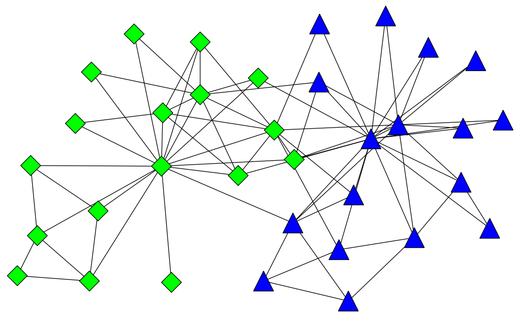
1 The configuration model

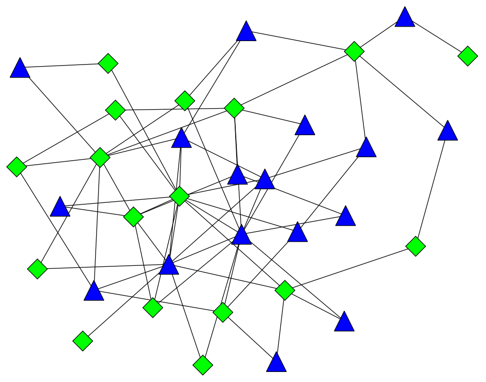
A particularly unrealistic aspect of the random graph model $G(n,p)$ is its degree distribution, which we showed follows a Poisson distribution when the graph is sparse. In contrast, most real-world graphs exhibit heavy-tailed degree distributions. We can improve this aspect of our random graph model by using a generalization called the "configuration model." Although there is no standard abbreviation by which we refer to the configuration model, reusing the notation from the Erdős-Rényi model, we might say $G(n,\vec{k})$ where $\vec{k}=\{k_i\}$ is a *degree sequence* and k_i is the degree of vertex i.

The degree sequence $\vec{k}$ can be any sequence so long as $\sum k_i$ is an event integer (do you see why?). This means we may, if we like, choose $\vec{k}$ to be a sequence of values drawn iid from some degree distribution $\Pr(k)$. When $k \sim \text{Poisson}(c/(n-1))$, the configuration model reduces to something very similar to the Erdős-Rényi random graph. When k is drawn from some other distribution, we get different networks. When working with empirical data, a particularly common choice of $\vec{k}$ is the particular degree sequence observed in the empirical network.

Given a degree sequence, we may construct a random graph by choosing a uniformly random matching on the degree "stubs" (half edges). The result is an instance of the ensemble corresponding to $G(n,\vec{k})$. Unlike the random graph model, the configuration model does not construct simple networks. Instead, self-loops and multi-edges are allowed. But, these represent a tiny fraction of all edges, and we typically lose little by just discarding or collapsing them.

As an example of what the configuration model produces, here are visualizations of our old friend the karate club and a random graph with the same degree sequence (via the configuration model). Note that the original group structure has been randomized.

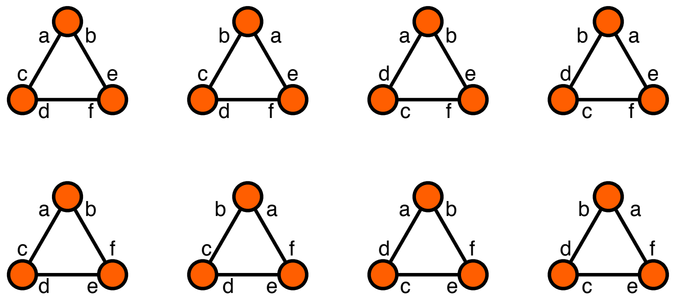




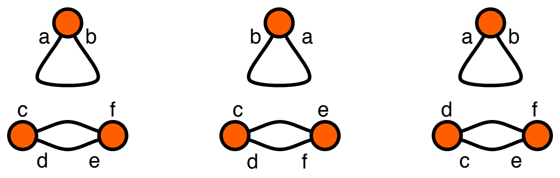
1.1 Generating networks using the configuration model

Given a degree sequence $\vec{k} = \{k_1, k_2, \ldots, k_n\}$, we can generate a random graph with exactly that set of degrees, but which is otherwise random, by taking a uniformly random matching on the "stubs" attached to vertices. Each such matching thus occurs with equal probability. This does not, however, imply that each network will occur with equal probability, as some matchings produce the same network.

Consider the set of matchings on six edges that form a triangle. The following figure illustrates the fact that distinct labelings of the edge stubs, and hence distinct matches, form exactly the same network. In the configuration model, we choose each of these with equal probability.



However, these are not the only possible matchings on these six edge stubs. The following figure shows three other distinct matchings, which produce non-simple networks, i.e., networks with self-loops and/or multi-edges.

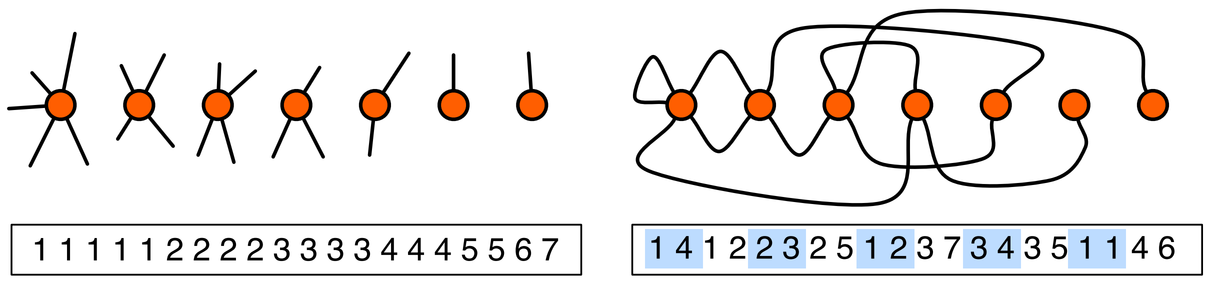


In practice, the fraction of edges involved in either self-loops or multi-edges is vanishingly small in the large-n limit, and thus we may generally ignore them without much impact. (However, there are applications in which these features are important, and so it is worth remembering that they exist.)

Once a degree sequence $\vec{k}$ has been chosen, e.g., by taking the degree sequence in some empirical network or by drawing a sequence from a degree distribution, to draw a network from the corresponding configuration model, we simply need an efficient method by which to choose a uniformly random matching on the $\sum_i k_i$ stubs.

Let v be an array of length $2m$ and let us write the index of each vertex i exactly k_i times in the vector v. Each of these entries will represent a single edge stub attached to vertex i. Then, we take a random permutation of the entries of v and read the contents of the array in order, in pairs. For each pair that we read, we add the corresponding edge to the network. (Once we have taking a random permutation of the stubs, we could choose the pairs in any other way, but reading them in order allows us to write down the full network with only a single pass through the array.)

For instance, the following figure shows an example of this "stub matching" construction of a configuration model random graph. On the left is shown both the vertices with their stubs, which shows the degree sequence graphically, and the initial contents of the array v. On the right is shown the wired up network defined by the in-order sequence of pairs given in the array, which has been replaced with a random permutation of v. In this case, the random permutation produces both one self-loop and one multi-edge.



Standard mathematical libraries often provide the functionality to select a uniformly random permutation on the contents of v. However, it is straightforward to do it manually, as well: to each entry v_i, associate a uniformly random variable $r_i \sim U(0, 1)$ (which most good pseudorandom number generators will produce). Sorting the r_i values produces a random permutation[1] on the associated v_i values, which can be done using QuickSort in time $O(m \log m)$.

[1]Each of these permutations occurs with probability equal to $1/n!$, and because there are $n!$ such permutations, we are choosing uniformly from among them. If we choose the r_i values uniformly at random, then the probability that any particular element v_i has the smallest r_i is $1/n$. Similarly, the probability that v_i has the ith smallest value is $1/(n - i + 1)$. By induction, the probability of choosing any particular ordering is $\prod_{i=1}^{n}(n - i + 1)^{-1} = 1/n!$. It is possible to choose a random permutation in $O(m)$ time using an in-place randomizer. Instead of sorting the uniform deviates, we instead loop from $i = 1$ to n within v and swap v_i with a uniformly randomly chosen element v_j where $i \leq j \leq n$. The proof that this produces a random permutation follows the proof above.

1.2 Mathematical properties

The precise mathematical properties for a configuration model random graph depend on the degree sequence. In the version of the model where we draw the degree sequence from some distribution, we may often calculate properties of the configuration model ensemble analytically (often using powerful techniques called generating functions).[2] Common choices of random graph models include the k-regular random graph, where every degree is exactly k, and the power-law random graph, where $\Pr(k) \propto x^{-\alpha}$.

The central mathematical property of the configuration model (and indeed, all random-graph models) is the probability that two vertices i, j are connected. Let k_i, k_j denote the degrees of two particular vertices in the network, and for convenience assume that both are positive integers (if either is zero, then the probability of the edge is zero). Under a random matching on the edge stubs, for a particular stub attached to vertex i, there are k_j possible stubs, out of $2m-1$ (excluding the stub on i under consideration), attached to j to which it could connect. And, there are k_i chances that this could happen. Thus, the probability that i and j are connected is

$$\begin{aligned} p_{ij} &= \frac{k_i k_j}{2m-1} \\ &\simeq \frac{k_i k_j}{2m} \ , \end{aligned} \tag{1}$$

where the second form holds in the limit of large m. Notice that this immediately implies that the higher the degrees are of i and j, the greater the probability that they connect under the configuration model.

Expected number of multi-edges.
Eq. (1) gives the probability that one edge appears between i and j. A closely related quantity is the probability that a second edge appears between i and j, and this quantity allows us to calculate the expected number of multi-edges in the entire network. The construction is very similar to that above, except that we must update our counts of stubs to account for the existence of the first edge between i and j.

The probability that a second edge appears is $(k_i-1)(k_j-1)/2m$, because we have used one stub from each of i and j to form the first edge. Thus, the probability of both a first and a second edge appearing is $k_i k_j (k_i-1)(k_j-1)/(2m)^2$. Summing this expression over all distinct pairs gives us

[2]For a good introduction to this technique, see Wilf *generatingfunctionology*, AK Peters (2006).

the expected number of multi-edges in the entire network:

$$\begin{aligned}
\sum_{\text{distinct } i,j} \left(\frac{k_i k_j}{2m}\right)\left(\frac{(k_i-1)(k_j-1)}{2m}\right) &= \frac{1}{2}\frac{1}{(2m)^2}\left(\sum_{i=1}^{n} k_i(k_i-1)\sum_{j=1}^{n} k_j(k_j-1)\right) \\
&= \frac{1}{2\langle k\rangle^2 n^2}\left(\sum_i k_i^2 - k_i\right)\left(\sum_j k_j^2 - k_j\right) \\
&= \frac{1}{2\langle k\rangle^2}\left(\frac{1}{n}\sum_i k_i^2 - \frac{1}{n}\sum_i k_i\right)\left(\frac{1}{n}\sum_j k_j^2 - \frac{1}{n}\sum_j k_j\right) \\
&= \frac{\big(\langle k^2\rangle - \langle k\rangle\big)^2}{2\langle k\rangle^2} \\
&= \frac{1}{2}\left[\frac{\langle k^2\rangle - \langle k\rangle}{\langle k\rangle}\right]^2 \quad . \qquad (2)
\end{aligned}$$

In this derivation, we used several identities: $2m = \langle k\rangle n$, which relates the number of edge stubs to the mean degree and number of vertices, and $\langle k^m\rangle = \frac{1}{n}\sum_i k_i^m$, which is the mth (uncentered) moment of the degree sequence.

The result, Eq. (2), is a compact expression that depends only on the first and second moments of the degree sequence, and not on the size of the network. Thus, the expected number of multi-edges is a constant[3] implying a vanishingly small $O(1/n)$ fraction of all edges in the large-n limit.

Expected number of self-loops.
This argument works almost the same for self-loops, except that the number of pairs of possible connections is $\binom{k_i}{2}$ instead of $k_i k_j$. Thus, the probability of a self-loop is $p_{ii} = k_i(k_i-1)/4m$, and the expected number of self-loops is

$$\frac{\langle k^2\rangle - \langle k\rangle}{2\langle k\rangle} \ ,$$

which is a constant depending only on the first and second moments of the degree sequence. Thus, just as with multi-edges, self-plops are a vanishingly small $O(1/n)$ fraction of all edges in the large-n limit when $\langle k^2\rangle$ is finite.

[3]So long as the first and second moments of the distribution producing $\vec{k}$ are finite, which is not the case if the degree distribution follows a power law with $\alpha < 3$. We are also ignoring the fact that we treated the $i = j$ case, i.e., self-loops, identically to the $i \neq j$ case, but this difference is small, as the next section shows.

2 At home

1. Read Chapter 13.0–13.2 (pages 428–445) in *Networks*
2. Next time: more configuration model