1 Representations and statistics of networks

In this lecture, we will learn about the three basic representations of networks, which are useful both for reasoning about network structure and dynamics, and for performing calculations using code. We will then learn about the basic statistical tools for summarizing the structure of a network, at both the level of individual nodes and the entire network.

1.1 Representing networks

There are three main ways to represent a network. Recall our example from Lecture 1.

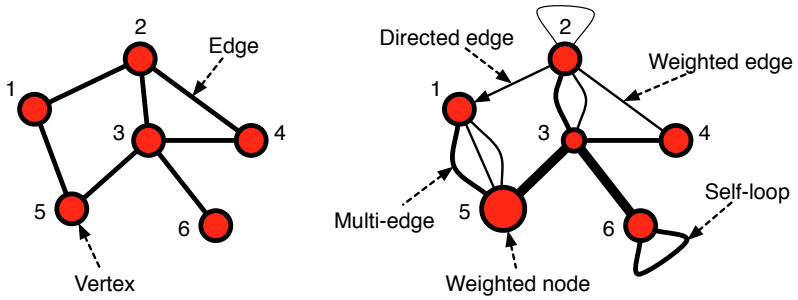


Figure 1: A simple network (left) and a network with more exotic graph properties (right).

1.1.1 The adjacency matrix

An **adjacency matrix** is an $n \times n$ matrix, typically denoted A, defined as

$$A_{ij} = \begin{cases} w_{ij} & \text{if } i \text{ and } j \text{ are connected} \\ 0 & \text{otherwise,} \end{cases}$$

where $n = |V|$ is the number of nodes, and $w_{ij} \in \mathbb{R}$ is the *weight* of the connection between i and j.[1] If A represents an *unweighted network*, then $w_{ij} = 1$ for all i, j.

For the two networks in Figure 1, their adjacency matrices are

$$A_{\text{simple}} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \qquad A_{\text{exotic}} = \begin{pmatrix} 0 & 0 & 0 & 0 & \{1,1,2\} & 0 \\ 1 & \frac{1}{2} & \{2,1\} & 1 & 0 & 0 \\ 0 & \{1,2\} & 0 & 2 & 3 & 3 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ \{1,2,1\} & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 2 \end{pmatrix}.$$

[1] In some systems, a connection (i, j) may exist but have $w_{ij} = 0$. Such situations require special handling.

Notice that for the simple graph, (i) the diagonal is all zeros (no self-loops), the non-zero entries are binary (unweighted), and that if $A_{ij} = 1$ then also $A_{ji} = 1$ (undirected, which implies that A is symmetric across the diagonal). For the non-simple graph, none of these properties holds: it is a directed, weighted, multigraph, with self-loops.

Adjacency matrices are most commonly used in mathematical expressions, e.g., to describe mathematically what a network algorithm or network calculation does.[2]

Sometimes, adjacency matrices are used in network algorithms, because an operation on the matrix A is more efficient (takes less time) than the same operation on another representation, e.g., checking whether an edge exists $(i, j) \in E$ is fastest in A because it's a constant-time $O(1)$ lookup. The tradeoff of fast access to each adjacency, however, is that A takes a quadratic amount $O(n^2)$ of memory to store, because it explicitly stores a value for every $\binom{n}{2}$ pair of nodes. Hence, on a modern computer, a simple network with only $n = 100{,}000$ nodes will take about 8 bytes$\times\binom{10^5}{2}/2^{30} = 37$ GB of space to store.

When a network is **dense**, meaning that most of the elements of A are non-zero, there is no way to be more efficient in storing the network than an adjacency matrix. This situation can occur edges are defined by pairwise correlation or similarity scores, because every pair will produce some score. But, most empirical networks are **sparse**, meaning that the number of non-zero entries in the adjacency matrix is roughly $c \cdot n$, for a small c, and an adjacency matrix stores this small number of non-zero elements inefficiently.

1.1.2 The adjacency list

An **adjacency list** stores only the non-zero elements of the adjacency matrix, using an array of (unordered) lists, one list for each of the n vertices, and the ith list stores the names of the neighbors of node i. Optionally, edge annotations w_{ij} can be stored together with the neighbor name j in a tuple, e.g., (j, w_{ij}). Hence, an adjacency list is similar to a hash table of adjacencies. For the two networks in Figure 1, their adjacency list representations are

$A_{\text{simple}} =$
[1] → (2, 5)
[2] → (3, 1, 4)
[3] → (2, 5, 4, 6)
[4] → (2, 3)
[5] → (1, 3)
[6] → (3)

$A_{\text{exotic}} =$
[1] → ((5,1),(5,1),(5,2))
[2] → ((1,1),(2,$\frac{1}{2}$),(3,2),(3,1),(4,1))
[3] → ((2,1),(2,2),(4,2),(5,3),(5,3))
[4] → ((2,1),(3,2))
[5] → ((1,1),(1,2),(1,1),(3,3))
[6] → ((3,3),(6,2))

[2]For this reason, many techniques from linear algebra have applications in network analysis and modeling. Here, we will not assume the reader has familiarity with matrix multiplication or spectral techniques, but will try to highlight such connections for those who do.

On the left, because the network is undirected, each of the $m = 7$ edges appears twice in the adjacency list, once as j in i's list, and once as i in j's list. The adjacencies for a given vertex are typically stored as a linked list, or in a more efficient data structure, like a self-balancing binary tree (e.g., a red-black or a splay tree). This form of representation is what is meant by a "sparse matrix" data structure, and is used in most mathematical programming languages.

The GML file format[3] is essentially a kind of adjacency list representation, written out in text. By only storing the non-zero entries of the adjacency matrix, an adjacency list takes a linear amount $O(n+m)$ space: $O(n)$ space for the array of n nodes and $O(m)$ space to store the m edges. In this representation, checking whether an edge exists $(i,j) \in E$ is slower than in an adjacency matrix, taking $O(m/n) = O(\langle k \rangle)$ time, where $\langle k \rangle$ is the average number of edges in an adjacency's list.

1.1.3 The edge list

The third representation is an *edge list*, which stores only the non-zero elements of the adjacency matrix. For the two networks in Figure 1, these are their edge lists:

$$\begin{aligned} A_{\text{simple}} &= \{(1,2),(1,5),(2,3),(2,4),(3,5),(3,6)\} \\ A_{\text{exotic}} &= \{(1,5,1),(1,5,1),(1,5,2),(2,1,1),(2,3,2),(2,2,\frac{1}{2}),(2,3,1), \\ &\qquad (2,4,1),(3,2,1),(3,2,2),(3,4,2),(3,5,3),(3,5,3),(4,2,1), \\ &\qquad (4,3,2),(5,1,1),(5,1,2),(5,1,1),(5,3,3),(6,3,3),(6,6,2)\} \ . \end{aligned}$$

In the simple graph case, each edge appears only as (i,j), which implies that it is unweighted, and the presence of (i,j) implies the existence of (j,i). Note that this implication is not present in the edge list itself, and has to be inferred or indicated externally. In the non-simple graph case, edges can be directed, and hence seeing (i,j) does not imply that (j,i) must also exist. And, here, edges are represented as a tuple (i,j,w_{ij}), where w_{ij} gives the weight of the edge.

Edge lists are typically only used as a convenient way to store a network in a file, and even then, formats like GML are generally better, because they avoid the ambiguities described above.

2 Summarizing networks using statistics

Regardless of which representation we use, networks are sufficiently complicated objects that we almost always need to use statistical methods in order to gain an intuitive sense of their shape and variations, or to describe their patterns, or test whether these patterns are meaningful. In this section, we will survey some of the most basic of these descriptive statistics. These statistics fall largely into three categories:

[3] See `https://en.wikipedia.org/wiki/Graph_Modelling_Language`.

1. **connectivity** measures: these are directly or indirectly related to node degrees, and to the overall degree structure of the network
2. **motif** measures: these count the frequency of specific subgraphs in a network
3. **positional** measures: these are related to where in the network a node sits and to the overall structure of "geodesic" paths in the network.

Many of these network measures come in two flavors:

- **node-level** or "local" measures, which we calculate for an individual node i, and
- **network-level** or "global" measures, which we calculate across the entire network.

Sometimes, we can obtain a network-level statistic of a particular quantity by simply computing the average node-level quantity, but not always.

Because network representations can vary, so too do the algorithms for calculating specific network statistics. As a result, before beginning any network analysis, we need to first identify the graph properties of representation being used: Is the network simple? Are edges directed? Are edges weighted? Do nodes have metadata? Does the graph have one or several components? Etc.

2.1 Node degree

The most basic of all network statistics is the **degree** of a node, which counts the number of connections it has in the network. The degree structure of a network is the *first-order* description of a network, and is often sufficient to drive many other statistical patterns in the network's structure. Later, we'll learn how to assess the extent to which degrees alone can explain other structural patterns in networks.

2.1.1 Undirected networks

When edges are undirected, an edge (i, j) contributes to the degree of both i and j. By convention, k_i denotes the degree of vertex i. For a simple adjacency matrix A, the degree of node i is

$$k_i = \sum_{j=1}^{n} A_{ij} = \sum_{j=1}^{n} A_{ji} \ . \tag{1}$$

That is, the degree k_i is just the sum of the ith column, or ith row, of the adjacency matrix A. The row- and column-sum equivalence holds only because edges are undirected. If A represents a weighted network, this sum is called the node **strength**; the term "degree" is reserved for unweighted counts.[4]

[4] If A is a non-binary adjacency matrix, then a simple row sum will yield the strength s_i of node i; if the summation is modified as a count of non-zero entries, we will get back the correct value of k_i.

Every edge in an undirected network contributes twice to some degree (once for each endpoint or "stub"), and so the sum of all degrees in a network must be equal to twice the total number of edges in a network m:

$$m = \frac{1}{2}\sum_{i=1}^{n} k_i = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=i}^{n} A_{ij} \quad . \tag{2}$$

Using the first part of Eq. (2) allows us to derive a simple an expression for the network's mean degree $\langle k \rangle$:

$$\langle k \rangle = \frac{1}{n}\sum_{i=1}^{n} k_i = \frac{2m}{n} \quad . \tag{3}$$

And finally, if we divide the number of edges by m by maximum number of possible edges, or divide the mean degree by its largest possible value, we obtain a quantity that is sometimes called the network's **connectance** [5] or **density**:

$$\rho = \frac{m}{\binom{n}{2}} = \frac{2m}{n(n-1)} = \frac{\langle k \rangle}{n-1} \quad . \tag{4}$$

2.1.2 Directed networks

In a directed network, a node i has three types of degrees: its **in-degree** k_i^{in}, meaning the number of connections to i; its **out-degree** k_i^{out}, the number of connections from i; and its **total degree** k_i^{tot}, the number of neighbors, ignoring edge directionality. The latter is equivalent to treating directed edges as undirected, and the "collapsing" multi-edges so that only one connection (i, j) exists.

in-degree, k^{in} out-degree, k^{out}

On a binary adjacency matrix A representing a directed network, the in-, out-, and total degrees

[5] Sometimes, connectance is defined as c/n, which is asymptotically equivalent to $c/(n-1)$.

are defined as

$$k_i^{\text{out}} = \sum_{j=1}^{n} A_{ij} \tag{5}$$

$$k_i^{\text{in}} = \sum_{j=1}^{n} A_{ji} \tag{6}$$

$$k_i^{\text{tot}} = \sum_{j=1}^{n} A_{ji} \vee A_{ij} \ , \tag{7}$$

where $\vee$ is an *or* function. In a directed network, the number of edges m is the sum of the entire adjacency matrix, since edge directed edge appears exactly once in A. The fact that each edge contributes exactly once to some in-degree and once to some out-degree also implies that the average in-degree equals the average out-degree:

$$\langle k^{\text{out}} \rangle = \frac{1}{n} \sum_{i=1}^{n} k_i^{\text{out}} = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{n} A_{ij} \right) = \frac{m}{n} = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} A_{ji} \right) = \frac{1}{n} \sum_{i=1}^{n} k_i^{\text{in}} = \langle k^{\text{out}} \rangle \ . \tag{8}$$

2.2 Degree distributions

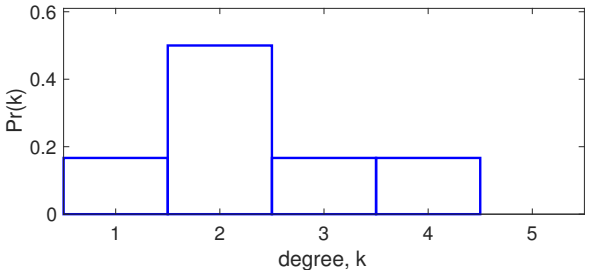
If we calculate the degrees of all the vertices in a network, and then write them out, we get something called a **degree sequence**, which enumerates the network's degree structure. For the simple graph in Figure 1, its degree sequence is

$$(1, 2, 2, 2, 3, 4) \ .$$

But a simple sequence doesn't provide much information about the *variability* of node degree across the network. To capture this behavior, we use something called the **degree distribution**, denoted $\Pr(k)$, which is simply the probability that a vertex selected uniformly at random will have k neighbors. A degree distribution is a normalized histogram of the degree values. For the simple graph in Figure 1, its degree distribution is

k	1	2	3	4
$\Pr(k)$	1/6	3/6	1/6	1/6

where $\Pr(k) = 0$ for all other values of k. The standard way to present a degree distribution $\Pr(k)$ is via a bar plot, like so:



2.2.1 Exploring the degree distribution

Most real-world networks, biological or otherwise, exhibit a "heavy-tailed" degree distribution, meaning that the variance[6] of the degree distribution σ^2 is large relative to its mean $\langle k \rangle$. Because degrees must be non-negative (they are "count" variables), this behavior implies that the degree distribution $\Pr(k)$ is "right-skewed," meaning that the largest degree vertices are often several orders of magnitude better connected than the typical node.

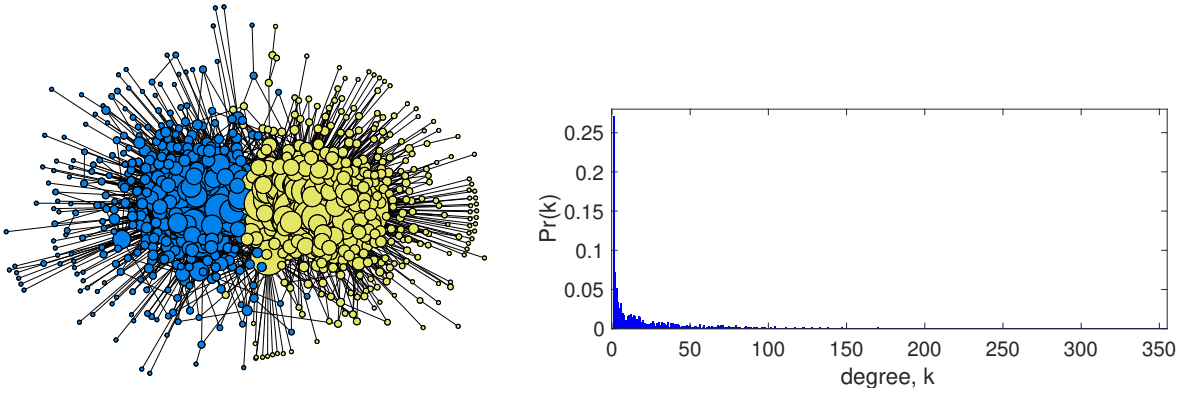
This kind of high variance, however, can be difficult to see in a simple distribution plot, and instead, it is conventional to (i) use a log-log plot, which visually compresses the variance in both k and $\Pr(k)$, and (ii) plot the *complementary cumulative distribution function* or CCDF, defined as $\Pr(K \geq k)$. In this expression, K is now a random variable, and the CCDF plots the fraction of nodes with degree *at least* some value k. If the minimum degree of a network is $k = 1$, then $\Pr(K \geq 1) = 1$, because all nodes have degree at least the minimum.[7]

To illustrate how the log-plots change the kind of information we can see in a degree distribution, and how plotting the CCDF improves the visual interpretability, we'll use a concrete example. Below is a network of $n = 1490$ weblogs from around the 2004 U.S. Presidential election, along with a standard plot of the degree distribution, which spans $m = 19090$ edges and has a mean degree $\langle k \rangle = 25.6$.[8] The details of how these data were collected, or what they mean is not that important, except to note that this is a web graph, and is thus a directed network. Notice that the degree distribution emphasizes the low-degree portion of the distribution, for $k \leq 10$ or so. In fact, the largest degree is $k_{\max} = 351$, a value more than 13 times larger than the mean, but you can't tell that from the plot.

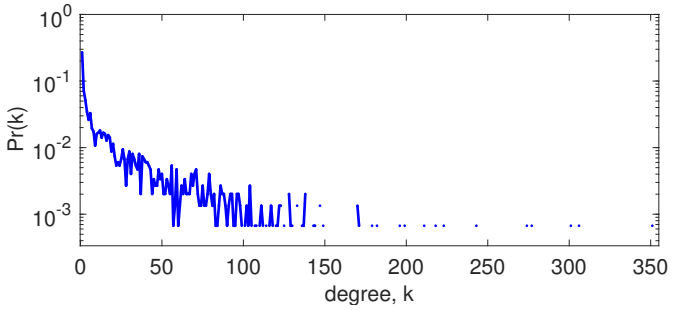
[6] Recall that the sample variance of the degrees would be defined as $\sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} (k_i - \langle k \rangle)^2$.

[7] Applying a `cumsum` function to the PDF of a degree distribution *almost* yields the CDF, which is defined as $\Pr(K < k)$. Do you see why? Think about what happens to the first element under `cumsum` and whether that is correct for $\Pr(K < \min(k_i))$. After correcting for this behavior, one can obtain the CCDF by noting that $\Pr(K \geq k) = 1 - \Pr(K < k)$.

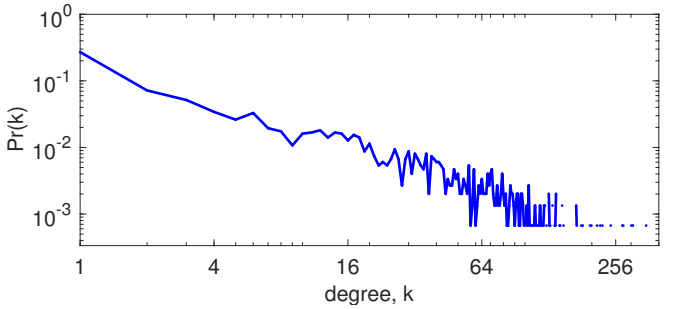
[8] The visualization comes from Karrer & Newman, *Phys. Rev. E* **83**, 016107 (2011), and the network comes from Adamic & Glance, *Proc. WWW-2005 Workshop on the Weblogging Ecosystem* (2005).



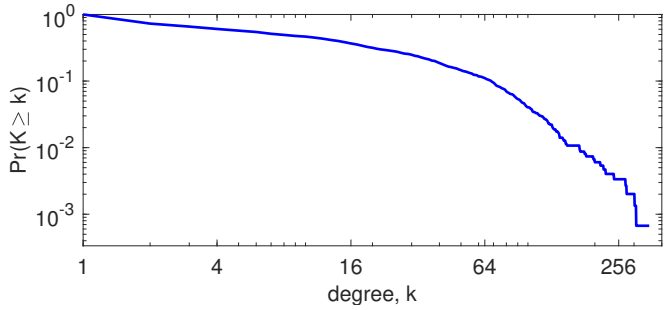
What follows is a sequence of plots, showing the same degree data, as (i) $\log_{10} \Pr(k)$ vs. k (a semilog-y plot), then (ii) $\log_{10} \Pr(k)$ vs. $\log_{10} k$ (a loglog plot), and finally (iii) $\log_{10} \Pr(K \geq k)$ vs. $\log_{10} k$ (a loglog plot of the CCDF).



In the semilog-y plot above, we can now see more of the low frequency / high degree events, which we call the "upper tail" of the distribution. The dots represent nodes with unique degree values; hence, $\Pr(k) = 1/n$, which is the lowest value possible in $\Pr(k)$. This minimum value is what causes the "flat-bottom" structure of the degree distribution above $k = 70$ or so.

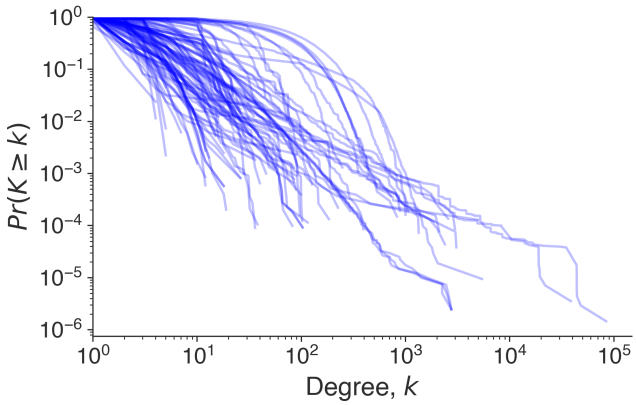


The loglog plot above stretches out the low-degree structure while compressing the range of the high-degree portion of the distribution, allowing us to see both parts together. But, the $1/n$ artifact is still apparent, and the jagged structure at $k > 60$ or so is too noisy to see any clear pattern. The solution is the CCDF, which is a monotonically decreasing function that smooths out many of the jaggies in the upper tail.



In the above plot, we can now see clearly that the distribution tends to bend down around $k = 64$, falling off more quickly than it did below that point. Moreover, we can read off, directly from the plot, the fact that 90% of nodes have degree $k \leq 67$ (at what values of x does a horizontal line at $y = 10^{-1}$ intersect the CCDF?), and only 1% of nodes have degree $k > 169$ (at what value of x does a horizontal line at $y = 10^{-2}$ intersect the CCDF?). Reflecting this extremely uneven distribution of edges, the bottom 90% of nodes, by degree, touch only 53% of all edges, while those top 1% touch 10% of all edges. This dramatic inequality is ubiquitous in real network degree distributions, and highlights the important role that these "high degree" nodes play in structuring the network. The clarity provided by the CCDF on a loglog plot makes it the most useful way to visualize degree distributions.

How much variability do degree distributions exhibit in reality? The figure below shows the results of applying the above CCDF analysis to 100 networks drawn from biological, social, information, and technological networks. The answer is: they vary enormously. In other words, real-world networks exhibit enormous structural diversity, even in something as simple as their degree structure.



2.3 Small motifs

A **motif** is a small, connected subgraph $G_\circ = (V_\circ \subset V, E_\circ \subset E)$ with a particular pattern of edges among a small set of nodes, and usually, we want to count how often a motif $G_\circ$ occurs within a larger network G. The computational cost of counting motifs scales up quickly with the size of $V_\circ$, and thus most motif-based measures focus on the frequencies of small motifs, e.g., 2, 3, or 4 nodes.[9]

2.3.1 Reciprocity (directed networks)

The smallest non-trivial motif in a directed network is a reciprocated edge, i.e., a *2-cycle* or a cycle of length exactly two, and we use a measure called **reciprocity** r to quantify their prevalence. Reciprocity comes in both node-level (local) and network-level (global) versions.

The prevalence of this motif is often interpreted functionally. In some ecological networks, reciprocity can indicate the degree of mutualism present in an ecosystem, while in gene regulatory networks, reciprocated links indicate the presence of self-regulating interactions.

[9]Computational complexity and *subgraph isomorphisms* are key problems for motif-counting algorithms. Suppose we are searching for a motif on $n_\circ$ nodes. If we blindly check all $f(n_\circ) = \binom{n}{n_\circ}$ groups of nodes, each found motif will occur $|n_\circ|!$ times in this search, due to isomorphisms. That is, there are $|n_\circ|!$ ways to permute the node labels of the motif, and it's still the same motif. Asymptotically, how expensive is such a search for $n_\circ = \{2, 3, 4, 5, 6\}$?

The reciprocity of a network r is given by the fraction of links that are reciprocated, meaning it has a maximum value of $r = 1$. To compute r, we simply count the number of 2-cycles and divide by the number of edges. When edges have unit weight, a network's reciprocity is defined as

$$r = \frac{1}{m} \sum_{ij} A_{ij} A_{ji} \ . \tag{9}$$

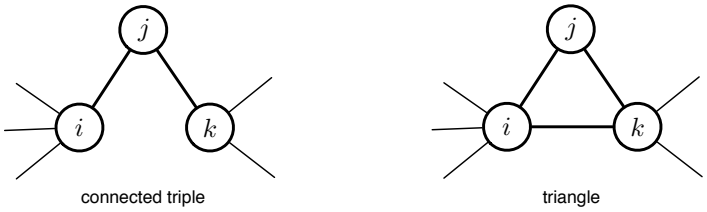
Note that in a directed network, m counts the number of directed edges, and hence we normalize by $m = \sum_{ij} A_{ij}$ to place r on the unit interval $[0, 1]$.

We can also quantify the reciprocity that an individual vertex experiences, which is a node-level measure called *local reciprocity*. Instead of summing over the entire network, in this case, we count the number of 2-cycles attached to vertex i, and divide by its out-degree:

$$r_i = \frac{1}{k_i^{\text{out}}} \sum_{j} A_{ij} A_{ji} \ . \tag{10}$$

2.3.2 Clustering coefficient (undirected networks)

In an undirected network, the smallest non-trivial motif is a triangle, and the **clustering coefficient** C is a network-level measure that captures the relative frequency or the density of triangles in a network. Concretely, C quantifies the fraction of "connected triples," that is, a pair of edges $(i, j), (j, k)$ that are "closed" by an edge (k, i) to form a triangle.



Like reciprocity, C is defined mathematically as a count of the desired motif (a triangle), normalized by a count of the number of possible motifs:

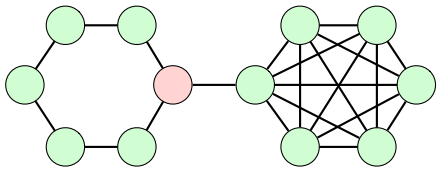
$$C = \frac{(\text{number of triangles}) \times 3}{(\text{number of connected triples})} \tag{11}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} A_{ij} A_{jk} A_{ki} \Bigg/ \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k \neq i}^{n} A_{ij} A_{jk} \quad , \tag{12}$$

where the factor of 3 in Eq. (11) comes from the symmetry of the triangle (every triangle contains three connected triples), and the $k \neq i$ restriction in the numerator of in Eq. (12) prevents counting

$(i, j), (j, i)$ as a triple.[10] Because the numerator can never exceed the denominator, and they are both count variables, $C \in [0, 1]$, and we say that it measures the density of triangles in the network.[11] Calculating Eq. (11) directly would take $O(n^3)$ time. How long would it take to calculate C using an adjacency list representation?

A worked example. Consider a simple network composed of cycle and a clique:



The clique has 6 vertices, making $\binom{6}{3} = 20$ triangles, and thus also 60 connected triples. The cycle contains 6 connected triples and no triangles. Finally, there are 2 connected triples starting from the cycle and 5 connected triples starting from the clique that use the edge joining the cycle to the clique. Adding this all up yields a clustering coefficient of $C = 60/73 = 0.82$.

Local clustering coefficient. The clustering coefficient can also be defined as a node-level measure, which captures the fraction of neighbors j, k of node i that are themselves connected by an edge (j, k):

$$C_i = \frac{\text{(number of pairs of neighbors of } i \text{ that are connected)}}{\text{(number of pairs of neighbors of } i\text{)}} \tag{13}$$

$$= \sum_{jk} A_{ij} A_{jk} A_{ki} \Big/ \binom{k_i}{2} \quad . \tag{14}$$

In the cycle-plus-clique example above, the highlighted vertex has a local clustering coefficient of $C_i = 0$ because it participates in no triangles. Its immediate neighbor in the clique is a more interesting case, with $C_i = 10/15 = 0.67$.

2.3.3 Feed-forward and feedback loops (directed networks)

In directed networks, and particularly in biological networks, two slightly larger motifs of special interest are the feed-forward loop and the feedback loop. These motifs are interesting because they

[10] In linear algebra terms, Eq. (12) is equivalent to $C = \text{trace}(A^3) \Big/ \sum_{i \neq j} A^2$. This is because the diagonal of the matrix A^q counts the number of cycles of length q that "originate" at each node.

[11] When the network is fully connected, in which every edge exists, $C = 1$ (do you see why?). Similarly, in any network without triangles, e.g., a tree, a bipartite network, etc., $C = 0$ (do you see why?).

represent basic biological circuitry for governing the flow of activation across molecular networks.[12]

A feed-forward loop is defined as one that has a two-step path $(i,j),(j,k)$ that is shortcut by an edge (i,k) that "feeds forward" the signal passing along the longer path. In this case, we would say that node j is the "feed forward" node. Similarly, a feedback loop is defined the same, except that the third edge is (k,i), which "feeds back" the signal that passed along the longer path.

If a network is directed, then we can count the number of FFL and FBL straightforwardly under an adjacency matrix representation:

$$\text{FFL} = \sum_{i=1} \sum_{j} \sum_{k \neq i} A_{ij} A_{jk} A_{ik} \tag{15}$$

$$\text{FBL} = \sum_{i<j<k} A_{ij} A_{jk} A_{ki} \ , \tag{16}$$

These adjacency matrix calculations take $O(n^3)$ time because each has three nested loops over n nodes, and hence don't scale well to large networks.

A more efficient approach, taking $O(n + m\langle k \rangle^2)$ time, would use a simple "path enumeration" algorithm on an adjacency list to find all paths $(i,j),(j,k)$ (where $i \neq k$) and, for each, check whether the edge (i,k) exists, indicating a FFL motif, or (k,i) exists, indicating a FBL motif.

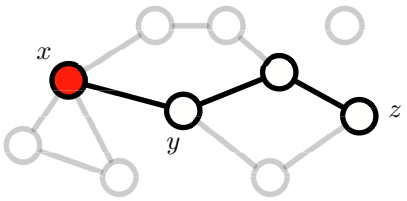
2.3.4 Larger motifs

A motif can be defined on any size subgraph $n_\circ < n$, and sometimes "large" motifs, when $n_\circ > 3$, are interesting. However, as $n_\circ$ increases, it becomes increasingly expensive to both enumerate and count such motifs. For instance, while there are 6 distinct motifs on $n_\circ = 4$ nodes, $n_\circ = 5$ nodes have 21, $n_\circ = 6$ nodes have 112, and $n_\circ = 7$ nodes have 853. Even if we successfully count the frequencies of a large motif, its interpretation also requires special care, because there can be many different ways to "normalize" that count relative to an expectation, as we did in Eq. (9) for reciprocated links and Eq. (11) for triangles. We will revisit the idea of assessing the frequency of larger motifs when we discuss random graph models.

[12] For instance, see `https://en.wikipedia.org/wiki/Feed_forward_(control)`.

2.4 Geodesic paths and network position

Recall that a *path* in a network is a sequence of distinct vertices $x \to y \to \cdots \to z$ such that for each consecutive pair of vertices $i \to j$, there exists an edge $(i, j) \in E$ in the network, and the *length* of the path is the number of edges it crosses. A *geodesic* or *shortest path* is the shortest of all possible paths between two vertices. These paths serve as the basis for a number of positional measures of network structure.



To calculate nearly any positional measure, we must first calculate the geodesic distances ℓ_{ij} between all pairs of nodes i, j. Usually, we accomplish this by applying an off-the-shelf algorithm for the *All Pairs Shortest Paths* (APSP) problem.[13] The output will be a *pairwise distance* matrix ℓ, where the entry ℓ_{ij} gives the length of the geodesic path between i and j. Crucially, if no such path exists (because i and j are in different *components* of the network, e.g., the singleton node vs. any other node in the example above), then $\ell_{ij} = \infty$, and we say that j is not *reachable* from i.

If G is a weighted network, the pairwise distance matrix ℓ will contain the *weight* of the *lightest* path from i to j. In this case, the lightest-weight path may, in fact, have more edges in it than the shortest length path (ignoring edge weights). In an unweighted network, edges have unit weight, and hence ℓ_{ij} is the number of edges in the path.

The path structure of a network also determines whether it is composed of one or multiple *components*, each of which is a set of nodes T such that for any pair of members $i, j \in T$, there exists a path $i \to \cdots \to j$ or vice versa. If all nodes are pairwise reachable from each other, then a network has a single component. Else, the network has multiple components, and there is always a *largest connected component* (LCC), i.e., the component with the largest number of nodes in it.[14] In an undirected network, the matrix ℓ is always symmetric, because the shortest path from i to j can be no longer than the shortest path from j to i.

[13] Most commonly, the Floyd-Warshall algorithm. Alternatively, applying Dijkstra's algorithm, for solving the single-source shortest path problem, once for each node will build the pairwise distance matrix ℓ one row at a time. If the network is weighted, there are some technical restrictions on the weight values.

[14] The number and size of components can be computed straightforwardly, by analyzing the output of either an APSP algorithm or a simple breadth-first or depth-first search algorithm. Do you see how?

2.4.1 Diameter

The first positional measure is the network's **diameter**, which is the length of the longest of geodesic path and is meant to evoke the notion of a volume in a metric space. The diameter of a network is a global measure, and is defined as

$$\text{diameter} = \max_{ij} \ell_{ij} \ . \tag{17}$$

If the network is not connected, i.e., if there is more than one component, then the diameter will trivially be infinite. In these cases, it is customary to drop all $\ell_{ij} = \infty$ elements from the calculation, and say the diameter is the longest geodesic among all paths that exist.

2.4.2 Eccentricity

While the diameter is a global measure, the **eccentricity** is an equivalent local measure—it measures the length of the longest path originating at some node i, and is defined as

$$\epsilon_i = \max_{j} \ell_{ij} \ , \tag{18}$$

where again, it is customary to drop all $\ell_{ij} = \infty$ elements from the calculation.

2.4.3 Mean geodesic path length

The **mean geodesic path length** is network-level (global) measure of the average distance between a pair of uniformly random nodes, and is defined as

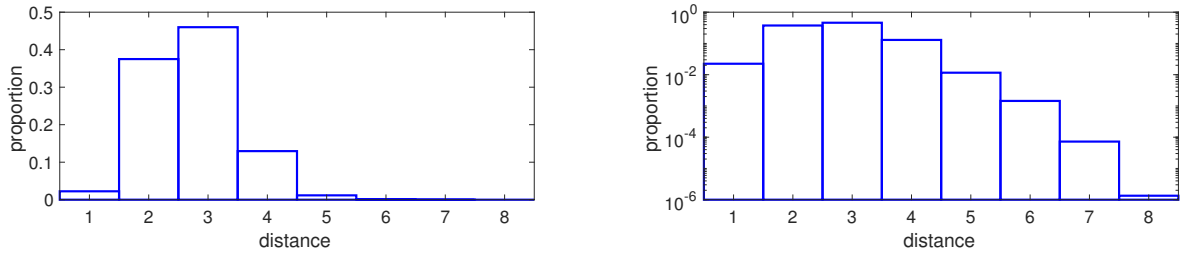
$$\langle \ell \rangle = \frac{1}{Z} \sum_{ij} \ell_{ij} \ , \tag{19}$$

where here it is customary to drop both all $\ell_{ij} = \infty$ elements and $\ell_{ij} = 0$ elements (the diagonal of ℓ) from the calculation, and Z counts the number of the remaining distances.

An example. Consider again the directed network of political blogs from Section 2.2.1, which has $n = 1490$ nodes. Applying an APSP algorithm to the network yields a 1490×1490 pairwise distance matrix ℓ. This network is not connected: it has 268 components, the largest of which contains 1222 nodes (82%), and the smallest of which contains 1 node.

Taking the set of non-zero and non-infinite geodesic distances, the following two plots show the normalized histograms of those distance, once as a simple bar plot and the other as a semilog-y bar plot. The mean geodesic distance is $\langle \ell \rangle = 2.74$, which we can see reflected in the simple bar plot, where the vast majority of distances are 2 or 3. The diameter, however, is not particularly visible in this version. The longest geodesic distance is $\ell_{\max} = 8$, and occurs exactly once, i.e.,

there is exactly one pair of vertices that are separated by a path of length 8, which is one path out of 1,492,064 pairs of connected nodes. This value is a bit easier to see in the semilog-y plot, which also shows more of a smooth fall-off in the distances above the mean, to the maximum at 8.



2.5 Other measures of network structure

These measures of network structure cover just the basics. For convenience, most of the measures above are summarized in Table 1 below. There are, of course, many other measures of network structure—in fact, there are an infinite number of them. You can think of each measure as a function f that takes as input a graph G and outputs either a single scalar value or a vector of values. That is, each $f(G)$ projects the space of all possible graphs onto a low-dimensional space. As a result of this projection, information is surely lost and thus each measure $f(G)$ is an incomplete description of the original graph G. A good function f projects graphs in a way that highlights the structural feature of interest by creating good separation among points in the low-dimensional space.

Because they are all measuring different aspects of the same network, most network measures are highly correlated with each other. For instance, having a high degree tends to be inversely correlated with having a high local clustering coefficient, but having many high-degree vertices in a network tends to correlate with having a small diameter. Etc.

3 Supplemental readings

1. Read Chapter 6.1–6.4, and 6.6–6.12, and 7.3–7.4 in *Networks*

network measure	scope	graph	definition	explanation
degree	L	U	$k_i = \sum_{j=1}^{n} A_{ij}$	number of edges attached to vertex i
in-degree	L	D	$k_i^{\text{in}} = \sum_{j=1}^{n} A_{ji}$	number of arcs terminating at vertex i
out-degree	L	D	$k_i^{\text{out}} = \sum_{j=1}^{n} A_{ij}$	number of arcs originating from vertex i
edge count	G	U	$m = \frac{1}{2}\sum_{ij} A_{ij}$	number of edges in the network
arc count	G	D	$m = \sum_{ij} A_{ij}$	number of arcs in the network
mean degree	G	U	$\langle k \rangle = 2m/n = \frac{1}{n}\sum_{i=1}^{n} k_i$	average number of connections per vertex
mean in- or out-degree	G	D	$\langle k^{\text{in}} \rangle = \langle k^{\text{out}} \rangle = 2m/n$	average number of in- or out-connections per vertex
reciprocity	G	D	$r = \frac{1}{m}\sum_{ij} A_{ij}A_{ji}$	fraction of directed edges that are reciprocated
reciprocity	L	D	$r_i = \frac{1}{k_i}\sum_{j} A_{ij}A_{ji}$	fraction of directed edges from i that are reciprocated
clustering coefficient	G	U	$c = \frac{\sum_{ijk} A_{ij}A_{jk}A_{ki}}{\sum_{ijk} A_{ij}A_{jk}}$	the network's triangle density
clustering coefficient	L	U	$c_i = \sum_{jk} A_{ij}A_{jk}A_{ki} \,/ \binom{k_i}{2}$	fraction of pairs of neighbors of i that are also connected
diameter	G	U	$d = \max_{ij} \ell_{ij}$	length of longest geodesic path in an undirected network
mean geodesic distance	G	U or D	$\ell = \frac{1}{\binom{n}{2}}\sum_{ij} \ell_{ij}$	average length of a geodesic path
eccentricity	G	U or D	$\epsilon_i = \max_i \ell_{ij}$	length of longest geodesic path starting from i

Table 1: A few standard measures of network structure. The "scope" of a measure indicates whether it describes a vertex-level (local, L) or network-level (global, G) pattern, and the "graph" column indicates whether it is calculated on an undirected (U) or directed (D) graph. In each definition, adjacency matrix notation is used and it is assumed that $A_{ij} = 1$ if the edge (i, j) exists in the network and $A_{ij} = 0$ otherwise. For the geodesic paths, we assume that the graph is connected.