Network Science
Autumn 2021

Notes on these slides

Examinable material: You will not be asked to analyze or write any Python code or pseudocode. You will not be tested on your understanding of NetworkX. The material on “Diffusion in 1D” in lecture 10 and “Network science and climate science” at the end of lecture 16 will also not be examined.

Contents

Network Science
Autumn 2021

Lecture 1

Module structure
Distances in a real-world network
Lattice graphs
Introduction to NetworkX

People

Module leader: Prasun Ray
p.ray@imperial.ac.uk

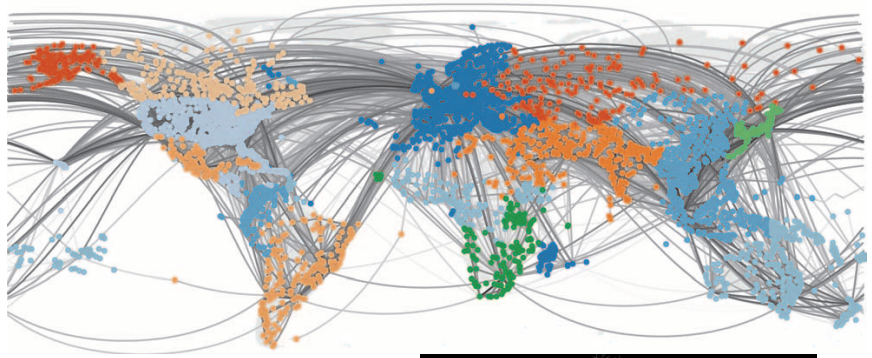
GTAs:
Victoria Klein
Ash Myall
Thanos Margaritis
Xing Liu
Hasnat Kazi

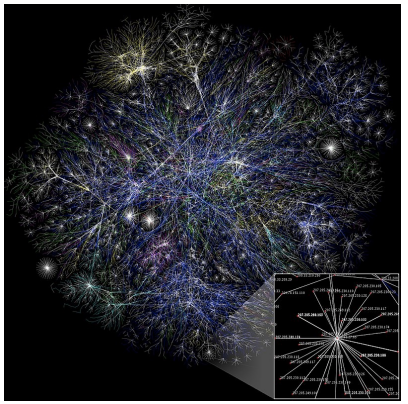
Network Science overview

The science of networks is an important, rapidly growing field

Examples of significant networks:

- **Twitter, Facebook**
- **Air transportation network**
- **World-wide web**
- **Human brain**





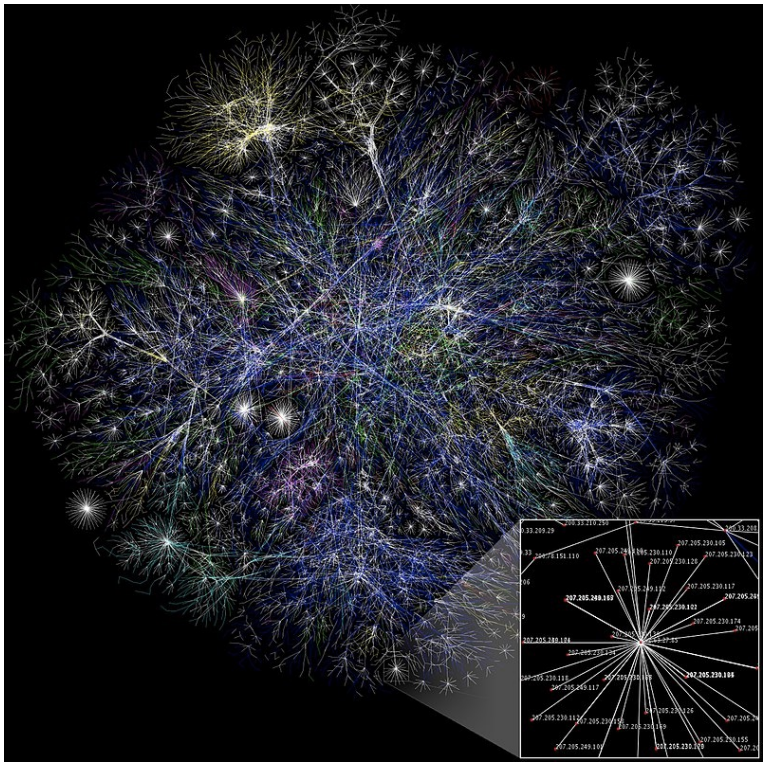
images from:
https://en.wikipedia.org/wiki/Complex_network
D.S. Bassett, How You Think: Structural Network Mechanisms of Human Brain Function
Brockmann & Helbing, The Hidden Geometry of Network-Driven Contagion Phenomena

Ultimately, the aim is to understand how the structure of a network influences its functionality

- **E.g. how does the structure of the air transportation network influence a global pandemic?**

This requires careful thinking about:

- **graph theory**
- **probability**
- **statistics**
- **linear algebra**
- **differential equations**
- **algorithms**



https://en.wikipedia.org/wiki/Complex_network

"Coincidentally" this strongly overlaps with your 1st-year modules!

Syllabus

- **Weeks 2-3: Graph properties and structure; using NetworkX**
- **Weeks 3-5: Random graph models**
- **Weeks 6-9: Dynamics on graphs: modeling, analysis, and simulation**
- **Week 10-11: Communities, community detection, networks & data science**

Questions we will consider:

- How does Google order your search results?
- How can we predict the spread of an infectious disease in a community social network?
- How can we identify communities in a pod of dolphins?
- How can we model the evolution of the Facebook friends network?

Module structure

- A detailed module guide has been posted on Blackboard, I'll just provide a quick overview here
- This module runs weeks 2-11 of term
- We have 9 live lectures (also streamed via Teams and recorded) and 7 pre-recorded lectures
    - So some weeks will have 2 lectures and some will have 1. I will post an outline of the week ahead on Blackboard at the beginning of each week (starting next Sunday)
- Slides will be posted on Blackboard in advance of live lectures and at the same time as pre-recorded lectures
- A single navigable pdf with all slides will be provided at the beginning of the exam revision period at the beginning of April. These are the typed lecture notes for the module.

- We will also have 6 computer labs (in the MLC) and 5 problem classes
  - The first computer lab is this Friday and the first problem class is next Thursday (see the module guide)
- There are weekly office hours on Teams: Tuesdays, 3-3:40pm
- Problem sheets will typically be posted on Mondays around noon.
- This week: Monday lecture, Tuesday office hour, and Friday computer lab

Assessment

2 Projects (10%, 20%)

1 Midterm (10%)

Final exam (60%)

Projects will contain substantial computational and open-ended components

Tentative project dates:

Project 1: Assigned 29/10, due 5/11

Project 2: Assigned 3/12 due 17/12

Online resources

- **Module Blackboard page**
  - **All course material will be posted here**
- **Ed discussion board (similar to Piazza):** https://edstem.org/us/courses/15185/discussion/
  Ask (and answer) questions on nearly all module-related topics here
  - **Questions on Ed will be prioritized over emails**
  - **You can post questions privately (only instructors/GTAs will see them)**
  - **You can also post questions where you are anonymous to other classmates (but not to instructors/GTAs)**
- **Microsoft Teams page:** MATH50007 - Network Science (Autumn 2021-2022)
  - **Meetings will be started here for live lectures and office hours**

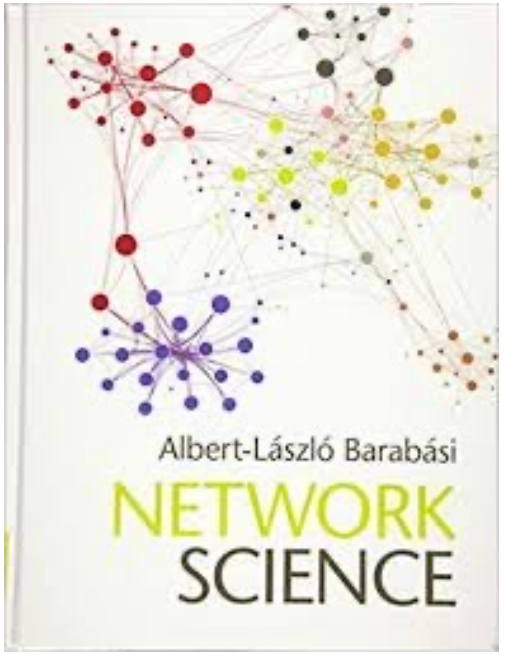
Computing

We will be using Python and it is recommended that you use/install the standard Anaconda package: https://www.anaconda.com/products/individual

- **This will give you the Spyder IDE and all of the packages that you need (Numpy, Scipy, Matplotlib, NetworkX, Pandas,…)**
- **For more information on software installation:** https://imperial-fons-computing.github.io/
- **If you have installation/software/hardware problems, please post your issue on Ed within the "Computer issues" category**

Reading

We will use Network Science **by Barabasi, one of the pioneers in the field**



- **The book is freely available online:** http://networksciencebook.com/
- **It provides great context and motivation for most of the module topics with frequent comparisons with data from real networks**
- **It is *not* written for mathematicians and much of the mathematical development lacks rigor – the lectures will attempt to compensate for this**
- **You should read chapters 2 of the book this week**
- **Afterwards, the required reading will be substantially reduced**
- **If you are unsure of what Network Science is, browse through chapter 1**

Getting started (week 1)

- **Make sure you are happy with your access to Python (terminal + editor)**
- **Read Chapter 2 of Barabasi:** http://networksciencebook.com/
  - **Much of the material will be familiar and/or straightforward, and you should go through those parts quickly**
  - **You can skip the "Boxes" and historical discussions**
  - **Don't skip section 2.13**
- **Ask questions during the Tuesday office hour, during the computer lab on Friday, or on Ed**

Distances in real-world networks

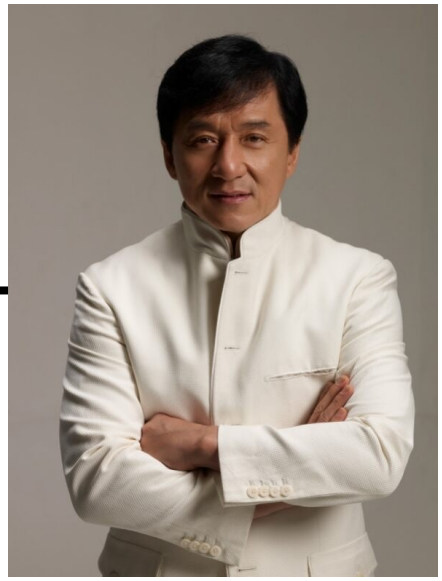
- **The title of this module is Network *Science***
  - **So we should always keep our minds on important questions related to the real world**
  - **For example, how close (or far apart) are Bruce Lee and Beyoncé?**

Bruce Lee, famous American actor and martial artist

Beyoncé, famous American actress and singer

- We first have to define “distance”
- Let’s say two actors are “linked” if they have appeared in the same film
  - We could then place a weight on a link based on the number of shared films
  - For example, Jackie Chan was a stuntman in 3 Bruce Lee films, so:

3

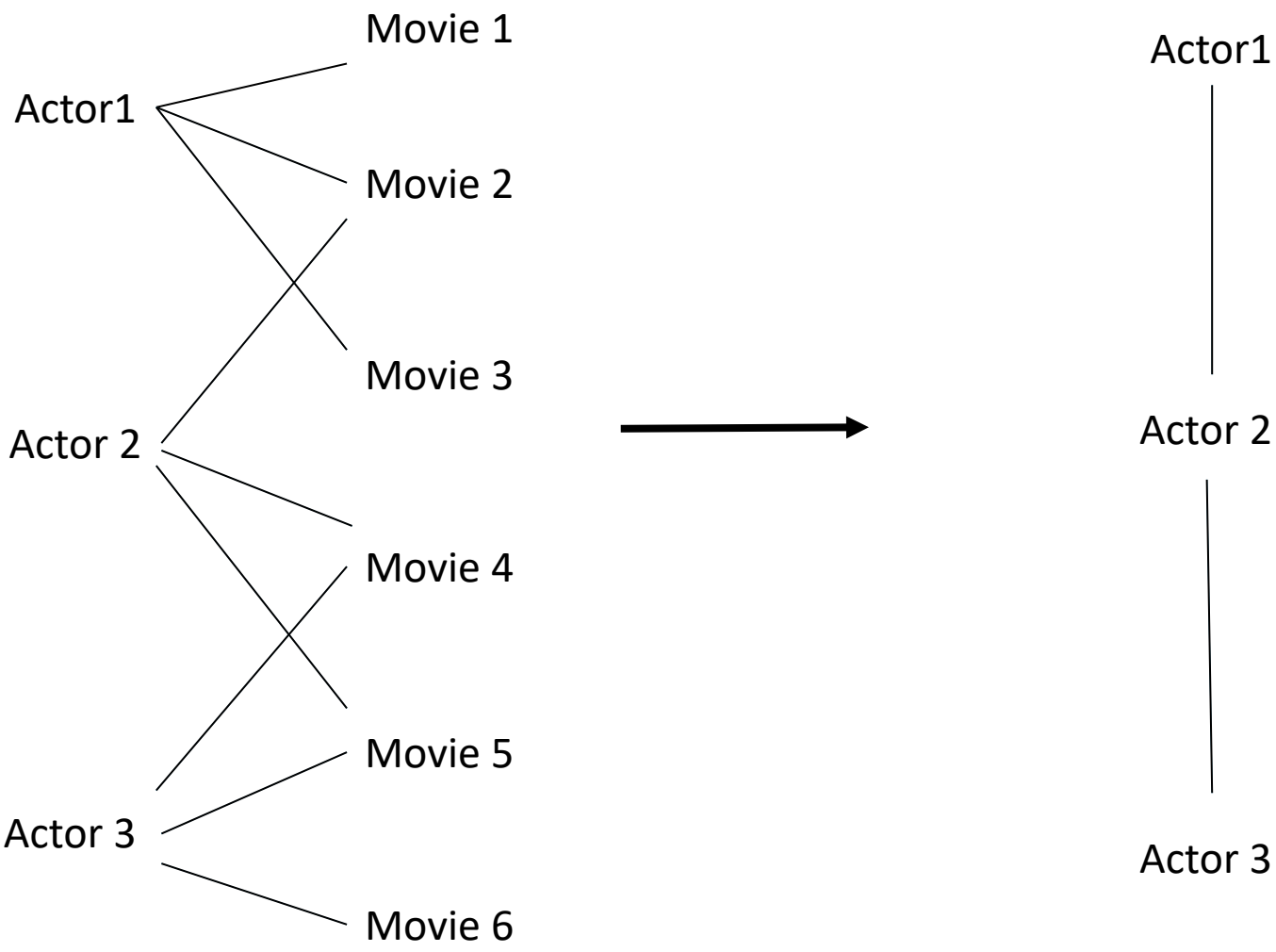


- **Let's ignore the edge weights, and count the smallest number of links to get from Bruce Lee to Beyoncé:**

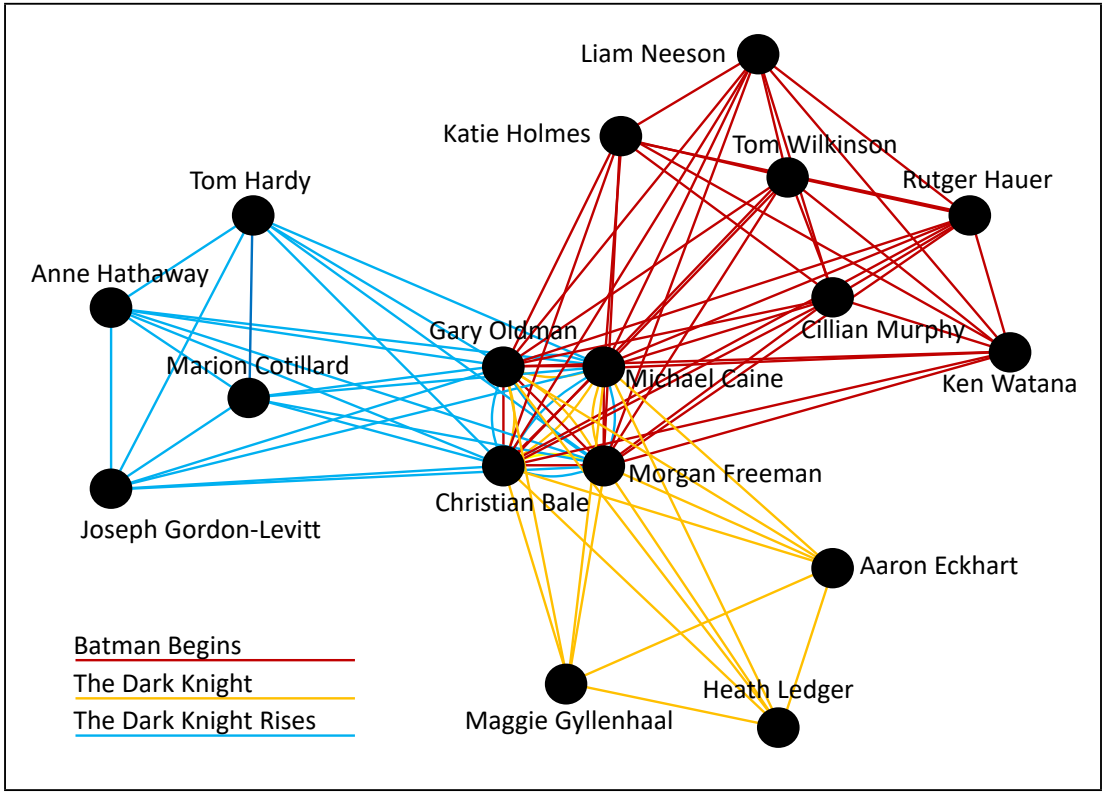
Image generated at oracleofbacon.org

- **We see that there is a path of distance 3 from Beyoncé to Bruce Lee**
- **Which is shorter than what I would have guessed!**
- **And there is more than one shortest path**
- **It is possible to study a much larger group of actors (say all actors in IMDB) and construct a *collaboration* graph**

- **A collaboration graph is a single-graph projection of a *bipartite* actor-movie graph:**



Here's a small example:



From: R. Lewis, Who is the Centre of the Movie Universe?

- Our Bruce Lee – Beyoncé question is a variation of the “6 degrees of Kevin Bacon” game
- The concept is that any (Hollywood) actor can be reached from Kevin Bacon via links in the actor collaboration network in 6 or less steps
- The underlying idea is that distances in large social networks tend to be “small”
  - And short distances are found in a large variety of complex networks
  - We will make this idea of “short distances” more precise soon

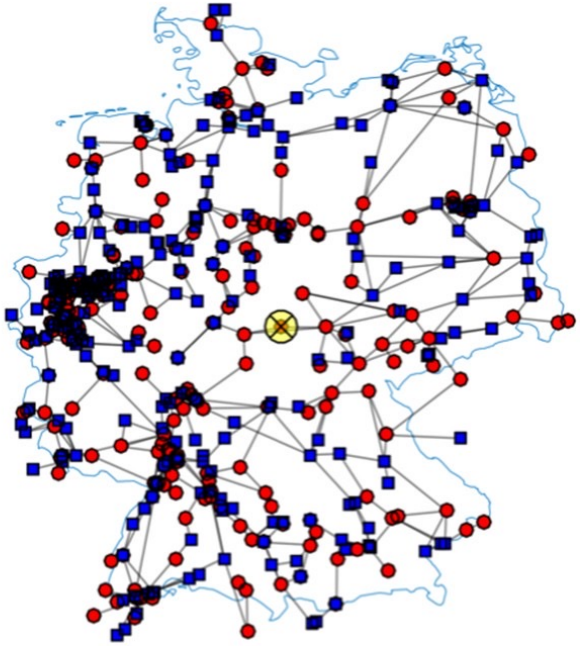
Kevin Bacon, famous American actor

- 6 degrees of Kevin Bacon is a play on the idea of 6 degrees of separation which comes from the work of social psychologist Stanley Milgram in the 1960s
- The claim is that any 2 people are no more than 6 links apart in the global social network of acquaintances
- Network Science is generally considered to be about 20 years old, but this ignores the foundation provided by social network analysis which is considerably older

Question for you: given data for the IMDB actor collaboration network, what would you want to explore and analyze?

Simple networks

- We are primarily interested in large *complex* networks like a national power grid →
- But we should first make sure we can analyze simpler cases!
- This will build intuition about useful quantities like the diameter and clustering coefficient
- And later, will provide a reference when analyzing complicated real-world problems



From: Tamrakar, S., Conrath, M. & Kettemann, S. Propagation of Disturbances in AC Electricity Grids. *Sci Rep* **8,** 6459 (2018)

Warm-up question:

- **What does this adjacency matrix→ correspond to?**

($A_{ij} = 1$ if there is a link to node i from node j and $A_{ij} = 0$ otherwise)

$$\mathrm{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

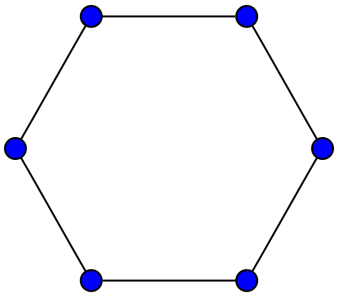
Warm-up question:

- **What does this adjacency matrix→ correspond to?**

($A_{ij} = 1$ if there is a link to node i from node j and $A_{ij} = 0$ otherwise)

$$\mathrm{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

- **This is a *cycle graph* with $N = 6$ nodes and $L = 6$ links**
  - **This is an *unweighted, undirected* graph**

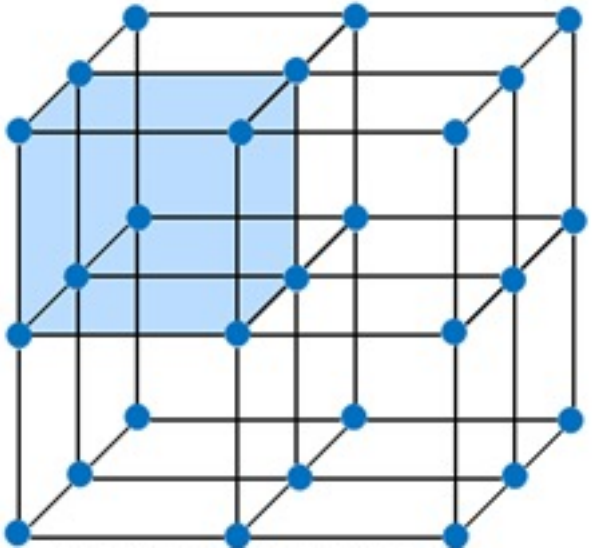


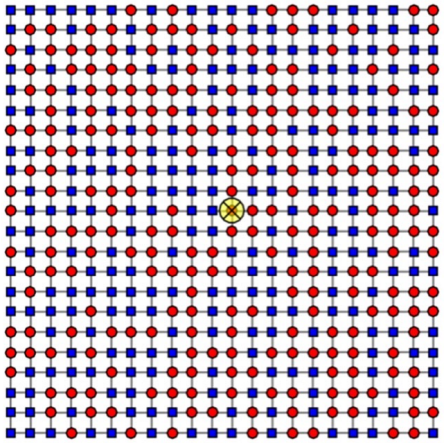
- **And if we set the "top-right" and "lower-left" elements to zero, we have a chain →**

Simple example #1: lattices

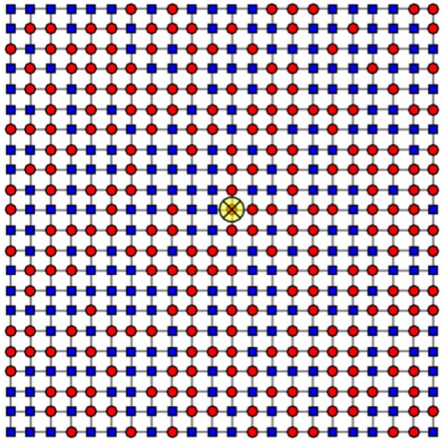
We can think of a chain as a “1-D lattice”

The two- and three-dimensional versions are both familiar





- **The adjacency matrix of a 2D lattice (A_2) has a block tridiagonal structure as shown below**
- **Here, A_1 is the adjacency matrix for an m-node 1D lattice, I is the $m \times m$ identity matrix, and the 2D lattice has $N = m^2$ nodes**

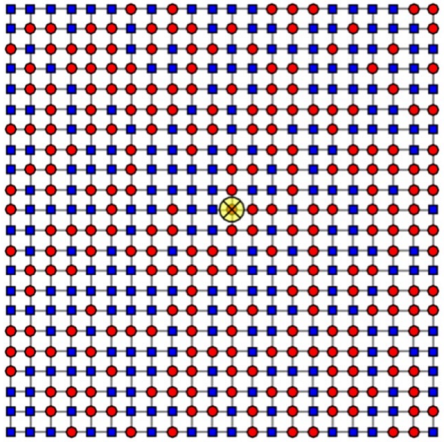


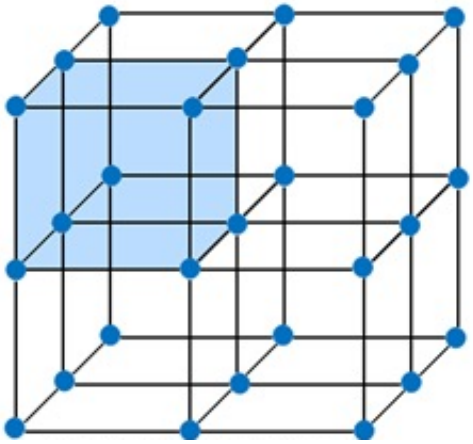
Optional exercise: **what is the structure of the adjacency matrix for a 3D lattice?**

$$\mathrm{A}_2 = \begin{pmatrix} \mathrm{A}_1 & \mathrm{I} & & & \\ \mathrm{I} & \mathrm{A}_1 & \mathrm{I} & & \\ & \ddots & \ddots & \ddots & \\ & & \mathrm{I} & \mathrm{A}_1 & \mathrm{I} \\ & & & \mathrm{A}_1 & \mathrm{I} \end{pmatrix}$$

Let's now think about node *degrees* in lattices. The degree of a node is the number of links it has.

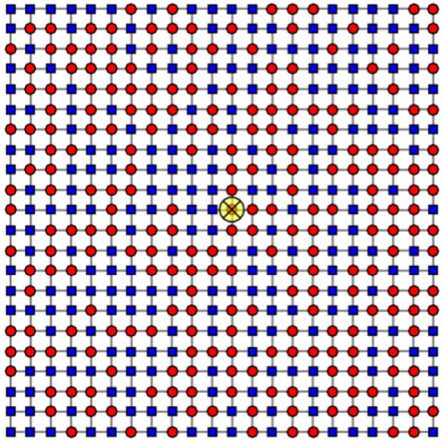
- **First consider "interior" nodes (as opposed to "boundary" nodes)**
- **Interior nodes have degree, $k = 2d$, where d is the lattice dimension**
- **The degrees of boundary nodes depend on the dimension**
- **For $d = 3$ and $N = m^3$ nodes, the 8 corner nodes have $k = 3$, and all other boundary nodes have $k = 4$**



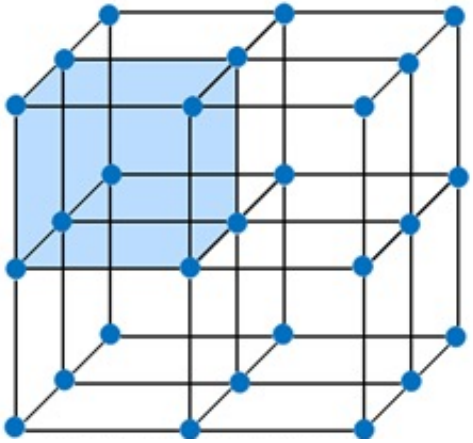


Now, let's think about the graph *diameter,* D, the longest shortest path between two nodes (assuming that the graph is *connected*)

- **For** d-dimensional **lattices, with** $N = m^d$ **nodes, we have** $D = d(m-1)$

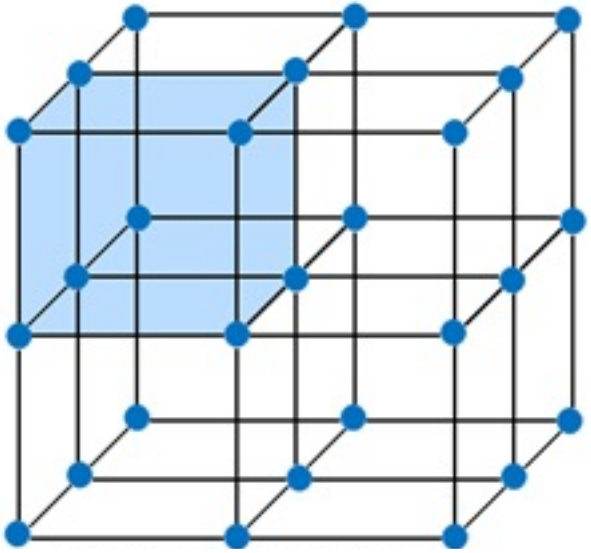


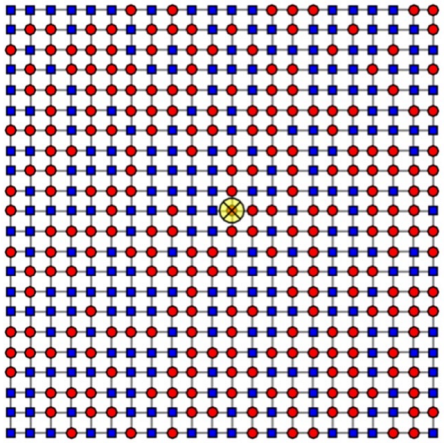
- **What we are interested in generally is how the diameter scales with** N**, particularly as** N **becomes large**
- **Rearranging the above expressions, we find,** $D = d\,N^{1/d} - d$
- **So for large** N**,** $D \sim N^{1/d}$



- **Are distances "short" in lattices? No, we will look at another simple model next lecture with different behavior.**

Chapter 2 of Barabasi introduces the local, average, and global clustering coefficients. Check your understanding by evaluating the following claim: ***The average and global clustering coefficients for these lattices are zero.***





Note: **Here we have only considered "rectangular" lattices. Lattices are constructed via regular tilings, and we could, for example, construct a *triangular* lattice**

A quick aside:

- **1- and 2D rectangular lattices are frequently found in manmade networks (e.g. small computer networks, planned cities)**
- **3D lattices are commonly found in nature -- crystals and crystalline solids have lattice-like molecular structure**
- **But we're mainly using them as preparation for analysis of complex networks…**

https://www.andrewalexanderprice.com/images/blog20-14.jpg

Galena Quartz

Pyrite

https://saylordotorg.github.io/text_general-chemistry-principles-patterns-and-applications-v1.0/

NetworkX

Network	Nodes	Links	Directed / Undirected	N	L	‹K›
Internet	Routers	Internet connections	Undirected	192,244	609,066	6.34
WWW	Webpages	Links	Directed	325,729	1,497,134	4.60
Power Grid	Power plants, transformers	Cables	Undirected	4,941	6,594	2.67
Mobile-Phone Calls	Subscribers	Calls	Directed	36,595	91,826	2.51
Email	Email addresses	Emails	Directed	57,194	103,731	1.81
Science Collaboration	Scientists	Co-authorships	Undirected	23,133	93,437	8.08
Actor Network	Actors	Co-acting	Undirected	702,388	29,397,908	83.71
Citation Network	Papers	Citations	Directed	449,673	4,689,479	10.43
E. Coli Metabolism	Metabolites	Chemical reactions	Directed	1,039	5,802	5.58
Protein Interactions	Proteins	Binding interactions	Undirected	2,018	2,930	2.90

Table 2.1

Canonical Network Maps

The basic characteristics of ten networks used throughout this book to illustrate the tools of network science. The table lists the nature of their nodes and links, indicating if links are directed or undirected, the number of nodes (N) and links (L), and the average degree for each network. For directed networks the average degree shown is the average in- or out-degrees $\langle k \rangle = \langle k_{in} \rangle = \langle k_{out} \rangle$ (see Equation (2.5)).

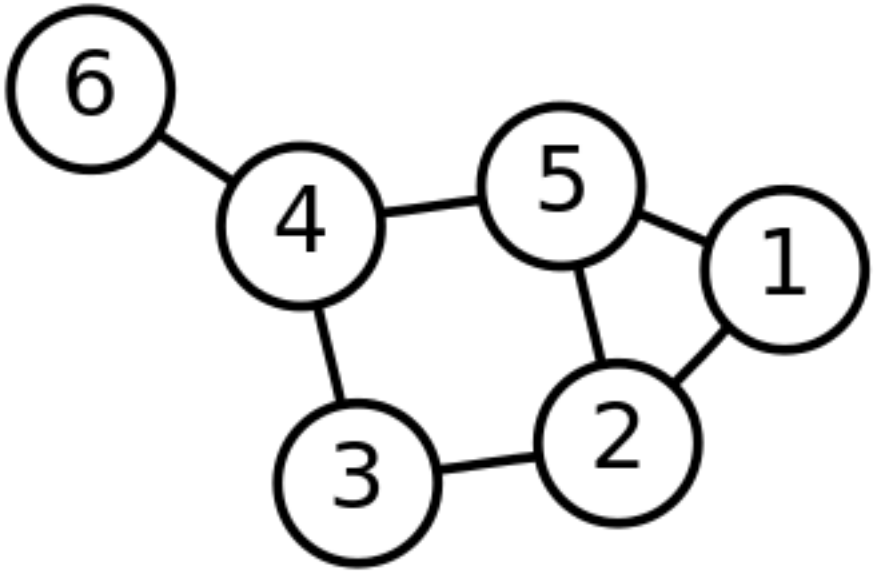
We are generally interested in large *complex* networks

Analysis of such networks can be complicated and expensive (classical example: computing shortest path between nodes)

NetworkX package provides a suite of tools for working with complex networks

More generally: avoid writing own code whenever possible! Many powerful highly-efficient libraries are available

NetworkX: basics

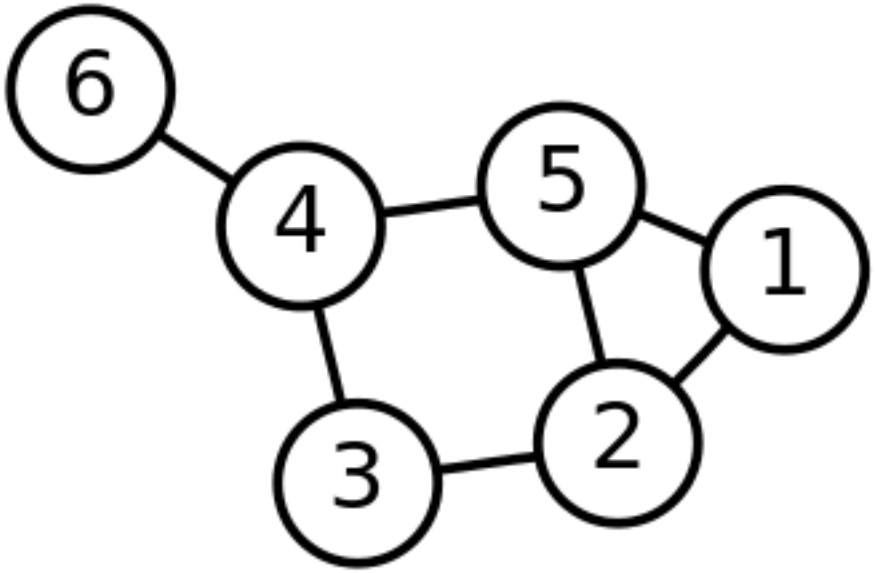


- **Let's work with this graph in NetworkX**
- **First, import the module, and initialize a graph:**

```
In [55]: import networkx as nx

In [56]: G = nx.Graph()
```

- **There are numerous methods for building a graph**

```
In [57]: G.add_edge(1,2)

In [58]: G.edges()
Out[58]: [(1, 2)]

In [59]: G.nodes()
Out[59]: [1, 2]
```

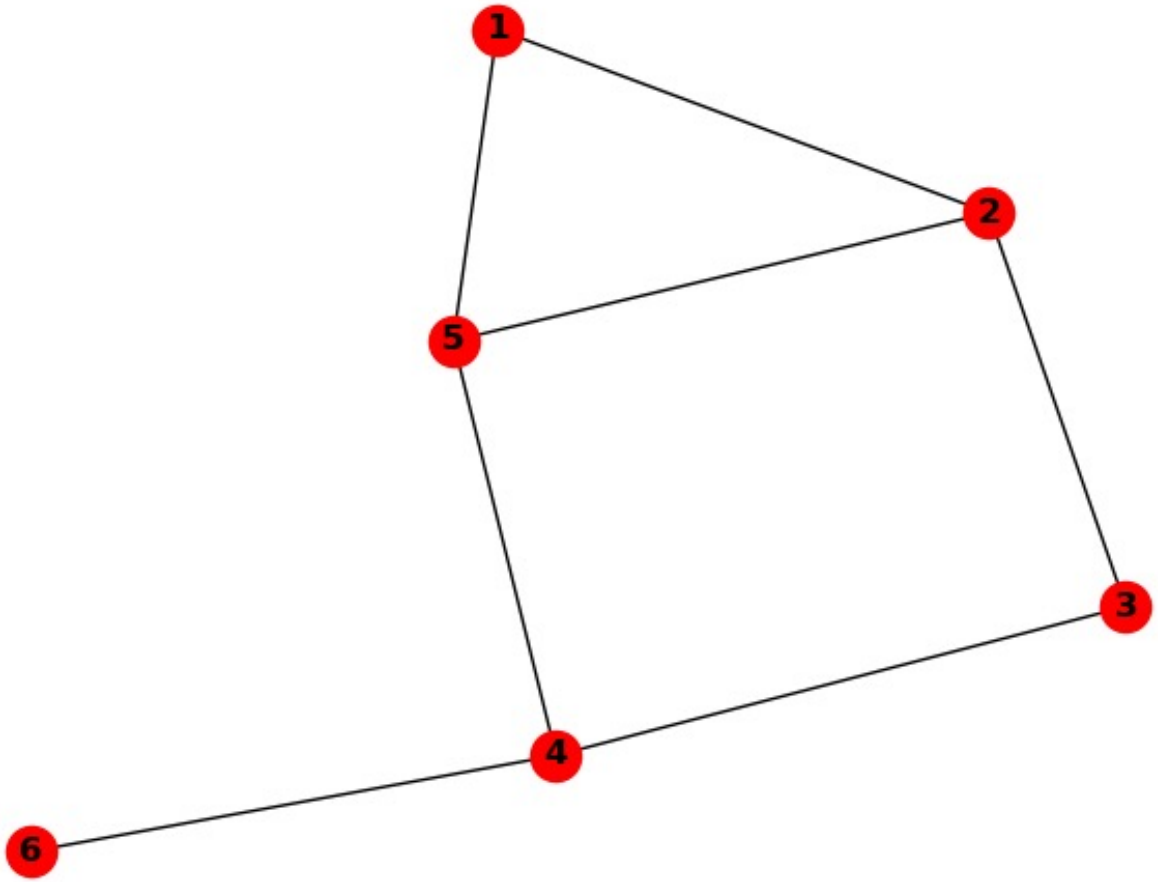


We can add several edges (or nodes) at once using an edge list:

```
In [65]: e = [(1,5),(2,5),(2,3),(3,4),(4,5),(4,6)]

In [66]: G.add_edges_from(e)

In [67]: G.edges()
Out[67]: [(1, 2), (1, 5), (2, 3), (2, 5), (5, 4), (3, 4), (4, 6)]

In [68]: G.nodes()
Out[68]: [1, 2, 5, 3, 4, 6]
```

Use nx.draw **to visualize the network:**

```
In [69]: import matplotlib.pyplot as plt

In [70]: plt.figure()
Out[70]: <matplotlib.figure.Figure at 0x1515e3fef0>

In [71]: nx.draw(G, with_labels=True, font_weight='bold')
```



We can now analyze the graph:

```
In [74]: A = nx.adjacency_matrix(G)


In [75]: type(A)
Out[75]: scipy.sparse.csr.csr_matrix


In [76]: A.toarray()
Out[76]:
array([[0, 1, 1, 0, 0, 0],
[1, 0, 1, 1, 0, 0],
[1, 1, 0, 0, 1, 0],
[0, 1, 0, 0, 1, 0],
[0, 0, 1, 1, 0, 1],
[0, 0, 0, 0, 1, 0]], dtype=int64)
```

- **Most complex networks are sparse, and the sparse format uses memory more efficiently**
- **And some calculations (e.g. matrix-vector product) are more efficient as well**

It is straightforward to compute the local clustering and to get the degree of each node:

```
In [57]:
nx.clustering(G)

Out[57]: {1: 1.0, 2: 0.3333333333333333, 3: 0, 4: 0, 5: 0.3333333333333333, 6: 0}


In [58]:
nx.degree(G)

Out[58]: DegreeView({1: 2, 2: 3, 3: 2, 4: 3, 5: 3, 6: 1})
```

The degree distribution, p_k, is also extremely important. For a single network, it is defined as:

p_k **= fraction of nodes with degree** k

The degree distribution can be computed using the output from nx.degree_histogram:

```
In [83]: nx.degree_histogram?
Signature: nx.degree_histogram(G)
Docstring:
Return a list of the frequency of each degree value.
Returns
-------
hist : list
A list of frequencies of degrees.
The degree values are the index in the list.

In [84]: h = nx.degree_histogram(G)

In [85]: h
Out[85]: [0, 1, 2, 3]
```

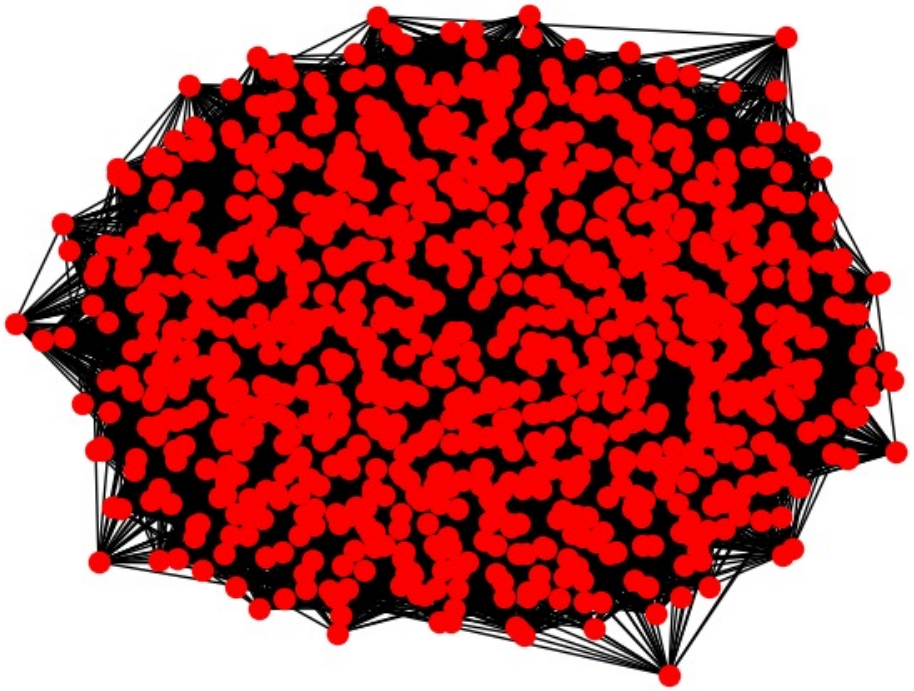
- **The *ith* element of** h **corresponds to the number of nodes with degree *i***

(degree distributions are more interesting for large networks!)

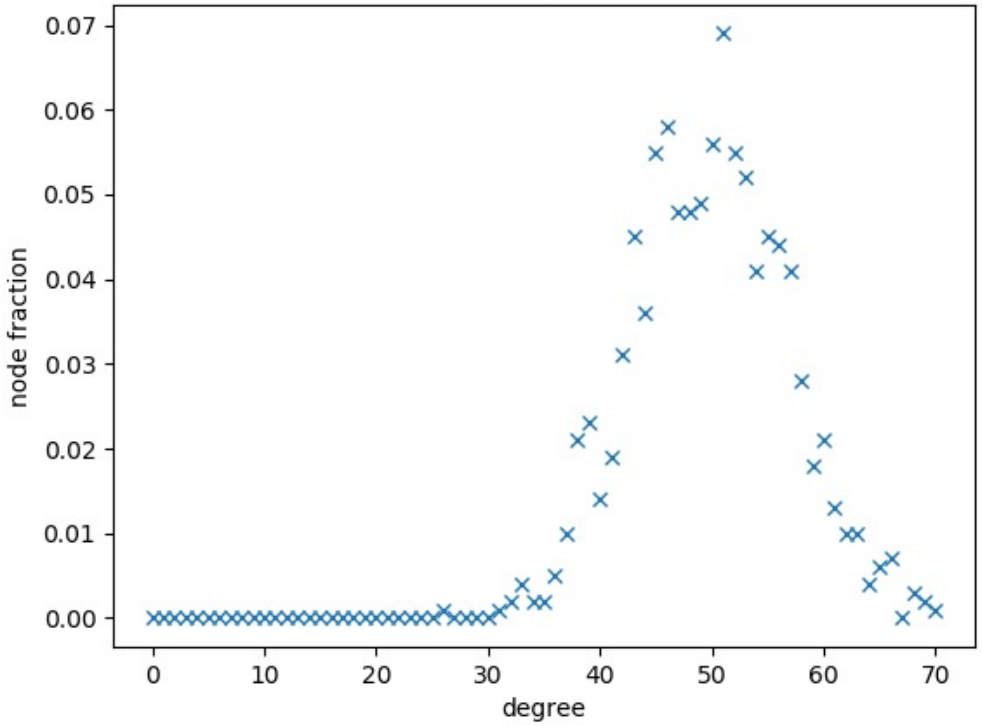
NetworkX also contains a number of functions for graph models

- **Here, we generate and visualize a graph generated with the G_{Np} random graph model: the graph has N nodes, and a link is placed between each pair of nodes with probability p**

```
In [119]: Grandom = nx.gnp_random_graph(1000,0.05)

In [120]: nx.draw(Grandom,node_shape='.')
```



Now the degree distribution is more interesting. We will analyze this distribution later in the term.



- **I should compute degree distributions for several graphs (with fixed N, p) and then average them**
- **Generally, when there is randomness in the problem, statistics are the quantities of interest (mean, variance, etc…)**

NetworkX: getting started

- **Read the online tutorial: https://networkx.github.io/documentation/stable/tutorial.html**
- **Browse through the online reference section: https://networkx.github.io/documentation/stable/reference/index.html**
- **Use NetworkX 2.x (I'm using 2.4)**
- **Come to lab 1 on Friday!**

Network Science
Autumn 2021

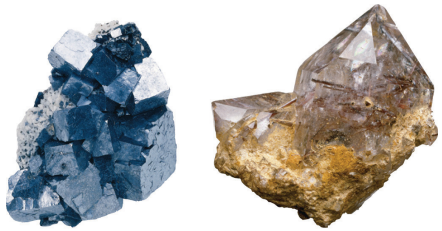
Lecture 2

Cayley graphs
Node centralities

Simple example #1: lattices

- 1- and 2D lattices are frequently found in manmade networks (e.g. small computer networks, planned cities)
- 3D lattices are commonly found in nature -- crystals and crystalline solids have lattice-like molecular structure
- But we're mainly using them as preparation for analysis of complex networks

https://www.andrewalexanderprice.com/images/blog20-14.jpg



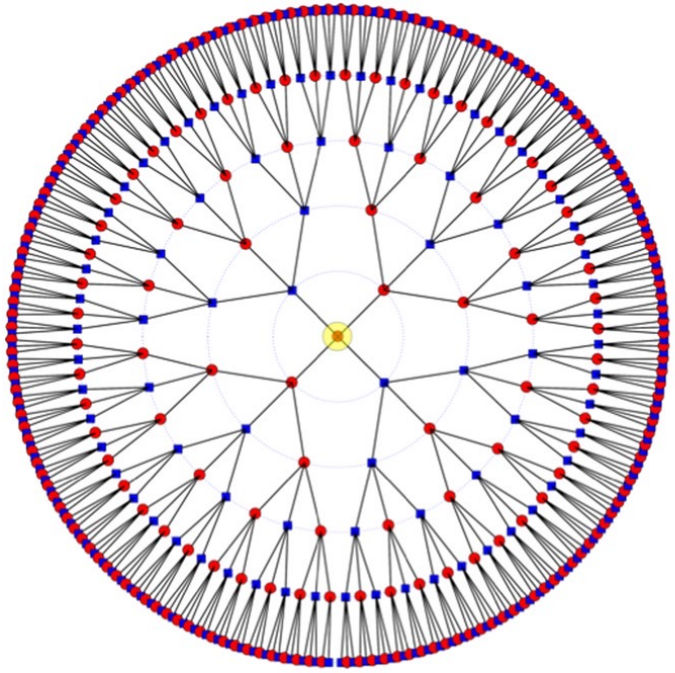
Galena Quartz

Pyrite

https://saylordotorg.github.io/text_general-chemistry-principles-patterns-and-applications-v1.0/

Simple example #2: Cayley trees

- The image shows a “4-regular” Cayley tree
  - Each node in the outer ring has degree=1
  - All other nodes have degree=4
  - A “tree” is a graph with no loops



Construction of a k-regular Cayley tree:

- Iteration 0: start with a single ‘root node’
- Iteration 1: add k nodes that link to the root
- Iteration i, $i > 1$: For each node n added during iteration $i-1$, add $k-1$ nodes which link to n

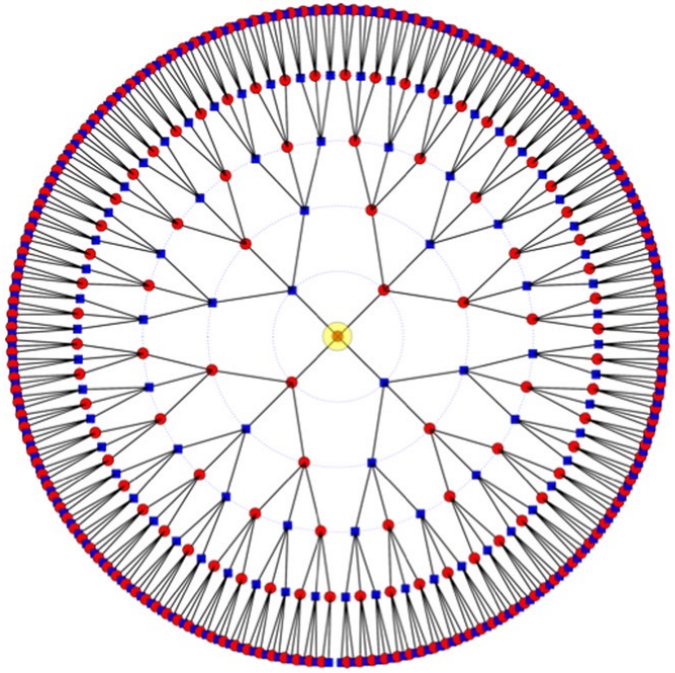
Check your understanding: how many iterations were used to make the graph pictured?

- **After r iterations, a k-Cayley tree has:**

$N = 1 + k(1 + b + b^2 + \cdots + b^{r-1})$ **nodes where** $b = k - 1$

This simplifies to, $N = 1 + \frac{k(b^r - 1)}{b-1}$

And there are $L = N - 1$ **links**



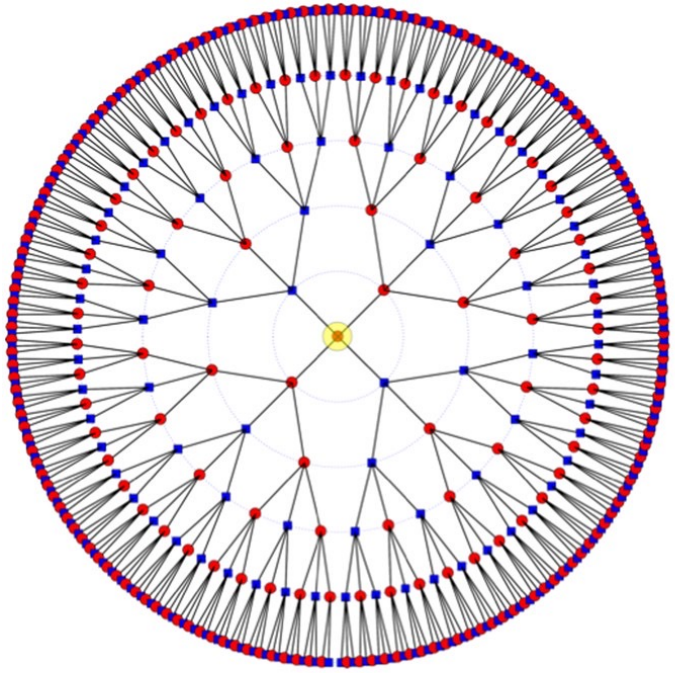
- **The global and average clustering coefficients for all Cayley trees is zero**
- **In fact, they are zero for *all* trees**
- **What about the diameter?**

- **The longest shortest path is a path from a leaf from the final iteration through the root node out to another leaf**
- **So we have $D = 2r$ and we want to rearrange this for comparison with our result for lattices**
- **From our expression for N, we find:**

$$r = \frac{D}{2} = \frac{\log[(N-1)(b-1)+k]-\log(k)}{\log(b)}$$

- **For large N, the diameter can be approximated as,**

$$D \approx 2\frac{\log(N)}{\log(b)} + const$$



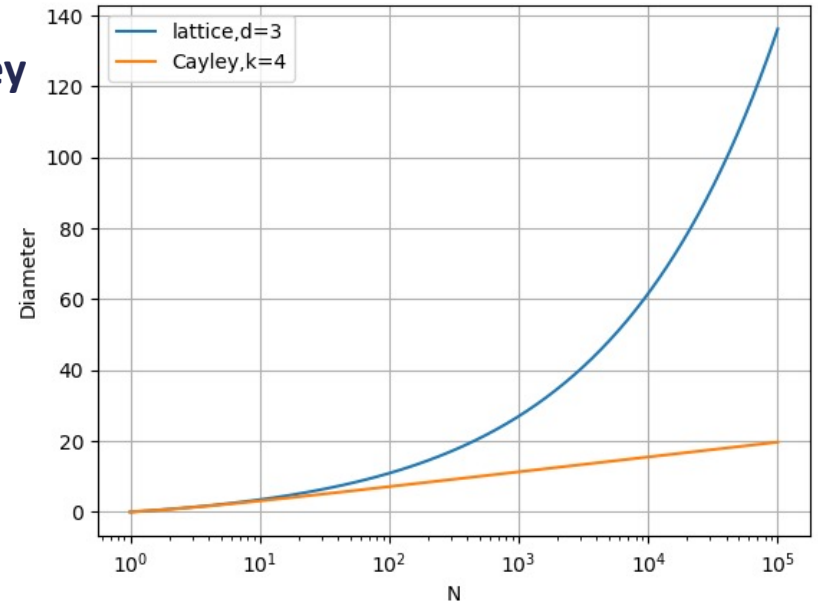
Lattices vs. Cayley trees

We have now seen that the diameter for a lattice shows power-law dependence on N, while the Cayley tree shows logarithmic dependence – is this important?

- **Yes, because logarithmic growth is "slow" →**

Questions to consider:

1. **Why have I used a logarithmic horizontal scale in the figure?**
2. **Does it make sense for a ring, 3D lattice, and Cayley tree to all have the same "clustering"?**



Centrality

- Let's now think about another aspect of graph structure: how can we identify important nodes?

- The simplest idea is the *degree centrality*: the higher the degree, the more important the node

    - The degree centrality has a few weaknesses (which we will discuss), so we will look at a few other centrality measures

- The node with highest degree in the IMDB actor network (in early 2020) is Nassar, a prolific actor in South Indian and Bollywood cinema →

Eigenvector centrality

- Consider one weakness of the degree centrality: say one node has 50 neighbors with degree=1. Is it as important as a node connected to 50 high-degree nodes?

- Instead, let's say that a node's centrality, x_i, should be proportional to the centrality of its neighbors: $x_i = \alpha \sum_{j=1}^{N} A_{ij} x_j$

- Here α is a proportionality constant which will be specified shortly

- In matrix-vector form, we have: $\mathrm{Ax} = \lambda \mathrm{x}$ with $\lambda = \alpha^{-1}$

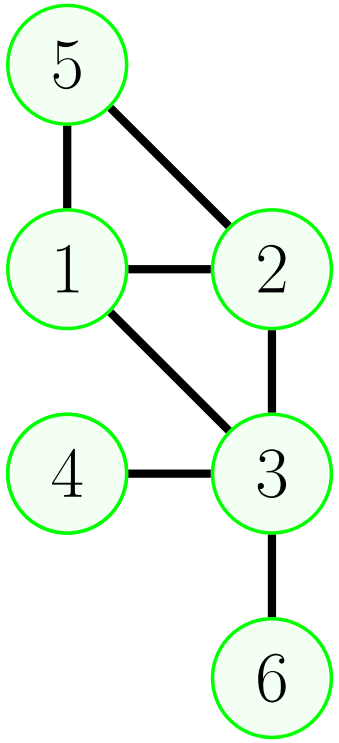
- This is of course an eigenvalue problem, and as with the linearized naïve network-SI model, the Perron-Frobenius theorem will be used for guidance
  - For an *undirected connected* graph, there will be exactly one eigenvector where all elements have the same sign, and this eigenvector corresponds to a simple positive eigenvalue of A (this eigenvalue is $\geq$ in magnitude to all other eigenvalues).

  - Scale this leading eigenvector so that all elements are positive (the magnitude of the scaling is not considered to be important)

  - Then the *eigenvector centrality* of node i is the i^{th} element of the scaled vector

- **An example:**

node	$\frac{k_i}{\|k\|}$	eigenvector centrality
1	0.47	0.51
2	0.47	0.51
3	0.63	0.51
4	0.16	0.19
5	0.32	0.38
6	0.16	0.19



- **Here, $|k| = \sqrt{\sum_{i=1}^{N} k_i^2}$ and the eigenvector centrality has also been normalized to have length=1**

- **Note that node 3 has a higher degree but the same e.c. (eigenvector centrality) as nodes 1 and 2**

- **For the IMDB actor network, the top 5 eigenvector centrality scores are for…**

```
Irving Bacon : 0.059419927711116469
Emory Parnell : 0.058154231305023625
Paul Fix : 0.05565088682410471
Russell Hicks : 0.05563688887989434
J. Farrell MacDonald : 0.05533389610888414
```

… 5 actors that I have never heard of!

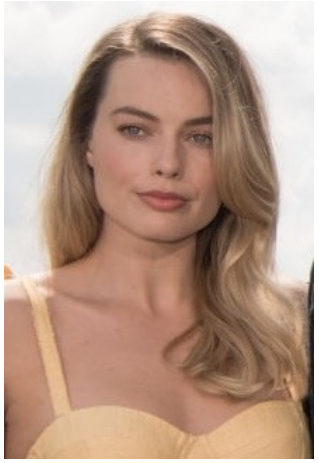
- **These are 5 American "character actors" who appeared in many films over many years in the mid-20th century (John Wayne is #10)**

Irving Bacon

- **If we instead limit the network to movies released in 2019, we see more-familiar names:**

```
Margot Robbie : 0.17784509362532366
Margaret Qualley : 0.1776352911801254
Brad Pitt : 0.17533278452858408
Clifton Collins Jr. : 0.17529419762375148
Joy Badlani : 0.17382963598017173
```

The top 4 were all in *Once upon a time in Hollywood* and we see actors benefitting from being in a large cast with actors which were in other movies with large casts.



Margot Robbie

https://commons.wikimedia.org/wiki/File:Margot_Robbie_at_%22Once_Upon_A_Time_In..._Hollywood%22_Photocall_in_Berlin_2019_(cropped).jpg

Katz centrality

- But what if a graph is directed?
- Then we need to decide between the left- and right-eigenvectors of A
- Let $A_{ij} = 1$ indicate that there is a link pointing from node j to node i
- We could then define the centrality as, $x_i = \alpha \sum_{j=1}^{N} x_j A_{ji}$
  - But typically it is the original definition that is better. This modified version rewards nodes which link to nodes that link to many other nodes. Having links pointing towards a node is usually a better indicator of importance.
- However, there is another important difficulty when considering directed graphs
  - We know that if a node has only out-links that its centrality will be zero, and this is fine
  - But a node receiving many links from nodes with zero in-links will also have zero centrality which is difficult to justify

- **The *Katz* centrality adjusts the definition of the eigenvector centrality to address this issue**
  - **The idea is to give each node a minimum centrality, and then in our example, the node with several in-links will have a higher centrality than the nodes with no such links**
  - **The Katz centrality is found from,** $x_i = \alpha \sum_{j=1}^{N} A_{ij} x_j + 1$
    - **Sometimes, the "1" is replaced by another parameter,** β**, or** $1 - \alpha$
    - **We now have to solve the linear system,** $(\mathrm{I} - \alpha \mathrm{A})\mathrm{x} = \mathrm{z}$ **where** z **is a** N**-element column vector of ones,** $\mathrm{z} = [1,1,1 \ldots, 1]^T$
    - **There is now the question of choosing** α **and we need to ensure that** $\det(\mathrm{I} - \alpha A) \neq 0$**, i.e.** α^{-1} **should not be an eigenvalue of** A
    - **We also would like to choose** α **to be as large as possible**
    - **We will again rely on the Perron-Frobenius theorem, with some modifications to the previously stated version**

- **The Perron-Frobenius theorem applied to any non-negative square matrix (all elements non-negative) tells us that there will be a real non-negative eigenvalue, λ_1, with $\lambda_1 \geq \max(|\lambda_i|)\ with\ i \in \{1, 2, \dots, N\}$**
  - $\rho(A) = \max(\{|\lambda_1|, |\lambda_2|, \dots, |\lambda_N|\})$ **is the *spectral radius* of** A
- **If $\lambda_1 > 0$, we set $\alpha^{-1} > \lambda_1$ and this will guarantee a non-trivial solution of our system.**
  - **We exclude peculiar cases where $\lambda_1 = 0$ (e.g. 2 nodes, 1 directed link)**
- **There is a useful alternative formulation of the Katz centrality based on a series expansion of $(\mathrm{I} - \alpha\mathrm{A})^{-1}$**

Network Science
Autumn 2021

Lecture 3

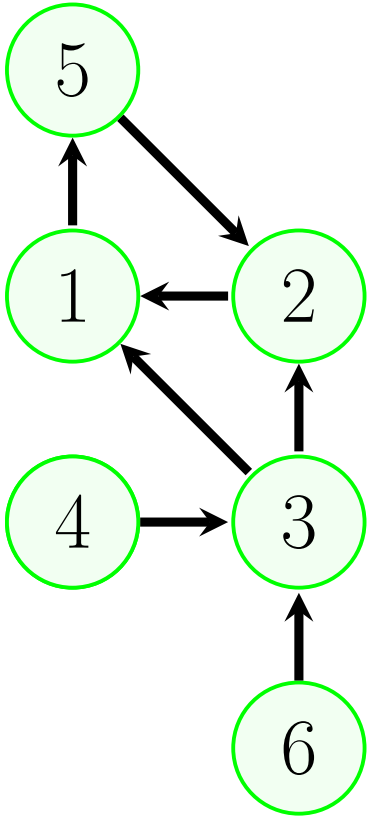
Katz and PageRank centralities
Node similarity

Katz centrality

- **Last time: the Katz centrality adjusts the definition of the eigenvector centrality to provide better behavior for directed graphs**
  - **The idea is to give each node a minimum centrality, and then nodes with several in-links will always have a higher centrality than nodes with no such links**
  - **The Katz centrality is found from,** $x_i = \alpha \sum_{j=1}^{N} A_{ij} x_j + 1$ **or equivalently, we have to find the solution to the linear system,** $(\mathrm{I} - \alpha \mathrm{A})\mathrm{x} = \mathrm{z}$ **where** z **is a** N**-element column vector of ones,** $\mathrm{z} = [1,1,1 \ldots, 1]^T$
    - **We set** $\alpha^{-1} > \lambda_1$ **and this will guarantee a non-trivial solution of our system. Here,** λ_1 **is the most positive real eigenvalue of** A

- **Another example:**

node	$\frac{k_i^{in}}{\lvert k^{in} \rvert}$	eigenvector centrality	Katz centrality ($\alpha = 0.5$)
1	0.55	0.58	0.59
2	0.55	0.58	0.54
3	0.55	0	0.32
4	0	0	0.16
5	0.28	0.58	0.45
6	0	0	0.16



- **Vectors are again normalized to have length=1**
- **Note that node 3 now has e.c.=0**
- **With the Katz centrality, all values are non-zero, and node 1 now has the highest centrality due to "help" from node 3**

- **There is a useful alternative formulation of the Katz centrality based on a series expansion of $(\mathrm{I} - \alpha \mathrm{A})^{-1}$. First we will state a few general results from linear algebra:**
  - $\mathrm{R}(\mathrm{M}; \mu) = (\mu \mathrm{I} - \mathrm{M})^{-1}$ **is the *resolvent* for a square matrix,** M
    - **The resolvent is defined when** $\mu \neq \lambda_i, i = 1,2, \dots, N$
  - **If** $|\mu| > \rho(\mathrm{M})$**, then we can expand the resolvent as,** $\mathrm{R}(\mu) = \sum_{l=0}^{\infty} \frac{\mathrm{M}^l}{\mu^{l+1}}$ **(*)**
  - **Here,** $\rho(\mathrm{M})$ **is the *spectral radius* of** M**: the magnitude of the eigenvalue(s) of** M **which is/are largest by magnitude:** $\rho(\mathrm{M}) = \max\{|\lambda_1|, |\lambda_2|, \dots, |\lambda_N|\}$
  - **Note that** $(\mu I - \mathrm{M}) \frac{\mathrm{M}^l}{\mu^{l+1}} = \frac{\mathrm{M}^l}{\mu^l} - \frac{\mathrm{M}^{l+1}}{\mu^{l+1}}$**, so multiplying both sides of (*) with** $(\mu I - \mathrm{M})$ **and truncating the series gives:**

$$\mathrm{I} = \lim_{\mathrm{T} \to \infty} (\mu I - \mathrm{M}) \sum_l^T \frac{\mathrm{M}^l}{\mu^{l+1}} = \mathrm{I} - \frac{\mathrm{M}^{T+1}}{\mu^{T+1}}$$

  - **Now, if we let** $T \to \infty$**, it can be shown that** $\frac{\mathrm{M}^{T+1}}{\mu^{T+1}} \to 0$ **if** $\mu > \rho(\mathrm{M})$**, which provides some intuition for why (*) converges**

- **Applying these results to the Katz centrality, we find,** $\mathrm{x} = \sum_{l=0}^{\infty} \alpha^l \mathrm{A}^l \mathrm{z}$ **provided that** $|\alpha^{-1}| > \rho(\mathrm{A})$.

- **We require** α **to be real and positive (otherwise, the resulting centralities will not be meaningful), so the condition above is equivalent to the condition we stated previously,** $\alpha^{-1} > \lambda_1$**.**

  - **The Perron-Frobenius theorem tells us that** $\lambda_1 = \rho(\mathrm{A})$

- **What is** A^l**? If we say** $\mathrm{B} = \mathrm{A}^l$**, then** B_{ij} **is the number of length-**l **paths from** j **to** i

- **So** x_i **is counting the number of length-**l **paths to** i**, weighted by** α^l**. For almost all real networks,** $\lambda_1 > 1$**, so we will have** $\alpha < 1$ **and the Katz centrality will place a larger weight on shorter paths.**

PageRank centrality

- **The Katz centrality is not perfect**
- **Consider the directed graph corresponding to the world-wide web**
- **We would expect** google.com **to have a very high centrality**
- **And then any website that has a link from Google would receive a large boost to its centrality. But is the behavior that we want?**
  - **Google links to many, many websites, but what if it linked to just a few?**
  - **Then those sites would deserve a higher contribution to their centrality**
  - **The idea behind PageRank centrality is to modify the Katz centrality to produce this behavior**

- **PageRank centrality can be defined as a modification to Katz centrality:**

$$x_i = \alpha \sum_{j=1}^{N} A_{ij} x_j / \max(k_j^{out}, 1) + 1$$

Where we use the max( ) **because the expression in the sum would otherwise be 0/0 when node *j* does not have any out-links**

- **However, it is usually presented a little differently,**

$$x_i = \sum_{j=1}^{N} \left(\frac{(1-m) A_{ij} x_j}{\max\left(k_j^{out}, 1\right)} + \frac{m x_j}{N} \right), 0 < m \leq 1$$

This is not identical to the previous expression, but it will have the same positive features.

- **The parameter m weights the 2 terms on the RHS. With m close to 1, there will be a tendency for all nodes to have the same centrality**

- **We can re-write our expression in matrix-vector form:** $\mathrm{Gx} = \mathrm{x}$ **where** $G_{ij} = \frac{A_{ij}(1-m)}{\max\left(k_j^{out},1\right)} + \frac{m}{N}$

- **We now need to establish a few properties of** x **to show that it will be generally useful**

  - **Specifically, we want to show that we can construct it so that all elements are non-negative and so that there is exactly one linearly independent solution for** $\mathrm{Gx} = \mathrm{x}$

- **First, let's establish a few properties of** G **for graphs where** $k_j^{out} > 0$ **for all nodes:**

  - **By inspection:** G **is *positive* (all elements are positive)**

  - **The sum of each column is one (this will help us establish that a solution to the system of equations above exists)**

- **The sum of the jth column is:**

$$\sum_{i=1}^{N} G_{ij} = \sum_{i=1}^{N} \left[\frac{A_{ij}(1-m)}{k_j^{out}} + \frac{m}{N} \right],$$

and,

$$\sum_{i=1}^{N} \left[\frac{A_{ij}(1-m)}{k_j^{out}} + \frac{m}{N} \right] = \frac{1-m}{k_j^{out}} \sum_{i=1}^{N} A_{ij} + m.$$

What is $\sum_{i=1}^{N} A_{ij}$? We know that $A_{ij} = 1$ if there is a link from j to i, so $\sum_{i=1}^{N} A_{ij}$ is the total number of links from j to other nodes. I.e. $\sum_{i=1}^{N} A_{ij} = k_j^{out}$ and:

$$\sum_{i=1}^{N} G_{ij} = 1 - m + m = 1$$

- **Since $\sum_{i=1}^{N} G_{ij} = 1$, we also know that $\lambda = 1$ is an eigenvalue of G:**
  - **Why? Consider $\mathrm{z}^{\mathrm{T}}\mathrm{G}$ where z is an N-element column vector of ones as before**
  - **The jth element of $\mathrm{z}^{\mathrm{T}}\mathrm{G}$ is the sum of the jth column of G so, $\mathrm{z}^{\mathrm{T}}\mathrm{G} = \mathrm{z}^{\mathrm{T}}$ or $\mathrm{G}^{\mathrm{T}}\mathrm{z}$=z, and $\lambda = 1$ is an eigenvalue of G^{T}**
  - **A (square) matrix and its transpose have the same characteristic polynomial, so they have the same eigenvalues: $\lambda = 1$ is an eigenvalue of G (with *left eigenvector* z)**
- **So a *right* eigenvector of G corresponding to eigenvalue $\lambda = 1$ is a solution to the PageRank equation.**
- **We need to use the version of the Perron-Frobenius theorem for positive matrices to establish that it is the only linearly independent solution and that all elements are non-negative**

- **For a positive square matrix, the Perron-Frobenius theorem tells us that:**
  - **The matrix will have a positive, real, simple eigenvalue strictly larger in magnitude than all other eigenvalues *and***
  - **All elements of the corresponding eigenvector will have the same sign**
  - **There are no other linearly independent eigenvectors where all elements have the same sign**
- **Taking all of the above together, we would like to define the leading eigenvector of G as the PageRank centrality, however does this eigenvector correspond to $\lambda = 1$?**
- **It turns out that it does; this week's problem sheet asks you to verify this statement.**

- PageRank centrality was introduced by Sergey Brin and Larry Page when thy were PhD students at Stanford in 1998 as part of their new Google search engine
- This work has proven to be… influential
- They reportedly initially used $m = 0.15$
- And they were thinking about Network Science at the time – they viewed the web as a huge directed graph, and were thinking about how to ascribe importance to web pages that they found while navigating through the graph
- Google still uses PageRank (or at least something similar) as part of its algorithm for deciding how to order search results
- But defining PageRank was just one important step – they also had to think very carefully about how to compute it for very, very large graphs

- There was no way to assemble all of the full matrices they needed on a single computer
- It was (and is) essential to use a sparse representation of the adjacency matrix so that zeros were not stored or used in additions or multiplications
- But in the end, it is a matter of crawling the web, collecting links, and constructing a graph, and then computing the leading eigenvector of the G matrix.
- Let's now discuss how to efficiently compute this eigenvector

PageRank Computation

- **For a dense matrix, the cost of computing all of the eigenvalues and eigenvectors is roughly** $O(N^3)$
  - **This is the cost to expect when using** np.linalg.eig
- **However there are more efficient methods if only 1 eigenvalue is needed**
- **The simplest of these is the Power method which I'll sketch now**

- **The basic idea of the power method is to repeatedly multiply the matrix of interest with a trial vector. With enough repetitions, the results will be dictated by the largest eigenvalue**

- **For convenience, let's assume that G is diagonalizable, so it has a "full" set of linearly independent eigenvectors.**
  - **Then a random vector $\mathrm{y} \in \mathrm{R}^N$ can be expanded as, $\mathrm{y} = c_1\mathrm{v}_1 + c_2\mathrm{v}_2 + \cdots + c_N\mathrm{v}_N$ where v_i is the i^{th} eigenvector of G, corresponding to eigenvalue λ_i**

- **Also assume that $c_1 \neq 0$, and that the eigenvalues are ordered so that , $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_N|$**

Now consider repeated multiplications of G with y:

$$\mathrm{Gy} = c_1\mathrm{Gv}_1 + c_2\mathrm{Gv}_2 + \cdots + c_N\mathrm{Gv}_N$$

- **We know that $\mathrm{Gv_i} = \lambda_i\mathrm{v}_i$, so:**

$$\mathrm{Gy} = c_1\lambda_1\mathrm{v}_1 + c_2\lambda_2\mathrm{v}_2 + \cdots + c_N\lambda_N\mathrm{v}_N$$

And: $\mathrm{G}^2\mathrm{y} = c_1(\lambda_1)^2\mathrm{v}_1 + c_2(\lambda_2)^2\mathrm{v}_2 + \cdots + c_N(\lambda_N)^2\mathrm{v}_N$

- **And after l multiplications:**

$$\mathrm{G}^l\mathrm{y} = c_1(\lambda_1)^l\mathrm{v}_1 + c_2(\lambda_2)^l\mathrm{v}_2 + \cdots + c_N(\lambda_N)^l\mathrm{v}_N$$

- **l can be chosen to be sufficiently large so that the 1st term on the RHS is much larger than all of the terms that follow, and:**

$$\mathrm{G}^l\mathrm{y} \approx c_1(\lambda_1)^l\mathrm{v}_1$$

- **The power method is not restricted to diagonalizable matrices, but some of the arguments above would have to be modified to consider more general square matrices.**

- **The key is the presence of a strictly dominant eigenvalue, an eigenvalue larger in magnitude than all other eigenvalues. The Perron-Frobenius theorem tells us that G will always have such an eigenvalue**

- **In practice, computing G^l can lead to large numerical errors, so instead a *power iteration* normalizes the results after each iteration, e.g.:**

$$\mathrm{y}^{(l+1)} = \frac{\mathrm{G}\mathrm{y}^{(l)}}{|\mathrm{G}\mathrm{y}^{(l)}|} = \frac{\mathrm{G}^{l+1}\mathrm{y}^{(0)}}{|\mathrm{G}^{l+1}\mathrm{y}^{(0)}|}$$

  and $\mathrm{y}^{(l+1)}$ is our approximation for the leading eigenvector after $l+1$ iterations of the power method

- **The power method is frequently quite effective though of course this depends on the "separation" between the 2 first eigenvalues**

- **An example:**

$$G = \begin{bmatrix} 6 & 3 \\ 3 & 2 \end{bmatrix}, y^{(0)} = [1, 0]^T, v_1 = [0.88167, 0.47186]^T$$

l	$G^l y^{(0)} / \lvert G^l y^{(0)} \rvert$	$\lvert y^{(l)} - v_1 \rvert$
1	$[0.89442719, 0.4472136]^T$	0.0277
2	$[0.88235294, 0.47058824]^T$	0.001439
3	$[0.88170982, 0.4717921]^T$	7.466e-5
4	$[0.88167643, 0.47185451]^T$	3.872e-6

- **We see very rapid convergence for this case**
- **Typically, we would see a slower rotation of the initial y toward v_1**
- **Note that the computation of the denominator is fairly expensive, and a good alternative is to use $\max(G^l y^0)$**

- **Say the eigenvector calculation requires q iterations to obtain a good estimate for the leading eigenvalue.**

- **Then the cost will in general be $O(q\ N^2)$ however for sparse networks this can be reduced to $O(q\ (L+N))$ (even though G will not be sparse).**

- **For networks with small distances, computations indicate that $q \sim \log(N)$ and the overall cost is estimated as $O((L+N)\log N)$**

Comments on centrality

- We have examined a few of the most widely-used centralities
- There are many other centrality measures out there, some of which are based on very different ideas
- We will look at one other centrality later (time permitting) based on the shortest paths in graphs

Node similarity

- **How similar are Bruce Lee and Beyoncé?**
- **To answer this question, we will introduce two measures of *node similarity***
- **There are a few different approaches to this problem, we will look at two that characterize similarity based on the number of common neighbors that two nodes share**

Bruce Lee, famous American actor and martial artist

Beyoncé, famous American actress and singer

- Since similarity shouldn't favor high-degree nodes, the number of common neighbors should be scaled by the node degrees
- Two popular approaches are the *cosine similarity* and the *Jaccard similarity*
- Let n_{ij} be the number of common neighbors of nodes i and j. Then the cosine similarity is defined as, $\sigma_{ij} = \frac{n_{ij}}{\sqrt{k_i k_j}}$ and for an undirected graph, $n_{ij} = \sum_{l=1}^{N} A_{il} A_{lj}$
- Where does the name come from? Let $\mathrm{a_i}$ be a vector that corresponds to the *ith* column of A. Then for an undirected graph, $n_{ij} = \mathrm{a_i^T a_j}$ and $k_i = \mathrm{a_i^T a_i} = |\mathrm{a_i}|^2$
- So, $\mathrm{a_i^T a_j} = |\mathrm{a_i}||\mathrm{a_j}|\sigma_{ij}$ and we see that σ_{ij} is the cosine of the angle between $\mathrm{a_i}$ and $\mathrm{a_j}$
- The Jaccard similarity uses a different scaling, it uses the total number of distinct neighbors of the two nodes: $\sigma_{ij} = \frac{n_{ij}}{k_i + k_j - n_{ij}}$
- Both of these measures have the range $0 \leq \sigma_{ij} \leq 1$

- **In lecture 1, we saw that the distance between Bruce Lee and Beyoncé in the IMDB actor network is 3, which means that (according to our two measures) they are not at all similar!**

- **Let's ask a different question: is Harrison Ford more similar to Bruce Lee or Beyoncé?**

Using the cosine similarity:

Beyonce-Harrison Ford: σ_{BH} =0.009

Bruce Lee-Harrison Ford: σ_{LH} =0.005

Using the Jaccard similarity:

Beyonce-Harrison Ford: σ_{BH} =0.0022

Bruce Lee-Harrison Ford: σ_{LH} =0.0019

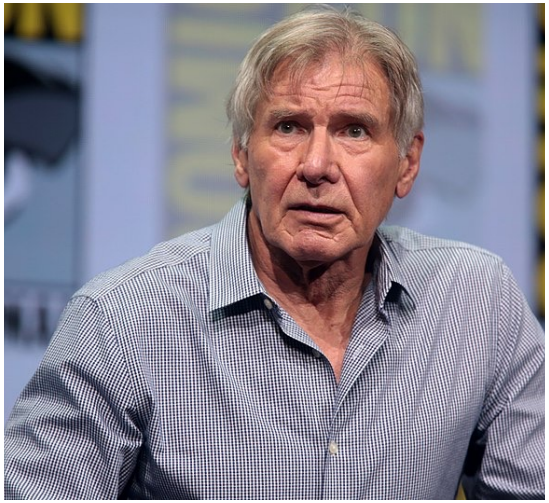


Photo credit: Gage Skidmore

So both measures tell us that Beyoncé is more similar to Harrison Ford than Bruce Lee!

- **Node similarity is a deceptively powerful concept**
  - **It is used for community detection**
  - **It is also used at the interface between network science and data science**
  - **We will discuss these applications towards the end of the module**

Perron-Frobenius theorem

- **We have applied the Perron-Frobenius (P-F) theorem to three different "classes" of real square matrices:**

1. ***Positive matrices*** **where each element of the matrix is positive (e.g. the Google matrix) Then, the theorem tells us that there is a real positive eigenvalue λ where:**
    - **$\lambda = \rho(\mathrm{A}) > 0$ and all other eigenvalues are smaller in magnitude**
    - **This eigenvalue is simple, all elements of the corresponding eigenvector have the same sign, and there are no other eigenvectors where all elements have the same sign**

2. ***Irreducible matrices:*** **Let $B_{ij} > 0$ if there is a link in a graph from node i to j with $B_{ij} = 0$ otherwise. Then B is irreducible if and only if the corresponding graph is strongly connected (i.e. every node is reachable from every other node). For irreducible matrices, there is a real, positive eigenvalue λ where:**
    - **$\lambda = \rho(\mathrm{B}) > 0$ and this eigenvalue is simple**
    - **All elements of the corresponding eigenvector have the same sign, and there are no other eigenvectors where all elements have the same sign**
    - **There may be other eigenvalues equal in magnitude to λ**

3. ***Non-negative matrices*** **where each element of the matrix is non-negative. There is a real, positive eigenvalue λ where:**

- $\lambda = \rho(\mathrm{A}) \geq 0$; **this eigenvalue is real-valued, and there may be other eigenvalues equal in value or equal in magnitude**
- **All non-zero elements of the corresponding eigenvector will have the same sign, and there may be other eigenvectors with the same property**
- **This version of the P-F theorem is considerably weaker than the other 2**

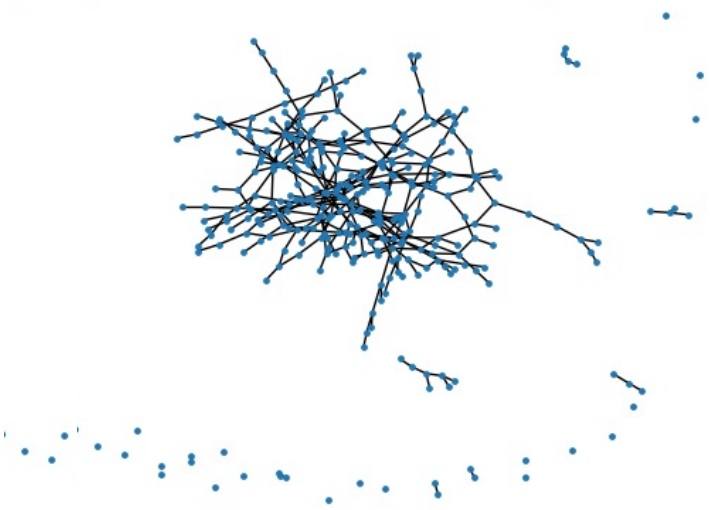
Network Science
Autumn 2021

Lecture 4

The G_{Np} random graph model:
Model definition
Key properties

Random Graph models

- We will now shift our focus from graph properties to graph models
- Specifically, we will analyze three important *random* graph models
- The aim will be to understand how rules for graph generation influence the final graph structure
- For simple deterministic models like Cayley trees and lattices, this influence is easy to understand
- If we instead introduce probabilistic rules for how links and/or nodes are added to a graph, we will see that even very simple rules can generate very complicated behavior
- Random graph models are also helpful for understanding real network data. For example, we can compare the degree distribution of a network to a model prediction



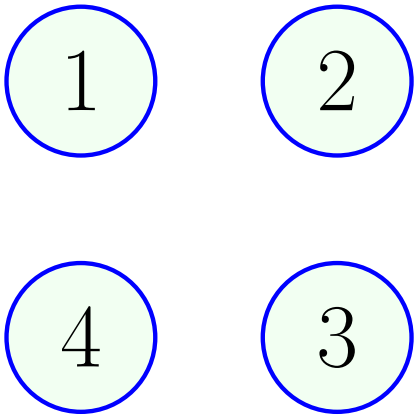
G_{NP} graph with N=400,P=0.05

G_{Np} random graph model

The first model we will analyze is the G_{Np} random graph model

- **Here, N is the number of nodes and p is the probability of a link being placed between each distinct pair of nodes**
- **G_{Np} is a model for generating graphs for a given $N$and p**
- **An individual *realization* of a G_{Np} graph can be constructed via a sequence of $N(N-1)/2$ Bernoulli trials**
  - **Each trial determines if a link is placed between one of the $N(N-1)/2$ distinct pairs of nodes in the graph**
- **We are interested in computing expectations over the set of graphs produced by the model for a given $N$and p**
- **Note that the model generates *simple graphs*: undirected graphs without self-loops or multiedges (this definition is a little different from Barabasi but more useful)**

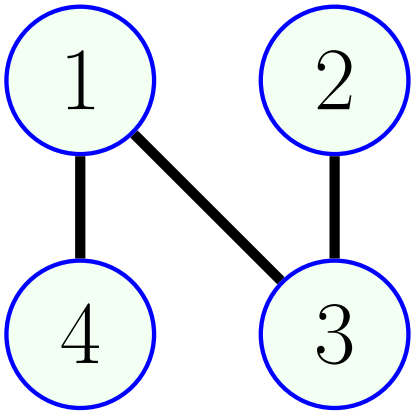
- **Let's "simulate" a small graph using the model with $N = 4, p = 0.3$**
  - **We start with the four nodes and generate 6 uniformly distributed random numbers between 0 and 1:**



In [8]: np.random.random(6)
Out[8]:
array([0.44646233, 0.10855423, 0.19685276, 0.29337877, 0.66279838,0.76902445])

- **Assign the random numbers to the 6 node pairs:** $(1, 2); (1, 3); (1, 4); (2, 2); (2, 3); (2, 4); (3, 4)$

- **Let's "simulate" a small graph using the model with $N = 4, p = 0.3$**
  - **We start with the four nodes and generate 6 uniformly distributed random numbers between 0 and 1:**



In [8]: np.random.random(6)
Out[8]:
array([0.44646233, 0.10855423, 0.19685276, 0.29337877, 0.66279838,0.76902445])

- **Assign the random numbers to the 6 node pairs:** $(1, 2); (1, 3); (1, 4); (2, 3); (2, 4); (3, 4)$
- **And finally add a link between a pair if the corresponding random number is less than p**

Notes: np.random.choice **would be more convenient than** np.random.random

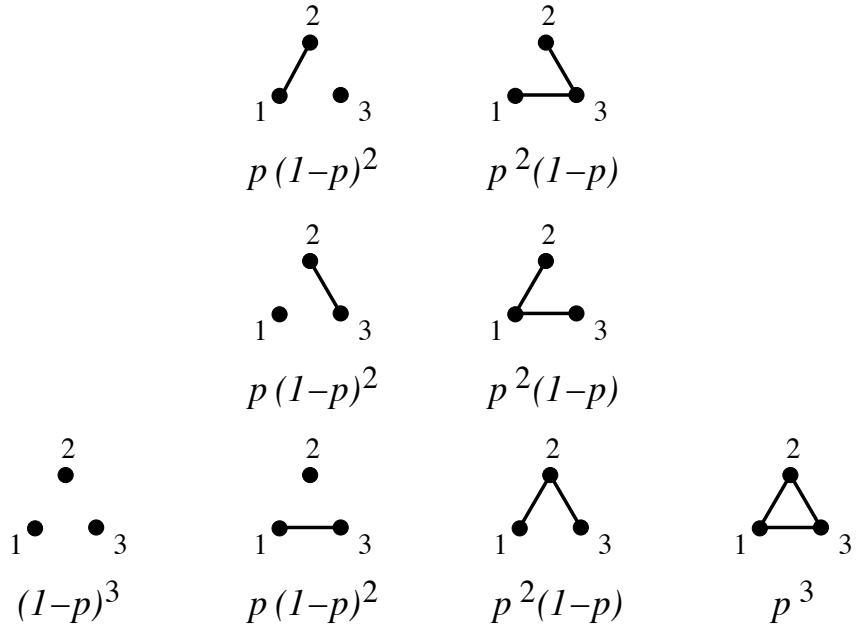
- **You have already seen how to create these graphs using NetworkX**

Check your understanding: What is the probability of generating the graph shown on the previous slide?

Check your understanding: What is the probability of generating the graph shown on the previous slide?

- The graph was generated with 6 Bernoulli trials with probability of success, p. Three of these were successful. So, the probability of generating the graph is, $p^3(1-p)^3$
- And there are of course several other graphs that could have been generated
- Let's now look more closely at this idea of an ensemble of graph realizations

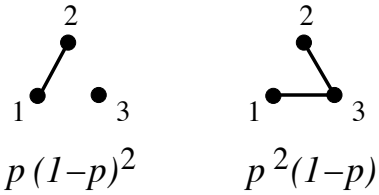
- **The figure on the right shows the sample space for G_{Np} with $N = 3$→**
- **The "realization probabilities" for each graph are shown (note that they sum to 1)**
- **More generally, for a given N, G_{Np} provides a probability measure that assigns a probability for each graph in a sample space, $G \in \Omega_N$:**
  $P(G) = p^L(1-p)^{N'-L}$
  - **Here, Ω_N is the set of all N-node graphs**
  - **G is an L-link realization generated by G_{Np}**
  - **And $N' = \binom{N}{2}$ is the maximum possible number of links in the graph**



- **We can use this probability measure to deduce other statistical properties of the model (for a given N)**

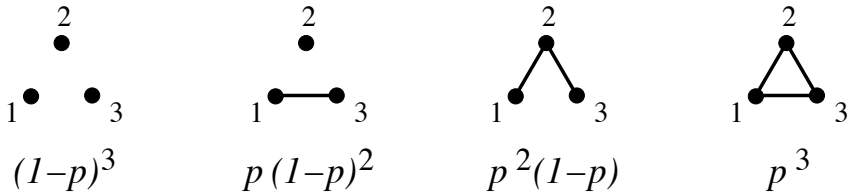
- **Set $N = 3$, and consider the degree of node 1, k_1**
  - **k_1 is a random number, what is its probability distribution?**

The probability of a node, i, having a particular degree, d, is the sum of the realization probabilities of graphs where $k_i = d$.



- **For example:**

$$P(k_1 = 0) = \sum_{G,k_1=0} P(G)$$



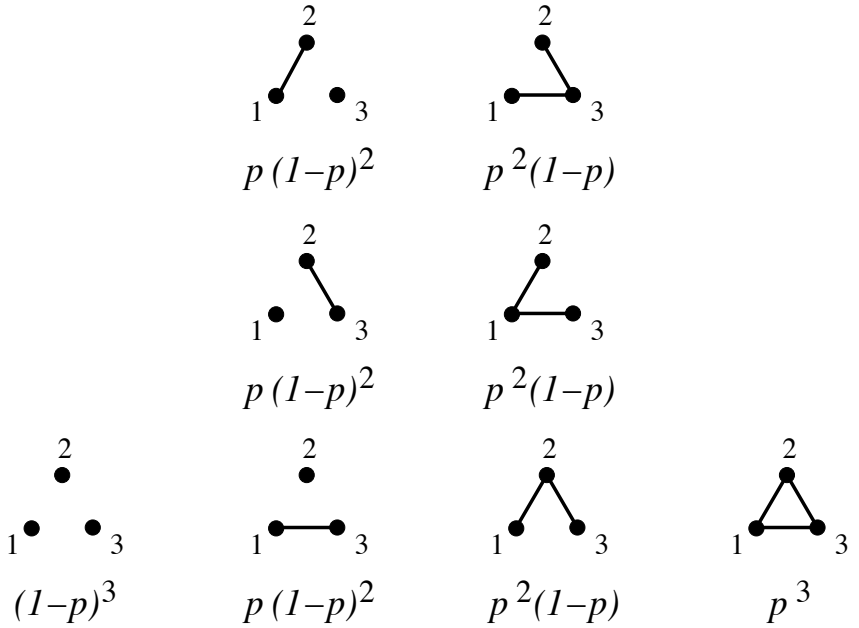
where the sum is over the two graphs shown where $k_1 = 0$:

$$P(k_1 = 0) = (1-p)^3 + p(1-p)^2 = (1-p)^2$$

- We can also compute the expected degree using this perspective.
- Let $\langle k_i \rangle$ be the expected degree of node i.

  Then, $\langle k_i \rangle = \sum_{G \in \Omega_N} P(G) k_i(G)$

- Here, the sum is over the entire sample space for a given N, and $k_i(G)$ is the degree of node i in realization G
- For our example with $N = 3$, the sum will be over 8 graphs.
  - The realization probability for each graph is shown, and we can also easily see what $k_i(G)$ is for each graph



- We could also have computed the expectation in the more conventional way using the probability distribution for the degree: $\langle k_i \rangle = \sum_{k=0}^{N-1} P(k_i = k) k$

- **Explicitly considering the ensemble of graph realizations is helpful for building an understanding of the G_{Np} model and can help us reach other useful conclusions about the model.**
  - **For example, the symmetry of the sample space tells us that the probability distribution and expectation of each node will be the same, e.g. $\langle k_1 \rangle = \langle k_2 \rangle = \langle k_3 \rangle$ when $N = 3$.**

- **However, this is not the best approach for large graphs as we then have to think about the realization probabilities of a large number of graphs (the size of the sample space is $2^{N'}$)**

- **So, we need to find a way to generalize the approach we have just discussed. Fortunately, this is not too difficult!**

Link and degree distributions

- **We'll now find the *link distribution,* p_L, for the G_{Np} model. This is the probability that a graph generated by the model has exactly L links.**

- **The probability that a G_{Np} graph has exactly L links is:**

(# of distinct configurations with L links)*(probability of generating one such configuration)

- **There are $\binom{N'}{L}$ distinct ways to place L links in an N-node graph (with a maximum of 1 link per pair of nodes). Recall that $N' = \binom{N}{2}$ which is the maximum number of links that a G_{Np} graph can have.**

- **The probability of creating a particular sequence of L links is, $p^L(1-p)^{(N'-L)}$**

- **Putting it all together gives the binomial distribution: $p_L = \binom{N'}{L} p^L(1-p)^{(N'-L)}$**

- **We could have just stated that the probability of L successes from N' Bernoulli trials is given by the binomial distribution**

- What about the expected number of links, $\langle L \rangle$? This can be computed from:

$$\langle L \rangle = \sum_{L=0}^{N'} L \, p_L$$

and the sum can be evaluated using the binomial expansion.

-
- However, we'll use a different approach to computing the expectation which uses an *indicator random variable*. This will be useful for other problems we will encounter later.

- The adjacency matrix is now a random matrix – its elements are random variables. A_{ij} will be 0 or 1 depending on the outcome of a Bernoulli trial. So, A_{ij} *indicates* if a link is placed between nodes i and j.

- The total number of links in a graph is, $L = \sum_{j=1}^{N-1} \sum_{i=j+1}^{N} A_{ij}$
  (the double summation is over each distinct node pair)

- Then, using linearity of expectation, we have, $\langle L \rangle = \sum_{j=1}^{N-1} \sum_{i=j+1}^{N} \langle A_{ij} \rangle$

- We now need to determine $\langle A_{ij} \rangle$

- **First, use the standard definition of the expectation:**

$$\langle A_{ij} \rangle = P(A_{ij} = 1) * (1) + P(A_{ij} = 0) * (0) = P(A_{ij} = 1)$$

This is the real advantage of an indicator random variable: its expectation is equal to the probability that it is 1

- **For the** G_{Np} **model, we know that** $P(A_{ij} = 1) = p$**, so:**

$$\langle L \rangle = \sum_{j=1}^{N-1} \sum_{i=j+1}^{N} \langle A_{ij} \rangle = \sum_{j=1}^{N-1} \sum_{i=j+1}^{N} p = N'p$$

- **This is an intuitive result: the expected number of successes from** N' **Bernoulli trials is** $N'p$**.**

- **The degree distribution can be constructed in a similar manner.**
- **The degree of a node is determined by the outcomes of $N-1$ Bernoulli trials**
- **Let p_k be the probability that a node has degree k. Then using the same reasoning used for the link distribution, we find that, $p_k = \binom{N-1}{k} p^k (1-p)^{(N-1-k)}$**
- **And $\langle k \rangle = (N-1)p$**

Node average

- One further point to consider is if there is any relationship between the average over all nodes in a single graph and the expectation, e.g. can we say something like:

$$\overline{k} \approx \langle k \rangle \text{ where, } \overline{k} = \frac{1}{N}\sum_{i=1}^{N} k_i$$

- Approximations like this are often implicit in network science – e.g. the fraction of nodes with a given degree in an individual graph is assumed to be comparable to a probability of a node having that degree

- Heuristic argument: The node average is an average of N realizations of random variables equal in distribution. As N becomes large, a large portion of the sample space is included in the node average and we can anticipate convergence to the expectation.

- A (somewhat) more rigorous argument relies on the law of large numbers

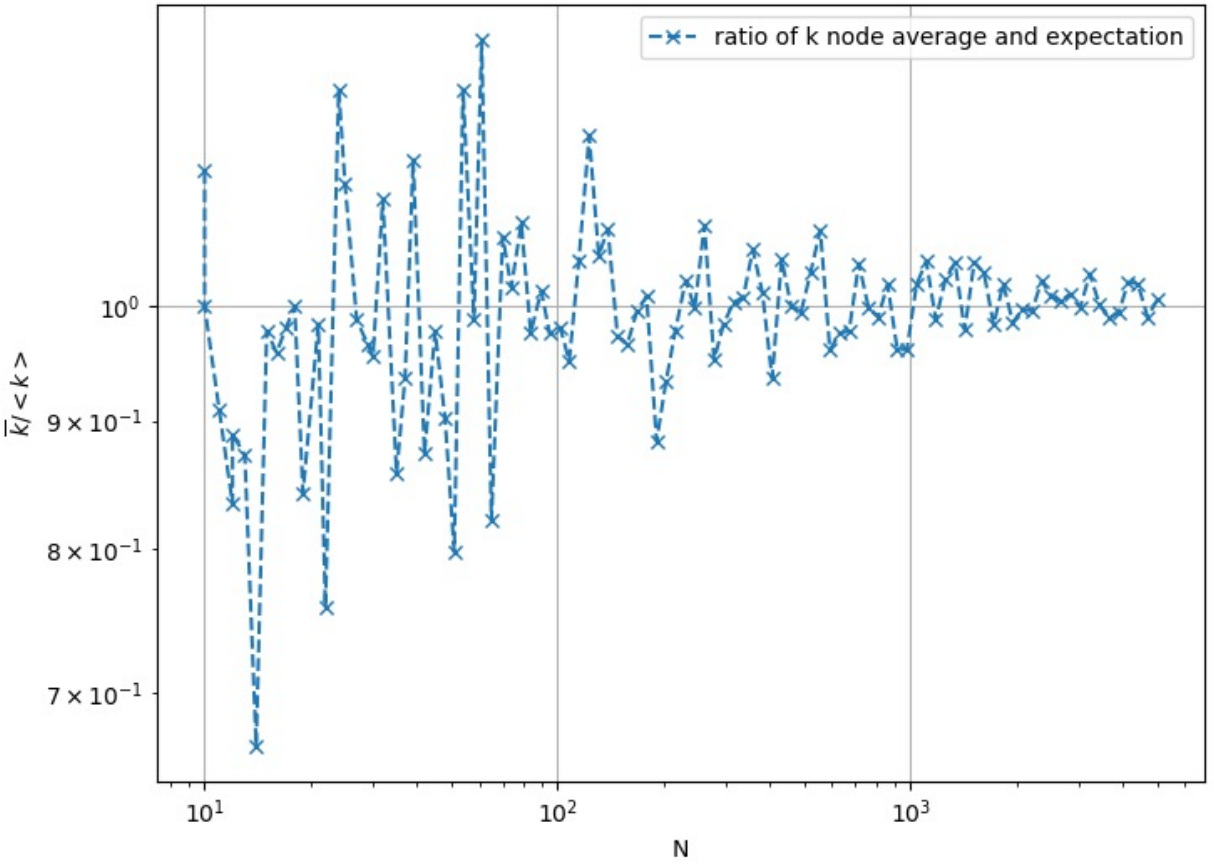
***Law of large numbers*: Let $X_1, X_2, \ldots, X_N$ be independent random variables, with finite expected value $\mu = \langle X_i \rangle$ and finite variance $\sigma^2 = var(X_i)$. Then for any positive ϵ:**

$$P\left(\left|\frac{(X_1+X_2+\cdots+X_N)}{N} - \mu\right| \geq \epsilon\right) \to 0 \; as \; N \to \infty.$$

- **However, there is one problem. The degrees of the the different nodes are not statistically independent. To see this, consider the "extreme" case where $N = 2$**
- **Fortunately, we can show that the degrees of two nodes become independent as $N \to \infty$ which leads to the result, $P\left(\left|\frac{(k_1+k_2+\cdots+k_N)}{N} - \langle k \rangle\right| \geq \epsilon\right) \to 0 \; as \; N \to \infty$ for any positive ϵ**
- **Expressions like this are often stated as: $\frac{1}{N}\sum_{i=1}^{N} k_i = \langle k \rangle \; w.h.p.$**
  - **Here "$w.h.p.$" is shorthand for "with high probability" and is used to describe the behavior of random variables when the problem size $\to \infty$**

- I haven't given you a complete proof on the previous slide, so you may be skeptical about the result!
- So let's compare the node average to the expectation using simulations of random graphs with varying N
- Experiment: Vary N with the expected degree held fixed to $\langle k \rangle = 3$
- Hypothesis: $\overline{k}/\langle k \rangle$ will approach one as N increases

- **Result: the average degree "approaches" the expected value as N increases**
- **For a given N, a computed node average will depend on the graph realization**
- **So what we see is that at larger N, there is a higher probability that the node average of a realization will be close to the expected value**



Comments

- We will continue working with the G_{Np} model next lecture
- Though the model may appear to be simple at first, it can produce very complicated behavior, and we will examine a few examples of such behavior

Notation

- N: **total number of nodes in graph**
- L: **total number of links in graph**
- N': **maximum possible number of links in a simple graph with** N **nodes;** $N' = \frac{N(N-1)}{2}$
- p_L: **probability of exactly** L **links in the graph**
- $\langle L \rangle$: **expected number of links;** $\langle L \rangle = \sum_{L=0}^{N'} L\, p_L$
- p_k: **probability of a node having** k **links**
- $\langle k \rangle$: **expected degree of node;** $\langle k \rangle = \sum_{k=0}^{N-1} k\, p_k$
- $\overline{f}$: **node average of** f**;** $\overline{f} = \frac{1}{N}\sum_{i=1}^{N} f_i$

Network Science
Autumn 2021

Lecture 5

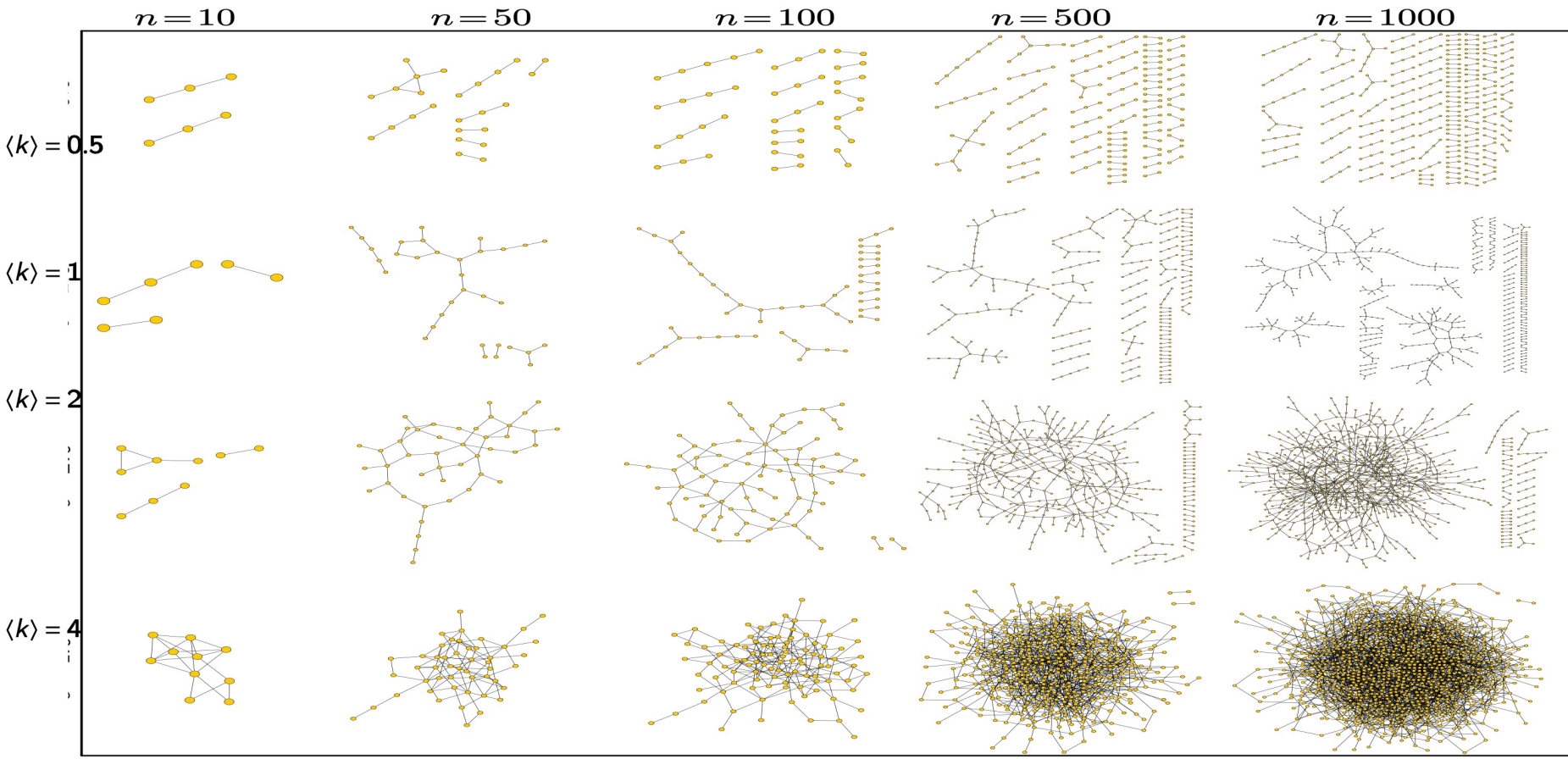
The G_{Np} random graph model: connectivity and structure

G_{Np} random graphs

- We have analyzed a few properties of the G_{Np} random graph model, and now we will examine what happens when we allow N and p to vary.
  - We are particularly interested in what happens when N becomes large

- The figure on the next slide shows several graphs for different N and $\langle k \rangle$.

  - Can we understand what is happening? It will be natural to focus on the connectivity of the graphs. E.g. when can we expect graphs generated by the model to be connected?



$$\langle k \rangle = (n-1)p$$

See also: video 3.2 in Barabasi

Triangles in G_{Np} graphs

- **Let's start with a "structural" question: *what is the expected number of triangles in G_{Np} graphs?***
  - **To answer this question, we will use an indicator (random) variable, X_{ijk}, which will allow us to count the number of triangles in these graphs**
  - **Let $X_{ijk} = \begin{cases} 1 \; if \; i,j,k \; form \; a \; triangle \\ 0, otherwise \end{cases}$ and we require $i, j,$ and k to be distinct.**

Check your understanding: $P(X_{ijk} = 1) = ?$

- **Let's start with a "structural" question: *what is the expected number of triangles in G_{Np} graphs?***
  - **To answer this question, we will use an indicator (random) variable, X_{ijk}, which will allow us to count the number of triangles in these graphs**
  - **Let $X_{ijk} = \begin{cases} 1 \; if \; i,j,k \; form \; a \; triangle \\ 0, otherwise \end{cases}$ and we require $i, j,$ and k to be distinct.**

Check your understanding: $P(X_{ijk} = 1) = ?$

  - **$P(X_{ijk} = 1)$ is the probability that three Bernoulli trials are all successful, so $P(X_{ijk} = 1) = p^3$.**
  - **We also know that $\langle X_{ijk} \rangle = P(X_{ijk} = 1)$**

- **Now, we simply need to count the number of triangles in the graph. Let t_{Np} be the total number of triangles. Then,**

  $t_{Np} = \sum_{distinct\ i,j,k} X_{ijk}$, **and using linearity of expectation,**

  $\langle t_{Np} \rangle = \sum_{distinct\ i,j,k} \langle X_{ijk} \rangle = \sum_{distinct\ i,j,k} p^3$

- **The sum is over each distinct "triple" of nodes in the N-node graphs, and we know there are $\binom{N}{3}$ such triples. So, the expected number of triangles is simply,**

  $\langle t_{Np} \rangle = \binom{N}{3} p^3$

- **Now, let's consider what happens when $N \to \infty$**
  - **If p is held fixed, then $\langle t_{Np} \rangle \to \infty$. Not particularly interesting!**
  - **What if we allow p to vary with N, so we have $p(N)$ with $p(N) \to 0$ when $N \to \infty$?**
  - **Then, the *rate* at which p approaches zero determines whether of not we expect triangles in infinitely large graphs**

- **We will now prove the following theorem:**

Let $\alpha(N)$ ***be a real-valued function such than*** $\alpha \to 0$ ***as*** $N \to \infty$***, and let*** $p(N) = \frac{\alpha(N)}{N}$. ***Then*** $t_{Np} = 0\ w.h.p.$

- **So we need to show that** $P\left(t_{Np} \geq 1\right) \to 0$ **as** $N \to \infty$

- **We will use *Markov's inequality*:**

If X ***is a non-negative random variable and*** $a > 0$***, then*** $P(X \geq a) \leq \frac{\langle X \rangle}{a}$

- **Setting** $a = 1$ **is particularly useful for discrete random variables:** $P(X \geq 1) \leq \langle X \rangle$

- **If we can show that** $\langle t_{Np} \rangle \to 0$ **as** $N \to \infty$**, we can then use this form of Markov's inequality to prove our theorem**

- **We have already shown that** $\langle t_{Np} \rangle = \binom{N}{3} p^3$ **and now, we also have** $p(N) = \frac{\alpha(N)}{N}$
- **It follows that,** $\lim_{N\to\infty} \langle t_{Np} \rangle = \lim_{N\to\infty} \frac{(N)(N-1)(N-2)\alpha^3}{6N^3} = 0$
- **Then applying Markov's inequality with** $a = 1$ **tells us that,**
  $P(t_{Np} \geq 1) \to 0$ **as** $N \to \infty$ **completing the proof.**
- **This result indicates that we can say that graphs will be "locally tree-like"** $w.h.p.$

Connectedness with fixed p

- **Generally, the G_{Np} model is not particularly interesting when $N \to \infty$ with p held fixed**
- **We can gain a sense of why from the following result: G_{Np} graphs have diameter $D \leq 2$ *w.h.p.* if p is held fixed as $N \to \infty$**
  - **This tells us that we will have densely connected graphs where a maximum of two links separates any two nodes**
- **Let's now show that this statement is correct. A graph has diameter $D \leq 2$ if every pair of nodes has at least one common neighbor**
- **We will use an indicator variable to count the pairs of nodes that *do not* have common neighbors**

- **Let** $X_{ij} = \begin{cases} 1, if\ distinct\ nodes\ i\ and\ j\ do\ not\ have\ a\ common\ neighbor \\ 0, otherwise \end{cases}$

- **Now, our goal is to compute the sum of X_{ij} over all node pairs and show that the expectation of this sum goes to zero when $N \to \infty$. Then, Markov's inequality can be used to complete the derivation.**

The sum: $X = \sum_{j=1}^{N-1} \sum_{i=j+1}^{N} X_{ij}$

Its expectation: $\langle X \rangle = \sum_{j=1}^{N-1} \sum_{i=j+1}^{N} \langle X_{ij} \rangle$

And we know that $\langle X_{ij} \rangle = P(X_{ij} = 1)$

- **What is $P(X_{ij} = 1)$? First consider the probability that nodes i and j both link to some third node. This probability is p^2. So the probability that this node is *not* a common neighbor for i and j is $(1 - p^2)$. There are $N - 2$ such "third nodes" to consider, so $P(X_{ij} = 1) = (1 - p^2)^{N-2}$, and $\langle X \rangle = \binom{N}{2} (1 - p^2)^{N-2}$**

- **We now need to determine** $\lim_{N\to\infty} \binom{N}{2} (1-p^2)^{N-2}$

We have a product of two terms, the first increasing quadratically, and the second decreasing exponentially (assuming $0 < p < 1$**), so the limit will be zero.**

Or more precisely, we know $(1-p^2)^{N-2} = \exp[a(N-2)]$ **where** $a = \log(1-p^2)$ **is negative. Then applying l'Hopital's rule shows that the limit is zero.**

- **We have shown that** $\langle X \rangle \to 0$**, and Markov's inequality tells us that,** $P(X \geq 1) \leq \langle X \rangle$**, so** $P(X \geq 1) \to 0$**, and we can say that the number of node pairs with no common neighbors is zero** $w.h.p.$

- **And this tells us that large graphs with fixed** p **will tend to have a single densely-connected component. However, this changes if we allow** $p \to 0$.

Connectedness with $p \to 0$

- We need to specify the rate at which $p \to 0$, and we will focus on one special case: $p = c\frac{log(N)}{N}$ with c a positive constant which must be specified.
- Why this case? It has been found that there is a transition from a disconnected state to a connected state when c is increased above one, and we will (partially) analyze this transition
- We will now prove the following:

Assume $p = c\frac{log(N)}{N}$ ***with*** $c < 1$***. Then*** G_{Np} ***graphs are not connected*** $w.h.p.$

- We will actually prove a stronger result, that graphs will have isolated nodes $w.h.p.$

- **We again introduce an indicator variable:**

$$X_i = \begin{cases} 1 \; if \; i \; is \; an \; isolated \; node \\ 0, otherwise \end{cases}$$

and we will show that $P(X = 0) \to 0$ **as** $N \to \infty$ **where** $X = \sum_{i=1}^{N} X_i$ **is the total number of isolated nodes in a graph.**

- **We will first examine the expected value of** X **and we will see that we will need to 1) use *Chebyshev's* inequality and 2) compute** $\langle X^2 \rangle$**.**

- **We now are familiar with the steps required to compute** $\langle X \rangle$**. A node is isolated if** $N - 1$ **Bernoulli trials are unsuccessful, so,** $P(X_i = 1) = (1 - p)^{N-1} = \langle X_i \rangle$

- **Then,** $\langle X \rangle = N\langle X_i \rangle = N(1 - p)^{N-1}$**. What happens as** $N \to \infty$**?**
  - $(1 - p)^{N-1} = \exp[a(N - 1)]$ **where** $a = \log(1 - p)$ **which for small** p **we can approximate as,** $a \approx -p = -c \log N / N$**.**
  - **So,** $(1 - p)^{N-1} \approx \exp\left[-c \log(N) \frac{N-1}{N}\right] \approx \exp[-c \log(N)] = N^{-c}$**, and:**

$\langle X \rangle \approx N^{1-c}$ **so the expectation** $\to \infty$ **if** $c < 1$.

- **But how can we construct a result related to $P(X = 0)$?**

- **Here, we need to use Chebyshev's inequality:**

Let X be a random variable and let $\epsilon > 0$ be any positive real number. Then,

$$P(|\langle X\rangle - x| \geq \epsilon) \leq Var(X)/\epsilon^2.$$

- **$Var(X)$ is the variance of X, and Chebyshev's inequality can be viewed as a corollary of Markov's inequality.**

- **We now need to manipulate the inequality into a form that is useful for our problem. We choose $\epsilon = \langle X\rangle$, and note that $P((\langle X\rangle - X) \geq \epsilon) \leq P(|\langle X\rangle - X| \geq \epsilon)$. Chebyshev's inequality can then be restated as, $P((\langle X\rangle - X) \geq \langle X\rangle) \leq Var(X)/\langle X\rangle^2$, or more simply:**

$$P(X \leq 0) \leq Var(X)/\langle X\rangle^2,$$

and now we need to evaluate, $Var(X)/\langle X\rangle^2$ when $N \to \infty$.

- **The variance is defined as,** $Var(X) = \langle X^2 \rangle - \langle X \rangle^2$**, and:**

$\langle X^2 \rangle = \sum_{j=1}^{N} \sum_{i=1}^{N} \langle X_i X_j \rangle$. **The summation can be rearranged as,**

$\langle X^2 \rangle = \sum_{i=1}^{N} \langle X_i^2 \rangle + 2 \sum_{j=1}^{N-1} \sum_{i=j+1}^{N} \langle X_i X_j \rangle$.

- **Now,** $\langle X_i^2 \rangle = P(X_i^2 = 1) = P(X_i = 1) = \langle X_i \rangle$**, so,**

$\langle X^2 \rangle = \langle X \rangle + 2 \sum_{j=1}^{N-1} \sum_{i=j+1}^{N} \langle X_i X_j \rangle$

- **The last major step is to derive an expression for** $\langle X_i X_j \rangle$**. From the definition of the expectation,** $\langle X_i X_j \rangle = P(X_i = 1, X_j = 1)$**, and then using the rule for conditional probability,** $\langle X_i X_j \rangle = P(X_j = 1 \,|\, X_i = 1) P(X_i = 1)$.

- **We have already stated that** $P(X_i = 1) = (1-p)^{N-1}$ **and using the same reasoning but accounting for** i **being isolated ,** $P(X_j = 1 \,|\, X_i = 1) = (1-p)^{N-2}$

- **So,** $\langle X^2 \rangle = \langle X \rangle + N(N-1)(1-p)^{2N-3}$

Let's collect our results:

- $\langle X \rangle = N(1-p)^{N-1}$
- $\langle X^2 \rangle = \langle X \rangle + N(N-1)(1-p)^{2N-3}$
- $\frac{Var(X)}{\langle X \rangle^2} = \frac{\langle X^2 \rangle}{\langle X \rangle^2} - 1 = \frac{N(1-p)^{N-1} + N(N-1)(1-p)^{2N-3}}{N^2(1-p)^{2N-2}} - 1$

And simplifying the last expression,

$$\frac{Var(X)}{\langle X \rangle^2} = \frac{1}{N(1-p)^{N-1}} + \frac{1-\frac{1}{N}}{1-p} - 1$$

Now letting $N \to \infty$: $\lim_{N\to\infty} \frac{Var(X)}{\langle X \rangle^2} = \lim_{N\to\infty} \frac{1}{N(1-p)^{N-1}}$ **since both** $\frac{1}{N}$ **and** p **go to zero.**

Earlier we showed that for large N, $(1-p)^{N-1} \approx N^{-c}$, **so**
$\lim_{N\to\infty} \frac{Var(X)}{\langle X \rangle^2} \approx \lim_{N\to\infty} N^{c-1} = 0$ **if** $c < 1$. **The modified form of Chebyshev's inequality gives,** $P(X \leq 0) \to 0$, **which implies isolated nodes are present** $w.h.p.$ ■

- **There is a complimentary result for $c > 1$:**

Assume $p = c\frac{log(N)}{N}$ ***with*** $c > 1$***. Then*** G_{Np} ***graphs are connected*** $w.h.p.$

- **We will not prove this, it requires thinking about all subsets of nodes in an N-node graph**

- **These results show that a slight change in the rate at which p goes to zero has a fundamentally important effect on the graph structure**

Giant component

- We have argued that if $p \to 0$ faster than $\frac{1}{N}$, then graphs will have no triangles $w.h.p.$ and we expect a (local) tree-like structure

- We have stated that if $p \to 0$ slower than $\frac{\log N}{N}$, then graphs will be connected $w.h.p.$

- What happens "in between" these states when $p \sim \frac{1}{N}$?

- What we find is that a "giant component" emerges if $p = \frac{c}{N}$, with c chosen such that $\langle k \rangle > 1$
  - A connected component is a giant component if its size increases linearly with N. A connected component is a subset of a graph where there is at least one path between each pair of nodes in the subset, and there are no links to nodes outside of the subset.

  - This is important. In many large real-world networks there is a large connected component which contains a tangible fraction of the total number of nodes in the network and whose connectivity is important for the functionality of the network.

- **Say that v is the probability that a randomly selected node is in the giant component. Then $u = 1 - v$ is the probability a node is not in this component.**

- **But there is another way to view these probabilities.**

- **A randomly selected node, i, will not belong to the giant component if for every other node in the graph it either: 1) does not link to that node or 2) does link to it, but the neighbor is not in the giant component**
  - **The probability of not linking to another node is $(1 - p)$ and the probability of linking to a node not in the giant component is pu. And there are $(N - 1)$ such nodes to consider**
  - **So the probability i is not in the giant component is $[(1 - p) + pu]^{N-1}$**

- **Consistency requires: $u = [(1 - p) + pu]^{N-1}$**

- **So we need to consider solutions to,** $u = [(1-p) + pu]^{N-1}$ **or rewriting the equation in terms of** v:

$$v - 1 = -[1 - pv]^{N-1}$$

- **By inspection, we can see that** $v = 0$ **is a solution indicating that there is not a giant component. Are there solutions with** $v > 0$?

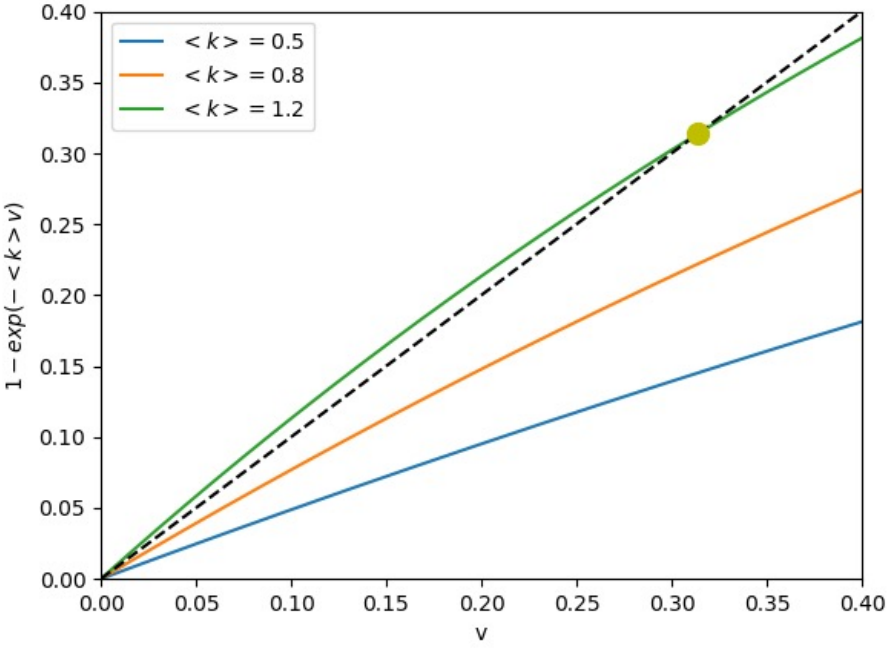
- **We will need a computer to answer this, but first we take the log of both sides of the equation and use** $p = \frac{c}{N}$ **:**

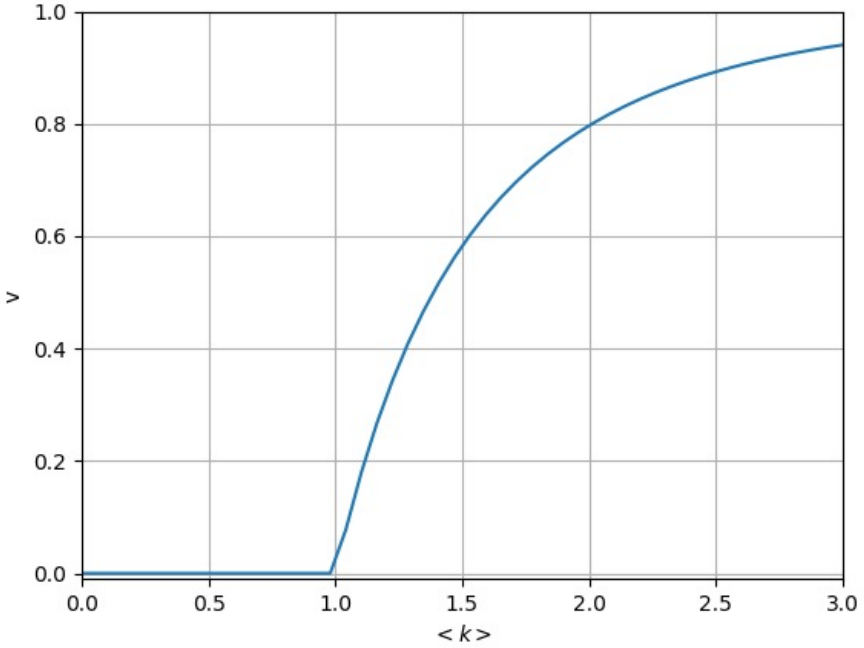
$$\log(v-1) = -(N-1)\log(1 - \frac{cv}{N})$$

- **For large** N, $\log\left(1 - \frac{cv}{N}\right) \approx -cv/N$, **and** $\frac{cv(N-1)}{N} = \langle k \rangle v$, **so:**

$$v \approx 1 - \exp(-\langle k \rangle v)$$ **(for large** N**)**

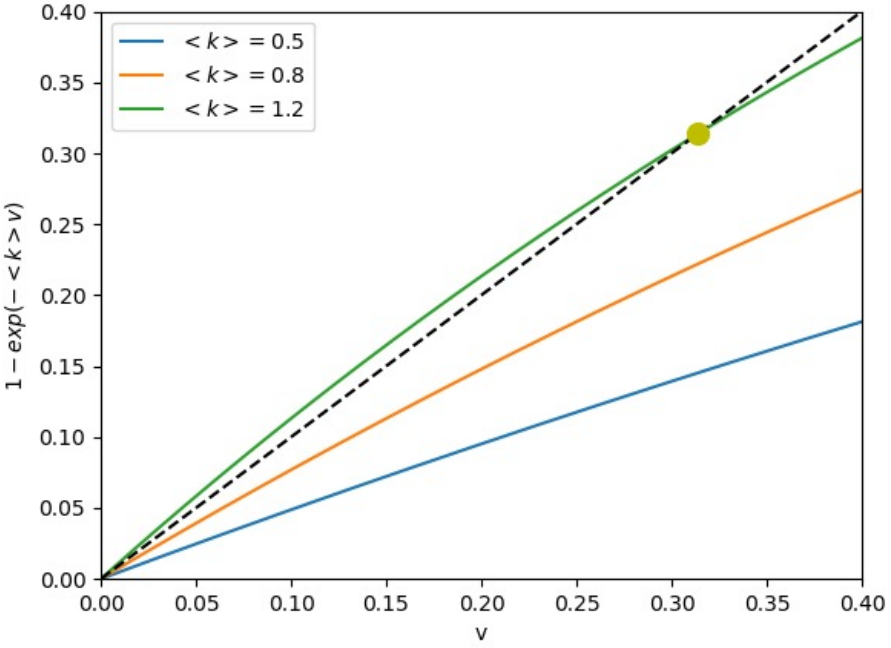
- **Can we choose** $\langle k \rangle$**so that there are non-zero solutions? We will "explore" this question by choosing a few values for** $\langle k \rangle$ **and plotting** $1 - \exp(-\langle k \rangle v)$ **vs.** v **and examining the results**

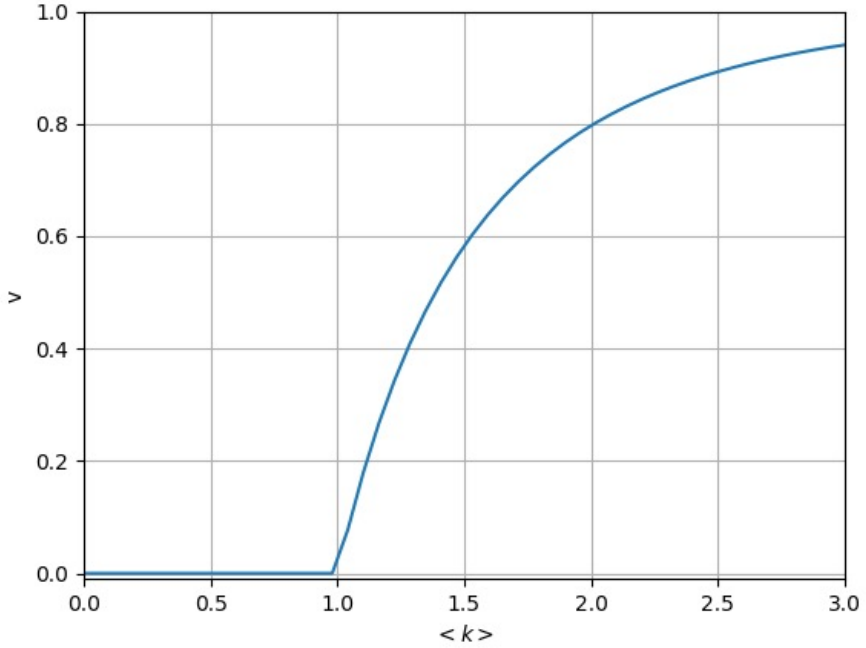




- **The figure on the left shows $1 - \exp(-\langle k \rangle v)$ vs. v for $\langle k \rangle = 0.5, 0.8, and\ 1.2$. The dashed curve is simply v vs. v and any intersection between a solid curve and the dashed curve is a solution to our equation.**

- **All three solid curves intersect the dashed curve when $v = 0$. Only the $\langle k \rangle = 1.2$ curve has an intersection for non-zero $\langle k \rangle$ which occurs at $v = 0.31$. Non-zero solutions are found only when $\langle k \rangle > 1$**





- **The figure on the right shows the (largest) solution with $\langle k \rangle$ varying. There is initially a rapid increase in $\langle k \rangle$ for $\langle k \rangle > 1$, and the figure indicates that the giant component will contain more than 90% of the graph's nodes when $\langle k \rangle > 3$**

- **Technically, we have only shown that $\langle k \rangle > 1$ is *necessary* for the existence of a giant component. With further work, it can be shown that this is also a sufficient condition, and that a unique giant component will exist if and only if $\langle k \rangle > 1$ and N is sufficiently large**

Announcements

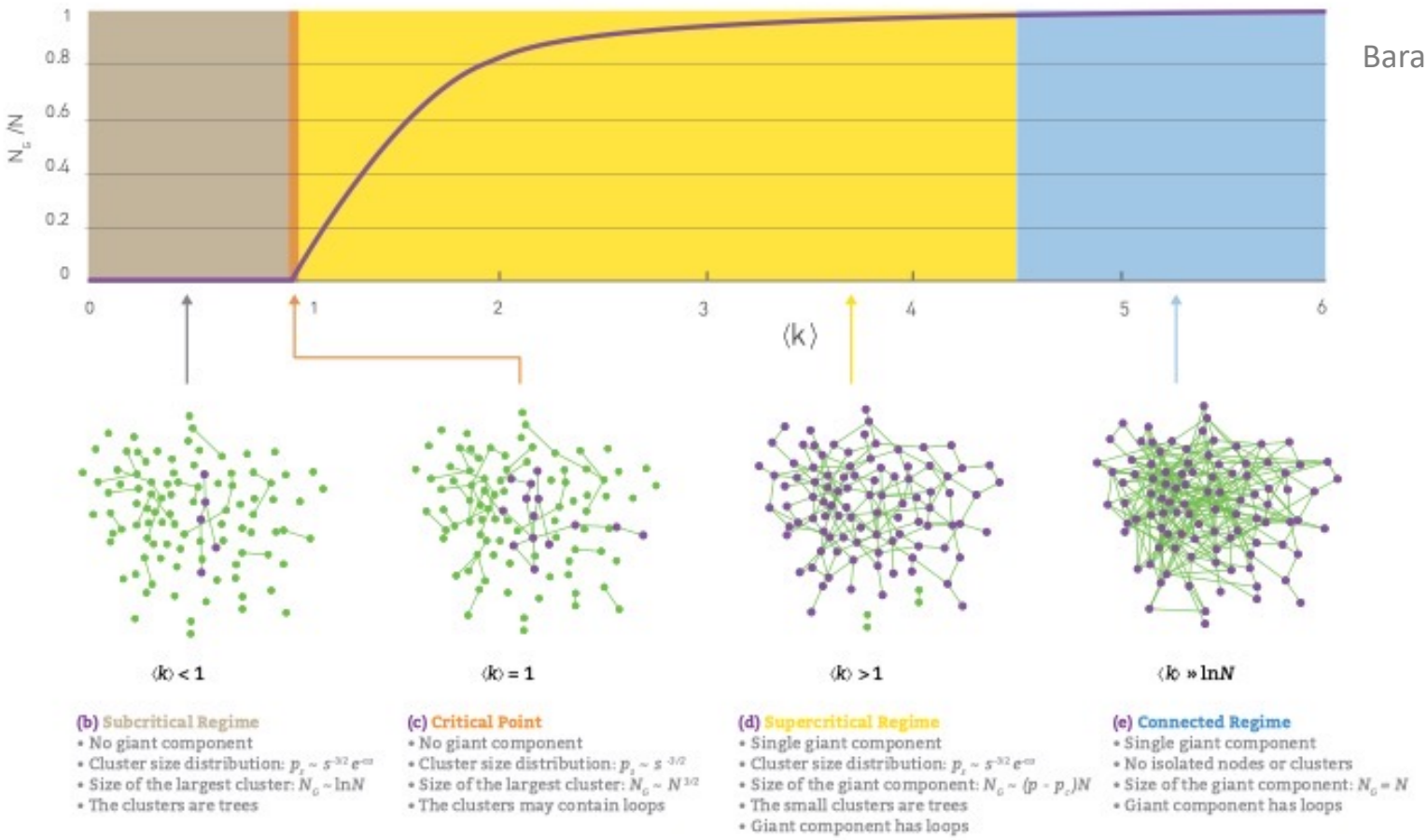
- Project 1 has been released and is due this Friday
  - I will post a few clarifications on Blackboard after today's lecture
- The midterm is Wednesday, 10/11
  - It is a 30 minute test that covers lectures 1-6 and the first video of lecture 7
  - I will release last year's midterm after today's lecture. I have removed one part of one question which is on a topic we are not covering this year.
- The lecture schedule on the module info pdf indicates a lecture recording will be released next week. I will delay this to a subsequent week that does not have a problem class.
- Recordings of live lectures are generally available on Teams shortly after the lecture. I have also posted lectures 1, 2, and 4 on Panopto
  - This class has 16 lectures. Nine are live; seven are pre-recorded.
- Solutions to problem sheets and labs are posted on Blackboard every week. Solutions to problem class questions are not released during term. Answers to these questions will be provided along with the collected slides at the beginning of April. In the meantime, you can ask questions in one of the three remaining problem classes.

Network Science
Autumn 2021

Lecture 6

Critical assessment of G_{Np} model
The configuration model

G_{Np} random graphs



Barabasi, figure 3.7

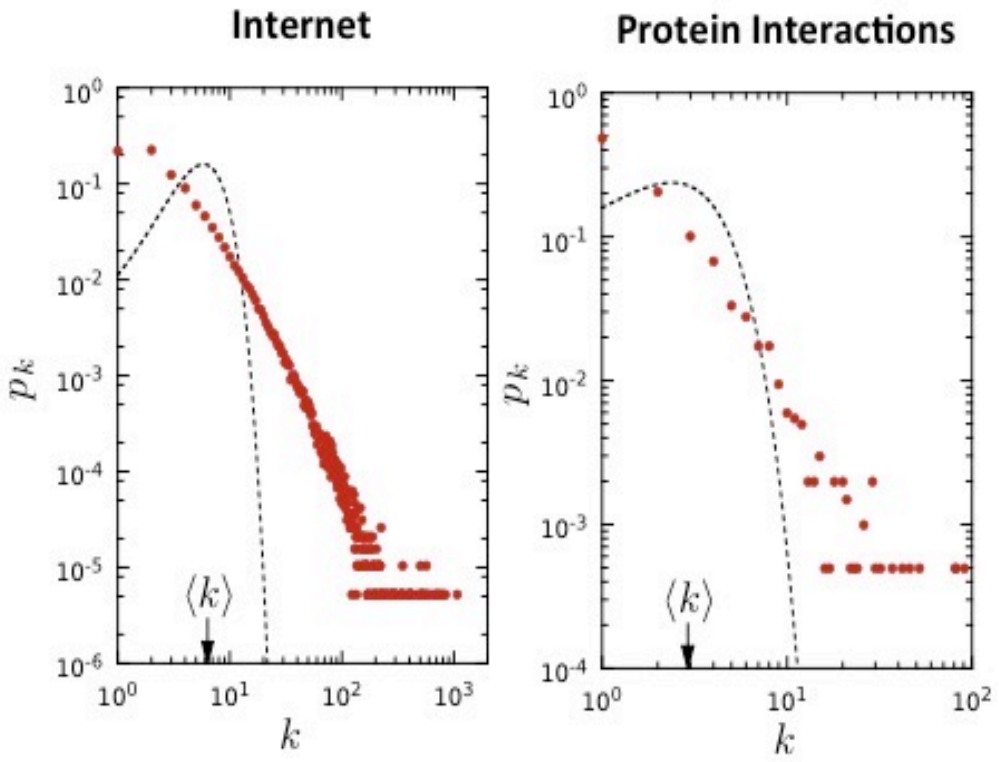
- **The figure above from Ch. 3 of Barabasi has connections to many of the ideas covered in lecture 5**
- **Despite the very simple rule for generating graphs, there is a wide variety of graph structures that can be observed. But is this model *useful*?**

- Let's critically compare three graph properties: 1) degree distribution, 2) clustering, and 3) distances

- The G_{Np} model **degree distribution** follows the binomial distribution with most nodes having degrees close to $\langle k \rangle$

- However, most large complex networks exhibit different behavior. They typically have large-degree hubs, and the average degree, $\overline{k}$ is not particularly important

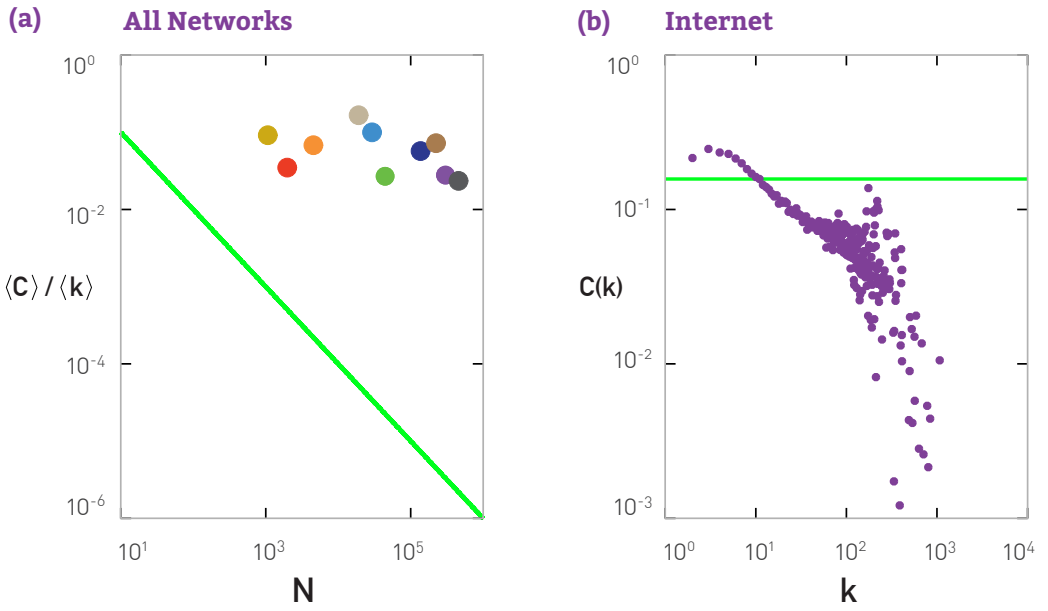
- The figure on the right compares the degree distribution of two real networks with p_k for G_{Np} graphs setting $\langle k \rangle = \overline{k}$ and matching N.



Barabasi, figure 3.6

- Let's critically compare three graph properties: 1) degree distribution, 2) clustering, and 3) distances

Barabasi, figure 3.13



- Say that node i has degree $k_i = k$ with $k > 1$. What is the expected number of links amongst these k neighbors? $p\binom{k}{2}$ since the maximum number of links is $\binom{k}{2}$. The expected clustering conditional on $k > 1$ is defined as $\langle C_i \rangle_{k>1} = \frac{p\binom{k}{2}}{\binom{k}{2}} = p$. It is independent of the node degree, and $\frac{\langle C_i \rangle_{k>1}}{\langle k \rangle} = \frac{1}{N-1}$

- Plot (a) on the right compares $\frac{1}{N-1}$ with $\frac{\overline{C}}{\overline{k}}$ for 10 real networks. The real networks have much higher clustering than what we would expect based on the G_{Np} model

- Plot (b) shows that for the internet (and essentially all real networks), the clustering depends on the node degree unlike what the G_{Np} model predicts.

- **Let's critically compare three graph properties: 1) degree distribution, 2) clustering, and 3) distances**

- **The diameter of the giant component in G_{Np} graphs is estimated to be $D \approx \frac{\log N}{\log\langle k \rangle}$, i.e. the diameter varies logarithmically (slowly) with the number of nodes.**

- **Consider the global social network. Take $N \approx 8 \text{ x } 10^9$ and $\overline{k} \approx 500$. Then, the G_{Np} model predicts, $D \approx 3.7$. I.e. there are 3-4 "degrees of separation" between any two people. This is reasonably close to what social scientists expect!**

- **Generally, it is the average distance $\overline{d}$ of real networks rather than the diameter that tends to be close to $\frac{\log N}{\log\langle k \rangle}$ rather than the diameter. The main point is that distances are "short" both in G_{Np} graphs and in real complex networks.**

- For the most part, the G_{Np} model does not accurately describe large real-world networks.

- This is not surprising. We know that networks are not designed or constructed by randomly placing links based on Bernoulli trials!

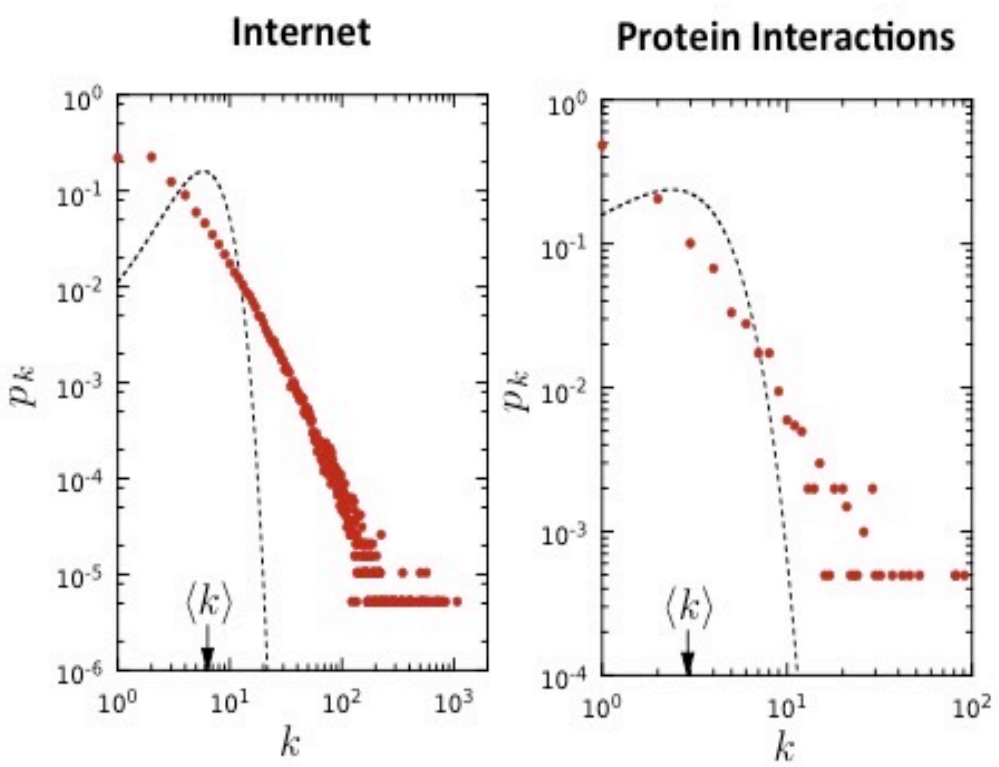
  - So why study this model? It is a useful reference when making comparisons with real networks.

  - It is also a useful reference for graph models. We will consider two more sophisticated models, and a good understanding of the G_{Np} model helps us understand the strengths and weaknesses of these advanced models which have been developed more recently.

  - Many mathematicians over the years have also found it to be "interesting".

Configuration model

- The next model we will analyze is the *configuration model*
- The aim is to adapt the G_{Np} model so that simulated graphs match a specified degree distribution
- The model was introduced by Bender & Canfield (1978) and then developed, analyzed, and named by Bollobas (1980)



We will:

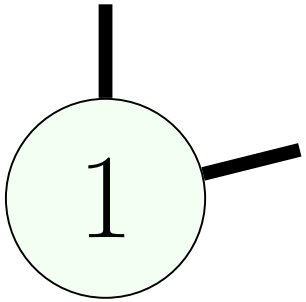
1. Introduce the model
2. Manually build a simple graph
3. Analyze a few important properties of the model

We will *not* comprehensively investigate the model

- **The configuration model doesn't specify a degree distribution, rather it specifies a degree sequence, $\mathrm{d} = (k_1, k_2, \ldots, k_N)$, which sets the degree of each node**
- **Note: the total degree, $K = \sum_{i=1}^{N} k_i$, must be even**
- **For convenience, we do not allow isolated nodes with $k_i = 0$**

Generating a graph:

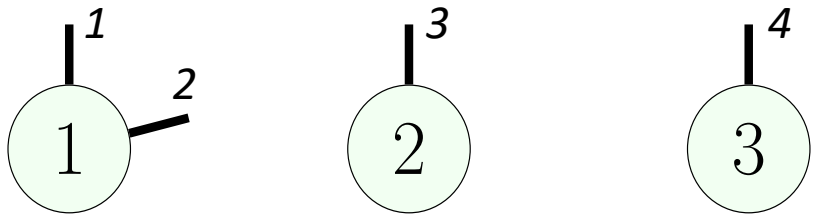
1. **Place k_i "stubs" on node i for each i**
2. **Assign a number to each of the K stubs**
3. **Randomly choose 1 pair of stubs and connect them (they are then no longer stubs)**
4. **Repeat step 3 until all stubs have been connected**



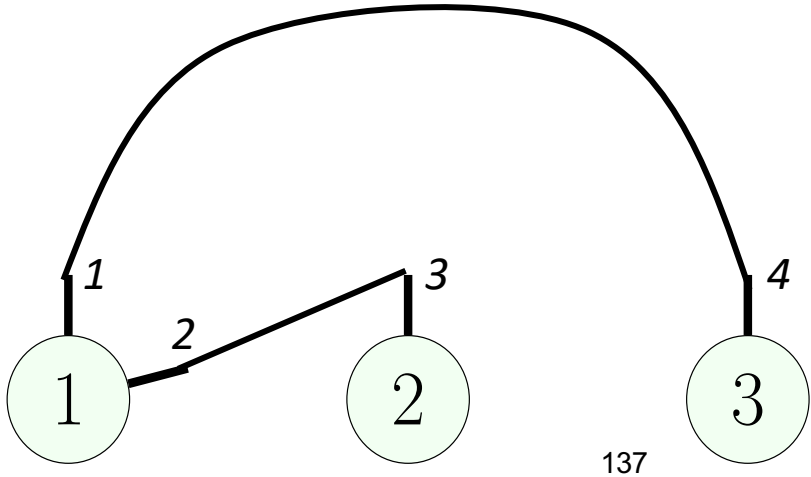
A node with 2 stubs

Simple example

- **Consider the degree sequence,** $(2,1,1)$
- **The nodes with labeled stubs are shown →**
- **A realization can be generated with the following crude algorithm:**



1. **Create a list of all stubs,** (s_1, s_2, s_3, s_4)
2. **Select** 2 **uniformly at random, place a link between the pair, and remove them from list**
3. **Repeat step 2 until two stubs remain, and place a link between them**
4. **So, if we 1st choose** s_2 **and** s_3**, we get →**

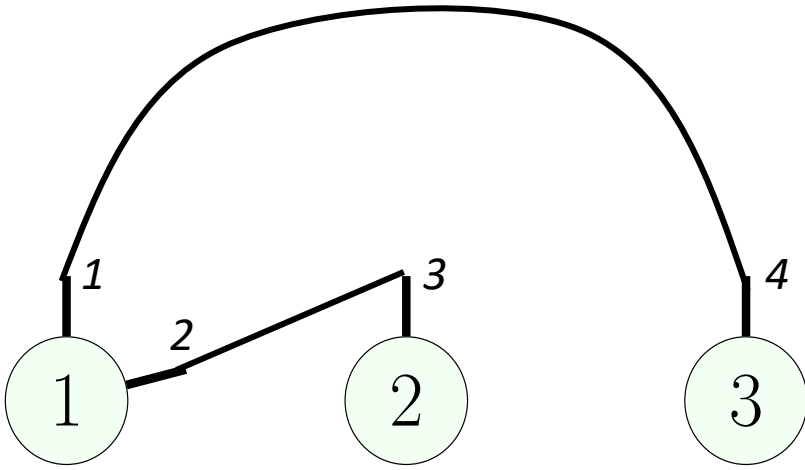


- This graph, G, is just one realization – the model defines a sample space of realizations for a given degree sequence. We can say that $G \in G_c(\mathrm{d} = (2,1,1))$ where $G_c(\mathrm{d})$ indicates the ensemble of graphs generated by the configuration model with degree sequence, d

- Any one stub is equally likely to connect to any other stub with probability $1/(K-1)$

- For this example, there are 3 graphs that can be generated with equal probability

- However, this is not generally the case. For a given degree sequence, it is possible for some graphs to be generated more frequently than others.



Notes:

- This model allows self-loops and multiple links between a pair of nodes

- Most real networks don't have these kinds of links, but it can be shown that they are "rare" as the graph size becomes large (and with some constraints on the degree sequence)

Model properties

Question 1: What is the expected number of links between distinct nodes i and j?

- **Set "a" as one of k_i stubs on node i and "b" as one of k_j stubs on j**
- **Let X_{ab} count the number of links between a and b (it will be either 0 or 1)**
- **Then $P(X_{ab} = 1) = \frac{1}{K-1}$ as stated earlier, and $\langle X_{ab} \rangle = \frac{1}{K-1}$**
- **Finally, let l_{ij} be the number of links between i and j (with $i \neq j$)**
- **Then, $\langle l_{ij} \rangle = \sum_{a=1}^{k_i} \sum_{b=1}^{k_j} \langle X_{ab} \rangle = \frac{k_i k_j}{K-1}$**

- **We immediately see that higher-degree nodes tend to link to each other more frequently**
- **Of course, this will only be important if the degree sequence contains a reasonably broad range of degrees!**

Question 2: What is the expected number of self-loops on node i?

- **Let "a" and "b" be distinct stubs on node i**
- **X_{ab} will again count the number of links between a and b, and: $P(X_{ab} = 1) = \langle X_{ab} \rangle = \frac{1}{K-1}$**
- **Then, $\langle l_{ii} \rangle = \sum_{a=1}^{k_i - 1} \sum_{b=a+1}^{k_i} \langle X_{ab} \rangle = \frac{k_i(k_i-1)}{2(K-1)}$ where $\langle l_{ii} \rangle$ is the expected number of self-loops on i**
- **The indices for the double sum have been set to ensure that a and b are distinct**
- **We see that if $K \gg k_i^2$, the number of expected self-loops on i will be small**

Question 3: What is $\langle s \rangle$, the expected number of self-loops in a graph? (assuming that $K \gg 1$)

- **Using linearity of expectation, this is $\langle s \rangle = \sum_{i=1}^{N} \frac{k_i(k_i-1)}{2(K-1)} \approx \sum_{i=1}^{N} \frac{k_i(k_i-1)}{2K}$**

- **We can put this in a more interpretable form by recognizing that:**
  - **The sum can be re-written as: $\sum_{k=1}^{k_{max}} N_k \frac{k(k-1)}{2K}$ where N_k is the number of nodes with degree k**
  - **$\frac{K}{N} = \overline{k}$, the average degree**
  - **$\frac{N_k}{N} = p_k$, the fraction of nodes with degree k and…**

- $\sum_{k=1}^{k_{max}} p_k\, k = \overline{k},\ \sum_{k=1}^{k_{max}} p_k\, k^2 = \overline{k^2}$

- **Putting all of this together, we have:** $\langle s \rangle \approx \frac{\left(\overline{k^2} - \overline{k}\right)}{2\overline{k}}$

- **A key conclusion is that if we hold $\overline{k^2}$ and $\overline{k}$ fixed to finite values and consider a sequence of graphs with $L \to \infty$, then the fraction of edges which are self-loops goes to zero, $\frac{\langle s \rangle}{L} \to 0$**

Summary

- We began this lecture pointing out a few important weakness of the G_{Np} model

- The configuration model produces network which:
  - Can have a realistic degree distribution
  - May have higher clustering G_{Np} graphs
  - Exhibit the small world property

- So, the configuration model is a step forward. But important questions remain as will be discussed in upcoming lectures

- The discussion here is really just a brief overview. There is much more one can do to analyze and improve the configuration model. For an advanced and somewhat different treatment, see the (highly-cited) paper by Newman, Watts, and Strogatz, "Random graphs with arbitrary degree distributions and their applications"

Network Science
Autumn 2021

Lecture7

More on the configuration model
The friendship paradox

Recap

In the previous lecture we introduced the configuration model which:

- **Is a random graph model that can produce graphs with realistic degree distributions**
- **Requires the specification of a degree sequence,** $\mathrm{d} = \{k_1, k_2, \dots, k_N\}$
- **And has the key property that the probability that any two stubs are linked is** $\frac{1}{K-1}$ **where** K **is the "total degree",** $K = \sum_{i=1}^{N} k_i$ **and is also the total number of stubs in the graph.**
- **When analyzing the configuration model, it is often convenient to work with** N_k**, the number of nodes with degree** k**, or** p_k**, the fraction of nodes with degree** k**.**
  - **We then aim (when we can) to provide expressions in terms of the moments of the degree sequence:**
    - $\overline{k} = \frac{1}{N}\sum_{i=1}^{N} k_i$
    - $\overline{k^2} = \frac{1}{N}\sum_{i=1}^{N} k_i^2$

Configuration model, continued

We continue with a fourth question related to the configuration model:
Question 4: What is the *probability* that node i does not have any self loops ?

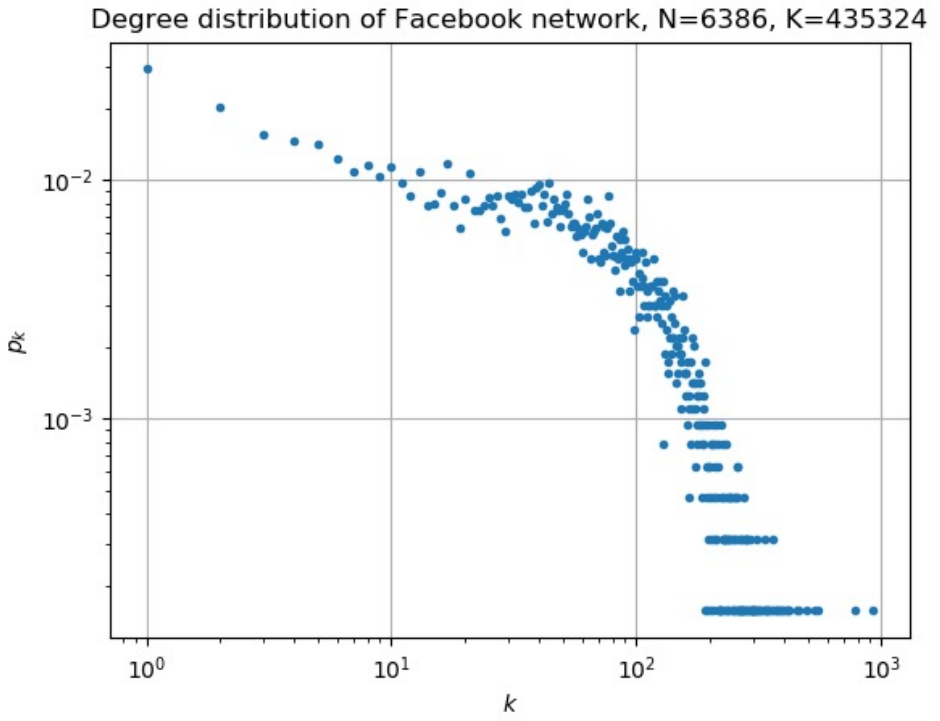
- **So we need to find $P(l_{ii} = 0)$**
- **Say that the stubs are numbered from 1 to k_i . The probability that stub 1 does not connect to any of the other stubs on i is $1 - \frac{k_i - 1}{K - 1}$**
- **The probability that neither stub 1 nor stub 2 connect to node i is,**

$$\left(1 - \frac{k_i - 1}{K - 1}\right)\left(1 - \frac{k_i - 2}{K - 3}\right)$$

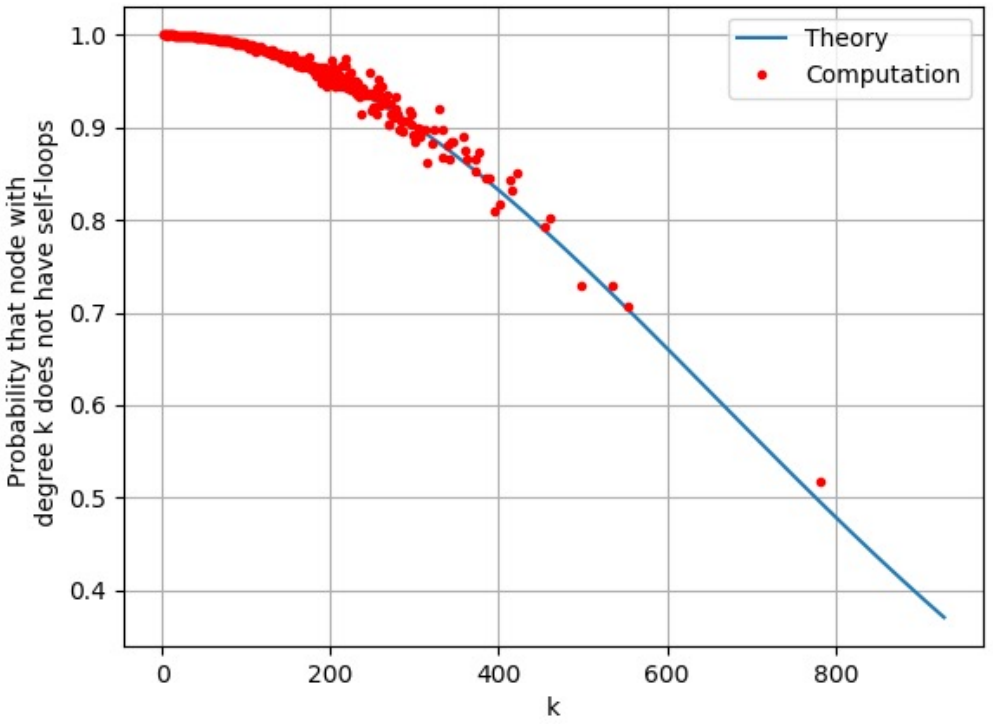
- **And continuing on to stub $k_i - 1$, we find, $P(l_{ii} = 0) = \prod_{m=1}^{k_i - 1}\left(1 - \frac{k_i - m}{K - 2m + 1}\right)$**
- **It is more challenging to work with probabilities than expectations when analyzing the configuration model**

Check your understanding: Why does the product go to $k_i - 1$ rather than k_i?

- **Let's look at an example, the figure shows the degree distribution for a university's Facebook friend network from 2005**
- $\overline{k} = 68.17, \overline{k^2} = 8495.74$
- **So,** $\langle s \rangle \approx 61.81$
- **Using the configuration model (and NetworkX) to generate 100 graphs with this degree sequence gives an average of** 61.27 **self-loops per graph**
- **Note as well that** $\frac{\langle s \rangle}{L} \approx \frac{61.81}{\left(\frac{435324}{2}\right)} \approx$ 3e-4. **The fraction of nodes expected to be self loops is small.**



- Let's also check our result for the probability that a node does not have self-loops
- Now generating 400 graphs, compute the total number of nodes with degree k that do not have self loops and divide that total by $400N_k$
- The figure compares this average with $P(l_{ii} = 0)$ for each unique degree in the degree sequence
- This indicates that the theory works pretty well, and we also see the (expected) result that hubs will have a higher tendency to have at least 1 self-loop



We will now state a few results for more familiar quantities.

- **The expected clustering for a node with degree k is, $\langle C \rangle \approx \frac{\left(\overline{k^2} - \overline{k}\right)^2}{N\,\overline{k}^3}$**

- **This approximation assumes that the graph is sufficiently large for the influence of self-loops and multi-edges to be negligible**

- **Recall that the clustering coefficient for a G_{Np} graph is $p = \frac{\langle k \rangle}{N-1}$**

- **So we find for both models that the clustering goes to zero as N is increased.**

- **However, for some reasonable degree sequences, the numerator $\left(\overline{k^2} - \overline{k}\right)^2$ can be large, and the configuration model will then produce much more clustering than a G_{Np} random graph (on average).**

- The configuration model exhibits the small world property, i.e. the diameter increases logarithmically with the graph size

- A giant component in a sparse family of configuration model graphs is present with high probability if and only if the ***Molloy-Reed criterion*** is satisfied: $\overline{k^2} - 2\,\overline{k} > 0$

  - A configuration-model graph family is sparse if the average degree remains finite as $N \to \infty$

  - Here we are considering a sequence of degree sequences with the length of the degree sequence increasing

  - Most large real-world networks satisfy this criterion and also contain giant components

Last question: Why (might) your friends have more friends than you?
We consider a related problem:

Construct a degree sequence corresponding to a social network, generate a graph with this sequence, choose a node, and then: what is π_k, the probability that an arbitrary stub on the chosen node connects to a stub on a node with k friends?

- **The graph has N_k nodes with degree k, and the probability of a stub connecting to any one of these nodes is, $\pi_k = \frac{kN_k}{K-1} \approx \frac{kN_k}{K}$ when $K \gg 1$**
  - **The numerator is just the total number of stubs connected to nodes with degree k, and we have assumed that the selected node does not have degree k (how would the above be modified if it did?)**
- **Then using the same simplifications used in lecture 6 when computing $\langle s \rangle$, we find:**

$$\pi_k \approx \frac{kp_k}{\overline{k}}$$

We can now compute the expected degree of the friend:

$$\langle k_{friend} \rangle = \sum_k \pi_k k = \frac{1}{\bar{k}} \sum_k k^2 p_k = \frac{\overline{k^2}}{\bar{k}}$$

- **I didn't specify how we choose a node; we could choose the same node each time and this result would hold. However, to partially address our initial question, it is best to choose the node uniformly at random each time a new graph is generated.**
  - **The expected degree of this node will then be $\bar{k}$**
  - **And then we can consider $\langle k_{friend} \rangle - \bar{k} = \frac{\overline{k^2} - \bar{k}^2}{\bar{k}}$**
- **The numerator on the RHS is the (biased) sample variance of the degree sequence and the expression will be positive provided that the degrees are not all the same.**
- **So we expect the friend to have a higher degree than the degree of the randomly chosen node. Real complex networks tend to have variances that are much larger than the average degree, so the "typical" friend can be considerably more popular.**

- For a real network, do we see anything like, $\langle k_{friend} \rangle \approx \frac{\overline{k^2}}{\overline{k}}$?

- Consider the following problem for a real network: Interpret each link as 2 stubs. Randomly select one stub, follow it via its partner stub to a node, f. What is $\langle k_f \rangle$?

- This is equivalent to randomly choosing a stub and considering its node:

  - We have kN_k stubs on nodes with degree k and there are K stubs in total, so $P(k_f = k) = \frac{N_k k}{K} = \frac{k p_k}{\overline{k}}$ where p_k is the fraction of nodes with degree k and $N_k = p_k N$

  - Then, $\langle k_f \rangle = \frac{\overline{k^2}}{\overline{k}}$ which will again be larger than $\overline{k}$, the expected degree of a randomly chosen node if the degree distribution has finite variance.

  - The simple underlying idea here is that choosing links rather than nodes generates a bias towards high-degree nodes

Friendship paradox

These ideas about friends of friends are loosely related to what is known as the "friendship paradox"

- **Consider the table shown taken from:** Feld, Scott L. "Why Your Friends Have More Friends Than You Do." *American Journal of Sociology*, vol. 96, no. 6, 1991, pp. 1464–1477

- **This is data from a small social network in an American high school**

A Summary of the Numbers of Friends and the Mean Numbers of Friends of Friends for Each of the Girls in Figure 1

	Number of Friends (x_i)	Total Number of Friends of Her Friends (Σx_j)	Mean Number of Friends of Her Friends ($\Sigma x_j/x_i$)
Betty	1	4	4
Sue	4	11	2.75
Alice	4	12	3
Jane	2	7	3.5
Pam	3	10	3.3
Dale	3	10	3.3
Carol	2	4	2
Tina	1	2	2
Total	20	60	23.92
Mean	2.5*	3†	2.99*

- **The 1st column is the degree of each student (the total is K, the mean is $\overline{k}$)**

- **The 2nd column is the total degree of all friends of an individual. Say there was a 9th student with 8 friends. Then she would contribute 8 to each entry in this column.**

  - **The "Total" and "Mean" in this column are familiar quantities. To see this, we just need to work with the network adjacency matrix.**

Friendship paradox

- Say that f_i is the total number of friends that node i's friends has. Then, $f_i = \sum_{j=1}^{N} A_{ij} k_j$.

- The total number of friends of friends is then: $\sum_{i=1}^{N} f_i = \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} k_j$. Swapping the sums on the right-hand side and rearranging:

$$\sum_{i=1}^{N} f_i = \sum_{j=1}^{N} k_j \sum_{i=1}^{N} A_{ij} = \sum_{j=1}^{N} k_j^2 = N\overline{k^2}$$

- Now we scale the result by the total number of friends:
$\frac{total\ number\ of\ friends\ of\ friends}{total\ number\ of\ friends} = \frac{N\overline{k^2}}{K} = \frac{\overline{k^2}}{\overline{k}}$ which is the same as the expression for $\langle k_{friend} \rangle$ obtained with the configuration model. We know that $\frac{\overline{k^2}}{\overline{k}} > \overline{k}$ for networks with varying degrees, but this does not really tell us what an individual experiences.

- Instead, we should consider the average number of friends that a person's friends has. Say that k_i^f is the average number of friends of friends for node i. Then, $k_i^f = \frac{1}{k_i} \sum_{j=1}^{N} A_{ij} k_j$. The average of this quantity for the network is, $\overline{k^f} = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{k_i} \sum_{j=1}^{N} A_{ij} k_j \right)$ which can't be simplified easily like the previous case.

- We know that $\frac{\overline{k^2}}{\overline{k}} = \frac{60}{20} = 3$ has to be greater than or equal to $\overline{k} = 2.5$
- But $\overline{k^f}$ is more difficult to analyze
- The relative magnitude of $\overline{k^f}$ and $\overline{k}$ depends on the details of the graph structure, i.e. do "popular" people tend to be friends with each other?

A SUMMARY OF THE NUMBERS OF FRIENDS AND THE MEAN NUMBERS OF FRIENDS OF FRIENDS FOR EACH OF THE GIRLS IN FIGURE 1

	Number of Friends (x_i)	Total Number of Friends of Her Friends (Σx_j)	Mean Number of Friends of Her Friends ($\Sigma x_j/x_i$)
Betty	1	4	4
Sue	4	11	2.75
Alice	4	12	3
Jane	2	7	3.5
Pam	3	10	3.3
Dale	3	10	3.3
Carol	2	4	2
Tina	1	2	2
Total	20	60	23.92
Mean	2.5*	3†	2.99*

- It has been empirically observed that typically $\overline{k^f}$ is tangibly larger than $\overline{k}$, and this is known as the "friendship paradox" – your friends tend to have more friends than you.
- Consider a simple illustrative example. A person has 50 friends but each of these friends has only 1 friend. Then, $\overline{k} = 100/51$ but $\overline{k^f} = \frac{1+50*50}{51} \gg \overline{k}$ and most would have far fewer friends than their friend
  - Alternatively if the 50 friends had relatively large degrees, this would reduce the difference between $\overline{k^f}$ and $\overline{k}$.

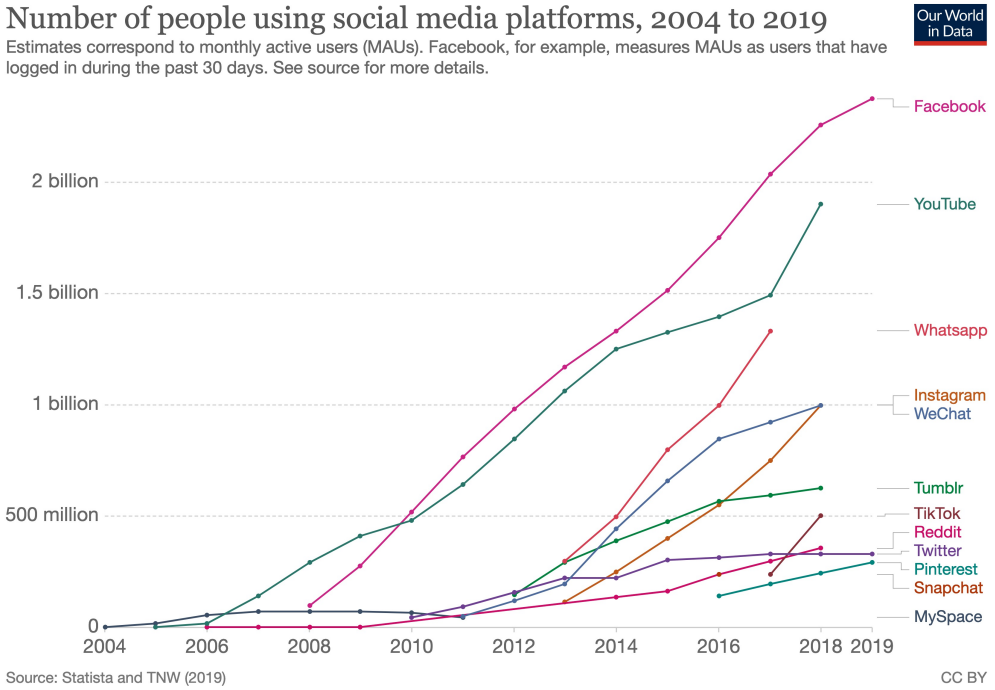
Network Science
Autumn 2021

Lecture 8

Dynamic graphs
Preferential attachment
Barabasi-Albert model

Dynamic graphs

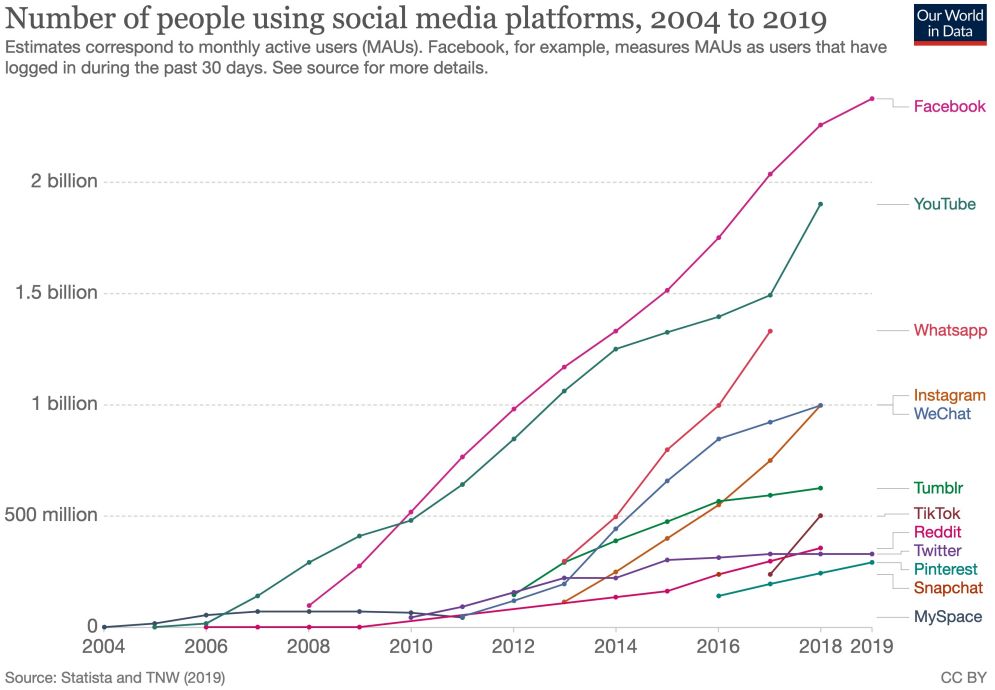
- **The configuration model resolves many of the weaknesses of the G_{Np} model**
- **So is there really any need to consider other models?**
  - **The configuration model mimics a network with a given degree distribution**
  - **It doesn't tell us *why* the network has a given distribution**



Another key point: most complex networks are not static, they evolve in time

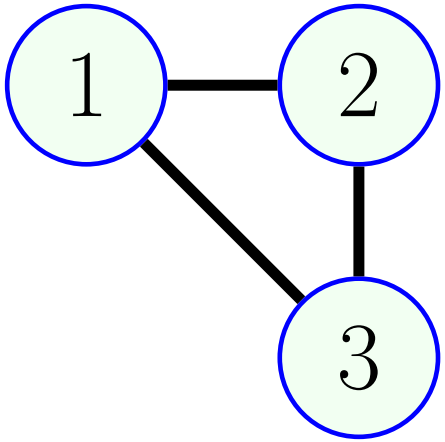
- **We know social networks evolve in time, a growing biological network can be observed here:** https://www.nikonsmallworld.com/galleries/2018-small-world-in-motion-competition/zebrafish-embryo-growing-its-elaborate-sensory-nervous-system
- **An *explanatory* model should show how networks evolve and how degree distributions develop**

- **We need to think about the addition and removal of nodes and links over time**
- **The probability of a node (or link) being added or removed will depend on the cost and benefit of the action**
- **Some links on Instagram have more value than others, creating an account has more value for some than others**
- **How can these ideas be translated to a mathematical model?**



- **We could assign a *fitness* to nodes and links which connect to a probability that nodes and links are added or removed**
  - **Similar ideas are commonly applied in evolutionary biology -- the relative fitness of a genetic mutation determines if it spreads in a population**

- Say our model network starts as shown →



- How do we assign fitness values? Should these values evolve in time?

- These are very complicated questions, a guiding principle to get started:

"*Everything should be as simple as it can be, but not simpler*" -
-- Einstein (paraphrased)

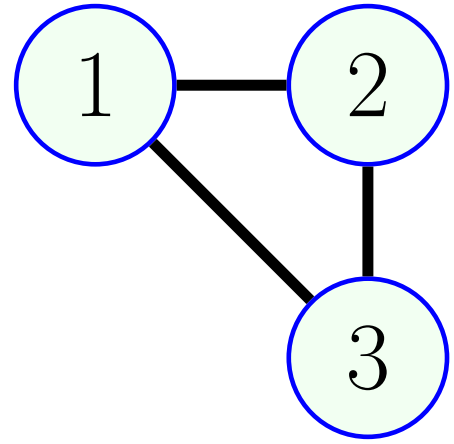
1. How fast should the network grow: 1 node per iteration

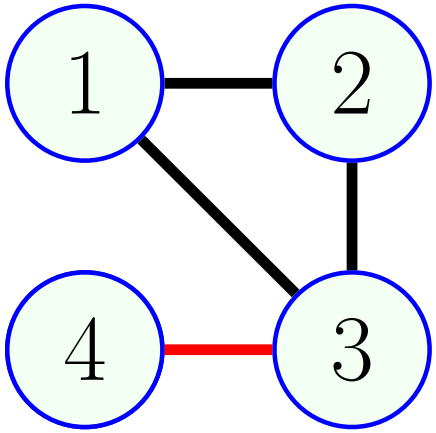
2. How/when should nodes/links be removed? In some complex networks node/link removal is rare, so once a node or link is added, let's keep it forever (see Barabasi, figure 6.11)

3. How should links be added? – this requires some thought

- Let's add 1 link per iteration
- Where should the new link be placed in our example? →
- We can just choose 1 node randomly for this case
- But what about the next iteration?
  - If we just place links randomly, the resulting model will have similar shortcomings to the G_{Np} model, i.e. this would be "too simple"
  - So, links should be placed with some consideration of fitness – which potential neighbor has the most value?

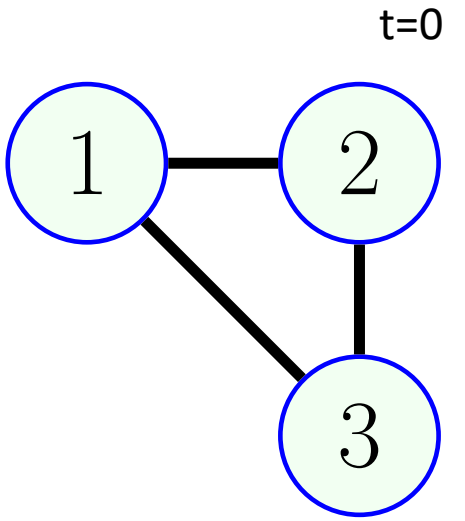
Preferential attachment: links to nodes with higher degrees have greater value





Preferential attachment

Preferential attachment: nodes with higher degrees have greater value



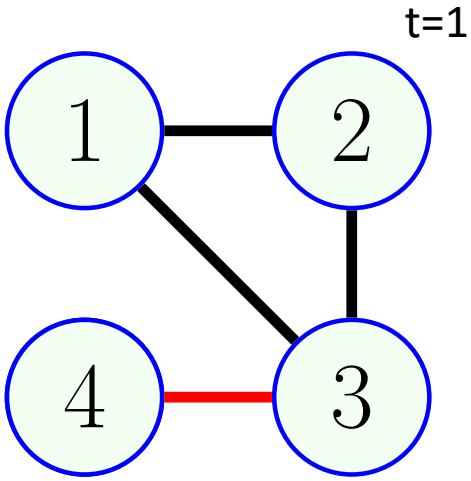
- **This needs to be translated into a mathematical statement**
- **The simplest approach is *linear* preferential attachment:**

  The probability of a new link attaching to an existing node is linearly proportional to that node's degree

  or

$$\rho_i(t+1|G_a(t)) = \frac{k_i(t|G_a(t))}{\sum_{i=1}^{N(t)} k_i(t)}$$

  where $\rho_i(t+1|G_a(t))$ is the probability of a link connecting to node i in graph $G_a(t)$ with $t \in \{1,2,3,\ldots\}$



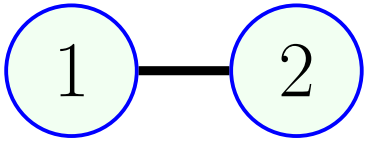
- **Simple perspective: interpret each link as 2 stubs. Choose 1 stub at random and place a link between the new node and the node which this stub connects to.**

- **To complete our model specification, we need to set the initial state of the model**

- **Again, we will keep things simple:**

  At $t = 0$, our network will be 2 nodes joined by a link

t=0



- **With this initial condition, the size of the network will be:**

$$N(t) = 2 + t, L(t) = 1 + t, t \in \{0,1,2, \dots\}$$

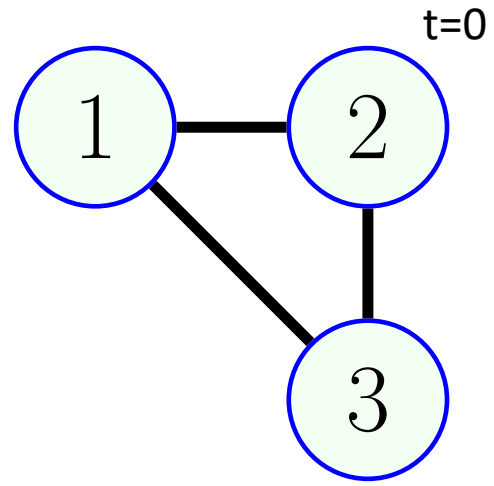
And: $\rho_i(t+1) = \frac{k_i(t)}{2L(t)} = \frac{k_i(t)}{2(1+t)}$

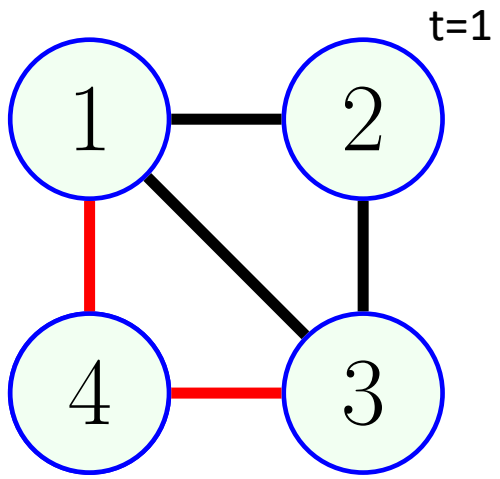
Note: this probability defines how the state of the graph at time t influences the state at time $t + 1$

- **The probability of a new link connecting node 3 to node 1 is $1/2$ or :** $\rho_1(t = 1) = \frac{1}{2}$

Barabasi-Albert model

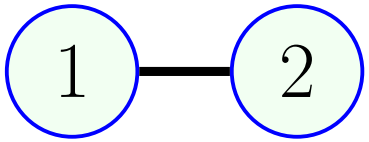
- Our model belongs to the family of *Barabasi-Albert* graphs
- The Barabasi-Albert model:
  - Allows q links to be added each iteration (q can be greater than 1)
  - The initial state consists of N_0 nodes where each node has at least one link and $N_0 \geq q$
  - The model is "underspecified" – see Box 5.1 in Barabasi
- Our model specification ($q = 1$ $N_0 = 2$) was chosen to simplify the analysis which follows





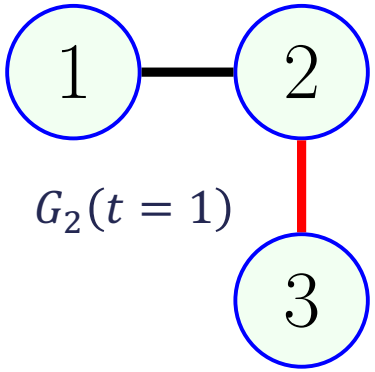
Example with $q = 2, N_0 = 3$

- **Let's think carefully about the 1st few iterations of our model**



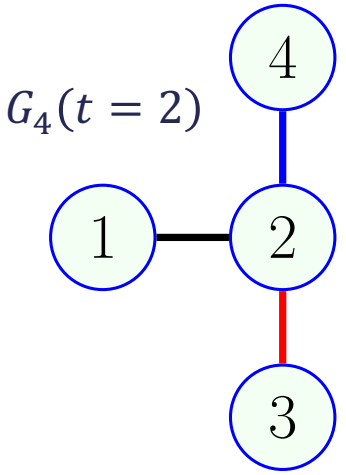
- **At $t = 0$, we have our specified initial state which we label, $G_1(t = 0)$**

- **At $t = 1$, there are two possible graphs which will be generated with equal probability ½ , $\{G_1(t = 1), G_2(t = 1)\}$. The subscripts are "assigned" arbitrarily**



- **What about $t = 2$? There are 3 possible "descendants" of each of the graphs generated at $t = 1$, so the size of the sample space is 6.**
  - **The probability of generating $G_4(t = 2)$ is:**

$P\big(G_4(2)\big) =$ **(Probability of generating $G_2(t = 1)$) * (Probability of adding link between nodes 2 and 4 to $G_2(t = 1)$)**



  - **This is straightforward to evaluate: $P\big(G_4(2)\big) = 0.5 * 0.5 = 0.25$. Note that the value is not $1/6$ – some graphs are more likely to be generated than others**

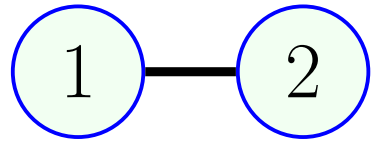
- **Generalizing, the size of the sample space at time t is $(t+1)!$**
- **The expectation at time t is computed as:**

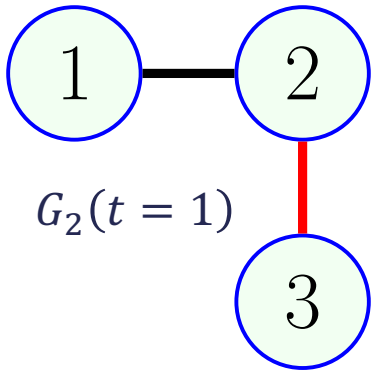
$$\langle f(t) \rangle = \sum_{m=1}^{(t+1)!} P\big(G_m(t)\big)\, f(t|G_m(t))$$

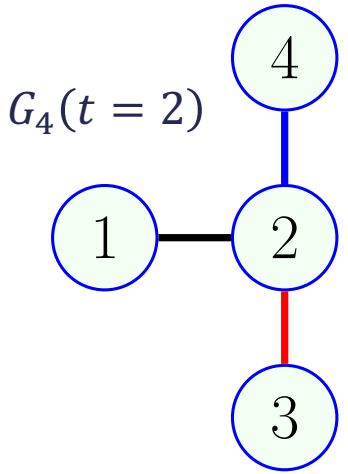
- **Here, $P\big(G_i(t)\big)$ is the probability of generating graph $G_i(t)$ and $f(t|G_i(t))$ is an arbitrary quantity evaluated on $G_i(t)$**
- **For example, $k_2(t=2|G_4(t=2)) = 3$**

Check your understanding: what is $\langle k_2(t=2) \rangle$?

- **Let's now apply these ideas to the computation of the degree distribution**

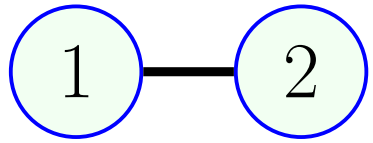


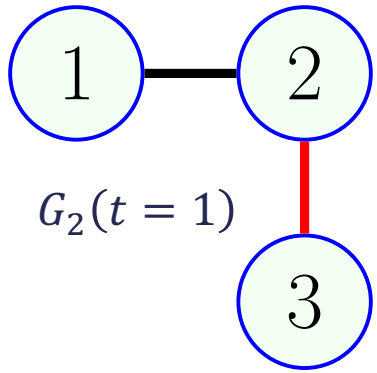


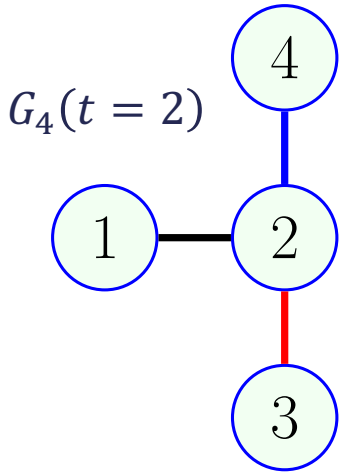


Degree distribution

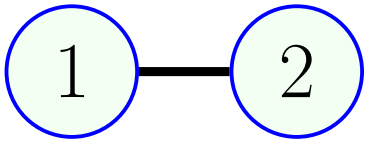
- **What is our degree distribution at $t = 0$?**
  - **All nodes have degree 1**
- **At $t = 1$?**
  - **Two nodes with degree 1, one with degree 2**
- **And then it becomes more complicated. Let's take an inductive approach**



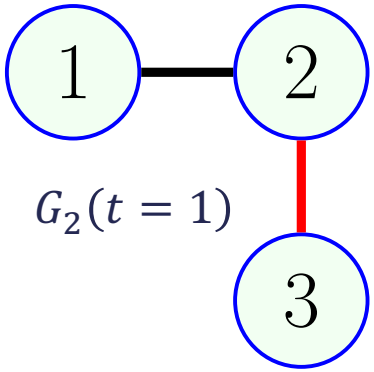


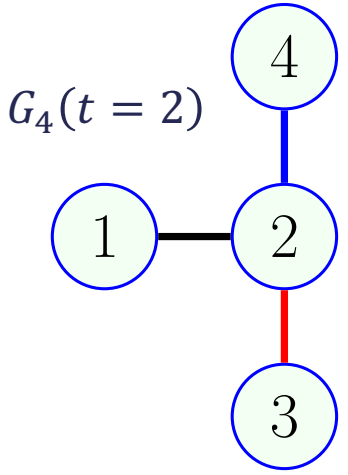


Say that we know the degree distribution at time t, $p_k(t)$. Then what is $p_k(t+1)$?



- **First, we introduce three random variables:**
  - $N_k(t)$: **number of nodes with degree k at time t. By definition,** $p_k(t) = \frac{\langle N_k(t) \rangle}{N(t)}$
  - $l_k(t+1|G_m(t))$: **An indicator variable, $l_k(t+1|G_m(t))$ is the number of links added to nodes in $G_m(t)$ with degree k. Will be either 0 or 1 (when $k = 1$ we will also have to account for the new node).**
  - **We will characterize an iteration with $N_k(t+1|G_m(t))$, the number of nodes with degree k at time $t+1$ given graph $G_m(t)$ at time t. This is a random variable which can take on three values. This notation may be a little confusing, so let's look at this variable more closely…**





There are three cases to consider for one iteration of our B-A model with $k > 1$:

- **Case A: If a node with degree k receives a link:**
$$N_k(t+1|G_m(t)) = N_k(t|G_m(t)) - 1$$

- **Case B: If a node with degree $k-1$ receives a link:**
$$N_k(t+1|G_m(t)) = N_k(t|G_m(t)) + 1$$

- **Case C: Otherwise:** $N_k(t+1|G_m(t)) = N_k(t|G_m(t))$

This perspective suggests the following approach for computing the expectation, $\langle N_k(t+1)\rangle$:

$$\langle N_k(t+1)\rangle =$$

$$\sum_{m=1}^{(t+1)!} P\big(G_m(t)\big)\,\{P(A)[N_k(t|G_m(t)) - 1\,] + P(B)[N_k(t|G_m(t)) + 1\,] + P(C)N_k(t|G_m(t))\}$$

Here, $P(A)$ is the probability of case A with equivalent definitions for the other cases, and $P(A) + P(B) + P(C) = 1$.

Next, we replace $P(A)$ and $P(B)$ using our indicator variable and obtain:

$$\langle N_k(t+1)\rangle =$$

$$\sum_{m=1}^{(t+1)!} P\big(G_m(t)\big)\,[N_k(t|G_m(t)) - P(l_k(t+1|G_m(t))=1) + P(l_{k-1}(t+1|G_m(t))=1)]$$

probability of adding link to node with degree k

probability of adding link to node with degree $k-1$

- **The above holds for $k>1$. For $k=1$, we modify the expression to account for a new node with $k=1$ being introduced each iteration and for the fact that there are no nodes with $k=0$:**

$$\langle N_1(t+1)\rangle =$$

$$\sum_{m=1}^{(t+1)!} P\big(G_m(t)\big)\,[N_1(t|G_m(t)) - P(l_1(t|G_a(t))=1) + 1]$$

- **We can simplify these equations substantially by recognizing that:**

$$P(l_k(t+1|G_m(t))=1) = \frac{kN_k(t|G_m(t))}{2L(t)}$$

  - **this follows from our definition of $\rho_i(t+1|G_m(t))$**

- **We now have the inductive results we need**

Master equations:

$\boldsymbol{k > 1}$**:**

$$N(t+1)p_k(t+1) = N(t)p_k(t) - \frac{p_k(t)N(t)k}{2L(t)} + \frac{p_{k-1}(t)N(t)(k-1)}{2L(t)}$$

$\boldsymbol{k = 1}$**:**

$$N(t+1)p_1(t+1) = N(t)p_1(t) - \frac{p_1(t)N(t)}{2L(t)} + 1$$

- **These equations, can be used to compute the evolution of the degree distribution**

- **These equations can be used for our next task -- obtaining an interpretable result when** $t \to \infty$

Network Science
Autumn 2021

Lecture 9

More on the Barabasi-Albert model
The power law distribution
Preferential attachment in real networks

B-A model: degree distribution

- **Previously, we derived the master equations for the degree distribution generated by the Barabasi-Albert (B-A) model**

Master equations:

$\boldsymbol{k > 1}$**:**

$$N(t+1)p_k(t+1) = N(t)p_k(t) - \frac{p_k(t)N(t)k}{2L(t)} + \frac{p_{k-1}(t)N(t)(k-1)}{2L(t)}$$

$\boldsymbol{k = 1}$**:**

$$N(t+1)p_1(t+1) = N(t)p_1(t) - \frac{p_1(t)N(t)}{2L(t)} + 1$$

- **Let's now examine what happens when the graphs become large**

We start with:

$$N(t+1)p_1(t+1) = N(t)p_1(t) - \frac{p_1(t)N(t)}{2L(t)} + 1$$

- **When $N \gg 1$, we see that we will have $p_1(t+1) \approx p_1(t)$ (remember that $N(t) = 2 + t, \;\; L(t) = 1 + t$), and as $t \to \infty$, we will have $p_1(t+1) = p_1(t) = p_{1,\infty}$**

- **The same argument can be used for general k giving, $p_k(t+1) = p_k(t) = p_{k,\infty}$ in the limit $t \to \infty$**

- **In order to find this *stationary distribution*, we rearrange the equation,**

  $$3p_1(t+1) - 2p_1(t) = t[p_1(t) - p_1(t+1)] - \frac{p_1(t)(2+t)}{2(1+t)} + 1$$

  and evaluate the limit as $t \to \infty$ while imposing the condition, $t(p_k(t+1) - p_k(t)) \to 0 \; as \; t \to \infty$.

- **This gives the result,**

$p_{1,\infty} = -\frac{p_{1,\infty}}{2} + 1$ **or simply,** $\boxed{p_{1,\infty} = \frac{2}{3}}$

and using the same approach for general k, we find,

$$p_{k,\infty} = -\frac{p_{k,\infty} k}{2} + \frac{p_{k-1,\infty}\,(k-1)}{2}$$

which is rearranged as, $\boxed{p_{k,\infty} = \frac{k-1}{k+2} p_{k-1,\infty}}$

We're almost done now, we just have to examine this recurrence

Rearranging our recurrence slightly:

$$p_{k+1,\infty} = \frac{k}{k+3}\, p_{k,\infty}$$

and noting that, $p_{1,\infty} = \frac{2}{3}, p_{2,\infty} = \frac{1}{4}\, p_{1,\infty} = \frac{1}{6}$

$$p_{3,\infty} = \frac{2}{5}\frac{1}{4}\frac{2}{3} = \frac{1}{15}$$

$$p_{4,\infty} = \frac{3}{6}\frac{2}{5}\frac{1}{4}\frac{2}{3} = \frac{2*1*2}{6*5*4} = \frac{1}{30}$$

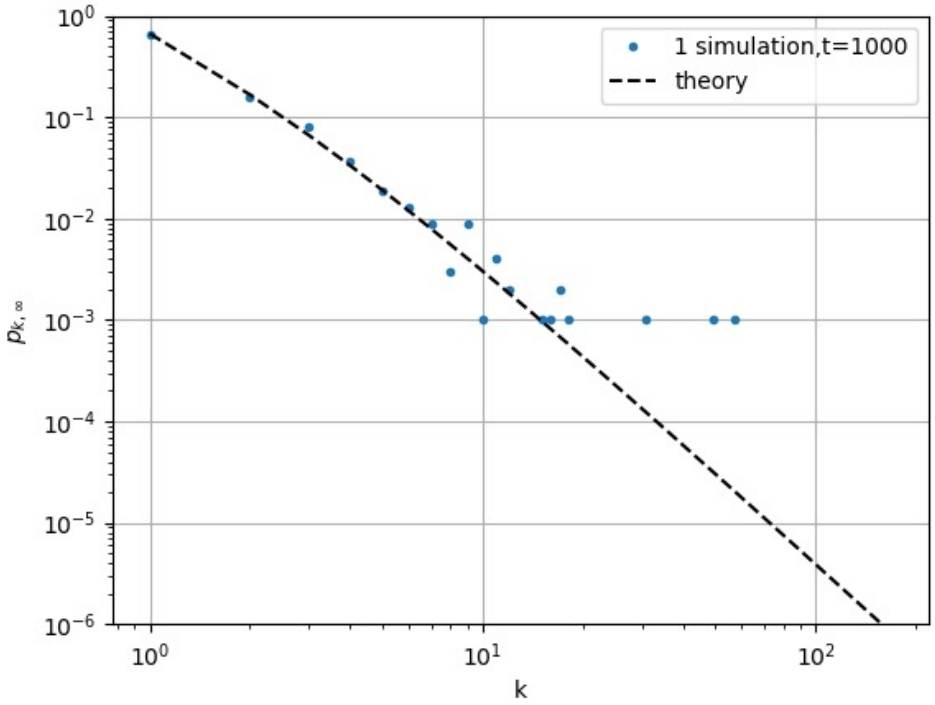
$$p_{5,\infty} = \frac{2*1*2}{7*6*5} = \frac{2}{105}$$

$$p_{k,\infty} = \frac{4}{(k+2)(k+1)k} \sim k^{-3}$$

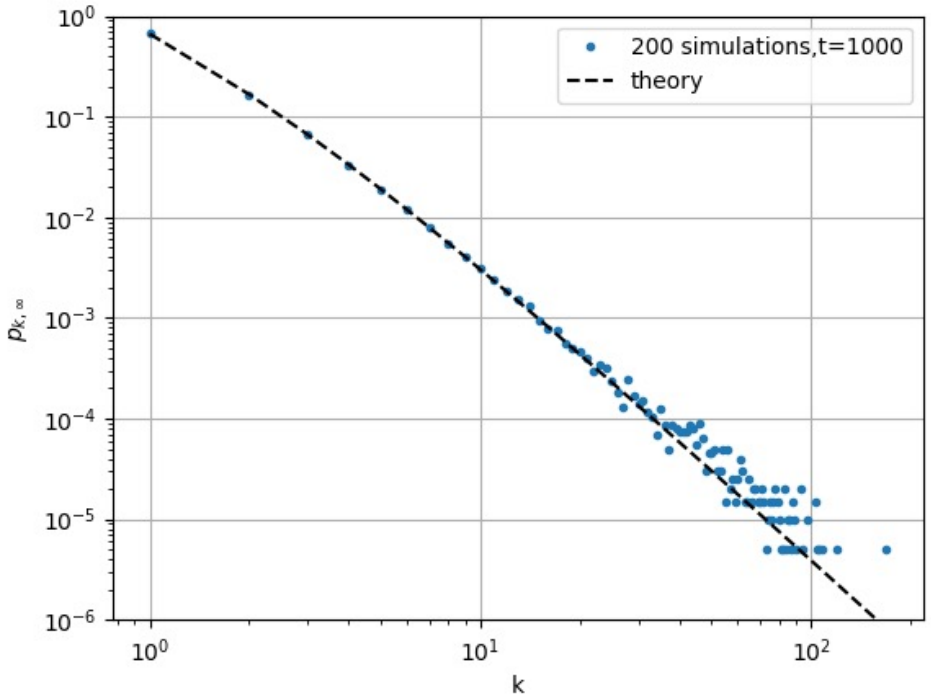
The approximation on the RHS applies for large k

Let's compare this expression to simulation results…

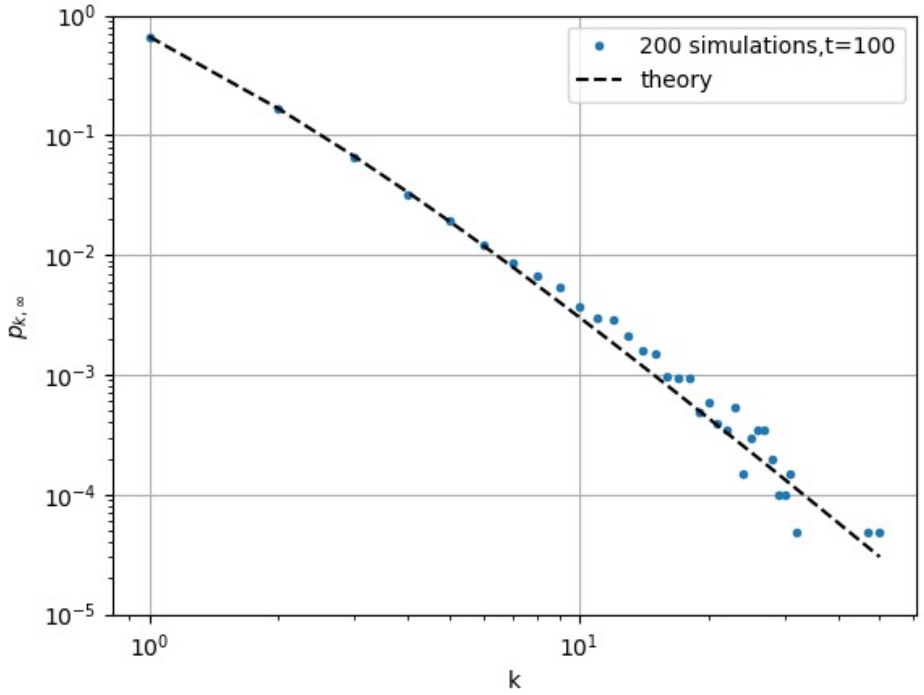
- **Running a simulation with 1000 iterations, we find the degree distribution shown**
- **There is some modest similarity with our theoretical result**
- **We have to remember that the theory corresponds to an expected value, so we should run several simulations and average the computed degree distributions**



- **Averaging results from 200 simulations, we see much clearer agreement**
- **Large degree hubs appear infrequently, so even more simulations are needed to accurately estimate the expected frequency of high-degree nodes**



- We see good agreement at earlier times as well →
- A rigorous comparison with theory would require independently varying the number of iterations and the number of simulations used for averaging
- We see that our simple model generates a power-law degree distribution which is a big improvement when compared to the G_{Np} model
- This model also produces "ultra-short" distances: $D \sim \log(N)/\log(\log(N))$



- However, the clustering coefficient is zero – the model is too simple!
- This is easily fixed by adding more links per iteration

B-A model: Node evolution

- **Our last piece of analysis for this model will focus on an individual node:**

  How does the degree of a node evolve in time?

- **Say the node is created at time t_0. Then, $k_i(t = t_0) = 1$**

- **We again introduce an indicator random variable:**
  - $X_i(t+1) = 1$ **if a link is added to node i at time t and is 0 otherwise**

- **So, $k_i(t+1) = k_i(t) + X_i(t+1)$ for $t \geq t_0$**

  - **Linearity of expectation gives: $\langle k_i(t+1) \rangle = \langle k_i(t) \rangle + \langle X_i(t+1) \rangle$**

  - **And we know that $\langle X_i(t+1) \rangle = P(X_i(t+1) = 1)$**

  - **What is $P(X_i(t+1) = 1)$?**

- **We rewrite $P(X_i(t+1)=1)$ as:**

$$P(X_i(t+1)=1)=\sum_{m=1}^{(t+1)!}P\big(G_m(t)\big)\,P(X_i(t+1)=1|G_m(t)),$$

and we know:

$$P(X_i(t+1)=1|G_m(t))=\rho_i(t+1)=\frac{k_i(t|G_m(t))}{2+2t},$$

so we have,

$$\langle k_i(t+1)\rangle=\langle k_i(t)\rangle+\sum_{m=1}^{(t+1)!}P\big(G_m(t)\big)\frac{k_i(t|G_m(t))}{2+2t}$$

- **The summation term is just, $\frac{\langle k_i(t)\rangle}{2+2t}$, and:**

$$\langle k_i(t+1)\rangle=\langle k_i(t)\rangle\left(1+\frac{1}{2+2t}\right)$$

- **Now using $k_i(t=t_0)=1$, we obtain an explicit expression for the expected degree:**

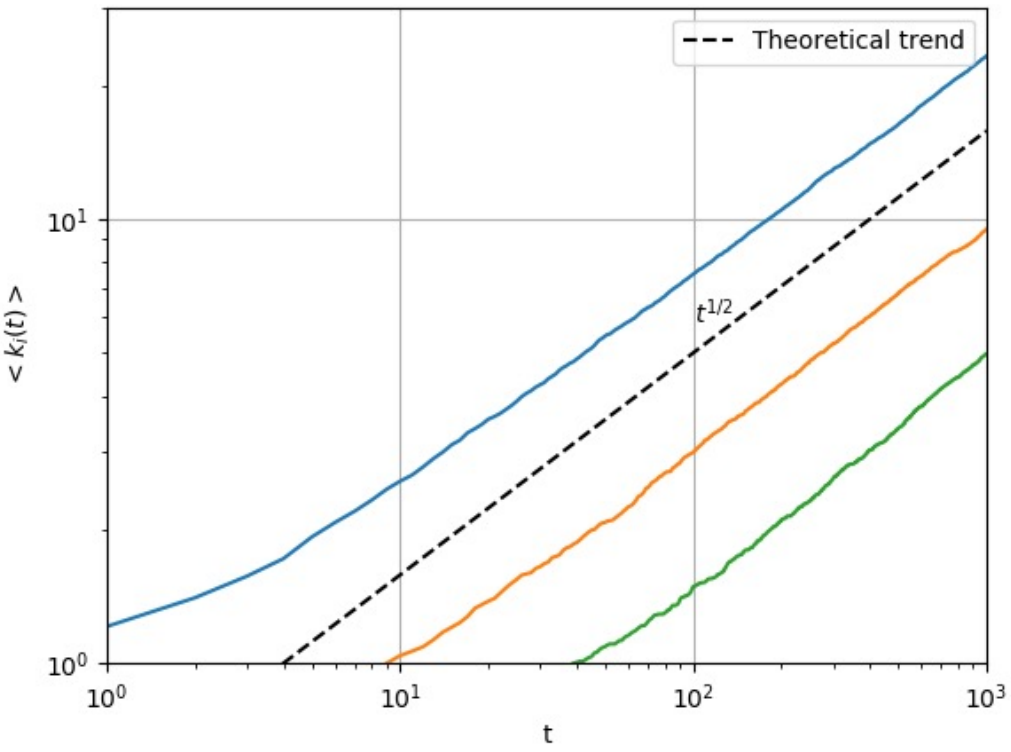
$$\langle k_i(t_0+j)\rangle=\prod_{n=0}^{j-1}\left[1+\frac{1}{2+2(t_0+n)}\right]$$

- **With some further work (see below), we can approximate the expected degree of node i as:** $\langle k_i(t_0 + j)\rangle \approx \sqrt{\frac{t_0+1+j}{t_0+1}}$

Note: to obtain the result above, we:

1. **Use the $\log$ function to convert the products into a series**
2. **Use the 1st term in a Taylor series approximation to convert the series into a harmonic series**
3. **Use a standard logarithmic approximation of the harmonic series**

- **The figure compares results from simulations (averaged over 200 realizations) with the trend from theory**
- **Aside from the early dynamics of the node introduced at $t = 0$, we see good agreement**
- **These trends illustrate the "1st mover effect" – the earlier a node is introduced to a network, the more links it tends to have**



Comments

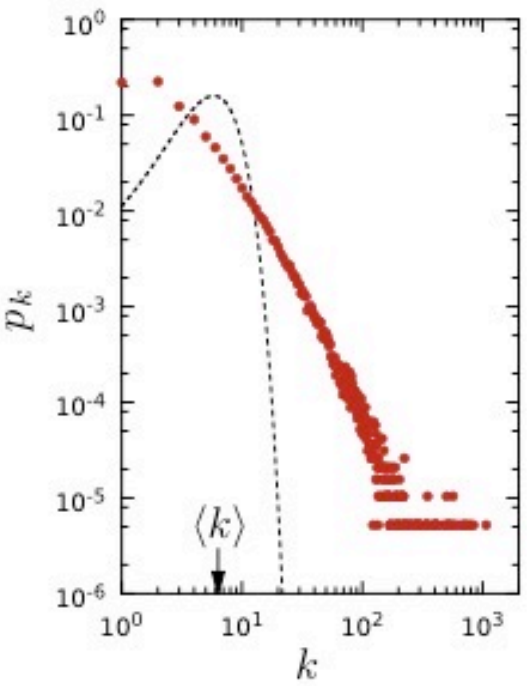
- The simple model introduced here generates graphs with many of the characteristics that we are looking for.
- By increasing the rate at which links are added, we can improve it further
- But even then, is it “too simple”?
- Yes.
  - This becomes apparent when we examine the growth rate of real networks
  - And also when we critically consider the model assumptions
- This is a starting point for “modern random graphs” and Network Science

Barabasi-Albert model

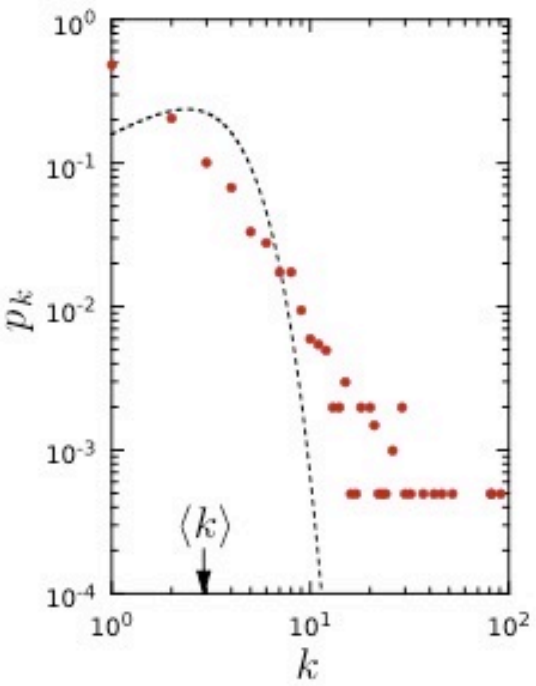
- We have seen that the simple linear preferential attachment model generates a "realistic" degree distribution, though the resulting graph is a tree with zero clustering

- If we add q links per iteration, the degree distribution is, $p_k \approx \frac{2q(q+1)}{(k+2)(k+1)(k)}$
- And the clustering is $\langle C \rangle \sim (\ln N)^2/N$

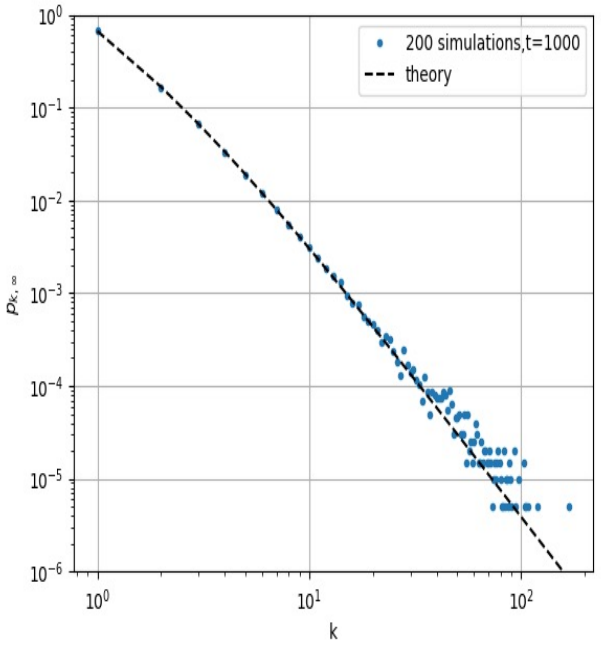
Internet



Protein Interactions



Barabasi-Albert



Power law distribution

- Let's examine the power law probability distribution more closely now
- The power law distribution is:

$$p_k = \frac{k^{-\gamma}}{\zeta(\gamma)}$$

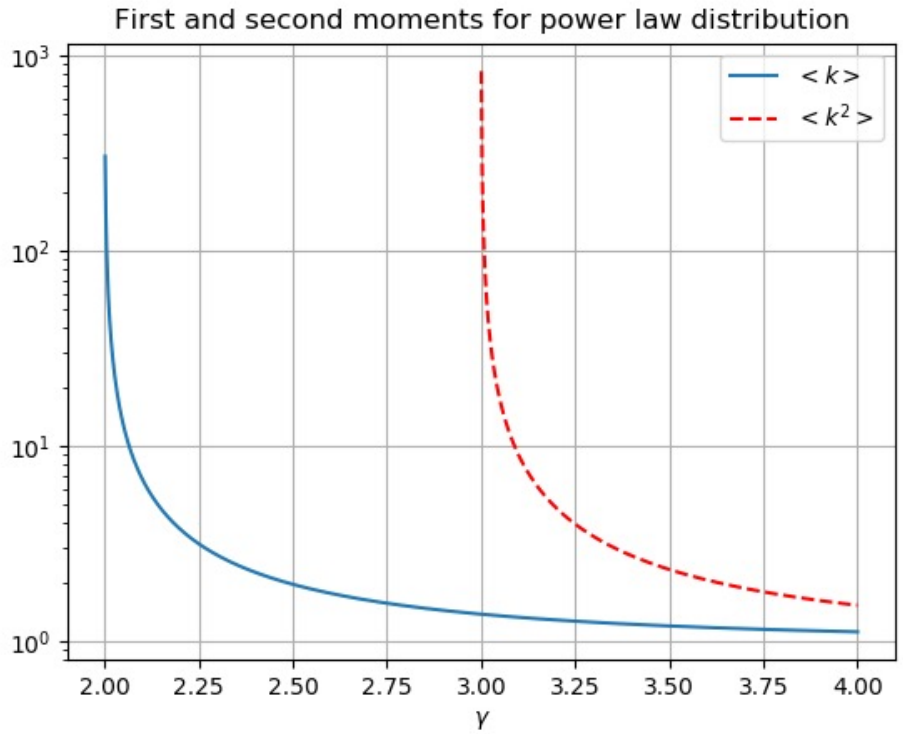
where $\zeta(\gamma)$ is the Riemann zeta function.

- The power law distribution is defined for $k \in \mathrm{Z}^+$ and the expected degree and 2nd moment are:

$$\langle k \rangle = \sum_{k=1}^{\infty} k p_k = \frac{\zeta(\gamma-1)}{\zeta(\gamma)}$$

$$\langle k^2 \rangle = \sum_{k=1}^{\infty} k^2 p_k = \zeta(\gamma-2)/\zeta(\gamma)$$

- $\langle k \rangle$ diverges for $\gamma < 2$ and $\langle k^2 \rangle$ diverges for $\gamma < 3$ →

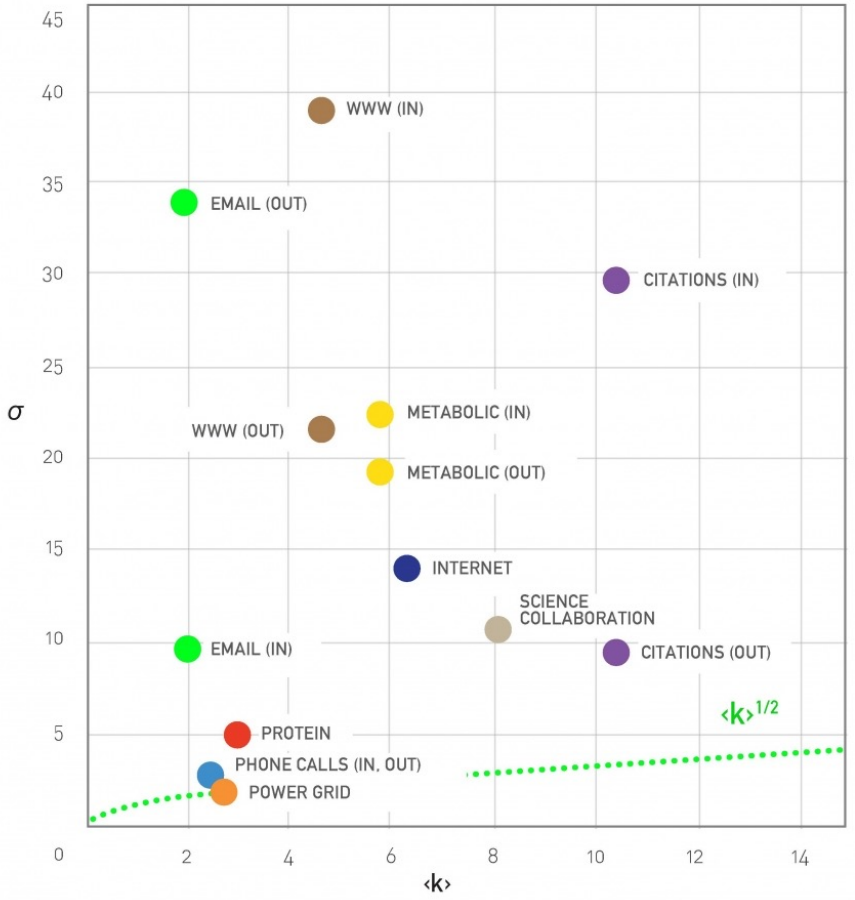


- **Many, but certainly not all, complex networks have been fit to power laws with $2 < \gamma < 4$ (see Barabasi, table 4.1)**

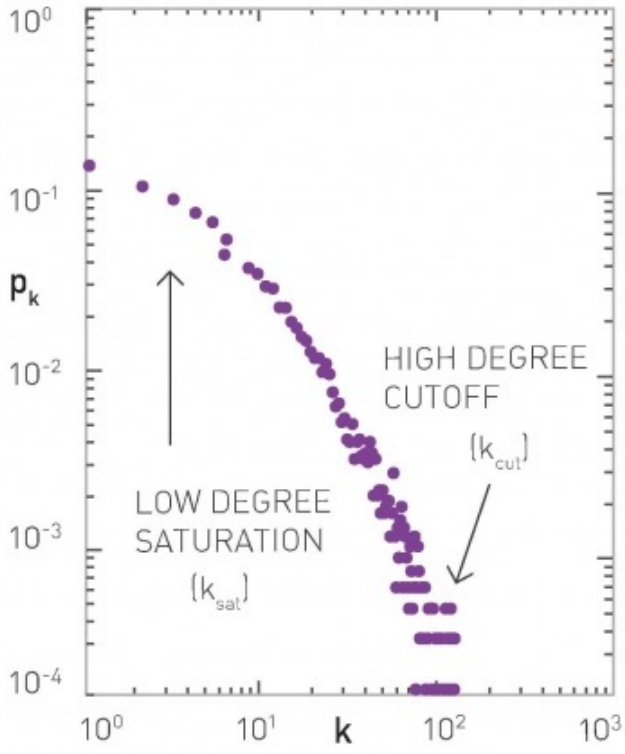
- **However, real networks are finite, so the variance will be large but of course not infinite**

- **This is nicely illustrated in figure 4.8 from Barabasi →**
  - **The standard deviation is shown rather than the variance**
  - **For G_{Np} graphs, $Var(k) = (1-p)\langle k \rangle$, so $\sigma \leq \sqrt{\langle k \rangle}$**

- **Is a large variance important? The configuration model provides some guidance:**
  - $\langle k_{neighbor} \rangle - \overline{k} = \frac{\overline{k^2} - \overline{k}^2}{\overline{k}}$
  - $\langle C \rangle = \frac{\left(\overline{k^2} - \overline{k}\right)^2}{N\,\overline{k}^3}$



- More generally, the standard deviation tells us how important the mean is
- Low variance means that most of the nodes will be similar to the node with the average degree as in G_{Np} graphs
- High variance tells us that the mean carries relatively little information and that there will be a broad range of scales
- In practice, complex networks "have" high variance, but they do not follow a power law across all degrees →
- Small-degree nodes, intermediate-degree nodes, and hubs all play distinct roles in networks



From Barabasi Figure 4.23

- A word of caution: reliably fitting a power-law to network data is not straightforward, and it is generally very difficult to argue that a given distribution follows a power law rather than a log-normal distribution

Preferential attachment

- Is network data consistent with this idea of linear preferential attachment?
- Barabasi et al. "measured" preferential attachment in real networks
  - The idea is to observe a network over a timespan (say, between t_1 and t_2)
  - Typically, links will be added in an irregular manner, so a precise comparison with the linear model is not possible
  - Instead, define S_t as the set of all nodes present at time t_1 and measure L_k, the number of links added to nodes in S_t with degree k.
  - Let L_0 be the total number of links added to nodes in S_t during the observation period, then, linear preferential attachment indicates that we should have $\frac{L_k}{L_0} \approx \frac{k}{2L(t_1)}$ where $L(t_1)$ is the total # of links at t_1

Preferential attachment

- It was observed that results were easier to interpret when a cumulative function was used, $\Pi_\kappa = \sum_{k=1}^{\kappa} L_k / L_0$

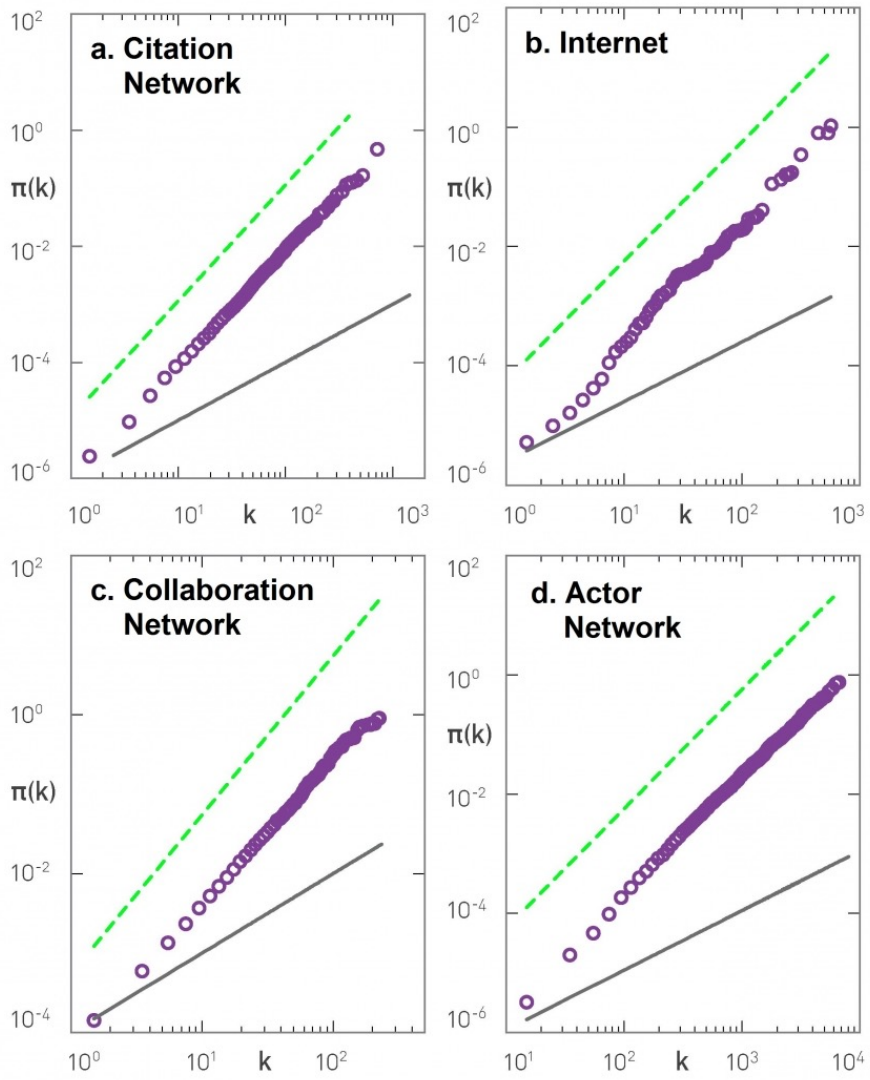
  and linear preferential attachment gives:

$$\Pi_\kappa \approx \sum_{k=1}^{\kappa} \frac{k_i}{2L(t_1)} = \frac{\kappa(\kappa+1)}{4L(t_1)} \sim \kappa^2$$

- So in the figure, measured Π_κ is compared to linear and quadratic trends which correspond to random attachment and linear preferential attachment, respectively

- The networks are fairly close to the expected trend, all show some sort of preferential attachment

Barabasi, figure 5.10



dashed green: k^2, solid black: k

Preferential attachment

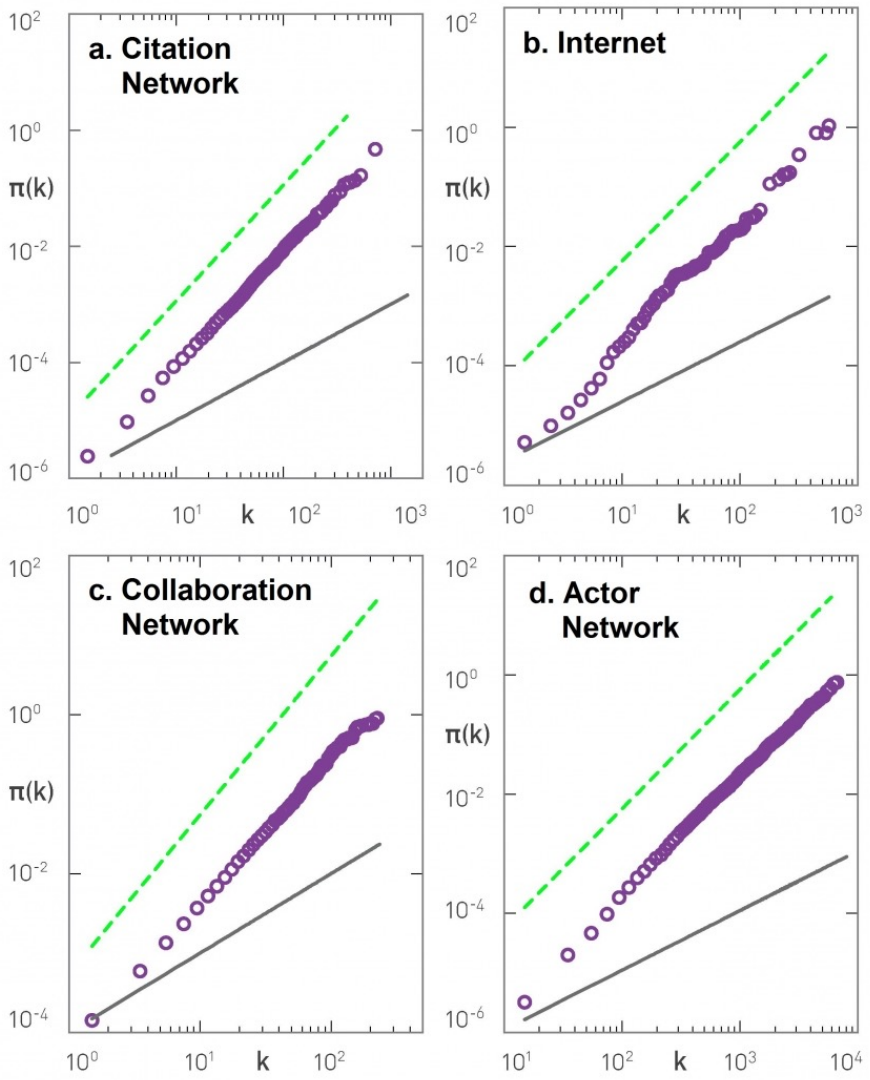
- **Their results suggest that the attachment model should be generalized, e.g., $\rho_i = C(k_i^{\alpha} + \beta)$ where α and β are model parameters and C is a normalization constant**
- **With $\alpha = 1$: $p_k \sim k^{-(3+\frac{\beta}{q})}$ where q is the number of links added per iteration**
- **With $\beta = 0, \alpha < 1$:**

$$p_k \sim k^{-\alpha} \exp(-2\mu(\alpha)/(< k > (1-\alpha)\, k^{1-\alpha})$$

This is the *stretched exponential* distribution (Barabasi §4.10)

- **With $\alpha > 1$, an unrealistic hub-and-spoke structure is generated**
- **This generalization provides an avenue for improving agreement between the model and observations via adjustment of α and β**

Barabasi, figure 5.10



dashed green: k^2, solid black: k

Network Science
Autumn 2021

Lecture 10

Spreading processes on networks
Diffusion in 1D
Diffusion in networks

Spreading processes on networks

- **The dynamics of many interesting and important complex systems can be modeled as spreading processes on complex networks**
- **Picture below: epidemic spreading via global air transportation network**

Other examples:

- **Memes spreading on social networks**
- **Viruses spreading via the internet**
- **Blackouts**

Image from Brockmann & Helbing, The Hidden Geometry of Network-Driven Contagion Phenomena

- How can/should we model these sorts of processes?

- The starting point is to think about diffusion
    - Examples: perfume spreading in room
    - a drop of ink spreading in water
    - thermal energy (heat) spreading through your dinner

    - Diffusion is driven by the seemingly random motion of particles

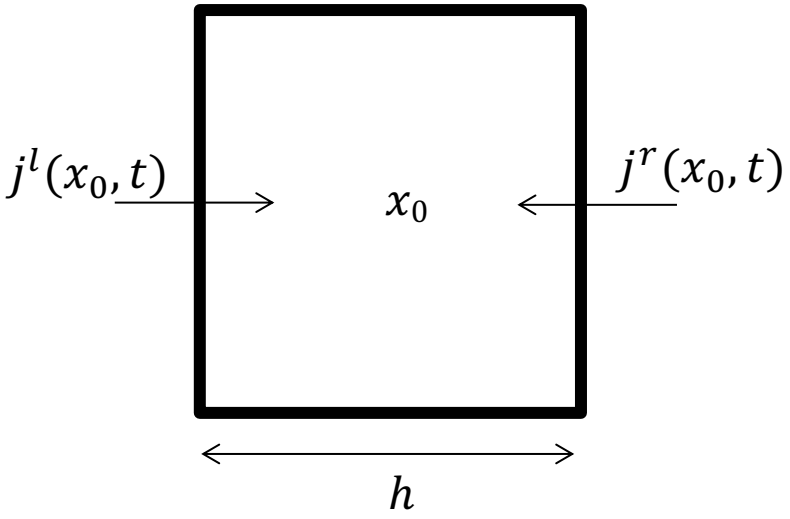
    - Large numbers of collisions between air and perfume molecules “push” some perfume away from its source

    - Let’s model this particle motion using random walks

    - This will lead to the *diffusion equation* which we will then modify to obtain a *graph diffusion equation*

1-D diffusion

- Consider $n(x_0, t)$ particles in the box shown → We will say there are n_{tot} particles distributed across an infinite "line" of such boxes



- How does $\langle n(x_0, t)\rangle$ change in time?

- Each particle is undergoing a random walk defined as follows
  - During a time step from t to $t + \Delta t$, a particle will move a distance $+h$ or $-h$ with equal probability.
  - So all particles inside the box at time t will be in an adjacent box at $t + \Delta t$

- Let $j^r(x_0, t)$ be the number of particles that enter the box crossing its right boundary between t and $t + \Delta t$. Then, what is $\langle j^r(x_0, t)\rangle$? This will depend on the number of particles in the box centered at $x_0 + h$ and the number of these particles that take a step to the left:

$$\langle j^r(x_0,t)\rangle = \sum_{a=0}^{n_{tot}} \sum_{b=0}^{a} P(n(x_0+h,t) = a, j^R(x_0,t) = b) * b$$

- **We know that:**

$$P(n(x_0+h,t)=a, j^r(x_0,t)=b) = P(j^r(x_0,t)=b|n(x_0+h,t)=a)P(n(x_0+h,t)=a)$$

so,

$$\langle j^r(x_0,t)\rangle = \sum_{a=0}^{n_{tot}}\left[P(n(x_0+h,t)=a)\sum_{b=0}^{a}P(j^r(x_0,t)=b|n(x_0+h,t)=a)*b\right]$$

- **The inner sum is straightforward to evaluate. We can think of a step of a random walk as a Bernoulli trial with $p=0.5$. The inner sum is just the expected number of successes from a such trials which is $pa=\frac{a}{2}$ and:**

$$\langle j^r(x_0,t)\rangle = \sum_{a=0}^{n_{tot}}\left[P(n(x_0+h,t)=a)(\frac{a}{2})\right]$$

- **This sum is even more straightforward to simplify: it is simply half of the expected number of particles in the box on the right:**

$$\langle j^r(x_0,t)\rangle = \frac{\langle n(x_0+h,t)\rangle}{2}$$

- **We are almost done now; we need to account for the box on the left and then consider what happens when $\Delta t \to 0$**

- **Using identical arguments as before: $\langle j^l(x_0,t)\rangle = \frac{\langle n(x_0-h,t)\rangle}{2}$,**

$n(x_0, t+\Delta t) = j^l + j^r$ and using linearity of expectation, $\langle n(x_0,t+\Delta t)\rangle = \langle j^l\rangle + \langle j^r\rangle$

- **Our primary interest is in the *change* in the expected number of particles:**

$\langle n(x_0,t+\Delta t)\rangle - \langle n(x_0,t)\rangle = \langle j^l\rangle + \langle j^r\rangle - \langle n(x_0,t)\rangle$ or,

$$\langle n(x_0,t+\Delta t)\rangle - \langle n(x_0,t)\rangle = \frac{\langle n(x_0-h,t)\rangle - \langle n(x_0,t)\rangle}{2} + \frac{\langle n(x_0+h,t)\rangle - \langle n(x_0,t)\rangle}{2}$$

- **Now we will let $\Delta t \to 0$ but note that Δt and h are not independent. Particles move larger distances over larger time steps and empirical observations indicate, $\frac{h^2}{2\Delta t} = \alpha = constant$**

- **The final steps in the derivation of the diffusion equation require the use of Taylor series expansions:**

$$n(x_0, t_0 + \Delta t) = n(x_0, t_0) + \frac{\partial n}{\partial t}\Big|_{x_0,t_0} \Delta t + O(\Delta t^2)$$

$$n(x_0 + h, t_0) = n(x_0, t_0) + \frac{\partial n}{\partial x}|_{x_0,t_0} h + \frac{\partial^2 n}{\partial x^2}|_{x_0,t_0} \frac{h^2}{2} + O(h^3)$$

- **Substituting these expansions (and the analogous expansion for $n(x_0 - h, t_0)$) into our equation for the expectation and working through some arithmetic gives:**

$$\frac{\partial \langle n \rangle}{\partial t} = \alpha \frac{\partial^2 \langle n \rangle}{\partial x^2} + O(\Delta t) + O(h^2)$$

- **Now if we let $h \to 0$ then, the number of particles in a box should also go to zero. Instead, introduce the *particle density*, $\rho(x,t) = \frac{n(x,t)}{h}$. Then, letting $\Delta t \to 0$ and $h \to 0$, gives the 1D *diffusion equation*:**

$$\frac{\partial \rho}{\partial t} = \alpha \frac{\partial^2 \rho}{\partial x^2}$$

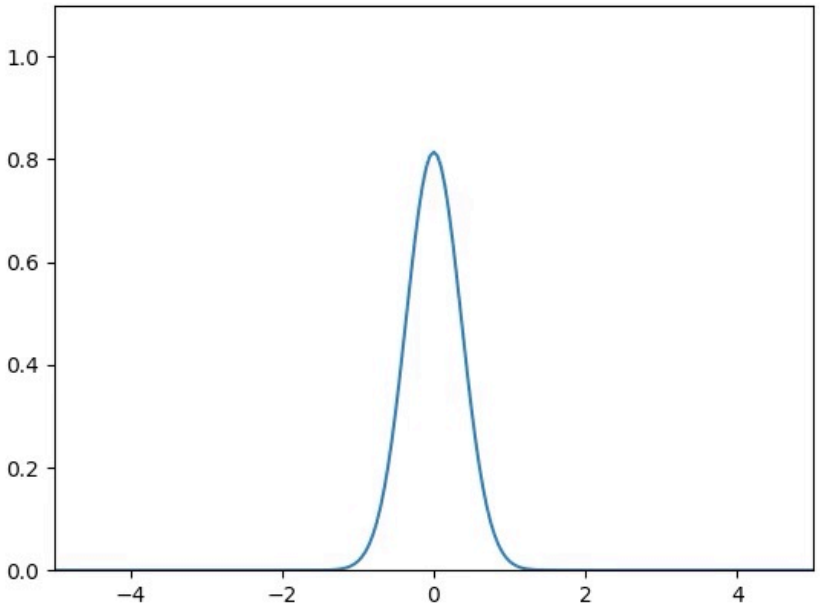
- **We will not delve into the solution of the diffusion equation in any detail**

- **One basic result:**

  The solution of $\partial\rho/\partial t = \alpha\, \partial^2\rho/\partial x^2$ **with** $\rho(x,0) = e^{-\alpha x^2}$ **is:**

$$\rho(x,t) = \frac{1}{\sqrt{1+4\alpha^2 t}} e^{-\frac{\alpha x^2}{1+4\alpha^2 t}}$$

- **This is a "spreading Gaussian":**



Diffusion in networks

- With networks, we no longer have spatial derivatives
- Can we use similar ideas?
- Let's think about the *net* flux of particles along a link between two nodes
- Then the change in $\langle n \rangle$ at a node will be the sum of these fluxes from its neighbors
- So how do we model this flux? Let's go back to our 1-D example:

$$\langle n(x_0, t+\Delta t)\rangle - \langle n(x_0,t)\rangle = \underbrace{\frac{\langle n(x_0-h,t)\rangle - \langle n(x_0,t)\rangle}{2}}_{J^l/2} + \underbrace{\frac{\langle n(x_0+h,t)\rangle - \langle n(x_0,t)\rangle}{2}}_{J^r/2}$$

- Using Taylor series expansions as before, this can be rewritten as,

$$\frac{\partial \langle n \rangle}{\partial t} = \frac{\alpha}{h^2}\left(J^l + J^r\right) + O(\Delta t)$$

- Here, J^l and J^r are the *net* fluxes of particles into the box at x_0 across its boundaries.

- **There is no natural equivalent to h in a graph, so we ignore it and interpret J^l and J^r as the net flux of particles from two *nodes* adjacent to a node at x_0.**

- **Generalizing, for two arbitrary nodes, the net flux of particles from node j to node i will be $J_{ij} = \langle n_j \rangle - \langle n_i \rangle$**

- **And the expected number of particles at node i will satisfy:**

$$\frac{d\langle n_i \rangle}{dt} = \alpha \sum_{j \in \mathrm{N}_i} J_{ij}(t) = \alpha \sum_{j \in \mathrm{N}_i} \left(\langle n_j \rangle - \langle n_i \rangle \right)$$

where the sum is over all neighbors of i and we have taken the limit $\Delta t \to 0$.

The sum can be rewritten using the adjacency matrix of the graph:

$$\frac{d\langle n_i \rangle}{dt} = \alpha \sum_{j=1}^{N} A_{ij} \left(\langle n_j \rangle - \langle n_i \rangle \right)$$ **(graph diffusion equation)**

- **Moving forward, I will use n_i instead of $\langle n_i \rangle$ for convenience.**
- **The graph diffusion equation can be simplified by noticing that: $\sum_{j=1}^{N} A_{ij} n_i = n_i \sum_{j=1}^{N} A_{ij}$**

  and $\sum_{j=1}^{N} A_{ij}$ is k_i , the degree of node i,
- **Now define a diagonal matrix, D, where $D_{ii} = k_i$. Then our graph diffusion equation becomes, $\frac{dn_i}{dt} = -\alpha \sum_{j=1}^{N} L_{ij} n_j$ where $\mathrm{L} = \mathrm{D} - \mathrm{A}$ is the graph *Laplacian matrix***

- **We have our model for diffusion on graphs:** $\frac{dn_i}{dt} = -\alpha \sum_{j=1}^{N} L_{ij} n_j$

- **How do we find solutions?**

- **The initial values need to be specified for each node**

- **Then, this is a system of linear constant-coefficient ODEs**
  - ***i.e.*** **it's an eigenvalue problem**

- **For undirected networks, the Laplacian is real-valued and symmetric**
  - **So its eigenvalues are real (they are also non-negative)**

- **How do we compute eigenvalues in Python?**
  - np.linalg.eig
  - scipy.sparse.linalg.eigs, scipy.sparse.linalg.eigsh

- **You should think about the relative advantages/disadvantages of these functions**

- **An example:**

```
In [3]: G  = nx.erdos_renyi_graph(15,0.2)


In [6]: A = nx.adjacency_matrix(G)

In [7]: A.todense()
Out[7]:
matrix([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0],
        [0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1],
        [0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0],
        [0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0],
        [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1],
        [0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
        [0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]],
```

- **An example:**

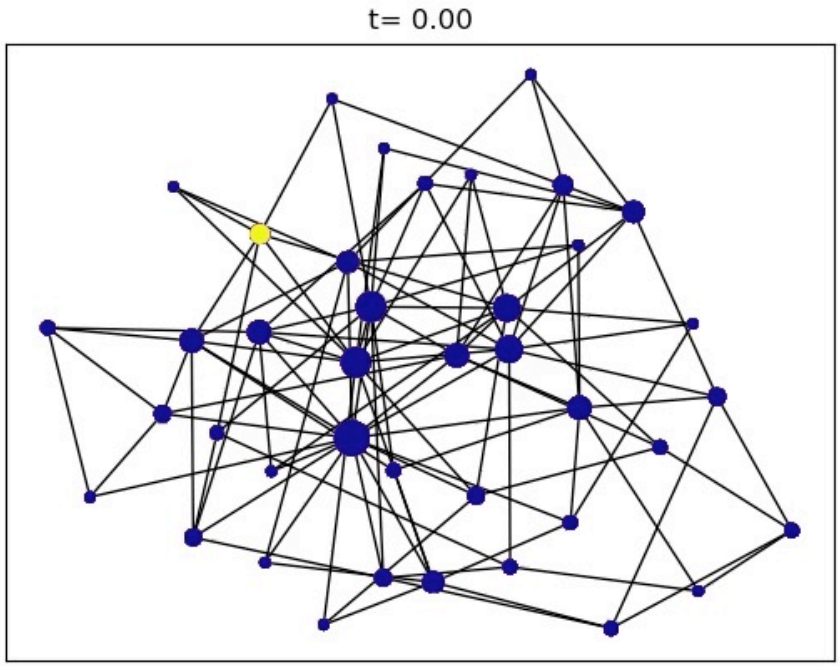
```
In [22]: D = A.toarray().sum(axis=1)

In [23]: L = np.diag(D)-A.toarray()


In [27]: e,v = np.linalg.eig(L)

In [28]: e
Out[28]:
array([8.46780548e+00, 6.22084072e+00, 5.59130087e+00, 4.25530240e+00,
       3.73535212e+00, 1.66002435e-15, 2.55436419e+00, 2.37762994e+00,
       1.87790207e+00, 1.69700500e+00, 1.46423645e+00, 8.66087659e-01,
       8.92173104e-01, 0.00000000e+00, 0.00000000e+00])
```

- **Important questions for you:**
  - **What is the significance of the signs of these eigenvalues?**
  - **There are three eigenvalues with value zero – are these important? What is their physical interpretation?**
  - **What should be done with the eigenvectors?**

Let's consider another example:

- **We'll use a Barabasi-Albert graph with α=1, so: $\frac{dn_i}{dt} = -\sum_{j=1}^{N} L_{ij} n_j$**

- **Initially all nodes have $n_j = 0$ (blue) except one with degree close to the average degree for the graph; for that node, $n_i = 1$(yellow).**

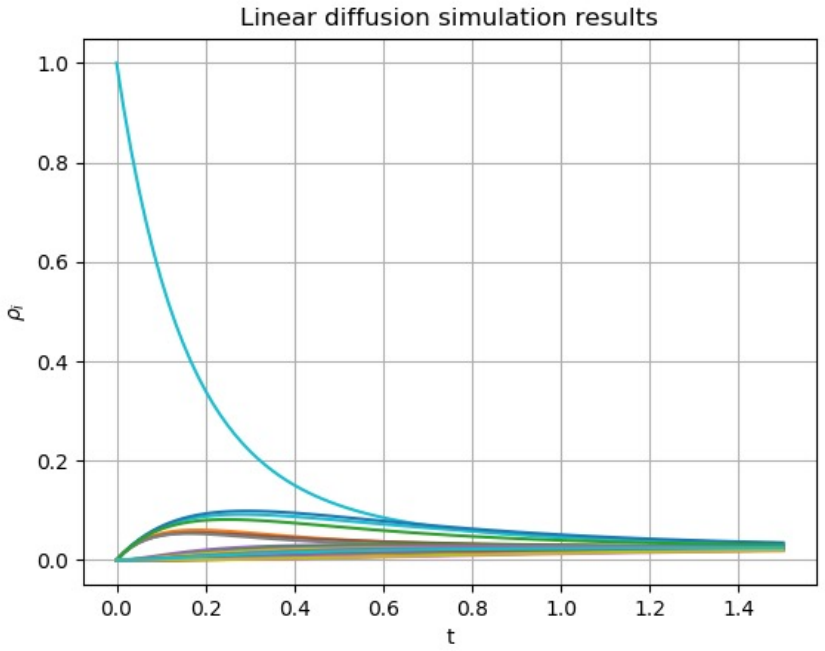


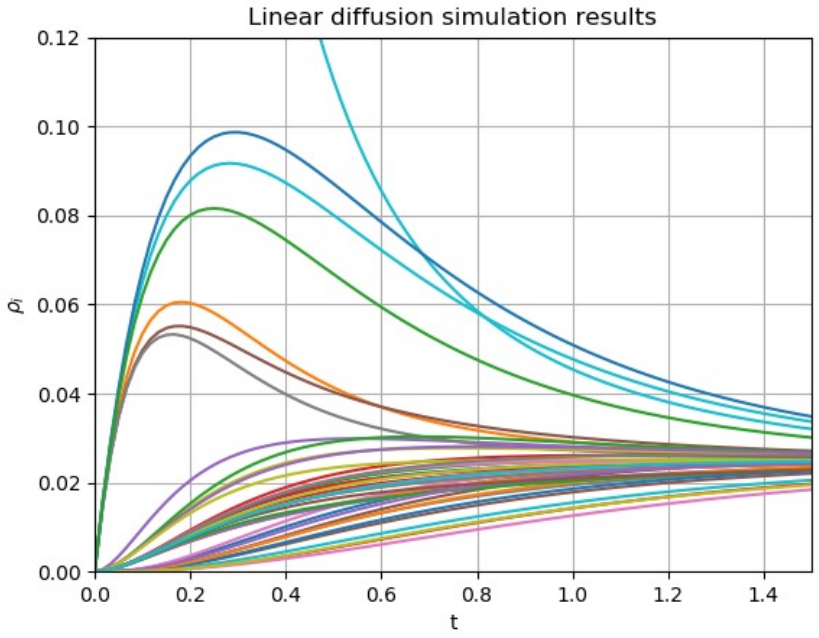
- **The size of each node reflects its degree**

- **If we look carefully at the early stages of the animation, we can clearly see the initial spread from the "yellow" node to its neighbors**

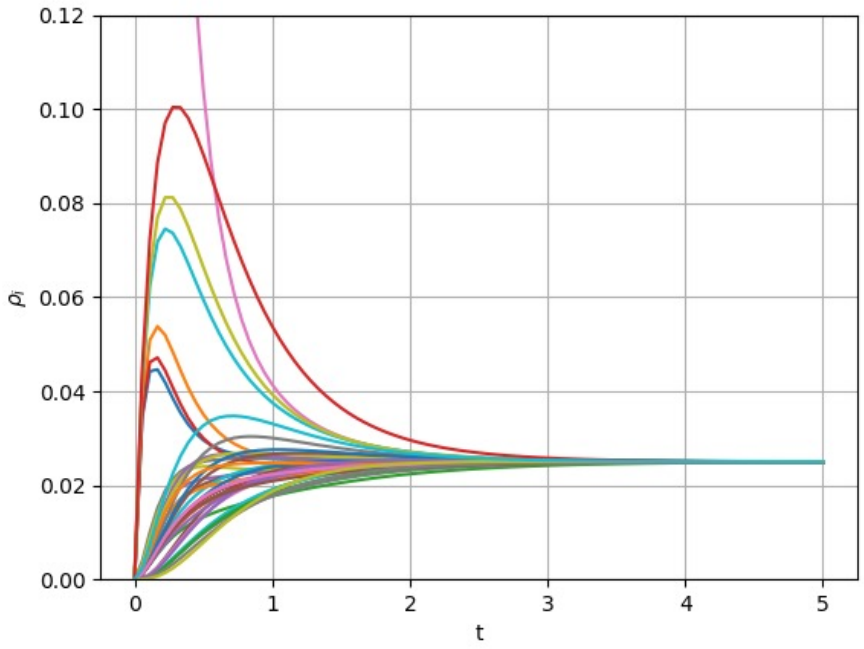
- **At later times, we see all nodes seem to have the same value…**

- **It is also helpful to look at plots of n_i vs $time$, the plots are the same with different limits for the vertical axis – proximity to the "source" node is important!**





- **At later times, we can see clear convergence:**

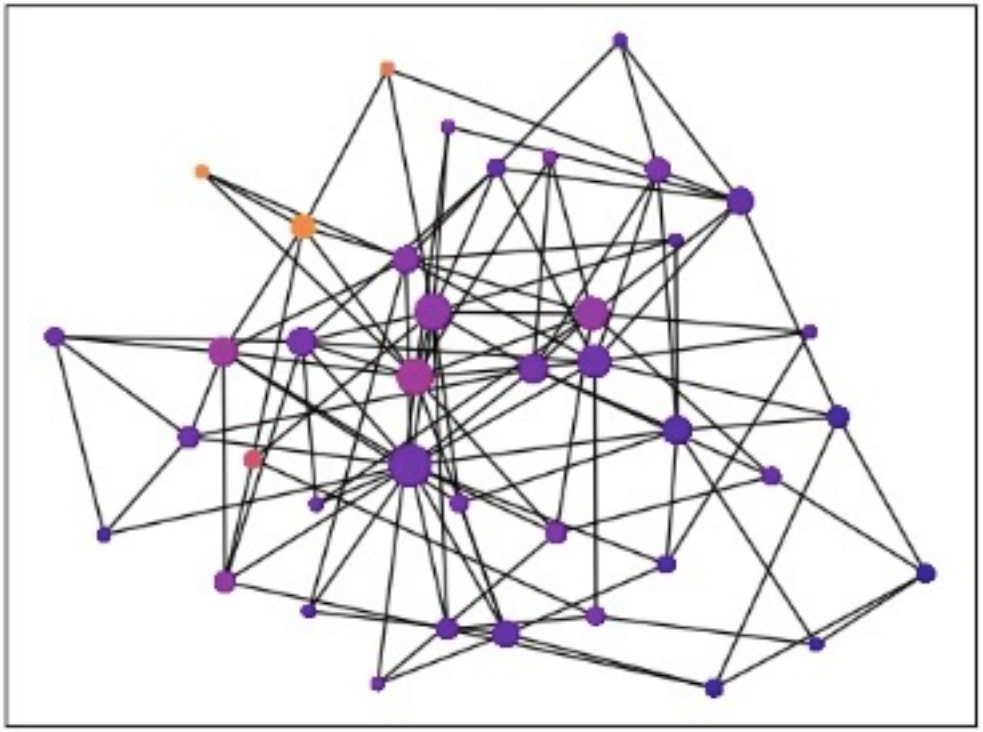


By thinking carefully about the eigenvalues of the Laplacian matrix as well as the eigenvector of the zero eigenvalue(s), you should be able to develop an explanation of this behavior for large times

Lecture 11

Spectral properties of graphs
Synchronization
Random walks on graphs

Spectral properties of graphs



Linear diffusion on a graph

- We will be working with eigenvalues and eigenvectors a fair amount in this part of the module
- Here, we will review some useful ideas/results
  - We will omit the proofs which can mostly be found in standard linear algebra textbooks
- We will also state some results on the bounds of eigenvalues

- Eigenvalues and eigenvectors (and linear algebra more generally) are hugely important in science and engineering applications
  - When you hear terms like stability, control, and optimization, there is a good chance that linear algebra is relevant

- The properties of graph eigenvalues (graph spectra) have also been studied for several decades because mathematicians find them "interesting"

- But as we have seen, eigenvalues (and eigenvectors) of graphs are also important for applications

- When discussing the spectral properties of graphs, we typically focus on the adjacency and Laplacian matrices (A and L)

  - However, there are other important matrices as well such as the "google matrix", G

- **Unless noted otherwise, we will be working with N x N square matrices with real-valued elements ($\mathbb{R}^{N \text{ x } N}$) where N is the number of nodes in the graph of interest**

- **We will also say that $\mathrm{v}_i \in \mathbb{C}^N$ is the i-th eigenvector of B with $\mathrm{B}\mathrm{v}_i = \lambda_i \mathrm{v}_i$, and λ_i is the i-th eigenvalue of B**
  - **Note: if $\lambda_i \neq \lambda_j$ then v_i and v_j are linearly independent**

- **Note for later that the *spectral radius* of a general square matrix B is defined as,**

$$\rho(\mathrm{B}) = \max\{|\lambda_1|, |\lambda_2|, \dots, |\lambda_N|\}$$

- **For undirected networks, both A and L are symmetric, so let's quickly review a few useful properties of symmetric matrices:**

  - **All eigenvalues are real, and eigenvectors can be chosen to be real**

  - **The eigenvectors of the matrix form a basis for $\mathbb{R}^N$ (even if some eigenvalues are repeated)**

  - **A square matrix is orthogonally diagonalizable if and only if it is symmetric**

- **Orthogonal diagonalizability means that a symmetric matrix B can be diagonalized as, $\mathrm{B} = \mathrm{V}\Lambda\mathrm{V}^{\mathrm{T}}$ where:**
  - **V is an orthogonal $N \times N$ matrix whose i-th column is v_i , and $\mathrm{v}_i^T \mathrm{v}_j = 1$ if $i = j$ and $\mathrm{v}_i^T \mathrm{v}_j = 0$ if $i \neq j$ (the eigenvectors are orthonormal, and $\mathrm{V}^{\mathrm{T}}\mathrm{V} = \mathrm{I}$)**
  - **Λ is a diagonal matrix where $\Lambda_{ii} = \lambda_i$**

- **Orthogonal diagonalization is useful when working with differential equations, and it can also be used to derive the following result:**

  - **Order the eigenvalues: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$**
  - **For a symmetric matrix B, the *Rayleigh quotient,* $r(\mathrm{B}, \mathrm{x}) = \frac{\mathrm{x}^{\mathrm{T}}\mathrm{Bx}}{\mathrm{x}^{\mathrm{T}}\mathrm{x}}$, is maximized when $\mathrm{x} = \mathrm{v}_1$ in which case $\frac{\mathrm{x}^{\mathrm{T}}\mathrm{Bx}}{\mathrm{x}^{\mathrm{T}}\mathrm{x}} = \lambda_1$ (to show this, you can orthogonally diagonalize B and show that λ_1 is an upper bound for r)**

  - **It may not seem like it, but this is an extremely useful result. We will use it here to develop bounds for λ_1 and use it again later when analyzing communities**

- **We now narrow our focus to finding bounds for graph eigenvalues**

- ***Gershgorin's theorem* can be used to develop an upper bound for the spectral radius of a square matrix:**

  Let $\mathrm{B} \in \mathbb{C}^{N \times N}$ **and suppose that** $\mathrm{X}^{-1}\mathrm{BX} = \mathrm{H} + \mathrm{F}$ **where** H **is diagonal and** F **has zeros on its main diagonal. Then, the eigenvalues of** B **lie on the union of the discs** $\Delta_1, \Delta_2, \dots, \Delta_N$ **where** $\Delta_i = \left\{ l \in \mathbb{C}: |l - H_{ii}| \leq \sum_{j=1}^{N} |F_{ij}| \right\}$

  - **Frequently, we simply choose** X **to be the identity matrix**

- **Let's apply this theorem to the adjacency matrix for a simple undirected graph with** $\mathrm{X} = \mathrm{I}$

- **Then,** $\mathrm{F} = \mathrm{A}$ **and the** i***-th*** **disc is defined by:** $|l| \leq \sum_{j=1}^{N} A_{ij} = k_i$

- **The largest possible radius of a disc is** k_{max} **and it follows that** $\rho(\mathrm{A}) \leq k_{max}$

- **We can develop a lower bound for λ_1 using our earlier result (in a slightly different form):** $\frac{\mathrm{x}^{\mathrm{T}}\mathrm{Ax}}{\mathrm{x}^{\mathrm{T}}\mathrm{x}} \leq \lambda_1$
- **If we choose $\mathrm{x} = \mathrm{z}$ (where z is an N-element column vector of ones), we then find, $\lambda_1 \geq \frac{K}{N} = \overline{k}$ and since $\lambda_1 \leq \rho(\mathrm{A})$, we know that: $\overline{k} \leq \lambda_1 \leq k_{max}$**
- **This is a pretty nice result!**
  - **And the upper bound can be used when setting the proportionality constant for the Katz centrality**
  - **We will use it next week when studying the spread of infectious diseases on networks**

- **The *Laplacian matrix* is defined as $\mathrm{L} = \mathrm{D} - \mathrm{A}$ where D is the diagonal degree matrix, $D_{ii} = k_i$**

- **We have seen that the Laplacian arises naturally when considering diffusive processes, we will later consider how it connects to graph partitioning**

- **A few notes on the eigenvalues of L for undirected graphs:**

  - **Using a similar approach to what we used for A, we find, $k_{max} \leq \lambda_1 \leq 2k_{max}$**

  - **We can show that all eigenvalues are non-negative and that L has at least one zero eigenvalue with eigenvector z (problem sheet exercises)**

  - **These results taken together are helpful for understanding graph diffusion)**

Synchronization

- **The graph diffusion equation provides a simple linear *coupling* between linked nodes:**
  $\frac{dn_i}{dt} = \alpha \sum_{j=1}^{N} A_{ij}(n_j - n_i)$ **or more compactly:** $\frac{d\mathrm{n}}{dt} = -\alpha \mathrm{Ln}$

- **Coupling of this form and in more complicated nonlinear forms can lead to *synchronization*.**

- **Real-world examples of synchronization:**
  - **An audience clapping:** https://youtu.be/Au5tGPPcPus?t=42
  - **Fireflies glowing:** https://youtu.be/QCWkzQqO7Ro
  - **Neurons activating in the human brain**
  - **Muscles flexing in the human heart**

- **We'll model the different elements in a system as a network of interacting nodes. Links will indicate where there are interactions, and we will have to decide how to model the interactions.**

- **The goal here is to analyze a simple model related to synchronization, but keep in mind that this is an active area of research where more-sophisticated models are typically used.**

- **Let's say that there is an "oscillator" on each node of a connected undirected graph with the oscillation on node i described by, $x_i = a\ cos\big(\theta_i(t)\big)$.**

- **So each oscillator is assumed to have the same amplitude of oscillation (a), and we will focus on the *phase*, $\theta_i(t)$, which has initial condition $\theta_i(t=0) = \theta_{0,i}$**

- **We'll assume that the phase of each oscillator is influenced by two effects: 1) a natural frequency, ω, and 2) coupling with other linked oscillators with general form, $f(\theta_j - \theta_i)$**

- **These assumptions lead to the following phase equation:**

$$\frac{d\theta_i}{dt} = \omega + \alpha \sum_{j=1}^{N} A_{ij}\, f\big(\theta_j - \theta_i\big); \quad f(0) = 0$$

  - **Note that there are other reasonable assumptions we could have made, e.g. each oscillator could have a distinct natural frequency, ω_i.**

  - **Here, α sets the relative importance of the coupling**

- **What happens when $\alpha = 0$? Then, $\theta_i = \omega t + \theta_{0,i}$ and each oscillator oscillates with the same frequency, but depending on the initial condition, the oscillators may be out of phase**

- **What should the coupling function be? In principle, this should be guided by the physical system being investigated**

- **A popular choice is to use sinusoidal coupling,** $f(\theta_j - \theta_i) = \sin(\theta_j - \theta_i)$

- **We will take a simpler approach. Let** $h_{ij} = \theta_j - \theta_i$**, and assume that** $|h_{ij}| \ll 1$ **then** $f(h_{ij}) \approx f(0) + h_{ij} \frac{df}{dh_{ij}}|_{h_{ij}=0}$ **and we have required** $f(0) = 0$**. So, assuming that** $\frac{df}{dh_{ij}}|_{h_{ij}=0} \neq 0,$ **our model becomes:**

$$\frac{d\theta_i}{dt} = \omega + \alpha \sum_{j=1}^{N} \frac{df}{dh_{ij}}\Big|_{h_{ij}=0} A_{ij} (\theta_j - \theta_i)$$

- **Finally, we assume that** $\frac{df}{dh_{ij}}|_{h_{ij}=0}$ **is the same constant for each** (i, j) **and its effect can be "absorbed" into** α **giving a model very similar to graph diffusion:**

$$\frac{d\theta_i}{dt} = \omega + \alpha \sum_{j=1}^{N} A_{ij}(\theta_j - \theta_i)$$

- **So, our (very simple) model for a system of coupled oscillators in matrix-vector form is,** $\frac{d\theta}{dt} = \omega z - \alpha L\theta$ **with** $\theta, z \in \mathbb{R}^N$

- **The question we want to consider is if/when the system will be fully synchronized, i.e.** $|\theta_i - \theta_j| = 0$ **for all distinct pairs of oscillators (assuming that the initial phases are not all the same)**

- **There are a few different ways to approach this. We will simply find the general solution and then consider** $\theta_i - \theta_j$ **when** $t \to \infty$

- **Step 1: Orthogonally diagonalize the Laplacian:** $L = V\Lambda V^T$
- **Step 2: Left-multiply both sides of the model equation with** V^T**:** $\frac{d(V^T\theta)}{dt} = \omega V^T z - \alpha\Lambda V^T\theta$ **where we have used** $V^T V = I$
- **Step 3: define** $w = V^T\theta$**:** $\frac{dw}{dt} = \omega V^T z - \alpha\Lambda w$

- **Step 4: simplify $\mathrm{V}^{\mathrm{T}}\mathrm{z}$. Assume that the first column of V contains the eigenvector $\frac{\mathrm{z}}{\sqrt{N}}$ which is orthogonal to all other columns in V. We then also have $\Lambda_{11} = 0$. So the inner product of the first row of V^{T} and z will be $\sqrt{N}$ and the inner product of all other rows with z will be zero: $\mathrm{V}^{\mathrm{T}}\mathrm{z} = \sqrt{N}\mathrm{e}_1$ where $\mathrm{e}_1 = [1, 0, 0, \dots, 0]^{\mathrm{T}}$. The model equation is now:**

$$\frac{d\mathrm{w}}{dt} = \omega\sqrt{N}\mathrm{e}_1 - \alpha\Lambda\mathrm{w}$$

- **Step 5: solve the system of equations above:**

$$w_1 = \omega t\sqrt{N} + \frac{\mathrm{z}^{\mathrm{T}}\theta_0}{\sqrt{N}}$$

$$w_i = \exp(-\alpha\Lambda_{ii}t)\,\mathrm{v}_i^{\mathrm{T}}\theta_0,\ i > 1$$

Here, w_i is the i-th element in w, θ_0 is the vector of initial conditions for the phase, and v_i is the i-th column in V.

Check your understanding: why does the solution for w_1 not have an exponential term?

- **Final steps: find the solution for θ_i and examine $\theta_i - \theta_j$. We know $\mathrm{w} = \mathrm{V}^{\mathrm{T}}\theta$. Using the orthogonality of V, we find, $\theta = \mathrm{V}\mathrm{w}$, so:**

$$\theta = \mathrm{v}_1 w_1 + \mathrm{v}_2 w_2 + \cdots + \mathrm{v}_N w_N$$

- **All eigenvalues for the Laplacian are non-negative, and for a connected graph, only one eigenvalue is zero (we will discuss this in more detail later)**

- **So, as $t \to \infty$, $w_i \to 0$ for $i > 1$ assuming that the graph is connected and $\alpha > 0$.**

- **In this long-time limit, we see that $w_1 \to \infty$, but what we are interested in is, $\theta_i - \theta_j$. We know that $\mathrm{v}_1 = \frac{\mathrm{z}}{\sqrt{N}}$ so $\theta_i \to w_1 t + w_{0,1}$ for each i and $\theta_i - \theta_j \to 0$ for all node pairs.**

- **The system synchronizes at long times even if the initial phases are all different. This is a simple illustration of how the "smoothing" effect of diffusion can produced synchronization on a network. Current research works with more complicated versions of this model. For example we could have: nonlinear coupling, a distribution of frequencies, a distribution of amplitudes, external noise…**

Random walks on graphs

- Previously, we considered random walks on a line and derived the 1-D diffusion equation
- An intermediate step in this derivation required consideration of the net flux of particles across the boundary of a box, and when we move to graphs, we chose the flux per unit time between nodes on a graph to be: $J_{ij}(t) = \alpha \left(n_j(t) - n_i(t) \right)$ -- this represents the tendency for transport from nodes with high numbers of particles to neighbors with low numbers
- But what if we directly modeled random walks of particles moving from one node to another? Would the resulting dynamics be similar to what we have seen with "graph diffusion" as I defined it?

- **Say a particle's position in a graph at time t is $x(t)$ with $x \in \{1,2,\dots,N\}$. One step of a random walk from t to $t+\Delta t$ on a graph is then defined as follows: *A particle will select a link on node $x(t)$ uniformly at random and follow it to a neighbor of $x(t)$***

- **We will restrict ourselves to undirected connected graphs for now, but will consider directed graphs a little later**

- **The probability that a particle moves from node i to j is the *transition probability,* $T_{ij} = \frac{A_{ij}}{k_i}$**

- **It is convenient to analyze random walks on graphs (RWGs) in terms of the probability that a particle is at node i at time t: $P(x(t) = i)$**

- **The probability that a particle is at node i at time t and is at node j at $t+\Delta t$ is:**

$$P(x(t+\Delta t) = j, x(t) = i) = P(x(t+\Delta t) = j | x(t) = i)\, P(x(t) = i)$$ **or**

$$P(x(t+\Delta t) = j, x(t) = i) = T_{ij} P(x(t) = i)$$

- **Then noting that** $P(x(t+\Delta t)=j)=\sum_{i=1}^{N}P(x(t+\Delta t)=j,x(t)=i)$**, we can write down the master equation for RWGs:**

$$P(x(t+\Delta t)=j)=\sum_{i=1}^{N}\frac{P(x(t)=i)A_{ij}}{k_i}$$

- **Let's now write this as a difference equation in matrix-vector form:**
  - **Let** $t_l=l\Delta t, l=0,1,2,\ldots$
  - **And say that** $\mathrm{p}^{(l)}$ **is an** N**-element row vector whose** i**-th element is:** $P(x(t=l\Delta t)=i)$.

- **Then, the master equation becomes,**

$$\mathrm{p}^{(l+1)}=\mathrm{p}^{(l)}\mathrm{D}^{-1}\mathrm{A}=\mathrm{p}^{(l)}\mathrm{T}$$

where D **is the usual diagonal degree matrix and** $\mathrm{T}=\mathrm{D}^{-1}\mathrm{A}$ **is the *transition matrix***

- **Our analysis of this equation will examine (1) the stationary state for this system and (2) the "relaxation" to this state**

- **A stationary state, p_∞, exists if there is a non-trivial solution where $\mathrm{p}^{(l+1)} = \mathrm{p}^{(l)} = \mathrm{p}_\infty$ or equivalently, $\mathrm{p}_\infty = \mathrm{p}_\infty \mathrm{T}$**

- **This is quite similar to the equation we had for the PageRank centrality, and we can again establish that $\lambda = 1$ will be an eigenvalue:**

  - **Since $\sum_{j=1}^{N} T_{ij} = \sum_{j=1}^{N} \frac{A_{ij}}{k_i} = 1$, we have $\mathrm{Tz} = \mathrm{z}$ where z is the usual column vector of ones**

  - **Combined with Perron-Frobenius (version 2), this also tells us that $\rho(\mathrm{T}) = 1$**

- **We know that p_∞ is the left eigenvector for T corresponding to this eigenvalue, and we can figure out what it will be by writing out the equation for its first element:**

$$p_{\infty,1} = \frac{p_{\infty,2}A_{21}}{k_2} + \frac{p_{\infty,3}A_{31}}{k_3} + \cdots + \frac{p_{\infty,N}A_{N1}}{k_N}$$

and then noticing that the equation will be satisfied if we choose, $p_{\infty,i} = k_i$. The same thing will happen for the other nodes, and then after normalizing, the solution for the stationary state is, $\boxed{p_{\infty,i} = k_i/K}$

- **This stationary state is very different from what we had for graph diffusion! There, at long times, each node tends to have the same density. With RWGs, the stationary state suggests a particle is more likely to end up on a node with high degree.**

- **The next part of our analysis will consider the evolution of p in time, and we will determine if the solution moves to this stationary state.**

- Let's now consider the evolution of $\mathrm{p}^{(l)}$ and check if/how the probability vector approaches this stationary state

- We need to solve, $\mathrm{p}^{(l+1)} = \mathrm{p}^{(l)}\mathrm{D}^{-1}\mathrm{A} = \mathrm{p}^{(l)}\mathrm{T}$ or,

$\mathrm{p}^{(l)} = \mathrm{p}^{(0)}\,\mathrm{T}^{l}$ where $\mathrm{p}^{(0)}$ is the specified initial condition

- Now, T is not symmetric, however we can use a similarity transformation to produce an equation with a symmetric operator that is relatively straightforward to solve

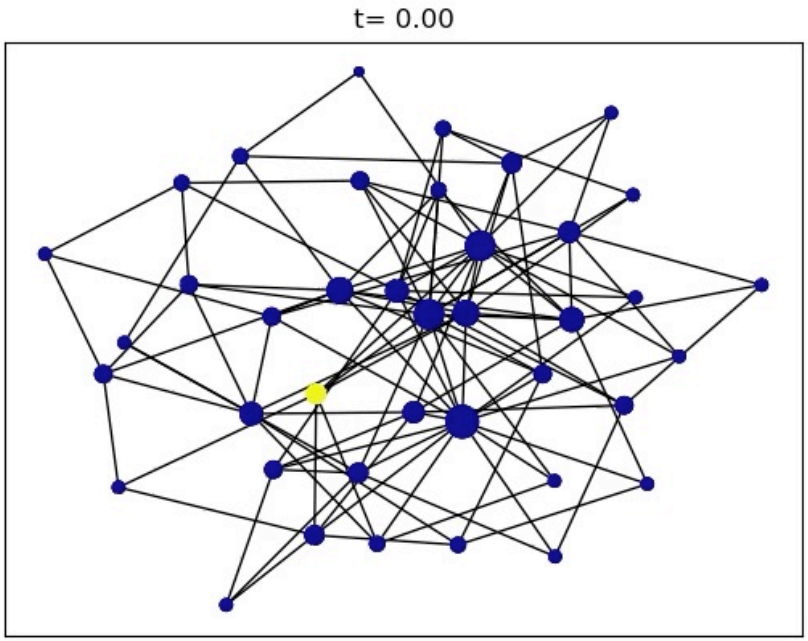
- First, notice that $\mathrm{D}^{1/2}\mathrm{T}(\mathrm{D}^{1/2})^{-1} = (\mathrm{D}^{1/2})^{-1}\,\mathrm{A}\,(\mathrm{D}^{1/2})^{-1}$ and the matrix on the RHS is symmetric (using index notation, the RHS is $\frac{A_{ij}}{\sqrt{k_i k_j}}$)

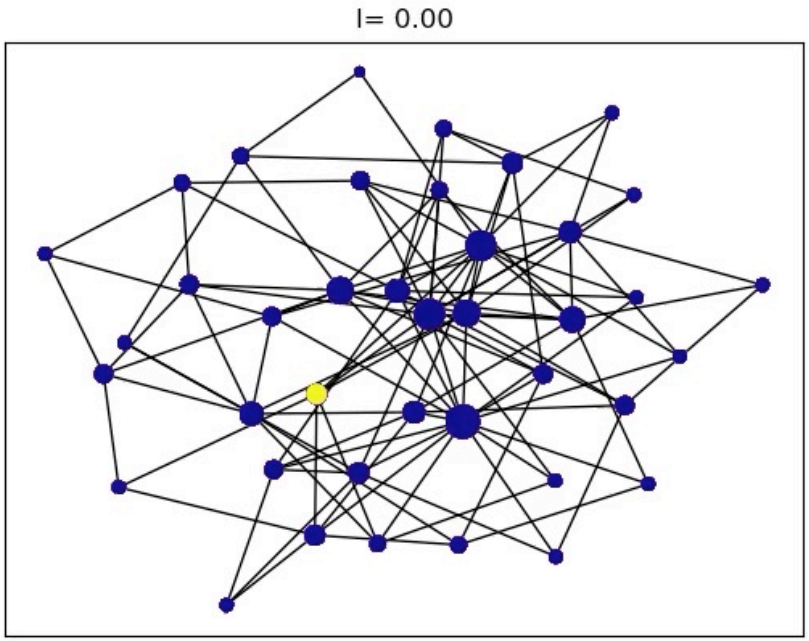
- The goal is to introduce a transformation of p which leads to a system governed by this symmetric matrix

- **The desired transformation is,** $\mathrm{w}^{(l)} = \mathrm{p}^{(l)}(\mathrm{D}^{1/2})^{-1}$
- **This gives,** $\mathrm{w}^{(l+1)} = \mathrm{w}^{(l)}\,\widetilde{\mathrm{T}}$ **with** $\widetilde{\mathrm{T}} = (\mathrm{D}^{1/2})^{-1}\,\mathrm{A}\,(\mathrm{D}^{1/2})^{-1}$
- **We then orthogonally diagonalize** $\widetilde{\mathrm{T}}$**:** $\widetilde{\mathrm{T}} = \mathrm{V}\Lambda\mathrm{V}^{\mathrm{T}}$**, to obtain,** $\mathrm{w}^{(l+1)} = \mathrm{w}^{(l)}\,\mathrm{V}\Lambda\mathrm{V}^{\mathrm{T}}$
- **Then one more transformation,** $\mathrm{y}^{(l)} = \mathrm{w}^{(l)}\mathrm{V}$**, gives us the decoupled system of equations,** $\mathrm{y}^{(l+1)} = \mathrm{y}^{(l)}\Lambda$**, and since** Λ **is diagonal:** $\mathrm{y}^{(l)} = \mathrm{y}^{(0)}\Lambda^{l}$**. We now need to think about the (real) eigenvalues of** $\widetilde{\mathrm{T}}$
- **A few notes:**
  - **If there are eigenvalues with magnitude larger than 1, then** $|\mathrm{y}^{(l)}|$ **will grow exponentially**
  - $\sum_{j=1}^{N} \frac{A_{ij}}{\sqrt{k_i k_j}}\sqrt{k_j} = \sqrt{k_i}$ **so** $\left[\sqrt{k_1}, \sqrt{k_2}, \ldots, \sqrt{k_N}\right]^{T}$ **is an eigenvector of** $\widetilde{\mathrm{T}}$ **with eigenvalue,** 1
  - **Then for a connected graph, we can apply P-F version 2, we know the spectral radius is 1,** $\rho\left(\widetilde{\mathrm{T}}\right) = 1$**, and that the eigenvalue** λ**=1 is simple and strictly larger than all other eigenvalues**

- So we know that $-1 \leq \lambda_i \leq 1 \; \forall i$ and there will not be exponential growth
- We will assume that the eigenvalues are strictly larger than -1. Aside from particular classes of multipartite graphs, this assumption is correct.
- Let's say that $\lambda_1 = 1$ with orthonormal eigenvector $\mathrm{v}_1 = \frac{1}{\sqrt{K}}\left[\sqrt{k_1}, \sqrt{k_2}, \ldots, \sqrt{k_N}\right]^T$.
- Then for sufficiently large l: $\mathrm{y}^{(l)} \approx \left[y_1^{(0)}, 0, 0, \ldots, 0\right]$
- And going "backwards" from y to p: we have $\mathrm{w}^{(l)} = y_1^{(0)} \, \mathrm{v}_1^T$ and $\mathrm{p}^{(l)} = \mathrm{w}^{(l)}\mathrm{D}^{\frac{1}{2}} = \frac{y_1^{(0)}\kappa}{\sqrt{K}}$ where $\kappa = [k_1, k_2, \ldots, k_N]$ and l is large
- By inspection, we can see that we would like $y_1^{(0)} = \frac{1}{\sqrt{K}}$ and if we use our transformations to relate $y_1^{(0)}$ to $\mathrm{p}^{(0)}$ we find that this is indeed the case if $\sum_{i=1}^{N} p_i^{(0)} = 1$ which is necessary for the problem to be well-posed.
- And we see that the probability vector will evolve towards the stationary distribution

- **Consider the initial condition where a walker is placed on node x so, $p_x^{(0)} = 1$**
- **Initially this probability will "spread" from node x to its neighbors and eventually to the entire (connected) graph**
- **Ultimately exponential decay will "win" and the probability will relax towards its stationary state**





Graph diffusion

RWG

- **Comparing RWGs to graph diffusion we see a basic similarity – exponential relaxation towards an equilibrium/stationary distribution – and a substantive difference, the RWG stationary state has a degree-dependence.**

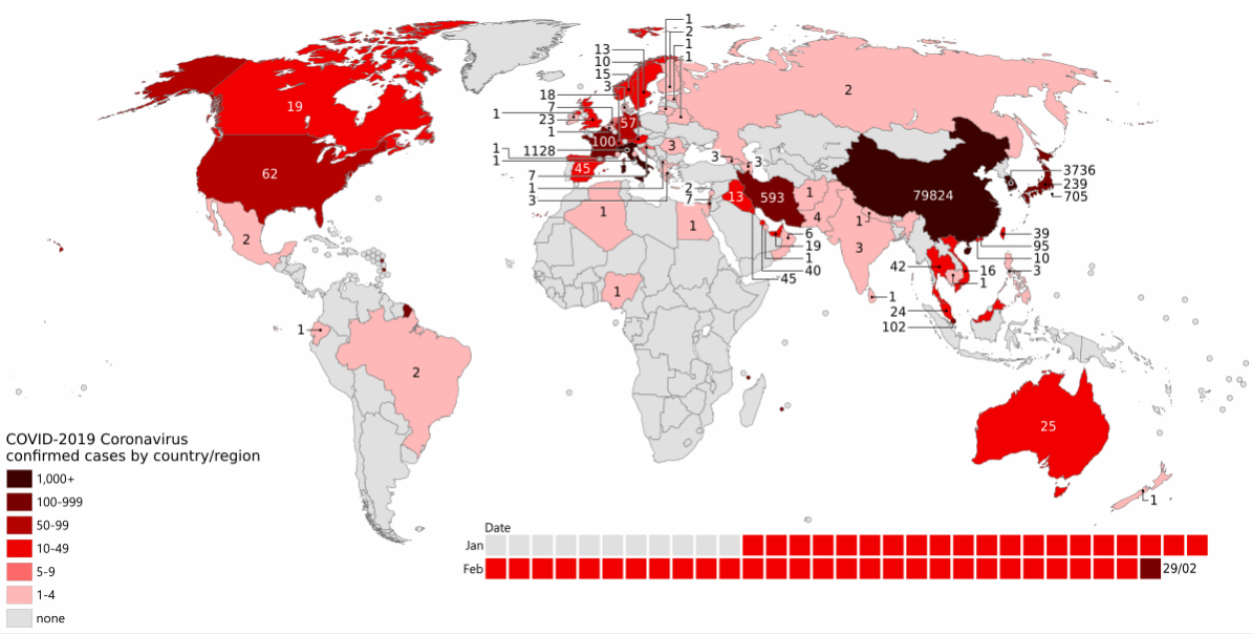
- **So which model is better? It depends on what you are modeling! Graph diffusion is a natural choice if synchronization (or consensus of opinion) is possible while RWGs (with minor modification) naturally model a search engine moving through the web and occasionally "teleporting" to another part of the web (see problem sheet 6)**

Lecture 12

Epidemics on networks I

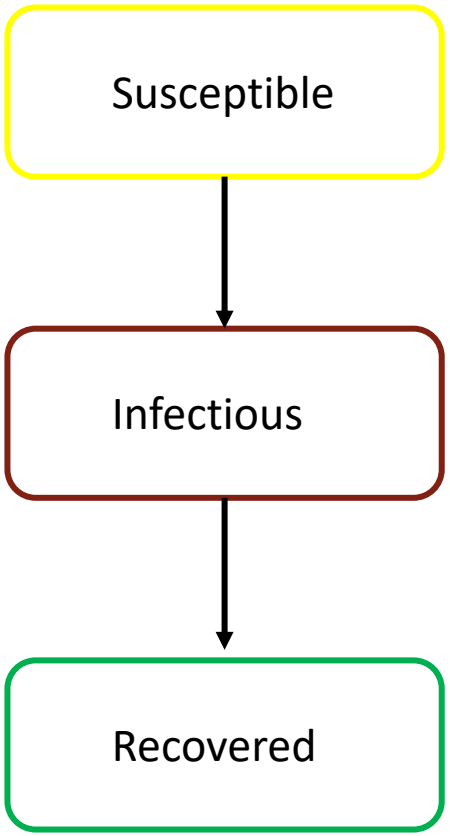
Background

- **Picture below: confirmed Covid cases by country, 29/2/20**
- **Accurate predictions of the spread of Covid within and between countries could have saved countless lives. Unfortunately, dubious models and reasoning were widely used in the early stages of the pandemic.**

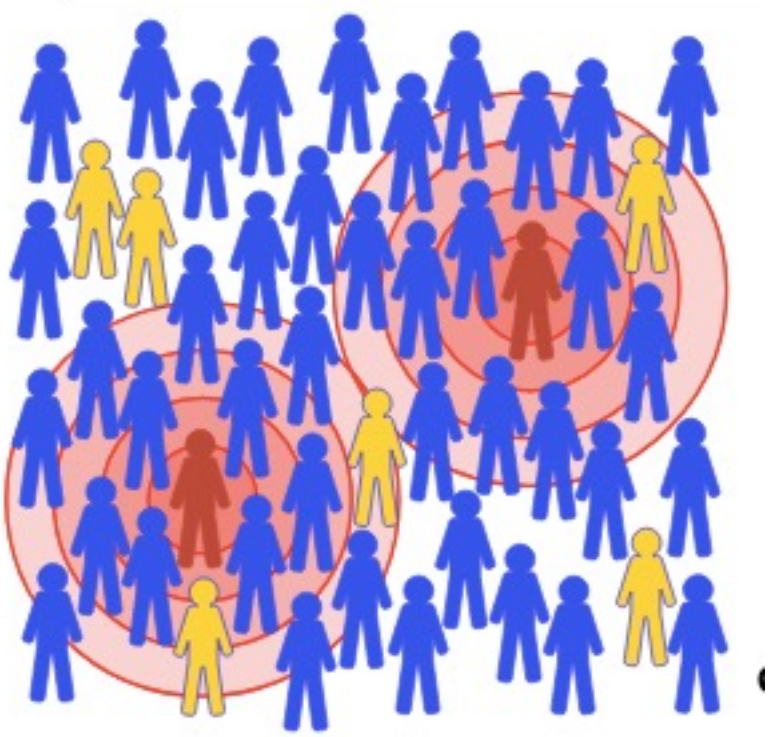


https://commons.wikimedia.org/wiki/File:COVID-19-outbreak-timeline.gif#file

- **"Compartment models" are widely used as a 1st step for understanding/simulating/predicting spread of infectious diseases**
- **The progress of a disease is characterized by distinct states, for example S → I → R indicates that a Susceptible person can become Infected (and contagious) and will then Recover with immunity**
- **There is a "zoo" of such models with various states connecting to each other in various ways (SI, SIR, SIS, SEIR, MSEIR, …)**
- **We will focus on the simplest of these: the SI model.**



- Rather than think about the detailed mechanisms of infection, we characterize a disease with a probability of infection and an expected rate of recovery
- With the SI model, there is no recovery: an infected person remains infected
- We start by considering these models and disease spread in a community



- There are simpler approaches which ignore social networks and simply model the fraction of people in a a community that are infected (and how that fraction evolves in time)

- However, these models assume that every susceptible individual is equally likely to contract the disease from an infectious person

- Like most complex networks, large social networks typically have multiscale degree distributions, so some people will come into contact with many others and could potentially be “superspreaders” which we know are important:

NEWSLETTERS
Sign up to read our regular email newsletters

NewScientist

News Podcasts Video Technology Space Physics Health More Shop Tours Events Jo

Finding coronavirus superspreaders may be key to halting a second wave

HEALTH 30 July 2020

By Clare Wilson

Network SI model

So how do we proceed?

- **We place individuals at the vertices of an N-node graph (with adjacency matrix, A) and place undirected links between people who are in "regular" contact with each other**
    - **We could place weights on the links indicating the "significance" of these contacts, but we will treat each link as the same here**
- **For simplicity we will assume there are 2 states, susceptible and infected**
- **The model parameter β determines how easily a disease is transmitted:**
    - $\beta\Delta t$: **Probability that a susceptible person is infected via a link to an infectious person over a timespan of duration, Δt**

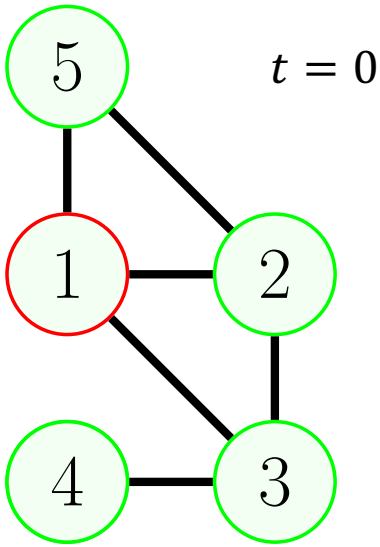
- **A very simple example – consider the 5 node network shown with node 1 initially infectious and all other nodes susceptible**

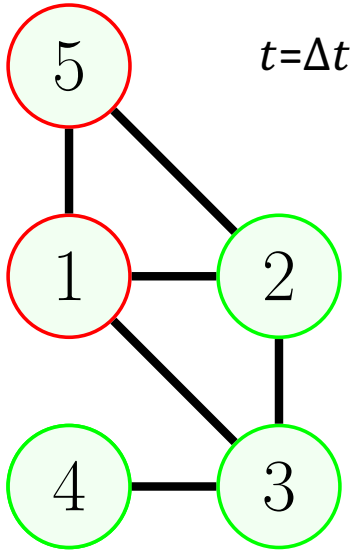
- **A node that become infectious always stays infectious**

- **Moving forward 1 time step, Δt, we carry out a Bernoulli trial for each link to node 1 with probability $\beta\Delta t$**
  - **Let's say $\beta\Delta t = 0.25$ and we generate three random numbers for links 1-2,1-3,1-5:**
    np.random.rand(3)
    Out[45]: array([0.39142964, 0.28799901, 0.20457327])
  - **So there will be transmission along link 1-5 →**
  - **Now, we generate random numbers for 1-2,1-3,5-2:**

    np.random.rand(3)
    Out[46]: array([0.81424296, 0.04301428, 0.45276639])

    and there will be transmission along link 1-3

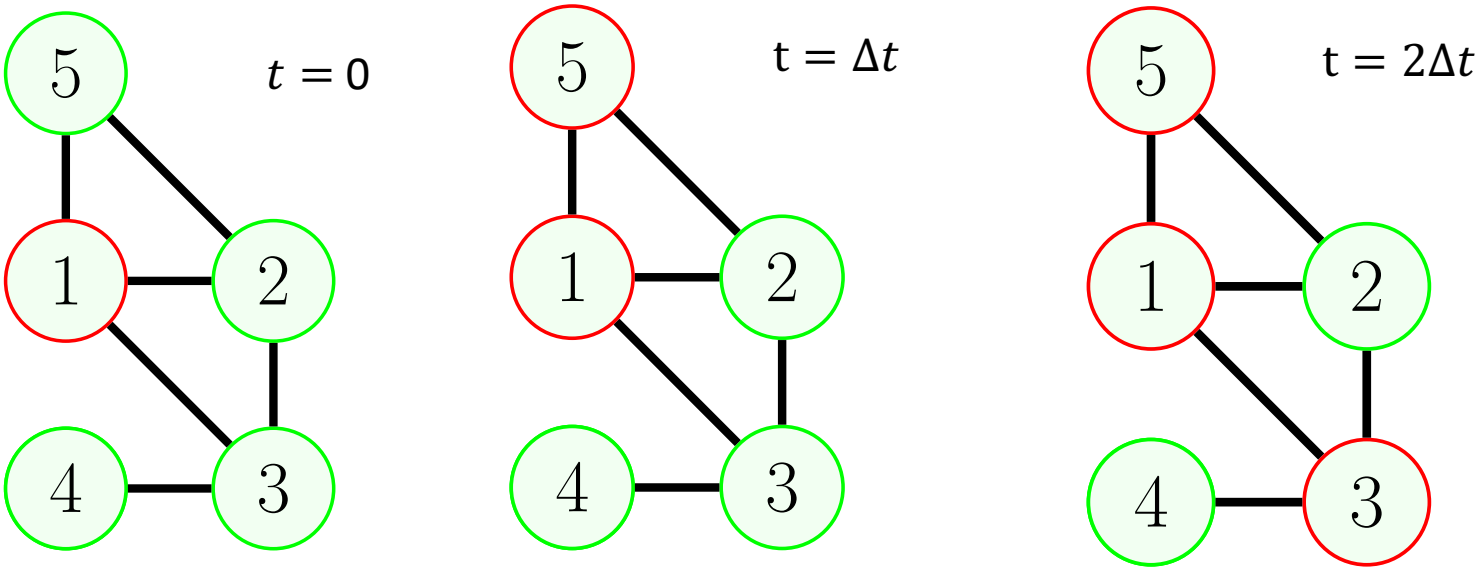




- **Now, we generate random numbers for 1-2,1-3,5-2:**

  np.random.rand(3)
  Out[46]: array([0.81424296, 0.04301428, 0.45276639])

  and there will be transmission along link 1-3



- **We can continue this process until all nodes are infected to complete one simulation/realization**

- But one particular realization, particularly for larger networks, is not that interesting. Can we predict what will happen on average?

- Let's say we have a graph with N nodes; at a given time, node i is either susceptible ($x_i = 0$) or infectious ($x_i = 1$)

- Here, $x_i(t)$ is a random variable which indicates the state of node i at time, t

  - As we have seen before with indicator variables: $\langle x_i(t) \rangle = P(x_i(t) = 1)$ where $P(x_i(t) = 1)$ is the probability that $x_i(t) = 1$

- Our goal is to relate the state of a node at time $t + \Delta t$ to the state of the network at time t

- We know that a node will be infected at $t + \Delta t$ if either:

  - The node was already infected *or*

  - It acquired the disease from a neighbor during between times t and $t + \Delta t$

- **So we have:**

  (Probability that node i is infected at time $t + \Delta t$) =
  (Probability that node i is infected at time t *or* that node i is infected by a neighbor between times t and $t + \Delta t$)

  which we write as,

  $$P(x_i(t + \Delta t) = 1) = P(x_i(t) = 1) + P(x_i \to 1)$$

  where $P(x_i \to 1)$ is the probability that node i is infected by a neighbor between times t and $t + \Delta t$

- **Transmission from neighbor j to node i depends on the joint probability: $P\big(x_i(t) = 0, x_j(t) = 1\big)$ and over the timespan Δt, the probability that node i becomes infected by j is: $\beta \Delta \mathrm{t}\, A_{ij} P\big(x_i(t) = 0, x_j(t) = 1\big)$**

- **What is the probability of transmission from any one neighbor of i?**

- **Summing the influence of each neighbor gives:**

$$P(x_i \to 1) = \beta \Delta t \sum_{j=1}^{N} A_{ij} P\big(x_i(t) = 0, x_j(t) = 1\big) + O(\Delta t^2)$$

  - **The $O(\Delta t^2)$ accounts for cases where transmission via two or more neighbors occurs during the same time step. We can neglect such cases if Δt is sufficiently small**

- **The expression above leads to the master equation for the network SI model:**

$$P(x_i(t + \Delta t) = 1) = P(x_i(t) = 1) + \beta \Delta t \sum_{j=1}^{N} A_{ij} P\big(x_i(t) = 0, x_j(t) = 1\big) + O(\Delta t^2)$$

- **It is convenient to restate this in terms of expectations:**

$$\langle x_i(t + \Delta t \rangle = \langle x_i(t) \rangle + \beta \Delta t \sum_{j=1}^{N} A_{ij} \left\langle \big(1 - x_i(t)\big) x_j(t) \right\rangle + O(\Delta t^2)$$

- **Next, divide both sides by Δt and let $\Delta t \to 0$:**

$$\frac{d\langle x_i \rangle}{dt} = \beta \sum_{j=1}^{N} A_{ij} \left\langle (1 - x_i) x_j \right\rangle \quad \textbf{(network SI model)}$$

- **There is one problem, here – there are more unknowns than equations!**

- **The problem is that the expectation of a product is not generally the product of an expectation. Some sort of approximation of the RHS is needed to make progress**

- **Naïve approach: assume that the states of nodes are all statistically independent of each other:** $\langle (1 - x_i)x_j \rangle \approx \langle 1 - x_i \rangle \langle x_j \rangle$ **(naïve approximation)**

  and our equation becomes,

$$\frac{d\langle x_i \rangle}{dt} = \beta \langle 1 - x_i \rangle \sum_{j=1}^{N} A_{ij} \langle x_j \rangle$$ **(naïve network SI model)**

- **We will discuss the "naïve approximation" and more accurate approaches later, but for now, I'll just state that this approximation works reasonably well for large complete graphs and graphs with a tree-like structure (i.e. few loops)**

- **How can we analyze the naïve network SI model? A good first step when analyzing differential equations is to look for equilibrium states (i.e. *fixed points*) where the solution is time-independent:** $\frac{d\langle x_i \rangle}{dt} = 0$ **for all** i**.**

- **We have an equilibrium state if,** $\beta\langle 1 - x_i\rangle \sum_{j=1}^{N} A_{ij} \langle x_j\rangle = 0 \quad \forall\, i \in \{1,2,\dots,N\}$
  - **We can see that there is an equilibrium infection-free state:** $\langle x_i\rangle = 0$ **for each node**
  - **Now, there is also an equilibrium "everyone is infectious" state:** $\langle x_i\rangle = 1$ **for each node**
- **It is important to understand if an equilibrium state is *stable.* Here, we will analyze the response of the infection-free state to *small* perturbations**
- **Let** $\langle x_i\rangle = 0 + \epsilon y_i + O(\epsilon^2)$ **with** $\epsilon \ll 1, y_i \sim O(1)$
  - **This represents the addition of a small amount of infection to the infection-free state. The parameter** ϵ **indicates how small the perturbation is but its precise value will not be needed.**
- **The naïve network SI model is then,** $\epsilon \frac{dy_i}{dt} = \beta(1 - \epsilon y_i) \sum_{j=1}^{N} \epsilon A_{ij}\, y_j + O(\epsilon^2)$

- **Dividing by ϵ and letting $\epsilon \to 0$ gives the *linearized* naïve network SI model:**

$$\frac{dy_i}{dt} = \beta \sum_{j=1}^{N} A_{ij}\, y_j$$

- **We now have a linear eigenvalue problem, and we need initial conditions for each node, $y_i(t=0) = y_{0,i}$**
  - **Then assume that $y_i = \tilde{y}_i\, e^{\lambda t}$ and let $\tilde{\mathrm{y}} = [\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_N]^T$**
  - **This gives, $\mathrm{A}\tilde{\mathrm{y}} = \frac{\lambda}{\beta}\, \tilde{\mathrm{y}}$ and for a given graph, we can solve for the eigenvalues and eigenvectors using Numpy as discussed for linear diffusion**
  - **However, we can use our results on bounds of eigenvalues of the adjacency matrix here. From lecture 11, we know that $\overline{k} \leq \max(\lambda) \leq k_{max}$**
  - **This tells us that there *will* initially be exponential spread of the disease, and that the graph structure will determine the rate of spread. The infection-free equilibrium state is unstable. Keep in mind though that the linear assumes $\langle x_i \rangle \ll 1$. The full nonlinear model will be needed when this condition is violated.**

Explanatory note on derivation of network-SI model:

- **Earlier in this lecture, the master equation for the network-SI model is presented as:**

$$P(x_i(t+\Delta t)=1) = P(x_i(t)=1) + \beta \Delta t \sum_{j=1}^{N} A_{ij} P\big(x_i(t)=0, x_j(t)=1\big) + O(\Delta t^2)$$

- **Why do we need this $O(\Delta t^2)$ term? Consider the following illustrative example. Say that node i has two neighbors which are nodes a and b.**

- **Let T_a represent the event of node i being infected by node a during a time step**

- **Then, our master equation for node i is,**

$$P(x_i(t+\Delta t)=1) = P(x_i(t)=1) + P(T_a \cup T_b)$$

- **Now, $P(T_a \cup T_b) = P(T_a) + P(T_b) - P(T_a \cap T_b)$**

- **The first two terms on the RHS correspond to the summation term in the master equation**

- **The $O(\Delta t^2)$ factor accounts for "multiple transmission events", like $P(T_a \cap T_b)$**

- **So what is $P(T_a \cap T_b)$? The two events are independent so, $P(T_a \cap T_b) = P(T_a)P(T_b)$ and, $P(T_a)P(T_b) = [(\beta\Delta t)P(x_i = 0, x_a = 1)][(\beta\Delta t)P(x_i = 0, x_b = 1)]$**

- **So, we see that the term is $O(\Delta t^2)$ and when we take the limit $\Delta t \to 0$, this term vanishes.**

- **More generally, a node may have more than 2 neighbors and then we will have higher-order terms like $O(\Delta t^3), O(\Delta t^4), etc$ …, but these will all vanish as well when $\Delta t \to 0$.**

Lecture 13

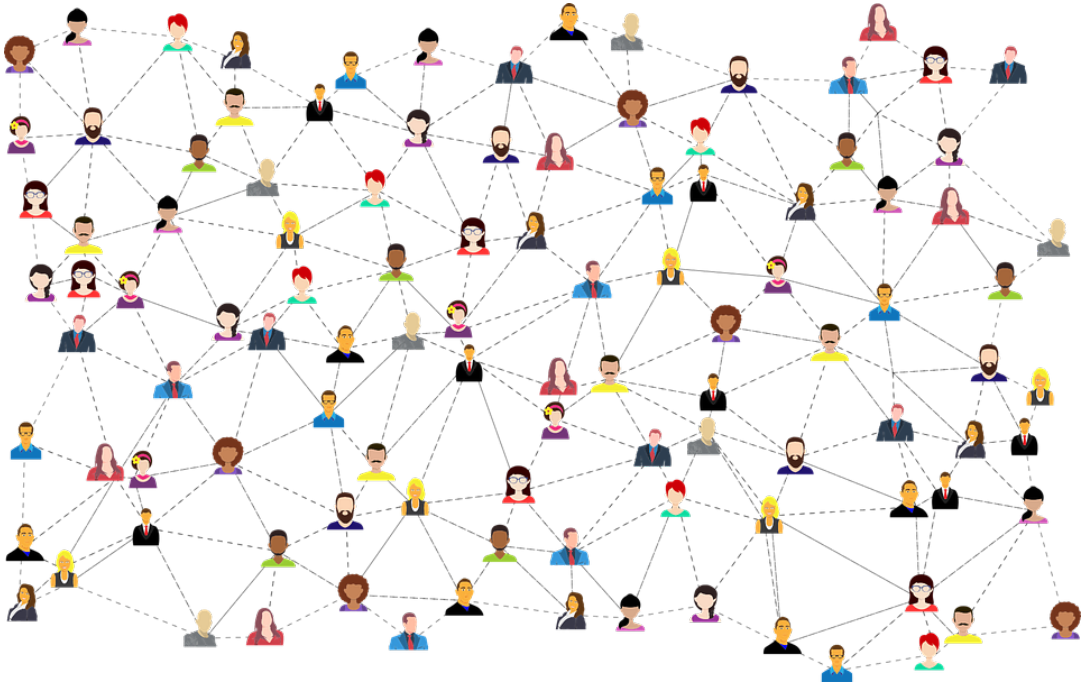
Epidemics on networks II: degree-based approximation and pair approximation

Epidemics on networks (continued)

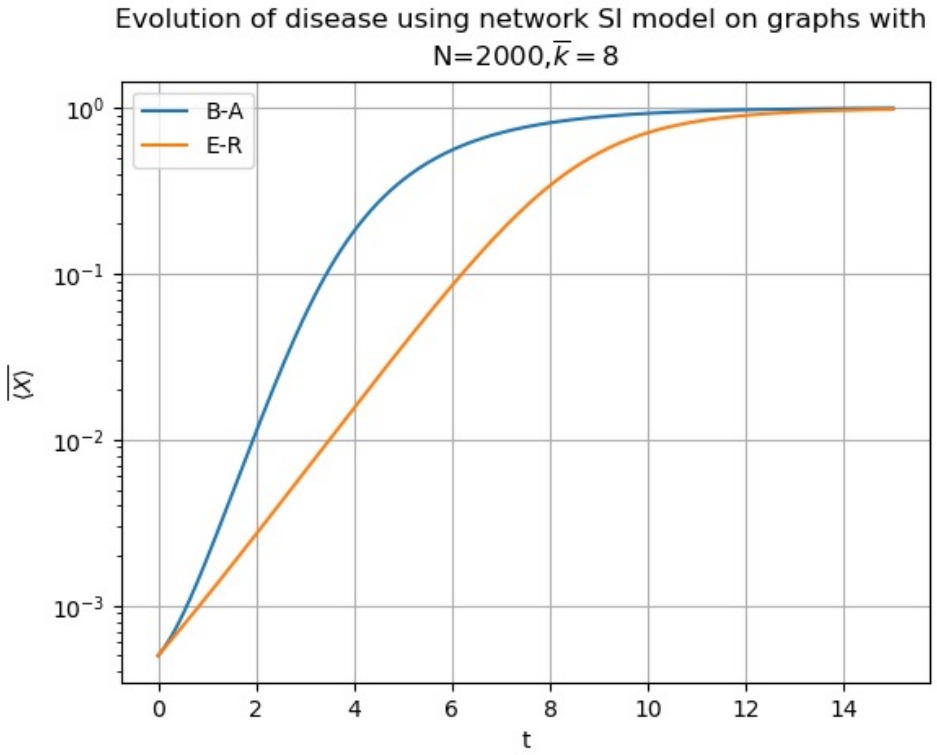
Let's continue thinking about epidemics on networks

1. **Can we obtain a clearer understanding of the connection between the network structure and epidemic spread?**

2. **How do we improve upon the "naïve approximation"?**



- **The figure shows simulation results for the naïve network SI model (with $\beta = 0.1$) on a B-A and a G_{Np} graph**
- **Both graphs have 2000 nodes and roughly the same average degree**
- **Initially, a node with degree equal to the average degree is infected**
- **We can see that the disease spreads more easily in the B-A graph. Why?**
- **We can guess that this may be due to the presence of hubs**



Degree-based approximation

- **We saw previously that the early spread of an epidemic is connected to the most-positive eigenvalue of the network adjacency matrix, and this partially supports our guess.**

- **A more direct connection to network structure can be made by using the *degree-based approximation* for a random graph:**

  - **Assume all nodes with degree k have the same probability of being infected: $P(x_i = 1) = P(x_j = 1)$ if $k_i = k_j$. Let ϕ_k be the probability that nodes with degree k are infected. We then have, $P(x_i = 1 | k_i = k) = \phi_k$**

  - **Assume that probability of a link on a node with degree $\tilde{k}$ being connected to a node with degree k' is a function of the degrees only, $\theta(\tilde{k}, k')$. Also assume that this probability and ϕ_k are independent for any k**

- **This approach is not something we can justify rigorously in advance, but it does simplify the problem to considering distinct degrees in the degree distribution**

- **How does the probability that a node with degree k is infectious evolve in time? We just need to adapt our previous work with $\langle x_i \rangle$**

- **As before, we will have $x_i(t+\Delta t)=1$ if (A) $x_i(t)=1$ *or* (B) it acquires the disease from a neighbor between times t and $t+\Delta t$**

- **How do we compute $P(\mathrm{B})$? The probability of transmission to i from an infectious neighbor is still:**

$$P(x_i \to 1) = \beta \Delta t \sum_{j=1}^{N} A_{ij} P\big(x_i(t)=0, x_j(t)=1\big) + O(\Delta t^2)$$

and we re-write this as a sum over the neighbors of node i:

$$P(x_i \to 1) = \beta \Delta t \sum_{j \in N_i} P\big(x_i(t)=0, x_j(t)=1\big) + O(\Delta t^2)\,.$$

Note: all probabilities here are conditional on $k_i = k$

- **Next restate the joint probability as a conditional probability,** $P\big(x_i(t) = 0, x_j(t) = 1\big) = P\big(x_j(t) = 1 \big| x_i(t) = 0\big)P(x_i(t) = 0)$ **and the probability of transmission becomes:**

$$P(x_i \to 1) = \beta \Delta t P(x_i(t) = 0) \sum_{j \in N_i} P\big(x_j(t) = 1 \big| x_i(t) = 0\big) + O(\Delta t^2)$$

- **By definition,** $P(x_i(t) = 1 | k_i = k) = \phi_k$ **, and all of the probabilities above are conditional on** $k_i = k$**, so we can replace** $P(x_i(t) = 0)$ **with** $1 - \phi_k$ **.**
- **We have to be more careful with the conditional probability. Since we know that one neighbor of node** j **is susceptible, it will be infectious if nodes with degree** $k_j - 1$ **are infectious. We have assumed that the network has been generated by a random graph model, so we also have to consider the distribution of values** k_j **can take. With these considerations, we write:**

$$P\big(x_j(t) = 1 \big| x_i(t) = 0\big) = \sum_{k'=1}^{k_{max}} \phi_{k'-1} \theta(k, k')$$

- **We have used** $\phi_{k'-1}$ **instead of** $\phi_{k'}$ **since we know that** i **is susceptible.**

- **Our final expression for the probability of transmission is:**

$$P(x_i \to 1) = \beta \Delta t (1 - \phi_k(t)) \sum_{j \in N_i} \sum_{k'=1}^{k_{max}} \phi_{k'-1}(t) \theta(k, k') + O(\Delta t^2)$$

$$= \beta \Delta t (1 - \phi_k(t)) k \sum_{k'=1}^{k_{max}} \phi_{k'-1}(t) \theta(k, k') + O(\Delta t^2)$$

- **And the "degree-based" master equation is,**

$$\phi_k(t + \Delta t) = \phi_k(t) + \beta \Delta t (1 - \phi_k(t)) k \sum_{k'=1}^{k_{max}} \phi_{k'-1}(t) \theta(k, k') + O(\Delta t^2)$$

- **Finally, dividing by Δt and letting $\Delta t \to 0$ gives,**

$$\frac{d\phi_k}{dt} = k\beta(1 - \phi_k) \sum_{k'=1}^{k'_{max}} \theta(k, k') \phi_{k'-1}$$

- **To make further progress, we need to specify $\theta(k,k')$, the probability that nodes with degrees k and k' are linked.**

- **Assume that the network was generated by the configuration model. For the configuration model, we know that $\theta(k,k') = \frac{k'p_{k'}}{\bar{k}}$, so we have**

$$\frac{d\phi_k}{dt} = k\beta(1-\phi_k)\sum_{k'=1}^{k'_{max}} \frac{k'p_{k'}}{\bar{k}}\phi_{k'-1}$$

- **Now, let's consider the initial spread of infection when $\phi_k \ll 1$.**

- **We let $\phi_k = \epsilon\tilde{\phi}_k + O(\epsilon^2)$ with $\epsilon \ll 1$. Substituting this expression into our equation above:**

$$\epsilon\frac{d\tilde{\phi}_k}{dt} = k\beta\left(1-\epsilon\tilde{\phi}_k\right)\sum_{k'=1}^{k'_{max}} \epsilon\frac{k'p_{k'}}{\bar{k}}\tilde{\phi}_{k'-1} + O(\epsilon^2)$$

- **Dividing by ϵ and letting $\epsilon \to 0$ gives the linearized system:**

$$\frac{d\tilde{\phi}_k}{dt} = k\beta \sum_{k'=0}^{k'_{max}-1} \frac{k'+1 p_{k'+1}}{\overline{k}} \tilde{\phi}_{k'} \text{ (*)}$$

The indices in the sum have been rearranged for convenience with the assumption that $\tilde{\phi}_0 = 0$.

These equations can be solved analytically. Let $\psi = \sum_{k'=0}^{k'_{max}-1} \frac{k'+1 p_{k'+1}}{\overline{k}} \tilde{\phi}_{k'}$ (**)

and the equation above becomes, $\frac{d\tilde{\phi}_k}{dt} = k\beta\psi$

Differentiating (**) with respect to time and using (*), we find,

$$\frac{d\psi}{dt} = \beta\psi \sum_{k'=0}^{k'_{max}-1} \frac{k'+1 p_{k'+1}}{\overline{k}} k'$$

and $\sum_{k'=0}^{k'_{max}-1} \frac{k'+1 p_{k'+1}}{\overline{k}} k'$ **=** $\sum_{k'=1}^{k'_{max}} \frac{k' p_{k'}}{\overline{k}} (k'-1) = \overline{k^2}/\overline{k} - 1$

- **So, our linearized equations now read as:**

$$\frac{d\psi}{dt} = \beta\psi(\overline{k^2}/\overline{k} - 1)$$

$$\frac{d\tilde{\phi}_k}{dt} = k\beta\psi$$

and we can write down the solution:

$$\psi = \psi_0 e^{t/\tau}$$

$$\tilde{\phi}_k = \frac{k\beta\psi_0}{\tau}(e^{\frac{t}{\tau}} - 1) + \tilde{\phi}_k(t=0)$$

$$\frac{1}{\tau} = \left[\beta\left(\frac{\overline{k^2}}{\overline{k}} - 1\right)\right] \rightarrow smaller\ \tau = faster\ spread$$

$$\psi_0 = \psi(t=0) = \sum_{k'=0}^{k'_{max}-1} \frac{k'+1 p_{k'+1}}{\overline{k}} \tilde{\phi}_k(t=0)$$

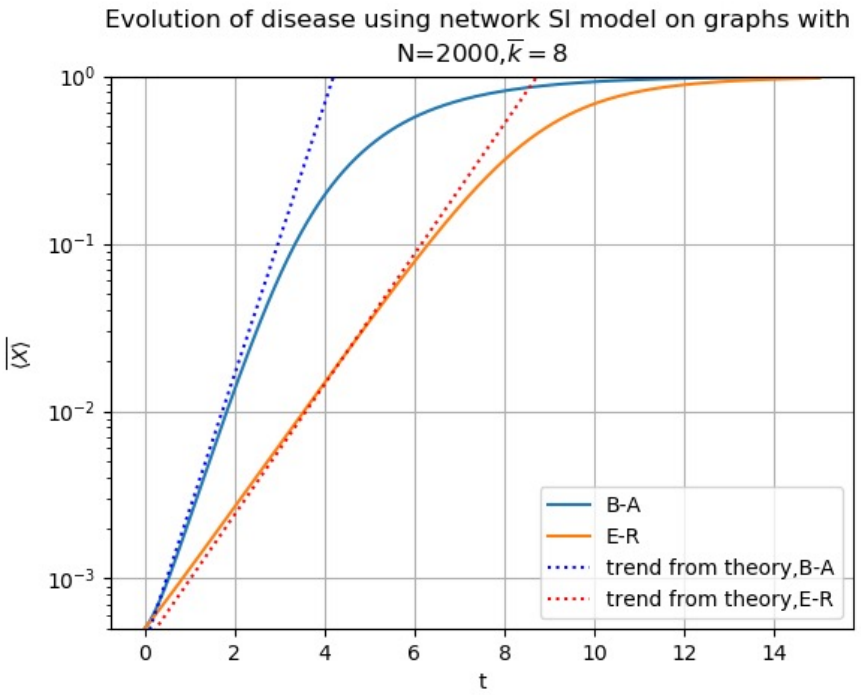
- **The initial rate at which the disease spreads is dictated by τ so we now have a much clearer connection between the network structure and the dynamics on the network**
- **But keep in mind that we made a number of simplifying assumptions to allow us to obtain this "elegant" result!**

- **Let's look back at our motivating example – our assumptions for $\theta(k, k')$ apply to the G_{Np} model if we replace $\overline{k}$ with $\langle k \rangle$. They do not directly apply to the B-A model, but let's use the results anyway.**

- **I have used the derived expressions for τ to make the trend lines →**

- **And we see pretty good agreement!**

- **In mathematical terms, the disease spreads more rapidly in the B-A graph because its degree distribution has much higher variance**



Pair approximation

- **Our network SI model is,** $\frac{d\langle x_i \rangle}{dt} = \beta \sum_{j=1}^{N} A_{ij} \left\langle (1 - x_i) x_j \right\rangle$
- **And up to now, we have used the "naïve approximation":** $\left\langle (1 - x_i) x_j \right\rangle \approx \langle 1 - x_i \rangle \langle x_j \rangle$
- **However, this is only reasonable in certain special (artificial) cases – complete graphs and tree-like graphs with few loops**
- **Many real-world networks will have high enough clustering for this approximation to be poor – think about how many of your social network neighbors are themselves neighbors**
- **Can we do better? A common approach is to use a "second-moment closure"**
  - **The idea is to derive an equation for** $\left\langle (1 - x_i) x_j \right\rangle$ **for linked node pairs where** $A_{ij} = 1$
  - **This equation will have terms with third moments, e.g.** $\left\langle (1 - x_i) x_j x_l \right\rangle$
  - **But, approximating these terms (in terms of the 1st and 2nd moments) produces much-improved results compared to the naïve approximation on graphs with high clustering**

- **How do we derive an equation for $\langle (1 - x_i)x_j \rangle$? Using the same approach we did to obtain our equation for $\langle x_i \rangle$**

- **We know that $\frac{d\langle (1-x_i)x_j \rangle}{dt} = \frac{d\langle x_j \rangle}{dt} - \frac{d\langle x_i x_j \rangle}{dt}$, and it is easier to develop an equation for $\langle x_i x_j \rangle$ which means we need to consider, $P(x_i(t + \Delta t) = 1, x_j(t + \Delta t) = 1)$**

- **We can use essentially the same reasoning as before. We just need to account for two nodes being infectious at $t + \Delta t$ rather than one:**

$$P(x_i(t + \Delta t) = 1, x_j(t + \Delta t) = 1) =$$
$$P(x_i(t) = 1, x_j(t) = 1) + P(x_i(t) = 1, x_j \to 1) + P(x_i \to 1, x_j(t) = 1) + P(x_i \to 1, x_j \to 1)$$

- **$x_i \to 1$ means that node i is susceptible at time t and infectious at $t + \Delta t$, and check that you understand what each of the four terms on the RHS represents.**

- **The first term on the RHS will be absorbed into the time derivative when $\Delta t \to 0$ and the last term will disappear as it is $O(\Delta t^2)$**

- **We need to find expressions for the 2nd and 3rd terms**

Term 2:

$$P\left(x_i(t)=1, x_j \to 1\right) = \beta \Delta t \sum_{l=1}^{N} A_{jl} P(x_i(t)=1, x_j(t)=0, x_l(t)=1)$$

Term 3:

$$P\left(x_i \to 1, x_j(t)=1\right) = \beta \Delta t \sum_{l=1}^{N} A_{il} P(x_i(t)=0, x_j(t)=1, x_l(t)=1)$$

The reasoning used to obtain these expressions is the same as for the network-SI model: $x_j \to 1$ requires j to be susceptible at t and it acquires the disease from an infectious neighbor.

- **Now rewriting these expressions as expectations and letting $\Delta t \to 0$**

$\frac{d\langle x_i x_j \rangle}{dt} = \beta \sum_{l=1}^{N} \left[A_{jl} \langle x_i s_j x_l \rangle + A_{il} \langle s_i x_j x_l \rangle \right]$ **where** $s_i = 1 - x_i$

and we can write down an equation for $\langle s_i x_j \rangle$ …

Our new equation:

$$\frac{d\langle s_i x_j \rangle}{dt} = \beta \sum_{l=1}^{N} \left[A_{jl} \langle s_i s_j x_l \rangle - A_{il} \langle s_i x_j x_l \rangle \right]$$

- **We now have to find approximations for the two 3rd moments.**

- **Term 1:** $A_{jl}\langle s_i s_j x_l \rangle$
  - **We are only interested in cases where nodes i and j are linked, and here we will also have nodes l and j linked. Assume that the only path from i to l is via node j. Then if node j is susceptible, node i will not influence l. So we assume:** $P(x_l = 1 | s_i = 1, s_j = 1) \approx P(x_l = 1 | s_j = 1)$ **and approximate term 1 as follows:**

$$A_{jl} P(s_i = 1, s_j = 1, x_l = 1) \approx A_{jl} P(x_l = 1 | s_j = 1) P(s_i = 1, s_j = 1) = \frac{A_{jl} P(x_l = 1, s_j = 1) P(s_i = 1, s_j = 1)}{P(s_j = 1)}$$

- **The same reasoning can be applied to the second term:**

$$A_{il}P\left(s_i = 1, x_j = 1, x_l = 1\right) \approx \frac{A_{ij}P(x_l=1,s_i=1)P\left(s_i=1,x_j=1\right)}{P(s_i=1)}$$

Converting probabilities into expectations, we have the equations that we want:

$$\frac{d\langle x_i\rangle}{dt} = \beta \sum_{j=1}^{N} A_{ij} \left\langle s_i x_j \right\rangle$$

$$\frac{d\langle s_i x_j\rangle}{dt} = \beta \sum_{l=1}^{N} [A_{jl} \left\langle x_l s_j \right\rangle \left\langle s_i s_j \right\rangle / \left\langle s_j \right\rangle - A_{il} \left\langle s_i x_l \right\rangle \left\langle s_i x_j \right\rangle / \langle s_i \rangle]$$

- **We initially had N equations and $N + L$ unknowns**
- **We now have $N + L$ equations and $N + L$ unknowns – is this really an improvement?**
  - **The key is that simple approximations for $\left\langle (1 - x_i)(1 - x_j)x_k \right\rangle$ and $\left\langle (1 - x_i)x_j x_k \right\rangle$ have been introduced which are much more effective than the naïve approximation for $\left\langle (1 - x_i)x_j \right\rangle$**

- **With these approximations, we have *closed* our model equations**

- **Closure problems are not unique to epidemics or network science, they typically arise when developing statistical models for complex nonlinear systems (e.g. atmospheric dynamics)**

- **The equations can be rearranged and written in a simpler form, but we will stop the theoretical development here. The last step is to critically consider the approximations we used.**

- **How can we justify:** $P\left(x_k = 1 | x_i = 0, x_j = 0\right) \approx P\left(x_k = 1 | x_j = 0\right)$**?**

    - **As with the degree-based approximation, there isn't a rigorous argument we can apply. We use it because it has been found to work well**

- The figure on the right compares "simulation" with "theory" for 2 graphs with different amounts of clustering
- "Simulation" refers to calculations using Bernoulli trials as sketched for the 5-node network in the previous lecture
- "First-order" theory refers to the network SI model with the naïve approximation
- "Second-order" uses the 2nd-moment closure
- "Transitivity" refers to the amount of clustering, and while the naïve approximation is ok for the tree-like low-transitivity network, the 2nd-moment closure works very well even when the graph contains many triangles

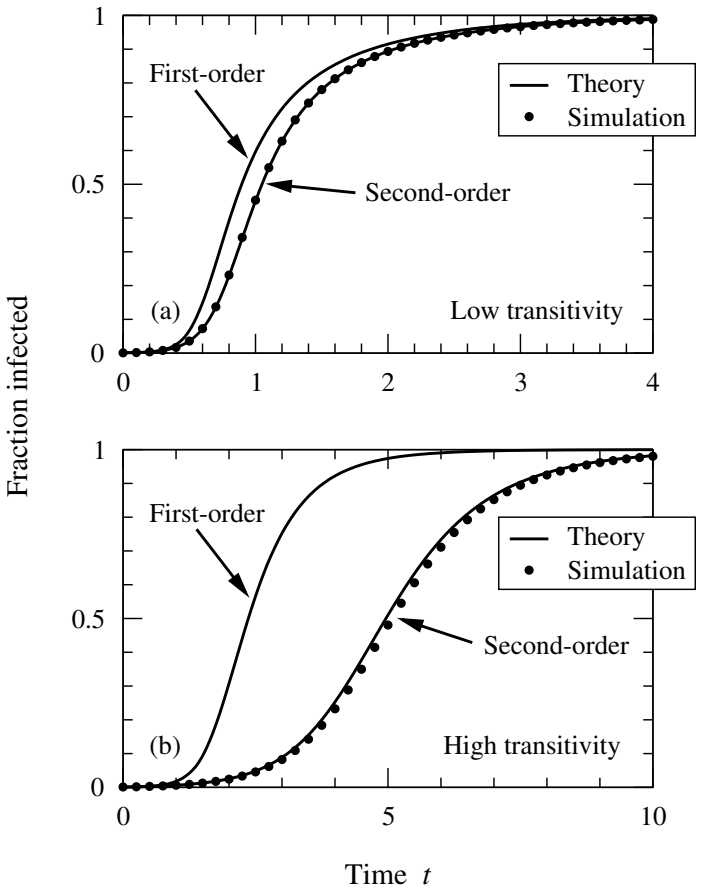


Figure taken from: Newman, Networks

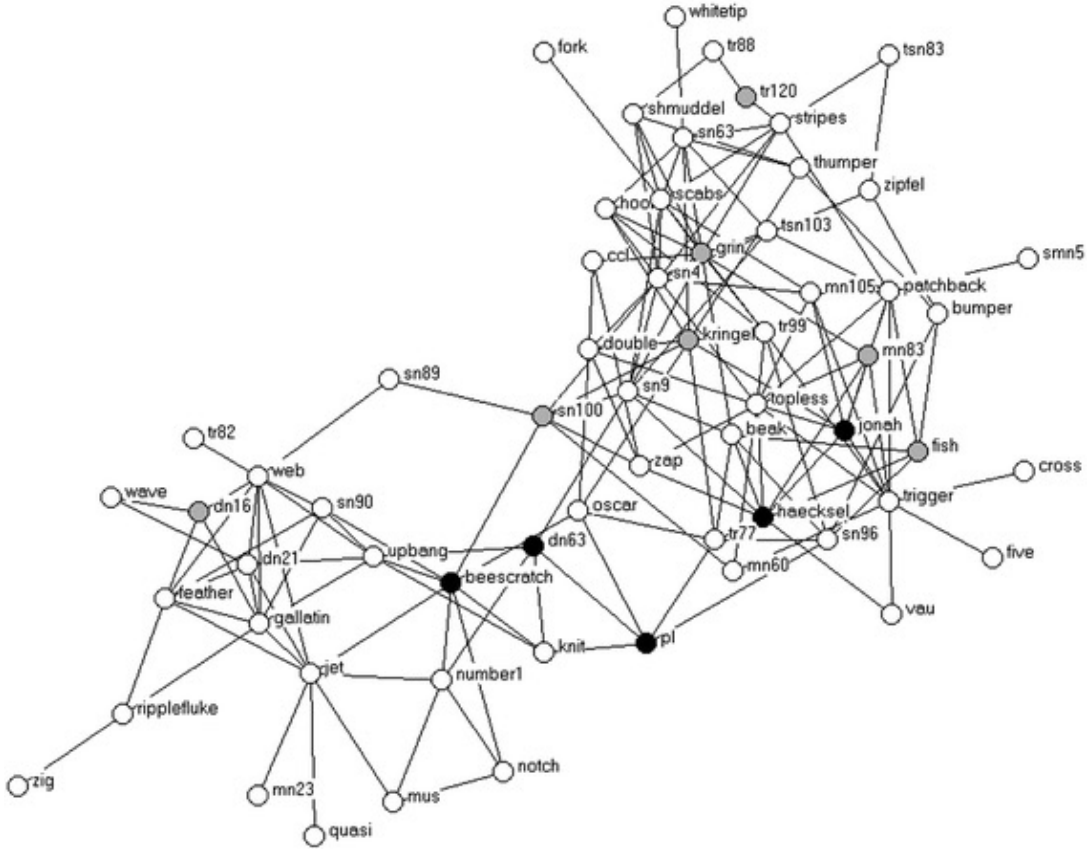
Lecture 14

Communities in networks
Modularity and modularity maximization

Communities in networks

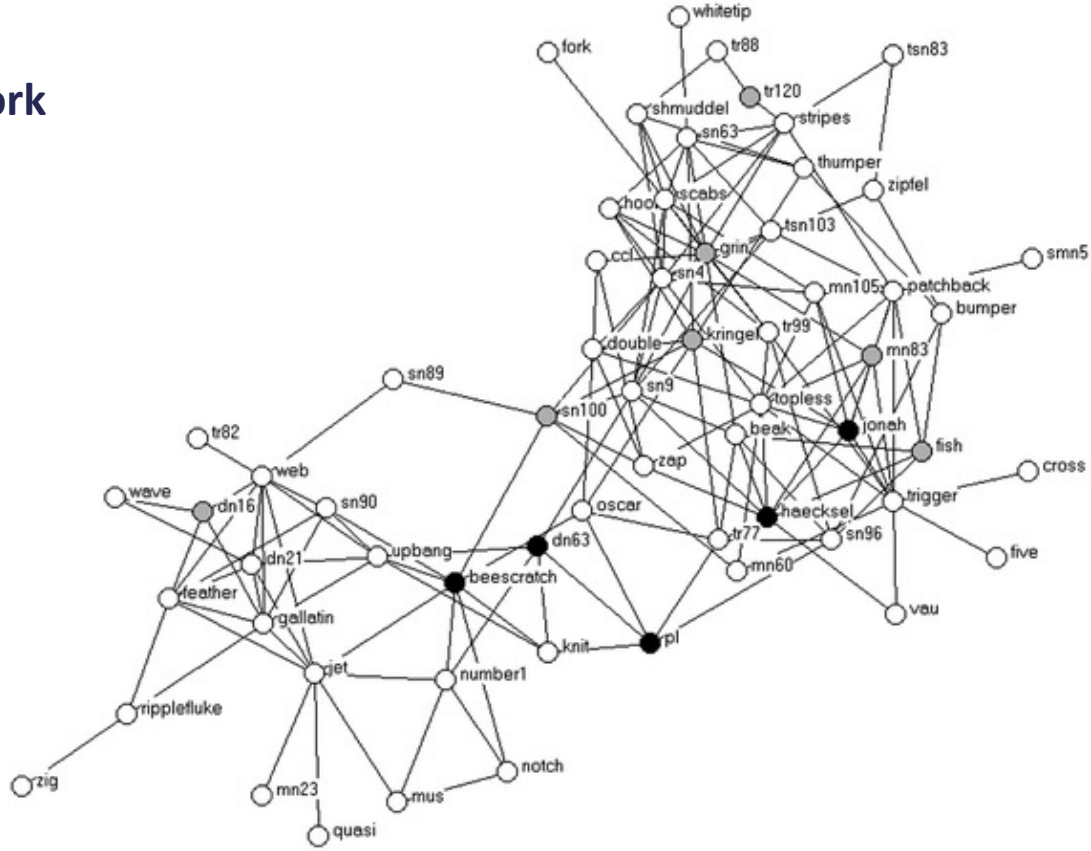
- The behavior of a population of social animals can be characterized by a network
- But are there important “sub-networks”?
- From our own experience, we know communities can form with interactions that are distinct from the full group
- How can we “scientifically” identify these communities?

- **The figure shows a social network for a group of 62 bottlenose dolphins observed over a number of years near New Zealand**
- **Links have been placed between dolphins observed spending more time near each other than would be expected from random pairings**
- **How do we analyze this graph?**
- **We can look at the usual quantities like clustering and the degree distribution**
- **But can we identify *communities* based on the graph structure?**



*Lusseau, D. Evidence for social role in a dolphin social network. Evol Ecol **21,** 357–366 (2007).*

- **Looking at the graph, we can guess that there could be 2 communities**
- **But how do we "draw" the partition? How do we compare 2 different partitions?**
- **Graph partitioning is an old problem in computer science.**
- **However, bespoke methods for network science have been developed, and we will focus on one of those here**
- **Keep in mind that there are many different methods and there is no universal best choice**



*Lusseau, D. Evidence for social role in a dolphin social network. Evol Ecol **21,** 357–366 (2007).*

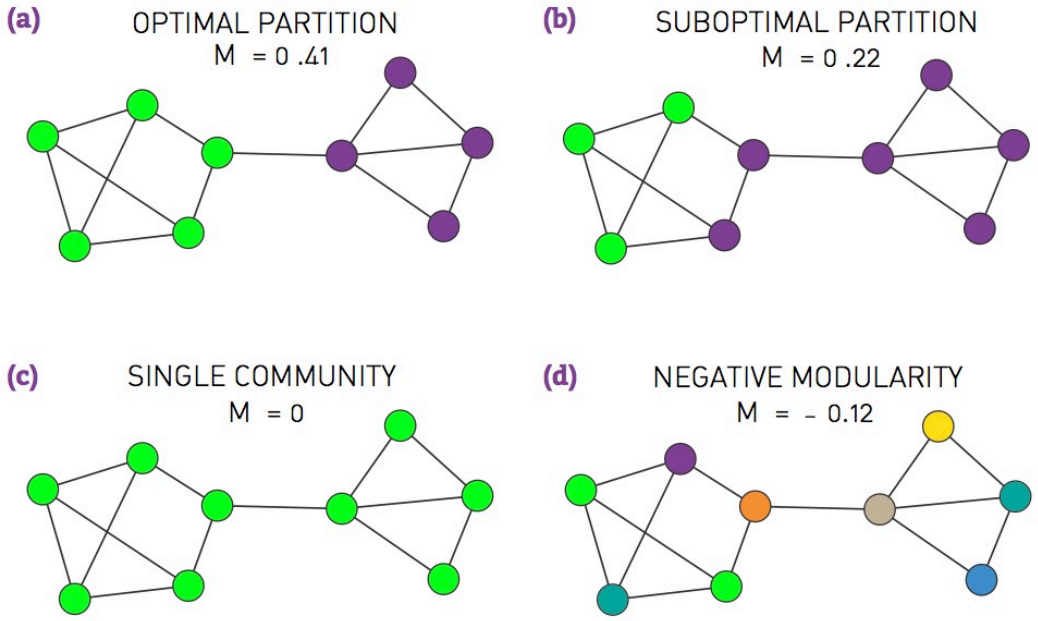
Modularity

- We need to define a quantity that will quantify the "quality" of a partition of a graph into 2 non-overlapping groups of nodes
- Qualitatively, nodes within each group should be densely connected relative to the connections between the groups
- One of the most widely-used such quantities is the *modularity*
- The basic idea is to compare the number of links within a group to the number expected if the nodes had been connected randomly (while preserving their degrees)
    - If the difference is large, we assume that the partition captures a relative preference for contacts within that group
    - And from our discussion of the configuration model, we know that the expected number of links between 2 nodes in a "randomly-wired" graph with a given degree distribution is, $\langle l_{ij} \rangle \approx \frac{k_i k_j}{2L}$

- **The *modularity* of a set of nodes, S_a , is defined as: $M_a = \frac{1}{2L}\sum_{i\in S_a}\sum_{j\in S_a}\left(A_{ij} - \frac{k_i k_j}{2L}\right)$**
  - **Note that $\sum_{i\in S_a}\sum_{j\in S_a}(A_{ij})$ is twice the number of links connecting nodes in the set to other nodes in the same set.**
  - **Also note that $\sum_{i\in S_a}\sum_{j\in S_a}\left(\frac{k_i k_j}{2L}\right) = \frac{1}{2L}\sum_{i\in S_a} k_i \sum_{j\in S_a} k_j = \frac{(K_a)^2}{2L}$ where K_a is the total number of stubs attached to nodes in S_a**
- **For a given partition of a graph into 2 (disjoint) sets of nodes, S_1 and S_2, the modularity of the partitioned graph is just the sum of the modularities of each set: $M = M_1 + M_2$**
- **And this generalizes as you would expect to any number of disjoint sets. For q disjoint sets: $M = M_1 + M_2 + M_3 + M_4 + \cdots + M_q$**

- A simple example from Barabasi (figure 9.16) is shown below

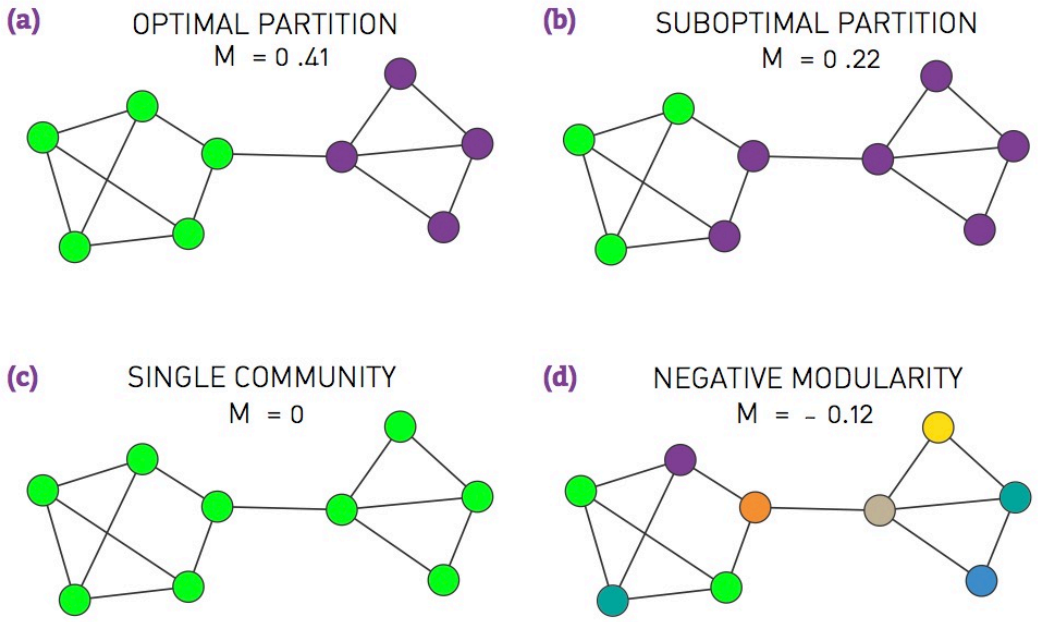
- Check your understanding: what is the modularity of the set of 4 purple nodes in the community on the right in example (a) below?



- A simple example from Barabasi (figure 9.16) is shown below

- Check your understanding: what is the modularity of the set of 4 purple nodes in the community on the right in example (a) below?

- The total number of links connecting purple nodes to other purple nodes is 5, so $\sum_{i\in S_a}\sum_{j\in S_a}(A_{ij}) = 2 * 5 = 10$. The total number of stubs attached to purple nodes is, $K_a = 11$, and $M_a = \frac{1}{2L}(10 - \frac{11^2}{2L})$ with $L = 13$.



Notes:

- If all nodes are in the same community, $M = 0$
- It can be shown that the maximum is $M = 1$
- The definition can be extended to partitions with an arbitrary number of parts – an example is shown in (d)

Modularity maximization

- Thinking back to our dolphin example, our goal, is to assign each node to either S_1 *or* S_2 so that the graph modularity is maximized
- There is a problem though: there are 2^{N-1} such partitions and it is infeasible to compute the modularity for all of them for large graphs
- In fact, there is no good way to find *the* maximum for large graphs. Instead, we aim to find a split that gives a result close to the maximum (modularity maximization is *NP-hard*)
- There are multiple approaches for this, we will look at a spectral method
- First, we introduce an indicator variable $s_i = \pm 1$ which tells us if node i is in S_1 or S_2
  - Then, $\frac{1}{2}\left(s_i\, s_j + 1\right) = 1$ if nodes i and j have been assigned to the same group and is 0 otherwise
- The modularity for a given partition can be written as,

$$M = \frac{1}{4L}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(A_{ij} - \frac{k_i k_j}{2L}\right)(s_i\, s_j + 1)$$

- **Introducing the *modularity matrix*** $\mathrm{B}, B_{ij} = \left(A_{ij} - \frac{k_i k_j}{2L}\right)$**, we have,**

$$M = \frac{1}{4L}\sum_{i=1}^{N}\sum_{j=1}^{N} B_{ij}\, s_i\, s_j = \frac{1}{4L}\mathrm{s}^{\mathrm{T}}\mathrm{Bs}$$

- **We then have a discrete constrained optimization problem:**
  Find s **such that** $\mathrm{s}^{\mathrm{T}}\mathrm{Bs}$ **is maximized with the constraint that each element of** s **is** ± 1

- **Finding a precise solution to this problem in a reasonable amount of time is extremely difficult in general, so instead, approximate methods have been developed which aim to "get close" to the maximum in a reasonable amount of time**

- **An effective approach is to relax the constraint to** $|\tilde{\mathrm{s}}|^2 = N$ **where the elements of** $\tilde{\mathrm{s}}$ **are allowed to be real numbers. As we will see, this gives a much simpler optimization problem but then we will have to "adjust" the solution to find** s **with** $s_i = \pm 1$

- We enforce the new constraint using the Lagrange multiplier, γ, so the optimization problem becomes:

$$\text{Find } \tilde{\mathrm{s}}, \gamma \text{ such that } Q = \tilde{\mathrm{s}}^{\mathrm{T}} \mathrm{B} \tilde{\mathrm{s}} - \gamma\left(\tilde{\mathrm{s}}^{\mathrm{T}} \tilde{\mathrm{s}} - N\right) \text{ is maximized}$$

  - Notice that if we set $\frac{\partial Q}{\partial \gamma} = 0$, we have $\tilde{\mathrm{s}}^{\mathrm{T}} \tilde{\mathrm{s}} = N$ and the constraint is satisfied

- It is easier (for me) to work with this expression in index notation,

$$Q = \sum_{i=1}^{N} \sum_{j=1}^{N} \tilde{s}_i B_{ij} \tilde{s}_j - \gamma \sum_{j=1}^{N} \left(\tilde{\mathrm{s}}_{\mathrm{j}}^2 - N\right)$$

And we require $\frac{\partial Q}{\partial \tilde{s}_l} = 0$ which, after some arithmetic, leads to, $\sum_{j=1}^{N} B_{lj} \tilde{s}_j = \gamma \tilde{s}_l$

- This is of course an eigenvalue problem with eigenvalue γ and eigenvector $\tilde{\mathrm{s}}$. So $\tilde{\mathrm{s}}$ should be an eigenvector of B with length $\sqrt{N}$, but which eigenvector should it be?

- B is symmetric so the eigenvalues will be real but may be positive or negative. Note that there will also be at least one zero eigenvalue which corresponds to the case where all nodes are assigned to the same community

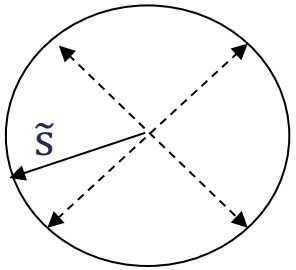
- **Order the eigenvalues of B such that , $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_N$ with $\mathrm{B}\mathrm{v}_i = \lambda_i \mathrm{v}_i$ and set $\tilde{\mathrm{s}} = \mathrm{v}_i$, with $|\mathrm{v}_i|^2 = N$. The approximate modularity is then,**

$$\widetilde{M} = \frac{1}{4L}\mathrm{v}_i^T\, \mathrm{B}\mathrm{v}_i = \frac{1}{4L}\lambda_i \mathrm{v}_i^T \mathrm{v}_i = \frac{\lambda_i N}{4L},$$

  - **We can see that we should choose $\gamma = \lambda_1$ and $\tilde{\mathrm{s}} = \mathrm{v}_1$ which will give $\widetilde{M} = \frac{\lambda_1 N}{4L}$.**

- There is one last task, to "adjust" the solution vector so that its elements are all 1 or -1
- Let s be this adjusted vector. The inner product of s with the eigenvector, $\tilde{s}$, is $\tilde{s}^{T}s = N\cos\theta$ where we have scaled s so that $|s|^2 = N$ and θ is the angle between the two vectors
- The goal then is to construct s so that $|\theta|$ is minimized or equivalently so that $\tilde{s}^{T}s$ is maximized
- This will occur if each element of s is chosen to have the same sign as the corresponding element of $\tilde{s}$ (if an element of $\tilde{s}$ is zero, it does not matter which choice is made)
- The geometric interpretation of this adjustment is that $\tilde{s}$ points to the surface of a $N-1$ sphere and s must point to a corner of a N-cube. We choose the corner that minimizes the angle between the two vectors.

Example: For $N = 2$, the possible solutions are the 4 dashed vectors, while the optimization problem will give an $\tilde{s}$ pointing to a point on the circle which has radius $\sqrt{2}$



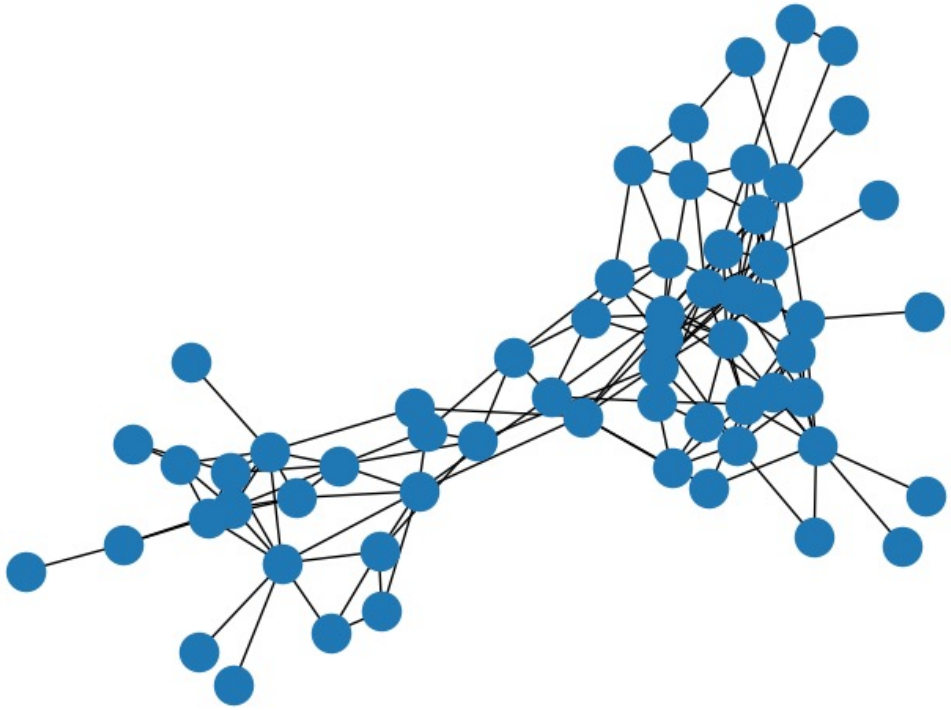
In summary, the spectral modularity maximization method requires the following steps

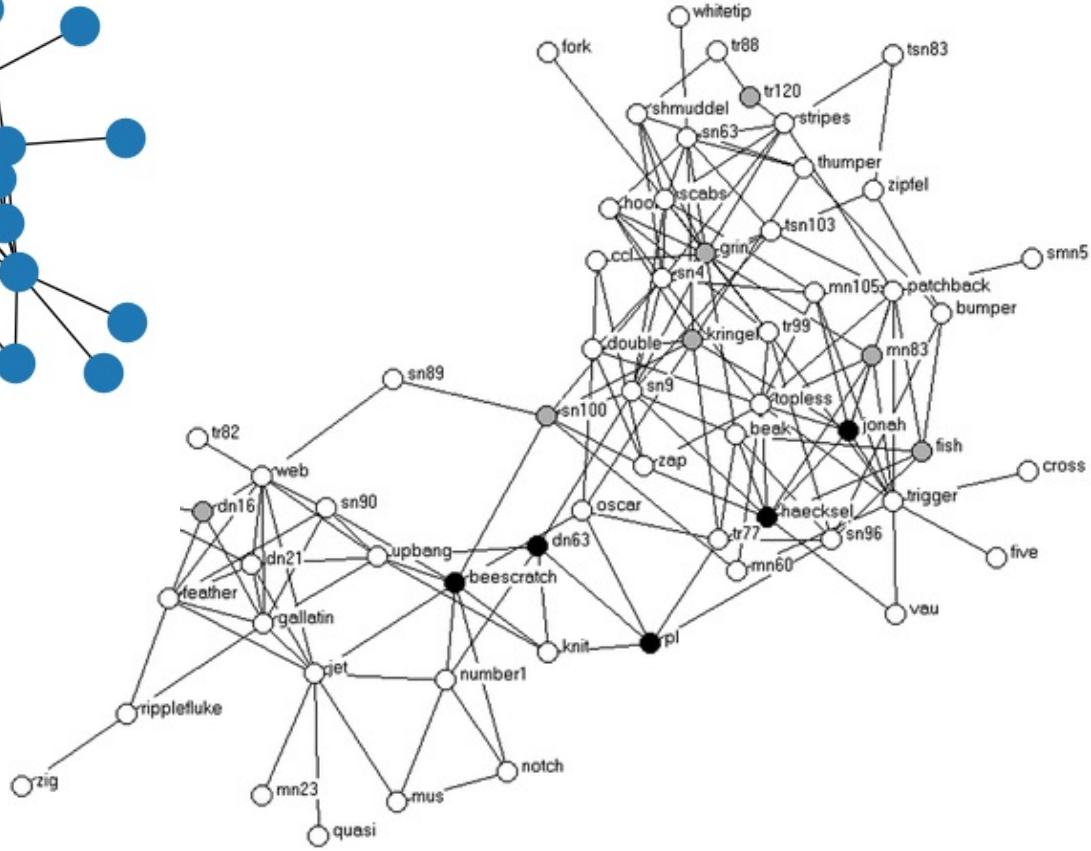
1. **Construct modularity matrix B**
2. **Compute leading eigenvalue of B and corresponding eigenvector, $\tilde{\mathrm{s}}$**
3. **Construct s based on the signs of the elements in $\tilde{\mathrm{s}}$**

It is straightforward to implement this in Python with numpy, scipy, and networkx.

Dolphin communities

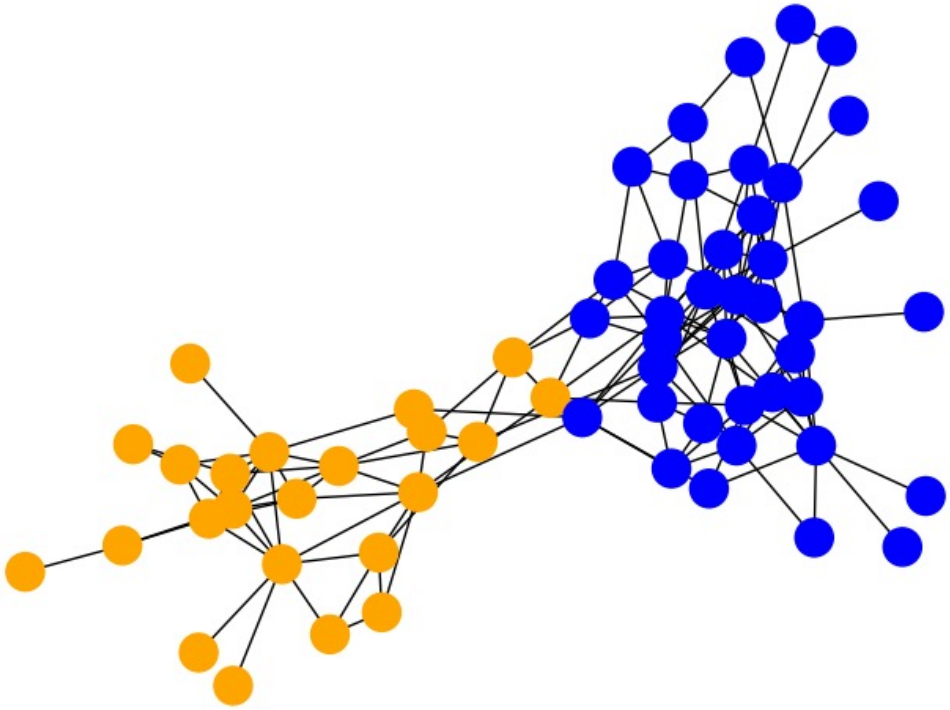
- **Let's apply this method to the dolphin social network**
- **The** NetworkX **visualization is shown as well as the original**





Lusseau, D. Evidence for social role in a dolphin social network. Evol Ecol ***21,*** *357–366 (2007).*

- Applying the spectral method, we obtain the communities shown
- This seems reasonable enough, however we can form a stronger conclusion

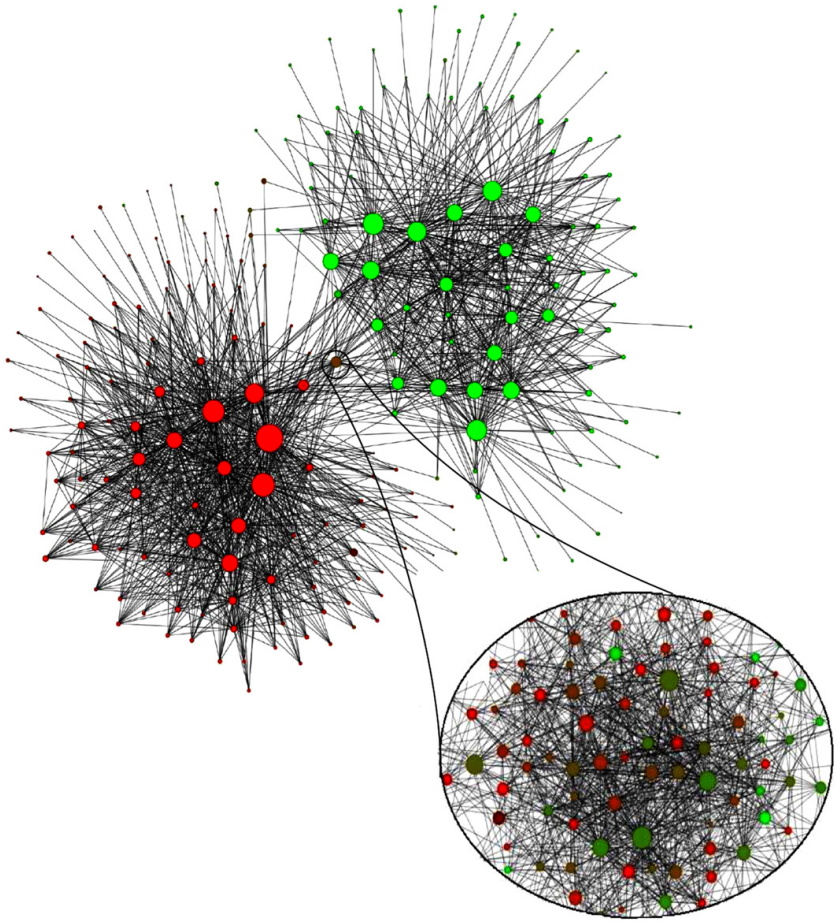


- During the period that these dolphins were observed, one dolphin left the pod for a while and then returned
- While this dolphin was away, the pod split into 2 separate groups
- Those 2 groups correspond almost exactly to the communities identified by the spectral method! There are two dolphins misclassified as “orange”.

Lecture 15

More on modularity
Laplacian graph partitioning

More on modularity



Barabasi figure 9.1: communities in a Belgium mobile phone operator's call network

- **Our initial view of community detection was based on modularity maximization**
- **However, there are weaknesses to this approach which we will now examine**

Let's take a critical view of modularity:

- **First, for very large networks, there are typically a large number of partitions with similar modularity scores**
- **It is used to compare different partitions for a single network but cannot (and should not) be used to compare different networks**
- **Nodes are not allowed to belong to multiple communities (see §9.5 in Barabasi)**
- **It also suffers from a "resolution limit" which we will now discuss in more detail**

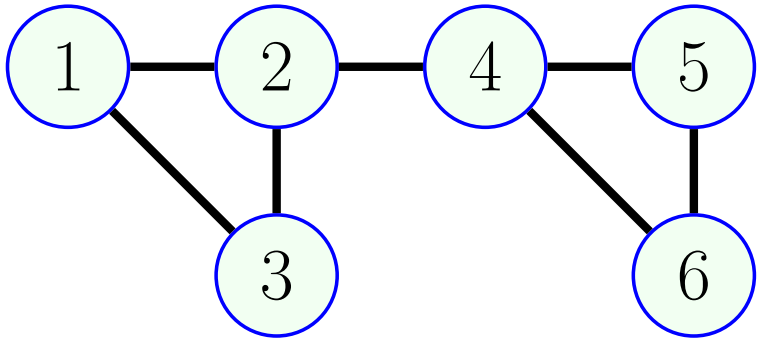
- **Consider cases where the number of distinct sets of nodes in a partition may be greater than 2**

- **The modularity of one set, S_a, is still, $M_a = \frac{1}{2L}\sum_{i\in S_a}\sum_{j\in S_a}\left(A_{ij} - \frac{k_i k_j}{2L}\right)$**

  - **But consider what happens when $\frac{k_i k_j}{2L} \ll 1$ for each node-pair in the set.**

  - **There will then be a general "preference" to combine small sets**

  - **As a result, modularity maximization can combine two sets which are clearly distinct communities**



- **Consider these 6 nodes in a larger graph**
- **Say the 1st 3 nodes are in set 1 and the rest are in set 2**
- **Then $M_1 = \frac{1}{K}(6 - \frac{7^2}{K}) = M_2$ where K is the total degree for the graph (including other nodes and links not shown)**

- **What happens if we combine these 2 sets?**
- **Then, the modularity for the 6-node 7-link set will be,**

$$M_{1,2} = \frac{1}{K}(14 - \frac{14^2}{K}) = M_1 + M_2 + \frac{1}{K}(2 - 2 * \frac{7 * 7}{K})$$

and if $K > 49$, the modularity of the combined set will be larger than the sum of the modularities of the 2 sets and a modularity maximization method will prefer the combined set.

- **This simple example is straightforward to generalize.**

Say that:

- L_a **is the number of links connecting nodes within set** a
- $\hat{L}_{a,b}$ **is the number of links that connect nodes in set** a **with nodes in set** b
- K_a **is the number of stubs connected to nodes in set** a **(so,** $K_a \geq 2L_a$**)**
- $M_{a,b}$ **is the modularity of the set of nodes formed by combining sets** a **and** b

Then, $M_a = \frac{1}{K}(2L_a - \frac{K_a^2}{K})$ **and** $M_{a,b} = M_a + M_b + \frac{2}{K}(\hat{L}_{a,b} - \frac{K_a K_b}{K})$

- **If** K **is larger, there will be a tendency** $\hat{L}_{a,b} - \frac{K_a K_b}{K} > 0$ **and for modularity maximization to combine sets** a **and** b**. This can be a problem when working with very large networks**

- **Applied to students in the department, modularity maximization may place you and your friends in a community, but when applied to students in the college, it may combine you and your friends with other communities**

- **A partial solution is to re-apply the method to the smallest communities found**

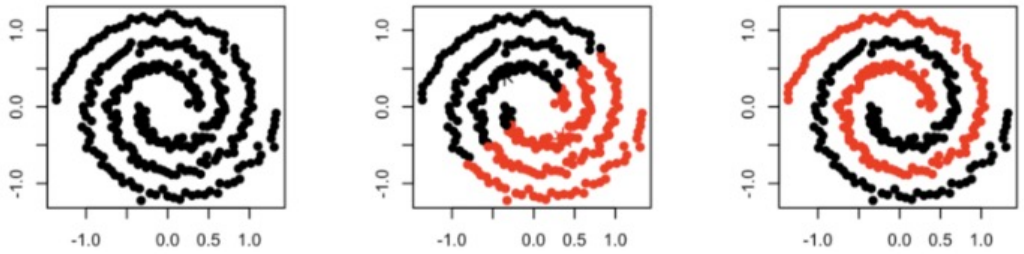
- We have just taken a brief look at community detection
  - Modularity and the spectral method were introduced about 15 years ago and have proven to be hugely influential
  - There have been a large number of other methods that been developed since then
    - The Louvain method maximizes modularity more efficiently than the spectral method (but is much more complicated) – see Barabasi §9.12.1
    - There are methods based on information theory (Barabasi §9.12.2) and statistical inference that are also popular
    - And there are methods that allow nodes to belong to multiple communities (Barabasi §9.5)
  - This is one topic where there is quite a lot of useful information in Barabasi that we will not cover – read through chapter 9 if you are interested and would like to learn more!

Network Science and data science

- **In the last part of this module we will look at problems at the intersection of network science and data science (and computer science)**

- **We will focus on *clustering*, trying to find ways to automatically group data into different clusters. As we will see, there are natural similarities with community detection.**

From Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 888–905.



Two motivating examples:

1. **Image segmentation, automatically identifying "objects" in images**

2. **More general data clustering: given a set of m-dimensional vectors, assign each vector to one of q clusters**

- There are two questions we have to consider:

    1. How do we represent an image or a collection of vectors as a graph?

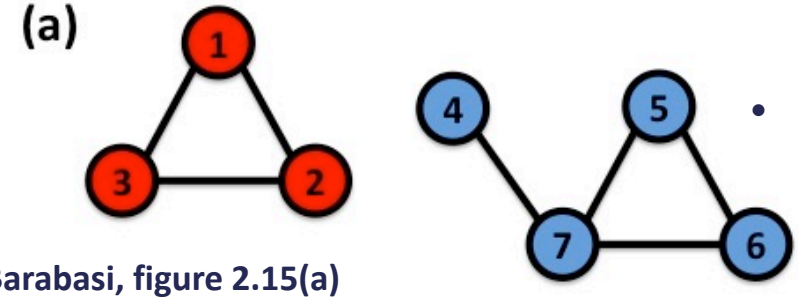
    2. Given a graph representation, how do we construct clusters?

- We will consider the 2nd question first as it naturally follows from our discussion of community detection

- The methods we focus on here take advantage of useful properties of the graph Laplacian whose eigenvalues and eigenvectors can be connected to the structure of the graph.

- First consider a graph with multiple connected components. The number of zero eigenvalues of L is equal to the number of connected components in the graph

- To see how this works, recall that the adjacency matrix of a graph with multiple components can be put in block diagonal form:



Barabasi, figure 2.15(a)

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

- The corresponding Laplacian will also be in block diagonal form

- Then, for the example above, consider the eigenvectors where 1) the first three elements are 1, and all other elements are zero, and 2) the first three elements are zero, and all other are 1. These are linearly independent eigenvectors with eigenvalues=0

- **Are there any other linearly independent eigenvectors? Note that z is equal to the sum of these two vectors and is not linearly independent.**

- **We can show that there are no other linearly independent eigenvectors corresponding to $\lambda = 0$ by showing that all elements of a zero eigenvector corresponding to nodes in a component must be the same – you will be given an exercise which examines this question in greater detail.**

- **It is natural to consider distinct connected components of graphs as distinct clusters. What about the identification of clusters within a connected component? The Laplacian is also helpful for this question and we now consider *Laplacian graph partitioning***

Laplacian graph partitioning

- **The idea behind Laplacian partitioning is to break a connected graph into two groups of nodes where the number of links crossing from one group to the other (the *cut size*, c) is minimized**

- **As we did when considering spectral community detection, we assign each node to one of two groups (a and b) with $s_i = 1$ if node i is in group a and $s_i = -1$ if node i is in group b**

- **Then, $\frac{1}{2}(1 - s_i s_j) = 1$ if nodes i and j are in different groups and is zero if they are in the same group.**

- **It follows that the cut size for a partition is,**

$$c = \frac{1}{4}\sum_{i=1}^{N}\sum_{j=1}^{N} A_{ij}(1 - s_i s_j)$$

which we now want to minimize.

- **The "trick" with Laplacian partitioning is to notice that:**

$$\sum_{i=1}^{N}\sum_{j=1}^{N} A_{ij} = K = \sum_{i=1}^{N}\sum_{j=1}^{N} s_i s_j k_j \, \delta_{ij}$$

- **Then,** $c = \frac{1}{4}\mathrm{s}^{\mathrm{T}}\mathrm{L}s$**. With modularity maximization, we had** $M = \frac{1}{4L}\mathrm{s}^{\mathrm{T}}\mathrm{B}\mathrm{s}$ **and the task was to find** s **such that** M **is maximized with the constraint that each element of** s **is** ± 1**.**

- **Here, the task is to find** s **such that** c **is *minimized* with each element of** s **set to** ± 1

- **As with spectral modularity maximization, we relax this constraint to** $|\tilde{\mathrm{s}}|^2 = N$ **and then we will later convert** $\tilde{\mathrm{s}}$ **to a vector of positive and negative ones**

- **How do we find the minimum of** $\tilde{c} = \tilde{\mathrm{s}}^{\mathrm{T}}\mathrm{L}\tilde{\mathrm{s}}$**? Well, we know that if** $\tilde{\mathrm{s}} = \mathrm{z}$ **then,** $\tilde{c} = 0$**, however this is just the trivial result that all nodes are in the same group.**

- **Instead consider the eigenvector,** $\mathrm{v}_{\mathrm{N}-1}$**, corresponding to the smallest positive eigenvalue of** L**,** λ_{N-1}**. Setting** $\tilde{\mathrm{s}} = \mathrm{v}_{\mathrm{N}-1}$ **(with the length scaled to** $\sqrt{N}$**) we find,** $\tilde{c} = \frac{1}{4}N\lambda_{N-1}$ **and with a bit more work we can show that this is the desired non-zero minimum.**

- **To show that $\tilde{\mathrm{s}} = \mathrm{v}_{N-1}$ is the "correct" choice, we will first expand $\tilde{\mathrm{s}}$ as a weighted sum of the eigenvectors of L:**

$$\tilde{\mathrm{s}} = a_1\mathrm{v}_1 + a_2\mathrm{v}_2 + \cdots + a_N\mathrm{v}_\mathrm{N} = \mathrm{Va}$$

where V is the orthogonal eigenvector matrix for L and $\mathrm{a} = [a_1\ a_2\ \ldots a_N]^T$

- **Then, after orthogonally diagonalizing L, we find, $\tilde{\mathrm{s}}^\mathrm{T}\mathrm{L}\tilde{\mathrm{s}} = \mathrm{a}^\mathrm{T}\mathrm{V}^\mathrm{T}\mathrm{V}\Lambda\mathrm{V}^\mathrm{T}\mathrm{Va} = \mathrm{a}^\mathrm{T}\Lambda\mathrm{a}$ (*)**

- **Say that the eigenvalues are ordered as $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{N-1} > 0$ and rewrite (*) as,**

$$\tilde{c} = \lambda_1 a_1^2 + \lambda_1 a_2^2 + \cdots + \lambda_{N-1} a_{N-1}^2$$

- **The minimum positive $\tilde{c}$ occurs when $a_i = 0$ for $i < N - 1$. We still have to decide what to do with a_N. We set it to zero, otherwise the minimization problem will again lead to the trivial result of all nodes being placed in the same group.**

- The final step could be the same as what we had with the spectral method – we set s by setting all positive elements of $\tilde{\mathrm{s}}$ to one and all negative elements to negative one

- This method tends to produce nice results when $\lambda_{N-1} \ll \lambda_{N-2}$ but it has some obvious shortcomings. E.g. if a node has degree=1, then placing the node in its own group will give the minimum non-zero cut-size

- Additional rules are usually imposed to ensure that one of the groups is not "too small"
  - For example, place all nodes with $\tilde{\mathrm{s}}_i \leq median(\tilde{\mathrm{s}})$ in a group.

- v_{N-1} and λ_{N-1} are known as the ***Fiedler vector*** and the ***algebraic connectivity,*** respectively. A key result from Fiedler is that if we select a real number, y, and place all nodes corresponding to the elements of $\mathrm{v}_{N-1} < y$ in group a, and all other nodes in b, then each group will be connected.

- Next week, we will look at "modern" clustering methods which can be viewed as extensions of Laplacian graph clustering

Lecture 16

Converting data to graphs
Normalized cut size and spectral clustering
Network science and climate science

Converting data into graphs

- In this lecture we will continue our discussion of *clustering*
- Recall that there are two questions to consider:
  1. How do we represent an image or a collection of vectors as a graph?
  2. Given a graph representation of the data, how do we construct clusters?
- We partially answered the 2nd question when we looked at Laplacian graph partitioning. We will briefly discuss two methods which adopt ideas from Laplacian partitioning but are more sophisticated
- But first, let's address the first question above. Let's assume that a dataset can be arranged as a set of N m-element vectors: $D = \{a_1, a_2, \dots, a_N\}$, $a_i \in \mathbb{R}^m$. How do we "convert" this into a graph?

- **Then the simplest "graphical" representation of the data is a weighted *complete* graph where the weight of a link between 2 distinct nodes (data points), W_{ij}, is a function of the distance between the i^{th} and j^{th} vectors in D. The function should convert smaller distances into larger weights, and usually, $W_{ii} = 0$.**

- **Let d_{ij} be the distance between a_i and a_j. Which distance measure should we use?**
  - **The obvious choice is the Euclidean distance: $d_{ij} = \sqrt{(\mathrm{a}_i - \mathrm{a}_j)^T(\mathrm{a}_i - \mathrm{a}_j)}$**
  - **But we may be more interested in the degree of alignment of two vectors in which case the *cosine distance* may be preferred: $d_{ij} = \cos^{-1}(\mathrm{a}_i\mathrm{a}_j/|\mathrm{a}_i||\mathrm{a}_j|)$ where the distance is chosen to be between 0 and π**

- **A black-and-white image can be represented as a set of 1-D vectors where each vector corresponds to the intensity of a pixel.**

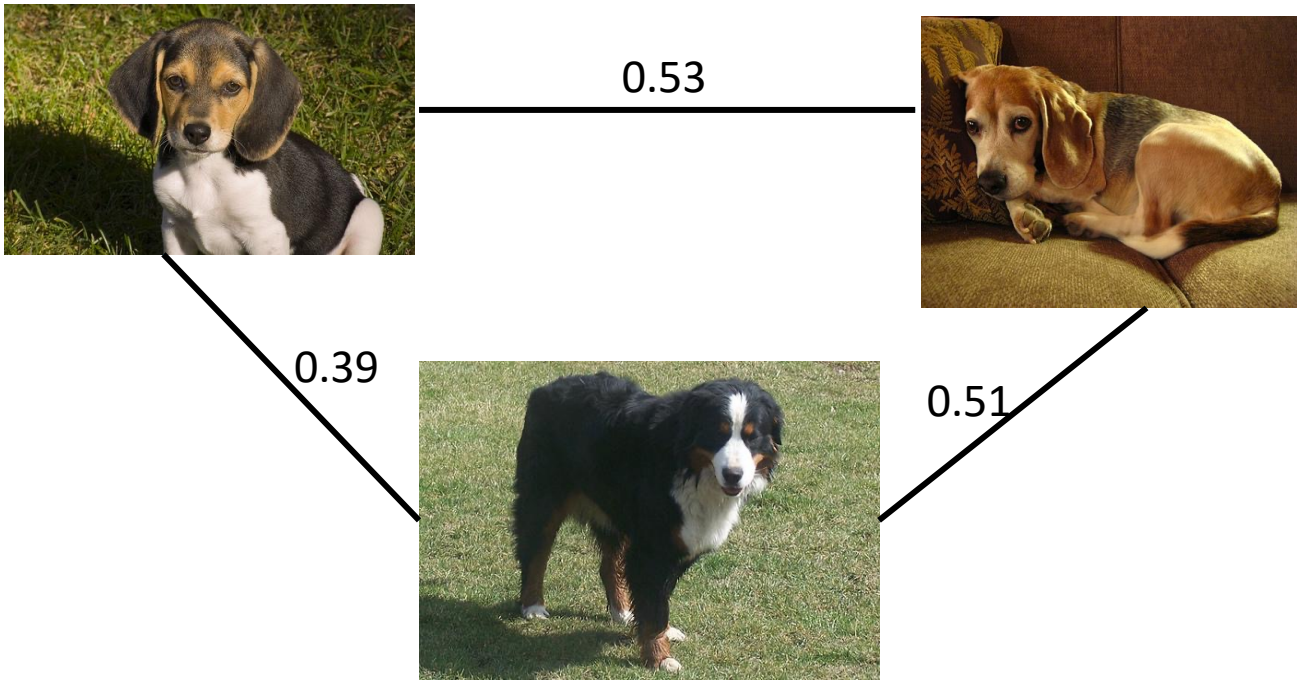
- **An image can also be represented as a single 1-D vector where each element corresponds to a pixel intensity. Then we can construct a network where each node corresponds to an image. Let's look at a very simple example…**

- **Consider the 3 images shown below:**

- **Each image corresponds to a $295 \times 500 \times 3$ matrix of pixel brightnesses scaled to be between 0 and 1**
- **We only use the 1st of the 3 "third dimensions" which correspond to red, green, and blue**
- **And convert each matrix into a 147500 element column vector**
- **Let's use these vectors to form a 3-node weighted graph**

- **Let the vectors $\{a_1, a_2, a_3\}$ correspond to the three images and let d_{ij} be the Euclidian distance between distinct vectors a_i and a_j.**

- **We then use a Gaussian kernel to convert the distances to weights, $W_{ij} = exp\left[-d_{ij}^2/(2\sigma^2)\right]$, where σ is a parameter which I will set to 100 and:**



We can see that the 2 beagles have the highest edge weight, but there is also a comparably high weight for the two "horizontal" dogs.

Image segmentation

- **Let's now return to the first of our two motivating examples:**

1. **Image segmentation, automatically identifying "objects" in images**

From Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 888–905.

- **The image on the left corresponds to an 80×100 matrix of pixel brightnesses. First, we construct a complete weighted graph with $N = 8000$ nodes (each node corresponds to a pixel). Let I_i be the brightness of i and let x_i correspond to the pixel's spatial position. Then the weight for the link connecting distinct nodes i and j is given by:**

$$W_{ij} = \exp\left[-\frac{(I_i - I_j)^2}{\sigma_1}\right] * F_{ij} \text{ with } F_{ij} = \exp\left[\frac{-d_{ij}^2}{\sigma_2}\right] \text{ if } d_{ij} < d_0 \text{ and } F_{ij} = 0 \text{ otherwise.}$$

We also set $W_{ii} = 0$

- **Here, $d_{ij} = |\mathrm{x}_i - \mathrm{x}_j|$ is the Euclidian distance between the two vectors, and σ_1, σ_2, and d_0 are parameters that must be specified. With these weights, two nodes are strongly linked if they have similar pixel brightness and are close together in the original image.**

- **We have specified how to convert the image into a weighted network. How do we construct a partition consisting of two groups of nodes? We will minimize the *normalized cut size*.**

- **Laplacian graph partitioning aimed to minimize the cut size, c, the number of links crossing from a node in one group to a node in the other. However, if a node has just one link, this can lead to the partition where that node is in a group by itself which is not useful. The normalized cut size addresses this problem. The normalized cut size is defined as: $\xi = c\left(\frac{1}{K_a} + \frac{1}{K_b}\right)$. Here, K_a is the total degree for nodes in group a, and K_b is the total degree for nodes in the other group in the partition. Now, if one group has a small number of small-degree nodes, it will be "penalized."**

- **Since we are working with weighted graphs, we have to modify the definitions above. The sum of the weights on links attached to a node will be taken to be its degree and, $\widehat{K}_a = \sum_{i \in S_a} \sum_{j=1}^{N} W_{ij}$ where the outer sum is over all nodes assigned to group a**

- **The cut size will also be redefined to account for the weights.**

- **As before, we define the partition with a vector, s, where $s_i = 1$ if node i is assigned to group a and $s_i = -1$ if node i is in group b**

- **The weighted cut size for the partition is,**

$$\hat{c} = \frac{1}{4}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}(1 - s_i s_j)$$

and the weighted normalized cut size is,

$$\hat{\xi} = \left(\frac{1}{\widehat{K}_a} + \frac{1}{\widehat{K}_b}\right) * \frac{1}{4}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}(1 - s_i s_j)$$

which we now want to minimize. We again relax the problem to finding $\tilde{s} \in \mathbb{R}^N$ with $|\tilde{s}|^2 = N$ such that $\tilde{\xi}$ is minimized where,

$$\tilde{\xi} = \left(\frac{1}{\widehat{K}_a} + \frac{1}{\widehat{K}_b}\right) * \frac{1}{4}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}(1 - \tilde{s}_i \tilde{s}_j)$$

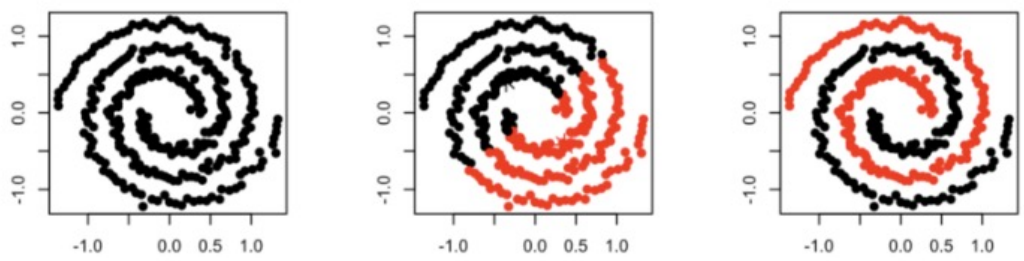
- **Laplacian partitioning required the computation of an eigenvector of the Laplacian. The solution to the normalized cut problem is more complicated:**
  - **First, compute v_{N-1}, the eigenvector corresponding to the second smallest eigenvalue of the weighted normalized Laplacian: $\widehat{\mathrm{L}} = \widehat{\mathrm{D}}^{-\frac{1}{2}}\left(\widehat{\mathrm{D}} - \mathrm{W}\right)\widehat{\mathrm{D}}^{-\frac{1}{2}}$ where $\widehat{\mathrm{D}}$ is a diagonal matrix with $\widehat{D}_{ii} = \sum_{j=1}^{N} W_{ij}$**
  - **Then, compute $\mathrm{y} = \widehat{\mathrm{D}}^{-\frac{1}{2}}\mathrm{v}_{N-1}$ which will be used to construct s**
  - **Choose i^*, a "threshold element" in y, and construct s so that $s_i = 1$ if $y_i > y_{i^*}$ and $s_i = -1$ otherwise. Choose i^* so that the weighted normalized cut size, $\hat{\xi}$, is minimized.**
- **Each node in the graph corresponds to a pixel in the original image, and we can then use the computed partition to construct two images. For example, for each pixel where $s_i = -1$, set the pixel intensity in the image matrix to zero (black); leave all other elements in the matrix as they were.**

From Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation.
IEEE Transactions on Pattern Analysis and Machine Intelligence, *22*(8), 888–905.

- **The images in the middle and on the right were constructed using this approach (the authors used $\sigma_1 = 0.1$, $\sigma_2 = 4$, and $d_0 = 5$ to construct the weighted graph)**

- **I have described the algorithm, but I have not explained why it is correct. You will be given a few exercises which will fill in some of the details.**

Spectral clustering



- Our 2nd motivating example is related to the image above. Given a set of N 2-dimensional vectors, assign each vector to one of two clusters

- The image in the middle was constructed using *K-means clustering*. For this class, all you need to know about this method is that given a set of N m-dimensional vectors and a positive integer, K, it assigns each vector to exactly one of K clusters.
  - It is a simple but very well-known method. It is easy to find more information online if you are interested (e.g. https://en.wikipedia.org/wiki/K-means_clustering)

- However, we can see that the middle image is not that useful. The image on the right is better and was generated using *spectral clustering* which combines K-means clustering with ideas taken from Laplacian partitioning

- We will take a very brief look here at spectral clustering. I just want to give a sense of how it is connected to Laplacian partitioning, but a detailed analysis will not be presented

- Our dataset will be represented as a set of vectors, $\{\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_\mathrm{N}\}, \mathrm{x}_i \in \mathbb{R}^m$. The first step is to construct a weighted graph from the N vectors,. The general approach is similar to what we have already discussed and is based on the Euclidean distance between two vectors, $d_{ij} = |\mathrm{x}_i - \mathrm{x}_j|$. The weight matrix for the graph is defined as:

$$W_{ij} = \exp\left(-\frac{d_{ij}^2}{2\sigma^2}\right) \text{ if } i \neq j \text{ and } W_{ii} = 0$$

- After constructing, W, construct $\widehat{\mathrm{D}}$ (the diagonal matrix with $\widehat{D}_{ii} = \sum_{j=1}^{N} W_{ij}$) and $\widehat{\mathrm{A}} = \widehat{\mathrm{D}}^{-\frac{1}{2}}\mathrm{W}\widehat{\mathrm{D}}^{-\frac{1}{2}}$.

- Compute the orthogonal eigenvectors of $\widehat{\mathrm{A}}$ corresponding to the K largest eigenvalues of $\widehat{\mathrm{A}}$ and collect them in a N x K matrix, X.

- Construct a new dataset of N K-dimensional vectors, $\{\mathrm{r}_1, \mathrm{r}_2, \dots, \mathrm{r}_N\}$, where r_i is the i^{th} ***row*** of X, and apply K-means clustering to this dataset (after normalizing each vector to have unit length).

- **Spectral clustering is a little different from the methods we have previously discussed:**
  - **It doesn't work directly with a Laplacian (or Laplacian-like) matrix, however the eigenvectors and eigenvalues of $\widehat{\mathrm{A}}$ can be related to those of $\widehat{\mathrm{L}}$ (problem sheet exercise)**
  - **It requires a set of eigenvectors rather than a single one and doesn't disregard eigenvectors where all non-zero elements are the same**
  - **Consider a weighted graph with multiple connected components. Then it can be shown that vectors r_i and r_j will be identical if nodes i and j are in the same component and will be orthogonal otherwise (we previously discussed a similar property for eigenvectors of the Laplacian matrix)**
  - **We won't go into why this then leads to good results when K-means clustering is applied. This is an example of a potential topic for further study which follows naturally from this class.**
- **There are also other important methods in machine learning which use ideas from Network Science (e.g. graph neural networks)**

Network science and climate science

- We will conclude this lecture and the class with a brief discussion of a 2017 study which used tools from network science to analyze climate dynamics:

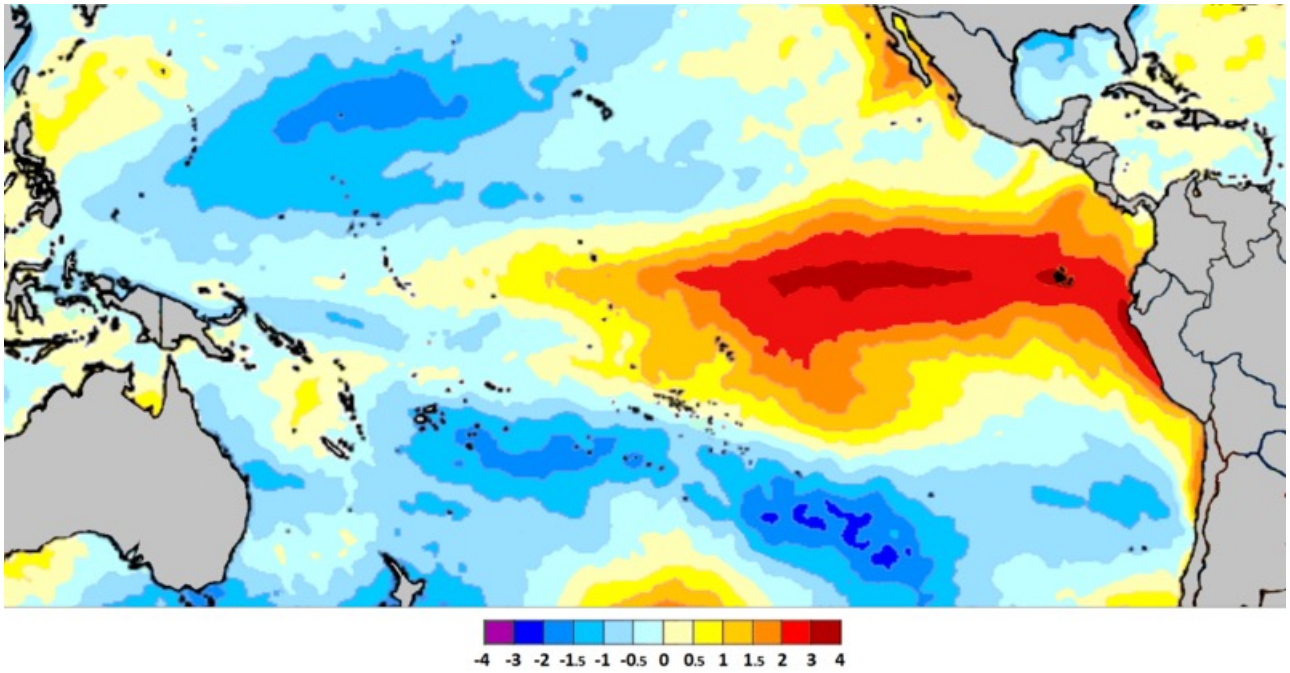
Network analysis reveals strongly localized impacts of El Niño

Jingfang Fan[a,1], Jun Meng[a,b,1], Yosef Ashkenazy[b,2], Shlomo Havlin[a], and Hans Joachim Schellnhuber[c,d,2]

[a]Department of Physics, Bar-Ilan University, Ramat-Gan 52900, Israel; [b]Department of Solar Energy & Environmental Physics, Blaustein Institutes for Desert Research, Ben-Gurion University of the Negev, Midreshet Ben-Gurion 84990, Israel; [c]Potsdam Institute for Climate Impact Research, 14412 Potsdam, Germany; and [d]Santa Fe Institute, Santa Fe, NM 87501

- The main points we will touch upon are:
  - What is El Nino?
  - What is the dataset used for the analysis?
  - How is a network constructed from the dataset?
  - How is the network analyzed and what does the analysis tell us?

- **What is El Nino?**
  - **El Nino refers to periods with abnormally high water temperatures in the central and eastern tropical Pacific ocean. This affects temperature and rainfall globally.**
  - **El Nino events typically last for a few years and then "go away" for a few years**



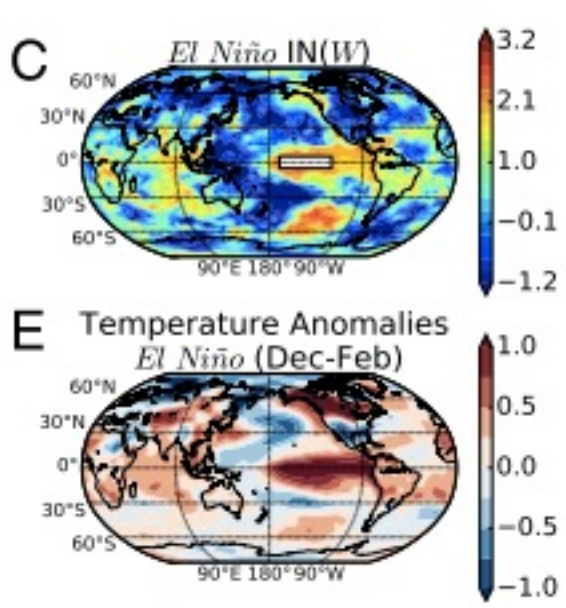
https://commons.wikimedia.org/wiki/File:El_Ni%C3%B1o_1982-83.png

- **What is the dataset used for the analysis?**
  - **The dataset corresponds to daily near-surface temperature from 1948-2016**
  - **It is constructed via a mix of measurements and simulations**
  - **Daily temperatures are available for 10512 regions on the Earth's surface.**
    - **57 of these regions are within the *El Nino Basin***

- **How is a network constructed from the dataset?**
  - **Each of the 10512 regions corresponds to a node and we construct a network for a 365-day period**
  - **A weighted directed link is created for each distinct node pair**
  - **Let $T_i(t_a)$ be the temperature at node i on day t_a. Define the annual *time* average as:**
    $\overline{T}_i = \frac{1}{365}\sum_{a=1}^{365} T_i(t_a)$
  - **The link between i and j is constructed using the cross-correlation between T_i and T_j:**
    $C_{ij}(\tau) = \frac{\frac{1}{365}\sum_{a=1}^{365}[T_i(t_a)T_j(t_a-\tau)]-\overline{T}_i\overline{T}_j}{\sigma_i\sigma_j}$
    - **σ_i is the standard deviation of $T_i(t_a)$, and σ_j is the standard deviation of $T_j(t_a-\tau)$**
    - **The time delay, τ, is varied between 0 and 200 and let τ^* be the value of τ for which $|C_{ij}(\tau)|$ is maximized. Let this maximum be C^*.**

- **The sign of τ^* tells us the direction of the link:**
  - **If $\tau^* \geq 0$, $W_{ij} = \frac{C^* - mean(C_{ij})}{std(C_{ij})}$ (and $W_{ji} = 0$)**
  - **If $\tau^* < 0$, $W_{ji} = \frac{C^* - mean(C_{ij})}{std(C_{ij})}$ (and $W_{ij} = 0$)**
- **We also set $W_{ii} = 0$ and this weight matrix defines the network to be analyzed (the paper also considers one other network which we will ignore)**

- **How is the network analyzed and what does the analysis tell us?**
  - **A simple quantity to analyze is the weighted in-degree for regions outside of the El Nino basin: $\sum_{j \in ENB} W_{ij}$ where the sum is over nodes within the basin. This provides a simple view of which regions are most strongly affected by El Nino.**



The top figure shows these in-degrees and there is a tangible (imperfect) correlation with regions with abnormal temperatures during N. American winter

- **Community detection (or clustering) methods can be applied to identify groups of regions which show unusually similar dynamics**

- **The 2017 study uses community detection differently. It constructs a weighted network where each node corresponds to a *year* in which El Nino occurred, and the link weight indicates how similar the two years were. Then modularity maximization (via the Louvain method) was used to construct three communities which represent three "types" of El Nino events (i.e. events where different areas are strongly influenced)**

- **Among the study's main conclusions are the identification of these different types of El Ninos and the observation that El Nino periods are characterized by stronger temperature anomalies in more localized regions**