Introduction to Networks

Networks

We now consider a new kind of data, **networks**, which are represented by **graphs.**

Motivation

Graphs allow us to represent and analyze complex relationships and structures in data.

By using nodes (vertices) and edges (connections), graphs can model various types of relationships and interactions, making it easier to visualize and understand the underlying patterns.

Example Applications in Machine Learning

- **Social Network Analysis**
  - Model social networks with nodes representing individuals and edges representing relationships or interactions.
  - Understand community structures, influence, and information spread.
- **Biological Network Analysis**
  - Model biological systems, such as protein-protein interaction networks.
  - Understand cellular processes and disease mechanisms.
- **Knowledge Graphs**
  - Store and retrieve structured information.
  - Enable better search and question-answering systems.

Graphs

A graph $G = (V, E)$ is a pair, where V is a set of **vertices**, and E is a set of unordered vertex pairs (u, v) called **edges**.

The term **nodes** is also used for vertices. The term **links** or **connections** is also used for edges.

We'll distinguish between **undirected** graphs and **directed** graphs.

NetworkX

We'll use the NetworkX library to create and manipulate graphs.

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

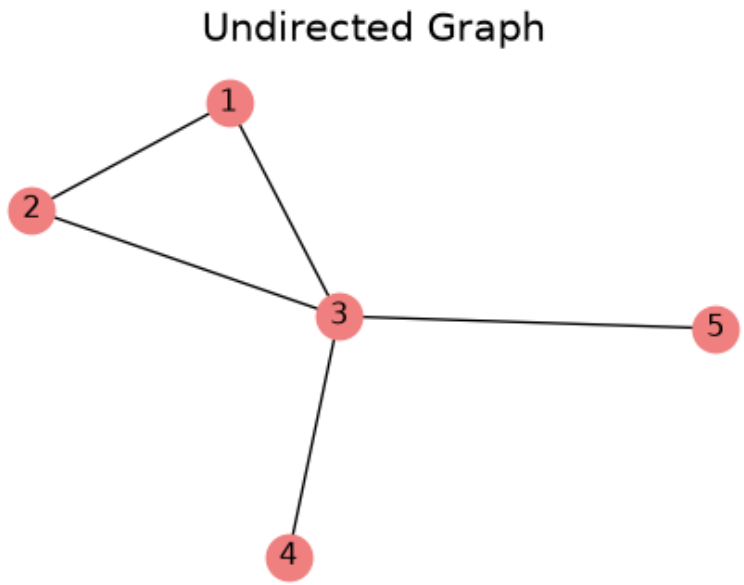
```
1 import networkx as nx
```

Undirected Graphs

In an undirected graph, an edge (u, v) is an unordered pair. The edge (v, u) is the same thing.

There are no orientations in an undirected graph.

▶ Code



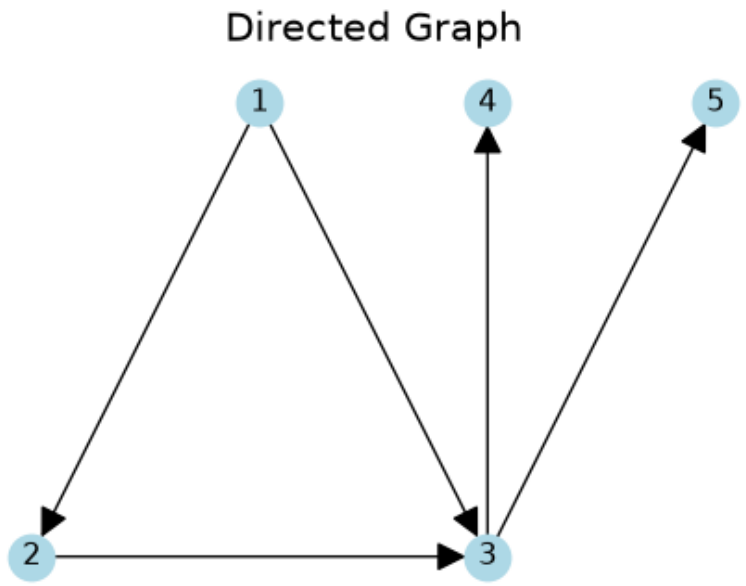
Directed Graphs

In a directed graph, (u, v) is an ordered pair and it is different from (v, u).

This means that the edges have an orientation. This orientation is indicated by arrows.

For example, the edge $(1, 2)$, is directed from node 1 to node 2.

▶ Code



Paths

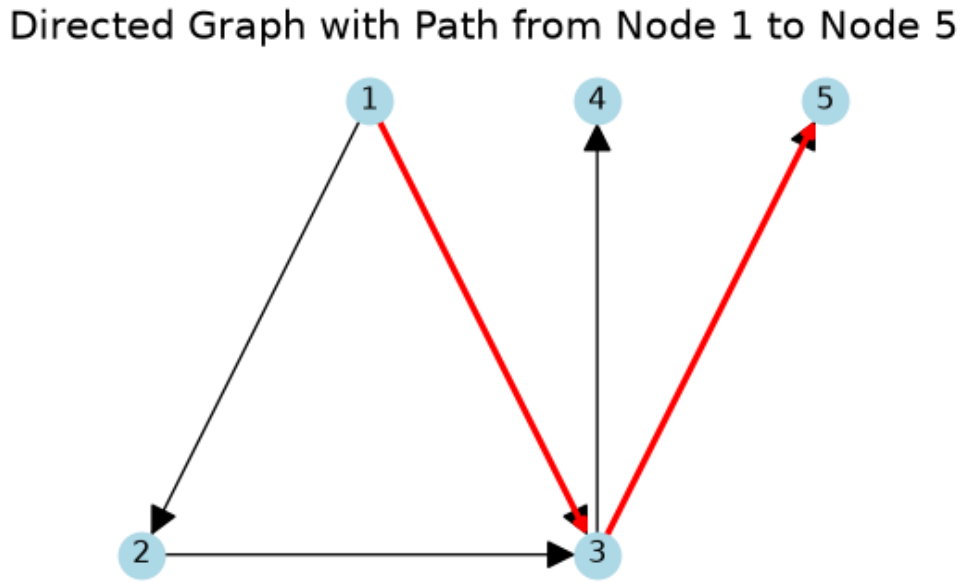
A **path** in a graph from node u to node v is a sequence of edges that starts at u and ends at v. Paths exist in both undirected and directed graphs.

In a directed graph, all of the edges in a path need to be oriented head-to-tail.

If there is a path from u to v, we say that v is **reachable** from u.

A path from node 1 to node 5 is illustrated in red.

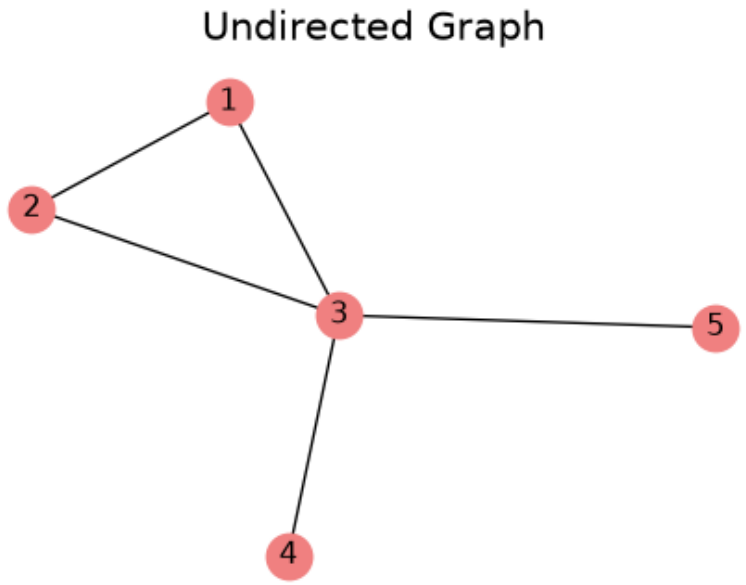
▶ Code



Degree

The **degree** of a node is the number of edges that connect to it.

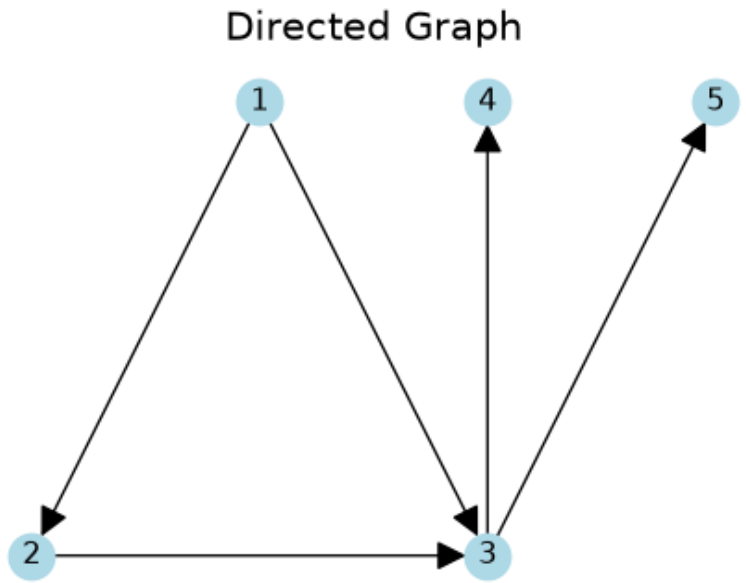
In our example undirected graph, node 3 has degree 4.



In a directed graph, we distinguish between:

- **in-degree**: the number of incoming edges to the node,
- **out-degree**: the number of outgoing edges to the node.

In our directed graph example, the in-degree of node 3 is 2 and the out-degree is 2. For node 1, the in-degree is 0, and the out-degree is 2.



In an undirected graph with n nodes and e edges, the average node degree is $2e/n$. Why?

Neighbors

The **neighbors** of a node are the nodes to which it is connected.

In an undirected graph, the degree of a node is the number of neighbors it has.

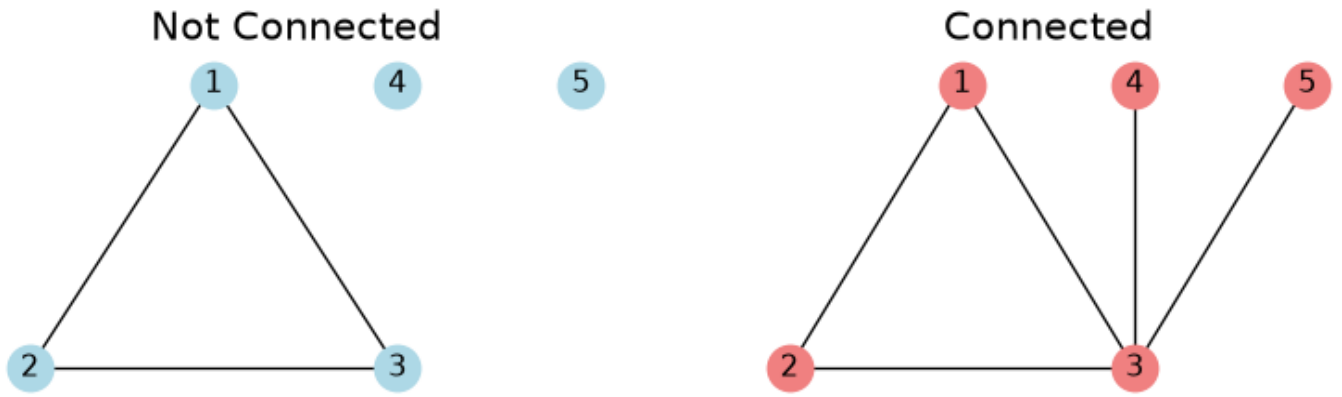
A directed graph has both outgoing and incoming neighbors.

Connectivity

The first question to ask about a graph is: is it **connected**?

An undirected graph is connected, if for each pair of nodes (u, v), u is reachable from v. A directed graph is connected when its undirected version is connected.

▶ Code



If the graph is not connected, it may contain **connected components**.

A connected component is a subgraph that is connected. The above graph on the left has a connected subgraph

In a directed graph, we can also ask if it is **strongly connected.**

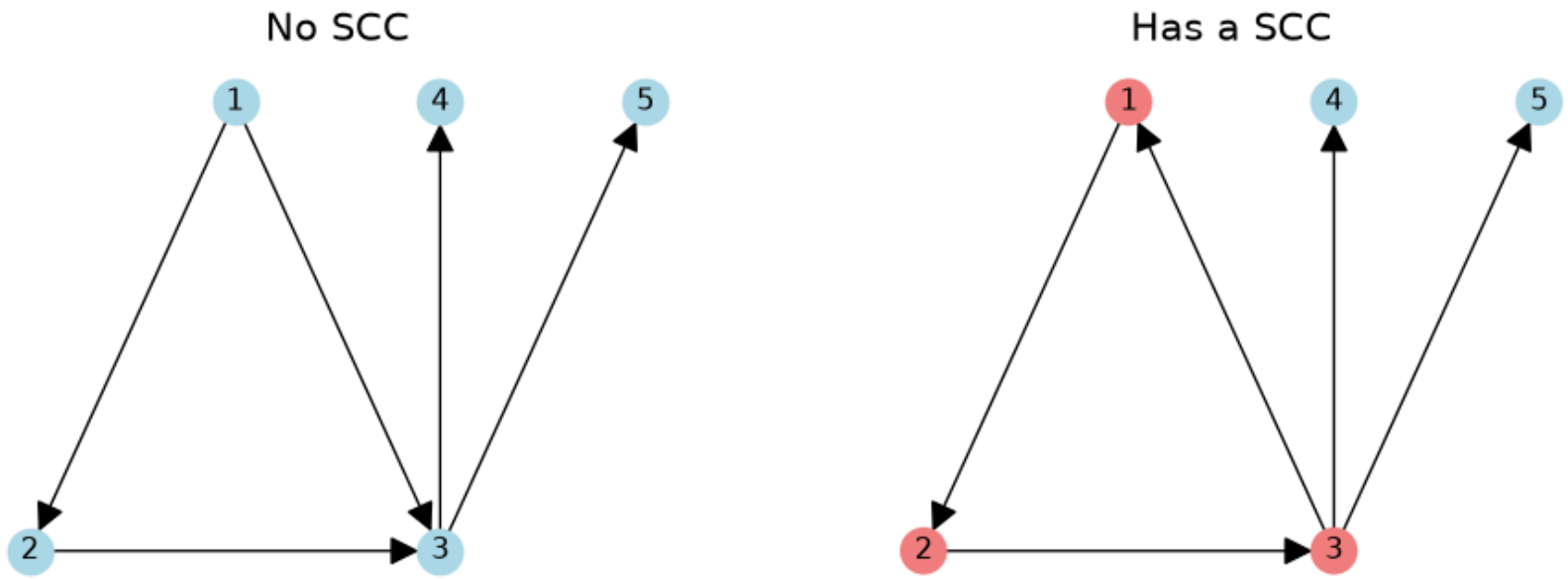
A directed graph is strongly connected if there is a (directed) path between any two nodes.

That is, any node is reachable from any other node.

Within a directed graph, only a subset of nodes may be strongly connected.

These are called the **strongly connected component** (SCC).

▶ Code



Characterizing Graphs

Given a graph, it is helpful to characterize the degrees, components, and other structures in the graph.

Comparison Case: the $G(n, p)$ Random Graph

The most common comparison is the $G(n, p)$ **random graph**. It is also called the **Erdős–Rényi** graph, after the two mathematicians who developed and studied it.

The $G(n, p)$ random graph model is very simple

- we start with a set of n nodes,
- for each pair of nodes, we connect them with probability p.

Average Node Degree of a Random Graph

Every node is potentially connected to $n - 1$ other nodes.

Therefore, the expected degree, call it $\langle k \rangle$, of a node is given by:

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

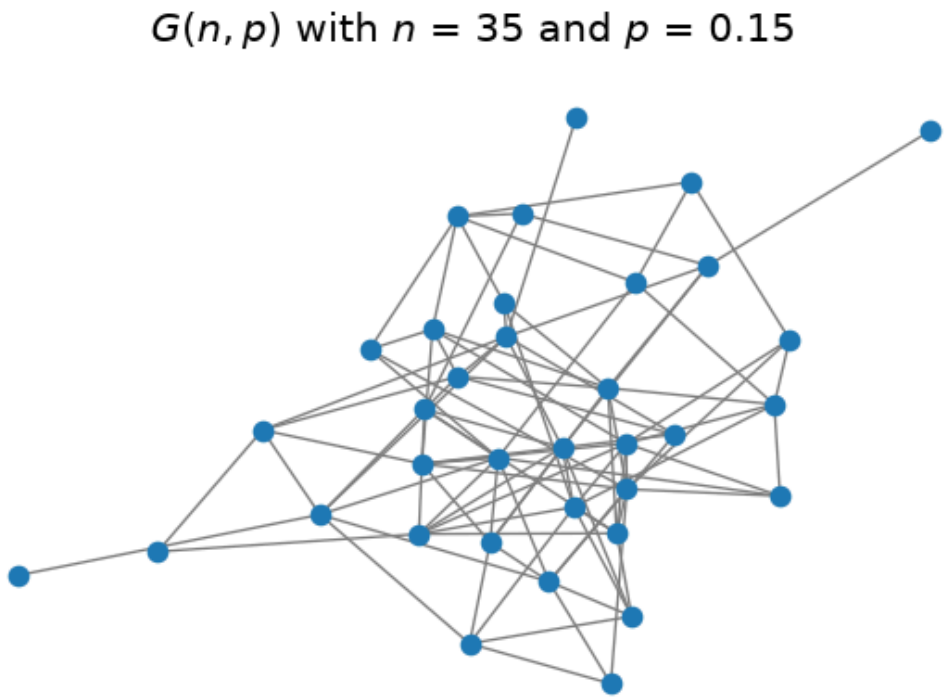
For large n, this can be approximated as:

$$\langle k \rangle \approx np$$

So for random graphs, the average node degree is np.

In this graph, the average degree is $np = 35 \cdot 0.15 = 5.25$

▶ Code



Most real-world graphs do **not** match the properties of $G(n, p)$ graphs, however it is useful to have a comparison to a *random* graph.

Degree Distributions

Understanding connectivity starts with determining the observed degrees in the graph.

This is captured in the **degree distribution**

$$P(X > x) = \text{probability that a node has degree at least } x.$$

We typically focus our attention on large values of x – nodes that are highly connected.

Power Law Degree Distributions

It's common for a degree distribution to **approximately** follow a power-law.

The simplest power-law distribution is called the Pareto distribution. If X is a random variable with a Pareto distribution, then the probability that X is greater than some number x is given by:

$$P(X > x) = \begin{cases} \left(\frac{k}{x}\right)^{\alpha} & x \geq k, \\ 1 & x < k \end{cases}$$

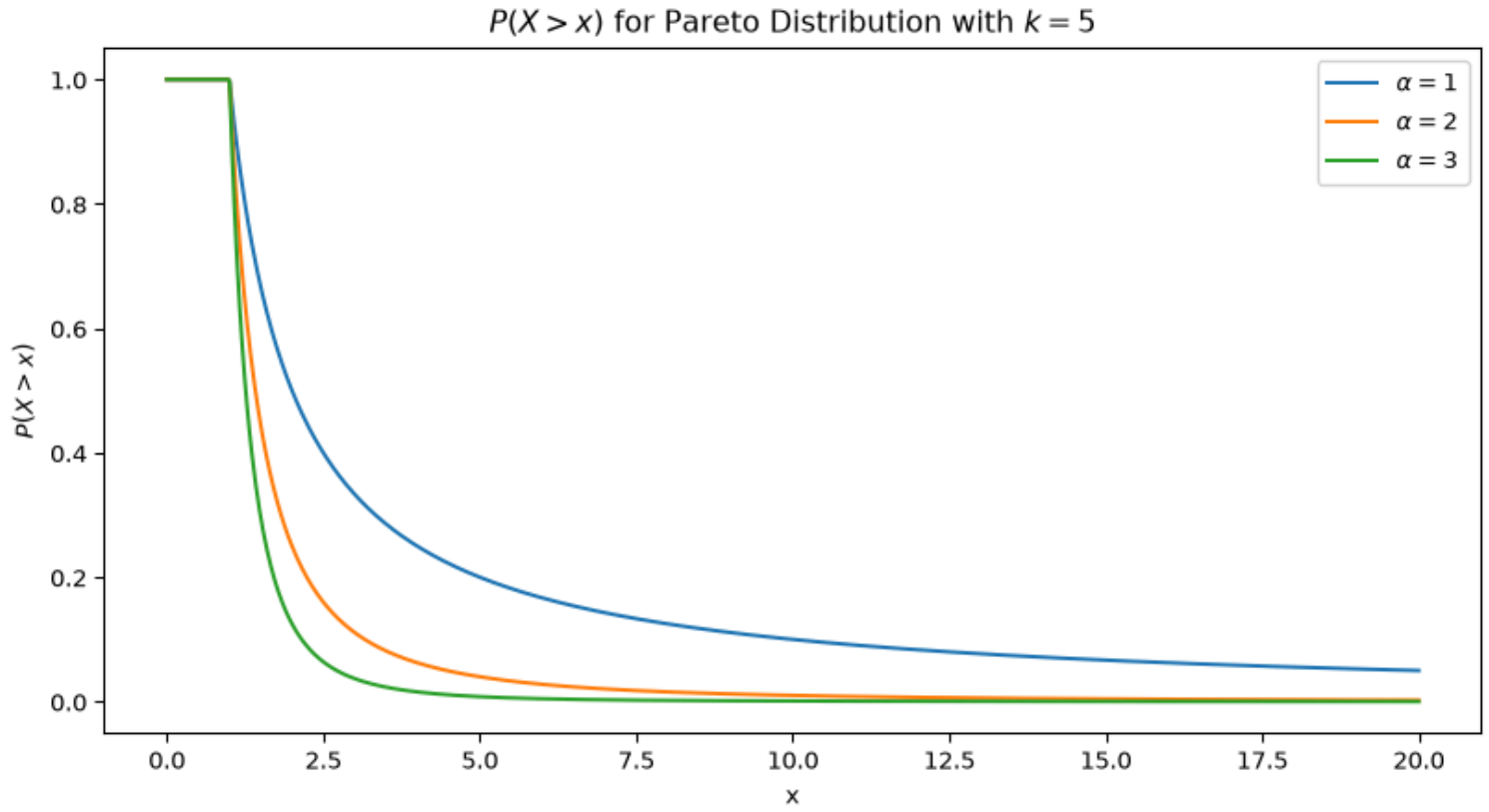
where k is the minimum value of X, sometimes called the **scale parameter**, meaning we can specify the *minimum* number of neighbors a node can have.

And α is the exponent, sometimes called the **shape parameter**.

Distribution Plot

Here is a plot for different values of α and $k = 1$.

▶ Code

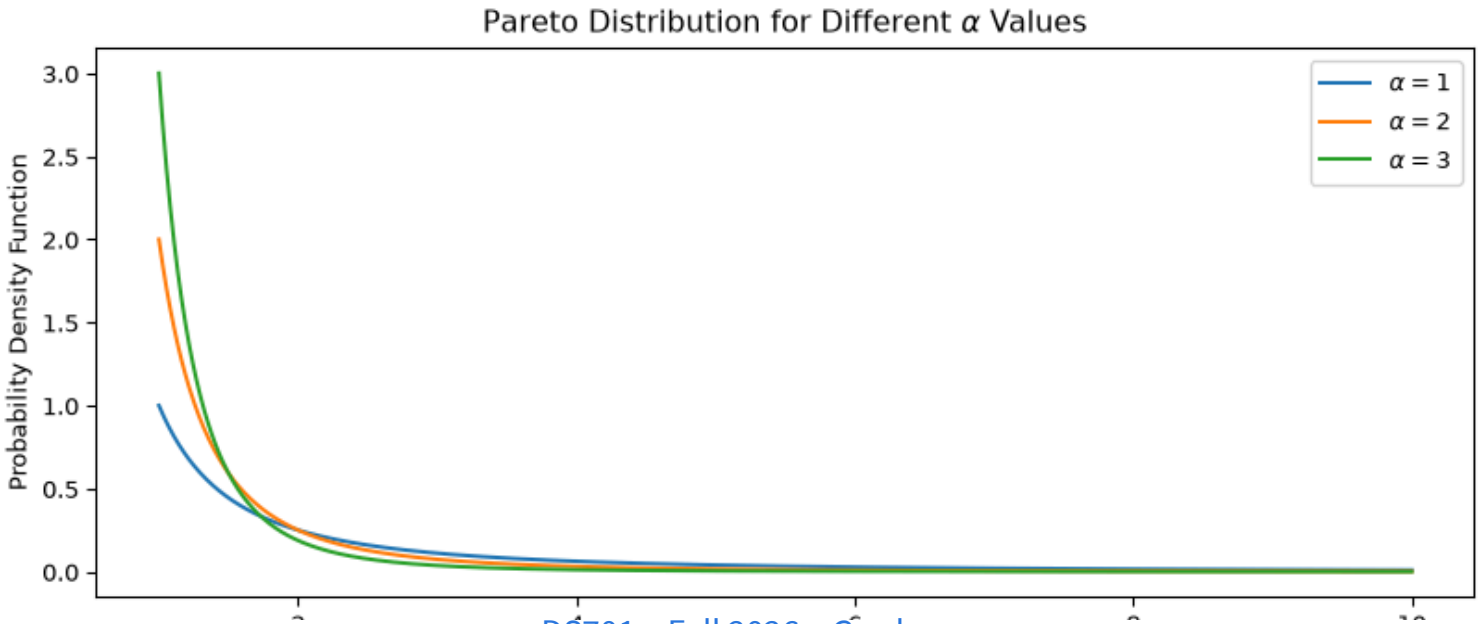


The pdf of the Pareto distribution is

$$p(x) = \begin{cases} \frac{\alpha k^\alpha}{x^{\alpha+1}} & x \geq k, \\ 0 & x < k. \end{cases}$$

Here is a plot of the Pareto probability density function for different values of α and $k = 1$.

▶ Code



In a distribution like this, almost all values are very small. However, there is a non-negligible fraction of values that are **very** large.

Vilfredo Pareto originally used this distribution to describe the allocation of wealth among individuals. The size of sand particles are also seen as approximately Pareto-distributed.

What does this mean for node degree?

It means that

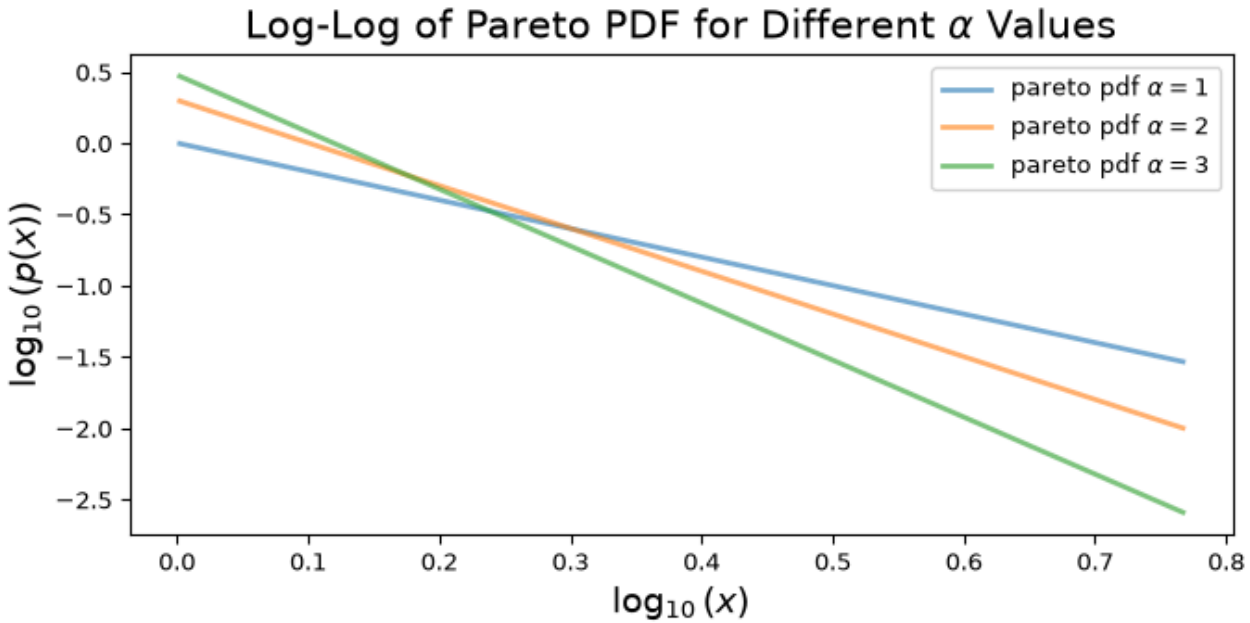
- most nodes have **few neighbors,** but
- an important small subset of nodes have **many, many neighbors.**

To capture such high-variable degree distributions, a common strategy is to plot them on **log-log** axes.

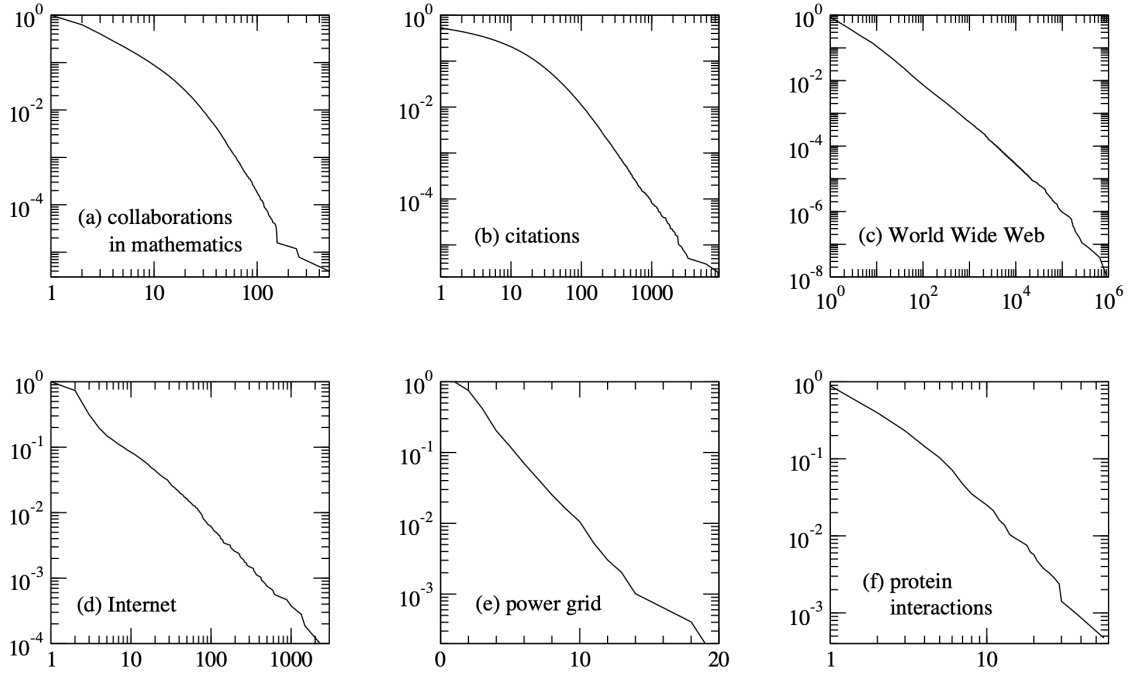
On log-log axes, a Pareto distribution appears as a straight line. You can verify this yourselves by computing

$$\log(p(x)) = \log(\alpha k^{\alpha}) - (\alpha + 1)\log(x).$$

▶ Code



Power Law Degree Distributions are Ubiquitous



> The networks shown are: (a) the collaboration network of mathematicians [182]; (b) citations between 1981 and 1997 to all papers cataloged by the Institute for Scientific Information [351]; (c) a 300 million vertex subset of the World Wide Web, circa 1999 [74]; (d) the Internet at the level of autonomous systems, April 1999 [86]; (e) the power grid of the western United States [416]; (f) the interaction network of proteins in the metabolism of the yeast S. Cerevisiae [212].
>
> *The structure and function of complex networks,* M. E. J. Newman
>
> https://arxiv.org/abs/cond-mat/0303516

Note that the x-axis of the power grid example is a linear scale but the y-axis is a log scale. This means that the degree distributions of the power grid example does not follow a power-law degree distribution. It has an exponential tail.

Clustering

The next important property of a network to understand is **clustering**.

In the context of networks, clustering refers to the tendency for groups of nodes to have higher connectivity within the group than the network-wide average.

The simplest measure of local clustering is **clustering coefficient**.

The clustering coefficient tells you if the neighbors of a node tend to be neighbors.

Specifically, the clustering coefficient measures the **probability that two of your neighbors are connected.**

To define this measure, we introduce the notion of a triangle in a network. A triangle is a set of three nodes in the graph that are connected to each other.

There are two definitions of the clustering coefficient. The first is

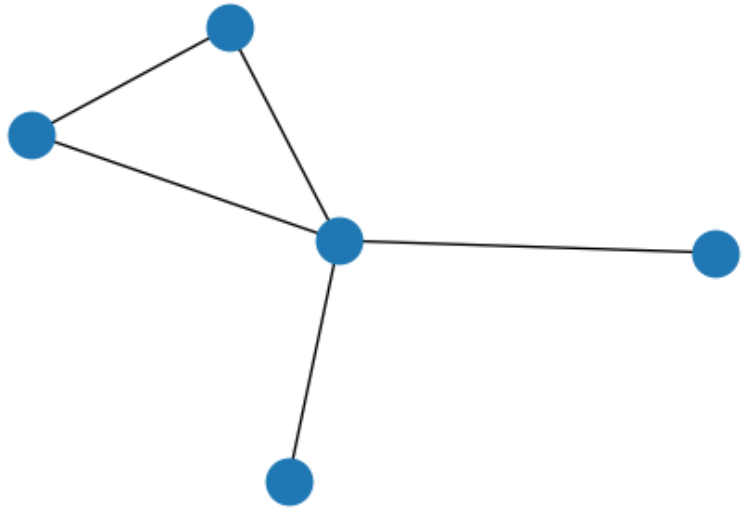
$$C^{(1)} = \frac{\sum_i \text{number of triangles that include node } i}{\sum_i \text{number of pairs of neighbors of node } i}.$$

This is the ratio of the mean triangle count to mean neighbor pair count.

It is the probability that a **random pair** of neighbors are connected.

Consider the following example graph.

▶ Code



The clustering coefficient $C^{(1)} = \frac{3}{1+1+\binom{4}{2}} = \frac{3}{1+1+6} = \frac{3}{8} = 0.375,$

where $\binom{4}{2} = \frac{4!}{2!(4-2)!}$.

The Example Graph Calculation

The graph has:

- **Nodes**: 1, 2, 3, 4, 5
- **Edges**: (1,2), (1,3), (2,3), (3,4), (3,5)

This creates a triangle with nodes 1-2-3, and node 3 connects to two additional nodes (4 and 5).

Step-by-Step Calculation

The clustering coefficient formula being used is:

$$C^{(1)} = \frac{\sum_i \text{number of triangles that include node } i}{\sum_i \text{number of pairs of neighbors of node } i}$$

Let's count for **each node**:

Node 1

The second way to measure the clustering coefficient is the mean of the ratios, i.e.,

$$C^{(2)} = \frac{1}{n} \sum_i \frac{\text{number of triangles that include node } i}{\text{number of pairs of neighbors of node } i},$$

where n is the total number of nodes.

This is the probability that neighbors are connected for a **random node**.

In the previous example, $C^{(2)} = \frac{1}{5}\left(1 + 1 + \frac{1}{6}\right) = \frac{13}{30} = 0.433$.

What is the clustering coefficient in a $G(n, p)$ random graph?

The probability that two of your neighbors are connected is the same as the probability that **any** two nodes are connected, i.e., $C^{(1)} = C^{(2)} = p$.

Real World Graphs

In practice, real world graphs show strong clustering.

For example, consider a social network. Your friends are much more likely to be themselves friends than two randomly chosen people.

Table 1: Clustering coefficients, C, for a number of different networks; n is the number of node, z is the mean degree. Taken from [146].

Network	n	z	C measured	C for random graph
Internet [153]	6,374	3.8	0.24	0.00060
World Wide Web (sites) [2]	153,127	35.2	0.11	0.00023
power grid [192]	4,941	2.7	0.080	0.00054
biology collaborations [140]	1,520,251	15.5	0.081	0.000010
mathematics collaborations [141]	253,339	3.9	0.15	0.000015
film actor collaborations [149]	449,913	113.4	0.20	0.00025
company directors [149]	7,673	14.4	0.59	0.0019
word co-occurrence [90]	460,902	70.1	0.44	0.00015
neural network [192]	282	14.0	0.28	0.049
metabolic network [69]	315	28.3	0.59	0.090
food web [138]	134	8.7	0.22	0.065

Clustering and Path Length: Small Worlds

The strong presence of clustering in networks leads to a question.

How long is a typical shortest path between nodes?

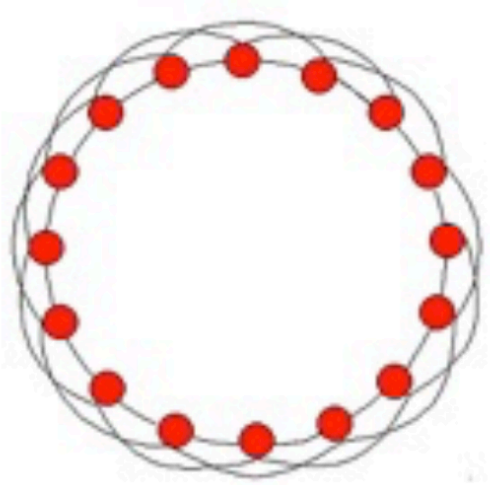
The average shortest path length between nodes is one way to measure a network's *diameter*.

If $d(i, j)$ represents the shortest path length between nodes (i) and (j), and N is the total number of nodes in the network, the average shortest path length L can be expressed as

$$L = \frac{1}{\binom{N}{2}} \sum_{i \neq j} d_{\text{shortest}}(i, j).$$

In this formula $\binom{N}{2}$ is the number of unique pairs of nodes. The sum $\sum_{i \neq j} d_{\text{shortest}}(i, j)$ is taken over all pairs of nodes (i) and (j).

Let's consider a highly clustered graph.



In this graph, each node has 4 neighbors. This means there are $\binom{4}{2} = 6$ possible pairs of neighbors. Of these 6 neighbors, 3 are connected, yielding a clustering coefficient of 0.5. This is quite high.

Real-world social networks are highly clustered. Based on this model, we might assume that the average path length between two people in a large social network is going to be quite large. Is this really true?

This statistic became famous when John Guare wrote a play called *Six Degrees of Separation*.

Given what we know about clustering in social networks, this is quite surprising. How can we explain it?

The first clue comes from another classic social science paper, called *The Stength of Weak Ties,* by Mark Granovetter.

This is sometimes referred to as the most famous paper in sociology (with over 60,000 citations).

Granovetter interviewed people about how they found their jobs. He found that most people did not get a job through someone that was a close friend, but rather through a **distant** acquaintance.

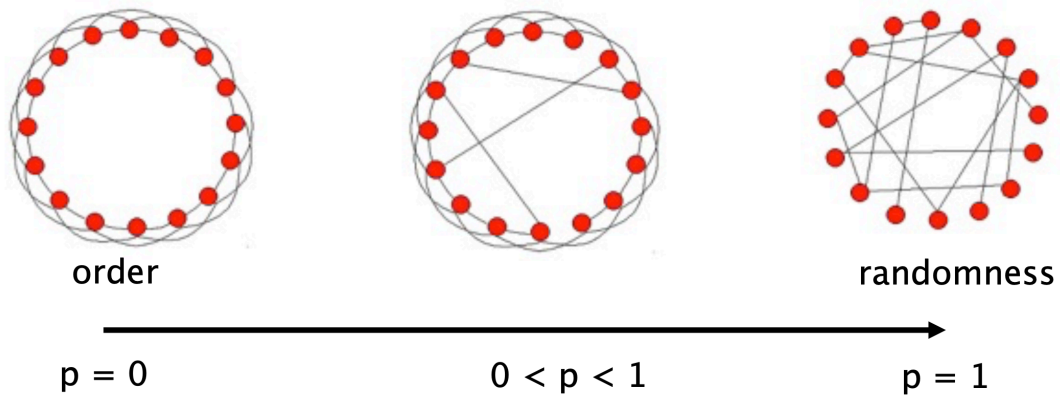
This suggests that an important way that information travels in a social network is via the rare connections that exist **outside** of the local clustering of friendships.

This was all put on an experimental basis in **another** classic paper, by the social scientist Duncan Watts and the mathematician Steve Strogatz.

In their paper *Collective Dynamics of Small-World Networks*, Watts and Strogatz perfomed an elegant experiment.

They asked, what if we take a highly clustered network, and slightly perturb it?

Specifically, they started with a network in which each node is connected to a fixed number of neighbors, and connections are made in a **highly clustered** way.

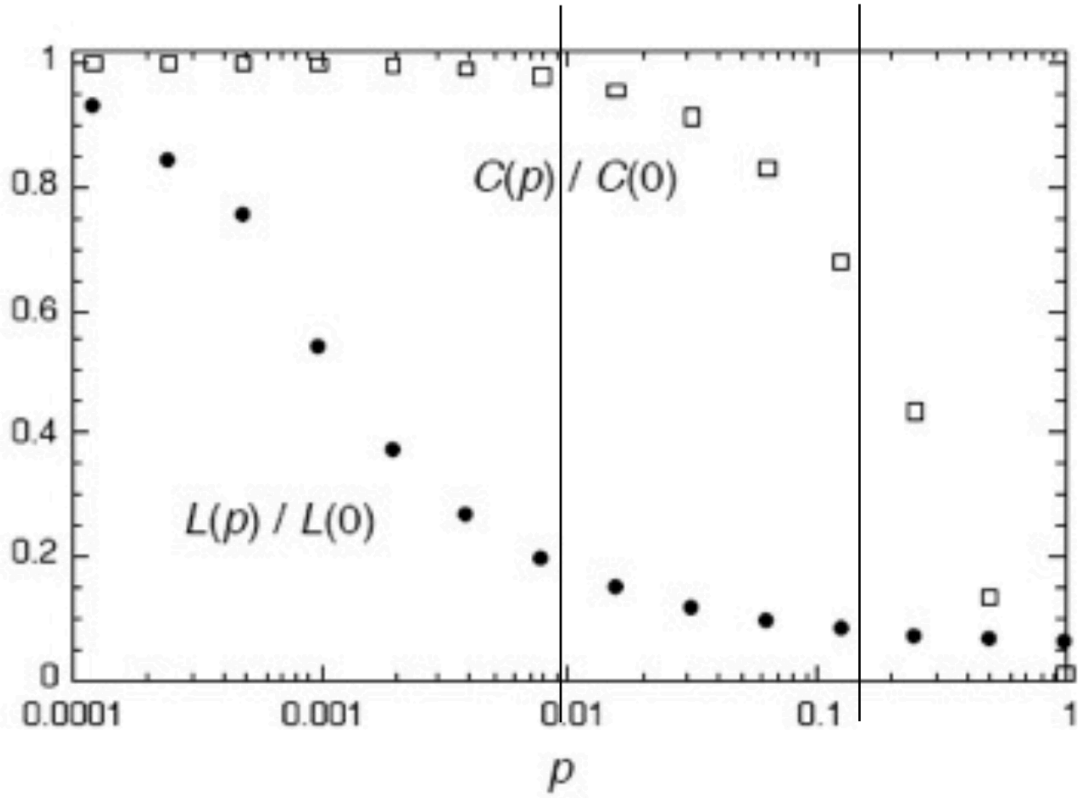


Then, with probability p, **rewire** each edge: change it to connect to a **random** destination.

As p varies, what happens to

- the average path length $L(p)$ and
- the clustering coefficient $C(p)$?

Here is the famous figure from that paper. Observe the log scale on the p axis. The values of both the average path length and the clustering coefficient are plotted, normalized by their values when $p = 0$.



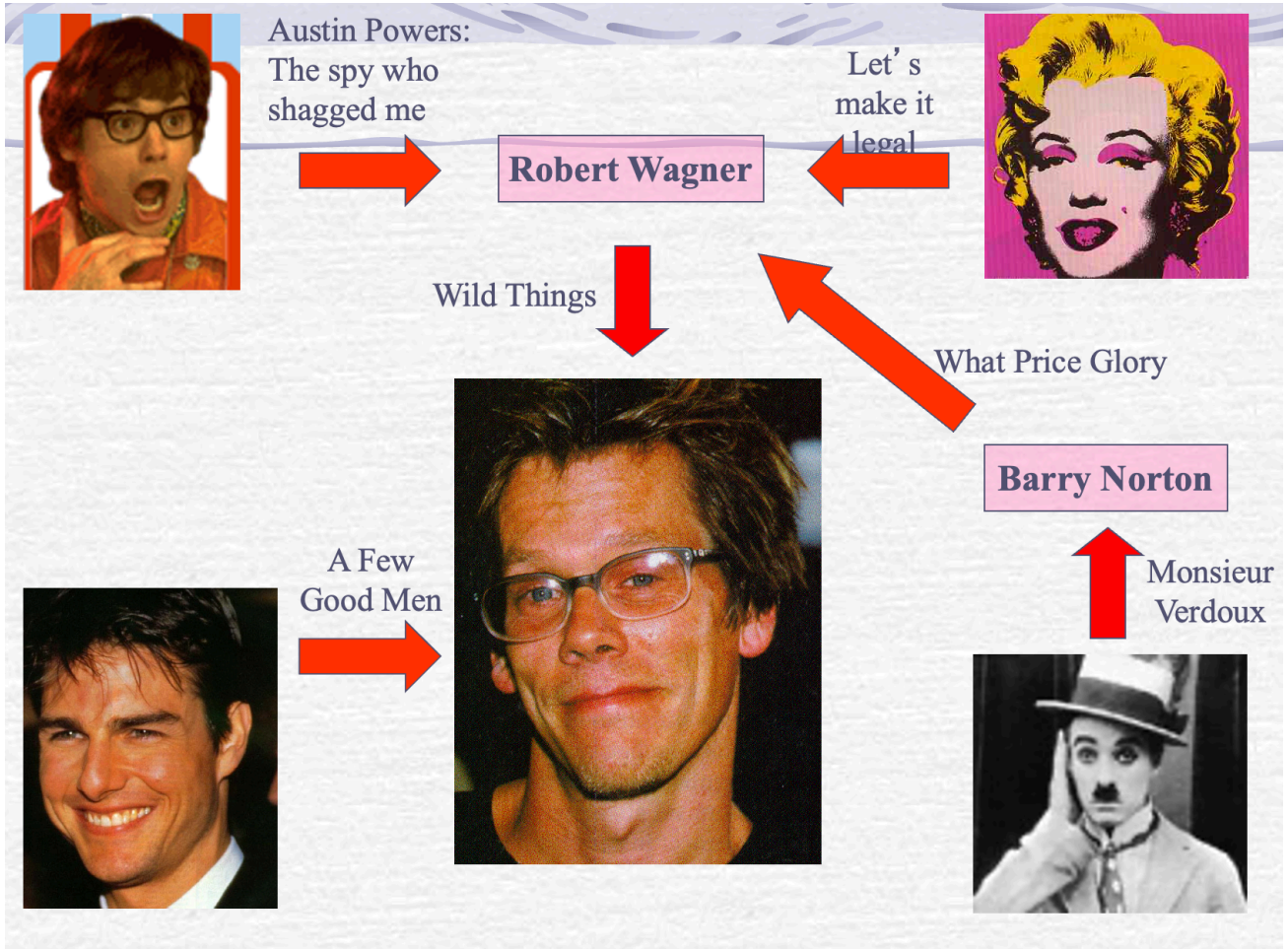
What Watts and Strogatz showed is that **it only takes a small amount of long-range connections** to dramatically shrink the average path length between nodes.

They showed that high clustering and short path lengths can coexist. They called networks with high clustering and short path lengths **small world networks**.

This phenomenon expresses itself repeatedly.

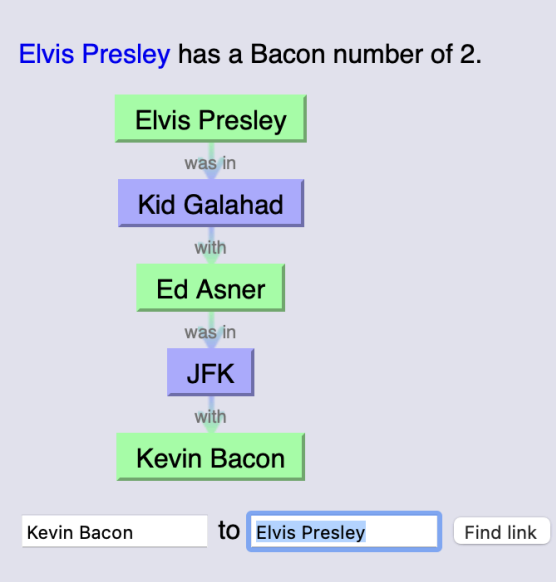
For example, consider the network of movie actors: two actors are connected if they appear in the same movie.

Thus we have the phenomenon of the **six degrees of Kevin Bacon**.



You can try your luck at The Oracle of Bacon.

For example, Elvis Presley:



What's special about Kevin Bacon?

Not really anything.

Because movie co-appearance forms a small world network, most any path between two actors is a short one.

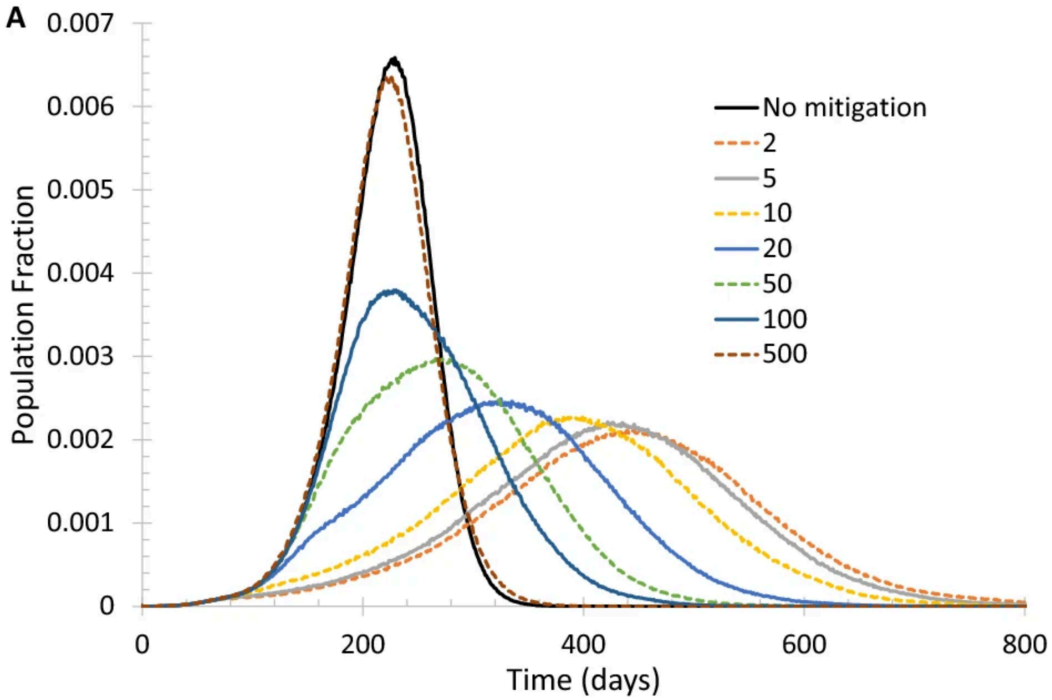
Implications of Small Worlds

Viruses spread rapidly in small worlds.

One of the goals of pandemic lock-downs is to prevent people circulating *outside* of their local social groups. This is an attempt to eliminate the effect of long-range (weak-tie) connections.

This is the idea behind travel bans and mandatory quarantining after travel.

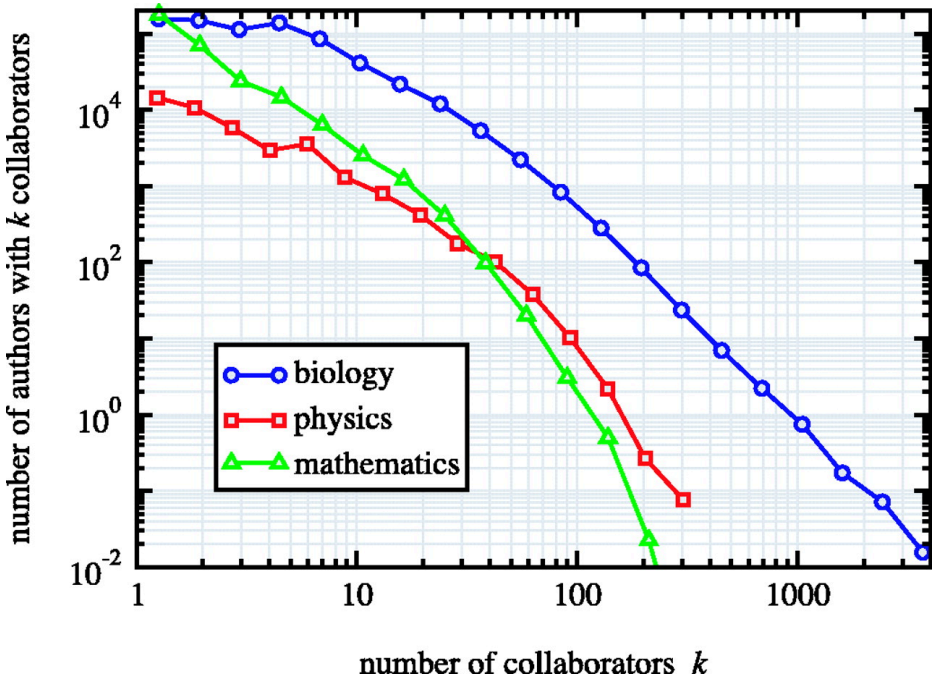
Here is a figure showing the effect on virus propagation of deleting most of the long-range edges from a small-world network. Different curves correspond to when the lock-down is deployed.



From *Mitigating COVID-19 on a Small-World Network*, Marvin Du, Scientific Reports Oct 2021.

Another question concerns how *shortcuts* arise. Recall that node degree distributions typically follow a power-law.

Despite most people have small acquaintance sets, a small subset of people are very highly connected.



These *high social capital* individuals play a big role in creating long-range, path-shortening connections in social networks.

Analyzing Graphs

We have seen different strategies to characterize a graph. To perform further analysis we will employ the following strategies:

- visualize the network in a way that communicates as much insight as possible, and
- compute important metrics of the network.

Visualizing Networks

As an example, we'll consider the following network, which records American football games between NCAA Division IA colleges in the Fall of 2000 (available here).

Each vertex represents a football team, which belongs to a specific conference (Big Ten, Conference USA, Pac-10, etc.).

An edge between two vertices v_1 and v_2 means that the two teams played each other. The weight of the edge (v_1, v_2) is equal to the number of times they played each other.

This data comes from M. Girvan and M. E. J. Newman, *Community structure in social and biological networks,* Proc. Natl. Acad. Sci. USA 99, 7821-7826 (2002).

```
1 football = nx.readwrite.gml.read_gml('data/football.gml')
2
3 print(f'The football network has {len(football.nodes())} nodes and {len(football.edges())} edges')
```

```
The football network has 115 nodes and 613 edges
```

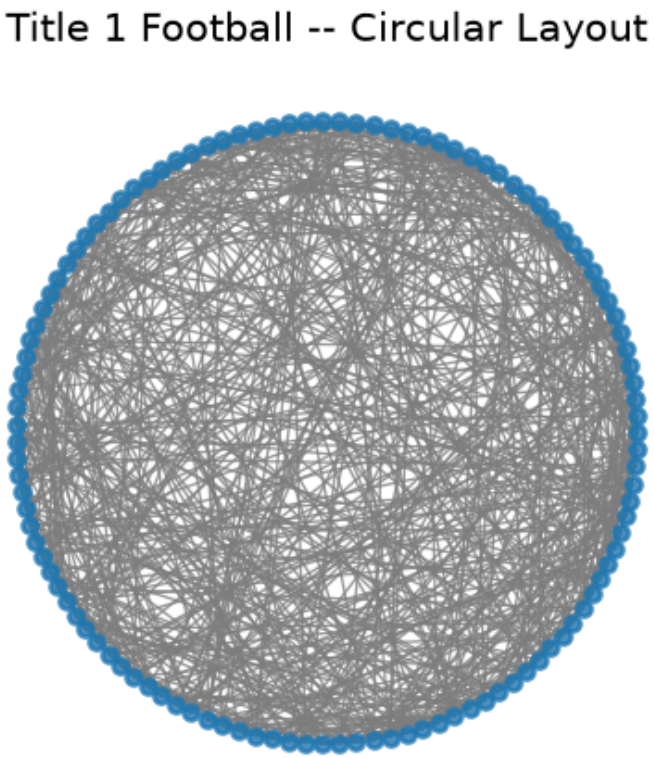
To get a sense of what is unusual here, we can compare this network to a $G(n, p)$ random network with the same number of nodes and edges.

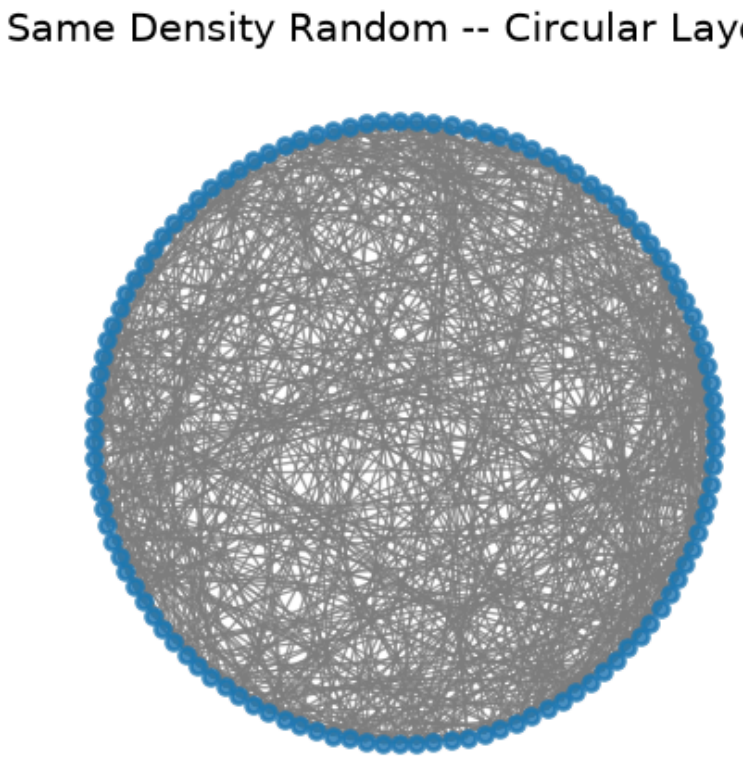
```
n = len(football.nodes())
e = len(football.edges())
p = e / ((n * (n-1))/2)
F_random = nx.erdos_renyi_graph(n, p, seed = 0)
```

One way to visualize is to use a circular layout, which keeps all the edges in the interior.

This can make things easier to see sometimes.

▶ Code





There is some non-random structure here, but it's not clear exactly what it is. To better investigate this network we will use a more informative layout.

The standard `networkx` routine uses what is called a *spring* layout.

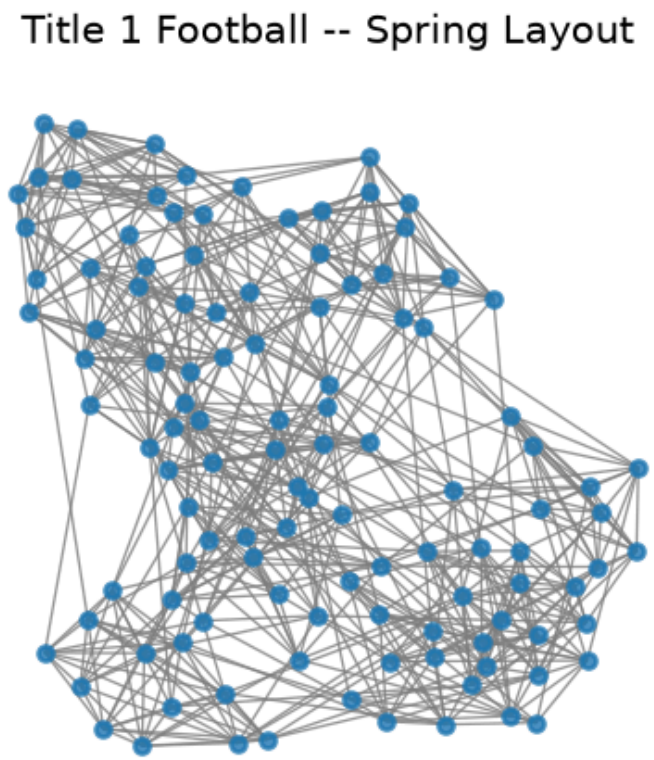
Here is how the *spring* layout works.

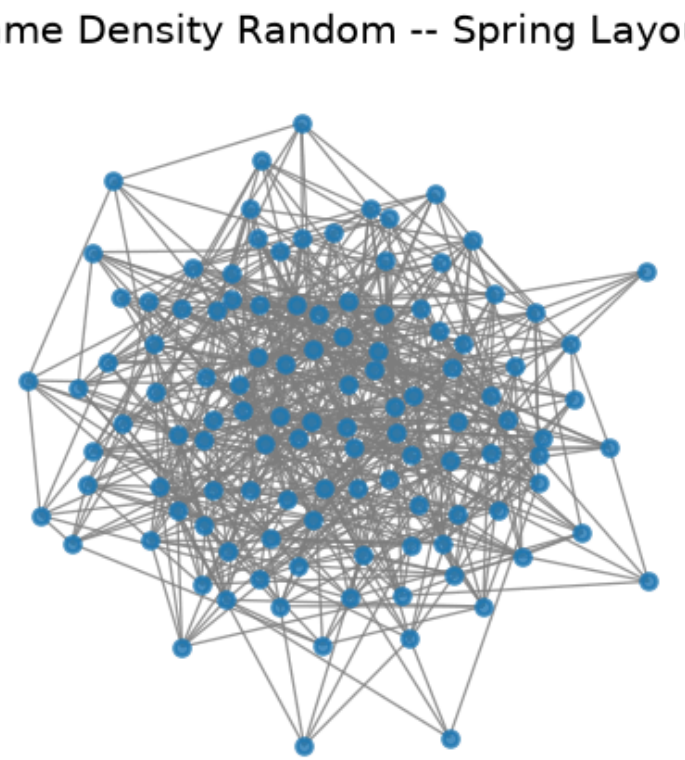
- Each edge has a `weight` parameter (could be 1 for all edges).
- The layout routine fixes
  - a spring of length = 1/`weight` between the nodes,
  - a repulsive force between each pair of nodes,
  - and then lets the set of all forces reach its minimum energy state.

This is a kind of minimal distortion in a least-squares sense.

Spring Layout

▶ Code



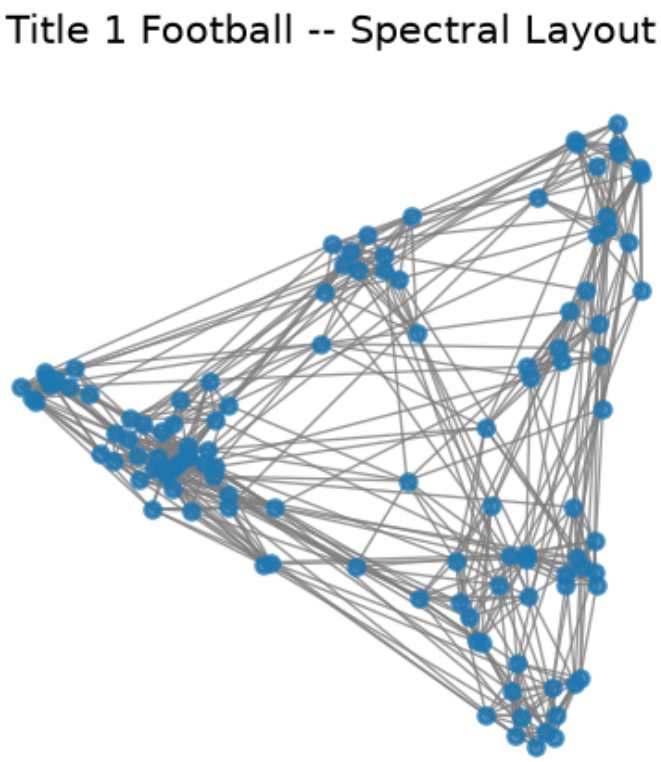


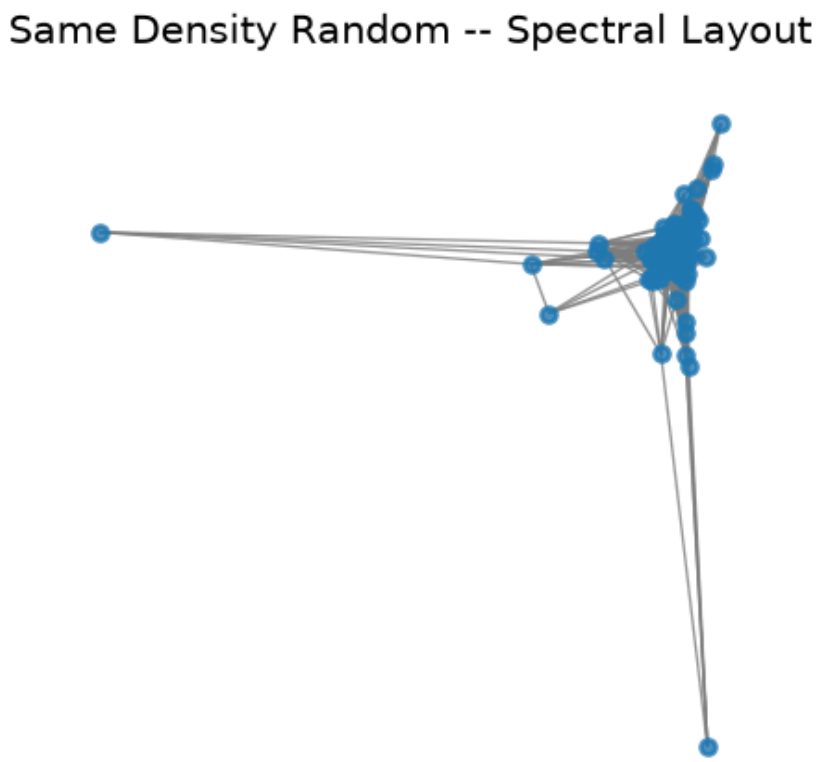
Notice how the spring layout tends to bring clusters of densely connected nodes close to

Spectral Layout

Finally, we can try the spectral layout. We will define the spectral layout in the next lecture.

▶ Code

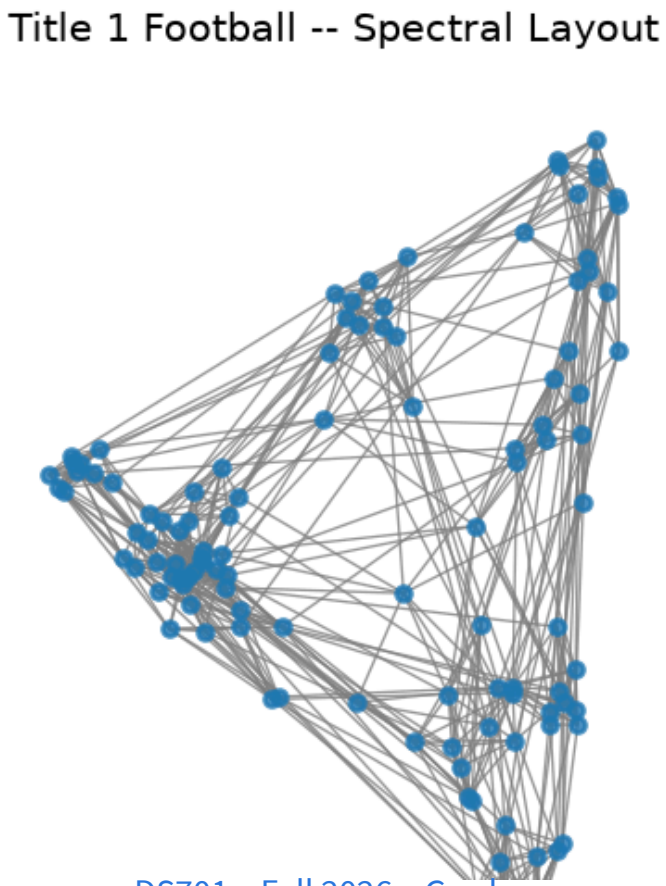




With the spectral layout, we can start to understand the structure of the network.

We see clusters of teams that correspond to conferences, and understand that there are not too many high-degree nodes.

▶ Code



Metrics

Another way to understand network structured data is to look at important metrics.

For example, we can start with the **clustering coefficient**:

▶ Code

```
The clustering coefficient of the Football network is 0.403
The clustering coefficient for the equivalent random network is 0.088
```

Another useful metric is the diameter[1].

▶ Code

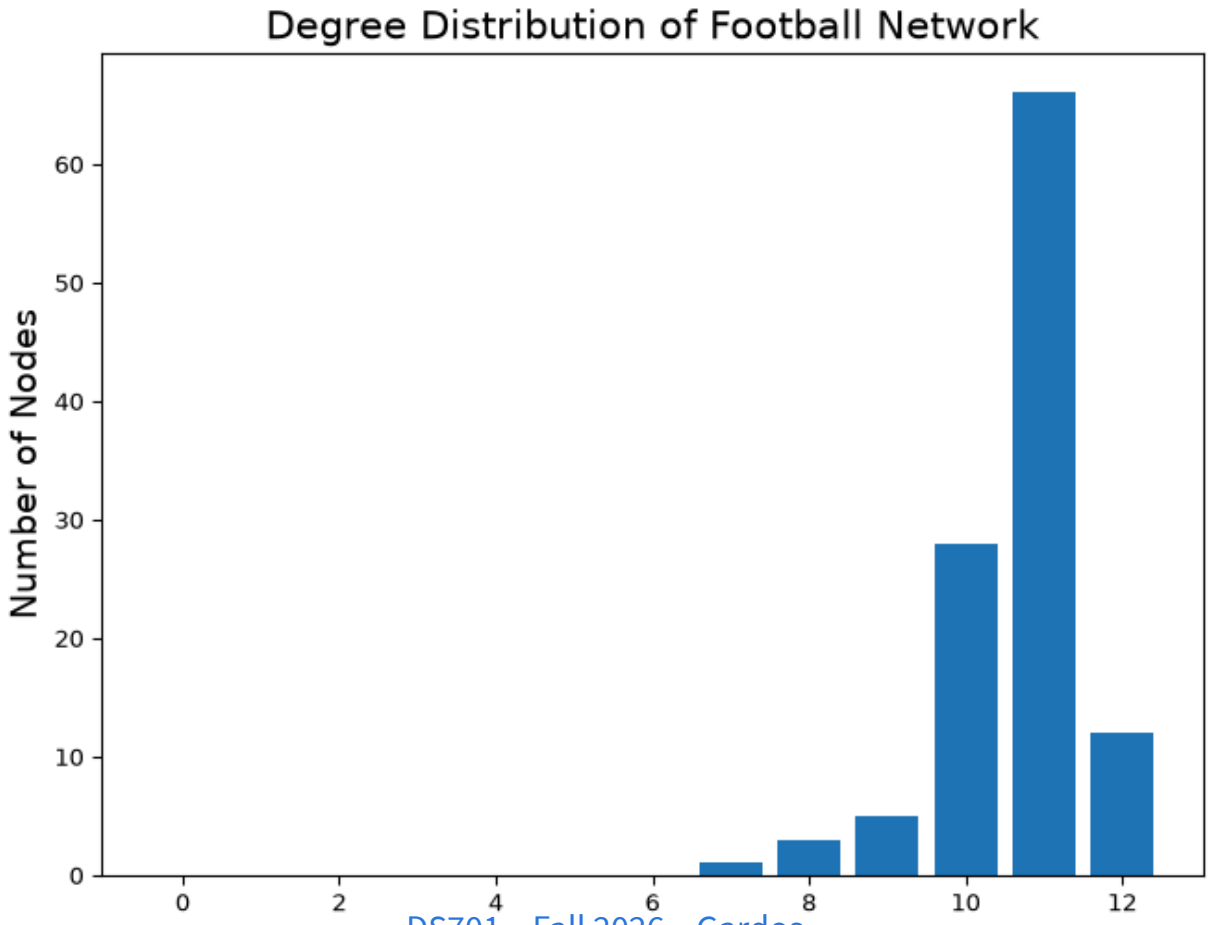
```
The diameter of the Football network is 4 and the average shortest path length is 2.508
The diameter of the equivalent random network is 4 and the average shortest path length is 2.215
```

1. The diameter is the greatest shortest path length between any two nodes.

The next property we can look at is the **degree distribution**.

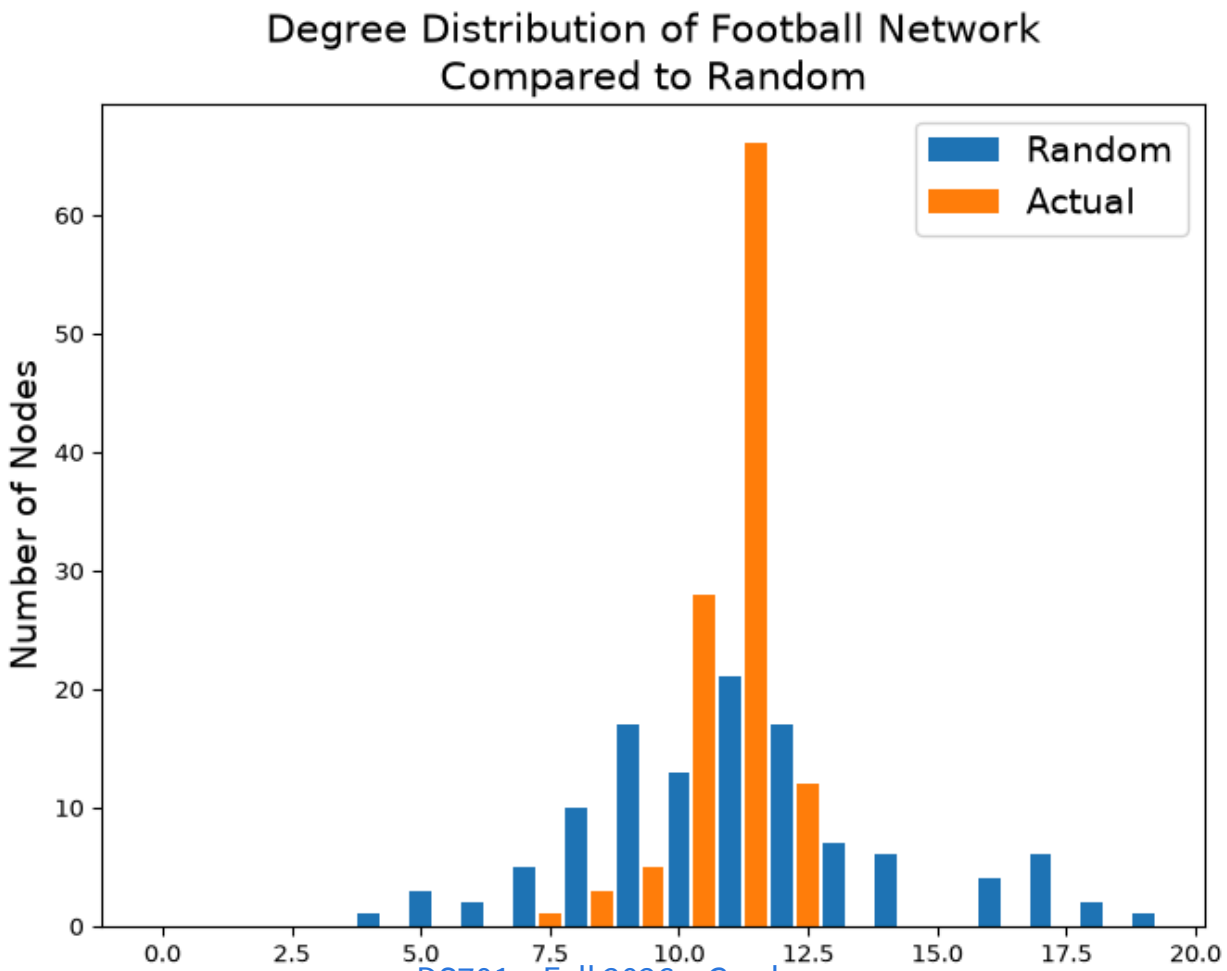
▶ Code

▶ Code



To get a sense of what is unusual here, we can again compare this to the equivalent random network.

▶ Code



Recap

We introduced

- Networks
  - directed, undirected graphs,
  - degree, paths, neighbors, connectivity
- $G(n, p)$ random graph
- Degree distributions
- Cluster coefficients
- Graph metrics
- Visualizing graphs, spring and spectral layouts