2 Network Metrics

A valuable feature of networks is their ability to quantify structural relationships. Much like a chemist, who, having purified their sample, can bring to bear a wide range of tools to examine its qualities, a network scientist, having transformed their data into a network, can leverage a wide range of tools to understand its structure.

Network analysis allows us to do this at three different scales:

- Micro: What are the structural properties of individual nodes and edges?
- Meso: What are the structural properties of local communities or **subgraphs** (i.e., subsets of the network)?
- Macro: What are the structural properties of entire networks?

By tuning our analysis across these three different levels, we can bring different features of the network into focus. For example, nodes have structural features such as how many neighbors they have (degree) and how far away they are from other nodes (shortest path length). Whole networks have the average of their individual nodes' features, but they also have graph-level properties such as overall connectivity (density), the distinctiveness of their communities (modularity), and the likelihood that similar nodes connect with one another (assortativity). Subgraphs – which can be identified using various methods – can be evaluated both at the node and graph level, as well as compared with one another.

The number of possible measurements is vast and there are too many to cover in any single book. There are many excellent resources for learning more than are presented here (e.g., Barabási & Posfai, 2016; Menczer et al., 2020; Newman, 2018). More pragmatically, it is not uncommon to develop one's own measures that are fit to purpose, and this book provides several examples of that as well. Here we will cover some of the most popular off-the-shelf measures as these are the ones researchers are most likely to encounter and use. Table 2.1 provides a quick reference for the nomenclature. These also provide a solid foundation for understanding and developing others. I will present these in a concise format, with brief examples and accompanying online code, noting that we will cover these in more applied cases later.

2.1 Nodes, Edges, and Density

Leo Tolstoy once noted that "all happy families are alike; each unhappy family is unhappy in its own way." If we consider a network as a kind of family, and happy families as being made up of individuals that are all happy with one another, then we can translate our

Table 2.1 Basic network measures.

Variable	Definition
N	Number of nodes
E	Number of edges
ρ	Density
L	Average shortest path length
D	Diameter
C	Clustering coefficient
k	Degree
b	Betweenness centrality
c	Closeness centrality
x	Eigenvector centrality
r	Assortativity
Q	Modularity

happy family into a network with edges between every pair of nodes. Such a network is a **fully connected graph** or a **complete graph**. When edges go missing, the house becomes divided, and the possible ways in which this can happen proliferate.

A graph-level measure of this division is **density**, which is the number of observed edges divided by the number of possible edges. For an undirected network, it is therefore the probability that if you select a random pair of nodes, you will find an edge between them.

If an undirected network is fully connected – all nodes are connected with one another – the density is 1. Groups of nodes that have this property are called a **clique**: they are fully connected and therefore have a density of 1 among themselves.

If there are no connections between nodes, this is an empty graph and the density is 0. Nodes that are completely unconnected are called **isolates** (or **hermits**).

How many possible edges are there for a network of size N? A square adjacency matrix with N nodes has N rows and N columns and is therefore size $N \times N$. If every node can have a directed edge to every other node, including itself, then every cell in the adjacency matrix could have a value of 1, giving us $N \times N = N^2$ possible edges. This powers *Metcalfe's law*: the value within network relationships grows as N^2.

If we don't include self-loops, then we subtract the diagonal, which gives us $N^2 - N = N(N-1)$, which is the possible number of edges in a directed network without self-loops.

An undirected network has only one edge possible between every pair of nodes; recall that it is symmetric and therefore $A_{ij} = A_{ji}$. So it has half as many possible edges as a directed network. The total number possible is then $N(N-1)/2$ for an undirected, unweighted network with no self-loops.

Thus, we can define density, ρ, for an undirected network as:

$$\rho = \frac{2E}{N(N-1)} = \frac{\textit{observed edges}}{\textit{possible edges}}$$

By definition, as the density increases it becomes more likely that nodes in a network will be connected together.

2.1.1 Paths and Path Length

"Six degrees of separation" and "it's a small world after all" reflect our surprise that people are often more socially near one another than we anticipate. In network terminology, we say that the **path length** is short. The number of edges we would need to traverse to get from one person to the other is small.

The shortest path between nodes *i* and *j* is called their **geodesic distance**, and it is the minimum distance in edges needed to travel between them. It can be written as d_{ij}. In a simple network, the distance is the count of edges. If the network is directed, the path must follow the direction of the edges.

The dark edges in panel (a) of Figure 2.1 show that the shortest distance between node 1 and node 5 is 3. There can be more than one shortest path if several paths have the same length. For example, the path here could also travel from $1 \rightarrow 3 \rightarrow 6 \rightarrow 5$.

If two nodes are not connected by a path, the path is infinite.

The **average shortest path length**, *L*, for a network is the average of the geodesic distances between all pairs of nodes that have a finite path length. Figure 2.2 provides an example with each node labeled with its average shortest path length. Table 2.2 provides the corresponding matrix of path distances for each node to every other node.

The **diameter**, *D*, of a network is the longest path length we would have to travel to get between the two most distant nodes if we took the shortest path between them. It

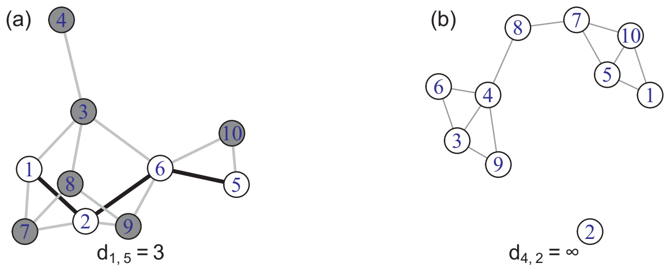

Figure 2.1 Examples of path length.

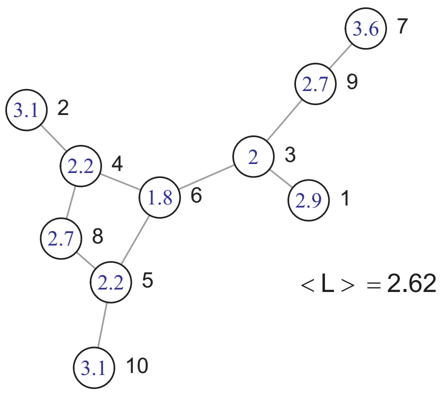

Figure 2.2 The average shortest path length. Each node is labeled with its average shortest path length to all other nodes (inside the circle). The average of these is the average shortest path length for the network. Each node is also labeled to the right with its node ID from Table 2.2.

Table 2.2 The distance table for the graph shown above. Each cell indicates the shortest path between the nodes indicated by the row and column.

	1	2	3	4	5	6	7	8	9	10
1	0	4	1	3	3	2	3	4	2	4
2	4	0	3	1	3	2	5	2	4	4
3	1	3	0	2	2	1	2	3	1	3
4	3	1	2	0	2	1	4	1	3	3
5	3	3	2	2	0	1	4	1	3	1
6	2	2	1	1	1	0	3	2	2	2
7	3	5	2	4	4	3	0	5	1	5
8	4	2	3	1	1	2	5	0	4	2
9	2	4	1	3	3	2	1	4	0	4
10	4	4	3	3	1	2	5	2	4	0

is therefore the longest of the shortest paths. It is analogous to measuring the diameter of a circle.

The path length of a weighted network can be computed in terms of number of edges (as noted earlier) or it can use the weights of edges. How we use the weights depends on what the weights represent. If the weight represents some measure of closeness (such as similarity), then to correspond to distance it should be construed as the inverse of the weight, $1/w$. If the weight indicates a form of distance along the edge, then path length can be computed straightforwardly as the sum of the weights.

Because infinite paths are computationally intractable, the average shortest path length is computed among only those nodes that share a path – that is, they are **reachable**. This can create some curious behavior. Imagine a network that changes its structure by breaking into dyads, like a dance party. When this happens the average shortest path length is 1, but most nodes are not connected at all. The average shortest path length would then become a poor indicator of the connectivity of the network. In general, as nodes become unreachable via paths, average shortest path length becomes a less desirable measure of network structure. Some other measure of connectivity is likely to be better, such as the number of components.

2.2 Components

Various estimates suggest there are on the order of 100 groups of Indigenous peoples who remain uncontacted by the larger non-Indigenous society we might think of as the world community. The majority of these isolated groups live in South America and, because they remain uncontacted, their true numbers are difficult to estimate. In the small world experiment of Stanley Milgram (1967), in which people attempted to send letters via acquaintances to people they did not know, these Indigenous peoples could not have been included in the data because they are, by definition, unreachable. The six degrees of separation we often use to describe the structure of our social environment – and which was

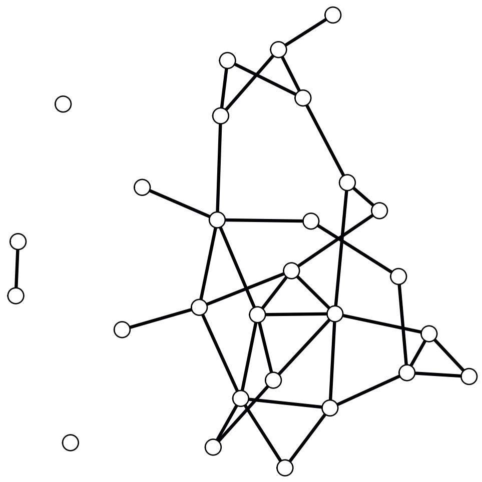

Figure 2.3 A network with 4 components.

in some ways confirmed by Milgram's experiment – does not really apply to everyone. It only applies to those who are reachable. Unreachable nodes are said to be in a different component.

A **component** is a collection of nodes that are all reachable, one to the other. Every node in a component can reach every other node in that component by traversing a path over consecutive edges. Figure 2.3 has four components: the largest or **giant component** with 24 nodes, a smaller **island** with 2 nodes, and two solitary isolates.

When a network is directed, a component is **strongly connected** when all its nodes are reachable by following the direction of path edges. A component is only **weakly connected** if some of its nodes are only reachable by swimming upstream – going the wrong direction along a directed edge.

The number of components in a network can be a useful measure of how disconnected a network is. Many network studies focus on resilience and attack tolerance by deliberately damaging networks to evaluate their resistance to fractionation into multiple components (see Chapter 13). The thresholding described in Chapter 1 is one example of this process for weighted networks.

When focusing on network-level measurements, many network analyses focus only on the giant component. This helps avoid some of the measurement issues for multicomponent networks, such as infinite path lengths.

Components are a form of meso-scale or community measure. Zooming out to the whole network we might see it is made up of multiple components. But we can also zoom in on a single component to see that some communities within it are collections of nodes that are all directly connected with one another: a clique. This should give you a sense of the richness of this meso-scale, of which we have only touched the surface.

2.2.1 Network Size, Edge Probability, and Components

When networks change their size, they can change their structure. They can do this even when the underlying processes in the network are the same. Consider a social constraint, like *Dunbar's number*, which suggests that humans can meaningfully hold relationships with on the order of 100 people. In a network of 100 people all exercising their Dunbarian extroversion, the density would be 1 – everyone knows everyone well. But in a 1,000 person network, the density would be 0.1. We would make an error in inference if we concluded that people were less social in larger networks because the density is lower. One would want to use a node-level measure to evaluate that.

If two different networks of the same size had different densities, the structural inference is less problematic. But when they are different sizes, things can become more confusing.

Consider isolates. The probability that any node remains an isolate is the probability that it does not share an edge with any other node in the network. One might imagine that the number of isolates should increase as the size of a network increases, but the intuition is backward if the probability of an edge remains the same. Suppose the probability of an edge is p. Then the probability of not sharing an edge with a specific other node is $(1 - p)$.[1] As there are $N - 1$ other nodes, the probability that a node is not connected to

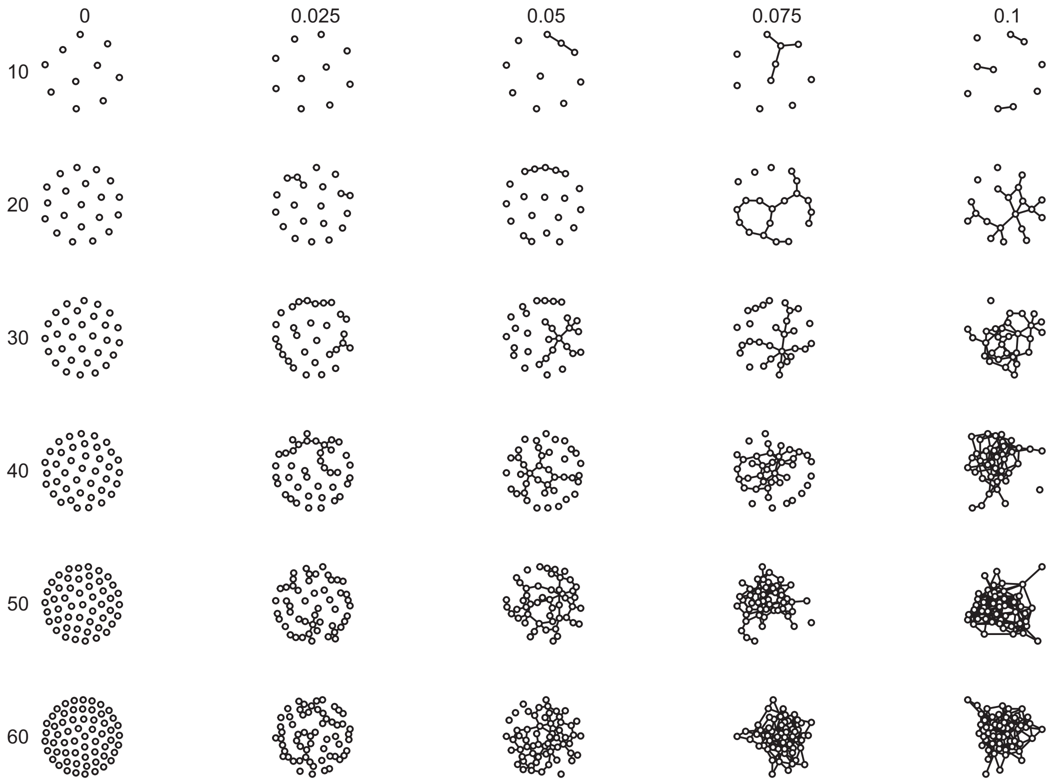

Figure 2.4 The influence of density and number of nodes on components and connectivity. Number of nodes is listed along the rows. Edge probability is listed along the columns.

[1] The network generating probability is p. The density, ρ, is the probability we observe from the network data. As an example, the probability of an edge in a two-node network might be $p = 0.5$, but it will always have an observed value ρ of either 1 or 0.

any other node is $(1-p)^{N-1}$. Because this approaches 0 as N increases, the probability that a node is an isolate shrinks as the size of the network increases. If we assume the probability of forming an edge is fixed then the more nodes there are to connect with, the more likely it is that there will be a connection.

This is science that any party-goer can understand. It is the basis of *the birthday paradox*: even if the probability that two people share a birthday is a mere $1/365$, the number of possible shared birthdays (edges) in a group of 23 people is 253. Therefore, the probability that at least two people will share a birthday in a group of size 23 is $1-(1-1/365)^{(23\times 22)/2}=0.5$. This is another consequence of Metcalfe's Law.

Figure 2.4 shows random networks generated for different edge probabilities and numbers of nodes. As the number of nodes increases, the size of the giant component gets larger and the number of isolates falls. Similarly for the probability of an edge, as the probability increases, giant components get larger and the isolates fall away.

2.3 Centrality

When we are more concerned with the structural properties of nodes than with the network as a whole, we are interested in measures of **centrality**. These vary from local measures that only look at relationships with direct neighbors (e.g., degree) to global measures that involve computing relationships with all other nodes in the network (e.g., betweenness and closeness). Understanding how nodes differ along these measures gives us a sense of the diversity in the network and how nodes differ in importance. Centrality measures therefore give us a node's-eye view of the network. Averaging over them gives us a sense of the average microlevel structure of the network.

There are many centrality measures because there are many different ways of measuring structural relationships.[2] They are each relevant to different structural questions. We cover the more common ones here and more specialized ones later in the book.

2.3.1 Degree

When we say that someone is an extrovert, what we mean is that they have many social relations: their degree is high. An introvert, on the other hand, has a low degree by comparison. Formally, the **degree** of a node, k, is the number of links it has to other nodes.

Hubs are nodes with a larger degree relative to other nodes in a network. Isolates have a degree of 0. In Figure 2.5, every node is labeled with its degree. For most networks, different nodes will have different degrees. A **regular graph** has nodes all of the same degree.

There is a clear relationship between degree and density. To see this, consider the three-node network shown in panel (a) in Figure 2.6. The maximum possible average degree would occur if each node is connected to the other two nodes and therefore has a degree

[2] The Periodic Table of Network Centrality gives a sense of the breadth available here, and this does not cover them all: www.schochastics.net/sna/periodic.html.

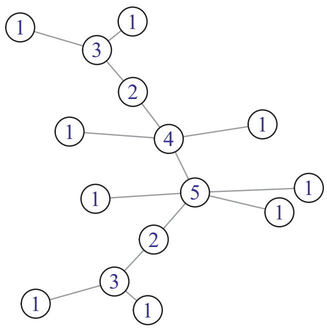

Figure 2.5 Degree. Each node is labeled with its degree.

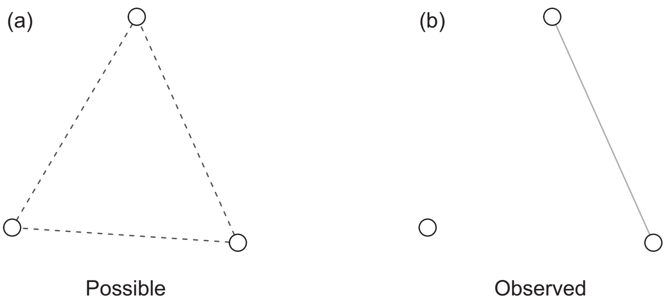

Figure 2.6 Density. Possible versus observed edges in a network with density 1/3.

of 2. In panel (b), the network only has one edge: two nodes have a degree of 1 and one has a degree of 0. The average degree is then $(1 + 1 + 0)/3 = 2/3$. The average degree, $2/3$, divided by the maximum possible degree, 2, is $1/3$. This is equivalent to the density: A three-node network has three possible edges. If it has one edge, then the density is $1/3$.

For undirected networks, the degree of node i is indicated by k_i and is the sum along the row *or* column of the adjacency matrix:

$$k_i = \sum_j A_{ij} = \sum_j A_{ji}$$

In the equation, A_{ij} is the value in the adjacency matrix that corresponds to the edge value between node i and node j. Here, Σ_j means to add up all the values of A_{ij} for all values of j.

Indegree and Outdegree If the network is directed, then each node will have an indegree and an outdegree. **Indegree** is the number of edges directed toward a node. **Outdegree** is the number of edges directed outward from a node. In Figure 2.7 each node is labeled with its indegree and outdegree.

Summing across a row gives the outdegree:

$$k_i^{out} = \sum_j A_{ij}$$

Table 2.3 The adjacency matrix for a directed network showing the row sums (outdegree) and column sums (indegree), which, when summed over all nodes, are equivalent.

	1	2	3	4	5	6	7	8	9	10	Row sums
1	0	1	1	0	1	0	0	0	0	0	3
2	1	0	0	0	0	0	0	0	0	0	1
3	0	1	0	0	0	0	1	0	0	1	3
4	0	1	0	0	1	0	1	1	0	0	4
5	0	0	0	0	0	0	1	0	1	0	2
6	1	1	0	0	0	0	1	1	1	0	5
7	1	1	0	0	0	1	0	0	0	0	3
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	1	0	0	0	0	0	0	1
10	0	0	0	0	1	0	0	0	0	0	1
Column sums	3	5	1	1	3	1	4	2	2	1	23

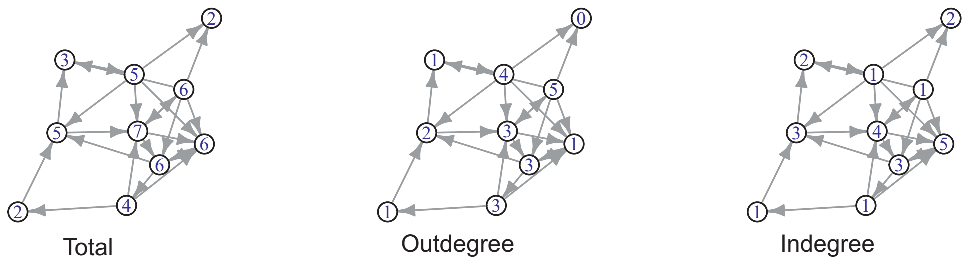

Figure 2.7 Indegree and outdegree alongside total degree. Each node is labeled with its total degree, indegree, or outdegree.

Summing down a column gives the indegree:[3]

$$k_i^{in} = \sum_j A_{ji}$$

Adding indegree and outdegree gives the total degree:

$$k_i^{total} = k_i^{in} + k_i^{out}$$

The average degree for the network is the sum of degrees over nodes divided by the number of nodes:

$$< k >= \frac{1}{N} \sum_{i=1}^{N} k_i$$

The number of out-directed edges equals the number of in-directed edges. Therefore, the average indegree and outdegree are the same, as shown in Table 2.3.

[3] A_{ij} is defined here as an edge from i to j. Some texts invert this for mathematical efficiency, assigning A_{ij} as an edge from j to i. It is best to check the source carefully, noting that sources will vary, as do computational tools.

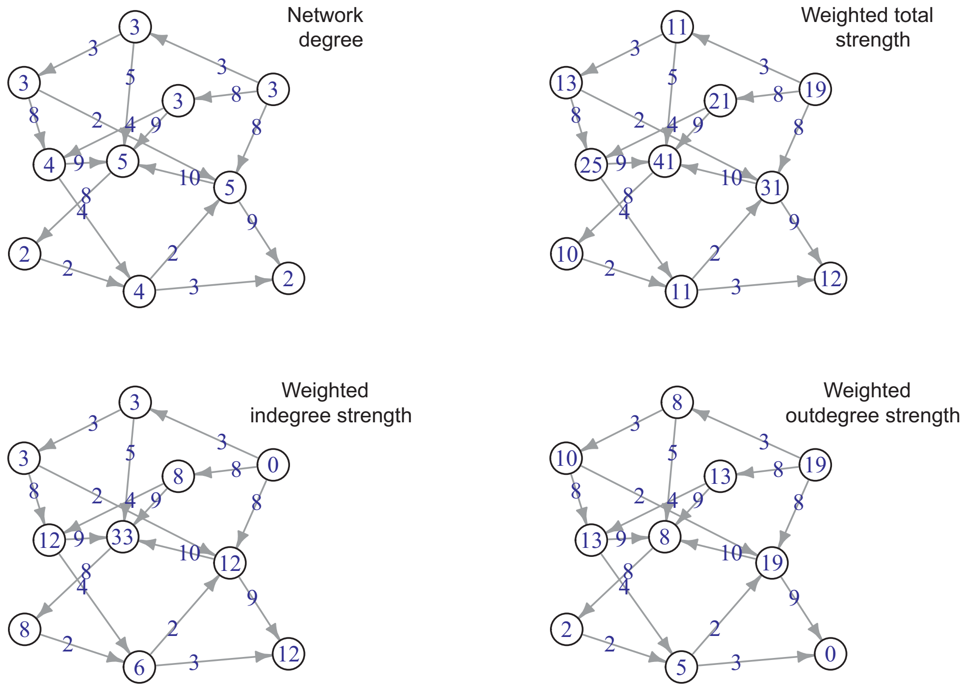

Figure 2.8 Various ways of computing the degree in directed and weighted networks. Nodes are labeled with their relevant degree or strength.

Strength When edges have weights, we want to be able to evaluate the overall strength of a node's relationships with other nodes. We can of course compute the unweighted degree for a weighted network by simply counting edges, not weights. But we can incorporate weights into the degree by summing the relevant weights. This is called **strength**. Networks that are both weighted and directed will have strengths associated with outdegrees and indegrees. See Figure 2.8.

2.3.2 Clustering Coefficient and Transitivity

If all my friends' friends are my friends, then we form a group of people who all know one another. Formally, we are a clique. Less formally, we are a cluster. Clustering is often formally computed using the **clustering coefficient** which measures the extent to which a node's neighbors are themselves neighbors.

The clustering coefficient has two forms. These view clustering from different vantage points. The first is from the perspective of an individual node – it is a node-level measure called the **local clustering coefficient**. This measures the proportion of a node's neighbors that are neighbors of one another. That is, it measures the extent to which friends are friends. See Figure 2.9

If all of a node's neighbors are connected to one another by an edge, then the clustering coefficient is 1. If all are unconnected, the clustering coefficient is 0.

Mathematically, the clustering coefficient for a node is the observed number of edges, e, between its k neighbors over the number of possible edges between its k neighbors. Like

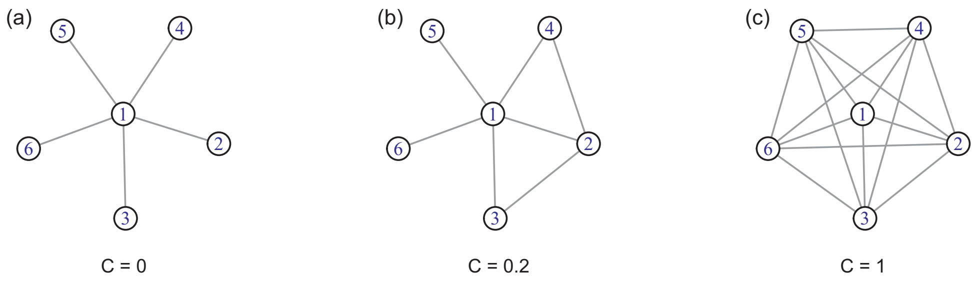

Figure 2.9 Local clustering coefficient. The clustering coefficient for the node (1) in panel (b) is shown below the network.

the network as a whole, the number of possible edges between k nodes is $k(k-1)/2$. Thus, the clustering coefficient is:

$$C_i = \frac{2e}{k_i(k_i - 1)}$$

Once we have the local clustering coefficient for individual nodes, we can average over all nodes to get the average clustering coefficient for the graph.

A second and different way to get a graph-level measure of clustering coefficient is called transitivity.[4] **Transitivity** measures the extent to which friends of friends are friends. However, rather than measure this for individual nodes, transitivity measures this for the graph as a whole. Transitivity does this by identifying all cases where two nodes share a common neighbor and are therefore friends of friends. Then it simply takes the proportion of all these cases where the friends of friends are friends.

In network language, these cases where two nodes share a common neighbor are called a **triplet**. The two nodes at the end of the triplet are friends of a friend. They either share an edge or not. If they share an edge, they make a closed triangle and are called a **transitive triplet** (see Figure 2.10).[5] If they don't share an edge, they are an intransitive triplet and form a 'Λ' shape: what is also called a **2-star**. A 2-star network has a single center node and two nodes that radiate out from it (who could be connected, or not).[6] So every transitive triplet has three 2-stars and every intransitive triplet has one. Transitivity measures the proportion of 2-stars in the network that are transitive (i.e., a triangle).

We can represent this iconically as follows:

$$T = \frac{3\Delta}{\Lambda}$$

[4] The function for computing local and global clustering coefficient in the online code uses igraph and is called 'transitivity'. It is best to ask for the local clustering coefficient or transitivity explicitly, so as not to get confused.

[5] Transitivity comes from the mathematical meaning of the term. If A = B and B = C then A = C, which is a consequence of the transitive property of mathematics.

[6] More generally, a **star network** is a network with a single hub node and all other nodes with degree 1 that radiate out from it.

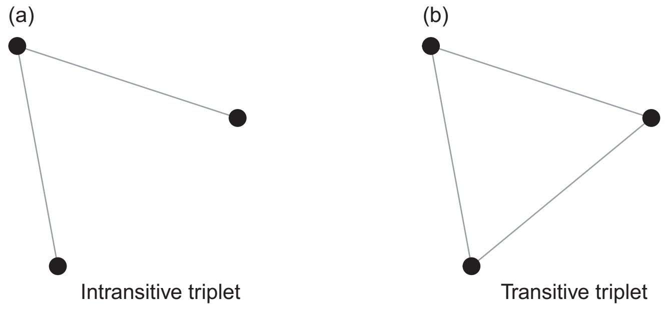

Figure 2.10 Intransitive and transitive triplets. Transitivity is only computed using triplets with either of these two formations. In panel (a), we have one 2-star. In panel (b), the transitive triplet has three 2-stars.

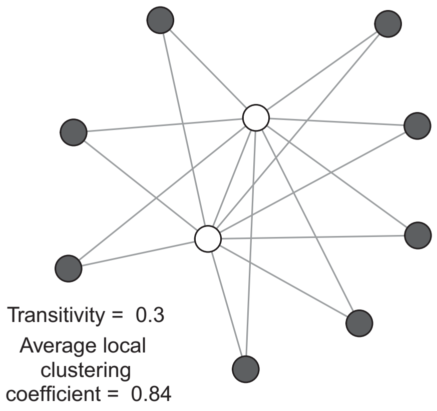

Figure 2.11 The wheel network demonstrates the difference between transitivity and average local clustering coefficient. As the outer nodes increase, the average local clustering coefficient approaches 1 and the transitivity approaches 0.

The total number of 2-stars is designated by Λ. The total number of transitive triplets is Δ. Because each transitive triplet contains three 2-stars, we multiply the number of triplets by three to account for all the 2-stars they make transitive.

Transitivity and the average local clustering coefficient are not the same. They can diverge. The majority of individual nodes may all have neighbors who are connected – making the average local clustering coefficient high – even though the majority of triplets are non-transitive – making the transitivity low. Figure 2.11 demonstrates this. For each new gray node added that connects with the two inner white nodes, the average local clustering coefficient rises toward 1. This is because each new gray node has a local clustering coefficient of 1. But each new gray node also adds many new intransitive 2-stars, causing the transitivity of the network as a whole to fall toward 0.

If people (or nodes) have only one friend, then there is no question about friends of friends – they do not exist. As a consequence, local clustering coefficient and transitivity are undefined for nodes of degree less than two; groups of isolates or dyads (two nodes connected by an edge) have neither an average local clustering coefficient nor a transitivity.

Because graph-level transitivity and average local clustering coefficient are not the same thing, I prefer to use transitivity when referring to the graph-level measure and average clustering coefficient (or clustering coefficient) when referring to the local measure.

2.3.3 Closeness Centrality

If you were looking to live near an airport with the shortest flights to all other cities, you would be looking for an airport with high closeness centrality. If you were hoping to identify a social influencer in a social network whose influence would spread most rapidly to others, you would again be looking for someone with high closeness centrality.

Closeness centrality is defined as 1 over the sum of the shortest path lengths to all other nodes:

$$c_i = \frac{1}{\sum_{j=i}^{N} d_{ij}}$$

Figure 2.12 shows three networks with each node labeled with its closeness centrality. Taking an end node from the network in panel (a), the sum of the distances to the two other nodes is $1 + 2 = 3$. The inverse of this gives the closeness centrality: 0.33. The same logic applies to the middle node, which gives us 0.5.

In the network in panel (b), nodes are again labeled with their closeness centrality. There are two things to notice here. First, closeness centrality goes down with the number of nodes included in the distance summation. Larger components will generally have lower average closeness centrality. Second, an island can contain nodes that have a closeness centrality of 1 even though they are unreachable from the majority of the nodes. Closeness is only calculated for reachable nodes. As a consequence, it is mainly only meaningful for nodes within a single component.

The network in panel (c) is the same as the middle, except here closeness centrality is normalized. The normalized closeness centrality is the same as the inverse of a node's average shortest path length. We can see this somewhat plainly if we consider the gray node near the center. One over its closeness centrality gives us $1/0.58 = 1.72$, which is the number of edges on average needed to travel to each of the other nodes in that component.

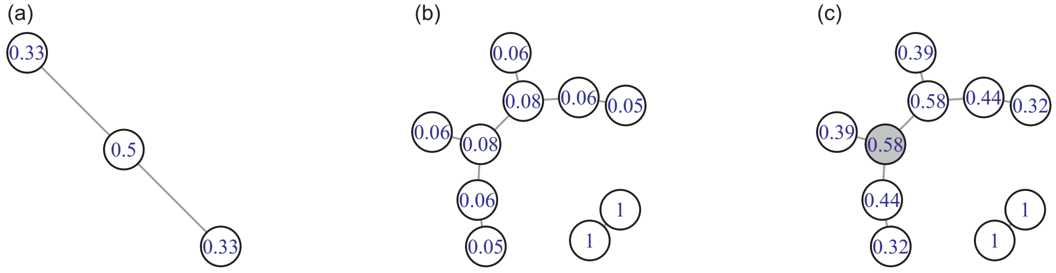

Figure 2.12 Closeness centrality. Each node is labeled with its closeness centrality.

2.3.4 Betweenness Centrality

When a node lies on the shortest path between many other pairs of nodes it has high betweenness centrality. Cities along the Silk Road, such as Xi'an and Constantinople, became what they are today because they had high betweenness centrality along early trade routes. Many port cities have benefited from the same middle-man status in economic trade. This is not always a good thing. Mexico's Juárez is a city with high betweenness in the drug trafficking trade, falling along the border between the USA and Mexico. Its high betweenness made it one of the murder capitals of the world as cartels fought for control of the city.

The **betweenness centrality** for a node is the number of shortest paths between all other nodes that pass through it. Formally, this is computed as:

$$b_i = \sum_{i \neq j \neq k} \frac{\sigma_{jk}(i)}{\sigma_{jk}}$$

Here, $\sigma_{jk}(i)$ represents the number of shortest paths that pass from j to k through i. To account for the fact that sometimes there are multiple shortest routes, we divide by the total number of routes: σ_{jk}. If all the shortest paths between j and k pass through i, this will add 1 to the betweenness centrality. If some don't pass through i, then the increase in betweenness for node i is the fraction of shortest paths that pass through i. Figure 2.13 labels each node with its betweenness centrality.

Betweenness centrality increases with the size of the network. As betweenness is the sum of paths, more nodes means more paths. To resolve this, betweenness is often normalized by dividing by the maximum possible number of shortest paths. If all paths pass through a node, as they do for the central node in Figure 2.13, the normalization scales its betweenness value to 1. Using the same logic, betweenness centrality can also be computed for edges. Wherever many paths converge to cross from one place to another, the betweenness is high.

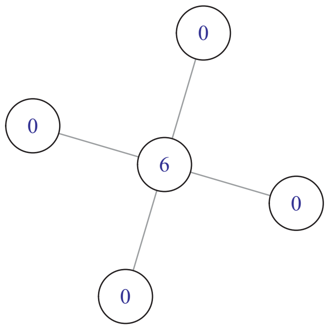

Figure 2.13 Betweenness centrality. Each node is labeled with its betweenness centrality – the number of shortest paths between other nodes that pass through it. Six paths travel through the central node, but none pass through the outer nodes.

2.3.5 Eigenvector Centrality

Eigenvector centrality measures how important a node is based on how important the nodes are that it is connected to. To provide an intuitive example, if a person has only one friend (degree one), they are still likely to be a more important person if their one friend is Paul McCartney than if it is, say, Jim Kooti from southern Oklahoma. Everyone knows Paul. Many fewer know Jim. Eigenvector centrality measures this difference in prestige. To be prestigious, one must receive prestige from others with prestige.

The definition is recursive. It requires that we know how prestigious nodes are before we can compute how prestigious nodes are. Like many chicken-and-egg problems, this one can be solved algorithmically by taking a random guess and then letting the network work out the implications. Assign nodes random prestige values and then allow them to transfer this to their neighbors. Receiving neighbors then take the prestige they receive and send it out to their neighbors. Repeated several times, the relative prestige values will stabilize. Because many people are likely to give prestige to Paul, and few give it to Jim, Paul can in turn give more prestige than Jim. If you're only going to have one friend, pick someone with good eigenvector centrality.

Of course, eigenvector centrality is not specific to prestige. It applies to any adjacency matrix and it is easily solved in practice using matrix algebra. Given a vector of values, $\mathbf{x}$, representing the prestige of every node such that the highest value is 1, we can multiply this by the adjacency matrix, $\mathbf{A}$. This distributes prestige to the neighboring nodes. When this distribution is stable, the product of this multiplication will be the original vector, $\mathbf{x}$, multiplied by a scaling factor, λ. The scaling factor is called the eigenvalue. Formally, that is:

$$\mathbf{Ax} = \lambda\mathbf{x}$$

Software like R will handle this readily, and some example eigenvector centrality measures are shown in the networks in Figure 2.14.

Eigenvector centrality is often most useful for undirected networks. **PageRank** and **Katz centrality** are useful alternatives for directed networks. Each makes slightly

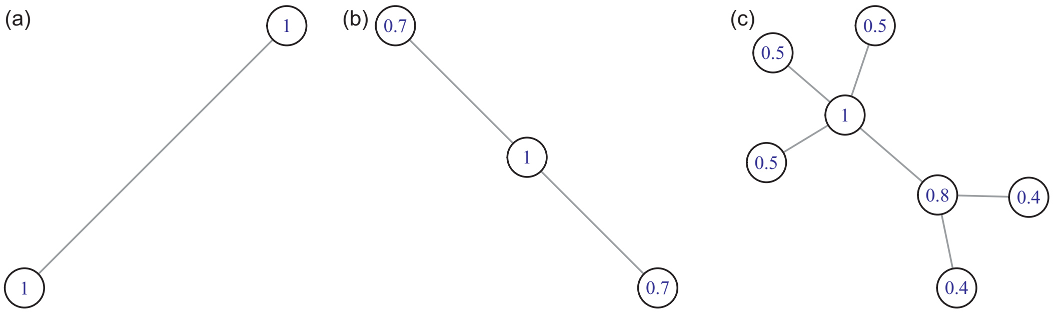

Figure 2.14 Eigenvector Centrality. Each node is labeled with its eigenvector centrality. Note that in the network in panel (c), nodes with the same degree can have higher scores if they connect with nodes of higher eigenvector centrality.

different assumptions about how value is distributed, but all are fairly equivalent in their underlying logic. Chapter 19 discusses these measures in more detail.

2.3.6 Comparison of Centrality Measures

It is useful to see centrality measures side by side. I have created the Mouse to demonstrate these different centrality measures, shown in Figure 2.15. It is modeled after the pocket mouse of the desert Southwest and it gives some personality to these measures, which emphasize the head, body, or tail, for example.

The Mouse is inspired by **Krackhardt's kite network** (Krackhardt, 1990). The Kite is provided here for further comparison (Figure 2.16). Together the Mouse and the Kite should help crystallize these centrality measures and where you might be likely to find them.

There is no reason not to consider developing your own centrality measures. Often a specific behavioral problem will require a measure that is fit to purpose. It is common practice to do this to solve specific problems when you are unable to find an existing measure.

In summary, high degree nodes can be found anywhere. High clustering coefficient nodes are found in communities (small or large). High closeness nodes are found near the center of the network. High betweenness nodes tend to separate different communities. And high eigenvector nodes tend to occur near higher degree nodes in assortative communities, which is discussed next.

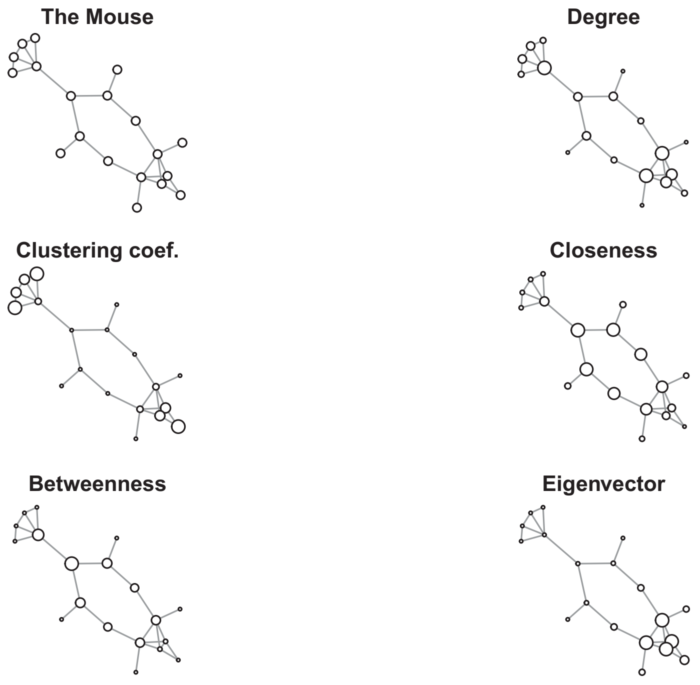

Figure 2.15 Comparison of the Mouse's centrality measures. Node sizes are scaled relative to the maximum and minimum of the centrality measure to highlight the differences.

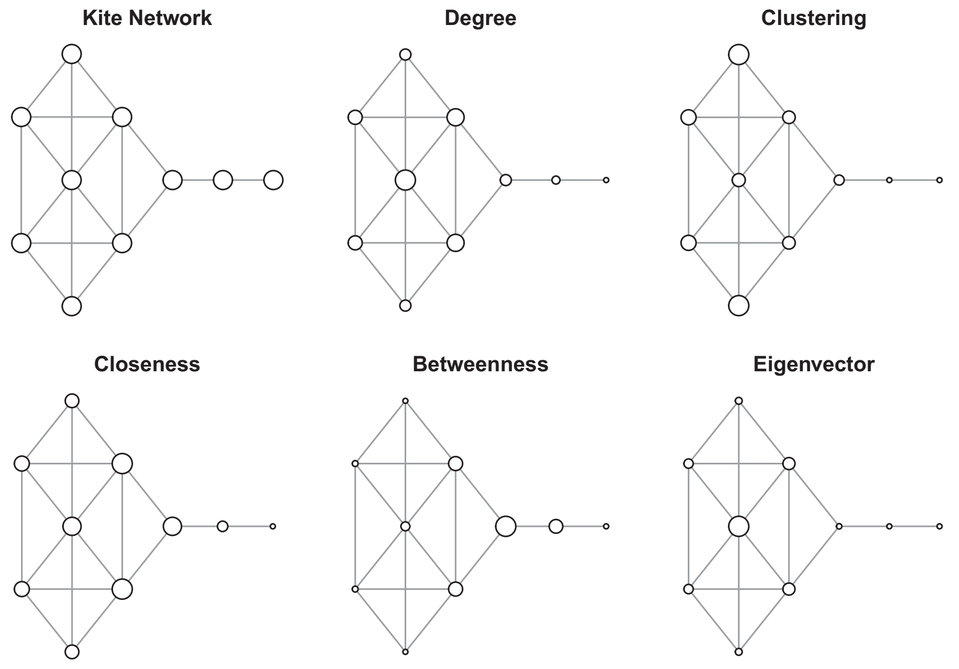

Figure 2.16 Comparison of centrality measures using Krackhardt's Kite. Node sizes are scaled relative to the maximum and minimum of the centrality measure to highlight the differences, as for the Mouse in Figure 2.15.

2.4 Assortativity and Homophily

Assortativity evaluates the degree to which nodes with similar properties connect with each other. In social networks, this is known as *homophily*: "birds of a feather flock together." People tend to associate with others who share similar features. Network analysis allows us to ask this about structural properties: for example, do nodes of a similar degree connect with one another? We can also ask about similarity in other attributes: for example, do individuals connect with others of a similar age or job?

To evaluate assortativity we compute an assortativity coefficient, r, which is the Pearson correlation between pairs of connected nodes in the network with respect to the value in question. To do this, we can generate an edge list from the network, replace the node labels with the value for each node, and take the correlation of the two columns of values.

The assortativity coefficient lies between -1 and 1, indicating the extent of the network's assortativity. When $r > 0$, the network has assortative mixing, or positive assortativity. When $r = 0$ the network is nonassortative. When $r < 0$ the network is disassortative or is negatively assortative – similar nodes are not connected.

Figure 2.17 shows two networks that differ in their assortativity coefficients. It is clear from the visualization that, in panel (a), nodes of similar sizes (indicating degree) are connected with one another. In panel (b), nodes tend to be connected with nodes of different sizes.

Assortativity can also be computed for node attributes. Figure 2.18 shows networks that are assortative or disassortative with respect to node color.

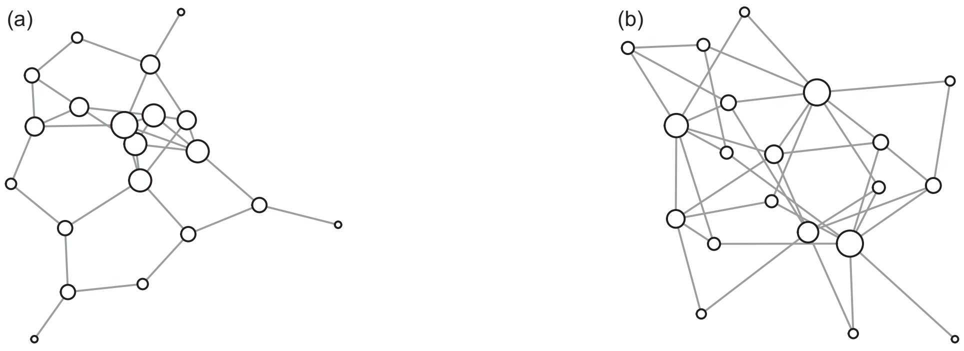

Figure 2.17 Two networks that are assortative or disassortative by degree. The assortativity coefficients are 0.45 (left) and −0.62 (right). Node sizes are scaled to reflect relative degree.

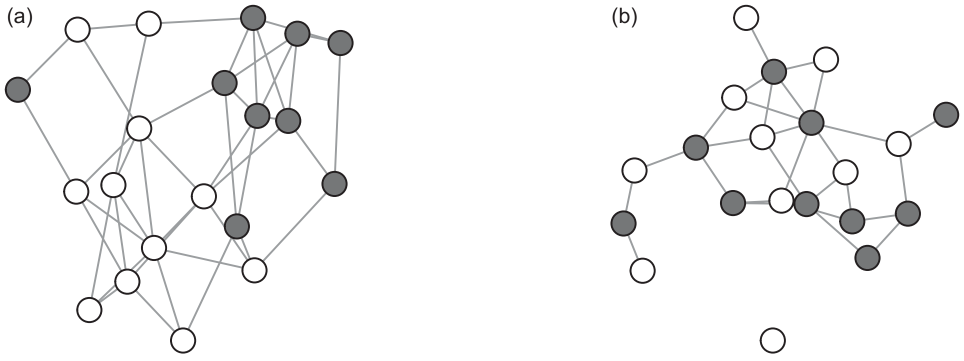

Figure 2.18 An assortative (left) and disassortative (right) network by color. The network in panel (a) has an assortativity coefficient of 0.57. The one in panel (b) is −0.65.

2.5 Community Detection

Networks are excellent for identifying communities. Given a set of people or things with some definition of edges, we can determine which nodes are in which communities as well as how many communities there are. This is powerful as it allows us to sort structured information into groups, identify communities who may be operating together, and predict relationships before they form. Social media platforms that always seem to know which of its many millions of users are our long lost friends are using community detection algorithms to identify them. Other recommender systems, which proffer us appealing movies, clothes, and videos, are also supported by community detection algorithms. With every choice we make, our community alignment becomes more well-defined and the recommendations get harder to refuse.

More formally, the **community detection problem** is one of identifying clusters of nodes that are more well connected to one another than they are to nodes in other communities. Dividing a network into sets of communities is called a **partition**. Identifying the partition that best satisfies the community detection problem is the goal. This is not trivial. The problem is that the number of possible partitions (the Bell number) grows rapidly with the number of nodes: If there are two nodes, there are two partitions: either the nodes are together or they are separate. If there are three nodes, there are five partitions: each node could be in its own community, two could share a community and the third could be on its own, or they could all be in the same community. With twelve nodes, there are more than four million partitions. The number quickly becomes computationally unfriendly.

The solution is to simplify and approximate. Different community detection algorithms make different assumptions to simplify the problem. Some methods are hierarchical clustering methods that assign nodes exclusively to specific groups (e.g., the Girvan-Newman method). Others allow nodes to belong to multiple groups (e.g., the clique percolation algorithm). Like centrality measures, they deserve a book of their own. Here we will cover a few of the more popular algorithms. Before we do that, it is first useful to describe modularity: the measure that allows us to recognize a good partition when we see one.

2.5.1 Modularity

If we saw some people at a party, it would be fairly easy to identify how many groups there were. We identify the groups by placing people who are close together in the same group. If we imagine that people who are near one another share an edge, then we can start to see how we might apply this logic to networks. We identify groups of nodes that share many edges among themselves and few edges with other groups of nodes. We can measure how good our grouping is using modularity. **Modularity**, Q, quantifies the difference between the observed links and the expected links within communities.

The mathematics to quantify this is based on two things. First, we can identify internal links within communities. Second, we can estimate the likelihood that we would see those internal links if edges were distributed at random.

To make this concrete, consider a network of two dyads: four nodes with two edges and each node with degree 1. Let our job be to assign nodes to two communities (let’s say black and white), each with two nodes. How many different ways are there to distribute these communities over the network? There are two ways, both shown in Figure 2.19: links can either be within communities or between communities.

If we have to assign communities – so that community members are more likely to be connected – we would do better to assign the same color to connected nodes. Modularity

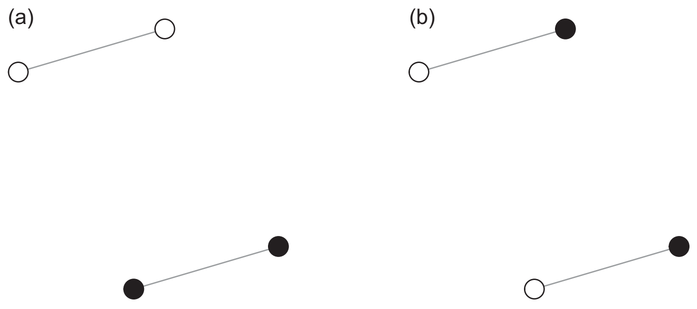

Figure 2.19 Modularity: Two possible network architectures for four nodes with two links, each node with degree 1, and two community assignments (black and white). The modularity for the network in panel (a) is Q = 0.5. For the network in panel (b), Q = −0.5.

tells us the same thing. If we observe links between nodes of the same color, we know there is a $1/2$ chance we would observe this even if the colors were distributed at random. The modularity, Q, is therefore 0.5. If we observe no edges between the same colors, we know there is a $1/2$ chance they could have been between colors. In this case, the modularity, Q, is -0.5.

Here is the equation that formalizes that math for networks of any size:

$$Q = \frac{1}{2E} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2E} \right] \delta(c_i, c_j)$$

We can walk through this as follows: First, the Kronecker delta function, $\delta(c_i, c_j)$ is 1 if two nodes, i and j, are in the same community: $c_i = c_j$. Otherwise, it is 0. So we are only concerned with cases where i and j are in the same community. Within the square brackets, A_{ij} represents the observed value of the edge between nodes i and j. The product of k_i and k_j divided by two times the number of edges is the probability of an edge between these two nodes observed at random. When considered over all pairs of nodes that are in the same communities, this gives us our observed edges versus our expected edges within communities. Finally, this is all normalized by two times the number of total edges, E. The result, in English, is that modularity is increased whenever two nodes in the same community are connected more than we would expect by chance and reduced whenever they are connected less than we would expect by chance.

Modularity makes the simplifying assumption that a community member's degree values are preserved. To help understand what this means, a three-node network with an edge between two group members and a third node not in the group has a modularity of 0. Because the third node has a degree 0, it can never share an edge with another node by chance. As a result, a set of group members who only have links between them and no links to a set of isolates has modularity 0. This is not always desirable or intuitive. Newman (2018) is a good source for additional thoughts on community detection.

The reason we need modularity is because community detection algorithms often generate many different partitions. Modularity can then be used to identify the partition that best assigns nodes to communities (Fortunato, 2010). Positive Q values that are closer to the maximum value of 1.0 indicate better community structure. In contrast, small or negative Q indicates poor community structure.

In practice, the Q values of large real-world networks such as the Internet and cellular phone networks are relatively high, ranging from 0.42 to 0.78 (Blondel et al., 2008). These values are found by using Q to identify the best partition produced by a community detection algorithm. In the following sections I describe several popular algorithms.

2.5.2 Community Detection Algorithms

2.5.2.1 The Girvan–Newman Method

The **Girvan–Newman Method** (Girvan & Newman, 2002), also called the **edge betweenness method**, is based on the observation that edges connecting separate communities should have high edge betweenness: the shortest paths between members of

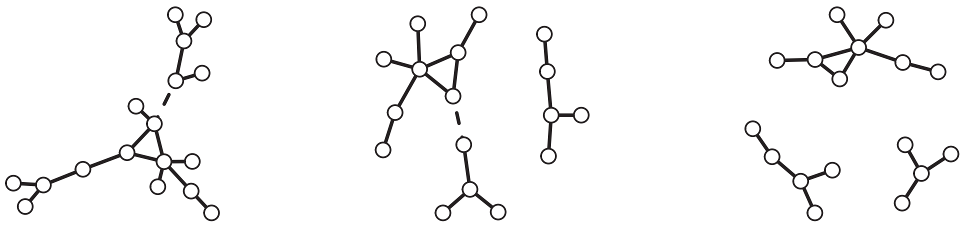

Figure 2.20 A series of partitions created by removing the edge with the highest edge betweenness. The edge to be removed is indicated with a dotted line. The partitions are created from left to right, each time removing the edge with the highest edge betweenness.

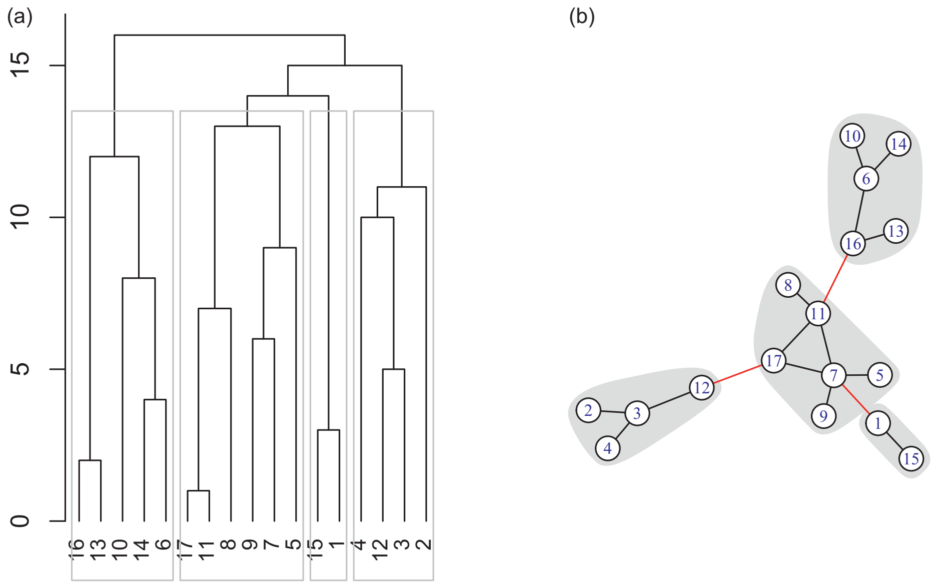

Figure 2.21 The hierarchy of communities identified by the Girvan–Newman method. Any horizontal division along the dendrogram on the left will produce a partition of the network into communities. The y-axis on the dendrogram shows the number of remaining edges for a given partition. The partition with the highest modularity is shown in panel (a) and identified by boxes in panel (b). The modularity of the community shown is 0.51.

different communities will pass through edges with high edge betweenness. This allows us to identify communities by sequentially removing the edge with the highest edge betweenness score. This in turn produces a hierarchy of partitions, which separate the network into smaller and smaller subnetworks until all nodes are isolates. Figure 2.20 shows a few steps in this process.

The hierarchy (or dendrogram) of partitions that is the outcome of this process is shown in Figure 2.21. By cutting the dendrogram horizontally, one identifies a specific partition. The partition in the hierarchy with the highest modularity is used to assign community membership to nodes. The partition with the highest Q is shown in the figure on the right.

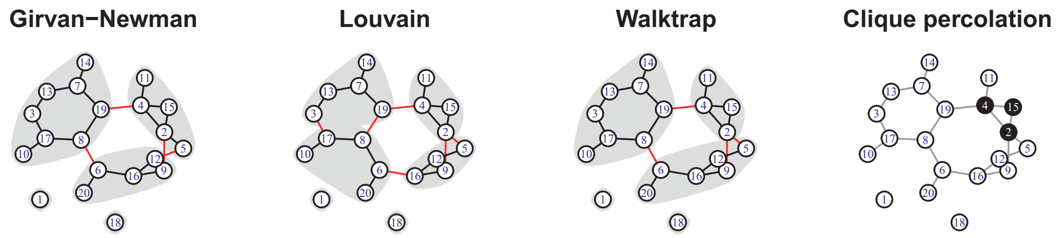

Figure 2.22 Comparison of different community detection algorithms. In the clique-percolation result on the right, a 3-clique is shown in black.

2.5.2.2 Other Methods

Community detection algorithms take a variety of forms. Some examples are shown in Figure 2.22. The **Louvain method** (Blondel et al., 2008) starts by assigning each node to its own individual community. Then communities (initially nodes) are chosen at random and pooled together with another community that best increases the modularity. This is repeated until there is only one community or until the modularity can no longer be increased. The partition with the highest modularity is chosen.

Random walker algorithms, like **Walktrap** (Pons & Latapy, 2005; see also Rosvall & Bergstrom, 2008), assume that random walkers moving between nodes via edges will tend to stay within communities more often than they pass between them. This allows the algorithm to identify a distance between communities using the movement patterns of random walkers. Partitions can then be sequentially identified based on the above distance assumption to produce a hierarchy of partitions. The hierarchy can then be evaluated using modularity.

The **clique percolation algorithm** (Palla et al., 2005) allows individual nodes to belong to multiple communities. It starts by identifying k-cliques, which contain k nodes that all share edges with one another. The smallest is a 3-clique or triangle. Communities are defined as sets of k-clique-adjacent k-cliques. k-cliques are k-clique-adjacent if they share $k - 1$ neighbors. This algorithm often assigns many nodes to isolate communities and preserves only those communities that are engaged in k-cliques of size 3 or more.

Note that some community detection algorithms (e.g., Louvain) apply themselves randomly to the nodes as they search for new partitions. As a consequence, the algorithm may not always return the same partition.