodes `list_of_nodes`

`union(G1, G2)` - graph union

`compose(G1, G2)` - combine graphs identifying nodes common to both

`complement(G)` - graph complement (join the unconnected nodes and disconnect the connected nodes)

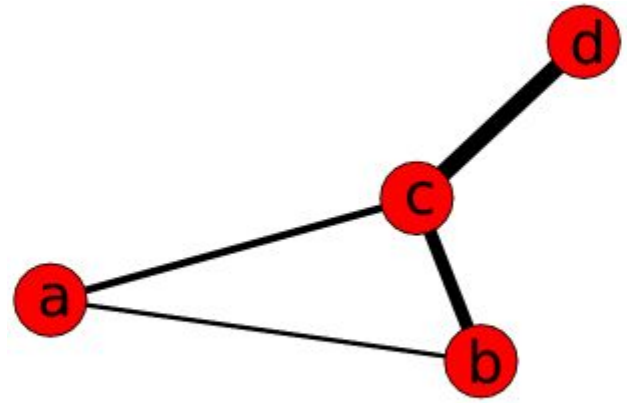
`create_empty_copy(G)` - return an empty copy of the same graph class

`convert_to_undirected(G)` - return an undirected representation of G

`convert_to_directed(G)` - return a directed representation of G

Shortest Path Example

Use Dijkstra’s algorithm to find the shortest path in a weighted and unweighted network:



```
g = nx.Graph()

g.add_edge(’a’, ’b’, weight=0.1)

g.add_edge(’b’, ’c’, weight=1.5)

g.add_edge(’a’, ’c’, weight=1.0)

g.add_edge(’c’, ’d’, weight=2.2)

print(nx.shortest_path(g, ’b’, ’d’))

[’b’, ’c’, ’d’]

print(nx.shortest_path(g, ’b’, ’d’, weighted=True))

[’b’, ’a’, ’c’, ’d’]
```

https://www.cl.cam.ac.uk/~cm542/teaching/2010/stna-pdfs/stna-lecture8.pdf

Graph Generators

Networkx supports many graph generator function while can create specific types of graphs

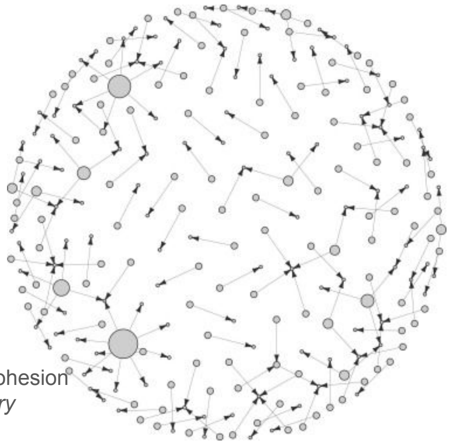
This includes random graphs which we can use like random numbers to model/test relationships

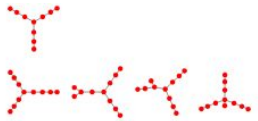
For example:

Cows are sold/moved between herds. We have empirical data on how often this happen and the average size of herd. We can generate random graphs using this data and then model the disease dynamics if one cow is infected.

Comper, J. Reilly, et al. "Descriptive network analysis and the influence of timescale on centrality and cohesion metrics from a system of between-herd dairy cow movements in Ontario, Canada." *Preventive Veterinary Medicine* 213 (2023): 105861.

```
# random graphs

er=nx.erdos_renyi_graph(100, 0.15)

ws=nx.watts_strogatz_graph(30, 3, 0.1)

ba=nx.barabasi_albert_graph(100, 5)

red=nx.random_lobster(100, 0.9, 0.9)
```



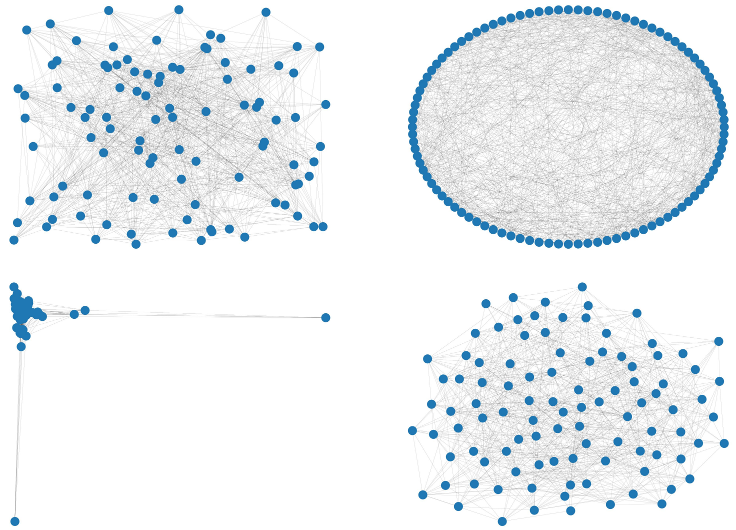


A lobster is a tree that reduces to a caterpillar when pruning all leaf nodes. A caterpillar is a tree that reduces to a path graph when pruning all leaf nodes.

Graph Drawing

NetworkX is not primarily a graph drawing package but it provides basic drawing capabilities by using Matplotlib.

Drawing a graph requires providing a layout:



```
import matplotlib.pyplot as plt

g = nx.erdos_renyi_graph(100,0.15)

nx.draw(g)

nx.draw_random(g)

nx.draw_circular(g)

nx.draw_spectral(g)

nx.draw_forceatlas2(g)

plt.savefig(“all_graphs_composed.png”)
```

Network Analysis: Centrality

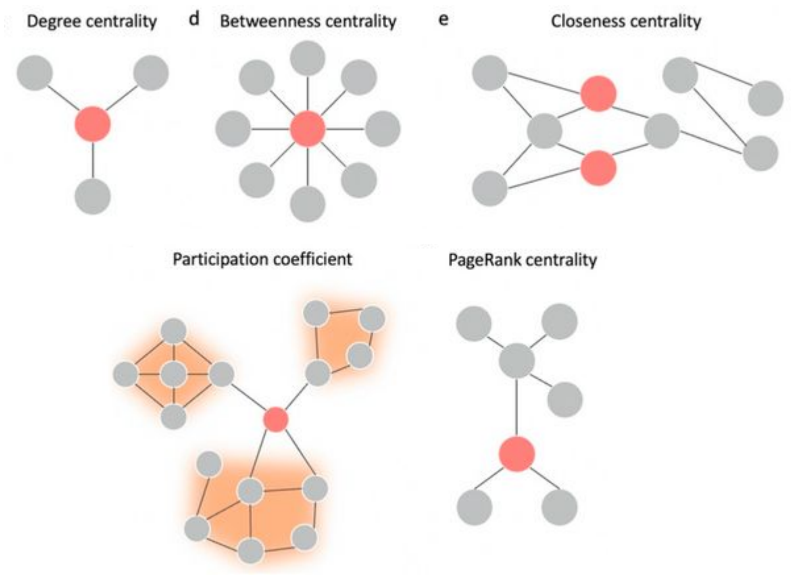
Often we are interested in finding the "most important" nodes in a network

Who are the key spreaders of disease?

Which servers going offline will take down the entire network?

Who socially connects different groups of people?

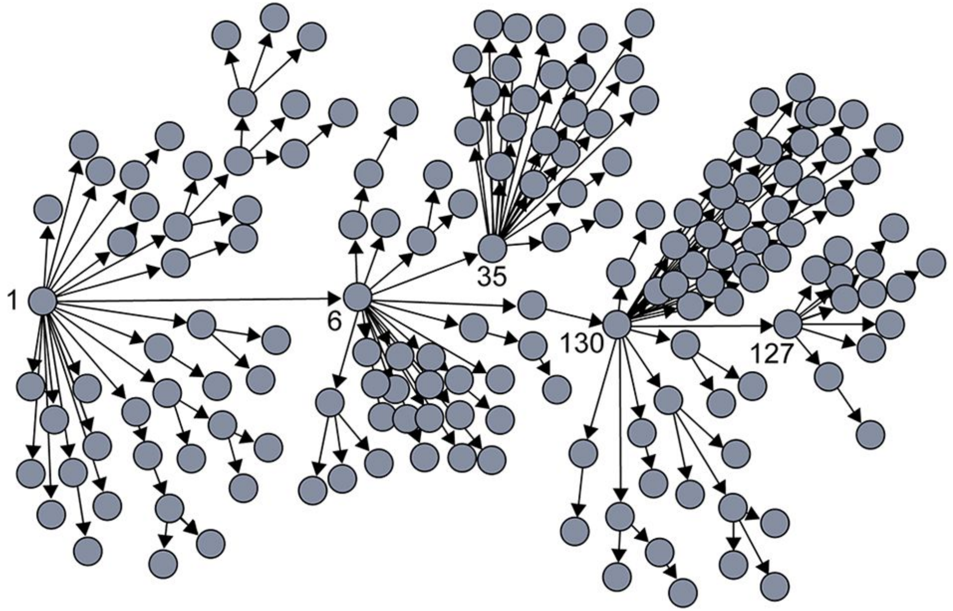
Lots of different ways to measure this **centrality**



Tanglay, Onur, et al. "Graph theory measures and their application to neurosurgical eloquence." Cancers 15.2 (2023): 556.

Identifying superspreader in contact network

```
G = nx.read_graphml("sars1_contact.gml")

out_cent = nx.out_degree_centrality(G)

between = nx.betweenness_centrality(G)

pagerank = nx.pagerank(G)

top = sorted([(node, measure) for node, measure in
pagerank.items()], key=lambda x: x[1], reverse=True)

print(top[0:5])

1

6

35

130

127
```



https://journals.plos.org/plospathogens/article/figure?id=10.1371/journal.ppat.1004092.g001#ppat.1004092-Stein1

Interlude: what would you change about this course?

Example Final Questions

Mid-Term Style Questions 1

If x = ‘2’ and y = ‘3’, what will the following evaluate to:

```
x*y

int(x)*int(y)

x*int(y)

x+y

x+int(y)

int(x)+int(y)
```

Mid-Term Style Questions 1

```
If x = ‘2’ and y = ‘3’, what will the following evaluate to:
```

```
x*y  # TypeError

int(x)*int(y)    # 6

x*int(y)  # ‘222’

x+y # ‘23’

x+int(y)  # TypeError

int(x)+int(y)  # 5
```

Mid-Term Style Questions 2

```python
def process_sequence(sequence):

    result = []

    for i in range(len(sequence)):

        if i == 0 or i == len(sequence)-1:

            result.append(sequence[i])

        else:

            result.append((sequence[i-1] + sequence[i] + sequence[i+1])/3)

    return result


data = [10, 20, 30, 40, 50]

processed = process_sequence(data)

print(processed)
```

What will this code print and why?

Mid-Term Style Questions 2

```
def process_sequence(sequence):

    result = []

    for i in range(len(sequence)):

        if i == 0 or i == len(sequence)-1:

            result.append(sequence[i])

        else:

            result.append((sequence[i-1] + sequence[i] + sequence[i+1])/3)

    return result


data = [10, 20, 30, 40, 50]

processed = process_sequence(data)

print(processed)
```

What will this code print and why?

List containing: [10, 20.0, 30.0, 40.0, 50]

First and last values will just be appended so kept as integers

Other values will return the mean of before + self + after which as input is even is just float of input

Explanatory Questions 1

Explain why you need to be careful if you use mutable datatypes as the default value for keyword arguments?

Explanatory Questions 1

Explain why you need to be careful if you use mutable datatypes as the default value for keyword arguments?

Default parameter values are evaluated only once when the function is defined, not each time the function is called. This means the same mutable object is reused across all function calls. So if the object is changed that side-effect propagates through all calls of that function

Explanatory Questions 2

Explain the difference between supervised and unsupervised learning. Provide an example of how each might be applied to analyze data in a scientific context.

Explanatory Questions 2

Explain the difference between supervised and unsupervised learning. Provide an example of how each might be applied to analyze data in a scientific context.

***Supervised Learning** is a type of machine learning where the algorithm learns from labeled training data. The algorithm is provided with input features and their corresponding output labels, and it learns to map inputs to outputs (e.g., regression and classification).*

Example: Predicting phenotypic antibiotic resistance from a bacterial genome

***Unsupervised Learning** is where the algorithm learns patterns from unlabeled data. Without output labels, the algorithm must find structure in the input data on its own (e.g., clustering or dimensionality reduction).*

Example: Searching for groups of participants in a psychological study based on their answers

Explanatory Questions 3

Explain the concept of reproducible research in computational science. Describe at least three Python features or practices that can help to make computational research more reproducible.

Explanatory Questions 3

Explain the concept of reproducible research in computational science. Describe at least three Python features or practices that can help to make computational research more reproducible.

Reproducible research refers to the ability of other researchers (or your future self) to recreate the exact same results from your computational analysis using the same data. This is both a critical aspect of the scientific method and should be bare minimum.

Python supports reproducibility through features such as:

- *Jupyter notebooks (which support documented re-runnable analyses)*
- *Comments/docstrings (explaining code)*
- *Random seeds (reproducible pseudorandom numbers)*
- *Automating data processing/visualisation in notebooks or scripts (avoiding manual/undocumented steps in your analyses)*
- *Modularising code to avoid copy and pasting errors and making it easier to apply the same code many times via functions & modules/packages*

Bug Hunting 1

Identify and fix the bug in this function so that it correctly calculates the mean temperature.

```
def calculate_mean(temperatures):

    total = 0

    for temp in temperatures:

        total = total + temp

    return total / len(temperatures)



temperatures = ['22.5', '23.1', '21.9', '22.8', '23.5']

mean_temp = calculate_mean(temperatures)

print(f"The mean temperature is {mean_temp} degrees
Celsius.")
```

Bug Hunting 1

Identify and fix the bug in this function so that it correctly calculates the mean temperature.

Convert temperatures to floats from strings (alternatively add exceptions/try)

```
def calculate_mean(temperatures):

    total = 0

    for temp in temperatures:

        total = total + float(temp)

    return total / len(temperatures)



temperatures = ['22.5', '23.1', '21.9', '22.8', '23.5']

mean_temp = calculate_mean(temperatures)

print(f"The mean temperature is {mean_temp} degrees
Celsius.")
```

Bug Hunting 2

Work out why this function will not return the correct values when evaluated using the test cases at the bottom. Fix this issue.

```python
def classify_growth_rate(doubling_time):

    if doubling_time <= 1.0:

        growth_rate = "high"

    if doubling_time <= 3.0:

        growth_rate = "medium"

    if doubling_time > 3.0:

        growth_rate = "low"

    return growth_rate

print(classify_growth_rate(0.5))  # Should print "high"

print(classify_growth_rate(2.0))  # Should print "medium"

print(classify_growth_rate(5.0))  # Should print "low"
```

Bug Hunting 2

Work out why this function will not return the correct values when evaluated using the test cases at the bottom. Fix this issue.

Convert to if-elif-else (or if elif-elif) statement instead of change of if's

```python
def classify_growth_rate(doubling_time):

    if doubling_time <= 1.0:

        growth_rate = "high"

    elif doubling_time <= 3.0:

        growth_rate = "medium"

    else:

        growth_rate = "low"

    return growth_rate

print(classify_growth_rate(0.5))  # Should print "high"

print(classify_growth_rate(2.0))  # Should print "medium"

print(classify_growth_rate(5.0))  # Should print "low"
```

Bug Hunting 3

This code uses functional programming concepts to filter a list of experimental samples based on certain criteria and then calculate their mean value.

Find and fix the bug.

```python
def process_experimental_data(data):
    # Filter samples with values above 50
    filtered_data = filter(lambda x: x['value'] > 50, data)
    # Extract just the values
    values = map(lambda x: x['value'], filtered_data)
    # Calculate the mean
    return sum(values) / len(values)


samples = [
    {'id': 1, 'value': 45.2, 'timestamp': '2024-03-15'},
    {'id': 2, 'value': 67.8, 'timestamp': '2024-03-15'},
    {'id': 3, 'value': 52.3, 'timestamp': '2024-03-16'},
    {'id': 4, 'value': 39.1, 'timestamp': '2024-03-16'},
    {'id': 5, 'value': 88.5, 'timestamp': '2024-03-17'}
]
mean_value = process_experimental_data(samples)
```

Bug Hunting 3

This code uses functional programming concepts to filter a list of experimental samples based on certain criteria and then calculate their mean value.

Find and fix the bug.

***Bug:** The bug occurs because filter() and map() return iterators (specifically generators), not lists. When len(values) is called there is a TypeError as these iterators don't implement `__len__` (and even if they did then the values iterator would have been consumed by the sum() function)*

```python
def process_experimental_data(data):

    # Filter samples with values above 50

    filtered_data = filter(lambda x: x['value'] > 50, data)

    # Extract just the values

    values = list(map(lambda x: x['value'], filtered_data))

    # Calculate the mean

    return sum(values) / len(values)



samples = [

    {'id': 1, 'value': 45.2, 'timestamp': '2024-03-15'},

    {'id': 2, 'value': 67.8, 'timestamp': '2024-03-15'},

    {'id': 3, 'value': 52.3, 'timestamp': '2024-03-16'},

    {'id': 4, 'value': 39.1, 'timestamp': '2024-03-16'},

    {'id': 5, 'value': 88.5, 'timestamp': '2024-03-17'}

]

mean_value = process_experimental_data(samples)
```

Overview

- Graph-structured Data
- Networks/Graphs Basics
- NetworkX
- Nodes/Edges in NetworkX
- Network I/O
- Network Visualization
- Centrality Analyses
- Final Exam Question Examples
    - Mid-Term Style Questions
    - Explain This Concept questions
    - Fix This Bug Questions