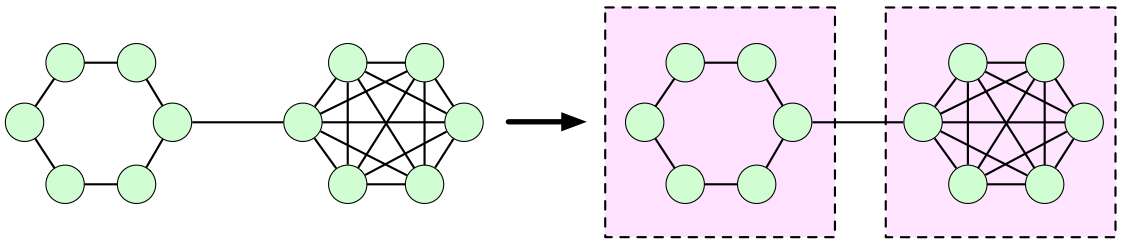
1 Large-scale structure in networks

Simple network-level measures can tell us a great deal about the shape of a network, e.g., the mean degree, the degree distribution, local or global clustering coefficients, edge reciprocity, mean geodesic distance or diameter, etc. However, these simple measures are not sensitive to how the structural pattern they measure might vary non-uniformly across the network. For example, reciprocity tells us the average number of bidirectional links in a directed network; the clustering coefficient gives the average number of closed connected triples attached to a vertex; etc. But, their specific values serve mainly to summarize the full distribution of structure across a network—this behavior is implicit in *any* average.

For instance, recall the ring-clique network from past lectures:



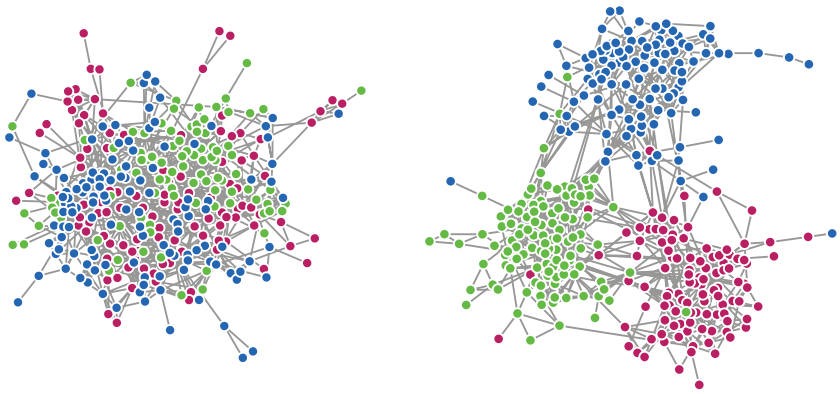
Although this network has a clustering coefficient of $C = 60/73 = 0.82$, nearly all of this value comes from the K_6 clique—the only part of the network with any triangles. If we had divided the network into two parts, this heterogeneity would be more easily revealed: the ring subgraph has a clustering coefficient of $C = 0$ while the clique subgraph has $C = 1$. (Do you see why the network value of C is not $C = 0.5$?)

Vertex-level measures do provide some insight into structural heterogeneity, as each calculates some property of a vertex's local environment. However, while network-level measures essentially over-aggregate or smooth out a network's heterogeneity, vertex-level measures do the opposite: they disaggregate heterogeneity so much that they can hide the tendency of vertices with similar measures to be found close to each other in the network.

Instead, what we want is to directly identify and extract the large-scale organizational patterns of structural heterogeneity within a network. When networks are small, these patterns can often be identified by eye alone.[1] But, as networks increase in size, the patterns we are looking for quickly become obscured and we must instead rely on quantitative tools to pick them out.

[1] That is, by visually inspecting a 2-dimensional embedding of the network. Nearly all network visualization algorithms are force-directed graph drawing techniques, the most common of which is the Fruchterman-Reingold algorithm. For a good introduction to these techniques, see the MathWorld page `http://bit.ly/b1mo3y`.

In addition, the larger the network, the richer the variety of large-scale "mixing" patterns a network can contain, and the more complicated the relationships between these patterns. The notion of *mixing* is a key concept in understanding large-scale structure, and represents the idea that vertices may mix differently with some types of vertices than with others.

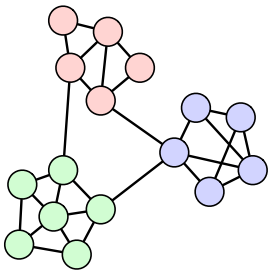


homogeneous heterogeneous

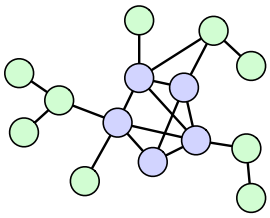
Mixing can either be *homogeneous*, meaning that every local region of the network exhibit similar statistical patterns (left-hand network above), or *heterogeneous*, meaning that local regions differ in their statistical patterns (right-hand network above). Simple random-graph models, such as the Erdős-Rényí model, or a k-regular random graph, or a simple random spatial network, are all homogeneous. Arguably, even random graphs with heavy-tailed degree distributions, *a la* the configuration model, are relatively homogeneous, as the only structural heterogeneities are driven by the skewed degree distribution. Indeed, their homogeneity is in part why simple random graphs can be poor models of real-world systems, where heterogeneity is practically a rule.

There are two classic types of heterogeneous mixing: *assortative* mixing ("like links with like") and *disassortative* mixing (like links with dislike). Vertices may mix on any feature, including observed vertex attributes, like age, sex, or geography in social networks, or unobserved attributes.

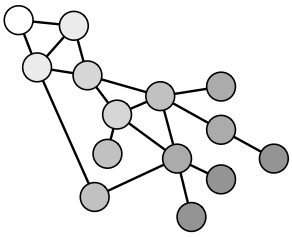
For assortative mixing patterns, a useful analogy is the common problem of clustering in vector data, in which points within a cluster are closer or more similar to each other than they are to points in other clusters. Thus, heterogeneity at the global level could be caused by mixing or averaging across several distinct parts of the data that are themselves relatively homogeneous. In networks, this pattern is often called *modular* or *community structure*, in which the "communities" are the homogeneous building blocks of the larger, heterogeneous structure. In biological networks, such structures may be functional clusters of genes or proteins, and they might represent targets of natural selection above the level of individual genes but below the level of the whole organism. In social networks, they could be human social groups with common interests, histories or behaviors.



modular*

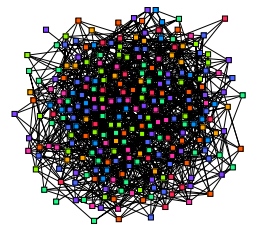


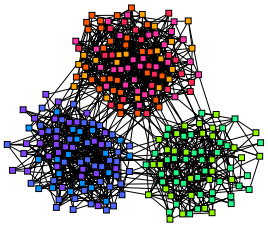
core-periphery

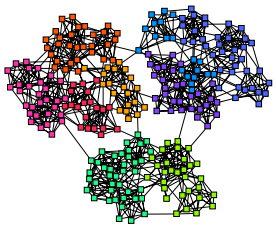


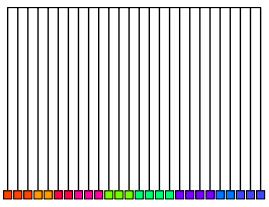
ordered

Community structure alone implies only a single level of organization, with relatively little structure within or between communities. But why stop at one level of organization? The natural generalization of community structure is called *hierarchical* community structure, in which vertices divide into groups that further subdivide into groups of groups, and so forth over multiple scales. This structure is often described using a *dendrogram* or tree that shows how the smallest groups are nested within larger groups, and they in turn are nested within still larger groups, etc. A common assumption for this kind of large-scale organization is that two vertices who are "closely related", that is, are separated by a short distance on the tree, are more likely to be connected. This is typical, for instance, of human social structures, for instance: our class sits within the larger Computer Science department, which itself sits inside the larger College of Engineering, which sits inside CU, which sits inside Colorado schools, etc.

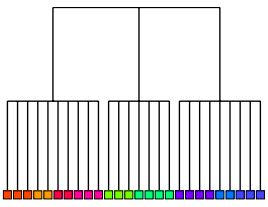




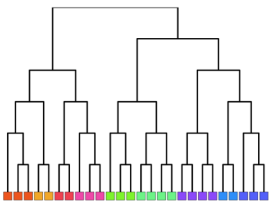




homogeneous

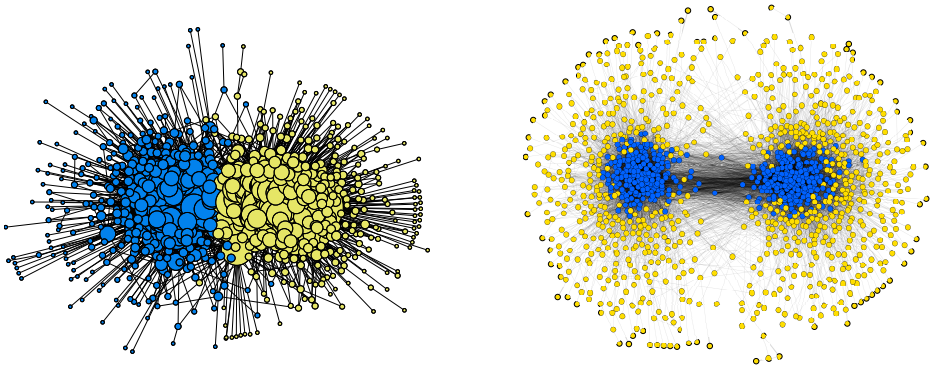


modular



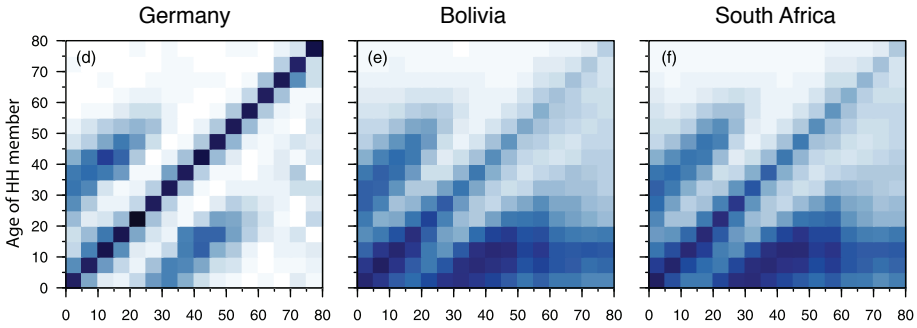
hierarchical

Another kind of large-scale structure is a *core-periphery* pattern, in which a dense core is surrounded by a more diffuse, weakly interconnected periphery. This notion has much in common with traditional measures of centrality, which can be viewed as proxy measurements for identifying "core" vertices. For instance, the figure below shows a modular division of the classic political blogs network on the left, and the same network on the right but with core and peripheral nodes within each module highlighted.[2]



modular core-periphery within modules

Similarly, an *ordered* pattern appears when nodes arrange themselves roughly into a linear hierarchy, with vertices at position k tending to connect only to those at $k \pm \Delta$ in the ordering. For instance, social interactions among people are ordered with respect to people's ages.[3]



[2]These figures reproduced from B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks." *Phys. Rev. E* **83**, 016107 (2011) and from X. Zhang, T. Martin, and M. E. J. Newman, "Identification of core-periphery structure in networks." *Phys. Rev. E* **91**, 032803 (2015).

[3]Figure reproduced from K. Prem, A. R. Cook and M. Jit, "Projecting social contact matrices in 152 countries using contact surveys and demographic data." *PLoS Comp. Bio.*, DOI: `10.1371/journal.pcbi.1005697` (2017).

2 Homophily and assortative mixing

Networks, and particularly social networks, often exhibit a property called *homophily* or *assortative mixing*, which simply means that the attributes of vertices correlate across edges. That is, suppose vertices i and j have attributes x_i and x_j. If we observe an edge (i, j), then the attributes x_i and x_j will tend to be more similar than if there were no such edge. Put succinctly, like links with like.[4]

In social networks, people have a very strong tendency to associate with others who are similar to them, e.g., in age, nationality, language, socioeconomic status, educational level, political beliefs, and many others. In fact, homophily is so strong in social networks that nearly every characteristic we can think of exhibits this assortative mixing pattern. However, the fact of homophily is only a statement of pattern and does not identify the underlying mechanism. If we observe a pattern of homophily in a social network, e.g., on political beliefs or obesity, we generally cannot distinguish between (i) the edge forming as a result of the attributes being similar, or (ii) the attributes becoming more similar as a result of the edge.

Less commonly, we may observe the reverse pattern, called *disassortative mixing*, which represents a pattern of association between vertices with dissimilar attributes. That is, if the edge (i, j) exists, then x_i and x_j tend to be less similar than a randomly chosen unconnected pair. For example, both dating and sexual contact networks are largely disassortative, with the large majority of edges running between men and women, with a smaller number of edges running among men or among women. Similarly, in food webs, predators tend to be connected to their prey, rather than to each other, and in economic networks, producers of widgets tend to connect to consumers of those widgets, rather than to each other.

There are two basic ways a network can be assortative, distinguished by where the attribute x is a label or enumerative value (i.e., an unordered type like vertex color or shape) or a scalar value. We will cover them both, and then consider one type of scalar mixing that is of particular interest.

2.1 Enumerative attributes

Enumerative attributes are those that lack any particular ordering, and are sometimes also called categorical variables. They represent things like vertex ethnicity, gender, vocational role, source genome, etc. For simplicity of notation, let $x_i \in \mathcal{S}$ denotes an enumerative vertex attribute and let the number of possible values be $|\mathcal{S}| = k$.

The typical measure of this form of assortativity, i.e., the degree to which like things are connected, is called the *modularity* Q. When we are given the attribute x value for each vertex, calculating the modularity answers this question:

[4] Or, birds of a feather link together.

How much more often do attributes match across edges than expected at random?

That is, in order to determine whether some attribute mixes assortativity, i.e., occurs at either end of an edge surprisingly often, we need to write down a null model for mixing at random. Given this model, we can then compute the total difference Q between the observed and expected fractions of edges for the $\mathcal{S} \times \mathcal{S}$ pairs of types. If $Q > 0$, then we conclude for assortative mixing and begin thinking about why this system should mix in this way.

There are two cases where we should expect $Q = 0$. The first is the trivial case where all vertices of are the same type. Here, there is only one type of edge and thus the observed fraction of edges must equal its expected value. The second is when attributes match no more or less frequently than we would expect at random. (Under what circumstances would this measure yield its maximum value of 1? Note that its minimum value is not zero, but is in fact -1. Under what circumstances would this value be achieved?)

The key question for writing down an expression for modularity Q is choosing a null model. For instance, consider a network in which a fraction q of all vertices are blue while the remaining are red. Under a simple random graph model where edges occur independently with the same probability, i.e., the Erdős-Rényi model, the expected number of edges that connect two blue vertices is $m \times q^2$ and the expected number for two red vertices is $m \times (1-q)^2$. If the empirical counts for these quantities is greater than their expected values, then we could conclude the presence of assortativity.

However, we know that simple random graphs are terrible models of real networks because they produce thin-tailed degree distributions. If some vertices have very high degrees, then their color label participates in more connected pairs, and this will skew our statistics if our null model cannot produce such high-degree vertices. Worse, if red vertices tend to be high-degree vertices, we might incorrectly rule against assortativity. The solution is to use the configuration model as the null model, rather than a simple random graph.
Given a labeling of vertices $\mathbf{x}$, the modularity for an undirected network is given by

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(x_i, x_j) \ , \tag{1}$$

where A is the adjacency matrix, k_i is the degree of vertex i, m is total the number of edges in the network, x_i is the label of vertex i, and $\delta(x_i, x_j)$ is the Kronecker delta function, which equals 1 when its arguments are the same and 0 otherwise.

Let us dissect this equation, starting with the summation. The sum is over all pairs of vertices i, j, regardless of whether there is an edge there or not. The delta function serves as a kind of filter, selecting only those pairs i, j whose labels are the same $x_i = x_j$. Thus, the summation is effectively only over pairs of vertices with the same type label. The inner term is the difference between the

observed fraction of all edges between i and j, which is $A_{ij}/2m$, and its expected value under a random graph with the same degree distribution. If the degrees of vertices i and j are k_i and k_j, then the probability that i and j are connected, if edges are distributed at random conditioned on respecting these degrees, is the number of chances they have to connect $k_i k_j$ divided by the total number of such pairs, which is $(2m)^2$. Because both terms have a common denominator of $1/2m$, we may factor this out, which gives the leading constant.

Notice that the summation is only over edges among vertices of the same type. This allows us to rewrite and simplify the equation by dropping the many zeros in the total summation. To accomplish this, we define two auxiliary quantities

$$e_{uv} = \frac{1}{2m} \sum_{ij} A_{ij}\, \delta(x_i, u)\delta(x_j, v) \ , \tag{2}$$

which is the fraction of edges that join vertices with label u to vertices with label v, and

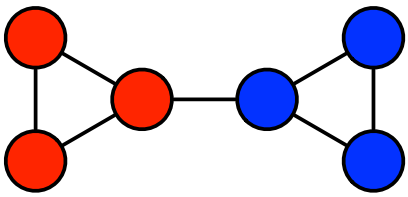
$$a_u = \frac{1}{2m} \sum_{j} k_j\, \delta(x_j, u) \ , \tag{3}$$

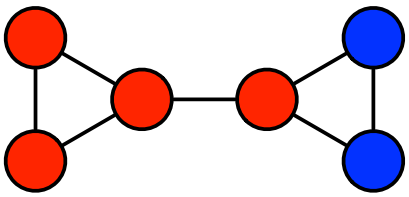
which is the fraction of ends of edges that are attached to vertices with label u. (Exercise: show that $a_u = \sum_v e_{uv}$, i.e., show that a_u is the column (or row) sum of the e_{uv} matrix.) The modularity is then

$$\begin{aligned} Q &= \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \sum_u \delta(x_i, u)\delta(x_j, u) \\ &= \sum_u \left[\frac{1}{2m} \sum_{ij} A_{ij}\, \delta(x_i, u)\delta(x_j, u) - \frac{1}{2m} \sum_i k_i\, \delta(x_i, u) \frac{1}{2m} \sum_j k_j\, \delta(x_j, u) \right] \\ &= \sum_u \left(e_{uu} - a_u^2 \right) \ , \end{aligned} \tag{4}$$

where e_{uu} is the observed density of edges among vertices with label u and a_u^2 is the expected density under the configuration model. This form of Eq. (1) is more compact and may be used even when we do not have the full adjacency matrix. That is, to compute Q using Eq. (4), we only need a matrix containing the number of connections between each pair of vertex types. Note also that both equations are defined only for undirected graphs. Directed or weighted versions are easily defined, although less commonly used.

To illustrate a modularity calculation, consider two distinct labelings on the small modular network with $n = 6$ vertices and $m = 7$ edges shown below. First, because the number of types $k = 2$, we construct a 2×2 matrix based on each labeling, where each matrix element counts the fraction of edges of a given pair type:





labeling 1	red	blue
red	3/7	1/14
blue	1/14	3/7

labeling 2	red	blue
red	4/7	2/14
blue	2/14	1/7

From these matrices, the modularity is straightforward to calculate, yielding $Q_1 = 5/14 = 0.357$ for the first, and $Q = 6/49 = 0.122$ for the second. In this case, labeling 1 yields a higher modularity than labeling 2 because it has a larger fraction of within-type edges, i.e., it displays more assortative mixing.

2.2 Scalar attributes

When vertex attributes are scalar values, like age, weight, or income, we may say not only when two vertices have the same value, as in the previous case, but also when two vertices are close in value. We again let x_i denote the vertex attribute, but now let x vary as an integer or real value.

The typical measure for this form of assortativity is called the *assortativity coefficient*, which is a kind of network-based generalization of the Pearson correlation coefficient. This measure answers the following question:

How much more similar are attributes across edges than expected at random?

That is, we again invoke a null model in order to decide whether vertices tend to link preferentially with those having similar attributes. Now, however, we are interested in attribute similarity, rather than simple matching.

To write down the expression for the assortativity coefficient, we adapt the standard expression of covariance to a network context. The mean value observed at either end of an edge is simply $\mu = (1/2m)\sum_i k_i x_i$, which weights each observed scalar value by the number of edges in which it participates.

The covariance of x across edges is then

$$\begin{aligned}\mathrm{cov}(\vec{x}, A) &= \frac{\sum_{ij} A_{ij}(x_i - \mu)(x_j - \mu)}{\sum_{ij} A_{ij}} \\ &= \frac{1}{2m}\sum_{ij} A_{ij}x_i x_j - \mu^2 \\ &= \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)x_i x_j \ , \end{aligned} \tag{5}$$

where we have reused the definition of μ, and simplified considerably, to get to the last line.

Note that this form of the covariance is remarkably similar to the definition of the modularity given in Eq. (1), except that instead of a delta function selecting only edges for which the attributes match, we now weight every edge by the product $x_i x_j$. When the values of the last term tend to agree, i.e., small values are multiplied by small values and large values multiplied by large values, the covariance will be positive, indicating assortative mixing, while when the opposite is true, the covariance is negative, indicating disassortative mixing.

Finally, just as in the case of the modularity, and in the case of the traditional definition of the Pearson correlation coefficient, it is useful to normalize the covariance so that it ranges from -1 to 1. This version has the form

$$r = \frac{\sum_{ij}\left(A_{ij} - k_i k_j /2m\right)x_i x_j}{\sum_{ij}\left(k_i\,\delta(i,j) - k_i k_j /2m\right)x_i x_j} \ , \tag{6}$$

where $\delta(i,j) = 1$ if $i = j$ and 0 otherwise. We call r the assortativity coefficient, and is exactly the covariance divided by the variance, i.e., it is a correlation measure.

2.3 Vertex degrees

Vertex degree is a scalar attribute that is of particular interest as it reveals important insights about the large-scale structure of the network and can be calculated from the network structure alone. That is, we set $x_i = k_i$, which slightly simplifies Eq. (6).

Assortative mixing by degree produces a network in which the high-degree vertices tend to connect to each other in dense, high-degree core, while the low-degree vertices also connect to each other, producing a sparse, low-degree periphery. In these networks, degree correlates with centrality. By contrast, disassortative mixing by degree produces a network in which the high-degree vertices tend to connect to low-degree vertices, producing star-like structures. In these networks, centrality correlates less strongly with degree. (See Figure 7.12 in *Networks* for a good visualization of this distinction.)

2.4 Homophily and social networks

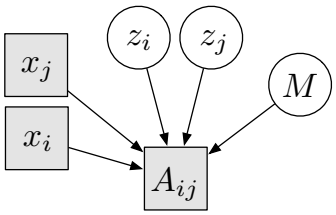
Within social networks, homophily is a nearly overpowering phenomenon. It is sufficiently strong that merely knowing the attribute labels or values for a subset of a vertex's neighbors can allow a researcher (or marketer) to make a fairly good guess about the label or value at the vertex. That is, knowing your friends' values tells me a great deal about your value, even if you have not disclosed it to me. That being said, homophily is merely a correlation, and thus the values of your friends' friends is only moderately predictive of your value.

And yet, quantifying the extent to which some attributes are homophilous can provide good insights into the overall heterogeneous structure of the network, and provide clues about the underlying organizing principles. In social networks, there are specific mechanisms by which homophily is expected to increase over time, i.e., friends can tend to become more similar to each other over time or can choose new friends based on similarity. In other networks, however, such mechanisms are less clear. For example, when assortativity is observed in gene networks, what is the mechanism by which similarity should appear or should increase over time?

3 Community structure and community detection

In many networks, vertex attributes are either completely unobserved (as in a simple, undecorated graph), partially unobserved (as in most social networks), or the observed attributes are only weakly related to the network structure itself. In these situations, we would still like to make quantitative statements about the heterogeneous mixing patterns of vertices and the network's large-scale structure. Here, we develop simple tools for doing so and sketch out a path to more powerful tools.[5]

To formalize the idea of dealing with structure caused by unobserved variables, consider the following graphical model:[6]



[5] A separate but related set of ideas focus on the impact that these large-scale patterns have on network dynamics, especially the propagation of information or diseases across networks. We will not cover that subject here, but will focus instead on the question of understanding the structure itself.

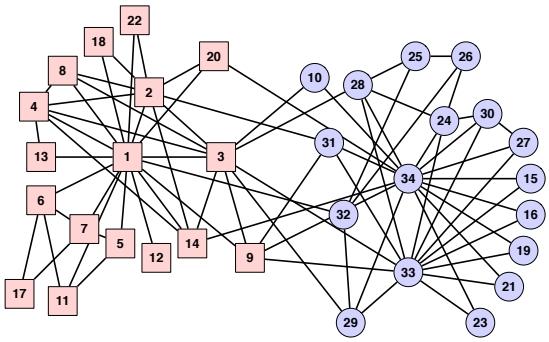
[6] A graphical model is a compact representation of the relationships among random variables. Circles denote unobserved or latent variables that must be estimated. Boxes represent observed variables. And an arrow indicates a conditional relationship with the variable at the arrow's head depending on the variable at the arrow's tail.

The observed variables here are the value of a connection A_{ij} (usually 0 or 1) and the attributes of the involved vertices x_i and x_j. The unobserved variables are a pair of "free" or latent vertex attributes z_i and z_j along with the mixing matrix M that describes how vertices with different attributes tend to connect. Modularity and the assortativity coefficient use the observed attributes and connectivity alone to estimate the matrix M. When the attributes x are unobserved, we instead search over the possible assignments of the free variables z to find one that produces a good matrix M.[7]

This kind of task is often called *community detection* because we want to detect the presence of communities given only the network connectivity. To do this, we assume that community membership governs connectivity and then treat the z variables as indicators of communities membership. Community detection thus answers the following question

What assignment of vertices to underlying communities best models the observed connectivity?

When vertices tend to preferentially connect with other vertices in the same community, we say the network exhibits assortative community structure (like links with like). When vertices connect preferentially with vertices in different communities, we call it disassortative community structure. The karate club is a classic example of assortative community structure.



3.1 Partitions and communities

A common approach to community detection solves the following problem. First, choose some *score function* f that takes both a network A and a *partition* P of its vertices (defined below) as in put, and returns a scalar value. Now, find the partition P that maximizes f. The idea is that f encodes

[7]The observed attributes x and the unobserved variables z are not necessarily equivalent. Vertex attributes, especially in social networks, are often highly correlated, and we usually think of the z variables as being some underlying or fundamental variable that structures the network. Thus, the z variables could be only partially or even completely unrelated to the x variables.

our beliefs about what makes a *good* partition with respect to representing community structure. If f prefers assortative structure, then it should assign higher scores to partitions that place more edges within groups than between them. Maximizing f over all partitions will yield the partition with the most assortative groups possible under f.

Network partitions are simply a division of a network into k non-empty groups (communities), such that every vertex v belongs to one and only one group (community). For a given value of k, the number of possible such partitions is given by the Stirling numbers of the second kind

$$S(n,k) = \left\{ \begin{array}{c} n \\ k \end{array} \right\} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^j \binom{k}{j} (k-j)^n \ . \tag{7}$$

For instance, there are $S(4,2) = 7$ possible partitions of a network with $n = 4$ nodes (indexed as 1,2,3,4) partitioned into $k = 2$ groups:

$$\{1\}\{234\},\ \{2\}\{134\},\ \{3\}\{124\},\ \{4\}\{123\},\ \{12\}\{34\},\ \{13\}\{24\},\ \{14\}\{23\}\ .$$

The number of all possible partitions of a network with n vertices into k non-empty groups is given by the nth Bell number, defined as $B_n = \sum_{k=1}^{n} S(n,k)$, which grows super-exponentially with n.[8] That is, the universe of all possible partitions of even a moderately sized network is very very big and an exhaustive maximization of f is not typically feasible.

Instead, most algorithms use some kind of heuristic[9] for estimating the maximum of f.

3.2 Detecting communities by maximizing modularity

The modularity function Q, which gives a measure of assortative structure on enumerative vertex attributes, is just such a choice of score function f. When labels on the vertices are given, we can calculate Q and thereby quantify the degree to which edges fall within those groups relative to the configuration model. Different labelings give different scores, and the higher the score, the "better" the communities. Thus, a reasonable approach to detect and identify assortative communities is to search over all possible labelings to find one that produces a high or even the highest possible modularity score.

[8] The first 20 Bell numbers (starting at $n = 0$) are 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597, 27644437, 190899322, 1382958545, 10480142147, 82864869804, 682076806159, 5832742205057.

[9] In computer science, the term "heuristic" is usually applied to any procedure that does not come with guarantees about its correctness, i.e., its ability to return the global maximum. This does not mean the heuristic is bad, only that we do not have a proof of its general correctness. The term "algorithm" is usually reserved for procedures with such guarantees. Outside of computer science, heuristics are nearly always called algorithms.

Recall from Eq. (4) that the modularity score Q can be written as a sum over the differences between the fraction of edges that join vertices in the uth module, denoted e_{uu} and the expected fraction, given by a_u^2. A "good" partition—with Q closer to unity—represents groups with many more internal connections than expected at random; in contrast a "bad" partition—with Q closer to zero—represents groups with no more internal connections that we expect at random. Thus, Q measures the cumulative deviation of the internal densities of the identified modules relative to a simple random graph model.[10]

Also recall that the number of possible partitions is for n vertices is B_n, which grows super-exponentially with n. It is generally infeasible to exhaustively search this space and it is known that maximizing Q is NP-hard. In practice, however, many search heuristics perform extremely well on real-world networks. Examples of these strategies include greedy agglomeration, mathematical programming (linear and quadratic), spectral methods, extremal optimization, simulated annealing and various kinds of sampling techniques like Markov chain Monte Carlo. In nearly every case, however, the partition returned by these heuristics is merely a high-scoring partition, rather than the best possible partition.

Many of these algorithms run quite slowly on real-world networks, taking time $O(n^2)$ or slower; a few run very quickly, for instance, taking time $O(n \log^2 n)$ and can thus be applied to very large networks of millions or more vertices. Because they are heuristics, of course, faster algorithms must make more aggressive assumptions about how to search the partition space and thus are less likely to find the globally optimum partition.

3.3 A simple algorithm: greedy agglomeration

A simple approach to finding a good partition is that of greedy agglomeration. There are several ways to design such an approach, but the idea is simple: start with every vertex being in its own group, and then recursively and greedily choose a pair of groups to merge.

A simple implementation goes like this.[11] Initially, when each vertex is in its own group, each term in Eq. (4) is simply $-a_i^2 = -(k_i/2m)^2$ (for a network without self-loops). At each pass of the algorithm, we choose the pair of groups that, if we were to merge them, would produce the

[10] It should be pointed out that a high value of Q only indicates the presence of significant deviations from the null model of a random graph. A large deviation could mean two things: the network exhibits (i) genuine modular structure in an otherwise random graph, or (ii) other non-random structural patterns that are not predicted by a random graph. See Section 5 at the end of this file for a brief discussion of this subtlety.

[11] This algorithm first appeared in Newman, "Fast algorithm for detecting community structure in networks." *Phys. Rev. E* **69**, 066133 (2004). It was subsequently improved in Clauset, Newman and Moore, "Finding community structure in very large networks." *Phys. Rev. E* **70**, 066111 (2004). A faster but different greedy agglomerative algorithm was given in Blondel, Guillaume, Lambiotte and Lefebvre, "Fast unfolding of communities in large networks." *J. Stat. Mech.*, P10008 (2008).

greatest increase (or smallest decrease) in the modularity ΔQ. We then merge that pair and repeat until there is only one group remaining. The progress of the algorithm can be represented as a dendrogram or tree, showing how groups merge together. A "cut" across the dendrogram then induces a partition of the vertices, each of which has some modularity score. The best cut is the one with the largest Q of any cut, and we return that partition as our output.

Since merging a pair of communities between which there are no edges cannot increase the modularity, we need only calculate ΔQ for pairs of connected communities. At any moment in the algorithm, there are at most m of these pairs. It can be shown that the change in modularity for merging two communities is

$$\Delta Q = e_{uv} + e_{vu} - 2a_u a_v = 2(e_{uv} - a_u a_v) \ , \tag{8}$$

where e_{uv} is the fraction of edges connect vertices in group u to vertices in group v and $a_u = \sum_v e_{uv}$ is the fraction of all ends of edges that are attached to vertices in group u. (Exercise: derive Eq. (8).)

Each time we merge a pair of groups r and s, we must update the elements in the rth row and column and delete the elements in the sth row and column (for merging s into r). This takes $O(n)$ time and computing the ΔQ for each edge in the network of groups takes $O(m)$ time. Thus, each round takes $O(n+m)$ time, and because there are exactly $n-1$ such merges to do, the total running time is $O((m+n)n) = O(n^2)$ for sparse graphs. Although this algorithm is not the fastest of the greedy agglomerative algorithms known for maximizing modularity, it is one of the most straightforward to explain and implement.

3.4 A small example

As an exercise for the reader, run the algorithm from Section 3.3 on the two-triangle network used earlier in these notes. Does it find the labeling we expect it to?

3.5 A large example: Amazon co-purchasing network

As another example, we consider applying the greedy agglomeration algorithm to a co-purchasing or "recommender" network from `amazon.com`. Amazon sells a wide variety of products, particularly books and music, and on each item A's webpage, Amazon lists several other items most frequently purchased by buyers of A. Figure 3.5 shows an example of this information.

These recommendations form a directed network in which vertices are items and an item A points to an item B if B was frequently purchased by buyers of A. In our analysis, we convert this directed network into an undirected one. The data we use are from a 2003 snapshot of Amazon's recommendation network, in which we focus on the largest component, which contains $n = 409,687$ vertices

and $m = 2,464,630$ edges.

Applying the greedy agglomeration algorithm to this network produces some interesting results. The resulting dendrogram is far too large to draw (since it contains nearly half a million joins), but Fig. 1 (left) shows the modularity score Q over the sequence of merges made by the algorithm. The maximum value is $Q = 0.745$, which is fairly large as these kinds of calculations go[12] This value occurs when there are 1684 communities remaining, each of which has an average size of 243 items. Fig. 1 (right) gives a visualization of the community structure at this point, where each vertex represents a group of items. There is plenty of interesting structure to be seen here, including the major communities, smaller "satellite" communities connected to them, and "bridge" communities that connect two major communities with each other.

If we examine the contents of the largest communities in the network, we find that they tend to con-

[12]Empirically, the maximum modularity score $Q_{\max}$ appears to correlate with n.

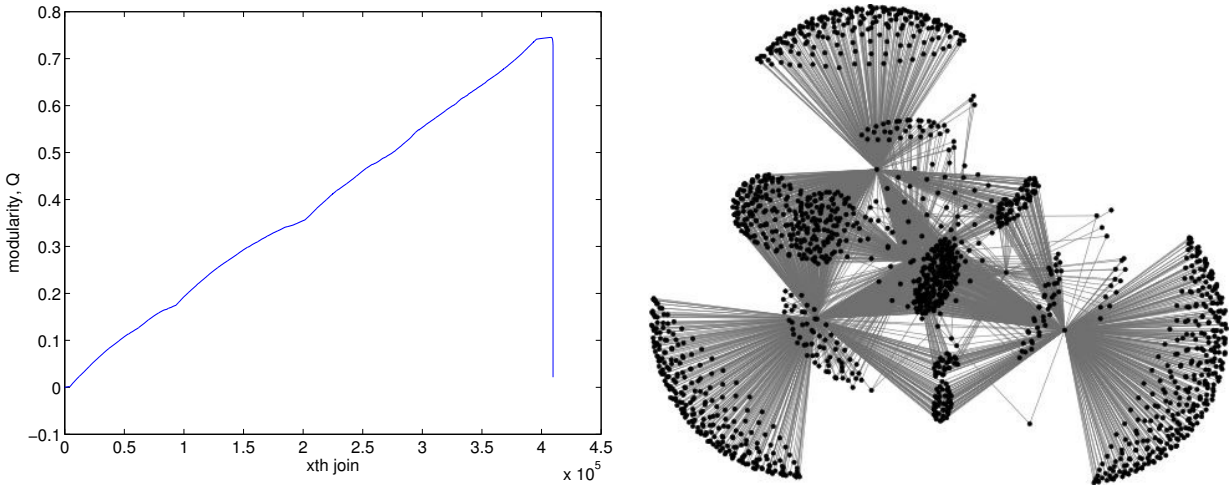


Figure 1: (left) The modularity score Q over the course of the agglomerative algorithm, with a maximum at $Q = 0.745$ for 1684 communities. (right) A visualization of the community structure at maximum modularity, where each vertex is a "super" node containing many items.

Rank	Size	Description
1	114538	General interest: politics; art/literature; general fiction; human nature; technical books; how things, people, computers, societies work, etc.
2	92276	The arts: videos, books, DVDs about the creative and performing arts
3	78661	Hobbies and interests I: self-help; self-education; popular science fiction, popular fantasy; leisure; etc.
4	54582	Hobbies and interests II: adventure books; video games/comics; some sports; some humor; some classic fiction; some western religious material; etc.
5	9872	classical music and related items
6	1904	children's videos, movies, music and books
7	1493	church/religious music; African-descent cultural books; homoerotic imagery
8	1101	pop horror; mystery/adventure fiction
9	1083	jazz; orchestral music; easy listening
10	947	engineering; practical fashion

Table 1: The 10 largest communities in the `amazon.com` co-purchasing network.

sist of items (books, music) in similar genres or on similar topics. Table 1 gives informal descriptions of the ten largest communities, which account for about 87% of the entire network. The remainder are generally small, densely connected communities that represent highly specific co-purchasing habits, e.g., major works of science fiction (162 items), music by John Cougar Mellencamp (17

items), and books about (mostly female) spies in the American Civil War (13 items).

3.6 A myriad of approaches

There are many other approaches to identifying communities in networks. Some are different ways of searching over partitions, but which use the modularity function to evaluate them. Otherwise use different score functions. We will not cover any of these in detail, except for the stochastic block model, which we will cover next. There are at least two qualitatively different approaches to clustering networks, which are worth mentioning.

The first is the *local* clustering algorithms. Note that the two approaches described above both require us to explicitly know the entire network structure. Thus, they require access to *global* information, e.g., the total number of edges in the network. Local techniques make no such assumptions and can be applied to networks like the WWW, which are too big to be fully known. Local methods often "crawl" outward from a particular starting point to identify the first "community" that encloses the starting vertex by looking for a boundary of vertices with relatively sparse connectivity to the rest of the network.

The second is the "soft" clustering algorithms for identifying *overlapping* communities, i.e., rather than sorting vertices such that they appear in one and only one cluster ("hard" clustering), vertices are allowed to belong to multiple communities, potentially with different strengths of association.

4 At home

1. Assortativity: Read Chapter 7.9–7.13 (pages 198–231) in *Networks*
2. Community Structure: Read Chapter 11.2–11.4 & 11.6–11.11 (pages 354–364 & 371–391) in *Networks*

5 Trouble with community detection

Finally, there are many caveats about community detection. Here are three.

The first ill omen is the observation that despite a tremendous amount of activity on identifying clusters in networks, for the most part, we have no good explanations of what social, biological or technological processes should cause clusters to exist in the first place, or what functional role they play in those systems. (Presumably, these two things are related.) The best we have is a verbal argument due to Herbert Simon for why modular designs a good way to construct complex systems. The argument goes like this:

Suppose we are tasked with assembling many small, delicate mechanical parts into a beautiful Swiss watch, a masterpiece of complexity. And, suppose that we are interrupted every now and then, perhaps by the arrival of email or the need to have a snack, and than if we are interrupted before we have fully completed the assembly of a single watch, the partially assembled watch falls to pieces and we must start afresh after our break is over. If, on average, we experience one or more interruptions during the long process of assembling a watch, we will never produce very many watches. Now, however, consider the advantage of a modular design to the watch. In this case, the final watch is produced by combining smaller groups of parts or modules. If these modules are themselves stable products, then now we can tolerate interruptions and still produce watches because at worst, we only lose the work necessary to build a piece of the whole, rather than the whole itself. Thus, it seems entirely reasonable to expect that, in the face of "interruptions" of all different kinds, complex systems of all kinds will exhibit a modular or even hierarchical organization.

A second ill omen comes from recent observations that many of the popular global techniques for identifying modules in networks exhibit two undesirable properties: (i) *resolution limits* and (ii) *extreme degeneracies*. The first observes that the technique cannot detect intuitively coherent communities that are smaller than some preferred scale. In general, the mathematics showing the appearance of this behavior always points to the same root cause, which is the random-graph assumptions built into the techniques' score functions. The second observes that there are an exponential number of alternative, high-scoring partitions, which makes both finding the best partition computationally difficult (in some cases, NP-hard) and interpreting the particular structure of the best partition ambiguous. These two properties make it somewhat problematic apply network clustering techniques in contexts where what we want is to find the *true* but unknown modules. For much more detail about these issues, see Good et al., "The performance of modularity maximization in practical contexts." *Physical Review E* **81**, 046106 (2010).

A third ill omen comes from the world of spatial clustering, a very old and still very active field. Many of the techniques for finding clusters in networks have antecedents in this domain. The general problem can be framed like so: we are handed a data set $\mathbf{x} \in \mathbb{R}^{\mathbf{n}}$ and our task is to identify the "natural" clusters of these data such that points in a cluster are closer (via some measure of distance) to each other than they are to all the other points.

However, in a 2002 paper titled "An Impossibility Theorem for Clustering", Jon Kleinberg showed, using an axiomatic approach, that no clustering function f can simultaneously satisfy three highly reasonably and practically banal criteria:

1. *Scale Invariance*: For any distance function d and any $\alpha > 0$, $f(d) = f(\alpha \cdot d)$. Intuitively, this requirement says that if we rescale all the distances between points by some constant factor α, the best clustering should stay the same.
2. *Richness*: Range(f) is equal to the set of all partitions of $\mathbf{x}$, which simply requires that all

possible partitions of our data be potentially okay candidates.

3. *Consistency*: Let d and d' be two distance functions. If $f(d) = \Gamma$, and d' is a Γ-transformation of d, then $f(d') = \Gamma$. In other words, if a particular distance function d yields the clustering Γ, then if we make a new distance function d' that reduces all of the distances arising within clusters while increasing all those distances between clusters—that is, we make the clusters internally more dense and move them further away from each other—then we should get the same clustering Γ under d'.

Thus, any practical solution to identifying clusters in spatial data can, at best, only have two of these properties. One consequence of this fact is that we can categorize clustering techniques into broad categories, based on which single criterion they violate. It's unknown if a similar impossibility theorem exists for network clustering.[13]

[13] It is not hard to see that the second and third criteria have natural analogs in the context of network clustering. The first criterion, however, has no clear analog.